HAL
open science

# An Efficient Computer-Aided Design Methodology for FPGA&ASIC High-Level Power Estimation Based on Machine Learning

Yehya Nasser

## To cite this version:

Yehya Nasser. An Efficient Computer-Aided Design Methodology for FPGA&ASIC High-Level Power Estimation Based on Machine Learning. Electronics. INSA de Rennes, 2019. English. NNT : 2019ISAR0014 . tel-02516046

**HAL Id: tel-02516046**

**https://theses.hal.science/tel-02516046v1**

Submitted on 23 Mar 2020

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# THESE DE DOCTORAT DE

L'INSTITUT NATIONAL DES SCIENCES APPLIQUEES RENNES
COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 601
*Mathématiques et Sciences et Technologies de l'Information et de la Communication*
Spécialité : *Électronique*

Par


## Yehya Abed El-Fattah NASSER


# An Efficient Computer-Aided Design Methodology for FPGA&ASIC High-Level Power Estimation Based on Machine Learning


**Thèse présentée et soutenue à INSA Rennes, le 25 Novembre 2019**
**Unité de recherche : IETR**
**Thèse N° : 19ISAR 23 / D19 - 23**


**Rapporteurs avant soutenance** :

**Muhammad Shafique** Professeur, Vienna University of Technology, Austria

**Dimitrios Soudris** Professeur, National Technical University of Athens, Greece

**Composition du Jury :**

Président : **Pascal Chevalier** Professeur, CNAM Paris, France

Examinateurs :

**Dimitrios Soudris** Professeur, National Technical University of Athens, Greece
**Muhammad Shafique** Professeur, Vienna University of Technology, Austria
**Pascal Chevalier** Professeur, CNAM Paris, France
**Francesca Palumbo** Maître de conférences, University of Sassari, Italy
**Jean-Christophe Prévotet** Maître de conférences, INSA Rennes, France
**Maryline Hélard** Professeure, Directrice de thèse, INSA Rennes, France

Invitées :

**Daniel Chillet** Professeur, ENSSAT, Université Rennes1, France

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

INSA RENNES

# An Efficient Computer-Aided Design Methodology For FPGA & ASIC High-Level Power Estimation Based on Machine Learning

**Yehya Abed El-Fattah NASSER**

**En partenariat avec :**

IETR INSTITUT D'ÉLECTRONIQUE ET DE TÉLÉCOMMUNICATIONS DE RENNES

UNIVERSITY OF CAGLIARI

*Document protégé par les droits d'auteur*

# Acknowledgements

First and foremost, I am grateful to Allah for the good health and well-being that were necessary to complete this dissertation. I thank Allah for the numerous blessings He has bestowed upon me throughout my dissertation journey.

At this moment of accomplishment, it gives me a great pleasure in expressing my gratitude to all those people who have supported me and had their contributions in making this thesis possible. I would like to thank my supervisor, Prof. Maryline Hélard, for the patient guidance, encouragement and advice she has provided throughout my time as her student. Thank you for allowing me to grow as a research scientist. Your advice on both research as well as on my career have been invaluable. My sincere thanks also go to Dr. Jean-Christophe Prévotet. I am extremely thankful and indebted to him for sharing expertise, and sincere and valuable guidance and encouragement. Thank you for steering me in the right direction whenever you thought I needed it.

I would also like to thank my committee members, Prof. Dimitrios Soudris, Prof. Muhammad Shafique, Prof. Pascal Chevalier, Prof. Daniel Chillet and Dr. Francesca Palumbo for serving as my committee members and for providing me with valuable comments and suggestions on this thesis. In parallel, I would like to thank all my colleagues in IETR lab at my engineering school INSA Rennes. Additionally, I would like to thank the EOLAB team members at University of Calgiari, Dr. Francesca Palumbo, Dr. Carlo Sau, and Dr. Tiziana Fanni, for their support during my research visit there.

Moreover, I would like to thank the School of Engineering at the Lebanese International University, LIU (Beirut, Lebanon), and all its faculty members, staff and colleagues for providing me a strong background to continue my Ph.D.

I would like also to thank my friends Ali El-Amine and Marwan El-Hajj for all the time we have spent in the last three years.

A special thanks to my parents whose undying love, unconditional support

and guidance are with me in whatever I pursue. They are my ultimate role models. Words cannot express how grateful I am to my father and my mother for all of the sacrifices that you have made on my behalf. Your prayer for me was what sustained me thus far. Thank you mom and dad for showing faith in me and giving me the liberty to choose what I desired. You shaped my values and made me the person that I am today.

I would like to express my special appreciation and gratitude to Salam, my soulmate, my beloved wife and my best friend, for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. She experienced all of the ups and downs of my research and so she deserves big thanks for helping me keep things in perspective. Thank you Salam for providing me with all the love and caring I needed to stay strong and to continue till the end to achieve my goals. Thank you for your concerned, late night vigils despite the geographical distance between you (in Lebanon) and me (in France). Thank you for always showing how proud you are of me. Thank you so much for the valuable advice, constructive criticism and timely suggestions throughout my research work. It is such a nice feeling to have a partner with the same engineering spirit like you. Without your precious support, and persistent help, this dissertation would not have been possible.

It is my fortune to gratefully acknowledge the people who mean a lot to me. Thank you my sister Doha for providing me an unending inspiration and for supporting me spiritually throughout writing this thesis and for always cheering me up. Thank you for listening to my problems and providing perspective.

Special thanks go also to my extended family, my father-in-law, my mother-in-law, my sister-in-law, and my brothers-in-law for their continued support and encouragement. I consider myself the luckiest in the world to have such a lovely and caring family, standing beside me with their love.

To not forget, my thesis was defended during the Lebanese Revolution 2019...

Finally, I would like to share with you the motto (words said by Steve Jobs) that I believe in : "You've got to find what you love. And that is as true for your work as it is for your lovers. Your work is going to fill a large part of your life, and the only way to be truly satisfied is to do what you believe is great work. And the only way to do great work is to love what you do. If you haven't found it yet, keep looking. Don't settle. As with all matters of the heart, you'll know when you find it. And, like any great relationship, it just gets better and better as the years roll on. So keep looking until you find it. Don't settle... ".

# Abstract of the Thesis

Nowadays, advanced digital systems are required to address complex functionalities in a very wide range of applications. Systems complexity imposes designers to respect different design constraints such as the performance, area, power consumption and the time-to-market. The best design choice is the one that respects all of these constraints. To select an efficient design, designers need to quickly assess the possible architectures. In this thesis, we focus on facilitating the evaluation of the power consumption for both signal processing and hardware design engineers, so that it is possible to maintain fast, accurate and flexible power estimation. We present *NeuPow* as a system-level FPGA/ASIC power estimation method based on machine learning.

We exploit neural networks to aid the designers in exploring the dynamic power consumption of possible architectural solutions. NeuPow relies on propagating the signals throughout connected neural models to predict the power consumption of a composite system at high-level of abstractions. We also provide an upgraded version that is frequency aware estimation. To prove the effectiveness of the proposed methodology, assessments such as technology and scalability studies have been conducted on ASIC and FPGA. Results show very good estimation accuracy with less than 10% of relative error independently from the technology and the design size. NeuPow maintains high design productivity, where the simulation time obtained is significantly improved compared to those obtained with conventional design tools.

# Résumé de la Thèse

## Résumé

Aujourd'hui, des systèmes numériques avancés sont nécessaires pour mettre en œuvre des fonctionnalités complexes. Cette complexité impose au concepteur de respecter différentes contraintes de conception telles que la performance, la surface, la consommation électrique et le délai de mise sur le marché. Pour effectuer une conception efficace, les concepteurs doivent rapidement évaluer les différentes architectures possibles. Dans cette thèse, nous nous concentrons sur l'évaluation de la consommation d'énergie afin de fournir une méthode d'estimation de puissance rapide, précise et flexible. Nous présentons NeuPow qui une méthode s'appliquant aux FPGA et ASIC. Cette approche système est basée sur des techniques d'apprentissage statistique.

Notamment, nous exploitons les réseaux neuronaux pour aider les concepteurs à explorer la consommation d'énergie dynamique. NeuPow s'appuie sur la propagation des signaux à travers des modèles neuronaux connectés pour prédire la consommation d'énergie d'un système composite à haut niveau d'abstraction. La méthodologie permet de prendre en compte la fréquence de fonctionnement et les différentes technologies de circuits (ASIC et FPGA). Les résultats montrent une très bonne précision d'estimation avec moins de 10% d'erreur relative indépendamment de la technologie et de la taille du circuit. NeuPow permet d'obtenir une productivité de conception élevée. Les temps de simulation obtenus sont significativement améliorés par rapport à ceux obtenus avec les outils de conception conventionnels.

## Introduction

Selon une étude récente menée par Cisco en 2018, 500 milliards d'appareils électroniques devraient être connectés à Internet d'ici 2030 [1]. La croissance rapide de cette technologie de communication dans des domaines d'applications très vastes oblige les fabricants à accélérer la production et de cherche de nouvelles améliorations à différents niveaux.

L'objectif de ces améliorations est de maintenir un certain niveau de qualité

de production dans un laps de temps très court, réduisant ainsi le temps de mise sur le marché (TTM) de ces dispositifs.

La qualité d'un appareil électronique n'est pas seulement évaluée par sa fonctionnalité, mais aussi par le temps qu'il peut fonctionner sans avoir besoin de le recharger (surtout lorsqu'il s'agit de petits dispositifs alimentés sur batterie). Pour cette raison, ces appareils doivent avoir des profils de consommation d'énergie très faibles. Par conséquent, la réduction de la consommation d'énergie dans les appareils électroniques complexes est devenue l'un des plus grands défis pour les fabricants de systèmes. De cette façon, les fabricants doivent toujours garantir trois principes : 1) proposer des appareils électroniques de haute qualité/performance, 2) garantir une mise sur le marché rapide des produits, et 3) fournir des systèmes à faible consommation d'énergie.

Généralement, il est possible de se faire une idée sur l'énergie en mesurant la consommation de l'appareil dans ses conditions réelles d'exécution. La mesure de la puissance n'est pas toujours facile à réaliser et les concepteurs doivent finaliser tout le processus de conception afin d'obtenir les premiers résultats, ce qui est souvent prohibitif. De plus, chaque changement mineur dans la conception impose d'effectuer une nouvelle évaluation de puissance (cf. FIGURE 1). Il peut également arriver qu'après un fonctionnement à plein régime, la puissance consommée évaluée ne soit pas compatible avec les exigences initiales. Dans ce cas, les concepteurs doivent ré-exécuter un flot de conception complet à partir de zéro, ce qui est bien sûr long, source d'erreurs, et sans aucune garantie de succès.

Une autre solution pour évaluer le profil de consommation d'énergie consiste à estimer la puissance au lieu de la mesurer. En utilisant cette approche, les concepteurs peuvent avoir une idée globale sur la puissance consommée par un circuit donné et le respect des contraintes énergétiques. Il est clair que les mesures fournissent des valeurs réelles, alors que les estimations ne le font pas. Néanmoins, si la fidélité et la précision du modèle choisi est suffisante pour permettre d'opter pour une configuration optimale, ou au moins d'éliminer un ensemble de configurations possibles, alors les prédictions peuvent clairement améliorer la vie des concepteurs et raccourcir les délais de mise sur le marché.

Les concepts de fidélité et de précision sont fortement liés au niveau d'abstraction, et il est donc possible de définir deux niveaux d'abstraction comme suit : 1) **High-level** et 2) **Low-level**. Au niveau supérieur, on dispose de moins d'informations sur le système, mais au niveau inférieur, plus de détails sont fournis.

Comme indiqué à la FIGURE 2, un niveau d'estimation de puissance a un impact significatif sur deux facteurs : la vitesse d'estimation et la précision de l'estimation. Le compromis entre la complexité des modèles et la précision doit être poursuivi. Par exemple, dans le contexte des architectures reconfigurables à grain grossier, les architectures sont capables d'atteindre une bonne précision d'estimation dans les modèles de puissance proposés, tout en garantissant un faible temps d'exploration [2, 3]. Ceci est possible en combinant des informations technologiques de bas niveau et des informations de haut niveau liées à la

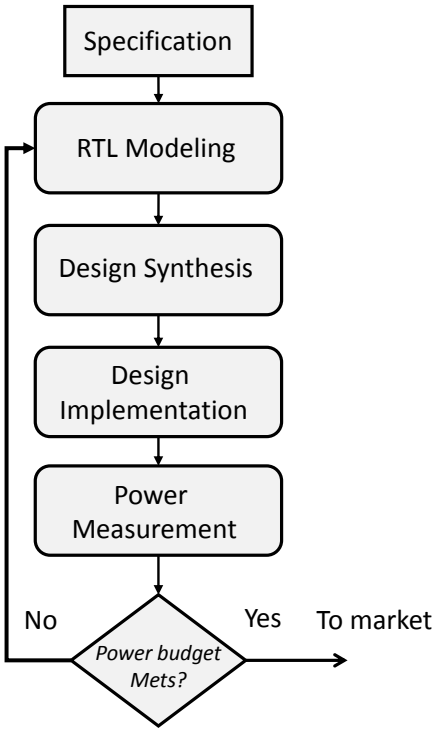



FIGURE 1 – Flot de conception matériel et cycle de mesures

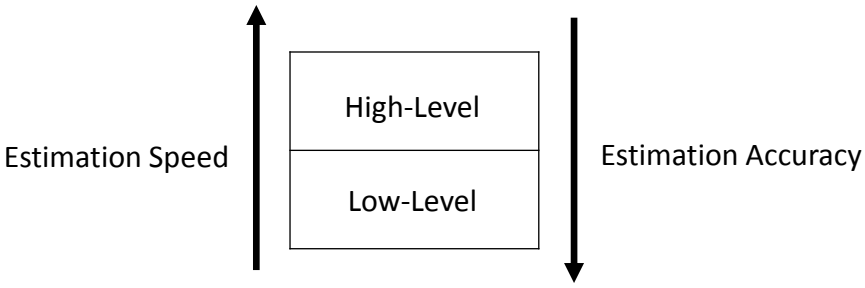



FIGURE 2 – Niveaux d'estimation en fonction de la vitesse et de la précision

représentation du flux de données de la fonctionnalité mise en oeuvre dans l'architecture. L'un des principaux enjeux de la modélisation est de déterminer le niveau correct d'abstraction et de granularité de la description des composants qui garantira un bon compromis entre la précision et la vitesse d'exploration. Un composant peut être un bloc de propriété intellectuelle (PI) doté d'une fonctionnalité et d'une structure complexes, ou il peut aussi être une simple porte logique. Le traitement des composants à grains fins, tels que les portes logiques, offre aux concepteurs une grande précision et une grande flexibilité, mais au détriment de

la vitesse et de la complexité de la simulation. D'un autre côté, travailler avec des composants à grain grossier permet d'accélérer le temps de simulation au prix d'une faible précision.

# État de l'art

De nombreux chercheurs ont exploité les approches basées sur des tables (LUT) afin de modéliser la puissance [4], [5], [6], [7] et [8]. Ces techniques consistent à tabuler des valeurs afin d'estimer la puissance globale d'un système. Pour traiter les valeurs stockées dans un tableau, différentes métriques peuvent être utilisées, telles que la densité de transition d'entrée, la probabilité statique d'entrée, les corrélations espace-temps, etc. En règle générale, une méthode supplémentaire est également nécessaire pour interpoler les valeurs manquantes dans le tableau. Dans cette partie, nous passons en revue tous les travaux existants utilisant la technique de modélisation de puissance à l'aide de tables. Bien qu'ils se soient avérés efficaces, atteignant une grande précision d'estimation, ils présentent aussi certaines limites car ils nécessitent une grande quantité de données à stocker, ce qui accentue l'effort de modélisation. D'autre part, cette technique est une méthode consommatrice de mémoire en raison de la grande quantité de données à stocker, et le processus de recherche se complexifie avec la taille du tableau. Ainsi, les chercheurs ont commencé à se focaliser sur des approches plus efficaces comme les techniques basées sur la régression et les réseaux neuronaux artificiels (ANNs).

Des techniques basées sur la régression sont utilisées pour faire correspondre les valeurs de puissance (obtenues à partir de simulations ou de mesures) à des paramètres spécifiques, tels que l'activité de commutation et la probabilité statique. Par conséquent, cette catégorie se concentre sur les techniques ascendantes basées sur une phase de caractérisation généralement réalisée à un faible niveau d'abstraction.

Shang et al. ont présenté une méthode de régression adaptative utilisée pour réduire le problème du nombre limité de séquences d'entraînement, en appelant un simulateur au niveau de la porte sur un petit nombre de cycles pour améliorer l'équation et obtenir une erreur relative de 3,1%. Bogliolo et al. ont proposé une technique de régression avancée, basée sur des arbres de régression afin d'améliorer la précision des modèles linéaires. Des variables de contrôle ont été définies pour déterminer les équations de régression les plus appropriées. Une caractérisation en ligne a également été proposée pour améliorer la précision de l'estimation de la puissance des petits circuits combinatoires de 34,6 % à 6,1 %. Jevtic à al. [9] a présenté un modèle de puissance pour les blocs DSP intégrés aux FPGAs et prenant en compte diverses statistiques de signal et tailles de multiplieurs. Le modèle a été réalisé à l'aide d'une régression multi-variables sur différentes mesures de puissance et une précision de 7,9 % a été obtenue.

Enfin, plusieurs travaux ont porté sur la création de modèles de puissance pour les opérateurs de base tels que les additionneurs ou les multiplieurs [10, 11, 12] sur FPGAs. En tenant compte de l'activité de commutation, de la fréquence de fonctionnement, du coefficient d'auto-corrélation et de la longueur des mots, un modèle de puissance peut être créé comme indiqué dans [12], entraînant une erreur moyenne de 10% sur FPGA. Toutefois, une telle approche ne tient pas compte des interconnexions supplémentaires lors de l'estimation de la consommation d'énergie d'une conception plus complexe. Dans [11], des modèles de surface et de puissance pour les IP en virgule fixe et en virgule flottante ont été développés par régression linéaire. Ici, le modèle de surface est utilisé pour estimer la puissance en considérant le nombre de ressources, les blocs de mémoire et le multiplieur. Les puissances statique et dynamique peuvent être estimées à partir de cette méthode, avec une erreur moyenne de 7%. Les opérateurs de modélisation sont bien adaptés pour estimer la puissance à un niveau d'abstraction plus élevé, ce qui conduit à une exploration rapide. Toutefois, de telles approches ne tiennent pas compte de la consommation d'énergie d'un système composé de différents composants. Par conséquent, l'exploration de l'espace de conception des différentes configurations possibles d'une conception donnée n'est pas possible.

Les travaux récents qui traitent de la modélisation de la puissance sont basés sur des approches d'apprentissage machine. Les ANNs sont généralement utilisés pour modéliser la consommation d'énergie. Ils peuvent être adoptés pour modéliser des systèmes non linéaires, puisqu'ils ont la capacité d'apprendre automatiquement les caractéristiques des données d'entrée. De plus, ils peuvent atteindre une très bonne précision avec une faible complexité. Par conséquent, au lieu d'utiliser un modèle mathématique, les modèles neuronaux sont appropriés pour la modélisation et l'estimation de la puissance des circuits numériques. Borovyi et al. [13] ont exploité les réseaux neuronaux pour estimer la consommation d'énergie des systèmes numériques. Ils ont utilisé des données réelles provenant du processeur ARM7TDMI afin de former l'ANN. La limite de cette approche est le manque de généralité, puisque les résultats ne sont valables que pour ce type de processeur. Hsieh et al. ont présenté une technique de modélisation de puissance des circuits séquentiels CMOS, basée sur l'utilisation des réseaux neuronaux récurrents (RNN). Ce travail tient compte de la caractéristique non linéaire des distributions de consommation d'énergie et de la corrélation temporelle des données d'entrée. Le nombre de paramètres nécessaires pour estimer la consommation d'énergie est d'environ 8, et il faut un cycle de calcul précis au niveau de l'entrée et de la sortie pour obtenir la dissipation de puissance. Puisque l'estimation de puissance nécessite le nombre de transitions de sortie d'un composant, une simulation comportementale est nécessaire au début. Les résultats expérimentaux témoignent d'une marge d'erreur d'environ 4,19%. Ce travail est limité par deux contraintes : 1) le nombre de paramètres nécessaires pour estimer la consommation d'énergie (qui est d'environ 8), 2) l'application de la technique de modélisation sur les circuits CMOS, sans profiter du fait que ces circuits CMOS

peuvent être composés de plusieurs composants matériels à grain fin. Hou et al. [14] ont exploité les réseaux neuronaux afin d'estimer la puissance des circuits VLSI en fonction de différents paramètres de spécifications tels que : fréquence, flash, ROM, RAM, GPIO, ADC, autres périphériques intégrés (DAC, Op Amp, Analog Comparator, DMA, Hardware Multiplier, SVS), segments LCD, timers, interface, paquet. Toutefois, ils ne tiennent pas compte des statistiques sur les stimuli qui ont un impact significatif sur la consommation d'énergie dynamique. De plus, ils ne présentent aucune évaluation du temps nécessaire pour effectuer l'estimation.

Contrairement aux approches précédentes, certains travaux ont choisi d'envisager un niveau de granularité inférieur en supposant qu'un système complet peut êre considéré comme un ensemble de composants interconnectés. Par exemple, Lorandel et al. se sont concentrés sur la puissance de modélisation au niveau IP. En particulier, ils ont utilisé les ANN pour modéliser, à la fois la consommation d'énergie et les activités de signal d'un bloc IP implémenté dans un FPGA. Ils ont démontré l'efficacité de leur approche qui permet d'estimer très rapidement la consommation d'énergie (le facteur d'accélération minimum atteint par les modèles neuronaux est d'environ 11500) et avec une bonne précision (environ 5%). Les auteurs discutent de la possibilité d'explorer l'espace de conception d'un système global, en estimant la consommation d'énergie d'un système basé sur un ensemble d'IP disposés en cascade, mais n'ont pas réalisé cette exploration.

Pour résumer, la plupart des limitations qui ne sont pas abordées dans les travaux antérieurs basés sur les ANNs sont les suivantes :

— Ces travaux ne tiennent pas compte de l'exploration de la puissance au niveau du système (la plupart des travaux portent sur des circuits et des blocs à grain grossier) ;

— Ils ne prennent pas en compte les défauts dans la phase de modélisation. Ces derniers ont un impact sur la consommation d'énergie.

— Ils omettent un facteur important comme la mesure de la fréquence d'horloge, qui a un impact important sur la consommation d'énergie dynamique ;

— Ils ne comparent pas leur approche avec d'autres techniques de modélisation existantes.

# Méthode

Dans cette section, nous cherchons à fournir un ensemble de modèles de consommation qui permettra aux concepteurs de systèmes numériques d'évaluer leurs choix de conception et de tester un grand nombre de scénarios d'utilisation. D'autre part, un outil de simulation associé permettra de simuler la consommation de ces systèmes en se basant sur les modèles précédemment développés et de tester les solutions proposées par les concepteurs. A cette fin, une bibliothèque

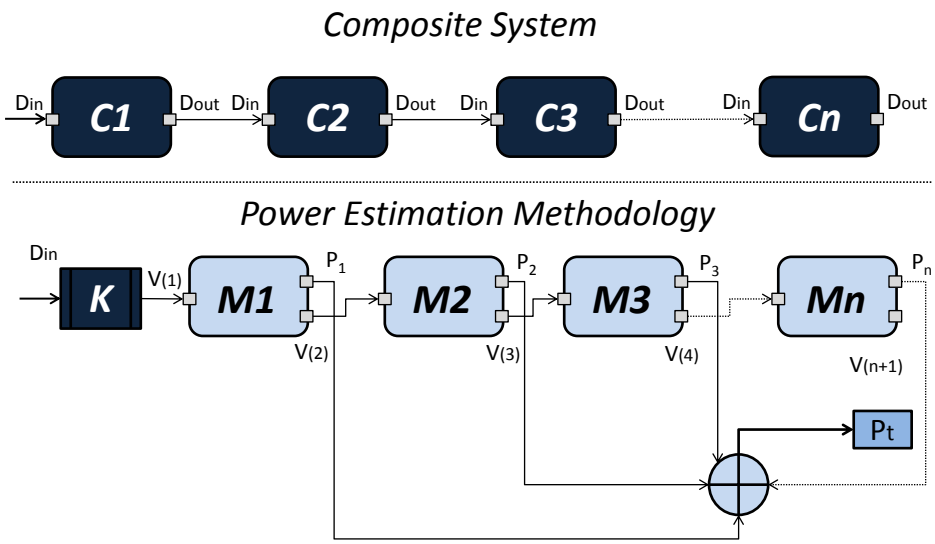



FIGURE 3 – Stratégie d'estimation proposée

de modèles de puissance doit être construite pour permettre aux concepteurs de tester facilement de nombreux scénarios d'architectures de circuits électroniques différents. Cette bibliothèque va certainement les aider à identifier la meilleure architecture en termes de consommation électrique.

Une méthode d'estimation sur des systèmes comportant plusieurs composants en cascade $(C1, C2, C3, ...., Cn)$ ou des couches décrites à un haut niveau d'abstraction est représenté dans le haut de la FIGURE 3. Parallèlement, la topologie du système peut être décrite à l'aide du modèle $M$ (qui comporte 2 sous-modèles), comme illustré dans le bas de FIGURE 4.22. La particularité de notre méthode est qu'elle propage les vecteurs de caractéristiques des entrées à travers tous les composants ou couches connectées, grâce au second modèle, qui estime les caractéristiques de sortie. A cette fin, il est possible d'obtenir une estimation de puissance au niveau du système $P_t$, qui correspond à la somme totale de la consommation électrique dynamique et qui peut être exprimée comme suit :

$$P_t = \sum_{i=1}^{i=n} P_i \tag{1}$$

où $i$ est l'indice du modèle du composant qui varie de 1 à $n$ composants.

Enfin, nous fournissons les étapes nécessaires à l'élaboration de notre stratégie d'estimation de puissance. A cet effet, deux méthodologies ont été proposées : une méthodologie de caractérisation et une méthodologie de modélisation. La première a pour but de caractériser la consommation d'énergie des composants, tandis que le second a pour objectif de modéliser la puissance elle-même. Comme illustré dans la FIGURE 4, ces deux méthodologies fournissent une bibliothèque

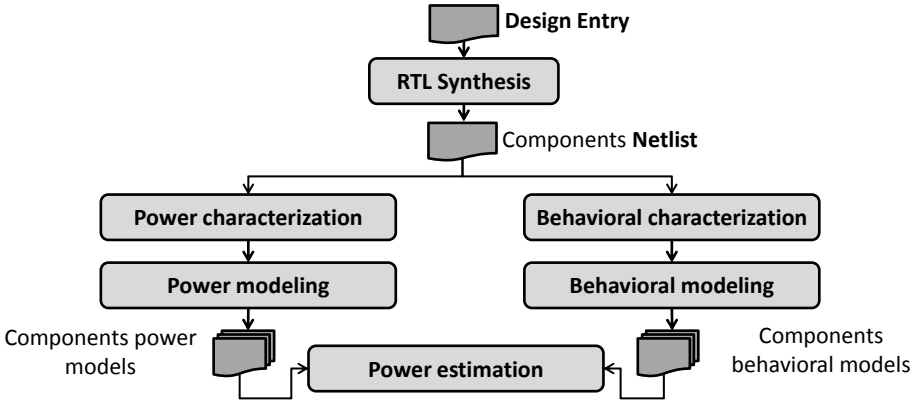



FIGURE 4 – Methodologies de caractérisation et de modélisation

de composants, avec les modèles de puissance associés, qui peuvent être utilisés pour estimer la consommation d'énergie pour différents choix de conception. La bibliothèque de modèles de composants est ensuite construite à partir des composants caractérisés qui existent sur une plate-forme matérielle/cible correspondant à la technologie ASICs ou aux plates-formes FPGAs. Ensuite, la bibliothèque de modèles correspondante est mise à la disposition des concepteurs travaillant à un haut niveau d'abstraction afin d'estimer la consommation électrique.

# Résultat

Dans cette partie, nous présentons les résultats qui justifient l'utilisation de la méthode d'estimation de puissance que nous proposons. Pour vérifier son efficacité, nous avons délibérément choisi des dispositifs reconfigurables tels que les FPGA, qui sont largement utilisés dans les systèmes de traitement de signaux numériques. Pour ce faire, nous développons des modèles de puissance basés sur des réseaux de neurones qui prédisent la puissance consommée par chaque composant numérique implémenté dans le FPGA. Dans ce travail, les modèles des composants sont interconnectés et l'information statistique des modèles de données se propage entre eux. Nous présentons également les estimations de puissance globale (ou au niveau du système) de différentes conceptions possibles. Pour évaluer la précision de nos modèles de puissance, nous avons effectué une comparaison avec les outils classiques qui sont basés sur des outils de conception assistée par ordinateur, tels que l'outil Xilinx XPower Analyzer (XPA).

Parallèlement, nous avons appliqué notre méthode d'estimation de puissance sur des ASIC semi-personnalisés. Nous avons également utlisé les réseaux neuronaux afin de modéliser la puissance des composants pour les technologies ASICs ($45nm$ et $15nm$). Ensuite, afin d'améliorer la première version de notre méthodologie NeuPow, nous avons proposé une nouvelle version qui prend en compte les fréquences. Nous avons prouvé l'efficacité de la méthodologie proposée en ef-

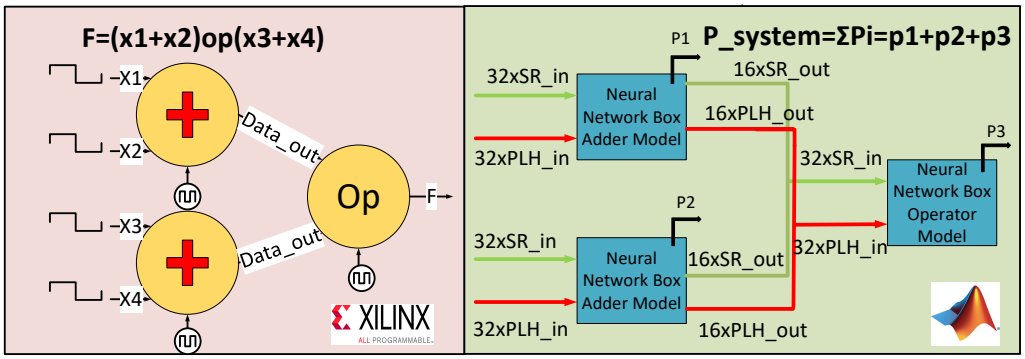



FIGURE 5 – System-level power estimation Xilinx XPA Vs Neural Model

fectuant différentes vérifications numériques à différents niveaux. De plus, nous avons mené plusieurs études de technologie et d'évolutivité pour prouver que notre méthode d'estimation est évolutive, rapide et robuste en termes d'estimation de puissance au niveau du système.

## Evaluations de la méthode sur le FPGA

Une vérification de plusieurs scénarios a été effectuée. Par exemple, Fig. 5 décrit une fonction composée de différents composants. Comme dans l'approche par composant, Xilinx XPA a été utilisé pour évaluer la puissance du système complet ainsi que les activités de commutation et le temps passé au niveau haut ($SR$ & $PLH$) des signaux. En parallèle, une estimation complète de la puissance du système a été modélisée à l'aide des modèles. Ce système, construit à l'aide de la plate-forme Matlab, consiste à interconnecter les modèles $M$ qui se composent de modèles $f$ et $g$ de chaque composant. Notez que les modèles $g$ des composants sont utilisés pour propager les paramètres ($SR$ & $PLH$) dans tous les composants et fournir une estimation de puissance globale.

Pour les deux estimations, 5000 échantillons de données ont été fournis aux modèles Xilinx XPA et neuronaux. Ces 5000 échantillons de données ont été extraits de Xilinx XPA (voir partie gauche de la FIGURE 5) pour la conception complète et des modèles neuronaux complets (voir partie droite de la Fig. 5). Notez que les deux fonctions ont été implémentées comme suit : $F_i = (a \text{ op } b) \text{ op } (c \text{ op } d)$, où *op* peut être une addition ou une multiplication. Comme dans l'Eq. 2, une erreur moyenne absolue en pourcentage est calculée sur 5000 échantillons de données (c.-à-d. $M = 5000$) au niveau système pour évaluer l'exactitude de la méthode proposée. $P_{XPA}$ est la consommation d'énergie obtenue à l'aide de l'outil XPA de Xilinx, et $P_{NN}$ est la consommation d'énergie estimée par les modèles de puissance du système composite. Selon les résultats fournis dans le tableau 1, on peut voir que notre méthode fournit une bonne précision au niveau du système avec une erreur qui est inférieure à 8%.

$$\%MAPE_{System-level} = \frac{100}{M} \sum_{i=1}^{i=M} \frac{|P_{XPA} - P_{NN}|}{P_{XPA}} \quad (2)$$

TABLE 1 – Précisions des modèles pour plusieurs fonctions

| Arithmetic Function | Neural models (MAPE %) |
|---|---|
| F1(op1=+, op2=+, op3=+) | 7.3699 |
| F2(op1=*, op2=*, op3=*) | 3.5158 |

Le tableau 2 montre les estimations initiales qui sont basées sur la somme de la puissance moyenne totale de chaque composant. Ces estimations sont obtenues en utilisant les modèles de puissance du composant $f$, sans tenir compte des modèles comportementaux $g$ (aucune propagation n'est effectuée). Nous remarquons que ces estimations présentent des résultats peu précis, avec une erreur relative supérieure à 15 %. Pour cette raison, la prise en compte des deux modèles (puissance et comportement) permet d'estimer avec précision la consommation d'énergie au niveau du système.

TABLE 2 – The effect of Signal Activity Propagation Versus Initial Estimates

| Arithmetic Function | XPA-based (mW) | Initial Estimates (mW) | Relative Error (%) |
|---|---|---|---|
| F1 | 4.6282 | 5.3637 | 15.8965 |
| F2 | 70.2986 | 82.2441 | 16.9925 |

Enfin, nous pouvons identifier les sources d'erreurs dans la méthodologie d'estimation proposée. Premièrement, les activités de commutation et le pourcentage élevé d'erreurs de modélisation, qui se propagent et s'accumulent tout au long de la conception. Deuxièmement, dans Xilinx XPA, le fait que les composants soient connectés à plusieurs voisins a un impact direct sur la capacité, et donc sur la consommation électrique des interconnexions. Notez ici que la consommation électrique des intra-connexions (à l'intérieur d'un composant) est prise en compte dans le modèle de puissance du composant. Bien que la méthode que nous proposons présente une erreur d'estimation d'environ 8 %, elle améliore quand même la productivité de la conception.

## évaluations de la méthode sur circuits ASIC

L'objectif de cette section est d'évaluer les avantages de la méthodologie d'estimation de puissance au niveau du système proposée, en utilisant différentes cas d'étude. Dans le tableau 3 les valeurs de puissance estimées en utilisant notre méthode sont fournies. Comme on peut le voir, la limite supérieure de l'erreur est

égale à 5,5%, par rapport à la cible. Dans NeuPow, le modèle comportemental $g$ pour chaque composant permet de propager l'activité de commutation entre les différents composants. Ces composants permettent d'obtenir des estimations de puissance plus précises au niveau du système. Cela permet aux concepteurs d'explorer différents choix de conception et de topologies.

TABLE 3 – System-level power estimation : Gate-level Vs Our Method, NeuPow

| Scalability Configurations | Gate-Level (Cadence®Genus) ($\mu W$ ) | NeuPow ($\mu W$ ) | % (Relative Error) |
|---|---|---|---|
| Complex Multiplier | 186.0959 | 190.9553 | 2.6112 |
| Magnitude Calculator | 372.2309 | 384.5835 | 3.3185 |
| Image Filter (4 Layers) | 746.2769 | 706.8775 | 5.3619 |
| Image Filter (6 Layers) | 1135.0000 | 1142.9000 | 0.6167 |
| Image Filter (8 Layers) | 1526.6000 | 1576.6000 | 3.2765 |
| Image Filter (10 Layers) | 1916.0000 | 2011.2000 | 4.9582 |

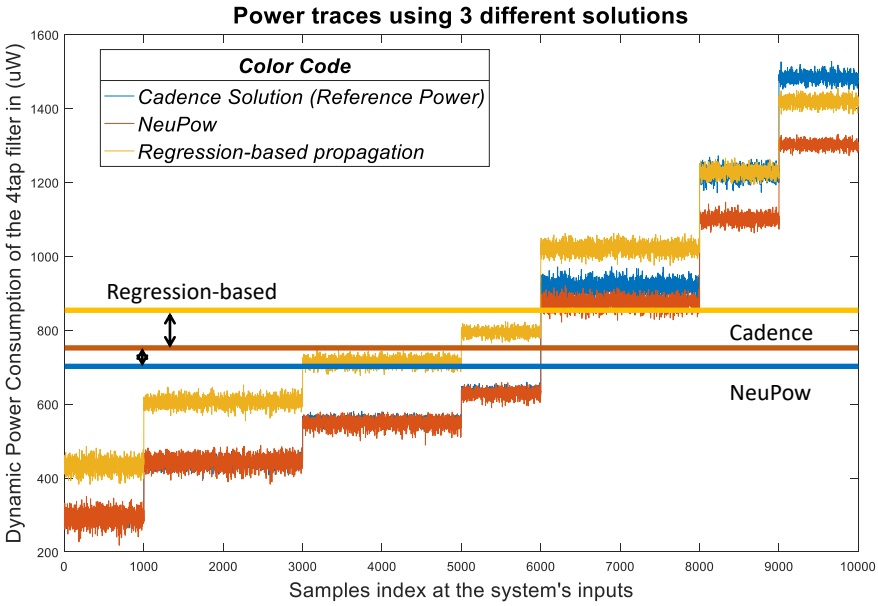



FIGURE 6 – Estimation de puissance basée sur la régression (i.e. filtre d'images à 4 taps )

En analysant les résultats numériques, nous pouvons remarquer que l'information relative à l'architecture est très importante pour estimer avec précision la consommation électrique au niveau du système. De plus, les estimations de puissance qui tiennent compte de la topologie du système composite (le cas de l'utilisation de modèles comportementaux $g$) donnent de meilleurs résultats que

dans le cas où l'information topologique fait défaut. De plus, le fait de conserver les informations sur les bits telles que le facteur d'activité et la probabilité statique au niveau des bits permet d'améliorer la précision sur l'estimation.

Pour conclure, la figure 6 montre les courbes de puissance obtenues en utilisant : 1) l'outil de cadence (référence), 2) la méthode basée sur la régression (méthode discutée ci-dessus), et 3) notre méthode proposée, NeuPow. Comme le montre cette figure, la courbe de puissance de NeuPow est supérieure à la courbe de régression pour la plupart des échantillons. Sur la base des lignes moyennes de puissance, il est clair que Neupow est relativement proche de la solution de cadence.

Dans le reste de cette section, nous présentons des résultats numériques qui visent à comparer le temps nécessaire pour exécuter les estimations de puissance à l'aide de notre méthode, par rapport à celui requis à l'aide d'un outil commercial de CAO, comme l'outil Cadence. Dans ce qui suit, nous supposons que le temps nécessaire à l'exécution de la modélisation, de la synthèse et de la mise en œvre du RTL est négligeable. Nous ne comparons que le temps nécessaire pour effectuer les estimations de puissance sur les deux solutions. Dans l'exploration ci-dessous, nous utilisons les mêmes conditions expérimentales (fréquence de fonctionnement, technologie cible et vecteurs de données d'entrée) pour chaque configuration de système considérée. La consommation d'énergie pour différentes tailles de filtres d'image (*4 couches*, *6 couches*, *8 couches* et *10 couches*) est extraite des deux méthodes, et les traces temporelles sont saisies. Chaque configuration est simulée afin d'obtenir 10000 valeurs de puissance correspondant aux 10000 vecteurs de données d'entrée différents. Table 4 affiche les résultats en termes de temps d'estimation (en minutes) pour chaque méthode. Par conséquent, pour explorer la consommation d'énergie pour chacun des 4 modèles considérés, les estimations de puissance au niveau de la porte nécessitent environ 223 minutes, alors que NeuPow ne nécessite que $2,3$ minutes. Cela correspond à un facteur d'accélération de 96 $\times$. A noter que le temps nécessaire pour caractériser et construire les modèles dans NeuPow n'est pas pris en compte dans les données rapportées dans Table 4. Quant aux simulations de puissance au niveau de la porte, elles sont effectuées sur un processeur Intel(R) Xeon (R) CPU E5-2620 @2.00GHz, tandis que les simulations *NeuPow* sont réalisées sur un Intel Core i5 @ 2.3GHz.

Table 4 – Comparaison du temps d'estimation de 2 approches : simulation au niveau portes Vs NeuPoW

| Case studies | Gate-Level (Cadence®Genus) (Minutes) | NeuPow (Minutes) | Speed-up factor X |
|---|---|---|---|
| *4 Layers* | 35 | 0.34 | 102.9412 |
| *6 Layers* | 48 | 0.47 | 102.1277 |
| *8 Layers* | 62 | 0.64 | 96.8750 |
| *10 Layers* | 78 | 0.84 | 92.8571 |
| **Total time** | **223** | **2.3** | **96.9565** |

# Conclusion

Dans cette thèse, nous avons vérifié numériquement la méthode d'estimation que nous avons proposée et qui est basée sur les réseaux neuronaux artificiels. Tout d'abord, des études de validation de principe ont été menées sur cible FPGA. Les résultats montrent que la méthode d'estimation proposée est capable d'estimer la consommation d'énergie au niveau des composants et du système. La précision de l'estimation au niveau des composants est inférieure à 1%. En ce qui concerne l'estimation de la puissance au niveau du système, une précision d'environ 8% est atteinte. Deuxièmement, la vérification de la méthode d'estimation sur des plates-formes ASIC semi-personnalisées a été effectuée. La précision de l'estimation sur les technologies CMOS 45nm et FInFET 15nm a également été étudiée. La méthode a montré une précision d'estimation d'environ 10 %. Sur cette base, d'autres études ont été menées. Des études d'extensibilité ont été réalisées, en explorant différentes tailles de conception. De plus, la vitesse d'estimation a été quantifiée en évaluant le temps nécessaire pour estimer un grand nombre de points de conception. Les résultats montrent que la méthode est évolutive et fournit un facteur d'accélération d'estimation de 96 $\times$ comparé au flot d'estimation de puissance classique.

A l'avenir, nous avons l'intention d'affiner davantage la méthodologie NeuPow en évaluant plus en détails la performance de la méthodologie proposée lorsque la fréquence et les paramètres technologiques varient.

# Contents

**Acknowledgements** **i**

**Abstract** **iii**

**Résumé** **v**

**1 Introduction** **1**
1.1 Context Study . . . . . 1
1.2 Power Estimation Issues . . . . . 2
1.3 Thesis Organization . . . . . 4
1.4 Publications and Awards . . . . . 5
1.4.1 Submitted Journal Papers . . . . . 5
1.4.2 International Conferences . . . . . 6
1.4.3 National Conferences . . . . . 6
1.4.4 Awards and Grants . . . . . 7

**2 Background** **9**
2.1 Introduction . . . . . 9
2.2 Power Consumption Dimensions . . . . . 9
2.2.1 Static Power . . . . . 10
2.2.2 Dynamic Power . . . . . 12
2.3 Hardware Platforms . . . . . 13
2.3.1 Field-Programmable-Gate-Array . . . . . 13
2.3.1.1 FPGA Design Flow . . . . . 15
2.3.2 Application-Specific Integrated Circuits . . . . . 17
2.3.2.1 ASIC Architecture . . . . . 17
2.3.3 ASIC Vs FPGA . . . . . 19
2.4 A Machine Learning Overview . . . . . 21
2.4.1 Unsupervised Learning . . . . . 21
2.4.2 Supervised Learning . . . . . 21

**3 Power Measures, Estimates and Modeling Methods: A Survey** **25**
3.1 Introduction . . . . . 25

3.2 Power Measurement . . . . . 26
3.2.1 On-board Solution . . . . . 26
3.2.2 External Solution . . . . . 28
3.2.3 Summary . . . . . 29
3.3 Power Estimation Techniques . . . . . 30
3.3.1 Simulation-based . . . . . 30
3.3.2 Statistical-based . . . . . 31
3.3.3 Probabilistic-based . . . . . 32
3.3.4 Power Estimation Tools . . . . . 33
3.3.5 Summary . . . . . 34
3.4 Power Modeling Techniques . . . . . 35
3.4.1 Analytical techniques . . . . . 37
3.4.2 Table-based power modeling technique . . . . . 41
3.4.3 Regression-based techniques . . . . . 44
3.4.4 Neural Networks based techniques . . . . . 46
3.4.5 Analysis and Discussions . . . . . 50
3.5 Literature Review Clustering . . . . . 55
3.6 Conclusion . . . . . 57

**4 Power Estimation Method 59**
4.1 Introduction . . . . . 59
4.2 Power Estimation Strategy . . . . . 59
4.2.1 Power Estimation from Users' Perspectives . . . . . 60
4.2.2 Model Definition . . . . . 61
4.2.3 System-level Power Estimation . . . . . 61
4.3 Characterization Methodology . . . . . 63
4.3.1 Parameters Motivation . . . . . 64
4.3.2 Parameters Boundaries . . . . . 66
4.3.3 Component Definition and Waveform Generation . . . . . 70
4.3.4 Power Characterization . . . . . 72
4.3.5 Behavioral Characterization . . . . . 73
4.4 Modeling Methodology . . . . . 74
4.4.1 Power Modeling . . . . . 75
4.4.2 Behavioral Modeling . . . . . 77
4.4.3 Library of component's model construction . . . . . 79
4.5 Method Extension . . . . . 79
4.5.1 Component's Model Extension . . . . . 80
4.5.2 Upgraded System-level Power Estimation . . . . . 80
4.6 Conclusion . . . . . 81

**5 Numerical Results for FPGA/ASIC Power Estimation Strategy 83**
5.1 Introduction . . . . . 83
5.2 NeuPow Assessments on FPGA . . . . . 83
5.2.1 Characterization and Modeling: Per-Component . . . . . 84
5.2.2 Power Estimation: Per-System . . . . . 86
5.3 NeuPow Verification on ASIC . . . . . 89
5.3.1 Use Cases Description . . . . . 89
5.3.2 Glitch Impact on Power Estimation . . . . . 91
5.3.3 Model Dimensioning: Per-Component . . . . . 92
5.3.4 Characterization and Modeling: Per-Component . . . . . 94
5.3.4.1 Neural-Based Modeling . . . . . 94
5.3.4.2 Regression Versus Neural-based Modeling . . . . 95
5.3.5 Power Estimation: Per-System . . . . . 98
5.3.5.1 Propagation-less (Straight-forward) Power Estimations . . . . . 99
5.3.5.2 Regression-based (Literature) Power Estimation . 100
5.3.5.3 Neural-based Approach (our) Power Estimation . 102
5.3.6 Technology Verification . . . . . 103
5.3.7 Summary . . . . . 105
5.4 NeuPow: Estimation Time Assessment . . . . . 105
5.5 NeuPow: Scalability Study . . . . . 108
5.6 NeuPow: Extension Evaluation . . . . . 109
5.7 Conclusion . . . . . 112

**6 Towards Accurate Power Models 113**
6.1 Introduction . . . . . 113
6.2 Power Modeling Method . . . . . 113
6.2.1 Measurement-based Power Characterization . . . . . 115
6.2.1.1 Characterization Parameters . . . . . 117
6.2.1.2 Measurement Process . . . . . 117
6.3 Experimental Results . . . . . 120
6.3.1 Power Characterization Results . . . . . 120
6.3.2 Tool Vs Measurement Results . . . . . 122
6.3.3 Preliminary Measurement-based Power Models . . . . . 124
6.4 Conclusion . . . . . 126

**7 Conclusion and Perspectives 127**
7.1 Conclusion . . . . . 127
7.2 Future Work and Research Directions . . . . . 130

**A Motivations 133**
A.1 Motivations . . . . . 133
A.1.1 Motivation To Propagation-based Power Estimation . . . . 133

A.1.1.1 Evaluation Method . . . . . 133
A.1.1.2 Numerical Results . . . . . 135
A.1.2 Why Neural Networks? . . . . . 138
A.1.3 Conclusion . . . . . 139

**B Demonstrations** **143**
B.1 Introduction . . . . . 143
B.2 Power Estimation Description . . . . . 143
B.3 Image Filter Power Profile . . . . . 144
B.4 Conclusion . . . . . 146

# List of Figures


1 Flot de conception matériel et cycle de mesures . . . . . vii
2 Niveaux d'estimation en fonction de la vitesse et de la précision . vii
3 Stratégie d'estimation proposée . . . . . xi
4 Methodologies de caractérisation et de modélisation . . . . . xii
5 System-level power estimation Xilinx XPA Vs Neural Model . . . xiii
6 Estimation de puissance basée sur la régression (i.e. filtre d'images à 4 taps ) . . . . . xv

1.1 Manufacturers constraints . . . . . 2
1.2 Hardware design flow . . . . . 3
1.3 Estimation levels Vs estimation speed and accuracy . . . . . 3

2.1 Digital system . . . . . 10
2.2 System on-Chip (SoC) power consumption trend from [15] . . . . 11
2.3 CMOS Inverter . . . . . 12
2.4 VLSI Classes . . . . . 14
2.5 Basic FPGA architecture [16] . . . . . 15
2.6 FPGA logic block [16] . . . . . 15
2.7 FPGA blocks [16] . . . . . 16
2.8 FPGA Design Flow . . . . . 16
2.9 Application Specific Categories . . . . . 18
2.10 Gate-Array ASICs [16] . . . . . 19
2.11 Standard-Cell-Based ASICs [16] . . . . . 19
2.12 Semicustom ASICs Design Flow . . . . . 20
2.13 Machine Learning Hierarchy . . . . . 22
2.14 Architecture of a Multi-Layer Perceptrons . . . . . 23

3.1 Power Consumption Evaluation Taxonomy . . . . . 26
3.2 On-board Power Measurement System . . . . . 27
3.3 External Power Measurement System . . . . . 28
3.4 Power Estimation Methods Trade-off . . . . . 35
3.5 Modeling Techniques Literature Review . . . . . 36
3.6 Neural Power Model . . . . . 47
3.7 Power Modeling Accuracy Versus The Observed Literature . . . . 51

3.8 Comparison on modeling techniques . . . . . 55
3.9 Power Characterization Techniques Versus Power Modeling Approaches . . . . . 56

4.1 Proposed Power Estimation Method from Users' Perspectives . . . 60
4.2 Component Model . . . . . 61
4.3 Proposed power estimation strategy . . . . . 62
4.4 Characterization and modeling methodologies . . . . . 63
4.5 Plot of the total power for an ALU for $D_{in} = 0.3$ and different $P_{in}$ from [7] . . . . . 66
4.6 Block data of N rows (inputs) and L columns (time window) . . . 67
4.7 Relationship between the activity factor and the static probability of a binary signal at given bit . . . . . 68
4.8 Plot of the relationship between the activity factor and the static probability of a binary signal (glitchy) at given bit . . . . . 69
4.9 Glitchy and glitch-free data packet/block stimuli generation flow. 70
4.10 The characterized component with N inputs . . . . . 71
4.11 Activity factor and static probability generation files preparation . 71
4.12 Waveform generation block . . . . . 72
4.13 Detail of power characterization . . . . . 73
4.14 Detail of behavioral characterization for a blackbox component . . 74
4.15 Component's model . . . . . 75
4.16 Power model $f : V_{in} \rightarrow P_{dynamic}$ . . . . . 76
4.17 Adopted artificial neural networks power model $f$. . . . . 77
4.18 Details of the behavioral model $g : V_{in} \rightarrow V_{out}$. . . . . 78
4.19 Adopted artificial neural networks for the component's behavioral model $g$ . . . . . 78
4.20 Off-the-shelf power models for high-level power estimation . . . . 79
4.21 Proposed Frequency-Aware Power Model . . . . . 80
4.22 Power estimation method with the extended power model . . . . . 81

5.1 Component's power and behavioral characterization flow on FPGA 85
5.2 Xilinx XPA Vs Neural Model Estimations . . . . . 86
5.3 System-level power estimation Xilinx XPA Vs Neural Model . . . 87
5.4 Library of components described at RTL . . . . . 90
5.5 Complex multiplier and magnitude square calculator . . . . . 90
5.6 Image filter (4layers/taps) . . . . . 91
5.7 Glitches illustration at the component's outputs . . . . . 92
5.8 Image filter (N layers/taps) . . . . . 99
5.9 Regression-based for system-level power estimation (i.e., 4- tap Image filter) . . . . . 101
5.10 Regression-based for system-level power estimation (i.e. 4 tap Image filter) . . . . . 103

5.11 Classical RTL description of a composite system . . . . . 108
5.12 NeuPow description of the digital filter based on our constructed library of models . . . . . 109
5.13 Mac dynamic power consumption versus the frequency for 3 different input configurations . . . . . 110
5.14 Frequency-aware power estimation (i.e. 4 tap Image filter) . . . . 111

6.1 Measurement-based power modeling flow . . . . . 114
6.2 Single Component's Architecture . . . . . 114
6.3 FPGA Board Overview . . . . . 115
6.4 FPGA Power Measurement Schematic . . . . . 116
6.5 Post-synthesis schematic of n instances . . . . . 117
6.6 FPGA Chip: Measurement circuit . . . . . 118
6.7 Automated FPGA Measurement Process . . . . . 119
6.8 Measurement bench . . . . . 120
6.9 Adder Power Consumption Per Component . . . . . 121
6.10 Multiplier Power Consumption Per Component . . . . . 122
6.11 Measurement-based Vs Xilinx Xpa for Multiplier . . . . . 124
6.12 Measurement-based Vs Xilinx Xpa for Adder . . . . . 124
6.13 Measurement-based Multiplier Power Model . . . . . 125
6.14 Measurement-based Adder Power Model . . . . . 126

A.1 Power estimation of a composite system using the Vivado Xpower Analyzer . . . . . 134
A.2 Power estimation for each component separately . . . . . 135
A.3 MAC and P2S components . . . . . 136
A.4 RTL schematic of a neural network implemented on FPGA . . . . 137
A.5 Parallel to serial power consumption Vs the signal statistical parameters . . . . . 140
A.6 Multiply-accumulate power consumption Vs the signal statistical parameters . . . . . 141

B.1 High-level power estimation for a 4tap filter composite system . . 144
B.2 A Zoom on MAC and P2S components' models features propagation 144
B.3 Dynamic power consumption profile for a 4tap image filter implemented on the 45nm ASIC technology . . . . . 145
B.4 A Zoom on the dynamic power consumption . . . . . 145

# List of Tables


1 Précisions des modèles pour plusieurs fonctions . . . . . xiv
2 The effect of Signal Activity Propagation Versus Initial Estimates xiv
3 System-level power estimation : Gate-level Vs Our Method, NeuPow xv
4 Comparaison du temps d'estimation de 2 approches : simulation au niveau portes Vs NeuPoW . . . . . xvii

2.1 FPGA vs ASIC: A Comparison . . . . . 21

3.1 Estimation techniques advantages and limitations . . . . . 34
3.2 Classification of Main Analytical Modeling Techniques . . . . . 40
3.3 A example of a 3D look-up table based power modeling method . 41
3.4 Classification of Table-based Modeling Techniques . . . . . 43
3.5 Classification of Regression-based Techniques . . . . . 46
3.6 Classification of Neural-based Techniques . . . . . 50
3.7 Metrics Reward and Penalty . . . . . 52
3.8 Power modeling techniques Assessment . . . . . 52

4.1 Characterization Parameters Selection . . . . . 65

5.1 Accuracy of the power models comparing to the Xilinx XPA . . . 86
5.2 Models Accuracy for Different Functions . . . . . 87
5.3 The effect of Signal Activity Propagation Versus Initial Estimates 88
5.4 Required design steps to switch from $F1$ to $F2$ configurations . . 88
5.5 Model Dimensioning . . . . . 93
5.6 Modeling results for each component model (Values of$\sqrt{MSE}$ and $P_{avg}^{ref}$ are in $\mu W$). Numbers after the component label indicate the total number of bits at the component's inputs. . . . . 95
5.7 RMSE for the regression-based power modeling technique . . . . . 97
5.8 Regression-based Vs Artificial Neural Networks Power Modeling techniques . . . . . 98
5.9 System-level power estimations: on the impact of the signal propagation . . . . . 100
5.10 System-level power estimation: Gate-level Vs Our Method, NeuPow102
5.11 Power Models on 15nm Technology. . . . . 104

5.12 15nm Technology Validation . . . . . 104
5.13 Static power vs dynamic power 15nm Technology . . . . . 105
5.14 Estimation time: Low-level gate level power simulations Vs NeuPoW power simulations . . . . . 107
5.15 NeuPow: Scalability Studies . . . . . 109
5.16 Frequency-aware power estimations: Gate-level Vs NeuPow (Upgraded) . . . . . 112

6.1 Data packet: characteristics extraction . . . . . 118
6.2 Experimental results: Adder power characterization results . . . . 122
6.3 Experimental results: Multiplier power characterization results . . 123
6.4 Measurement Vs Estimation Tool (Xilinx Xpower Analyzer) . . . 123

A.1 Required resources for both MAC and P2S units at RTL (xc7z045ffg900-2 FPGA) . . . . . 136
A.2 Resource of the implemented neural network at RTL . . . . . 138

# Chapter 1

# Introduction

## 1.1 Context Study

According to a recent study conducted by Cisco in 2018, 500 billion electronic devices are expected to be connected to the Internet by 2030 [1]. Each of these devices is characterized by carrying several sensors that interact with the environment, and communicate over a network. This network, built from all of these connected devices (e.g, sensors and actuators), produces data that electronic devices use to collect, analyze, and provide observations. These observations can be exploited by decision centers for better actions (even remotely), avoiding by this any possible problem.

The fast growing of this important communications technology over the very wide fields of applications forces the leaders of the technology manufacturers to accelerate production, where improvements at different levels are needed. The objective of these improvements is to maintain a certain level of **quality production** within a short time slot, decreasing by this the **time-to-market** (TTM) of these devices.

The quality of an electronic device is not only assessed by its functionality, but also by how much time it can operate without the need of power outlets for recharging (specially when they are small-scale battery-based). For this reason, these devices must have very **low power consumption profiles**. Consequently, reducing power consumption in complex electronic devices is becoming one of the biggest challenge for manufacturers. The manufacturers should always guarantee three main factors: 1) to maintain **high quality/performance** electronic devices, 2) to guarantee a **short time-to-market** products, and 3) to provide **low power consumption** systems. These factors constitute the manufacturers' constraints that are illustrated in Figure 1.1.

Nowadays, power consumption has become a critical performance metric in a wide range of electronic devices, from autonomous embedded systems based on batteries to more complex systems requiring a high computing power. To over-

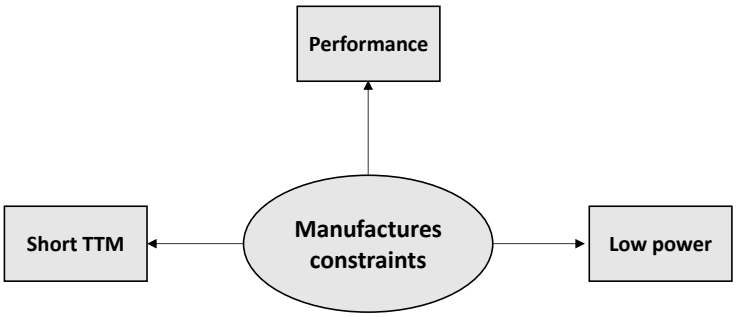



Figure 1.1 – Manufacturers constraints

come the problem of high power consumption, power reduction in Complementary Metal Oxide Semiconductor (CMOS) Very Large Scale Integration (VLSI) electronic devices is considered as an efficient solution [17]. Hence, having low power CMOS VLSI circuits is very crucial to avoid power dissipation, and extend battery lifetime. It also helps in reducing the thermal dissipation, which may affect the lifetime of the VLSI circuits and hence may degrade the performance. Consequently, many advantages in terms of costs are directly appreciable such as: a longer battery lifetime, a long lifetime of the circuit, a high performance, and a decrease in the leakage power.

## 1.2 Power Estimation Issues

One attempt to reduce the consumed power of devices would consist in allowing designers to explore various hardware architectures at an early stage of the design process. This would tend to reach an optimal architecture in terms of performance and power consumption. In order to optimize the power consumed in the end-product, it is necessary to have an idea about the power profile and to perform a deep analysis of the different factors that may impact power.

Generally, this is possible by measuring the device consumption in real conditions of execution. Measuring power is not always easy to perform and designers must complete the full design flow in order to obtain the first results, which is often prohibitive. Moreover, each minor change in the design imposes to perform a new power evaluation (see Figure 1.2). It may also happen that, after running a full design flow, the evaluated consumed power is not compatible with the initial requirements. In this case, designers have to re-run a full design process from scratch, which is of course time consuming and error prone, and without any guarantee of success.

An alternative solution for evaluating the power consumption profile is to estimate power instead of measuring it. Using this approach, designers may have

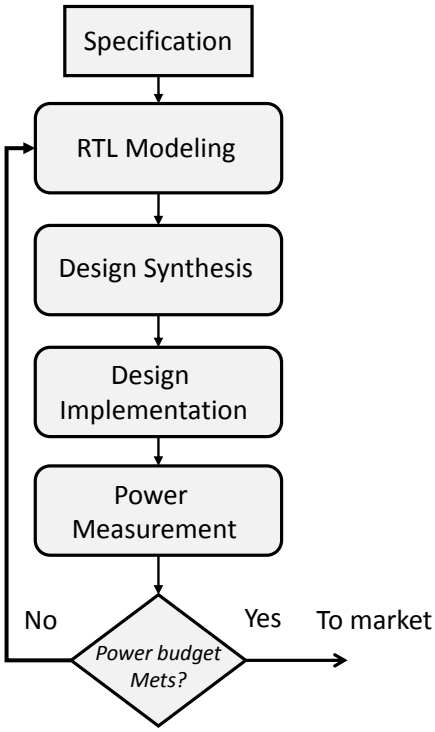



Figure 1.2 – Hardware design flow

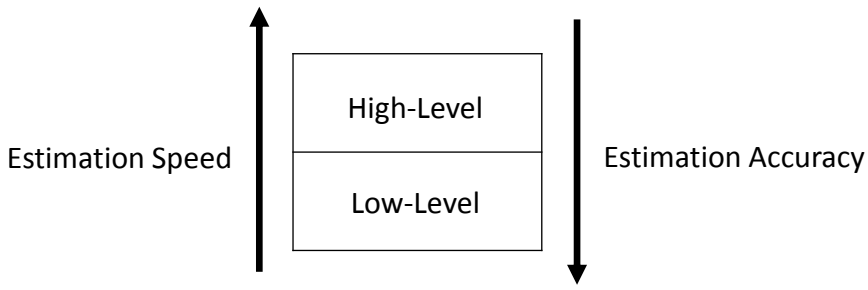



Figure 1.3 – Estimation levels Vs estimation speed and accuracy

a global idea whether the power consumed by a given design is compatible with the specified power budget. Clearly, measurements provide actual values, whereas estimations do not. Nevertheless, if the fidelity and the accuracy of the chosen model is good enough to allow to opt for an optimal configuration, or at least to eliminate a set of possible configurations, then predictions can clearly improve designers' lives and shorten time to market.

Fidelity and accuracy concepts are strongly linked to the level of abstraction. Therefore, it is possible to define two levels of abstraction as follows: 1) **High-**

**level** and 2) **Low-level**. At High-level, less information about the system is available, however, at low-level, more details are provided. As depicted in Figure 1.3, a power estimation level has a significant impact on the estimation speed and the estimation accuracy.

For instance, working at very **Low-level** of abstraction (at transistor or gate levels) provides very accurate models to the detriment of the simulation speed and flexibility [18]. In contrast, at **High-level**, a full system can be seen as a set of models without any implementation details, and can be easily modified and simulated by simply adjusting specific high-level parameters. Note here that the accuracy of power estimation at a high-level is typically very low since hardware-dependent factors are not taken into consideration. However, at **high level**, flexibility still needs to be sacrificed to obtain accurate power predictions. For instance, the authors in [19] presented a power-predictive environment specific for power-aware finite impulse response (FIR filter design) based on the Remez algorithm. On the other hand, more flexible solutions were proposed while focusing on modeling, with the advantage of enabling power saving techniques. For example, authors in [20] demonstrated an approach for power gating management in network-on-chip designs, while adopting a cycle accurate simulator.

In general, the trade-off between complexity of the models and accuracy should be pursued. As an example, in the context of coarse-grained reconfigurable architectures, the proposed power models were able to achieve good estimation accuracy while keeping the design exploration time low [2, 3]. This was performed by combining low level technological information and high level information related to the data flow representation of the functionality implemented in the architecture. A major issue when dealing with modeling is to determine the correct level of abstraction and granularity of the components description that will guarantee a good trade-off between accuracy and exploration speed. A component may be an Intellectual Property (IP) block with complex functionality and structure, or may also be a simple logic gate. Dealing with fine grain components, such as logic gates provides designers with high accuracy and flexibility, but to the detriment of simulation speed and complexity. Working with coarse grain components, on the other side, enables faster simulation time at the price of poor accuracy.

## 1.3 Thesis Organization

While the power budget for a given application is specified in the first stage of the design, it should be respected during the digital design implementation. To guarantee this matching, two solutions are possible, either measuring the power or estimating it. As previously discussed, power measurements introduce many limitations in terms of cost and time. Therefore, power estimation is often considered as an alternative solution that replaces measurement.

Our goal in this thesis was first to study different estimation methods, and compare them in terms of **estimation speed** and 2) **estimation accuracy**. In addition, our goal is to suggest an optimal estimation approach that takes into account the speed and precision of the estimation.

More specifically, we introduce in **Chapter 2** some background information related to power consumption, hardware platforms (FPGAs/ASICs) and machine learning techniques.

In **Chapter 3**, we deliver a detailed literature review about the existing power measurement methods, power modeling techniques and power estimation levels as well as techniques used in VLSI circuits.

In **Chapter 4**, we detail all the required steps to enable high-level power estimation. We also describe the proposed power and behavioral characterization method of hardware components, then we elaborate the adopted machine learning technique that solves the modeling issue for both power and behavior of several FPGAs/ASICs hardware components. In addition, we extend the proposed power models and finally deliver a library of accurate power models.

In **Chapter 5**, we enable high-level power exploration by utilizing the power models of the components for a global power estimation of a composite design. Equally important, we assess our proposed methodology at different levels: 1) speed level, 2) accuracy level, 3) scalability level and 4) technological level. Additionally, we provide a new estimation parameter and present a comparison among different modeling techniques.

In **Chapter 6**, we present the power measurement methodology on FPGA, with some assessments. Finally, a comparison between estimation tools and measurements method is provided.

Finally, we conclude the thesis in **Chapter 7** and propose new perspectives to enhance our estimation method.

## 1.4 Publications and Awards

### 1.4.1 Submitted Journal Papers

1. **Yehya Nasser**, Carlo Sau, Jean-Christophe Prévotet, Tiziana Fanni, Francesca Palumbo, Maryline Hélard, and Luigi Raffo. NeuPow: A CAD Methodology for High Level Power Estimation Based on Machine Learning. Submitted to the Special Issue on Machine Learning for CAD (ML-CAD). ACM Transactions on Design Automation of Electronic Systems 1, 1, Article 1 (January 2020), 29 pages. https://doi.org/10.1145/3388141
2. **Yehya Nasser**, Jordane Lorandel, Jean-Christophe Prévotet, and Maryline Hélard. Power Modeling and Estimation Techniques for FPGA and ASIC: A Survey. Submitted to the IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. Accepted (Major Review), March 2020.

### 1.4.2 International Conferences

1. **Yehya Nasser**, Jean-Christophe Prévotet, and Maryline Hélard. Characterization Methodology for Low Power Consumption Components on FPGA. **Accepted** in the 31st IEEE International Conference on Microelectronics, ICM 2019, December 15-18.
2. **Yehya Nasser**, Carlo Sau, Jean-Christophe Prévotet, Tiziana Fanni, Francesca Palumbo, Maryline Hélard, and Luigi Raffo. 2019. NeuPow: artificial neural networks for power and behavioral modeling of arithmetic components in 45nm ASICs technology. In Proceedings of the 16th ACM International Conference on Computing Frontiers (CF '19). ACM, New York, NY, USA, 183-189. DOI: https://doi.org/10.1145/3310273.3322820
3. **Yehya Nasser**, Jean-Christophe Prévotet, and Maryline Hélard. 2018. Power modeling on FPGA: a neural model for RT-level power estimation. In Proceedings of the 15th ACM International Conference on Computing Frontiers (CF '18). ACM, New York, NY, USA, 309-313. DOI: https://doi.org/10.1145/3203217.3204462
4. **Yehya Nasser**, Jean-Christophe Prévotet, and Maryline Hélard. "Power Aware Framework for Fast & Early Power Estimation on FPGA", DATE 2018, PhD Forum, Dresden, Germany.
5. **Yehya Nasser**, Jean-Christophe Prévotet, Maryline Hélard and Jordane Lorandel, "Dynamic power estimation based on switching activity propagation," 2017 27th International Conference on Field Programmable Logic and Applications (FPL), Ghent, 2017, pp. 1-2. doi: 10.23919/FPL.2017.8056783
6. **Yehya Nasser**, Jean-Christophe Prévotet, Maryline Hélard and Jordane Lorandel, "Power estimation on FPGAs based on signal information propagation through digital operators," 2017 IEEE Sensors Networks Smart and Emerging Technologies (IEEE SENSET), Beirut, 2017, pp. 1-4. doi: 10.1109/SENSET.2017.8125052

### 1.4.3 National Conferences

1. **Yehya Nasser**, Jean-Christophe Prévotet, Maryline Hélard, "NeuPow: Artificial Neural Networks for Power and Behavioral Modeling of Arithmetic Components in 45nm ASICs Technology", GdR SoC-SiP 2019, Montpelier, France.
2. **Yehya Nasser**, Jean-Christophe Prévotet and Maryline Hélard, "A Neural Model for RT-Level Power Estimation on FPGAs", GdR SoC-SiP 2018, Paris, France.
3. **Yehya Nasser**, Jean-Christophe Prévotet, Maryline Hélard and Jordane Lorandel, "Statistical Information Propagation Across Operators for Dynamic Power Estimation on FPGAs", GdR SoC-SiP 2017, Bordeaux, France.

### 1.4.4 Awards and Grants

1. Scholarship from Rennes Métropole (4000 Euros) for the research mobility (5 month) to EOLAB – Microelectronics and Bioengineering Lab, Department of Electrical and Electronic Engineering of University of Cagliari, Italy, 2018.
2. ACM Grant (1500 dollars) from the ACM International Conference on Computing Frontiers (CF '18), 2018.
3. DATE2018 Conference, Grant (300 euros), in the PhD Forum, DATE2018, Dresden, Germany.
4. Scholarship from the Foundation BPO, (5000 Euros) for our project GREENCOM, 2017.
5. First price, Best Poster price (750 Euros), GDR SOC2, University of Bordeaux, 2017.
6. Full scholarship for the PhD from the French Government (3 years) 2016-2019.

# Chapter 2

# Background

## 2.1 Introduction

In this chapter, we start by presenting some background information about the power consumption sources, different hardware platforms, and machine learning techniques. This part should help the reader to understand the basic concepts that are related to our proposed power estimation methodology. For this purpose, section 2.2 describes the sources of power consumption in VLSI and provides a detailed background on the model of power consumption. Furthermore, this section discusses the different components of power consumption. Section 2.3 describes the different hardware platforms and the design flow, and illustrates the traditional design and power estimation flow to be compared to our proposed method. Finally, section 2.4 offers a background on machine learning, particularly on artificial neural networks, which are used as a modeling method.

## 2.2 Power Consumption Dimensions

Any digital signal processing algorithms may be implemented by a typical digital system and may consist of a number of basic components as depicted in Figure 2.1. From an implementation perspective, the digital system may be implemented using a reconfigurable technology such as the Programmable Logic (PL) or the Application Specific Integrated Circuit (ASIC) technology.

Regarding the total power consumption $P_{total}$ of a digital system, it is decomposed into two components (as formulated in eq. 2.1):

$$P_{total} = P_{dynamic} + P_{static} \tag{2.1}$$

where

- $P_{static}$ is the **static power consumption**, which mainly results from the leakage current, that is always present when the circuit is powered on, and

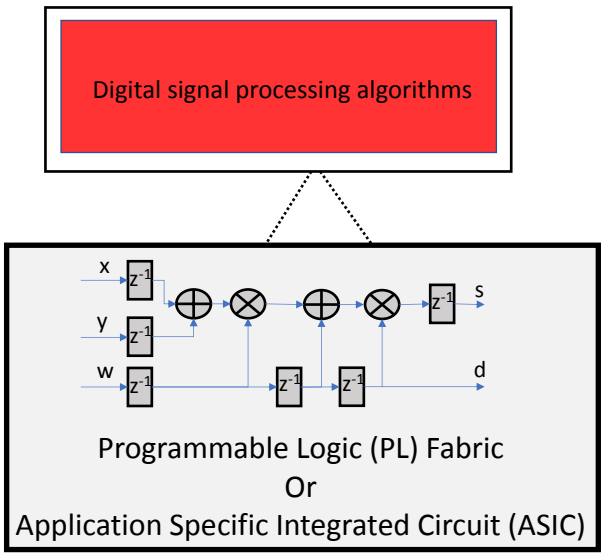



Figure 2.1 – Digital system

– $P_{dynamic}$ is the **dynamic power consumption**, which directly depends on the system switching activity, on operating frequency and the architecture of the system.

The first component causes circuits to consume power when no transitions are taking place, whereas the second arises when the nodes are in transition states [21].

In the following section, we present a theoretical background about power consumption in the digital CMOS circuits and its two components: 1) the static or also called the leakage power and 2) the dynamic power or also called the switching power. As illustrated in Figure 2.2, switching power and leakage power consumption increase with the advent of the technology. Switching power however will represent about 75% of the total power consumption by 2026 in SoC [15].

### 2.2.1 Static Power

With the advent of technology, the leakage or static power consumption (which is consumed by CMOS circuits when transistors are not switching) is becoming non negligible. This type of power consumption has three fundamental factors that contribute to the power dissipation: 1) the gate leakage, which is between the transistor's channel and gate, 2) the sub-threshold leakage current, which is between transistor's source and drain, and 3) the reverse bias current which is between the transistor's drain and the substrate. Note here that at the high-k dielectric devices, the main source of static power is due to the sub-threshold leakage current $I_{leakage}$ [22] that can be expressed as:

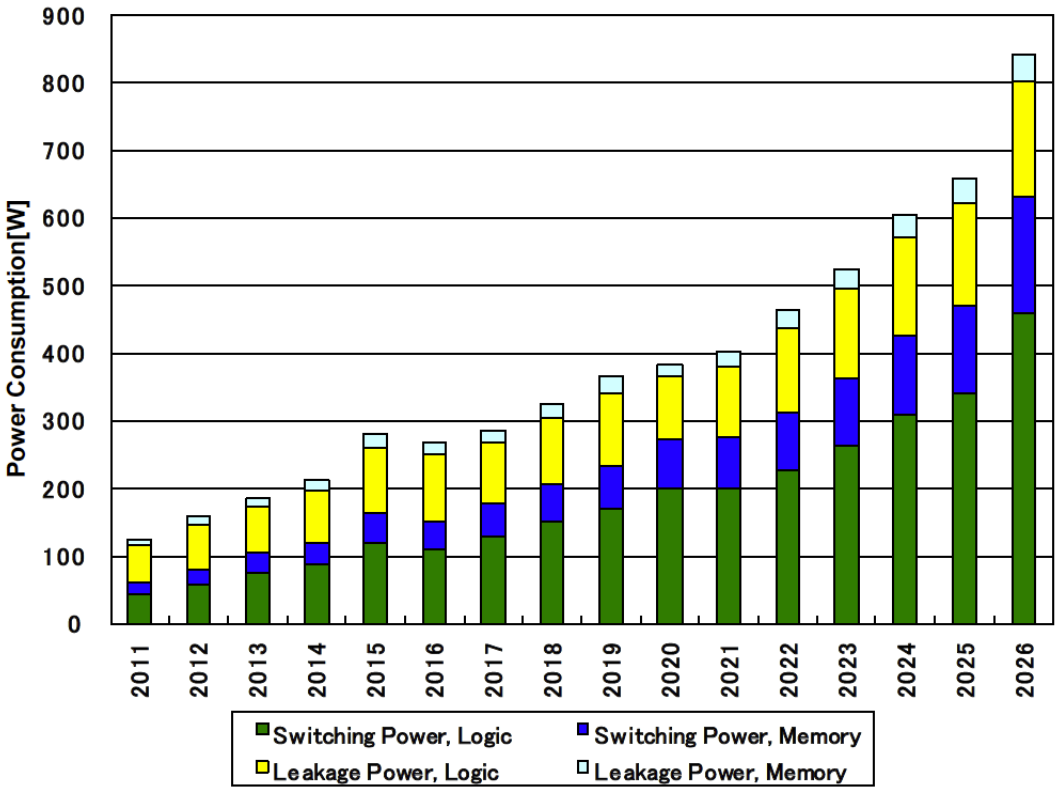



Figure 2.2 – System on-Chip (SoC) power consumption trend from [15]

$$I_{leakage} = I_0 e^{\frac{-qV_{th}}{\beta k_B T}} \tag{2.2}$$

where

- $I_0$ denotes a constant that depends on the dimension and the fabrication technology of the transistor;
- $\beta$ is a technology-dependent factor;
- $k_B$ stands for the *Boltzmann* constant, that equals to $1.380649 \times 10^{-}23$ J/K;
- $T$ represents the temperature, SI unit is Kelvin K;
- $q$ represents the charge of the electrical carrier, which equals to $1.6021765 \times 10^{-}19$ C;
- $V_{th}$ represents the threshold voltage, SI unit is Volt V.

The leakage current depends exponentially on $V_{th}$, so that every voltage drop in the threshold voltage leads to an increase in the leakage current. Moreover, eq. 2.2 points out that the leakage current depends exponentially on the transistor temperature $T$. This means that the environmental temperature significantly

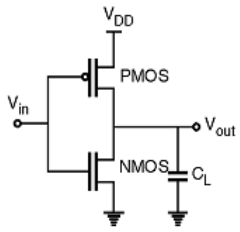



Figure 2.3 – CMOS Inverter

alters the leakage current, and hence the amount of static power can be expressed as in 2.3:

$$P_{static} = V_{dd} I_{leakage} \tag{2.3}$$

where $V_{dd}$ represents the supply voltage. By this, the main three factors that influence static power consumption are therefore the supply voltage $V_{dd}$, the threshold voltage $V_{th}$ and the temperature $T$.

### 2.2.2 Dynamic Power

Logic gates in CMOS circuits are implemented using two transistor types: n-channel MOS (NMOS) and p-channel MOS (PMOS). As shown in Figure 2.3, each gate is made of these two types of transistors, typically one terminal of the PMOS is connected to $V_{dd}$ and one terminal of the NMOS is connected to $V_{SS}$.

To better understand the gate operation, we start by assuming that the input voltage $V_{in}$ is at logic low. The PMOS transistor is then in the ON state with low impedance, while the NMOS is in the OFF state with high impedance. Then, a current, which flows to charge the load capacitance $C_L$ until the output voltage $V_{out}$ reaches the supply voltage $V_{dd}$ or the logic high. Therefore, the load capacitance models the total amount of output diffusion capacitance of the NMOS and PMOS transistors, the input capacitance of the fanout gates and parasitics and wires capacitance. Assuming now that the input voltage switches from logic low to logic high, then the PMOS transistor is in the OFF state with high impedance, while the NMOS transistor is in the ON state with very low impedance. Then, a path that connects the fully charged load capacitance to the ground $V_{SS}$ to discharge the charges until the output reaches the $V_{SS}$ or the logic low.

Therefore, the accumulation of the total energy consumed for charging and discharging the load capacitance $C_L$ within a cycle becomes equal to $C_L V_{dd}^2$, and the power consumption for a time interval $t$ can be then written as in eq. 2.4

$$P_t = \frac{C_L V_{dd}^2}{t} \tag{2.4}$$

where

- $C_L$ denotes the load capacitance;
- $V_{dd}$ represents the supply voltage;
- $t$ denotes the total time, which equals to $k.T_{clock}$;
- $k$ corresponds to the number of cycles;
- $T_{clock} = \frac{1}{F_{clock}}$, stands for the clock period time in a synchronous digital design.

For a node that toggles $N$ times over time interval $t$, the final dynamic power consumption model can be hence written as in eq. 2.5

$$P_{dynamic} = NP_t = N\frac{C_L V_{dd}^2}{t} = N\frac{C_L V_{dd}^2}{k.T_{clock}} = \alpha C_L V_{dd}^2 F_{clock} \tag{2.5}$$

where $\alpha$ denotes the average number of switches per cycle and is also called the activity factor. It can be expressed as in eq. 2.6.

$$\alpha = \frac{N}{k} \tag{2.6}$$

A small amount of $P_{dynamic}$ is due to a short circuit current that appears during the switching of both PMOS and NMOS transistors. This current causes the short circuit power, that is due to the fact that input data stimuli signals typically are not sharp signals. Consequently, during a small period of time, both NMOS and PMOS transistors are turned on simultaneously. This current flows between the supply voltage and the ground, leading to a short-circuit, that can contribute to up to 10% of the total dynamic power consumption [23].

## 2.3 Hardware Platforms

As illustrated in Figure 2.4, VLSI (Very Large Scale Integration) can be grouped into 2 categories: Programmable circuits and Hardware-defined circuits. In this part, we focus on the FPGA that stands for Field Programmable Gate Array, which is an integrated circuit that can be programmed to serve different designs specifications. We also target the ASIC platforms, which stand for Application Specific Integrated Circuits, that are specific for a dedicated application.

### 2.3.1 Field-Programmable-Gate-Array

FPGAs are used for rapid prototyping, so that any application can be implemented on such circuits as a first hardware prototype. Note here that FPGA prototyping allows designers to test the real physical environment of the applications. Besides, in some applications FPGAs are used as final-end product platforms. In this case, the use of the FPGAs depends on many factors: the size of the project, the desired performances and the production costs.

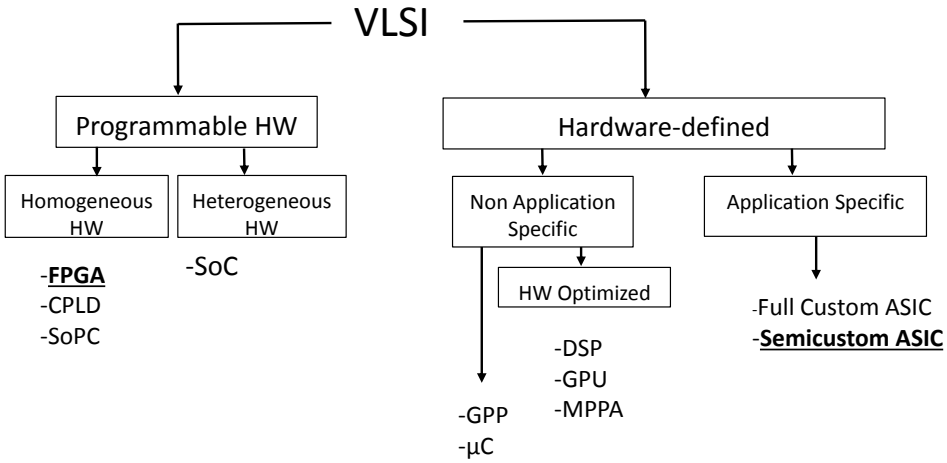



Figure 2.4 – VLSI Classes

FPGAs can be considered as off-the shelf devices, and can be configured at any time to perform different applications. Additionally, it is possible to alter the design many times until the design meets the designer's specifications and constraints. The main advantages of reconfigurable devices are listed as follows:

1. Reusability: the reconfigurable platforms provide the capability to easily alter the design to test different possible solutions and applications. This is possible due to the rewritable memory technology offered by these systems;
2. Scalability: these devices are capable to implement a few logic elements up to huge Intellectual Property (IP) blocks.

As depicted in Figure 2.5, an FPGA architecture consists of a two dimensional array of logic cells. The reconfigurable parts within FPGAs consist of configurable logic cells, the inputs/outputs (I/Os directions configuration) and the interconnections between the logic cells. FPGAs families differ from each other by the technology used, the physical architecture to implement the logic blocks, the interconnections layout, number of the logic blocks, number of block Random Access Memory (RAM) , and number of Digital Signal Processing (DSP) blocks .

The fundamental logic blocks (See Figure 2.6) are look-up tables (LUT) implemented as memory. LUTs have been taken as the mainstream logic block, and each Z-input LUT can implement any function with up to $Z$ variables and consists of $2^Z$ SRAM settings bits that store the required boolean truth table and $2^Z$ multiplexer that selects the appropriate SRAM content based on the specified LUT outputs. Note here, that a typical size for LUTs is 4 (See Fig.2.6) or 6.

Finally, Figure 2.7 shows an example of an FPGA fabric with its resources for an FPGA vendor such as Xilinx, and the resources are:

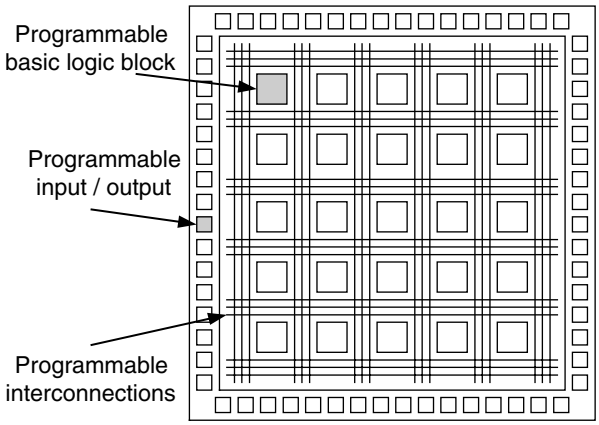



Figure 2.5 – Basic FPGA architecture [16]

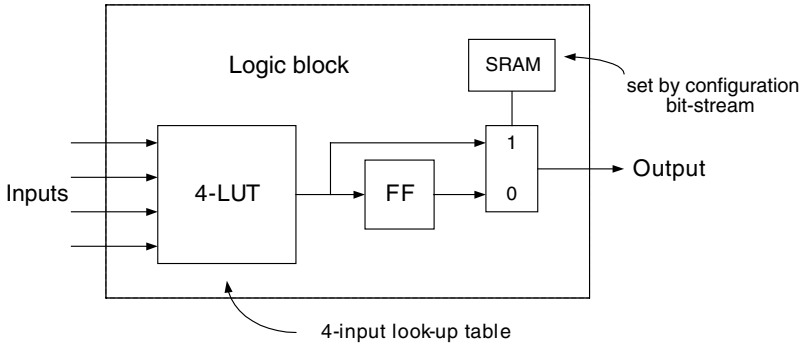



Figure 2.6 – FPGA logic block [16]

— Configurable logic blocks (CLBs), which contain combinational logic and registers;

— Input/output blocks (IOBs), interface between the FPGA and the external world;

— Programmable interconnections (PIs);

— RAM blocks;

— Three-state buffers, global clock buffers

— Dedicated multipliers.

#### 2.3.1.1 FPGA Design Flow

The FPGA design flow consists of the following steps: design entry (Specification), RTL Modeling, design synthesis, design implementation, bitstream generation, and device configuration. In parallel, design verification, which includes

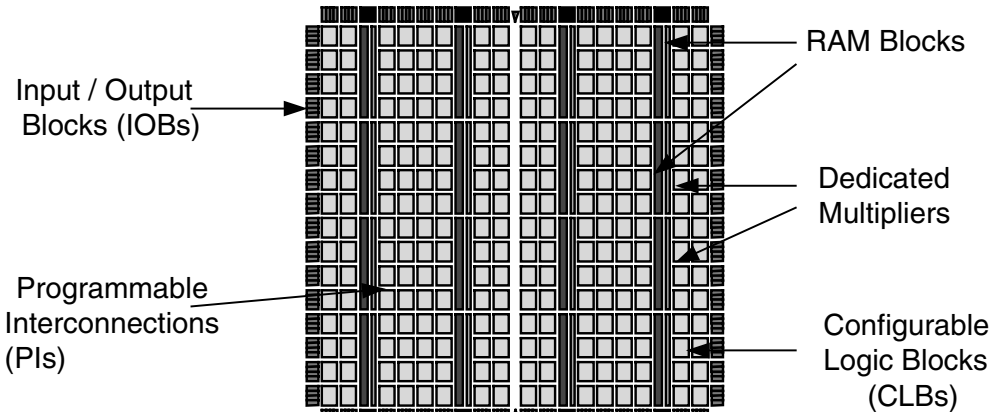



Figure 2.7 – FPGA blocks [16]

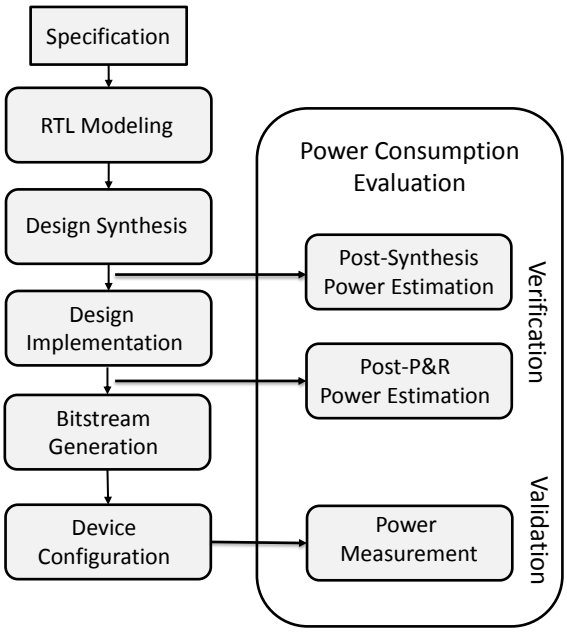



Figure 2.8 – FPGA Design Flow

both functional verification and timing verification, takes place at distinct phases during the design process. In addition to this, power estimation can be performed either after synthesis (Post-synthesis) or after implementation (Post-P&R). Finally, the FPGA design flow steps are described in details as depicted in Figure 2.8 and as follows:

— Specification: This step specifies the application requirements and constraints;

— RTL Modeling: It describes the design using hardware description language;

(HDL).

— Design Synthesis: It transforms the high-level of abstraction description to a low-level of logic abstraction, note here that the output of this phase is the netlist files;

— Design Implementation: This step assigns each logic element to a specific physical element, to map the logic gates into specific locations in the FPGA fabric, and to connect logic elements;

— Bitstream Generation: Generate the bit-stream file which is used to configure the FPGA;

— Device Configuration: Download the bit-stream into the FPGA target;

— Power Consumption Evaluation: Simulate the design in order to check the design power consumption. Note here that the power estimation can be performed at different levels: 1) Post-Synthesis level and 2) Post-P&R level. At both level, timing simulation can be achieved using:

a) the **standard delay file (SDF)**, where the generated timing information is stored; note here that this information represents the delays, timing constraints, incremental and absolute delays, conditional and unconditional module path delays and timing checks, library-specific information, Scaling, environmental, technology, and user-defined parameters.

b) the **netlist** (description of connectivity between the components).

Regarding the power estimation step, the main difference between the Post-synthesis and Post-P&R power estimations is the timing information. The former uses the estimated timing information, and the latter uses more precise timing information, that are accurately extracted taking into consideration the physical implementation details [24]. Finally, after the device configuration, it is also possible to start the power measurement process to validate the power budget of a given design.

### 2.3.2 Application-Specific Integrated Circuits

Application-specific integrated circuits (ASICs) point out to those integrated circuits specifically built for predefined functions.

#### 2.3.2.1 ASIC Architecture

It is possible to classify the application specific technology into two classes as follows: 1) full-Custom ASICs and 2) semicustom ASICs as depicted in Figure 2.9.

**Full-Custom ASIC** In full-custom ASICs, all mask layers are customized, and the design offers the highest performance and the smallest die size. However, a very high time-consuming design flow, a high design complexity and cost, and a

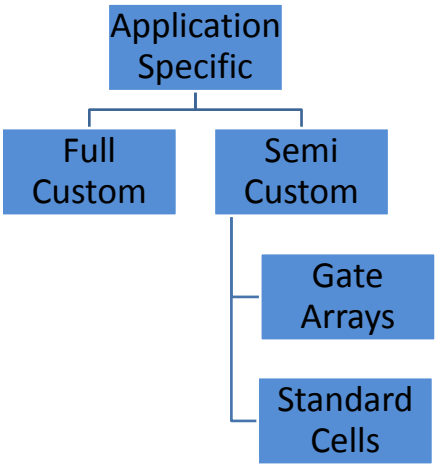



Figure 2.9 – Application Specific Categories

high risk of defects are true challenges. To this aim, fewer projects are really based on the full-custom ASICs. Examples of full-custom ASICs are the particular components of high-voltage, analog and mixed signal processing equipment.

To reduce the cost and the time of full-custom ASICs in most projects, a wide variety of design approaches have been developed to automate the design processes. These approaches progressively led to semi-custom ASICs.

**Semi-custom ASIC** Semi-custom designs are generally carried out at the gate-level. Gate-level modeling is more effective than full custom design in terms of design productivity, but at the cost of flexibility in the design method offered by the full customized ASIC method. Semi-custom solutions can be divided into two classes: gate arrays and standard cells.

**Gate-Arrays ASICs** are basically composed of continuous arrays of p- and n-type transistors. The silicon vendor provides the base wafers, to be then used according to the interconnection information provided by designers. Thus, designers may give the information that provides the interconnections data. Note here that the mapping from transistors to gates is performed through a CAD tool. It is possible to define two types of gate arrays: 1) the channeled and 2) the channelless (See Figure 2.10). In the channeled gate arrays, the interconnections are drawn within predefined channels between rows of logic cells. In the channelless gate arrays, there are no connection channels and the connections are drawn with upper metal layers, that is, on the top of the logic cells [16].

**Standard-Cell-Based ASICs** Standard cells are logic elements such as basic gates, multiplexers, adders, and flip-flops. These components are designed and stored in a library, so that each design may be composed of these pre-defined cells. A CAD tool automatically transforms the design into a chip layout. Standard-cell designs are typically organized on the chip, as rows of constant height cells

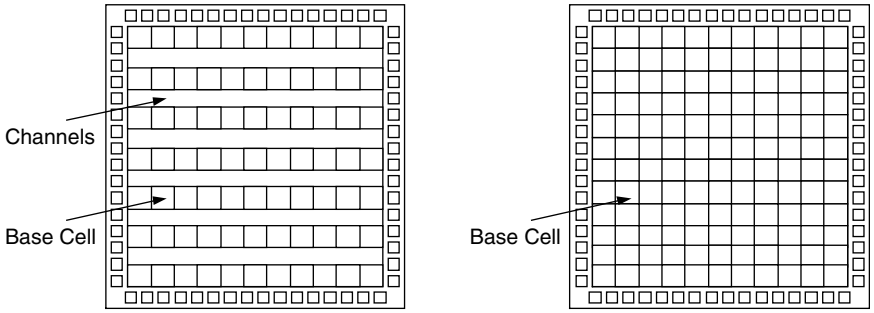



Figure 2.10 – Gate-Array ASICs [16]

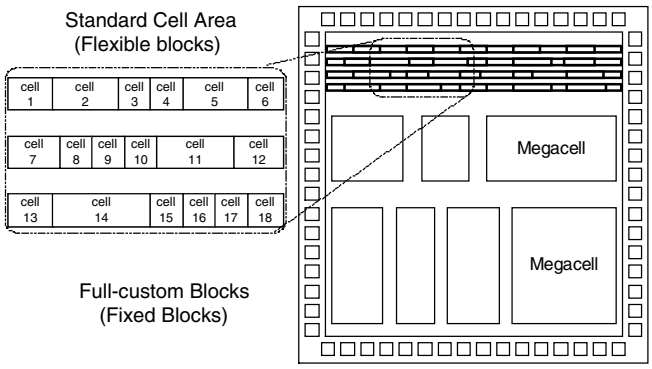



Figure 2.11 – Standard-Cell-Based ASICs [16]

(See Figure 2.11).

The design flow of semi-custom ASICs is very similar to the FPGAs flow. For instance, Figure 2.12 shows the different steps of the semi-custom ASIC design process. Note here that it is also possible to perform Post-synthesis and Post-P&R power verification using computer aided design tools. Regarding the power measurement phase, it is very difficult to measure the power consumption for each design configuration because of the high expense of the circuit tape out.

### 2.3.3 ASIC Vs FPGA

The main difference between FPGA and ASIC is that the final design is permanently encapsulated in the case of ASICs, while for FPGAs the design is implemented by configuring and interconnecting the various available resources, such as logic cells, routing nets and existing Digital Signal Processing (DSP) components. We may present the advantages for the ASICs compared to the FPGA counter.

Many benefits for ASICs at different levels, ASIC circuits produce high efficiency or also called processing speed, with low power consumption compared to

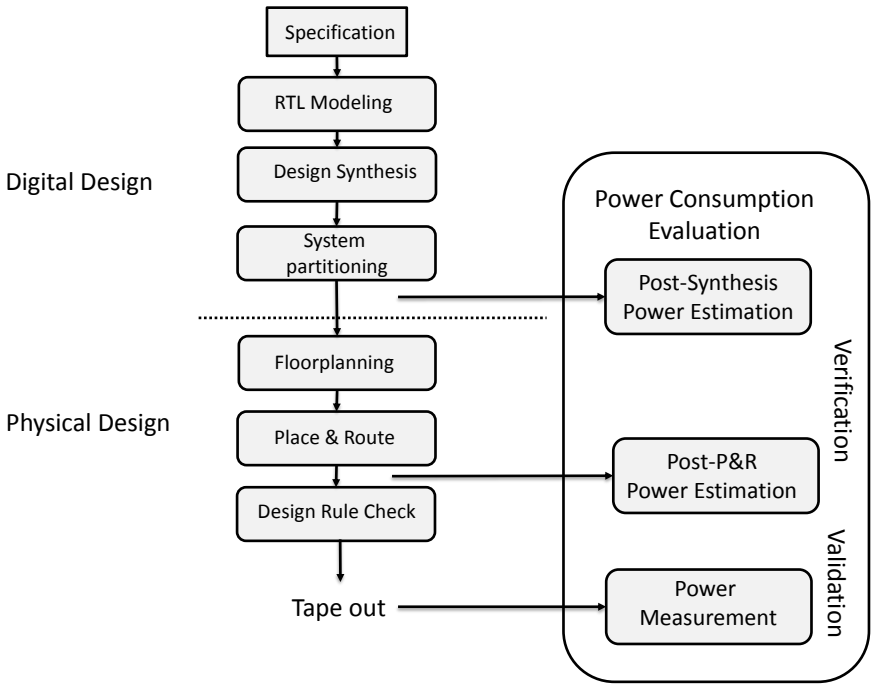



Figure 2.12 – Semicustom ASICs Design Flow

the reconfigurable hardware. It also provides a low cost solution relative to mass production, while ensuring a high level of design security.

However, many disadvantages that can also be identified. For instance, long turnaround times for silicone providers, at a very expensive cost, especially for low-volume manufacturing. It should be noted that a very time-consuming design process and a high amount of investment in engineering manpower and CAD tools are required to ensure a high quality product. Finally, the ASIC circuit is a fixed design that can not be modified once it is attached to the silicone. Moreover, Table 2.1 summarizes the main differences between FPGA and ASIC platforms. The rest of this chapter outlines the machine learning techniques used in power modeling in FPGA and ASIC platforms.

Table 2.1 – FPGA vs ASIC: A Comparison

| ***Aspects*** | **FPGAs** | **ASICs** |
|---|---|---|
| **Reconfigurability** | Reconfigurable circuit | Permanent circuit |
| **Design entry** | HDL: Verilog or Vhdl | HDL: Verilog or Vhdl |
| **Cost** | Low cost | Very expensive |
| **Mass production** | Low-volume | Very high-volume |
| **Power consumption** | Very-high | Very-low |
| **Operating frequency** | Slower | Very-high |
| **Upgrading** | Possible | Is not possible |
| **Analog/Digital designs** | Is not possible | Possible |
| **Time to market** | Faster | Longer |

## 2.4 A Machine Learning Overview

Machine learning is the science of algorithms and statistical models used by computer systems to perform a specific function, such as enabling computers to operate without specific instructions. It has provided us a much better knowledge of the human genome over the previous century with self-driving cars, speech recognition, and efficient internet searching. Machine learning techniques are usually based on the observations of a system behavior which may constitute the training examples. Then, learning algorithms use these examples to build a model, which helps in estimating the future actions or values [25]. Machine learning methods are very efficient in several problems, such as pattern recognition, prediction, control, and classification problems. These techniques are generally divided into two main categories according to their learning process (see Figure 2.13): 1) Unsupervised Learning and 2) Supervised Learning [26].

### 2.4.1 Unsupervised Learning

In this category, learning process involves finding hidden structures in the input data sets. Therefore, patterns should be discovered by the machine learning algorithms in its inputs. The importance of these algorithms appears when the scientists do not know what to extract from this data. In fact, the system is able to teach the user new things after it learns the patterns in the data. This type of algorithms are used to mine for rules and to group the data points (data are used as input) which may aid in obtaining a meaningful information. Clustering and Principal Component Analysis are two common unsupervised teaching methods.

### 2.4.2 Supervised Learning

Supervised learning is a machine learning technique that learns the behavior of a function based on its input-output data. The output of supervised learning

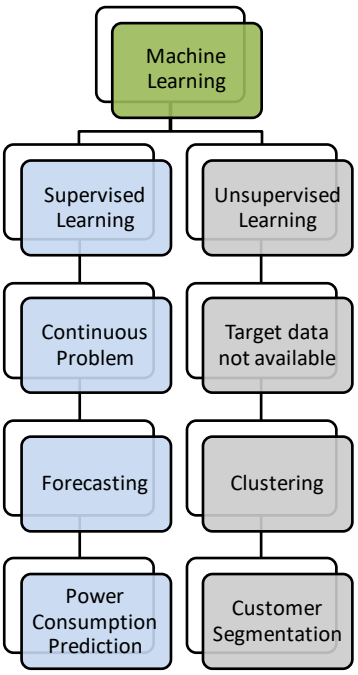



Figure 2.13 – Machine Learning Hierarchy

is a trained function, which maps an input to an output based on example input-output pairs or also called labeled data. Based on a set of training examples, this method is able to approximate a specific function, where an algorithm is trained in the back-end. Note that these data samples, which are provided in the form of a pair of data $(X, Y)$, allows the machine to learn the trend of the data output. Then, the best function $f$ which maps the input data $(X)$ to the one that for a given $X$ provides the good prediction of $Y$ $(f : X \rightarrow Y)$ is selected. Finally, in supervised learning, the algorithms try to model the relationships and dependencies between the output data (desired) $(Y)$ and the input $(X)$ data such that it can estimate the output values for new incoming data based on the trained model.

As an example, Artificial Neural Networks (ANNs) constitute supervised entities that are capable of learning from a set of examples. These networks consist of algorithms that can be used to perform non-linear statistical modeling and provide an efficient alternative to logistic regression, the most commonly used approach for developing predictive models. These networks have a lot of advantages, including requiring less formal statistical training, and having the ability to establish complex and non-linear relationships between input variables. Moreover, these techniques have been massively used in the past and have demonstrated their efficiency. For instance, artificial neural networks were used in [27] to approximate any continuous function.

From a conceptual point of view, ANNs are based on connected neurons, sim-

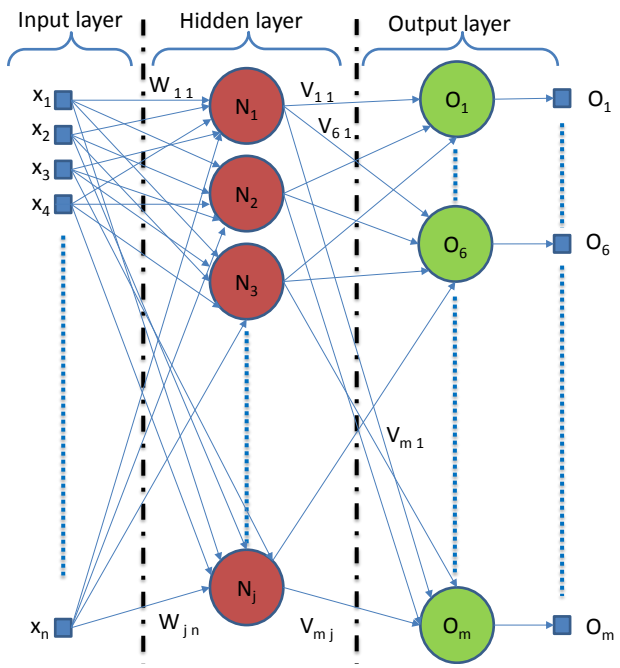



Figure 2.14 – Architecture of a Multi-Layer Perceptrons

ply like the ones present in the biological brain. More specifically, the connections between the neurons are similar to the synapses in the brain that are responsible for transmitting the signals from one neuron to another. These processing units or the so-called neurons, in their turn, operate in parallel to solve complex computational problems. In our work, we consider basic Multi-Layer Perceptrons (MLP) feed forward networks, in order to benefit from their capability to model complex behaviors and to perform multi-dimensional functions approximation [28]. Such networks generally have one or more hidden layers composed of neurons with non-linear transfer functions, and provide an output layer that implements the output neurons.

Figure 2.14 shows a typical architecture of an MLP neural network. Three layers have been considered: input, hidden and output layers. Each layer receives its inputs from the precedent layer and forwards its outputs to the subsequent one. In the forward phase, the hidden layer weight matrix is multiplied by the input vector $X = (x_1, x_2, x_3, \ldots, x_n)^T$ to constitute the input of each neuron in the hidden layer, as expressed in eq. 2.7:

$$z_k = \theta_k + \left( \sum_{i=1}^{i=n} w_{ki} x_i \right) \tag{2.7}$$

where $n$ is the number of inputs, $w_{ki}$ stands for the weight connecting input $i$ to unit $k$ in the hidden layer, and $\theta_k$ denotes an offset termed bias that is also

connected to each neuron in the hidden layer. To compute the output of each neuron unit in the hidden layer, eq. 2.8 can be followed:

$$N_k = \phi(z_k) \tag{2.8}$$

where $\phi(z_k) = \frac{1}{1+exp-z_k}$ represents the activation function used in the hidden layer, which is usually the sigmoid function. The input at each output neuron unit in the output layer is expressed as in equation 2.9.

$$p_k = \beta_k + \left( \sum_{i=1}^{i=j} V_{ki} N_i \right) \tag{2.9}$$

where $j$ is the number of neurons in the hidden layer, $V_{ki}$ denotes the weight connecting output $i$ to unit $k$ in the output layer layer, and $\beta_k$ is an offset termed bias that is also connected to each neuron in the output layer. Finally, the output of each unit in the output layer can be expressed as in equation 2.10.

$$O_k = \psi(p_k) \tag{2.10}$$

where $\psi(p_k) = p_k$ is a linear activation function used in the output layer.

Back-propagation algorithm such as the Levenberg-Marquardt [29] can be used during the training phase. This algorithm consists in presenting a set of examples to the neural network as well as a set of corresponding targets. During an iterative process, errors between the actual output and the desired target are back-propagated and all the weights in the network are modified according to their contributions to the error. The training process is generally stopped after cross-validation on another set of data denoted as test set. This latter is primarily used to estimate the ability of an ANN to perform on unseen data.

# Chapter 3

# Power Measures, Estimates and Modeling Methods: A Survey

## 3.1 Introduction

In this chapter, we survey the existing works that propose power measurement, power estimation techniques as well as modeling methods. We intentionally focused on FPGA circuits, but we also considered ASICs because most power estimation and modeling methods were originally developed for these circuits. Our review aims to help digital designers identify the most suitable technique for estimating the power consumption of their FPGA or ASIC designs. Up-to-date studies, including power measurement techniques, commercial power estimation tools and power modeling methods, are being evaluated.

Furthermore, Figure 3.1 presents the structure of the chapter. It is possible to assess the power consumption either by measurement methods (See section 3.2) or by estimation. Section 3.2 consists of two solutions, the on-board and the external. Power estimation can be performed either using the computer aided-design tools (See section 3.3) or the proposed modeling methods (See section 3.4). Regarding computer aided-design tools, are based on 3 estimation techniques (probabilistic, simulation or statistical-based). Concerning the modeling methods, 4 modeling techniques (analytical, table-based, regression, and neural-networks) are discussed. Additionally, a way of methods comparison according to specified metrics are analyzed in section 3.4.5. Finally, we cluster all the works under different clusters, which are labeled according to their analyzed characteristics.

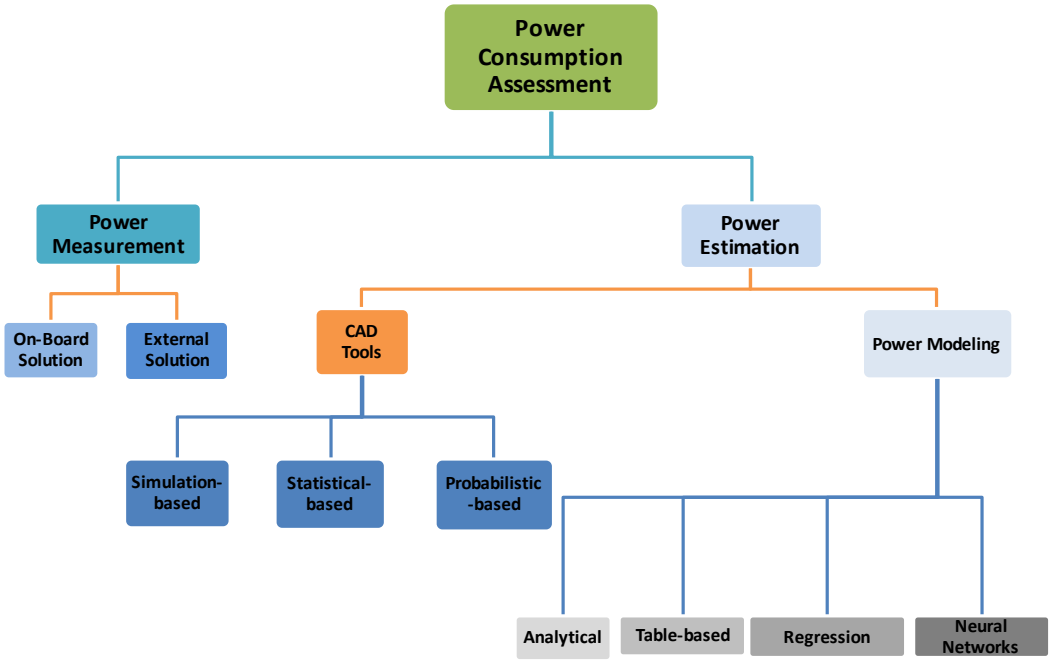



Figure 3.1 – Power Consumption Evaluation Taxonomy

## 3.2 Power Measurement

The most intuitive way to evaluate power of devices consists in performing real measurements on the circuit directly. However, this requires to perform all design steps before obtaining any power consumption profiles [30]. Either on-board may be used to monitor metrics such as the supply voltage and the current drawn by the FPGA or ASIC circuit. An external instrumentation setup has then be defined to properly evaluate power consumption.

### 3.2.1 On-board Solution

The on-board power measurement solution is a portable solution that can help the digital designers to evaluate their designs as shown in Figure 3.2. This can be achieved using the available on-board components, like current sensors, and voltage regulators. In the following, we review the existing works that are related to this solution.

Recently, development boards that permit to perform voltage and current measurements at specific circuits locations are released. FPGAs manufacturers like Xilinx and Intel offered such solutions for real power measurements. For instance, Xilinx provided power measurement solution with Texas Instruments (TI) that helps designers in providing power measurements on Xilinx boards [31].

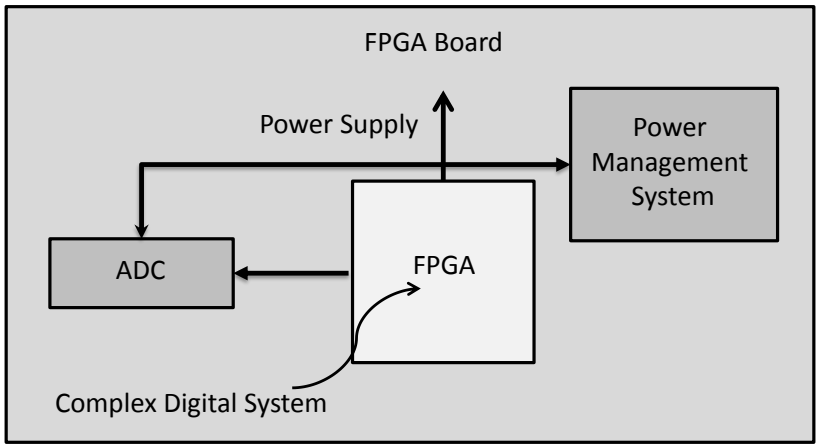



Figure 3.2 – On-board Power Measurement System

This solution was based on several parameters: 1) the built-in current sensors, 2) the power regulators with a JTAG to USB adapter that connects the board to the host PC to monitor the current, and 3) the supply voltage on the FPGA boards. However, this solution was considered a limited one due to some hardware noise caused by the built-in regulators and sensors, and to the on-board Analog to Digital Converters (ADCs) which were limited to a fixed sampling frequency and a low bit resolution. Knowing that these constraints depend on the manufacturer, they are hence not accessible to the designers. This leads to real problems in terms of measurement accuracy and measurement bench flexibility.

For example, authors in [32] presented an investigation about the power consumption overhead of the Dynamic Partial Reconfiguration (DPR) application. They relied on the on-board current sensors to measure the power consumption, and used the power regulators to supply the FPGA core. Additionally, on-board power solution is used in [33]. In this work, authors also measure the power consumption during the DPR. Some limitations can be seen such as the limited sampling frequency of the on-board's ADC. This, in turn, limits the data acquisition at the right speed while DPR is being carried out in a very fast way.

Knowing that Xilinx's on-board power measurement solution is not precise, and a real benchmark that allows designers to access all FPGA pins (with an option to tune and tweak parameters) could be considered as an alternative solution. This alternative is more suitable and can deliver more flexible power measurement solution. Therefore, an external solution can provide more accurate and flexible power measurement, which may help to efficiently characterize and model the power consumption in digital circuits.

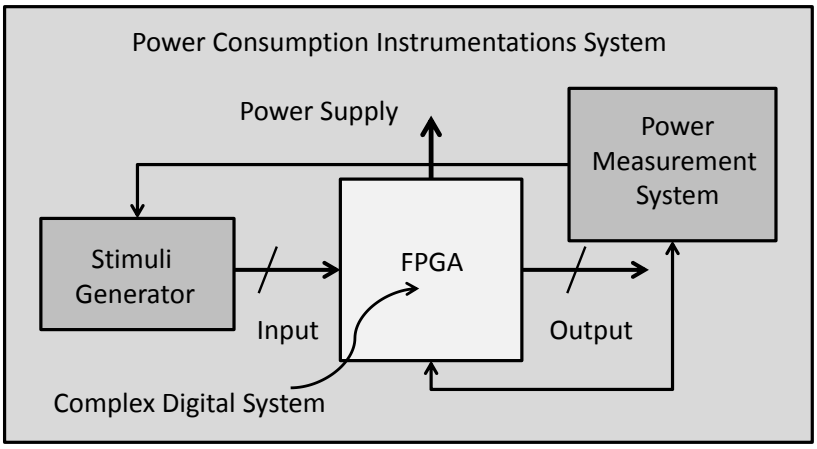



Figure 3.3 – External Power Measurement System

### 3.2.2 External Solution

To obtain more accurate power measures, an external power measurement setup is often considered as depicted in Figure 3.3. It can be constructed using an external stable low noise power supply to supply the FPGA core power pin, an external data acquisition board to acquire the current across a shunt resistor connected in series with the FPGA core, a control board that sends data and control signals to the inputs of the digital component implemented in the FPGA. In this case, power measures represent the amount of power consumed by the digital component implemented in the FPGA with no additional contribution of other components. To this end, this solution should help in characterizing the power consumption of a specific hardware design. Unfortunately, the existing power measurement instrumentation measures the power consumption for the entire system and not for each hardware component by itself [34]. For this reason, researchers start to build some power measurement benchmarks to characterize the power consumption of digital components [35]. In the following, we discuss the existing benchmarks that deal with power measurement in FPGAs and we underline their limitations.

In the existing works of [36], [35], and [37], the authors presented many power measurement platforms to measure the power consumption in FPGAs. The authors in [36] proposed a methodology based on real measurements, in order to let the designers model the application power consumption for different architectural and algorithmic parameters. However, the details about instrumentation and the associated methodology is missing in this work.

In the works described in [35], the authors present another measurement methodology and study power characterization to allow the separation of distinct power groups in FPGAs (such as interconnect power, logic power and clock

power). They measure the resulting voltage when 10000 distinct vectors were applied to the inputs of a module. Unfortunately, due to the limited memory size of the FPGA platform which are used as a data source, this power consumption was measured during a small time window. However, a bigger time window can assist designers test a wider time window. This can be achieved by using a larger input sequence length (more than 10000) that enables more input vector combinations. A larger time window therefore helps to have more precise power values. Having a big amount of information points leads to a more accurate average value.

The authors in [37] propose a system that is based on a current-frequency conversion block that measures the power consumption of a running application in real-time. However, the authors present the power measurement methodology without mentioning the different types of noise that can be induced by the different electronic blocks and their effects on the accuracy and the precision of the physical measurements.

Finally, external power measurement may help to accurately characterize the power consumption of digital components and designs. For example, in [38], reconfiguration requests are managed by a Microblaze processor which downloads bitstreams into Partial Reconfiguration Regions (PRR). The measured power consumption corresponds to both power consumed by the softcore and the hardware components. However, the authors aim to only measure the power of the hardware accelerators. For instance, it is also possible to use a hard-core to control DPR without measuring the power of the softcore as proposed in [33] while taking into consideration the external power measurement solution. This, in turn, provides a more suitable scenario, and can help designers to only evaluate the power consumption in the programmable logic part only.

### 3.2.3 Summary

To summarize, the different measurement methods reviewed in this section demonstrate that the power measurement could only be performed at the late design point. These power measurement techniques are based on 1) on-board solution and 2) external solution. The on-board solution can provide a quick solution to measure power consumption, but with less accurate results. On the other hand, external solutions can provide more accurate results but are limited to: 1) the expensive tools, 2) the need for hardware instrumentation, and 3) the long setup time. The power modeling as well as the estimation methods are reviewed for this purpose. Remember that the power measurement provides the accurate way to obtain the real power consumption of a given circuit or design.

## 3.3 Power Estimation Techniques

Power estimation techniques are very efficient alternatives to measurement-based methods. They are also very practical to perform design exploration. In fact, such techniques can be considered at different levels of abstraction of the design. Here we review the power estimation methods, and we can describe a power estimation technique as noted in the literature as follows: it is the method used to characterize the power consumption for a specified design, and three different methods can be identified: 1) simulation-based, 2) statistical-based , and 3) probabilistic-based.

### 3.3.1 Simulation-based

Simulation-based power estimation is used by most of the computer-aided design tools. This type of technique consists in applying data stimuli to the design inputs of a digital design and perform a simulation to determine the corresponding outputs. Depending on the abstraction level, the type of information that is required to obtain power estimation is different, going from current and voltage values, capacitance, frequency to the switching activities of all signals.

Circuit simulators such as SPICE [39] use big matrix solutions of Kirchhoff current law (KCL) equations to determine nodal currents at transistor level. Basic elements such as resistors, capacitors, inductors, current sources, voltage sources, and higher-level diode and transistor device designs are used to correctly predict the current and voltage drop. Even the non-linear capacitance that is present at transistor nodes may be simulated. Although highly precise, these tools become quickly unpractical as the size of the circuits increases.

Another transistor level simulator called PowerMill [40] uses linear piece-by-piece transistor modeling to store transistor characteristics in lookup tables. It also uses an event-driven timing algorithm to attain speeds comparable to logic simulators. The difference with other approaches is that it does not consider logic transitions but rather changes in node voltages. Using lookup tables leads to inaccuracy, but results are provided of 2 to 3 times faster compared to SPICE.

Gate level simulation includes the use of logic parts such as NAND / NOR gates, latches, flip flops and interconnection networks. The most popular technique of assessment includes an event-driven model [41]. When an event occurs at a gate input, it may generate an output event after a simulated time delay. Power consumption is predicted by computing the charging/discharging capacitance at the gate and by evaluating the activity of this node. However, each gate is described as a black box and the inner system is not modeled. For example, short circuit power and inner capacitance are not taken into account [42].

Cell-based power estimation techniques adopt comparable methods in which cell libraries are distinguished by electrical (SPICE-level) modeling for all feasible input configurations and fan-in / fan-out options [43]. Logic simulation

utilizes these data to predict power. The accuracy relies on the precision of the capacitance given at cell unit [44].

The main advantages of simulation-based power estimation techniques are their precision and their genericity. This technique demonstrates a very good accuracy and it is easy to be deployed on any circuit, regardless of its implementation details like the technology used or the architect of the design. However, this method may also have some drawbacks such as: 1) it needs a large amount of memory resources to store all signals' information, 2) it requires a very long time to simulate the circuit (for instance, complex circuits may need hours or may be days to be simulated in order to estimate the power consumption), 3) it is a size-dependant technique, i.e., the estimation depends on the simulated circuit (number of gates, inputs, outputs etc.).

To counteract these drawbacks, authors in [45], [46], [47], [48] and [49] proposed methods to accelerate simulation by applying statistical sampling methods. For instance, authors in [49] demonstrate a new statistical sampling technique which applies a stratified random sampling to improve the sampling efficiency. Results show that these sampling techniques are 3-10 times faster than that of Markov-based Monte Carlo simulation methods presented in [45]. In the sequel, we review the works dealing with statistical-based power estimation techniques. These are considered as an accelerated approach of the simulation-based techniques.

### 3.3.2 Statistical-based

The statistical power estimation techniques are used to get the power consumption for a given design, after waiting for the convergence of the estimated power values to a desired average power. Statistical methods use randomly generated input stimulus, which are applied to the primary inputs of a given circuit. Then, the design is simulated using the power simulator, which waits for a desired precision to be achieved. As an example, the Monte Carlo analysis is also considered as a statistical based power estimation, where the design is simulated iteratively, and the power values are monitored, while using several statistical average estimation methods.

Authors in [50] present McPower, a Monte Carlo power estimator, where the design is simulated iteratively, and the simulation is stopped when sufficient precision is achieved with a specific confidence. Note that this technique is also time consuming but in comparison with simulation-based methods, it provides faster solution.

Letting statistical estimates converge for all gates would be inefficient, especially as it would take a large number of vectors to converge for those nodes that switch rarely. In this respect, authors introduce an effective statistical sampling technique in [51] which classifies the inner nodes into two classifications: 1) nodes with high transition densities and 2) nodes with low transition densities. This

allows the designers to define the amount of error tolerance and it is also possible to recognize nodes with minimal tolerance. Note that nodes with low transition densities dissipate a small amount of power, resulting in a low effect on the precision of the estimation. The technique has been enhanced upon [52] by using distinct error values for distinct nodes, unlike using a steady designer-defined estimation error. For nodes with greater switching, error levels are thus determined in such a manner as to evaluate them more correctly.

Other works of a statistical type include estimating power in input vector requirements under uncertainties. Since power is extremely conditional on the vector models implemented, any uncertainties in their requirements can complicate the estimation method. This issue was discussed by understanding average power as a scope (minimum average power, maximum average power) and statistically estimate the sensitivity of average power consumption with respect to the uncertainties of the input vectors [53].

### 3.3.3 Probabilistic-based

The probabilistic power estimation method relies on the static and transition probabilities of the data stimuli. It uses the data stimuli characteristics instead of the actual data stimuli as metrics depicting data signals information. On one side, the static probability of a signal $s_i$ denotes by $P(s_i)$ can be described as the probability of this signal having a logic high or 1. On the other side, the $TP(s_i)$ transition probability of a signal represents the probability that this signal will change its state from high to low logic or vice versa. In this regard, probabilistic power estimation method exploits these two metrics to calculate the power consumption of a given digital design. It also propagates these metrics throughout the nodes and gates within a given circuit to get a global power estimation of a digital circuit.

Probabilistic-based power estimation technique is defined by a good estimation speed, which makes it quicker than the technique based on simulation-based or statistical. Although this provides a good estimation speed, the accuracy of the estimation is also essential and has a significant impact on the selection of the design choice, particularly since the power budget is a design constraint.

Authors in [54] and [55] propose additional metrics to be taken into account in order to improve the accuracy of the probabilistic techniques, such as 1) the temporal signal correlations and 2) the spacial signal correlations. The signal spacial correlations take place when the bit value of an input depends on the other input bit. As for the signal temporal correlation, it occurs when the bit value of an input bit depends on the previous bit value for the same input. The authors in [56] show that the signal correlations may affect the estimation results. For instance, neglecting the temporal correlation increases the error from 15% to 50%, and neglecting the spatial correlation degrades also the error from 8% to 120%.

Note that the power estimation concentrates on the propagation of transition density and static probability. In this regard, the techniques discussed above assume that at the same time there is no more than one transition, in other words the glitch is not taken into consideration. They also used only deterministic delay models, meaning that the transition density of the data signals consider fixed delay models of the gates. However, delay fluctuations and uncertainties have become very significant concerns with the scaling of the technology and have a significant impact on power consumption.

To counteract this issue, authors in [57] suggested an improvement to propagate the transition density of data signals and to take into account the uncertainty of the delay models. The probability of the data signals is then described as a continuous function of time, which provides more accurate results compared to the fixed delay models

To summarize, all the mentioned estimation methods such as simulation-based, statistical-based, and probabilistic-based are incorporated in most Computer Aided Design (CAD) tools. For this purpose, the power estimation tools are discussed in the next section.

### 3.3.4 Power Estimation Tools

We evaluated the power estimation methods, and here we discuss the existing tools that are built on these methods. For instance, Synopsys offers tools such as PrimePower [58] in order to accurately analyze the power of a full-chip of cell-based designs throughout the design implementation process. Note that the power analysis is performed from early estimation to implementation (sign-off).

Cadence delivers Genus as power estimation tool at Register Transfer Level (RTL) and gate-level [59]. Genus RTL power solution provides time-based RTL power profile with system-level runtimes and capacity, along with high-quality estimates of gates and wires. Note that both tools offer vector-free or also called probabilistic-based and vector-based peak power and average power analysis or also called simulation-based. Note here that the vectors used are either RTL or gate-level simulation results in the Value Change Dump (VCD) format or Switching Activity Interchange Format (SAIF). In this regard, power estimation is based on a detailed power profile of the design based, which takes the circuit connectivity, the switching activity, the net capacitance, and the cell-level power behavior data in tools database format(.db) library. It is good to mention here that these tools also supports Nonlinear Power Model (NLPM) libraries, which calculates the power behavior for a circuit at the cell-level and reports the power consumption at the chip, block, and cell levels.

In FPGAs, power estimation is usually evaluated using spreadsheets which are usually specific to a device. An example of a vendor's tools is the Xilinx Power Estimator (XPE) [60]. These tools typically aim at providing power and thermal estimates at an early stage of the design flow. Moreover, Xilinx presents Xpower

analyzer [61] to analyze the power consumption at different level of abstractions. Vector-free and vector-based power estimation is also supported. In parallel, Intel provides PowerPlay [62] as power analysis features, including early power estimators and the Intel Quartus Prime software power analyzer that give the opportunity to estimate power consumption from early design stage up to design implementation.

A typical power estimation flow requires simulation at RTL using a simulator and the output generally consists of a (VCD) file to be provided to the power estimation tool. If vectors are not available, switching activities may be assigned to the inputs and propagated using an activity estimator such as ACE [63]. In addition, academic FPGA tool flows such as VTR also has a power estimator (VersaPower) [64] which relies on activities generated by ACE to perform power estimation.

Recently, a SPICE based power estimation tool called FPGA-SPICE which is integrated with the VPR framework was developed by [65]. This tool can run SPICE level simulations for a given design mapped to the FPGA and provides cell level power values or total full chip power as specified by the user. The power values obtained using FPGA SPICE are more accurate as compared to VersaPower but the runtime is significantly longer and it is not scalable for large designs.

### 3.3.5 Summary

In this section, we present a summary about the different estimation techniques, which are used to estimate the power consumption in digital circuits. To this end, Table 3.1 classifies the state of the art and discusses their advantages and limitations. First, simulation-based techniques present 2 important advantages such as : 1) the high accuracy and 2) the generality. As for the accuracy, this technique performs the estimation at very low level of abstraction, where current waveforms is computed, and with the knowledge of the supply voltage,

Table 3.1 – Estimation techniques advantages and limitations

| **Estimation Techniques** | **State of the art** | **Advantages** | **Constraints** |
|---|---|---|---|
| Simulation-based | [39], [40], [41], [43] | 1) high accuracy;<br>2) generic. | 1) large amount of memory resources;<br>2) low estimation speed. |
| Probabilistic-based | [54], [66]<br>[48], [67]<br>[68], [55]<br>[56] | high estimation speed. | low accuracy. |
| Statistical-based | [50], [51]<br>[52], [53] | moderate accuracy. | moderate estimation speed. |

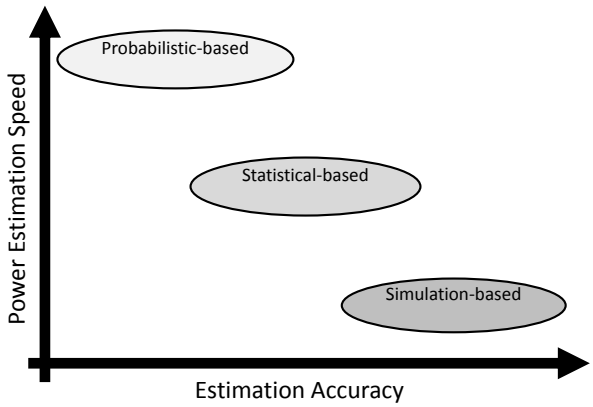



Figure 3.4 – Power Estimation Methods Trade-off

the average power dissipation is calculated [69]. Finally, as for the generality, this technique can be applied to any kind of circuits [70]. However, time consuming simulation is an issue of such kind of technique, where the power estimation needs to wait until all current waveforms at all the nodes to be generated in order to perform the power calculation. In addition to this, memory resources are also violated.

As for the probabilistic-based, this technique delivers a high level of estimation speed as well as it does not need to wait for waveforms generation. However, signal and transition probabilities are used to estimate the power consumption. Hence, low accurate results are obtained due to the use of simplified delay models for the circuit components to ease the probabilistic analysis [69].

Statistical-based power estimation technique incorporates the accuracy of the simulation-based and the estimation speed of the probabilistic-based techniques [45]. For this purpose the estimation speed and accuracy can be considered as moderate.

Finally, Figure 3.4 categorizes the estimation techniques with respect to the most important factors (accuracy and estimation time). To conclude, the aforementioned techniques provide power estimations that can be performed using one of these techniques (probabilistic-based, simulation-based, or statistical-based). In addition to this, advantages and limitations are discussed, and to overcome the limitations such as the **accuracy** and **estimation speed**, modeling techniques are proposed.

## 3.4 Power Modeling Techniques

In this section, we analyze the power modeling methods tailored to the VLSI circuits, namely the (FPGAs) and (ASICs). Usually these methods are used to abstract the power consumption of a digital component, digital design, or

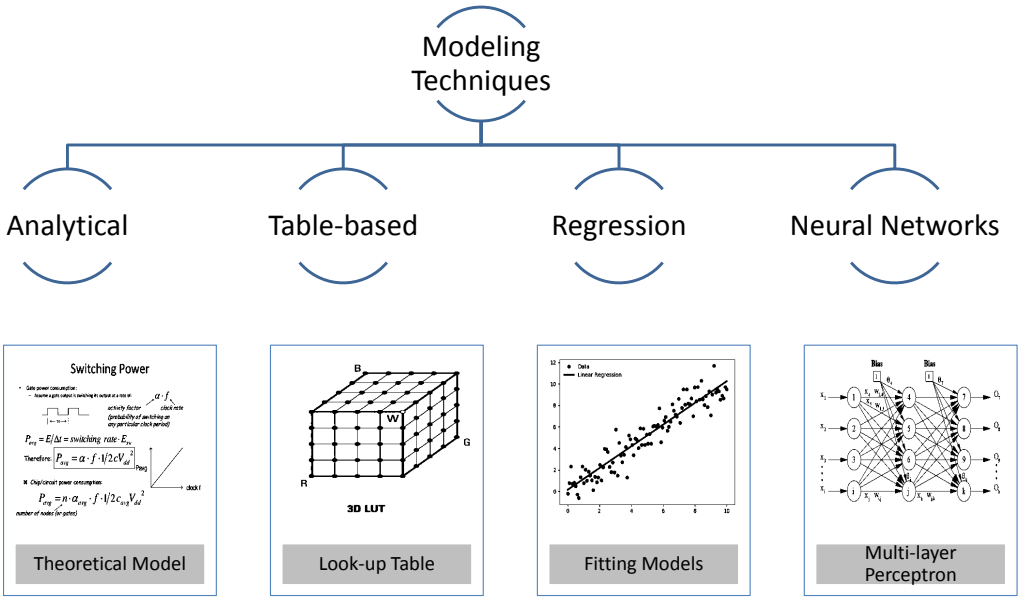



Figure 3.5 – Modeling Techniques Literature Review

electronic device. Here we clarify the fundamental concepts of these methods that produce the power models used by designers and scientists to assess the power consumption of different hardware designs at a high-level of abstraction.

The purpose of these models is to help explore many architectural design decisions. These models can be built from either power characterization and modeling methods or analytical models for a given architecture. Note that power characterization is the process of extracting power consumption data with respect to different simulation parameters. However, the modeling techniques use this resulting data as input to construct an abstract power model.

In the following sections, after classifying them into four categories, we review each modeling technique separately as observed in the literature. These categories are illustrated in Figure 3.5 and may be defined as follows:

— Analytical techniques

— Table-based techniques

— Regression-based techniques

— Neural Networks based techniques

In order to compare the different works for each modeling technique, we define the following four *metrics*:

1. **Modeling level:** It represents the level at which the model is intended to be used. More specifically, a power model can model power consumption for either a *circuit* or a *component* in a circuit. For instance, modeling

the power consumption of a multiply-and-accumulate (MAC), which is a component in a digital filter circuit, is considered as a component level modeling;

2. **Modeling effort**: It evaluates the effort needed to build the model. We propose three levels for this metric: low, moderate and high, represented as $*$, $**$, and $***$ respectively. A high modeling effort could require multiple iterations of the design flow and some low-level characterization while a low modeling effort could necessitate only few simulation runs with little information to build the power model;
3. **Model granularity:** It reflects the abstraction level of the parameters used to build the power model. Two granularity models can be defined here: *fine grain* model and *coarse grain* model. The first appears when the predictors used to estimate power consumption carry very low-level information on each bit of the data signals, such as the transition rate and the static probability. The second, however, arises when the predictors used do not consider low-level parameters, but high-level information such as the number of bits and operating frequencies;
4. **Estimation technique:** Characterization-based power modeling methods are based on estimated power values. These values can be obtained on the basis of the three estimation techniques as described above and is shown as follows: 1) *simulation-based*, 2) *statistical-based* and 3) *probabilistic-based*. Note that we can model the power consumption analytically based on a well defined design architecture without characterizing the power a priory. Consequently, analytical modeling methods are not considered in the classification. However, table-based, regression-based, and neural networks based on power characterization are considered due to the need for power characterization.

   According to these metrics, a classification is presented in the following sections.

### 3.4.1 Analytical techniques

In analytical techniques, the power models are created without any power characterization phase. In fact, analytical approaches attempt to relate the power consumption of a design to fundamental quantities that describe the switching activity and the capacitance of a design [71]. More specifically, this technique consists of developing power models, which are based on the theoretical equation of the power dissipation for a CMOS transistor indicated as in eq. 3.1:

$$P_{dynamic} = \alpha C_L V_{dd}^2 f_{clock} \tag{3.1}$$

where $\alpha$ represents the activity factor, $C_L$ stands for the load capacitance, $V_{dd}$ denotes the supply voltage and $f_{clock}$ is the clock frequency.

Analytical power models were initially created for simple combinational circuits, and more specifically for targeting ASIC devices. They were later extended to be applied to sequential and more complex designs and to FPGAs. The definition of the analytical modeling technique is also inspired from [72], where they go further by sub-dividing this technique into activity-based and complexity-based ones. First, the activity-based models analytically address the power modeling issue, where the entropy concept is borrowed from information theory in order to use it as a measure of the average activity in a circuit. This is to try to relate between the power consumption of a functional block and the amount of computational work it performs. Note that this technique does not consider the timing information, which is a very important metric that should enter into the above calculations, and hence it does not take into account the glitching power. Second, as for the complexity-based technique, it gathers as an example the techniques that roughly estimate capacity through the complexity of the design in terms of logic gates. The main drawback of this technique is that it does not consider the data pattern at the input, given that the data pattern affects the average number of transitions per clock cycle. In other words, the $\alpha$ parameter of eq. 3.1 is not taken into account. Moreover, to estimate the total capacitance $C_L$ of a design, analytical models can be developed based on the Rent's rule [73]. Note that Rent's rule concerns the organization of computing logic, specifically the relationship between the number of external signal connections to a logic block and the number of logic gates in the logic block. When going up in abstraction, i.e., at the gate-level, another solution consists of evaluating the hardware complexity through the number of equivalent gates used in a design [72]. It is then possible to estimate power if the average power consumption of an equivalent gate (e.g., 2-input NAND) is known. However, such techniques ([73] and [72]) do not always consider the switching activity of the block, leading to a low accuracy. However, they are very useful when comparing several design options, while requiring only few information to obtain an estimation of power consumption.

Recall that the power consumption of a circuit heavily depends on the primary input probabilities and activities. Thus, some techniques tried to model the relationship between the power consumption of a hardware block and the entropy of its inputs/outputs [74], i.e., the amount of information processed by this block. In [74], a relationship between the capacitance and the activity for estimating the power was proposed, for which the area was also used as a metric for estimating the physical capacitance. Probabilistic techniques are usually preferred for the switching activity estimation because of their computational efficiency as compared to the simulation-based. In [75, 51], the Transition Density Model (TDM) was proposed to estimate the switching activity by considering the number of transitions per second at a node.

It was demonstrated in [76] that spatial and temporal correlations should be taken into account when estimating the switching activity using probabilities in order to improve this accuracy. While the original TDM was primary applied

for small combination circuits and ASICs, it was extended to sequential circuits, by considering signal statistics such as the transition probabilities to model the transition densities of outputs. Based on this, word-level signal statistics were used to the model power consumption of several operators e.g., adders, while delivering an accuracy of around 10% against a Xilinx power estimation tool called XPA [77]. Another limitation of TDM is that it does not consider glitches, which have a significant impact on the power.

Another interesting approach was proposed in [78, 79], in which an effective power model based on Markov chains was used to accurately estimate the power sensitivity to the primary inputs for both the combinational as well as the sequential CMOS circuits. This power sensitivity represents the changes of power consumption induced by signal inputs. Average error of less than 5% has been achieved in this work.

A similar approach to [77] but targeting FPGA was proposed in [80] in order to estimate the dynamic power of the dividers based on signal statistics. In comparison to ASICs, FPGAs own specific reconfigurable routing resources built on MUX and pass-transistors. Thus, some works focused on the power modeling of these elements by using the equations which are related to the charge/discharge capacitance [81]. Their fine-grain models achieve an accuracy of about 5%.

In addition, the analytical model proposed in [82] allows to evaluate the area-efficiency and logic depth of designs implemented on FPGAs by determining the relationship between the logic blocks and the cluster parameters. By combining such models with a delay model, this approach can be used to quickly evaluate a wide variety of lookup-table/cluster architectures, but still without taking the power into consideration.

So far, many approaches have focused on the dynamic power modeling, which has represented the main power dissipation source for the last decades. Authors in [83] presented additional models of short-circuit and leakage powers for FPGA devices. Whereas switching activity is estimated using transition density of every nodes, dynamic power can be estimated as follows:

$$P_{dynamic} = \frac{1}{2} \sum_{all_nodes} C_y.V_{supply}.V_{swing}.D(y).f_{clock}$$

where $V_{swing}$ denotes the swing voltage of each node, $V_{supply}$ represents the supply voltage, $D(y)$ stands for the transition density at node $y$, and $C_y$ is the capacitance of node $y$ representing the energy per clock cycle relative to a clock frequency. Their models achieved an error of 4.8% for routing segment up to 20% for other resources.

More recently, analytical technique models area, delay and power, allowing both static and dynamic power evaluation during design exploration [84, 85]. This allows designers to explore the FPGA architecture parameters, including the Configurable Logic Block (CLB) number and the associated switch boxes,

Table 3.2 – Classification of Main Analytical Modeling Techniques

| References | Metrics | | | | | | |
|---|---|---|---|---|---|---|---|
| | Inputs | Outputs | Modeling level | Error (%) | Modeling effort | Model granularity | Year |
| [77] | $r_{xx0}$, $r_{yy0}$, $r_{xx1}$, $r_{xy1}$, $r_{yx1}$, $r_{yy1}$ | Power | Component | 10%(Avg) | *** | Fine | 2005 |
| [83] | $C(y)$,$D(y)$,$f$, $Vsupply$,$Vswing$ | Power | Circuit | 5%(Avg) | ** | Fine | 2005 |
| [81] | $C$,$Vmax_{in}$,$Vmax_{out}$, $\tau$ | Power | Component | 5%(Avg) | ** | Fine | 2012 |
| [84] | $F_{C_{in}}$,$F_{C_{out}}$,$F_s$,L,W,I,K | Static P. | Component | 15%(Avg) | ** | Fine | 2013 |

wire lengths, and clock frequency. For instance, the authors in [84] improved the Poon's model (See [83]) accuracy by integrating the FPGA channel width ($W$) and length models ($L$), which delivered a static energy model that takes into account logic and routing architecture parameters. Other architectural parameters such as the I/O connection-box ($F_{C_{in}}$, $F_{C_{out}}$) and the switch-box flexibility ($F_s$) were also considered in the model as well as the logic parameters, such as the LUT size (K), the cluster size (N), and the number of inputs per cluster (I). A major advantage of this approach is that it ensures a CAD-independent static estimation while achieving a satisfying level of accuracy, i.e., a mean percentage error of 15%. Another approach for modeling the static and the dynamic power is the one presented in [85]. However, it differs from [84] by being formulated using Geometric Programming, which is a kind of mathematical optimization problem that is based on objective and constraint functions, recently applied to circuit design issue.

As a summary, Table 3.2 presents the main aforementioned *analytical models* along with their corresponding inputs that allow the power estimation. The error and **modeling effort** are also indicated as well as the **model granularity** and the **modeling level**.

Many **advantages** can be offered by the analytical power models, which achieve a relatively good accuracy against low-level simulation tools, especially when spatial and temporal correlation are taken into account in the model. However, most of the approaches model the power for the component only, without considering the additional connections that are required when interconnecting multiple components. Since analytical power models are generally used at high-level of abstraction, they make it possible to obtain results very fast, and to enable fast design exploration, especially if the number of hardware resources is a parameter of the power model. Regarding FPGA or even ASIC, technology keeps on evolving by modifying the size of the LUT, the CLB or even the LE architecture, making the generalization of the power model for different technologies very difficult.

However, the analytical power modeling technique presents also many **disadvantages**. More specifically, it is very hard to take into account the effect of glitches with this kind of power model. This is of further importance when the model is developed for a component (and not for an entire circuit), destined

to be connected to other elements. Additionally, analytical power modeling are only suitable for behavioral blocks that are organized in a regular way such as cache, queues, registers and buffers [71]. Moreover, it is complicated to analytically derive the power consumption of more complex digital systems, and to take low level information into consideration. Here appears the importance of adopting new techniques, which are taking into accounts low level information based on power characterization, like the approaches that we detail in the following sections.

### 3.4.2 Table-based power modeling technique

*Look-up table* or *Table-based* power modeling technique, is the tabulation process of the power values [4]. For example, this technique is illustrated in Table 3.3. To access the power values ($P_1...P_n$), such as input transition density ($x_1...x_n$), static probability ($y_1...y_n$), and spatio-temporal correlations ($z_1...z_n$) are used. These power values have generally been obtained after a power characterization process that consists in modifying the model's input parameters. In addition, an interpolation method is also used to estimate the missing power values in the table. Note here that it does not require any mathematical model compared to the analytical approach. In this regard, look-up table based power modeling technique gained a lot of attention from researchers.

Authors in [4] presented a modeling technique to estimate the switching activity and the power consumption of components at register-transfer level (RTL), and more specifically for data-path and the control logic. In this work, the glitches of different signals are taken into consideration, in order to increase the accuracy of the estimated power. This is done by using piecewise linear models that take the fluctuation of output glitching activity and power consumption with various word-level parameters like mean, standard deviation, spatial and temporal correlations, and glitching activity at the component's inputs. To create the glitch models, the authors used analytical models as well as look-up tables for the power models. They also considered more than six factors as input entries that corre-

Table 3.3 – A example of a 3D look-up table based power modeling method

| Metrics | | | Power Consumption |
|---|---|---|---|
| Input Entry 1 | Input Entry 2 | Input Entry 3 | |
| x1 | y1 | z1 | P1 |
| x2 | y2 | z2 | P2 |
| x3 | y3 | z3 | P3 |
| ... | ... | ... | ... |
| xn | yn | zn | Pn |

spond to the present and previous input values of the component (see Table 3.4). The authors claimed that the obtained power estimation accuracy is around 7%.

Authors in [5] presented a modeling method that evaluates the power consumption of a combinational circuit. This method quantifies the effect of I/O signals switching activity. The studied parameters are the average input signal probability, the average input transition density and the average output zero-delay transition density. They use the resulting power values to build a three dimensional look-up table according to the studied parameters in order to estimate power for any given I/O signal statistics. This method has been implemented and verified for many benchmark circuits. This method achieved an accuracy around 6%. Note that on the contrary to [4], this work deals with a table dimension which is independent of the number of inputs component. This constitutes an advantage to reduce the table dimension.

Moreover, authors in [6] analyzed the work proposed in [5], which is related to the look-up table based methodology (that can be used in the behavioral simulation), and recognized its drawbacks. They suggested some enhancements such as: 1) proposing a solution based on the interpolation method, which can be helpful if there is no entry to be discovered in the look-up table. Note that this method is based on the two closest neighboring entries of the missed entry value, and the power is calculated by linear interpolation between the respective closest power values; 2) correctly produce metrics such as static probability and transition density, while building the look-up table by proposing an algorithm that decreases the difference between the desired and the randomly generated parameters. Finally, they evaluated the impacts of the suggested improvements on the model's precision and flexibility.

Furthermore, authors in [7] adjusted the look-up table based power modeling by adding a new parameter to the input signals, which increases the table's dimension. This attribute was used to represent the average spatial correlation coefficient. Although, this method showed an RMS error of about 4% and an average error of about 6%, but it ignored the glitch power, which made this error computation not accurate. Therefore, ignoring the glitch power during the power estimation using only the zero delay models demonstrated a new RMS error that is below 1%. Note here that this error was computed using the zero delay models as reference, and this error represents the modeling error with respect to power values that do not take into consideration the glitch power. For this reason, this method induces a significant inaccuracy in the power estimation, and hence is far from the real circuits environment, especially that the input data stimuli signals of a circuit may hold unnecessary signals (glitches) in real circumstances. To this end, it is better to rely on the results that take into account the glitch power, which can be used as more reliable reference.

Finally, authors in [8] proposed a power estimation technique to the register transfer level model of the digital circuits. The proposed approach enables the designers, at a high-level of abstraction, to estimate the power dissipation of

Table 3.4 – Classification of Table-based Modeling Techniques

| References | Metrics | | | | | | |
|---|---|---|---|---|---|---|---|
| | Inputs | Estimation Technique | Modeling level | Error (%) | Modeling effort | Model granularity | Year |
| [4] ($T2$) | $A(t)$, $A(t-1)$, $B(t)$, $B(t-1)$..., | Statistical | Component | 7% (Avg) | *** | Fine | 1996 |
| [5] ($T2$) | $P_{in}$, $D_{in}$ & $D_{out}$ | Statistical | Circuit | 6% (RMS) | ** | Fine | 1997 |
| [6] ($T3$) | Same as [5] | Simulation | Component | $<$ 10% (Avg) | ** | Fine | 1999 |
| [7] ($T2$) | Same as [5] + $SC_{in}$ | Statistical | Circuit | 4% (RMS) or 6% (Avg) | *** | Fine | 2000 |
| [8] ($T2$) | Same as [7] + $T_{in}$, $P_{out}$, $D_{out}$, $S_{out}$ & $T_{out}$ | Statistical | Component | 1.84% (Avg) | *** | Fine | 2006 |
| [86] ($T2$) | Same as [8] | Statistical | Circuit | 15% (Avg) | *** | Fine | 2014 |

intellectual property (IP) components, using the look-up table based modeling technique. To map the power dissipation, several metrics of the I/O statistics are used, such as the average input signal probability $P_{in}$, the average input transition density $D_{in}$, the input spatial correlation $S_{in}$, the input temporal correlation $Tin$, the average output signal probability $P_{out}$, the average output transition density $D_{out}$, the output spatial correlation $S_{out}$ and the output temporal correlation $T_{out}$. The results of this method at IP level demonstrated an average error of 1.84%. Although this is a very good improvement in terms of estimation accuracy, however, additional attributes and computations are required. More specifically, the authors used eight metrics of the look-up table to get the power dissipation. In addition, the output metrics should be calculated before performing the power estimation. Under these conditions, additional computational effort and time is needed. The authors in [86] extended the work of [8], and they demonstrated the use of table-based method in composite system power estimation, and proved the scalability of the method. To this end, the method allowed system-level assessment of power consumption based on earlier characterized power models at component-level.

Table 3.4 summarizes the different power modeling techniques that use the look-up table as a modeling method. It is possible to compare the different works based on 4 metrics, such as **accuracy**, **modeling effort**, **modeling level** and **modeling granularity**. Among all these works, the most significant improvement in terms of accuracy at component level appears in [8]. Note also that the input entries that abstract the signal characteristics of the input and the output are more reliable for power estimation with better accuracy. In addition to this, spatial and temporal correlations of the primary input and outputs may enhance the precision of the look-up table-based power estimation technique.

Recall that these techniques consist of tabulating the values in order to estimate the global power of a design. To address these values that are stored in the table, different metrics may be used such as the input transition density, the input static probability, and the space-temporal correlations. Note here that knowing that some missing values may appear in the table, an additional method is hence required to interpolate these values. Although these techniques demonstrated their effectiveness by reaching a high estimation accuracy, but they also present some limitations such as: 1) **this technique is a memory consuming**

due to the large amount of the stored data, 2) **the modeling effort is thus considered as moderate**, and this factor depends on the number of attributes to consider, which may be significant, 3) **the computational effort** increases as the table grows, because of the dense search performed to get the proper power value for the given input entries. This encouraged the researchers to begin focusing on more effective methods.

### 3.4.3 Regression-based techniques

Long timing simulation and large number of design parameters encountered during the cycle-accurate power estimation constrain the design space exploration. To overcome this problem, *regression-based* power modeling is used for predicting power for various designs. To this end, it is possible to define the regression analysis as a statistical interpretation method, where the relationship between a dependent variables (power consumption) and one or more independent variable (i.e., the design parameters or also called the predictors) is established [71]. It is good to mention here that the regression-based techniques are those methods for which power values are obtained from simulations or measurements according to specific parameters (e.g., capacitance, switching activity, and clock frequency), that are modeled using a curve fitting approach e.g., linear regression. Thus, this category focuses on bottom-up techniques that usually require a characterization phase generally performed at the low-level.

In [87], regression-based power models for Digital Signal Processing (DSP) blocks were developed and achieved an accuracy of 20% against a gate-level simulator. In [88], the work of [87] is extended by considering spatial and temporal correlations of input signals when estimating switching activity using probabilities. Their regression-based approach demonstrated an improvement of the models, with an error lower than 2%.

Besides, the authors in [89] achieved a better accuracy by defining subsets of signals depending on their nature e.g., control or data signals. An adaptive regression method was employed to build a model considering statistical parameters. This provides a good trade-off between accuracy and simulation time. In fact, in a first step, a limited number of input sequences for a trace generator is used during to build the model. Then, a second step consists in calling a gate-level simulator only on a small number of cycles to improve the accuracy of the model. Finally, their approach for FPGA and CPLD achieved an average relative error of 3.1%.

Moreover, the authors in [90] proposed an advanced regression technique. This technique was presented to boost the accuracy of linear models using regression trees algorithm. In fact, if the best model is strongly non-linear, a linear approximation may lead to unacceptably large errors. Control variables were defined to choose the most appropriate regression equations among different ones. An online power characterization was also proposed to improve the power modeling

accuracy of small combinational circuits from 34.6% to 6.1%.

Regression-based power modeling technique is also considered for **FPGA devices**. FPGAs integrate multiple elements such as embedded DSP blocks, RAM blocks, Look-up-tables, and flip-flops. Some approaches tend to go further by taking into account the number of specific hardware resources in their modeling approach. In [91], a linear relationship between the amount of hardware resources, capacitance, I/Os switching activity and power was determined to build a general model over a set of diverse IPs. Low-level simulations were performed to gather signal activities while XPower Analyzer delivered power estimates. For a fixed number of IPs and training sequences, the average error of the model was around 6%, but when introducing new IPs and patterns, the power estimation error increased to 35% (in the worst-case). In [9], a power model for embedded DSP blocks of FPGAs was proposed, while taking into account various signal statistics and multiplier sizes. The model was realized using a multi-variable regression over different power measurements, achieving an accuracy of 7.9%.

Rather than considering the architectural elements of FPGA devices, another solution could consist of creating power models for basic operators, such as adders or multipliers [10, 11, 12]. While taking into account the switching activity, the operating frequency, the auto-correlation coefficients and the word length, a power model can be created as shown in [12] leading to an average of 10% on a Virtex-2 Pro FPGA. However, such approach does not consider the interconnection between the elements when estimating the power consumption of a more complex design.

When going up in abstraction, systems are composed of Intellectual Properties (IP) blocks. In [11], area and power models for fixed point and floating point IP were developed using curve fitting and linear regression. The main target was a Virtex-2 Pro board. The area model was used here to estimate the power by considering the number of slices ($T_{Slices}$), the memory blocks ($T_{BRAM}$) and the multipliers ($T_{MULT}$). Both the static and the dynamic power can be estimated from this method, with an average error of 7%. Note here that modeling operators is well-suited when estimating power at a higher level of abstraction, typically at the algorithmic level. Fast exploration is thus made possible. The introduced power models in [11] are recently improved in [92] by considering power optimization technique such as clock gating. The average error obtained for a set of several IPs was around 3%.

A complete framework was presented in [93]. This methodology, called Functional Level Power Analysis, aims at decomposing the system into functional blocks. The Components, that were activated in the same function, were clustered and real power measurement were performed.

In the aforementioned works, both algorithmic and architectural parameters are used, and linear regression was adopted to build the model at every considered level. The achieved estimation error was less than 5% against real measurements. Despite a good accuracy, one can notice that several real measurements

Table 3.5 – Classification of Regression-based Techniques

| References | Metrics | | | | | | |
|---|---|---|---|---|---|---|---|
| | Fitting Parameters | Estimation Technique | Modeling level | Error (%) | Modeling effort | Model granularity | Year |
| [87] ($R2$) | $P_{in}$, $D_{in}$, $SC_{in}$, $D_{out}$ | Statistical | Component | 20%(Avg) | ** | Fine | 1999 |
| [88] ($R3$) | Same as [87]+$S_{in}$, $T_{in}$ | Simulation | Component | 1.8%(Avg) | *** | Fine | 1999 |
| [90] ($R3$) | I/Os switching activities | Simulation | Component | 6.1% (Avg) | ** | Fine | 2000 |
| [89] ($R3$) | Same as [7] | Simulation | Component | 3.1%(Avg) | ** | Fine | 2001 |
| [10] ($R3$) | $SC_{in}$,$D_{in}$, bitwidth | Simulation | Component | 2% (Avg) | ** | Fine | 2004 |
| [11] ($R1$) | $T_{Slice}$,$T_{Mult}$,$T_{BRAM}$, | Probabilistic | Component | 7% (Avg) | * | Coarse | 2008 |
| [9] ($R4$) | $SWCVf$ | Measure | Component | 3%(Avg) | ** | Coarse | 2010 |
| [91] ($R1$) | $SWC_{eff}$ | Probabilistic | Component | 6%(Avg) | ** | Coarse | 2011 |
| [12] ($R3$) | $F, \rho, w, SW, N_{IO}, N_{Slices}$ | Simulation | Component | 10% (Avg) | ** | Coarse | 2012 |
| [92] ($R1$) | Same as [11] | Probabilistic | Component | 3% (Avg) | ** | Coarse | 2019 |

are required before obtaining the power model. To this purpose, a complete and automatic framework was developed during the OpenPeople project [94]. This approach was applied to FPGA devices in [95].

Finally, we summarize the main power modeling approaches based on regression in Table 3.5. Several metrics were chosen to guide the designers through the choice of techniques according to the **model parameters**, the **level of accuracy**, **the modeling effort and level** and **the model granularity**.

According to the table, developing power models with a coarse granularity produces good results with an error lower than 10%. Recall that a coarse granularity means that some abstraction is performed on hardware by only considering the number of hardware resources (slices, bram, etc.). We may note also that most of the approaches propose power models for specific hardware component, such as operators (dividers, adder, DSP) or interconnection elements (multiplexers). From this, it appears that architectural conditions such as clock gating has a significant impact on the accuracy of the power model [11, 92] due to the reduction of the impact of the switching activity.

Finally, such approaches do not consider 1) **more complex digital circuits** and 2) **the power consumption for a composite system** that is composed of different components. Hence, the exploration of the design space of the different possible configurations of a given design is not feasible. For this reason, new modeling techniques that are capable to model the power consumption of more complex or bigger system are needed. To this end, artificial neural networks are being proposed as a new modeling candidate.

### 3.4.4 Neural Networks based techniques

*Artificial Neural Networks* (ANNs) are based on connected neurons devices, which are responsible of transmitting the signals from one neuron to another, similarly to the synapses in the biological brain. As illustrated in Figure 3.6 and observed in the literature, ANNs can be used to solve the power modeling

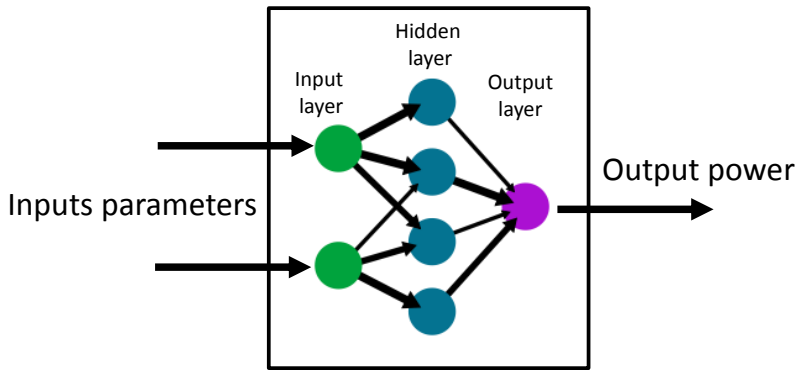



Figure 3.6 – Neural Power Model

problem in digital circuits. Existing works have proved the capability of ANNs to approximate generic classes of functions, including continuous and integrable ones [96]. Additionally, ANNs represent one of the tools that are used to model a non-linear system, while having the ability to learn the trends of the input data automatically. The advantage of this kind of networks is that it can catch the model easily, while achieving a very good accuracy with low complexity.

Authors in [97] proposed a new power modeling technique for the CMOS sequential circuits, while building on the recurrent neural networks (RNN). The main goal of using these neural networks was to learn the relationship between the I/O signal statistics and the corresponding power consumption. The work in [97] also considered the nonlinear characteristics of the power consumption distributions, as well as the temporal correlation of the input data. The results of the conducted experiments showed that the estimations were accurate with an error range of about 4.19%. In fact, this work was limited by two constraints. First, the large number of the necessary parameters that are needed to estimate the power consumption (which is about 8), while also requiring a cycle bit accurate computation in order to compute the predictors at the I/Os, which are used to estimate the power dissipation. Second, the application of the modeling technique on the CMOS circuits, without benefiting from the fact that these circuits may be composed of several hardware components that can offer more flexibility while exploring the power consumption of several design choices. For this reason, modeling the power consumption at component-level could be a better strategy than the one applied to a composite circuit.

Other existing works model the power consumption in CMOS digital circuits using another type of neural networks (instead of the RNN approach of [97]), i.e., the back-propagation neural network (BPNN), as in [98]. In this work, the authors modeled the relationship between the power consumption and the circuit's primary I/O statistics. The main difference between [98] and [97] resides in the behavioral simulation phase, which is a requirement in the latter. This

is because the power estimation models in [97] need to have the output features of the component at the input of the model. To this end, the power model needs to wait a certain time (behavioral simulation time) to compute the output transitions of the component. As a solution, the work presented in [98] can help in estimating the power estimation in a very short time (within seconds), without performing any behavioral simulation during the power estimation, which is only based on the primary inputs statistics information. The experiments conducted on the ISCAS-85 circuits showed an average absolute relative error below 5.0% for most of the circuits. It is good to mention here that both works show the same estimation accuracy, while [98] offers less modeling effort and faster estimation strategy.

The importance of the neural network based power modeling technique was also demonstrated in [99], especially for the high-level power estimation of the logic circuits, while selecting a simple BPNN to be trained with the available data sets. The work in [99] also provided a comparison among the previous proposed modeling methods, such as the look-up table based and the regression-based techniques. Relatively to the works of [97] and [98], the proposed neural network model performed better since it took into consideration the power consumption of a fine grain hardware component like the multiplier. This can help then in exploring any digital design composed of multipliers. However, the main limitation of this work is related to the number of the inputs of the model, since it is highly dependent on each input's bits width of the modeled components.

In addition to the digital circuits, other authors use the neural network to model the power consumption of an analog circuit [100]. They conduct several measurements in order to get instantaneous energy consumption according to inputs and operating parameters, such as the voltage and the frequency. Afterward, they use this data to train the neural networks. Finally, they include the neural models into a high-level simulator to get an overall power estimation at the system level. Validation results show an accuracy of about 1.53%. Note here that the work in [100] incorporates two estimation methods to estimate the overall power consumption of a heterogeneous system: it considers the neural networks to estimate the analog part only of the heterogeneous system, and it uses the look-up table based estimation technique to estimate the digital part. However, it could be better if the authors adopt the neural networks approach to estimate both parts, i.e., the analog part as well as the digital one in order to avoid any conflict, and to achieve more accurate results. It is good to mention here also that the authors do not precise the accuracy of the table based power estimation technique, and this affects on the design's choice by preventing the designers from knowing which model is affecting the estimation precision.

Neural network based technique was also used to model the power consumption of a chip as presented in [14]. In this work, the power consumption of this chip was modeled using different parameters, such as: frequency, flash, ROM, and RAM capacity. However, this work has several limitations worth noting: first, it

does not take into account the inputs' activity. Second, it is only valid for a given chip and cannot be generalized easily. Finally, there is no information regarding the estimation time, which is an important metric in modeling and estimation methodologies.

Other works considered special types of neural networks. For instance, authors in [101] presented a Radial Base Function (RBF) Artificial Neural Network (ANN) model to estimate the energy and the leakage power in standard cell register files. This ANN model used the number of words in the file (D), the number of bits in one word (W) and the total number of read and write ports (P). However, this limits the power model to specific registers and technology, making it not applicable to all technologies. Therefore, the technology factor should be taken into consideration and should be added as an input to the neural network, especially that, as we know, the leakage power consumption is a technology dependant metric. Hence, the general power models that are based on ANN can be modified by just adding the transistor length of the target technology to the neural network inputs. By this, the designers can work at a high-level of abstractions to explore technology effects on the design.

As an application of RBF, authors in [102] presented an artificial neural network based method for power estimation of ISCAS'89 Benchmark circuits, by exploiting Back Propagation Neural Network (BPNN) and Radial Basis Function Neural Network (RBFNN). They used the number of the inputs and outputs and the number of gates (such as the number of AND, OR, NOR, and DFF gates) in the VLSI circuit as predictors (without the need of the detailed architect of the circuit and its interconnection information) to deliver the power consumption as an output. The power estimation results based on BPNN were then compared to the estimation results of the RBFNN. However, the use of the neural network is not well motivated in this work. Moreover, the presented example about the power estimation of the ISCAS'89 Benchmark circuits ignored the fact that the power consumption is data dependant. This means that the power consumption of the same circuit resulting from the neural power model could have varied significantly if the data stimuli were taken into consideration. Hence, the presented model can be improved by taking into account the statistical information about primary inputs/outputs data stimulus.

Finally, authors in [103] presented the neural networks as a powerful modeling tool to perform both the power and the signal activities modeling of an IP (intellectual property) FPGA-based hardware block. They used the simulated data that can be obtained from low-level tool simulations, and evaluated the estimation time. The results showed that the minimum speedup factor achieved by the neural models was about 11500X. This approves the neural networks model's capability of estimating the power consumption in a very fast way, and with a good accuracy. Moreover, the authors discussed the design space exploration point of a global system, and their aim to estimate the power consumption of an IP-cascaded system. However, the authors did not show the details about the

Table 3.6 – Classification of Neural-based Techniques

| References | Metrics | | | | | | |
|---|---|---|---|---|---|---|---|
| | Fitting Parameters | Estimation Technique | Modeling level | Error (%) | Modeling effort | Model granularity | Year |
| [97] ($N3$) | $TI_{00}, TI_{01}, TI_{10}, TI_{11}, TO_{00}, TO_{01}, TO_{10}, TO_{11}$ | Simulation | Circuit | 4.19% (Avg) | *** | Fine | 2005 |
| [98] ($N3$) | $P_{in}, D_{in}, S_{in}, T_{in}$ | Simulation | Circuit | 2% (RMS) 5% (Avg) | ** | Fine | 2005 |
| [99] ($N3$) | $TN_I, TN_O, N1_I, N1_O$ | Simulation | Component | 2.25% (Avg) | ** | Fine | 2007 |
| [100] ($N4$) | $V_{in}$, $F_{sampling}$, & $F_{operating}$, | Measure | Circuit | 1.53% (Avg) | * | Coarse | 2010 |
| [14] | I/O number, frequency, & flash depth | - | Circuit | - | * | Coarse | 2010 |
| [101] | depth, width, & port | - | Component | 10.94%(-) | * | Coarse | 2012 |
| [102] ($N2$) | Number of I/O & Number of gates | Statistical | Component | 8.5% (Avg) | * | Coarse | 2013 |
| [103] ($N1$) | $SW_{in}$ & $P_1$ | Probabilistic | Component | 1.31% (Avg) | ** | Fine | 2016 |

system level methodology, and they missed to verify it. In other words, they assessed the proposed method on a single IP hardware block that was implemented on an FPGA, and all the presented results were conducted on a single IP (such as the IFFT block) only.

As a summary, we collect in Table 3.6 the aforementioned works that adopt the neural networks as a modeling technique to model the power consumption. While analyzing this table, we focus on the works of [99], [100] and [103], where most accurate results are presented. Note here that they show an estimation error which is less than of about 3%. However, the modeling in [100] is performed at the circuit level, limiting by this the capability for the design space exploration, since the different architecture of the circuit cannot be then explored. This is in contrast to the works of [99] and [103], where the modeling is preformed at the component-level. Comparing these two approaches, [103] delivers more accurate results than the work in [99].

### 3.4.5 Analysis and Discussions

In this section, we provide a detailed analysis and discussion about all the aforementioned power modeling techniques. We start by an evaluation of the modeling accuracy with respect to the existing works followed by an interpretation for these studied techniques.

In Figure 3.7, we show how the accuracy of the four power modeling techniques changes according to the literature. As for the axes, the y-axis corresponds to the modeling error (in percentage) and the x-axis to the literature, where each number on the second axis denotes the reference's number present in one of the tables (3.2, 3.4, 3.5 and 3.6). As we can see in Fig.3.7, it is possible to assess the performance of the different existing works and to classify them according to the modeling accuracy by setting a threshold on the modeling error. Based on Fig.3.7, we assume this threshold to be about 5% to divide the existing works into two categories: accurate approaches when the modeling error is below the threshold and less accurate when the error is exceeding it. We may also notice that most of the works that correspond to the analytical and table-based techniques reside in the upper side of the threshold, while the most of the works that belong to

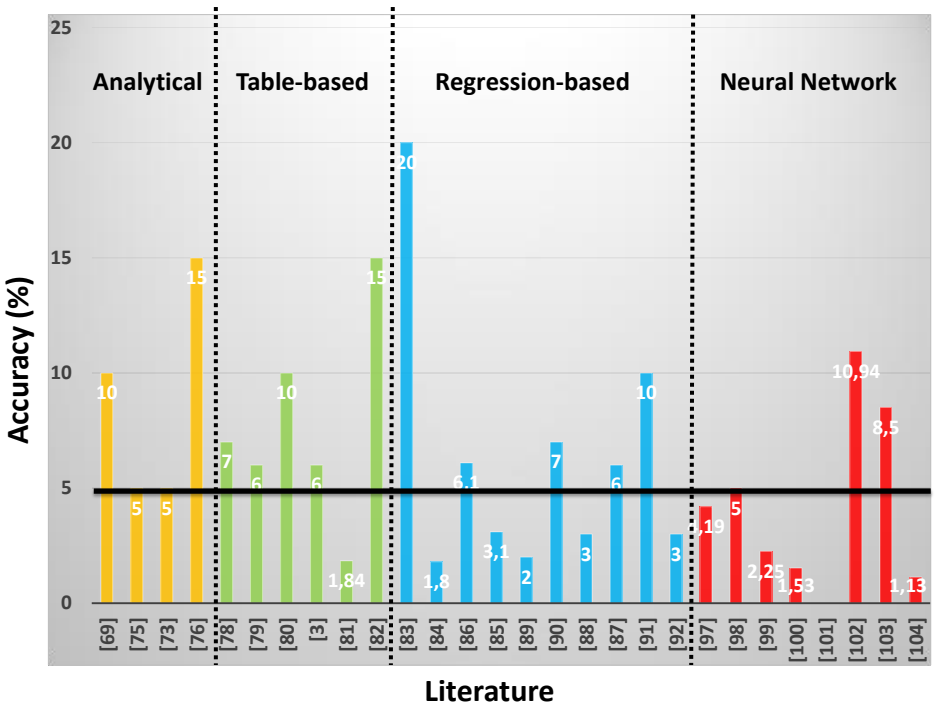



Figure 3.7 – Power Modeling Accuracy Versus The Observed Literature

the regression-based and neural networks lie below this threshold. Comparing the regression-based technique to the neural networks one, on one hand, about 50% of the works that belong to the regression-based method are considered less accurate (modeling error > 5%). On the other hand, 72% of the neural networks works can be considered as accurate methods, while representing a modeling error less than 5%. Therefore, this classification proves that the neural networks based approaches provide more accurate results than the regression-based one. For instance, as we can see in Fig.3.7, some works [103, 100] which belong to the neural networks achieve a modeling accuracy of less than 1.5%.

In the rest of this section, we analyze and compare the different power modeling techniques based on six metrics:

1. **Modeling effort**: We retain here only three levels of efforts for simplicity: 1) high, 2) moderate, and 3) low. These levels depend on how much *information* and *time* is required to build the model. For example, a high modelling effort may necessitate either a characterization phase or a significant time due to the large number of data to prepare prior to model creation. As for the moderate and low modeling effort, less number of data points and time are needed to develop the model;

2. **Memory resources**: We can define two attributes for this metric: high and low memory consuming. The first type appears when the methods re-

Table 3.7 – Metrics Reward and Penalty

| Modeling Techniques | Metrics | | | | | |
|---|---|---|---|---|---|---|
| | Modeling effort | Memory resources | Computational effort | Power Characterization | Accuracy | Model Scalability |
| **High** | -1 | -1 | -1 | - | +1 | - |
| **Moderate** | 0 | 0 | 0 | - | 0 | - |
| **Low** | +1 | +1 | +1 | - | -1 | - |
| **Yes** | - | - | - | -1 | - | +1 |
| **No** | - | - | - | +1 | - | -1 |

quire a large amount of memory to store modeling information, whereas the second one arise when the method does not require any memory allocation;

3. **Computational effort**: It represents the computational *resources* and *time* needed by the model to perform the estimation process;
4. **Power characterization**: We can determine here two approaches: the bottom-up and the top-down power modeling approaches. The first one appears when some methods require a power characterization phase in order to start the power modeling phase, whereas the second one arises when the methods do not involve this phase;
5. **Accuracy**: It shows the modeling error;
6. **Model scalability**: It relies on the ability to extend models to conduct composite system power estimation based on component level power models. For example, some techniques are restricted to one component or one digital circuit, while other techniques are scalable to support more complicated and composite digital systems.

In addition to the metrics described above, we present a novel metric that we call "the total score" to relatively assess the modeling techniques. For this, we first define the reward and penalty table as shown in Table 3.7, which quantitatively reflects each metric with respect to each level.

Then, we identify in Table 3.8 the different modeling techniques in the state of the art. Each modeling technique is studied according to the six extracted metrics. In the following paragraphs, we analyze each modeling technique separately.

We start by analyzing the **Analytical** power modeling techniques that are based on the theoretical equations of a CMOS transistor, as defined in the liter-

Table 3.8 – Power modeling techniques Assessment

| Modeling Techniques | Metrics | | | | | | Total Score (S) |
|---|---|---|---|---|---|---|---|
| | Modeling effort | Memory resources | Computational effort | Power Characterization | Accuracy | Model Scalability | |
| **Analytical** | high(-1) | low(+1) | low(+1) | no(+1) | low (-1) | no(-1) | **-2** |
| **Table-based** | moderate(0) | high(-1) | moderate(0) | yes(-1) | low (-1) | yes(+1) | **-2** |
| **Regression-based** | moderate(0) | low(+1) | low(+1) | yes(-1) | moderate(0) | no(-1) | **0** |
| **Neural Networks** | high(-1) | low(+1) | low(+1) | yes(-1) | high(+1) | no(-1) | **+1** |

ature [71]. These techniques try to analytically find a relationship between the power and the fundamental parameters that impact the power such as the switching activity and the capacitance. These metrics can be then estimated in several ways as we already described in the analytical section (See section 3.4.1). This type of techniques has some advantages and drawbacks. As for the drawbacks, we may notice that 1) it generally requires a high *modeling effort* due to the complex architecture of the digital systems and 2) as discussed in Section 3.4.1, the main drawback of such approach is its lack of information about the physical implementation details like the glitches, which may lead to less *accurate* results. At the same time, advantages of this technique can be noticed as well as it does not require neither high *memory resources* nor a high *computational effort*. This is because the model itself does not need a memory to store data or even a high computational effort to perform the power estimation. In addition to this, this approach does not rely on any *power characterization* data and information, so that they do not require any power characterization phase to build the power models, and it is based on a top-down approach, hence, no characterization effort and time is needed. The most important advantage of the analytical power modeling technique is the only power modeling technique that does not rely on a power characterization phase to build a power model. To give more details, power characterization is the process of extracting the power consumption information according to different simulation/environmental parameters. For example, one can physically measure the power consumption of a design or use a tool to estimate it while varying the frequency, the statistical features of the input patterns, and the voltage. These obtained power data are then used by the modeling technique to construct an abstract power model. Finally, the estimation *scalability* is not maintained in this technique due to the hard derivation of the composite power consumption model of a complex system, while building on the component level power models.

As for the **Table-based** technique, it is possible to consider that the *modeling effort* is at the moderate level (in average). Recall that this metric depends on two factors: 1) the number of the used attributes/predictors to build the table/model and 2) the number of the used data points to prepare the table. According to the literature (See Section 3.4.2), the number of predictors depends on the number of the inputs of the component/circuit (in some cases) and may vary from 2 till 8 as in [86]. Concerning the *memory resources*, this technique can be considered as a memory consuming because of the large number of data points required to be stored to build up the power model. Note here that the *computational effort* increases as the table dimensions grow. This is because of dense search that is required to perform in order to fetch the proper power values for given input entries. To this end, we can consider it as moderate level. Moreover, this technique usually employs a bottom-up approach, since the *power characterization* is often needed. Note also that the *precision* in this technique is not guaranteed, where most of the works present more than 5% error, making this

technique performing at a low level of accuracy as depicted in Fig.3.7. The main source of error here goes back to the lack or missing of some data points from the table. Finally, this technique can be considered as a scalable one. This *model scalability* was shown in [86], where the table-based power models of IP-based modules were extended and connected together to get the power consumption of a composite system, such as the System-on-Chip device.

As for the **Regression-based** power models, they may be built with a moderate *modeling effort*. This is because the polynomial models, which are generally used to model the power consumption, do not require a large amount of data points and time to create the model. With regard to the *memory resources*, they do not need to store data, so low memory resources are one of the crucial features. The *computation effort* here is also not high, as these techniques do not need to perform any additional operation to estimate the power. Moreover, a *power characterization* is needed to build these power models. Regarding the estimation accuracy, we may consider the regression-based models as moderate ones since 50% of the existing works in the literature present an error which is less than 5 % (See Fig. 3.7). Recall here that the precision of this method depends on the number of the used coefficients and the degree of the model. Finally, the *modeling scalability* is impossible in this technique because of the complexity of deriving a composite power model from the component based power models, as described in [86].

Finally, the **neural networks** modeling technique needs a high *modeling effort* since the training time could be significant, and a large number of data sets are needed to train the neural networks. However, less *memory resources* are required, where this memory is only used to store the weights and the biases. Regarding the *computational effort*, it is considered low due to the time needed to perform the computation of the output with respect to a given input. Note also that this metric is directly related to the network size. Additionally, this technique requires a *power characterization* phase in order to prepare the training data. This method shows a high *precision* in terms of power estimation, as depicted in Fig.3.7, where most the observed works provide an estimation error with less than 5%. Regarding the *modeling scalability*, it is possible to be maintained as well because of the introduced behavioral model, which can be used in order to be cascaded with others models to perform a system level power estimation as introduced in [103].

To summarize, Figure 3.8 shows the evaluation of each technique separately, and the assessment is performed with respect to the defined metrics at the beginning of this section. For this purpose, quantitative results are provided and average score per technique is computed. Average score $S$, which offers a quantitative number to assess each technique in terms of average quality or performance. As shown in table 3.8, analytical and table based techniques are very limited and they present a very low score, which is of about -2. However, Regression and neural networks present a better score (large area) of about 0 and +1 respectively.

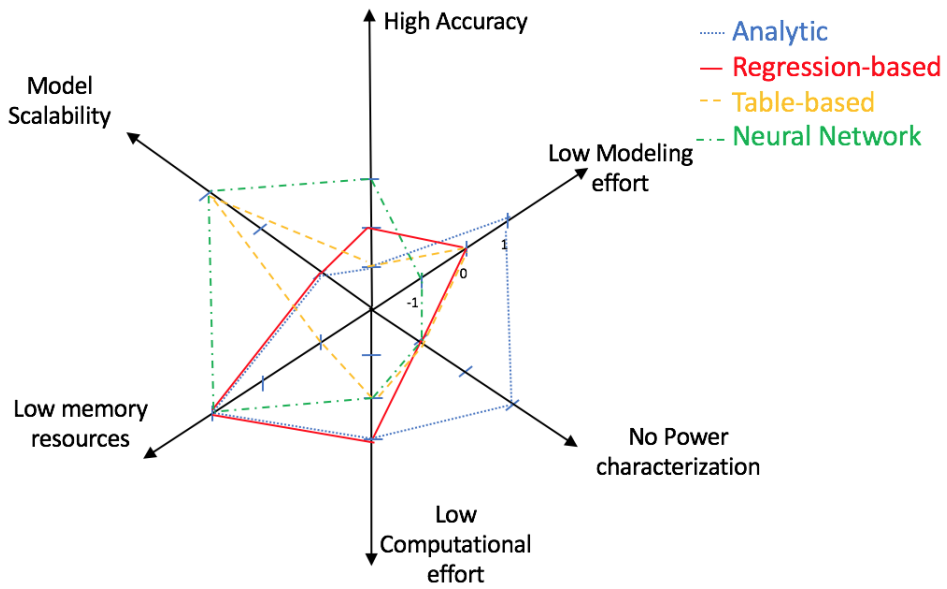



Figure 3.8 – Comparison on modeling techniques

Note here that the **neural network** technique overcomes all the techniques with the highest score.

## 3.5 Literature Review Clustering

In this section, we present a detailed analysis about all the existing works with respect to two approaches: the modeling technique as well as the characterization one. In Figure 3.9, we show the different regions where the literature can be clustered. We start by defining each of these clusters with respect to 1) *the characterization* technique on the X-axis and 2) the *modeling technique* on the Y-axis. Afterwards, we may label each cluster by one of the following four labels: 1) Accurate (in terms of modeling), 2) Fast (in terms of estimation time), 3) Model Scalability, 4) Less modeling effort, and 5) Less Memory resources. In addition to this, we associate to each characterization technique a color code as follows:

— The *red* color corresponds to the **probabilistic-based** technique;

— The *orange* color is linked to the **statistical-based** technique;

— The *green* color is coupled with the **simulation-based** technique;

— The *light green* color is associated with the **measurement-based** technique.

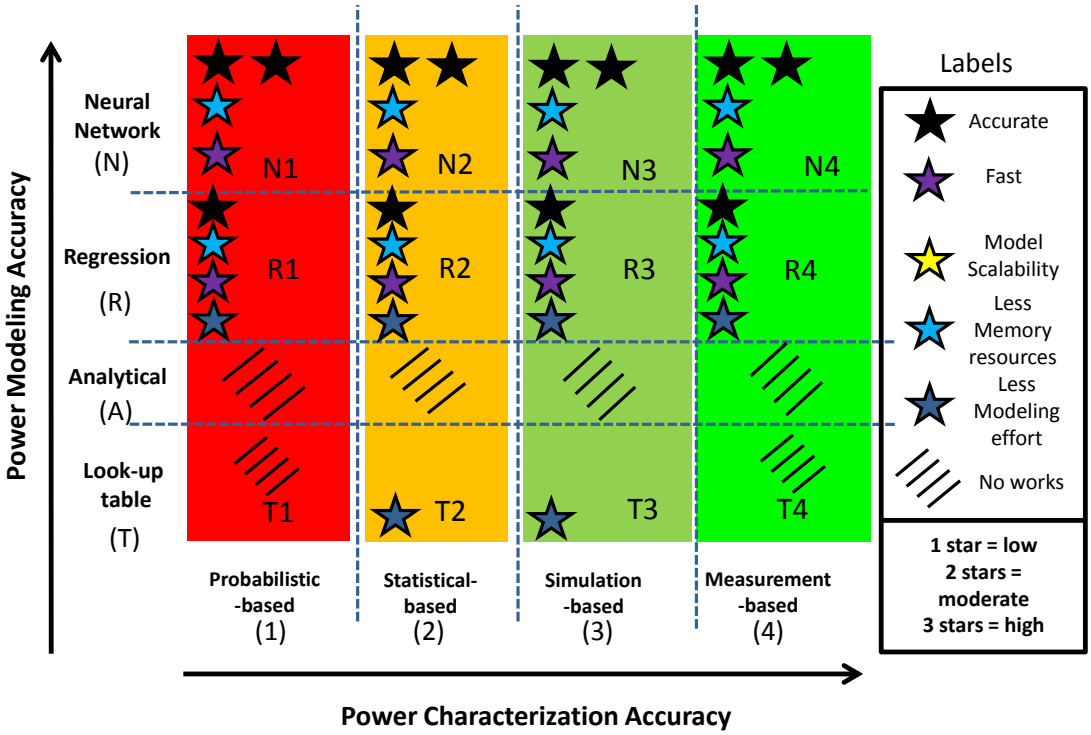



Figure 3.9 – Power Characterization Techniques Versus Power Modeling Approaches

Regarding the power characterization techniques, they are classified from the less (probabilistic-based) to the most accurate ones (measurement-based). Based on Section 3.4.5, the modeling techniques are also grouped from the less (look-up table) to the most accurate ones (neural-networks). The overall accuracy depends then on the used characterization technique (more realistic models are based on real data) and on the modeling one (the best model is the one that fits the data properly).

Firstly, it is possible to define two clusters $R1$ and $N1$ for the probabilistic-based power characterization (Red Color) corresponding to the regression-based and to the neural network power modeling techniques, respectively. Note here that the works that belong to Cluster $R1$ and $N1$ are fast and need less memory resources. However, the works that belong to $R1$ do not require exceptional effort to create the model as in $N1$ case, where the training phase is required prior for the model creation [103]. Comparing both clusters, we may notice that the work presented in $N1$ offers more accurate results relative to the minimum error delivered in [92] by about 1.3%.

Secondly, we may identify three clusters $R2$, $T2$ and $N2$ for the statistical-based power characterization corresponding to the regression, table-based and

neural network, respectively. Regarding the labels, we identify the accurate results in $N2$ and $R2$. However, a faster estimation speed is always presented in $N2$ and $R2$. From a modeling effort perspective, $T2$ and $R2$ require less effort than $N2$, while large memory resources are needed for $T2$. However, all the previously mentioned works do not take into account a significant metric, i.e., the model scalability, except for the work in [86] that has proved this scalability for the table-based power modeling method. This is demonstrated by the power estimation, which is extended from the IP-level power models to IP-based system-level power estimation.

Thirdly, it is possible to have three clusters $R3$, $T3$ and $N3$ for the simulation-based technique. Concerning the accuracy label, most accurate works are present in cluster $N3$, where the maximum error that can be detected is about 5% [98]. As for $T3$ and $R3$, they both provide less accurate results. In terms of the estimation speed, the works in $N3$ and $R3$ are faster than the ones presented in $T3$. At the same time, $T3$ and $R3$ are better than $N3$ from a modeling level perspective, as they need less effort to perform the power modeling (no training required).

Fourthly, concerning the measurement-based power characterization, it is possible to classify the works into two clusters $R4$ and $N4$. Cluster $R4$ includes the work of [9] and $N4$ contains the approach that is described in [100]. Comparing both works that target the power modeling, we may notice that the methods that are based on neural networks provide more accurate power estimation results (of about 1.53%) [100] compared to the regression-based technique discussed in [9], which provides an error of about 3%.

To conclude, characterization-based power modeling techniques are as follows: 1) table-based, 2) regression, and 3) neural-networks. The most accurate cluster is the one that combines the most accurate characterization technique (such as the measurement) and the best modeling technique (such as the neural network). This intersection represents the optimal solution for accurate and realistic power estimation. Finally, we may notice that only the work of [86] offers model scalability, while only the work in [100] provides models that are based on the intersection between measurement (concerning the characterization technique on the X-axis) and neural networks (as for the modeling technique on the Y-axis).

## 3.6 Conclusion

High power consumption in FPGAs and ASICs digital circuits implies that reviewing and understanding the different power measurement, estimation and modeling approaches is very crucial to have more efficient computer-aided-design (CAD) strategies. This motivated the work of this chapter in order to check which method is the best in terms of **estimation accuracy**, **estimation speed** and **estimation scalability**.

For a better understanding, we classified the *power measurement* technique

into two classes: on-board and external solutions. We also provided a detailed discussion while presenting the limitations and advantages for each of these classes. The studied results showed that using an external power solution to measure the power consumption is better (in terms of accuracy) than measuring it using on-board solution. However, measuring the power is considered an expensive and time consuming solution. For this purpose, we conducted a deep analysis to adopt an alternative solution, i.e., the *power estimation* technique, that is usually used to characterize the power consumption for the digital circuits. The analyzed power estimation techniques presented different level of accuracy and estimation speed. Consequently, we use these metrics as tools to evaluate the different techniques. We noticed that although the simulation-based power estimation technique is more accurate than the probabilistic-based one, but the latter showed a high estimation speed comparing to the former.

After discussing the problems resulting from the power estimation techniques, we conducted a deep analysis for the proposed solutions that are described in the *power modeling techniques* section 3.4. Actually, we reviewed four modeling techniques (analytical, table-based, regression and neural networks) while defining different metrics to perform a fair comparison. In addition to this, we specified quantitative metrics that allow to evaluate the different power modeling techniques using numbers such as: **modeling effort, memory resources, computational effort, power characterization, accuracy and model scalability**. The analysis showed that the regression-based and neural networks methods are superior to the analytical and the table-based power modeling approaches in terms of estimation accuracy and estimation speed. Comparing the neural networks to the regression-based technique, the former outperforms the latter in terms of accuracy, which is a very significant factor. Moreover, Section 3.4.5 proved that the neural networks-based power modeling has the highest score, and hence is considered the most efficient power modeling technique. All of this motivated us to adopt the neural networks approach to model the power consumption, where the power characterization stage precedes the model creation one.

Finally, we built our work in this thesis on the existing works in [97, 98, 99, 103]. Although these works provide more reliable power models that are based on neural networks than the aforementioned estimation techniques, however model scalability is missed. The model scalability issue is very crucial due to its use at system-level power exploration. Taking this into consideration, we extend the work described in [103], where we investigated the scalability of the neural networks in order to provide useful tool, which aids designers to efficiently estimate power consumption. Integrating our work within the literature, we can say that it belongs to $N3$ cluster that is depicted in Figure 3.9.

# Chapter 4

# Power Estimation Method

## 4.1 Introduction

In this chapter, we propose two methodologies that enable a high-level power estimation. These methodologies can be listed as follows: 1) characterizing the power and behavior of a hardware component, 2) adopting neural networks to model both power and behavior of several FPGAs/ASICs hardware components. The aim of the proposed methodologies (characterization and modeling) is to provide a library of components with its associated power models, that can be used to model any digital signal processing design. The library of components is built according to different hardware platforms. The advantage of this library of models is to have an off-the-shelf solution, and hence a more flexible strategy to evaluate the power consumption at system-level. Indeed, the space of possible design solutions, i.e., the different combinations of the components within the library, could be easily and efficiently evaluated.

## 4.2 Power Estimation Strategy

Our power estimation strategy is based on the assumption that any hardware system can be represented by a set of hardware components that are dedicated to a specific function. The main idea consists in estimating the power consumption of a composite digital system from an accurate power estimation of its sub-components models. Each of these components (available in a dedicated library) has been fully characterized at low-level. This allows designers to have the possibility to construct their own architecture by connecting components in a high-level design entry tool that may consider schematics or by using a hardware description language.

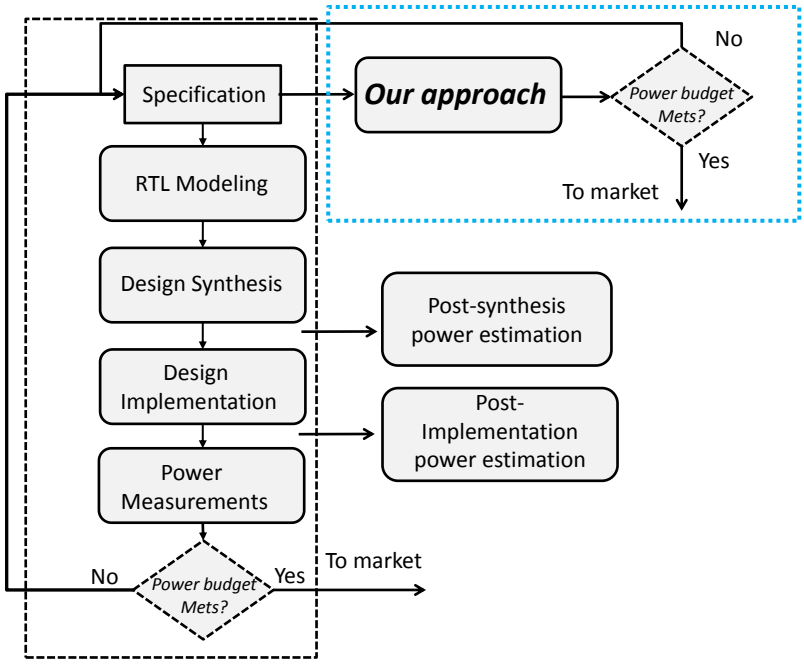



Figure 4.1 – Proposed Power Estimation Method from Users' Perspectives

### 4.2.1 Power Estimation from Users' Perspectives

In general, classical power evaluation flow consists of design entry (specification), RTL modeling, design synthesis, and design implementation. As shown in Figure 4.1 (left branch), power consumption can be evaluated either by measurement or by estimation. Regarding the estimation, it is possible to conduct it either at post-synthesis or post-implementation stage. Note that the lower the estimation level, the greater the precision is achieved. From the users' point of view, our technique enables designers to skip all the required steps to evaluate the design's power consumption. In this respect, a high-level modeling flow that is used in tools such as (MATLAB/Simulink), Labview, SystemC, etc. is sufficient to be adopted with our power estimation approach (See Figure 4.1 (right branch)).

Moreover, the full design may then be simulated at a high-level of abstraction to evaluate the performance of the proposed architecture and to obtain an accurate evaluation of the design power consumption. Note also that our methodology makes it possible to estimate the power consumption of various combinations of components (and then of architectures) very easily by simply modifying component's model in a high-level design entry tool. In addition, the simulation results may be achieved in a very short time since our power models are not computationally intensive. This allows designers to explore a lot of design choices with different configurations in few minutes. Compared to classical tools, this type

of simulation is generally not possible since designers usually need to implement all designs from scratch when a simple change is performed on the architecture. This may actually lead to several hours or days of simulations to get an accurate power consumption of a full design.

### 4.2.2 Model Definition

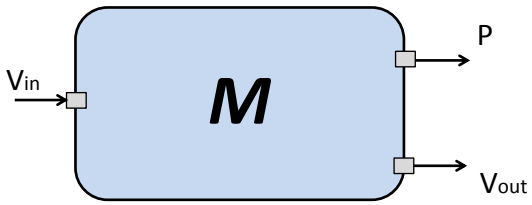



Figure 4.2 – Component Model

In this section, we describe the component's model, which is used to estimate power consumption. An example of such a component can be a simple operator such as an adder, a subtractor, a multiplexer, a multiplier, a decoder, a register, etc.. Each component has its own functionality with a defined number of inputs and outputs. In order to assess the power consumption of a given circuit, we propose to associate each component with a power model $M$ as depicted in Figure 4.2, where the power consumption depends on the features of its primary inputs embedded in a vector $V_{in}$. Actually, each of these component's model consists of two outputs: the first one predicts the power consumption $P$ of a given features vector $V_{in}$ of its primary inputs, whereas the second one makes it possible to estimate the output features vector $V_{out}$ of the primary outputs according to the component's primary inputs features. This second model is particularly useful when designers want to propagate these features among all cascaded or connected components in their design. In order to obtain a global power estimation of the entire design, it is then possible to sum up the power contribution of each component at a high-level of abstraction.

### 4.2.3 System-level Power Estimation

Our estimation method deals with composite systems that involve several cascaded components $(C1, C2, C3, ..., Cn)$ or layers described at high-level of abstraction as depicted in the top of Figure 4.3 (Composite System). In parallel, the system topology can be described in terms of the component's model $M$, as illustrated in the bottom of Figure 4.3 (Power Estimation Strategy). The peculiarity of our method is that it propagates the features vectors of the primary inputs through all the connected components or layers, thanks to the second

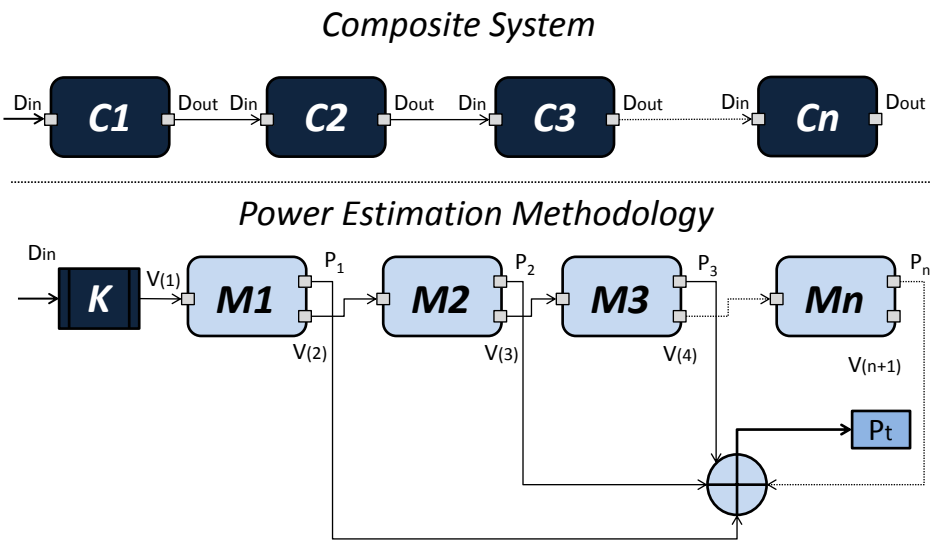



Figure 4.3 – Proposed power estimation strategy

model, which estimates output features. To this end, it is possible to obtain a system-level power estimation $P_t$, which corresponds to the total amount of the dynamic power consumption and that can be expressed as follows:

$$P_t = \sum_{i=1}^{i=n} P_i \tag{4.1}$$

where $i$ is the index of the component's model that ranges from 1 to $n$ components.

To build our power estimation strategy, two methodologies have been proposed: a characterization methodology and a modeling one. The aim of the former is to characterize the power consumption of the components, whereas the latter's objective is to model the power itself. As illustrated in Figure 4.4, after the hardware component is synthesised, the component power and behavioral characterization phase is performed. Then, power and behavioral modeling process is started in order to generate the component's models. Consequently, these two methodologies provide a library of components, with the associated components models that can be used to estimate the power consumption of a composite system. The library of component's models is then built according to the characterized components that exist on a hardware platform/target that corresponds to ASICs technology or FPGAs platforms. Finally, the corresponding library of models is provided to the designers working at high-level of abstraction in order to estimate the power consumption.

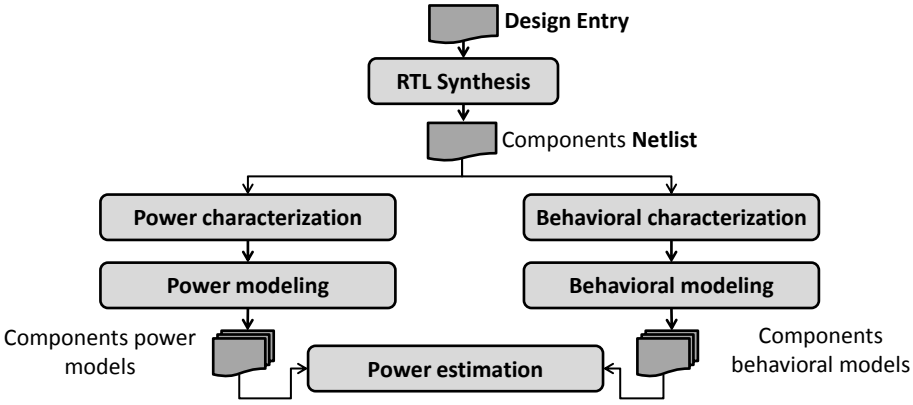



Figure 4.4 – Characterization and modeling methodologies

## 4.3 Characterization Methodology

The characterization phase is an essential phase to obtain pertinent information regarding the power that is consumed by a specific component. This phase consists in implementing the component on a given hardware platform/target, and hence obtaining power results after applying various configurations. This process is performed at a low-level of abstraction in order to take into consideration the physical details of the implementation as much as possible, and obtain the best accuracy in terms of estimation. For this purpose, power estimation tools can be used to prepare a huge database of power values that take into account very low information details.

The complete characterization flow is depicted in Figure 4.4. The flow starts by a common branch and then splits into a behavioral (right branch) and a power (left branch) characterization. The common branch requires at first to describe the components involved in the system at RTL level, by means of Hardware Description Languages (HDL), like Verilog or VHDL. Such RTL descriptions of the components are then synthesized according to the desired target technology and to the operating frequency. The RTL synthesis is mandatory here to retrieve:

— the netlist, that is the description of the system and its components in terms of native cells or gates. The gate-level description is unique, in the sense that it is specific to the selected target technology, such as ASIC or FPGA;

— the Standard Delay Format (SDF) file, that stores the timing information generated by the adopted CAD tool. This SDF file is not only unique, but it is also design process-dependent, since it contains scaling, environmental, and technological parameters.

The characterization step is based on timing simulation that takes the netlist and the SDF file, which are coming from the previous synthesis step, as inputs. Moreover, it also receives the timing libraries of the chosen technology as an

input. Consequently, additional information about the timing of cells involved in the netlist, and a testbench to feed the netlist with given input stimuli. For power estimation of the targeted platform, the netlist and SDF file represent a good trade-off between two metrics, the speed and the accuracy. They lie halfway between behavioral (not accurate, but fast) and post-implementation (more accurate, but slower) simulation and power estimation possibilities.

### 4.3.1 Parameters Motivation

The characterization phase is performed at two levels: the power and behavioral levels. The aim of the first is to model power consumption as a function of given input parameters. For the second, its objective is to run the timing simulation in order to get the output characterization parameters. These parameters are obtained by post-synthesis simulations and are directly used in the modeling phase. Note here that the characterization phase requires the netlist and the SDF file as inputs. For this purpose, the power and the behavioral characterization phases are highly dependent on the following three important factors:

— **the technology**;

— **the timing library**;

— **the characterization parameters**.

Therefore, we suppose that it is feasible to restrict the number of variables that change the characterization outcomes by performing the characterization on a **selected technology** using a specified **timing library**. Now, we can assume that the characterization depends only on the **characterization parameters** such as:

1. Operating frequency;
2. Switching activities.

According to [104] and [105], the dynamic power, which is caused by signal transitions (switching activities) in the circuit, is the dominant portion of the total power consumption. Additionally, a higher operating frequency leads to more frequent signal transitions and results in an increased power dissipation. Therefore, we may consider that the **operating frequency** is one of the characterization parameters.

The most significant source of dynamic power consumption in CMOS circuits is the capacitance's charging and discharging phases, that depend on signals transitions. This was proved in the literature, where the authors in [7] studied the effects of some parameters that represent the signal transitions. Finally, authors in [7] showed that the dynamic power depends on the circuit primary inputs **switching activities**. This means that it is very important to take these parameters into consideration to build a power model. More specifically, Figure 4.5

Table 4.1 – Characterization Parameters Selection

| Literature | Parameters | Error(%) |
|---|---|---|
| [7] | $P_{in}$, $D_{in}$, $S_{in}$ & $D_{out}$ (Block-level) | 6% (Avg) |
| [103] | (Signal probability & transition density at Bit-level) | 1.31%(Max) |

shows how dynamic power consumption changes in function of the average transition density and the average static probability of the primary inputs data. Note here that primary inputs signal statistics can be obtained from a high-level functional simulation, avoiding time consuming gate-level simulations.

As aforementioned, the **operating frequency** and the **switching activities** are considered as reliable and high-level characterization parameters. For this reason, Table 4.1 shows the difference between the presented works in [7] and [103] at two levels: the estimation accuracy and the used input parameters. On one hand, the authors in [7] used the average input signal probability $P_{in}$, the average input transition density $D_{in}$, the average input spatial correlation coefficient $S_{in}$, and the average output (zero-delay) transition density $D_{out}$ parameters to characterize and model power consumption. Note that, these values are computed based on a block of data at the component's inputs. On the other hand, the authors in [103] introduced the use of the input signal probability (also called static probability) and transition density (or also called activity factor), but at bit-level or signal-level to obtain more accurate results. By this, considering average numbers at data block level as in [7] is not enough to characterize power consumption. In addition to this, authors in [103] considered an IFFT block leading to a maximum estimation error of about 1.31%, in contrast to [7] which adopted a small circuit like an interrupt control (that consisted of 160 gates) showing an average error of about 6%. We notice then a crucial improvement in terms of estimation accuracy, while only considering the signal probability and transition density at bit-level.

Moreover, Figure 4.5 shows that the relationship between the dynamic power consumption and the average static probability $P_{in}$ is nonlinear and the plots do not have a consistent shape. Consequently, these results motivate the use of the static probability to model power consumption. Based on this, the used characterization parameters can be defined as follows:

**The activity factor** $AF[n]$, represents the average number of transitions ($0 \rightarrow 1$ and $1 \rightarrow 0$) in a data sequence, which is associated to a component's input $n$, and can be expressed as follows:

$$AF[n] = \frac{\sum_{i=0}^{i=L-1}(x_{i,n} \oplus x_{i+1,n})}{L-1} \tag{4.2}$$

**The static probability** $P_1[n]$, shows the percentage of ones (1's) or the logic high in a data sequence, that is associated to component's input $n$, and can be

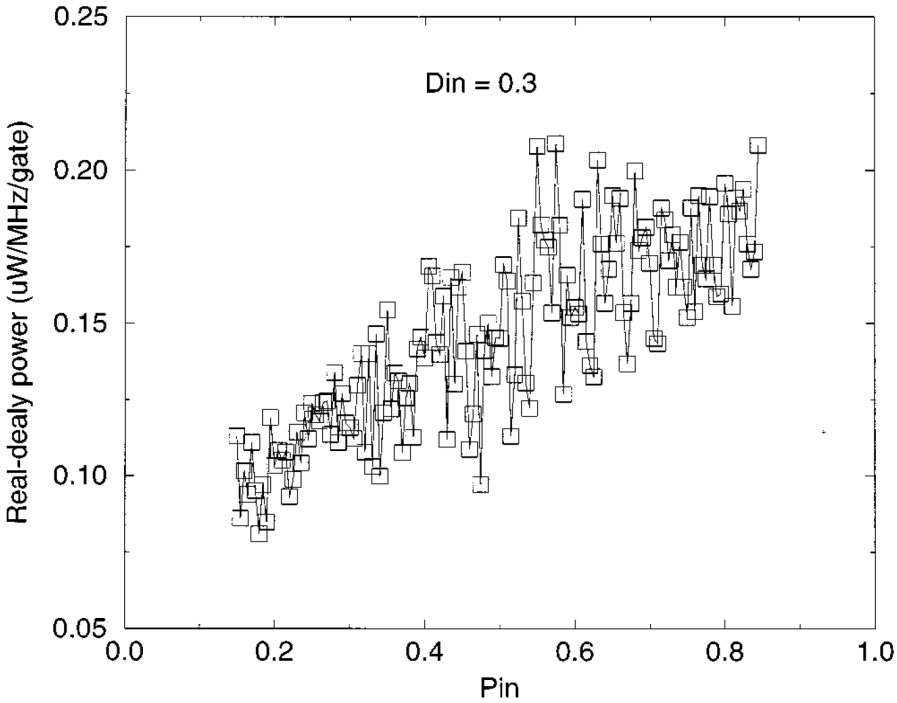



Figure 4.5 – Plot of the total power for an ALU for $D_{in} = 0.3$ and different $P_{in}$ from [7]

expressed as follows:

$$P_1[n] = \frac{\sum_{i=0}^{i=L-1}(x_{i,n})}{L} \tag{4.3}$$

In eq. 4.2 and 4.3, $i$ represents the column index, and $n$ is the row index that represents the $n$-th bit of the input data block (See Figure 4.6). Being $L$ the sequence length of the data (binary signal), that represents also the number of clock periods per data block or the time window. $N$ is the total number of bit (width) of component's inputs, and $x_{i,n}$ is the value (0 or 1) of the $n$-th bit and $i$-th column. Note that $n$ ranges from 0 to $N-1$, and $i$ from 0 to $L-1$.

### 4.3.2 Parameters Boundaries

In this section, we discuss the coverage of the described characterization parameters. We also define the boundaries of the activity factor per bit $AF[n]$ and of the static probability $P_1[n]$ (see definition provided by Equation 4.2 and 4.3).

Regarding the **activity factor** $AF[n]$ that can be better understood by comparing the corresponding waveform with the clock signal one, it is possible to bound it between 0 and 1, as depicted in Equation 4.4:

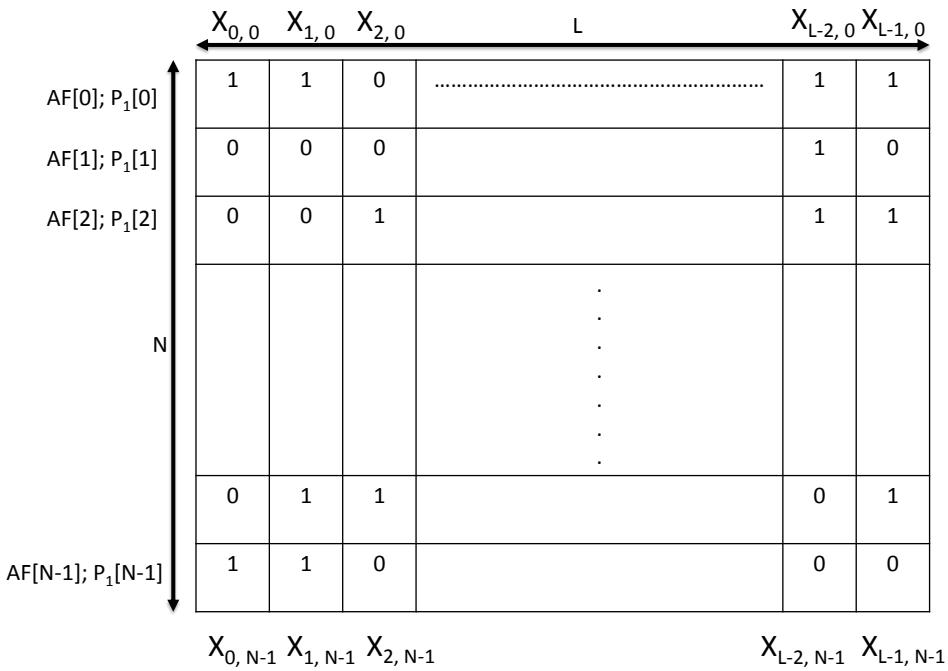



Figure 4.6 – Block data of N rows (inputs) and L columns (time window)

$$0 \leq AF[n] \leq 1 \tag{4.4}$$

A value of 0 refers to the status when no transition occurs in the data sequence (binary signal). As for its upper bound, i.e., the value of 1 means that the binary signal is flipping one time every clock cycle. In our proposed methodology, the used parameters are related to the primary inputs and outputs bits of the characterized components. To this end, the activity factor of the output bits depends on the internal behaviour of the components. Ideally, given the synchronized signals at the input, the data signals change only at the rising edges of the clock signal, and the components have registered inputs and outputs, which limit the output signals to flip more than once per clock period.

The **static probability** is described as the average fraction of clock cycles in which the final value of the signal is a logic high. This probability is limited since the proportion of time the signal can be high depends on the flips (value changes) of the same signal. In particular, when $AF[n] = 1$ then $P_1[n]$ becomes equal to 0.5 since the signal is always changing state during the observed interval (at each clock period), by this making the percentage of seeing it equal to 50%. On the other hand, when $AF[n] = 0$ then $P_1[n]$ becomes equal to 0 or 1 since the signal is not changing, but stays at its initial state, that can be 0 or 1.

Consequently, according to [7], the resulting boundaries for the static probability can be written as in Equation 4.5.

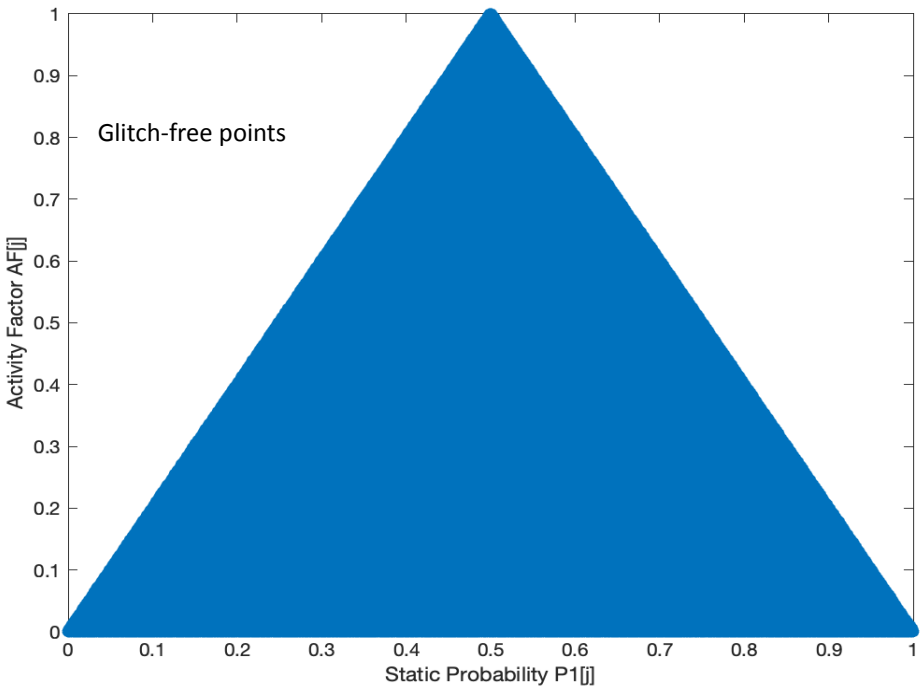



Figure 4.7 – Relationship between the activity factor and the static probability of a binary signal at given bit

$$0 \leq \frac{AF[n]}{2} \leq P_1[n] \leq 1 - \frac{AF[n]}{2} \leq 1 \tag{4.5}$$

In Figure 4.7, we show the activity factor $AF[n]$ as a function of the static probability $P_1[n]$, such that the activity factor is restricted to the shaded region.

In practical scenarios, even if we consider synchronized signals at the component's inputs, with registered component's inputs and outputs, the propagation of the signals from the inputs to the component's outputs can introduce additional transitions called glitches [106]. This is even more evident if one or both assumptions, synchronized inputs and registered inputs/outputs, are not verified. In practical circumstances, the activity factor $AF[n]$ of the output bits can also reach the value of 2, meaning that the output signal is flipping like the clock, i.e., twice per clock period (from 0 to 1 and it returns back to 0). Therefore, when glitchy signals are considered, the boundaries for $AF[n]$ and $P_1[n]$ are derived as shown in Equations 4.6 and 4.7:

$$1 < AF[n] \leq 2 \tag{4.6}$$

$$\frac{AF[n]}{4} < P_1[n] \leq 1 - \frac{AF[n]}{4} \tag{4.7}$$

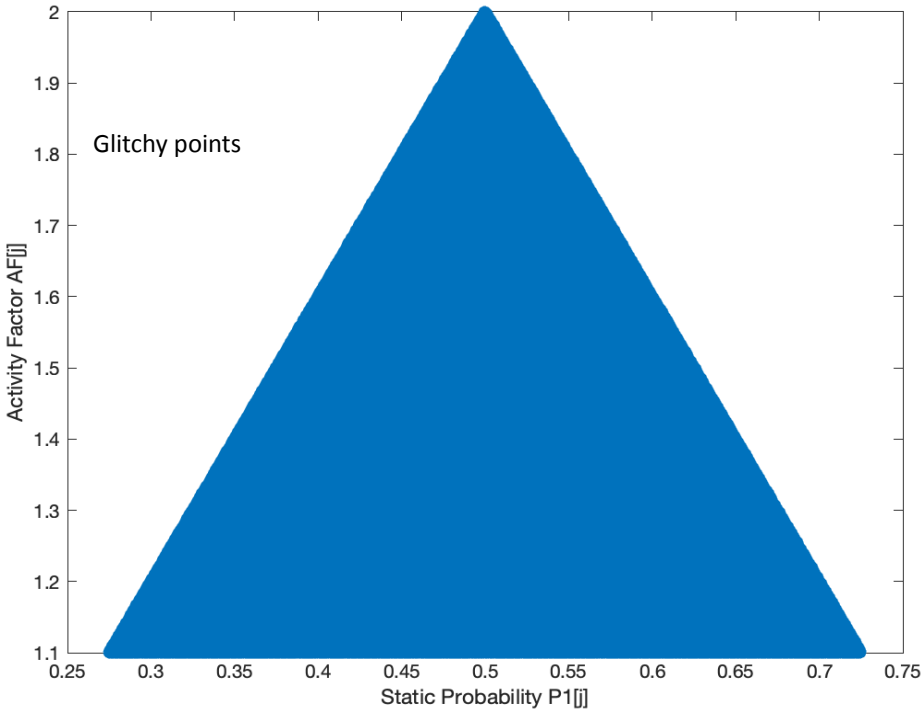



Figure 4.8 – Plot of the relationship between the activity factor and the static probability of a binary signal (glitchy) at given bit

Note here that the boundary of 2 for $AF[n]$ in Equation 4.6 strongly depends on how the cells are modeled in terms of timing information in the considered target technology. This value has been derived directly by post-synthesis timing simulations using the gate-level tool in Cadence [59]. In Figure 4.8, we show the new relationship between the activity factor and the static probability, such that the activity factor is restricted in the shaded region.

In order to properly generate input data packets according to the boundaries fixed by the glitchy or not glitchy nature of the generated signals, the generation script that implements the algorithm is depicted in Figure 4.9. At first, an activity factor $AF[n]$ is randomly generated within the range $[0, 2]$. Then, if the generated $AF[n]$ is less than or equal to 1, the signal is classified as **glitch-free** and hence its boundaries are defined by Equation 4.8 to randomly generate the static probability $P_1[n]$.

$$(AF[n], P_1[n]) := \begin{cases} 0 \leq AF[n] \leq 1 \\ \frac{AF[n]}{2} \leq P_1[n] \leq 1 - \frac{AF[n]}{2} \end{cases} \tag{4.8}$$

Otherwise, the signal is classified as **glitchy** and thus Equation 4.9 is considered during $P_1[n]$ generation.

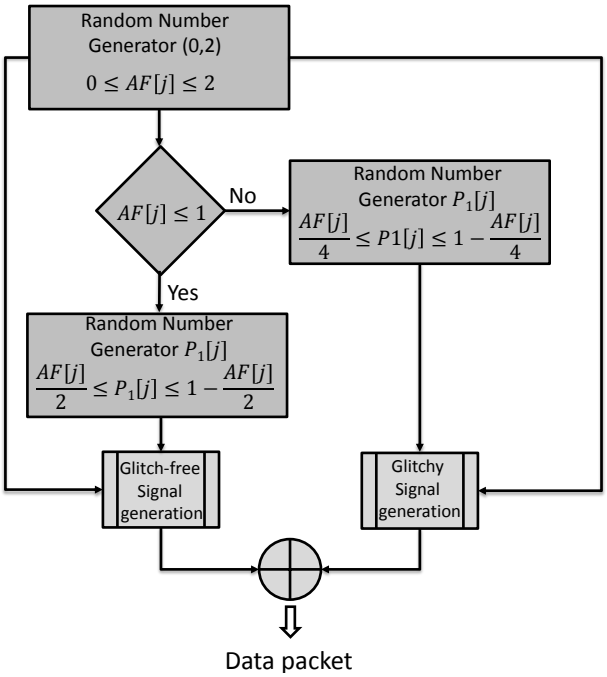



Figure 4.9 – Glitchy and glitch-free data packet/block stimuli generation flow.

$$(AF[n], P_1[n]) := \begin{cases} 1 < AF[n] \leq 2 \\ \frac{AF[n]}{4} < P_1[n] \leq 1 - \frac{AF[n]}{4} \end{cases} \tag{4.9}$$

Finally, the boundaries of the characterization parameters are discussed, and are used to characterize the power and the behavior of a component. Indeed, glitchy signals generated by a component are responsible for a significant amount of power consumption in the full digital design. To this end, the introduction of glitchy packets/blocks in the dataset that are adopted for the characterization phase have a direct impact on the estimation accuracy of our proposed method.

### 4.3.3 Component Definition and Waveform Generation

In Figure 4.10, an example of a standalone model design of a generic component is described. This component can represent any combinational circuit such as an adder, multiplier, subtractor, a register, multiply and accumulate, etc. Regarding the component's inputs and outputs, they are registered to isolate the module and confine the combinational path.

Based on the described component and using the discussed parameters (See section 4.3.1), we perform the component's characterization. Figure 4.11 shows the activity generator block that takes the number of component's inputs $N$, the number of samples or packets $M$, and the needed boundaries as inputs. Based on these inputs, two files are generated: the first is the activity factor ($AF$) file,

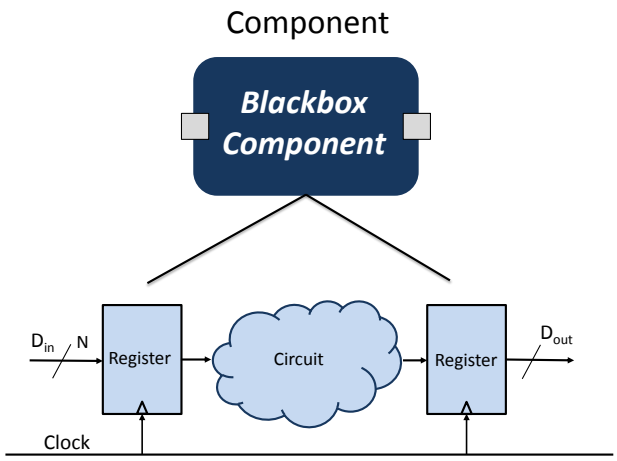



Figure 4.10 – The characterized component with N inputs

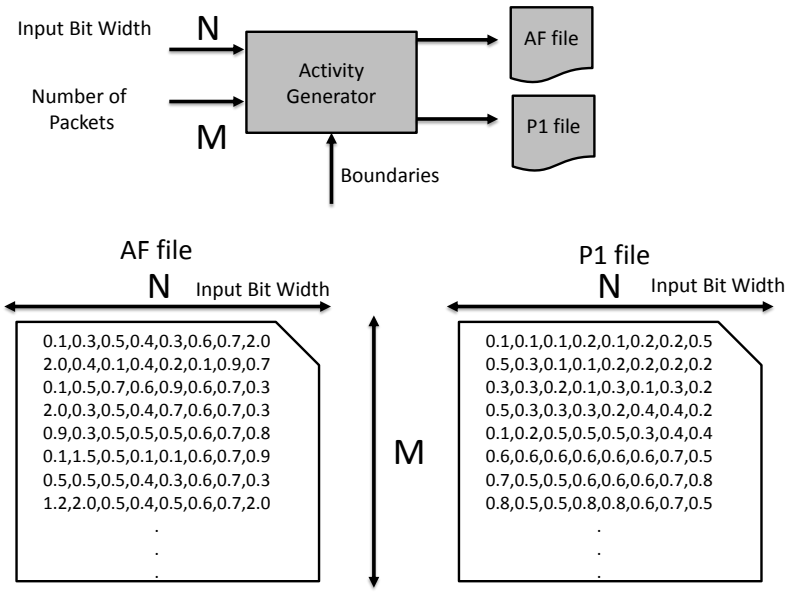



Figure 4.11 – Activity factor and static probability generation files preparation

and the second is the static probability ($P_1$). In each of the following files, the row represents the activity factor of all bits (or the static probability) of the $m$-th sample. However, the column represents the activity factor (or the static probability) of the $n$-th input bit.

Based on the generated files, waveforms are generated to conduct characterization at a low-level of abstraction. Figure 4.12 shows the block packets generator, that is used to generate the $M$ waveforms/packets. This block takes the activity factor and the static probability files, the total number of component's input $N$ (input bit width), and the sequence length or the packet length $L$ (time window),

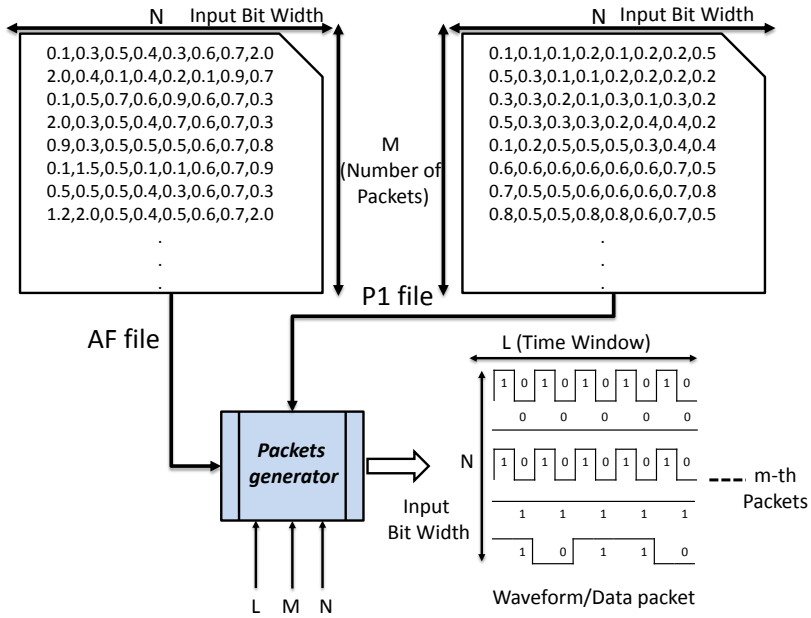



Figure 4.12 – Waveform generation block

and the total number of packets $M$ as inputs. Regarding the output, the block packets generator generates the needed waveforms for the characterization phase.

### 4.3.4 Power Characterization

Power characterization is the step where the power consumption profile of a given component is extracted according to the provided input data stimuli (waveforms). Power consumption values are gathered using the synthesized component (netlist) obtained through the synthesis tool using the component RTL description as input, the generated timing information of the component (SDF) file, and the considered waveforms (See Figure 4.13). Note here that a power estimation engine is capable of providing the corresponding power consumption for each combination of the waveforms for the component under characterization.

Power characterization is performed in two steps: the first is a post-synthesis timing simulation step which is performed to compute the internal signals activities that are propagated to the component outputs as function of the injected input waveforms. In addition to the input waveforms, this step also requires the netlist of the component and the related timing information. Secondly, the switching activities of the blackbox component (input, internal and output activities), computed during the post-synthesis simulation, are injected into the power estimation engine tool. In this regard, the usage of the netlist and the related timing libraries ensure that the delays of the different cells instantiated in the netlist are correctly taken into account, leading to more accurate power characterization results. Note that, the power estimator provides one scalar power value

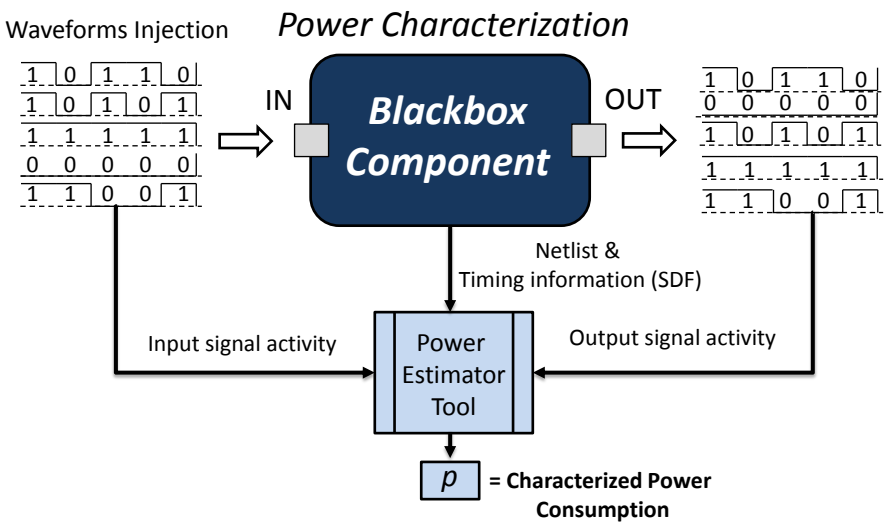



Figure 4.13 – Detail of power characterization

for each input data waveform. This obtained power value represents an average power consumption, which is calculated during $L \times T$ seconds, where $L$ and $T$ represent the waveform length and the clock period, respectively.

### 4.3.5 Behavioral Characterization

Behavioral characterization is intended to provide data that represent the internal behavior of a component under characterization. For this purpose, the output waveforms are used to gather information about the internal behavior. We may consider any component as a blackbox component, and to characterize it without the need of knowing the internal architecture. In this regard, the waveforms (data stimuli) are injected to the component's inputs, and then the post-synthesis timing simulation is executed in order to obtain the output waveforms as depicted in Figure 4.14.

In particular, for each output bit, this phase evaluates the activity factor $AF$ and the static probability $P_1$ features for a given input data stimuli, here referred also as packet or waveform. Recall that the $AF$ is related to the number of bit transitions during the simulation, while $P_1$ corresponds to the probability of each bit to be set to one during simulation. Both $AF$ and $P_1$ are given per bit so that the whole component's inputs/outputs are represented by vectors of features. The conversion from waveforms (matrices of zeros and ones) to feature vectors is necessary at the output side (See bottom of the Figure 4.14), as will be explained in details in the next section. For instance, the upper side of the vector $V_{out}$, represents the activity factors ($AF$) for each bit starting from the most significant bit (MSB) to the least significant bit (LSB). Regarding the lower part, it represents the static probability ($P_1$) for each bit (from MSB to LSB).

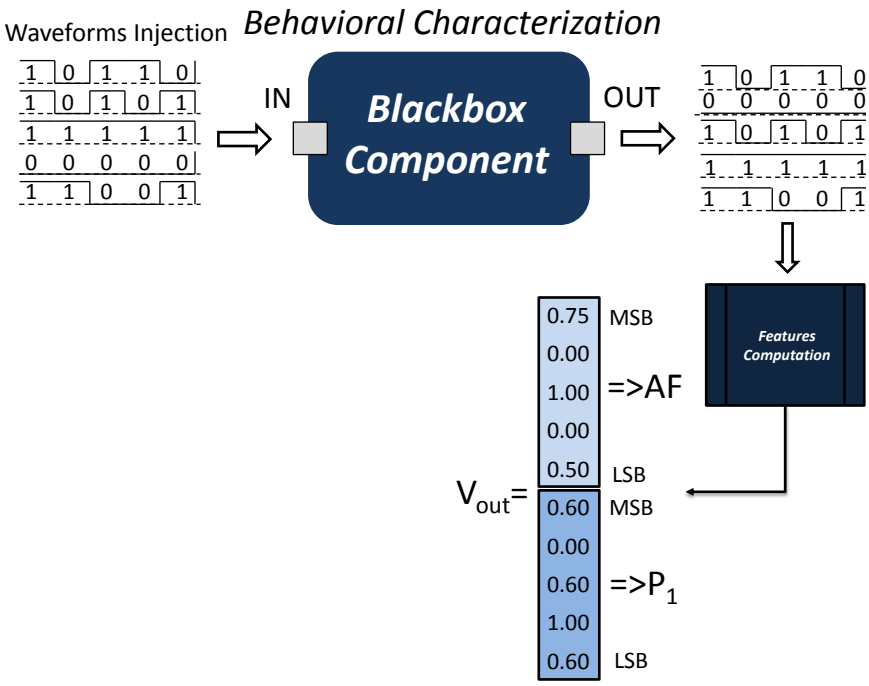



Figure 4.14 – Detail of behavioral characterization for a blackbox component

To summarize, the behavioral characterization is performed by means of a post-synthesis simulation, while taking the input data stimuli, as well as the netlist of the component and the timing information about the involved cells and connections (SDF file plus timing libraries) as inputs. As for the output signals, they are available at the end of the simulation, as matrices of zeroes and ones (see Figure 4.14). These are then processed in order to extract $AF$ and $P_1$, which are necessary for the behavioral modeling for each component model (more details are given in section 4.4).

## 4.4 Modeling Methodology

In this section, we present the proposed methodology that exploits the results obtained from the described characterization phase. In this respect, power characterization is suggested to provide the needed information to construct the power model. Moreover, the characterization of the component's behavior is intended to deliver the required information to create the component's behavioral model. On one side, the power model is responsible for mapping the input characteristics such as the activity factors $AF$ and the static probability $P_1$ to the power consumption of the component. On the other side, the behavioral model is accountable for mapping the input characteristics to the output's characteristics.

We adopt here two distinct models for each considered component: one model

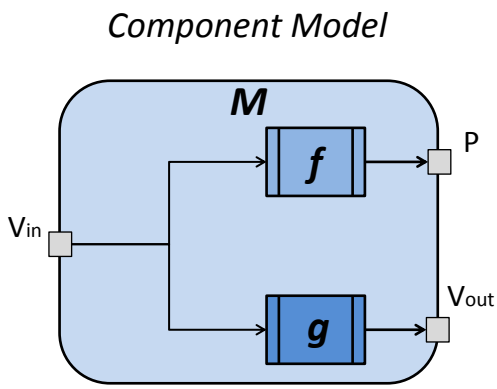



Figure 4.15 – Component's model

for estimating power and another for determining the propagation of the activity. In order to prepare these models properly, the components need to be deeply characterized in order to provide a representative dataset. For instance, Figure 4.15 illustrates a component model, named as $M$, which consists of two sub-models: the portrayed model $g$ is the one that accounts for the behavioral model and the model $f$ estimates the component's power. More specifically, model $g$ has to know the signals propagation processes from the inputs to the outputs of the component, in terms of the corresponding characteristics embedded in the propagated vectors $Vin$ and $Vout$. As for the model $f$, it has to know the interactions between the dynamic power usage $P_{dynamic}$ and the input information features vector $V_{in}$ of a given data. The relationship introduced by the component's model $M$ (behavior's and power's models) can be expressed in Equation 4.10.

$$M : \begin{cases} f : V_{in} \rightarrow P_{dynamic} \\ g : V_{in} \rightarrow V_{out} \end{cases} \tag{4.10}$$

### 4.4.1 Power Modeling

In this part, we outline $f$, the power model, which maps the $V_{in}$ vector to the dynamic power consumption $P$. As shown in Figure 4.16, to each input's vector, an output corresponds and represents the average component's dynamic power consumption. To solve this modeling problem, we rely on the universal approximation theorem, which states that a feedforward network with one hidden layer containing a finite number of neurons is capable of approximating generic nonlinear continuous functions [107, 108]. Based on this, the power model $f$ can be approximated by means of artificial neural networks with one hidden layer.

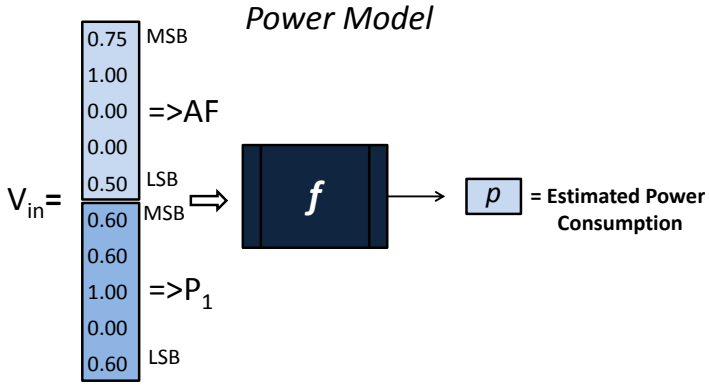



Figure 4.16 – Power model $f : V_{in} \rightarrow P_{dynamic}$

As discussed in section 2.4, ANNs generally require a representative dataset to perform efficient modeling. This representative information set, i.e., a large and sufficiently varied set of data, indicates a profound characterization of the components. In general, the dataset is divided into training, validation, and test sets. Since the dataset is partitioned into different subsets, only the training datasets are used to specialize the ANN models in order to let them learn the trend and fit with the desired functionality. The addressed problem here is a multi-dimensional mapping problem, where the ANN models have multiple inputs and outputs. Moreover, it is a problem that deals with continuous data, since $AF[n]$ can assume any value between 0 and 2, $P_1[n]$ can take any value between 0 and 1. Note that, we adopt a two layer feed forward ANN model with sigmoid neurons in the hidden layer and linear neurons in the output layer, as shown in Figure 4.17:

— the input layer $x_n$, where its size is equal to the number of inputs of the model, and thus equal to twice the bit width ($2N$) of the input signals of the corresponding component; note that with regard to inputs, we combine the activity factor $AF$ and the static probability $P_1$ for each input bit with the neural network input as shown in Equation 4.11.

— the hidden layer $N_j$, that involves all the sigmoid neurons whose number is usually set empirically; note that the model's feature is all focused within the hidden layer containing sigmoid neurons.

— the output layer, which is composed of linear neurons, such that its number is fixed by the expected number of outputs of the specific model. For example, in the power model, only one output is needed; with respect of the model output, we associate the power consumption with the neural network output as follows $O_1 = P_{dynamic}$.

$$x_n = \begin{cases} AF[n] & \text{if } 1 \leq n \leq N; \\ P_1[n-N] & \text{if } N+1 \leq n \leq 2N; \end{cases} \tag{4.11}$$

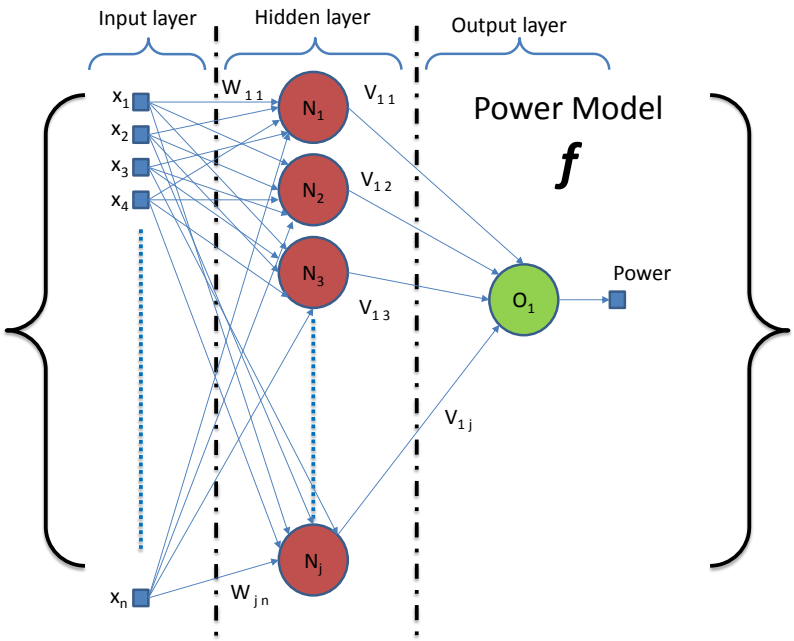



Figure 4.17 – Adopted artificial neural networks power model $f$.

where $n$ goes from $1, ..., 2N$ and $N$ is the number of bits of the component inputs.

### 4.4.2 Behavioral Modeling

In this part, we present the modeling phase consisting of building the relationship between the input and the output features corresponding to the primary inputs and outputs respectively. Hence, we describe the behavioral model $g$ that is responsible for giving information about the propagation of signals from the inputs to the component's outputs. The behavioral model's input is the same as the power one $f$, knowing that the $V_{in}$ features vector is related to the primary inputs bits (See Equation 4.11). As for the model's outputs in this case, it is no more scalar but a vector of features $V_{out}$ that corresponds to the output bits, as shown in Figure 4.18.

As it is illustrated in this figure, both power and behavioral ANN models have the same number of inputs. However, the difference between the two ANN models resides in the output layer, and in particular, in the number of outputs. Recall that the power model $f$ takes as input the vector of features $V_{in}$ that corresponds to its input bits, and gives as output a scalar value $P_{dynamic}$. This value expresses the dynamic power consumption which depends on the combination of inputs. As for the behavioral model $g$, it delivers an output that can be defined according to Equation 4.12 and Figure 4.19.

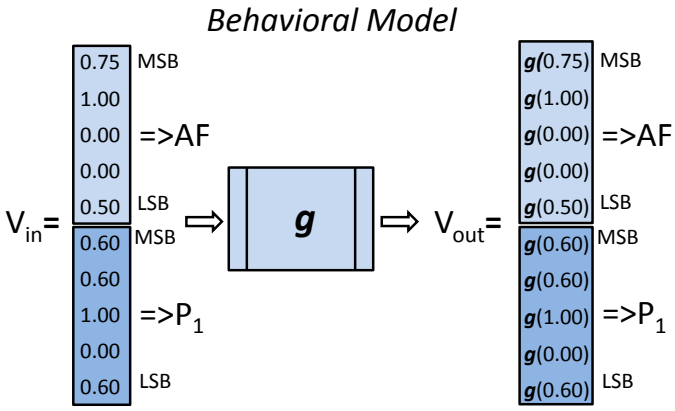



Figure 4.18 – Details of the behavioral model $g : V_{in} \rightarrow V_{out}$.

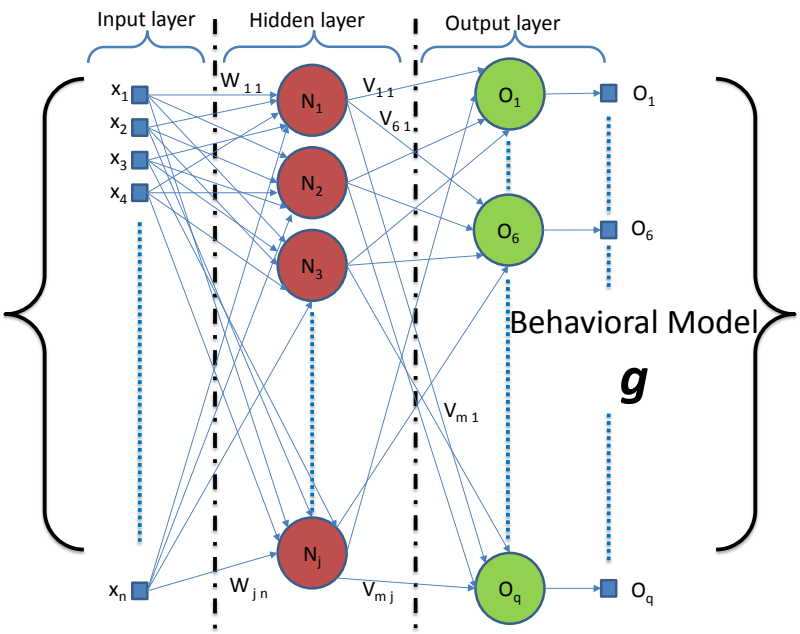



Figure 4.19 – Adopted artificial neural networks for the component's behavioral model $g$

$$O_n = \begin{cases} AF[n] & \text{if } 1 \leq n \leq Q; \\ P_1[n-Q] & \text{if } Q+1 \leq n \leq 2Q; \end{cases} \tag{4.12}$$

where $n$ goes from $1, ..., 2Q$, and $Q$ is the number of bits of the component's outputs.

### 4.4.3 Library of component's model construction

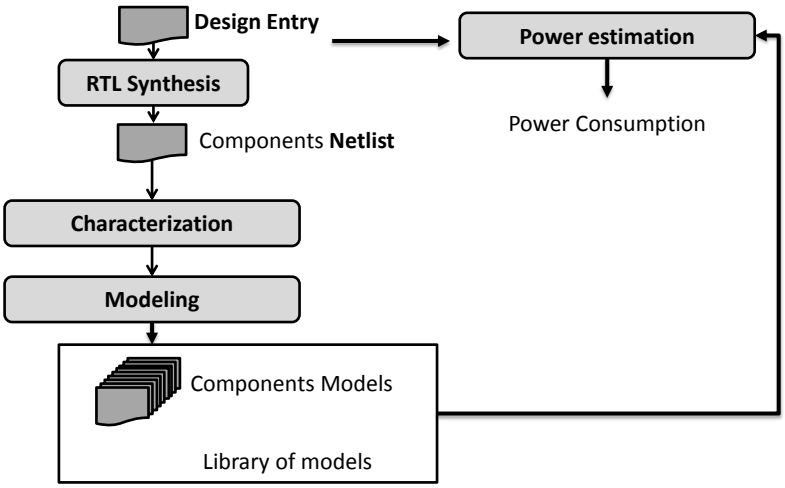



Figure 4.20 – Off-the-shelf power models for high-level power estimation

Based on all the previous sections, we can conclude that the characterization and the modeling phases are very important in preparing the high-level power models. This is because these models are used to explore the design space in terms of power consumption. Therefore, it can be assumed that any digital design can be composed of a set of components, and hence, it is possible to provide a library of models that represents the basic components, in order to compose any digital design. The proposed library of models involves many different components. To validate and test this library, we synthesized, characterized, modeled and analyzed all these components, with a computer aided design tool that considers an operating frequency $F_{clock}$. Then, this library of models is used as an off-the-shelf solution due to the ease of use and plug-and-play models in order to accelerate the power estimation flow of a full design at high-level of abstraction as depicted in Figure 4.20.

## 4.5 Method Extension

In digital circuits, dynamic power consumption depends on several parameters, such as: 1) the activity factor, 2) the static probability, 3) the operating frequency, and 4) the technology used. Our method has originally built while taking into consideration the first two metrics at a fixed operating frequency ($f_{clock} = 100\ MHz$) and targeting a certain technology on ASICs and FPGAs. For this reason, we study a more complex model that improves the component's model, by taking into account the system operating frequency while estimating the power consumption.

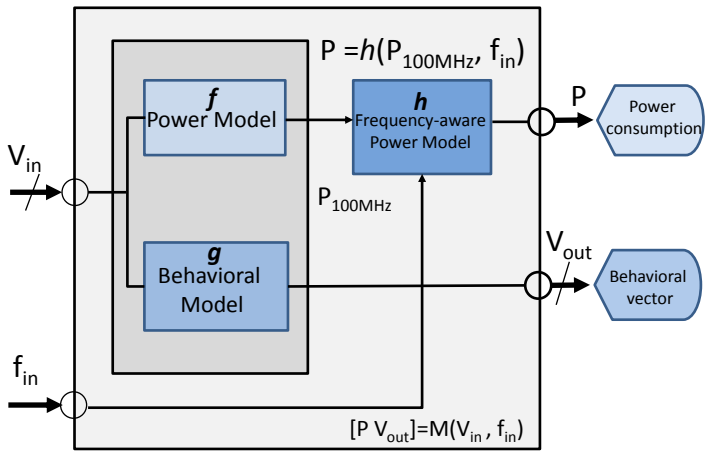



Figure 4.21 – Proposed Frequency-Aware Power Model

### 4.5.1 Component's Model Extension

Figure 4.21, illustrates our new proposed model that we call Frequency-Aware Power Model. To build this Frequency-Aware Power Model, we adopt the power model $f$ as our baseline, while taking all the steps that are related to the characterization, the modeling and the library construction phases into account, but with doing some modifications. The power characterization is performed here at different frequency points, leading then to different dynamic power consumption values. The frequency parameter $f_{in}$ is thus added as a new input to the power model, and the output features vector $V_{out}$ of the behavioral model is forwarded to the successive component. Regarding the output of the power model $f$, that constitutes one of the input of a second ANN power model denoted by $h$, taking as input also the operating frequency $f_{in}$. The characterization of the new ANN power model $h$ is made using the power values resulting from $f$ (trained with the same input features as used before to characterize the power) and by the frequency $f_{in}$, so that it can be possible to predict the new dynamic power consumption taking into account the frequency points. The improved model $M$ is shown in Equation 4.13. Then, the new power model $h$ is also based on ANN, and it maps the dynamic power consumption $P_{100MHz}$ computed at $100MHz$ from the previous model $f$, and the input frequency $f_{in}$ to the corresponding, frequency-aware, dynamic power consumption $P$.

### 4.5.2 Upgraded System-level Power Estimation

The power estimation strategy is expanded and new parameters such as frequency are added to the estimation method. To this regard, a composite system's power consumption can be achieved taking into account activity factors, static

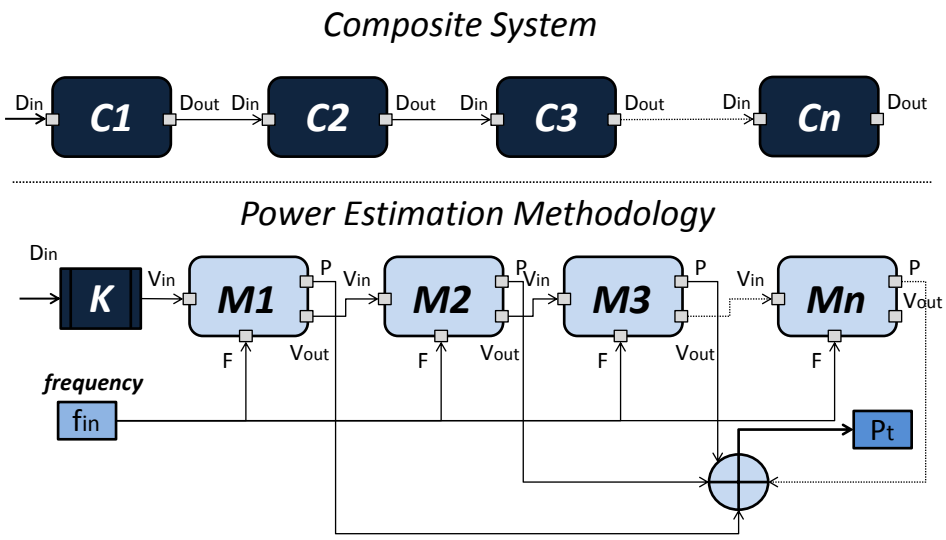



Figure 4.22 – Power estimation method with the extended power model

probabilities, and operating frequencies as shown in Figure 4.22.

$$M : \begin{cases} f : V_{in} \rightarrow P_{100MHz} \\ g : V_{in} \rightarrow V_{out} \\ h : P_{100MHz}, f_{in} \rightarrow P \end{cases} \tag{4.13}$$

Knowing that the power consumption and the output switching activity of a given component depends on the input switching activity of its predecessors. It is possible to express the total dynamic power consumption of a full design as follows:

$$P_t = h_1(V(1), f_{in}) + \sum_{j=2}^{j=n} h_j[g_{j-1}(V(j-1), f_{in})] \tag{4.14}$$

where $j$ is the index of the component's model that ranges from 1 to $n$ components.

## 4.6 Conclusion

In this chapter, we have presented a novel power estimation methodology, based on characterization and modeling stages, while relying on machine learning techniques. The purpose of the first stage, i.e., the characterization one, is to prepare the suitable information that will be used in the modeling one. As for the

modeling stage, it necessitates two models, the power model and the behavioral one. The former predicts the power consumption, while the latter estimates the activity factors and the static probabilities at the component's outputs. Note that the importance of the behavioral model appears in making the model more scalable. By this, the complex systems can be composed of connecting the library of components and estimating the respective power consumption. Such estimation leverages the behavioral models to propagate the signal activity across the different components. The power models are then capable of offering a system-level power estimation by taking the input signal activity from the system inputs or from prior components. To make our method even more general, we also provide in this chapter an extension of the power model which allows to evaluate the power consumption for distinct design topologies at different operating frequencies. Finally, the aim of all of this is to propose a power estimation strategy at a very early design stage. Before starting the verification of the proposed method above, some preliminary results that motivate the use of the signal statistics propagation to estimate the power consumption of the composite system are provided in Section A.1.1. In addition, the numerical results that bring the use of neural networks as a generic modeling technique are discussed in Section A.1.2. In the following chapter, we will discuss the results that will numerically validate the efficiency of the proposed power estimation method.

# Chapter 5

# Numerical Results for FPGA/ASIC Power Estimation Strategy

## 5.1 Introduction

In this chapter, we present the numerical results that motivate the use of our proposed power estimation method. To verify its efficiency, we deliberately chose reconfigurable devices such as FPGAs, that are widely used in digital signal processing systems. We also present the overall (or also called system-level) power estimations of different designs. To evaluate the accuracy of our power models, we perform a comparison against the classical tools that are based on computer aided design tools, such as the Xilinx XPower Analyzer (XPA) tool.

Moreover, we also apply our power estimation method on semi-custom ASICs. We adopt neural networks in order to model the power of digital components for $45nm$ and $15nm$ ASICs technology. We prove the effectiveness of the proposed methodology by performing different numerical verification at different levels. Additionally, we lead several technology, scalability and estimation time assessments to demonstrate that our estimation method is scalable, fast and robust in terms of system-level power estimations.

## 5.2 NeuPow Assessments on FPGA

In this section, we present the results that help in verifying the efficiency of our proposed method, while applying it on FPGA platform and dealing with different case studies. In this part, we propose to use our library of component's model, which is based on the artificial neural networks. The main concept consists of estimating the power of a composite system, based on an accurate power estimation of its sub-components. Each component (available in a dedicated library) has been fully characterized, producing then accurate models. By this, the designers have the possibility to construct their architecture by connecting

components in a design entry tool that is similar to schematics. The full design may then be simulated at a high-level to get the power consumption.

In the following sections, we focus on describing the characterization and the modeling phases that aim to develop the power models of different components. To this end, we describe first the per-component power characterization and the different modeling stages. Then, we present the system-level power estimation evaluation. To demonstrate the feasibility of our approach and to quantify the model's accuracy, we perform several case studies on different components. In this part, we implement a full design composed of basic components (adders and multipliers) on a xc7z045ffg900 FPGA device, and perform two types of power estimation: the first corresponds to the classic power estimation that is achieved in the last steps of an FPGA design flow (after place and route or also called the implementation step). Regarding Xilinx FPGAs, this step is implemented using Xilinx Xpower Analyzer. The second has been performed using our neural networks models. In this case, the simulation is performed using high-level tools such as Matlab/Simulink. The idea is to guarantee a fair comparison between the results obtained by our system-level estimations and Xilinx tools. Both power estimation types have been achieved on two types of designs. The first consists of a single component (adder or multiplier), and the second is composed of a more complex combination of these components, i.e, a composite function. This aims to demonstrate the use and the propagation of signal rates throughout a full design.

### 5.2.1 Characterization and Modeling: Per-Component

In order to prepare the data that are used to model power consumption, a characterization phase has been performed. This phase consists in implementing the component on a given FPGA target and obtaining the power consumption values after applying various configurations of switching activities and percentage logic high at the component's inputs. Note that this process is performed at a low-level of abstraction (after implementation) in order to get as much technical details as possible and to obtain the best accuracy in terms of power estimation. Tools such as Xilinx Xpower analyzer are used to build huge datasets that take all ranges of switching activities and percentage logic high for every component's inputs into consideration. For more details, we describe here the power characterization process in Figure 5.1. A TCL script reads the data samples file and sends them to the Xilinx Xpower Analyzer tool to execute the following two tasks: 1) to perform power estimation, and write back the values of estimated power into a file, and 2) to accomplish the behavioral estimation, where the switching activities and the percentage high for each signal are extracted from the Xpower at the component's outputs and then written into a separate file.

Then, the power modeling phase uses the data obtained from the recorded two files during the characterization phase. To this end, we perform the power

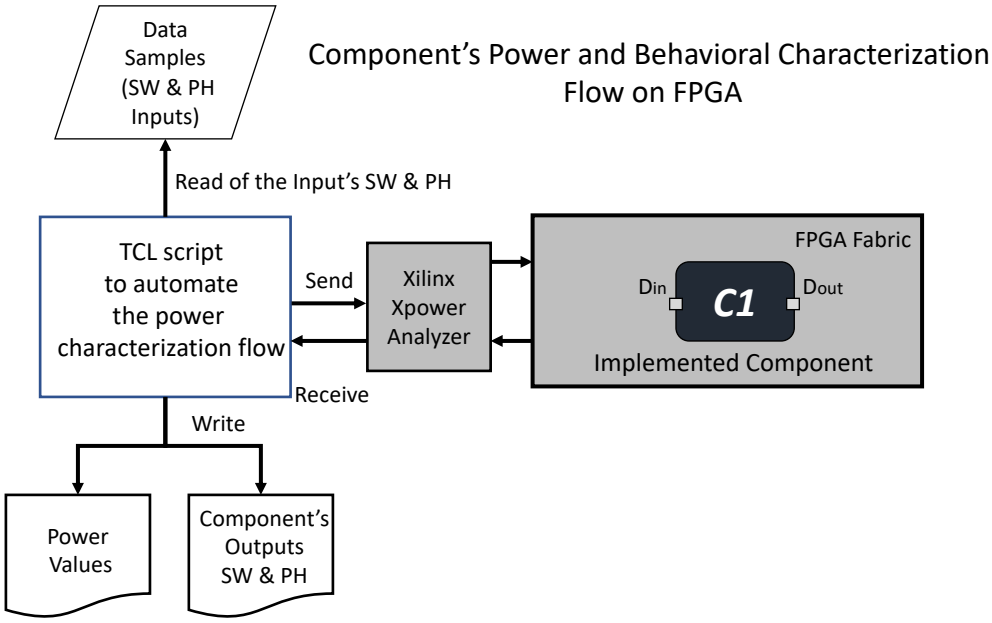



Figure 5.1 – Component's power and behavioral characterization flow on FPGA

modeling using the power consumption data, while we adopt the components' outputs switching activities and percentage high for the behavioral modeling. At the component-level, a verification phase is performed to calculate the accuracy of the proposed models. As shown in Figure 5.2, Xilinx XPA has been used to estimate the power consumption of a 16 bits adder and a 16 bits multiplier. Then, for the same components, the Matlab tool has been used to estimate the dynamic power consumption using the component's model that is based on neural approach. Both $f$ and $g$ neural networks (power and behavioral) have thus 16 (bits) x 2 (inputs) x 2 (signal rates + PLH) = 64 inputs, and are grouped in a single block model (Neural Network Operator Model) as depicted in the right side of the Figure 5.2. Note that the left side represents the implemented component in the FPGA platform. The modeling phase for both models is performed with 100 and 150 neurons, respectively, in their hidden layer. Regarding the training set of each neural network, it consists of 64x10000 data-samples (80% for training, 10% for validation and 10% for testing) with different combinations of switching activities and percentage high. At the output of the $f$ neural model, only one power value is provided. The $g$ neural model provides 16 (bits)x 2 (signal rates + PLH) = 32 outputs to propagate the signal activity and the static probability to other subsequent models.

To evaluate the accuracy of the power models, we have provided 10000 sample combinations of switching activities and percentage high (i.e., M=10000) for both, i.e., the 16 bits adder and the 16 bits multiplier. Then, we run 10000 simulations to extract the dynamic power consumption using Xilinx XPA. The

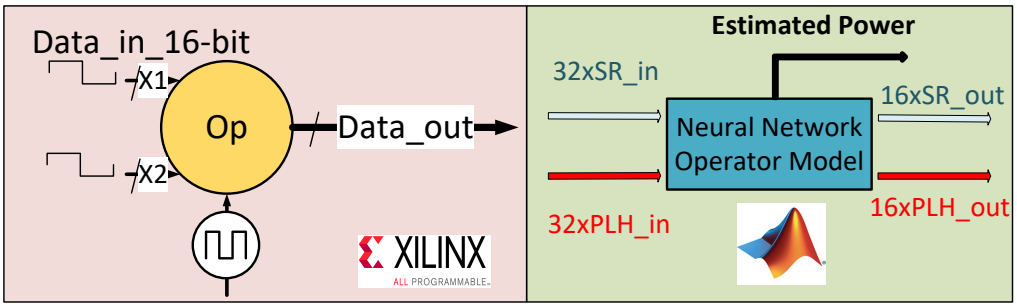



Figure 5.2 – Xilinx XPA Vs Neural Model Estimations

same exact inputs that were given to Xilinx XPA were also provided to the neural models to evaluate the power consumption as well as the switching activities and percentage high. Table 5.1 shows the modeling results, and mean relative error at the component-level in terms of power estimation is evaluated. Finally, the mean relative error has been calculated according to new 10000 sample combinations. The results shown in this table indicate a relative error that is very small (around 0.01%). This shows that neural networks have learned the behavior of XPA and are able to model it with a very good precision.

Table 5.1 – Accuracy of the power models comparing to the Xilinx XPA

| Components | Power Estimation (XPA) (mW) | Neural Model (mW) | Avg. Relative Error (%) |
|---|---|---|---|
| Adder($16 \times 16$) | 1.7879 | 1.7881 | 0.0112 |
| Multiplier($16 \times 16$) | 27.4147 | 27.4136 | 0.0040 |

### 5.2.2 Power Estimation: Per-System

At the system-level, a verification of several scenarios has been performed. For instance, Figure 5.3 describes a function that is composed of different components. The left side of the Figure 5.3 represents the implemented function in the FPGA, while the right side reflects the corresponding modeling block, which is based on our developed library of neural models. As in the per-component approach, Xilinx XPA has been used to evaluate the power of the composite system as well as the switching activities and the percentage high ($SR$ & $PLH$) parameters. In parallel, a full system for the power estimation has been modeled using the components' models.

For both estimations, new inputs combinations of 10000 data-samples have been provided to both Xilinx XPA and neural models. These 10000 data-samples ($M = 10000$) of dynamic power values have been extracted from the Xilinx XPA (See left part of Fig. 5.3) for the full design and from the full neural models (See

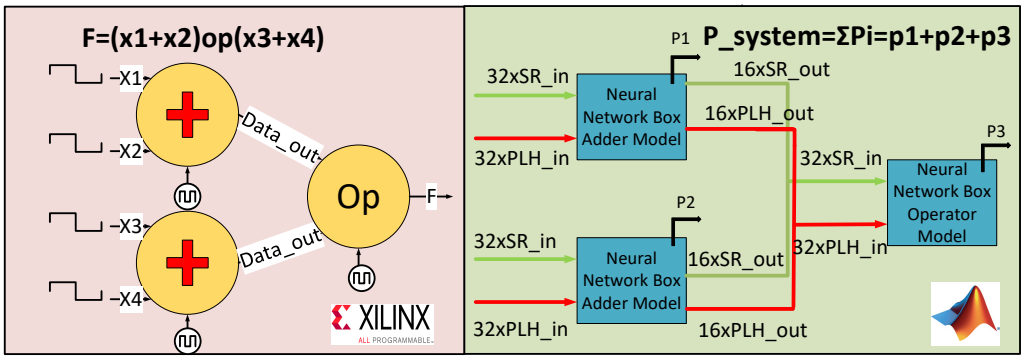



Figure 5.3 – System-level power estimation Xilinx XPA Vs Neural Model

right part of Fig. 5.3). Note that the two functions have been implemented as follows: $F_i = (a + b)$ op $(c + d)$, where *op* can either be an adder or a multiplier. As in Eq. 5.1, a mean absolute percentage error ($\%MAPE_{System-level}$) has been calculated at the system-level to evaluate the accuracy of our proposed method:

$$\%MAPE_{System-level} = \frac{100}{M} \sum_{i=1}^{i=M} \frac{|P_{XPA} - P_{NN}|}{P_{XPA}} \tag{5.1}$$

where $P_{XPA}$ is the power consumption obtained using the Xilinx XPA tool, and $P_{NN}$ is the power consumption estimated by the power models of the composite system. According to the results captured in Table 5.2, it can be seen that our method provides a good accuracy at the system-level with an error that is less than 8%.

Table 5.2 – Models Accuracy for Different Functions

| Arithmetic Function | Neural models (MAPE %) |
|---|---|
| F1(op1=+, op2=+, op3=+) | 7.3699 |
| F2(op1=+, op2=+, op3=*) | 3.5158 |

It is also possible to estimate the power consumption based on the developed power models only. Table 5.3 shows the initial estimates that consist of the sum of the total average power of each component. These estimations are obtained using the component's power models $f$, without considering the behavioral models $g$ (no propagation is performed). We notice that these estimates exhibit low accurate results, with a relative error that is greater than 15%. For this reason, considering both models (power and behavioral) helps to estimate the power consumption at system-level with a more accurate way. Moreover, we may identify many sources of errors in the proposed estimation methodology. First, the switching activities and the percentage high modeling errors, that are propagating and being accumulated through the full design. Second, in Xilinx XPA, the fact that the

Table 5.3 – The effect of Signal Activity Propagation Versus Initial Estimates

| Arithmetic Function | XPA-based (mW) | Initial Estimates (mW) | Relative Error (%) |
|---|---|---|---|
| F1 | 4.6282 | 5.3637 | 15.8965 |
| F2 | 70.2986 | 82.2441 | 16.9925 |

components are connected to several neighbors, which has a direct impact on the capacitance, and thus on the power consumption of the interconnections. Note here that the power consumption of the intra-connections (within a component) is taken into consideration in the component's power model.

In the following, we compare our approach to the classic FPGA design flow that makes it possible to obtain an accurate power estimation of a full design. In order to underline the impact of the design productivity in terms of power exploration effort, let us first consider that the designers decide to start implementing arithmetic function $F1$. The designers would like to evaluate the overhead that is required to obtain new power estimation results if the design is modified to implement another function, e.g., function $F2$. Using the Xilinx classic design flow, designers have to re-run the complete flow and run the Xilinx XPA estimation tool for the new $F2$ version. In this case, the design entry has to be modified accordingly and the synthesis as well as the implementation steps have to be executed. In addition to this, designers have to run the Xilinx XPA power estimation tool to get new power estimations. Although quite fast in this simple example, it is important to note that these steps may become prohibitive when dealing with complex systems composed of a large number of components. Changing a single component requires then to run the complete flow for the modified design. With our approach, designers need only to modify the instances in the design entry tool and perform high-level simulations using the developed component's neural models delivered in a dedicated library. Here, no additional steps are necessary. Furthermore, power estimation is completely integrated in the simulator tool that can perform complex simulations in a very short time (less than few seconds). The different design steps that need to be followed are summarized in Table 5.4. To conclude, although our proposed method shows an estimation error of about 8%, however, it still improves the design's productivity.

Table 5.4 – Required design steps to switch from $F1$ to $F2$ configurations

| Design steps | Xilinx Vivado | Proposed Approach |
|---|---|---|
| Design entry | yes | yes (very easy) |
| RTL synthesis | yes | not required |
| Implementation | yes | not required |
| Power Estimation | yes | yes |

## 5.3 NeuPow Verification on ASIC

In this part of the thesis, we present the results of our proposed estimation methodology for both, the component- and the system-level on ASIC design. Estimation results are gathered after adopting a dataset that is generated by the stimuli generator script and composed of 10000 data packets ($M = 10000$ samples according to the parameters introduced in Section 4.3.3). These samples are randomly divided into: 80% for training, 10% for validation, and 10% for testing. For all the reported numbers, we adopt available Cadence®tools: Genus Synthesis Solution for the RTL synthesis of the components, Incisive Enterprise Simulator for post-synthesis simulation required for the characterization phase, and, again, Genus Synthesis Solution for power analysis purposes. Besides being used during characterization phase, the data resulting from these tools constitute also the term of comparison for most of the results, as will be shown in the following sections. Unless explicitly specified in a specific section, the considered operating frequency for all the experiments is 100 MHz, and the target ASIC technology library is the Cadence®GPDK 45 nm.

In Section 5.3.1, we describe the use cases that are used to validate our proposed methodology. Section 5.3.2 shows the impact of the glitches on the power estimation by comparing two estimation scenarios. In Section 5.3.3, the results about ANN modeling are reported, showing how the ANN models have been dimensioned. Section 5.3.4 instead shows several assessment numbers of the proposed methodology at the component-level, prior to go into the system-level assessment with Section 5.3.5. In addition to this, it demonstrates the effectiveness of the proposed power modeling method based on neural networks technique relative to the regression-based technique. Finally, Section 5.3.6 shows a preliminary evaluation on a more recent technology, such as the 15nm FinFET-based Open Cell Library [109].

### 5.3.1 Use Cases Description

In this section, we describe the basic library of components we have characterized and used in all the conducted experiments. The library is composed of the following components: Multipliers (Mult), Adders (Add), Subtractors (Sub), Square, Multiply-accumulate (Mac), Register (Reg) and Shift and Clip (S&C), as depicted in Figure 5.4. Note that this library is used as a foundation for all the use cases that will be explored in the different sections below, namely: 1) the complex multiplier, 2) the magnitude square calculator and 3) the image filter, which is explored with 4 different design architectures. Hence, the use cases are shortly presented hereafter:

— **The complex multiplier** is used in several digital signal processing algorithms. Considering two complex numbers $Z_1$ and $Z_2$, it is possible to

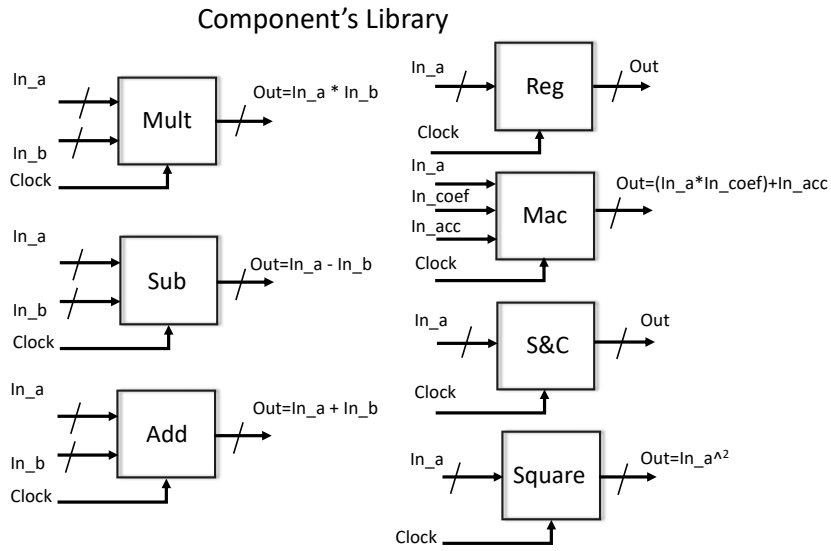



Figure 5.4 – Library of components described at RTL

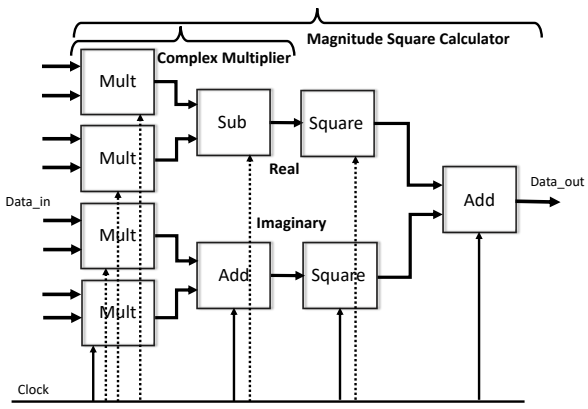



Figure 5.5 – Complex multiplier and magnitude square calculator

express these numbers as $Z_1 = a + ib$ and $Z_2 = c + id$, where $a$ and $c$ represent the real part, while $b$ and $d$ are the imaginary parts. The complex multiplication can be then written as $Z_1 \times Z_2 = (a \times c - b \times d) + (a \times d + b \times c)i$. We consider a simple complex multiplication that can be performed here using four $4 \times 4$ multipliers (Mult), one $8 \times 8$ subtractor (Sub) and one $8 \times 8$ adder (Add) (See Figure 5.5).

— **The magnitude square calculator** is used to compute the magnitude square of the resulting complex number after a complex multiplication, which can be expressed as $Real^2 + Imaginary^2$ (see Figure 5.5). Thus, it is composed of the same components of the complex multiplier (4 Mult, 1 Sub and 1 Add) plus two components to calculate squared real and imaginary

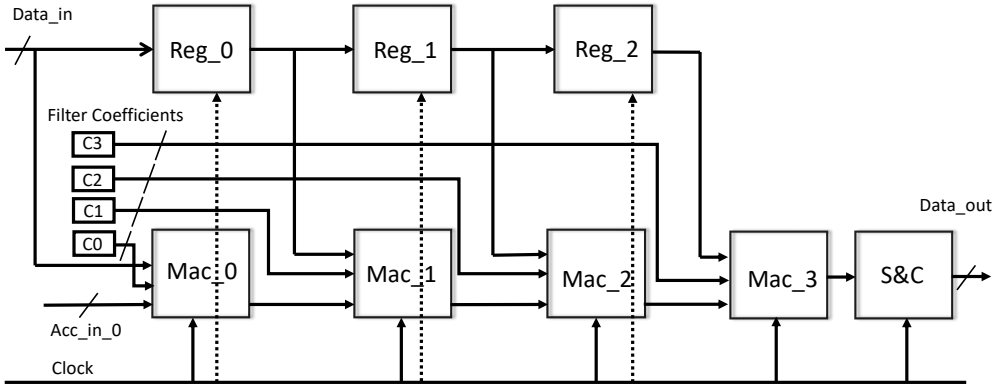



Figure 5.6 – Image filter (4layers/taps)

part (Square) and one final adder to compute the magnitude (Add).

— **Image filter** (see Figure 5.6) is a one-dimensional convolution filter adopted in different image and video processing applications. Basically, its composition depends on the number of taps (convolution kernel size): each filter stage stores one value (Reg) and computes a multiplication between the input data and the corresponding kernel coefficient, and an accumulation of the multiplication result and the result coming from the previous stage (MAC). In order to keep the result into meaningful or practical values for an image (0-255), a final shift and clip (S&C) is required.

### 5.3.2 Glitch Impact on Power Estimation

In this section, we discuss the results that highlight the importance of the glitches during the power estimation process. To illustrate this, Figure 5.7 shows the glitches at the component's outputs. The glitchy signals are represented by the signals that switch more than one time during one clock period such as the signals that switch 2 times during the clock period. As discussed in Section 4.3.1, the glitches appeared when the components' inputs are injected by stimuli data waveforms. We highlight the results of the glitches effects on the complex multiplier case study. First, glitch-free packets are used during the characterization and modeling phases for all the considered components. Second, we introduce the glitchy packets in the dataset adopted for for characterization and modeling (training set) that has a direct impact on the accuracy of the power models. Indeed, glitchy signals generated by a component are responsible for a significant amount of power consumption in the following components. We evaluated the difference between results obtained in our previous glitch-free work [110] and the proposed methodology below that takes the glitches into consideration. On the same complex multiplier use case previously adopted, the methodology that takes the glitches into account is capable of achieving a better average relative

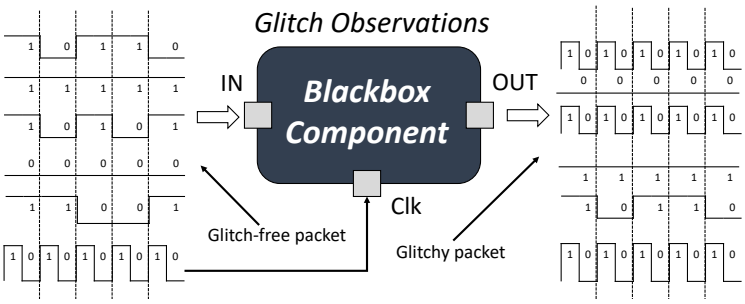



Figure 5.7 – Glitches illustration at the component's outputs

absolute error (%MRAE) of 2.6% instead of the 8.5%, which corresponds to the glitch-free framework developed in [110]. This means that, for the considered test case, by admitting glitchy signals in the power modeling phase, it is possible to achieve system-level power estimations that are more than 3 times accurate than considering only glitch-free signals. Finally, in the rest of the sections below we consider the glitches during the characterization and the modeling stages.

### 5.3.3 Model Dimensioning: Per-Component

To properly re-size the ANN models that are adopted for power and behavioral estimation in the proposed methodology, we have performed an exploration on the unique parameter left free by the considered ANN model (see Section 4.4): the number of neurons in the hidden layer. The metric used to evaluate the accuracy of ANN models is the Mean Square Error ($MSE$). This metric is the average squared difference between the estimated outputs and the desired ones (resulting from Cadence®tools). The $MSE$ can be expressed as in Eq. 5.2:

$$MSE = \frac{1}{N} \times \sum_{k=1}^{k=N} (T(k) - E(k))^2 \tag{5.2}$$

where $E$ is the value estimated using the proposed methodology, $T$ denotes the target value provided by Cadence®tools (Genus Synthesis Solution for power models and Incisive Enterprise Simulator for behavioral ones) and used to train the model, and $N$ stands for the number of samples used for the test phase. Note here that a small value of $MSE$ is desired: a small $MSE$ means that the model is able to estimate the considered output value with high accuracy regarding the desired values that have been provided in its training phase, or at least as well as the adopted gate-level power estimation tool (Genus Synthesis Solution).

Table 5.5 shows how the $MSE$ changes when the number of neurons in the hidden layer of the ANN model changes. For the sake of brevity, model dimensioning results that are related to power models $f$ are only reported here. However, the same selection process can be performed to select the number of

Table 5.5 – Model Dimensioning

| Component's Model | ANN Architectures | MSE |
|---|---|---|
| **f(Mult)** | 16x25x1 | 2.40E-2 |
| | 16x50x1 | 2.22E-2 |
| | **16x75x1** | **2.17E-2** |
| | 16x100x1 | 2.18E-2 |
| **f(Add)** | 32x25x1 | 5.37E-2 |
| | 32x50x1 | 4.21E-2 |
| | 32x75x1 | 3.11E-2 |
| | 32x100x1 | 2.68E-2 |
| | **32x125x1** | **2.14E-2** |
| | 32x150x1 | 3.96E-2 |
| **f(Sub)** | 32x25x1 | 5.14E-2 |
| | 32x50x1 | 4.29E-2 |
| | 32x100x1 | 3.08E-2 |
| | **32x125x1** | **2.34E-2** |
| | 32x150x1 | 2.35E-2 |
| **f(Square)** | 16x25x1 | 1.09E-1 |
| | 16x50x1 | 1.07E-1 |
| | **16x100x1** | **1.05E-1** |
| | 16x125x1 | 1.06E-1 |
| **f(Mac)** | **72x25x1** | **1.2266** |
| | 72x50x1 | 1.4685 |
| **f(Reg)** | **18x25x1** | **1.4298E-4** |
| | 18x50x1 | 1.5435E-4 |
| **f(S&C)** | 36x25x1 | 7.73E-2 |
| | **36x50x1** | **5.36E-2** |
| | 36x75x1 | 6.97E-2 |

neurons for the behavioral models. The number of hidden neurons selected to build the final ANN models of the components corresponds to the one with the smallest $MSE$ value is highlighted in bold for each component. For instance, considering the $f$ power model for the Mac component, Table 5.5 demonstrates that changing the width of the neural networks (number of neurons in the hidden layer) provides less accurate results due to the increase of the $MSE$ from 1.2266 to 1.4685. For example, 1 hidden layer with 25 neurons shows better results than 1 hidden layers with 50 neurons.

### 5.3.4 Characterization and Modeling: Per-Component

In this part, we discuss first the accuracy of the neural-based power and the behavior modeling method. Second, we evaluate the neural-based power modeling technique against the polynomial regression method, while applying both at the component-level.

#### 5.3.4.1 Neural-Based Modeling

Concerning the component-level, the results are related to the stand-alone components models, that are independent and isolated. To evaluate the accuracy of the adopted ANN models (in addition to the $MSE$ introduced in Section 5.3.3 with Equation 5.2) we also use the regression index $R$ expressed in Eq. 5.3. This index measures the correlation between the estimated outputs (resulting from the proposed ANN models) and the desired targets (from Cadence®Genus Synthesis Solution for power, and from Cadence®Incisive Enterprise Simulator for behavior).

$$R = \frac{N \sum TE - (\sum T \sum E)}{\sqrt{[N \sum T^2 - (\sum T)^2][N \sum E^2 - (\sum E)^2]}} \tag{5.3}$$

where $E$ and $T$ have the same meaning of Eq. 5.2. In this case, having a value of $R$ close to 1 is desirable; this would imply then a good correlation between the obtained $E$ and the desired $T$, leading then to a more accurate ANN model.

The results for each component model are presented in Table 5.6. As we can see, the considered metrics attain promising values: the $MSE$ is generally small and $R$ is close to 1 for all the considered components. To prove the accuracy of the power ANN models, Table 5.6 reports additional metrics, as per [90]. Based on this, the Relative-Root Mean Square Error ($R - RMSE$) is considered, and can be expressed as:

$$\%R - RMSE = 100 \times \frac{\sqrt{MSE}}{P_{avg}^{ref}} \tag{5.4}$$

where $P_{avg}^{ref}$ is the average dynamic power consumption of the tested samples. This average is obtained using the Cadence, Genus Synthesis Solution that is considered as the reference for power estimations evaluations. The power deviation of each power model ($RMSE$) is calculated based on the test samples (new samples), and can be defined as:

$$RMSE = \sqrt{MSE} \tag{5.5}$$

The results in Table 5.6 show that ANNs are suitable to learn the relationship between the power consumption and the input features. It also proves that the power models can predict the components power with an error that is, in general,

Table 5.6 – Modeling results for each component model (Values of $\sqrt{MSE}$ and $P_{avg}^{ref}$ are in $\mu W$). Numbers after the component label indicate the total number of bits at the component's inputs.

| Models | MSE | R | $\pm RMSE$ | $P_{avg}^{ref.}$ | %R-RMSE |
|---|---|---|---|---|---|
| $f_{Mult8}(16 \times 75 \times 1)$ | 2.17E-2 | 0.99 | $\pm$0.1473 | 28.6909 | $\pm$0.5134 |
| $g_{Mult8}(16 \times 25 \times 16)$ | 1.3E-4 | 0.99 | - | - | - |
| $f_{Add16}(32 \times 125 \times 1)$ | 2.14E-2 | 0.99 | $\pm$0.1463 | 36.0838 | $\pm$0.3842 |
| $g_{Add16}(32 \times 50 \times 16)$ | 1.6E-4 | 0.99 | - | - | - |
| $f_{Add32}(64 \times 100 \times 32)$ | 0.1159 | 0.99 | $\pm$0.3404 | 52.6360 | $\pm$0.6467 |
| $f_{Sub16}(32 \times 125 \times 1)$ | 2.34E-2 | 0.99 | $\pm$0.1530 | 35.9500 | $\pm$0.4256 |
| $g_{Sub16}(16 \times 50 \times 16)$ | 1.8E-4 | 0.99 | - | - | - |
| $f_{Squ.8}(16 \times 100 \times 1)$ | 0.105 | 0.99 | $\pm$0.3240 | 47.6160 | $\pm$0.6804 |
| $g_{Squ.8}(16 \times 100 \times 32)$ | 1.04166E-4 | 0.99 | - | - | - |
| $f_{Mac36}(72 \times 25 \times 1)$ | 1.2266 | 0.99 | $\pm$1.1075 | 142.4904 | $\pm$0.7772 |
| $g_{Mac36}(72 \times 25 \times 36)$ | 6.3322E-4 | 0.99 | - | - | - |
| $f_{Reg9}(18 \times 25 \times 1)$ | 1.4298E-4 | 0.99 | $\pm$0.0120 | 24.4146 | $\pm$0.0492 |
| $g_{Reg9}(18 \times 25 \times 18)$ | 1.02141E-7 | 1.00 | - | - | - |
| $f_{S\&C18}(36 \times 25 \times 1)$ | 5.87E-2 | 0.98 | $\pm$0.2423 | 7.6077 | $\pm$3.1849 |

always less than $\pm$1.3% (see $\%R - RMSE$ columns). The shift and clip (S&C) component represents an exception, with an error of $\pm$3.2%. This is due to the fact that the average power $P_{avg}^{ref}$ of $S\&C$ is the lowest in the library; for approximately the same magnitude of $\sqrt{MSE}$, the corresponding $\%R - RMSE$ is larger.

#### 5.3.4.2 Regression Versus Neural-based Modeling

In this section, we compare with a commonly used approach, the regression-based power modeling technique [9, 11, 12, 111] to demonstrate the suitability of ANNs to address the considered power modeling and estimation problems. As discussed in the literature review chapter (Chapter 3, Section 3.4.5), the regression-based modeling technique shows more accurate results than the analytical and table-based approaches. For this reason, we select the regression-based technique for this comparison. For example, authors in [111] present a linear relationship between the power consumption and the average switching activity. To achieve better modeling accuracy, we propose polynomials with higher degree. In addition, authors in [7] show that the average static probability has a strong relation with the power consumption in digital circuits. To this aim, an additional parameter that represent the average static probability is added to the average switching activity that is introduced in the linear relationship in [111]. To this aim, the power consumption of a component ($X$) can be better approximated by

means of a polynomial regression of degree 2, as follows:

$$P_X = a + b \times D_{in} + c \times P_{in} + d \times D_{in}^2 + e \times D_{in} \times P_{in} \tag{5.6}$$

where $P_X$ is the dynamic power of the component, $D_{in}$ and $P_{in}$ represent the average switching activity (in terms of bits) and static probability respectively at the primary inputs of a component, while $a$, $b$, $c$, $d$, $e$ are parameters of the model that have been fixed during a characterization phase. Note here that these coefficients are computed after the power characterization for each component, considering 10000 different input data. To obtain such regression-based model for a given component, it is then necessary to define $D_{in}$ and $P_{in}$ that is not keeping per bit information (differently from ANNs activity factor). A component, composed of $N$ input bits, can be characterized by an average switching activity and static probability expressed as:

$$D_{in} = \frac{1}{N} \times \sum_{n=1}^{n=N} AF(n) \tag{5.7}$$

$$P_{in} = \frac{1}{N} \times \sum_{n=1}^{n=N} P_1(n) \tag{5.8}$$

where $AF[n]$ is the activity factor that represents the average number of transitions ($0 \rightarrow 1$ and $1 \rightarrow 0$) in a data sequence, which is associated to a component's input $n$, and can be expressed as follows:

$$AF[n] = \frac{\sum_{i=0}^{i=L-1}(x_{i,n} \oplus x_{i+1,n})}{L-1} \tag{5.9}$$

and the static probability $P_1[n]$, shows the percentage of ones (1's) or the logic high in a data sequence, that is associated to component's input $n$, and can be expressed as follows:

$$P_1[n] = \frac{\sum_{i=0}^{i=L-1}(x_{i,n})}{L} \tag{5.10}$$

where $L$ is the sequence length of the data (binary signal), $N$ is the total number of bit (width) of component's inputs, and $x_{i,n}$ is the value (0 or 1) of the $n$-th bit and $i$-th column. Note that $n$ ranges from 0 to $N-1$, and $i$ from 0 to $L-1$.

To build the regression-based power model, the power characterization of the different components has been carried out using RTL synthesis, post-synthesis simulations and power estimation offered by Cadence®tools. A database of the power consumption values for the different input data ($D_{in}$ and $P_{in}$) has been built. The dataset size (10000 samples), the operating frequency (100 MHz) and the target technology (Cadence®GPDK 45 nm technology) are the same as

Table 5.7 – RMSE for the regression-based power modeling technique

| Power Models | ± RMSE ($\mu W$) |
|---|---|
| $P_{Mac36}$ | 4.1000 |
| $P_{S\&C18}$ | 1.2000 |
| $P_{Add16}$ | 0.6200 |
| $P_{Squ.8}$ | 0.5800 |
| $P_{Sub16}$ | 0.5300 |
| $P_{Add8}$ | 0.5100 |
| $P_{Mult8}$ | 0.4000 |
| $P_{Reg9}$ | 0.0130 |

for the ANN models. Finally, the power models results of each component are presented as follows:

— $P_{Squ.} = 12.39 + 48.51 \times D_{in} + 1.763 \times P_{in} - 10.47 \times D_{in}^2 + 45.97 \times D_{in} \times P_{in}$

— $P_{Sub} = 14.73 + 45.23 \times D_{in} + 0.2866 \times P_{in} - 7.63 \times D_{in}^2 - 0.3071 \times D_{in} \times P_{in}$

— $P_{Add8} = 14.56 + 43.02 \times D_{in} + 0.7256 \times P_{in} - 7.623 \times D_{in}^2 + 4.43 \times D_{in} \times P_{in}$

— $P_{Add16} = 29.52 + 40.68 \times D_{in} - 0.1372 \times P_{in} + 83.27 \times D_{in}^2 + 5.94 \times D_{in} \times P_{in}$

— $P_{Mult} = 9.909 + 2.855 \times D_{in} - 0.4007 \times P_{in} + 11.86 \times D_{in}^2 + 57.91 \times D_{in} \times P_{in}$

— $P_{Mac} = 12.79 + 2302 \times D_{in} + 12.44 \times P_{in} - 5216 \times D_{in}^2 + 826.7 \times D_{in} \times P_{in}$

— $P_{S\&C} = 4.708 + 10.88 \times D_{in} + 0.7047 \times P_{in} + -0.02836 \times D_{in}^2 - 16.59 \times D_{in} \times P_{in}$

— $P_{Reg} = 134.5 \times D_{in} + 38.94$

Table 5.7 shows the results of the regression-based power modeling technique for the components that are used in our cases study. Results show that the Mac component has a very low accuracy comparing to the others. The reason for that is the component's complexity. In other cases, when the components are much simpler than Macs, such models are then more suitable, while reaching lower $RMSE$.

$R - RMSE$ represents a deviation relative to an average power consumption per component. Hence, it is worth to compute the percentage of the relative $RMSE$ ($\%R - RMSE$) to assess the accuracy of the modeling method as in Eq. 5.4. To this end, Table 5.8 presents a comparison between the ANNs and the considered regression-based power models. The regression-based power modeling technique shows very low accuracy when compared to ANN models. ANN models are able to guarantee an error always less than 0.8%, while the regression-based reaches up to 15.8% of error. This comparison demonstrates that ANNs are capable of overperforming the commonly used power modeling approaches, like regression-based ones, for the considered use case and context. Lastly, the accuracy of the component-level modeling affects the one of the system-level power estimations. This is because the system-level estimates are based on the

Table 5.8 – Regression-based Vs Artificial Neural Networks Power Modeling techniques

| Models | ± RMSE ($\mu W$) | | $P_{avg}(\mu W)$ | ± %R-RMSE | |
|---|---|---|---|---|---|
| | Regression-based | ANN-based | | Regression-based | ANN-based |
| Mac | 4.1000 | 1.1075 | 142.4904 | 2.90 | 0.78 |
| S&C | 1.2000 | 0.2423 | 7.60770 | 15.8 | 3.20 |
| Add16 | 0.6200 | 0.3404 | 52.6360 | 1.18 | 0.64 |
| Squ. | 0.5800 | 0.3240 | 47.6160 | 1.22 | 0.68 |
| Sub | 0.5300 | 0.1530 | 35.9500 | 1.48 | 0.42 |
| Add8 | 0.5100 | 0.1463 | 36.0838 | 1.42 | 0.40 |
| Mult | 0.4000 | 0.1473 | 28.6909 | 1.40 | 0.51 |
| Reg | 0.0130 | 0.0120 | 24.4146 | 0.05 | 0.04 |

component models that compose the system. To this end, accurate estimates of the power at the system-level can be accomplished using accurate components models. For this purpose, we present below the system-level assessments.

### 5.3.5 Power Estimation: Per-System

The objective of this section is to assess the benefits of NeuPow, the proposed system-level power estimation methodology for a composite system, using different case studies. In the result analysis below, we compare the obtained power values in four different ways:

1. *Gate-Level Power Estimations* (Reference): The power estimations are provided by the Cadence Genus Synthesis Solution. To perform these estimations, we need to have 1) the gate-level netlist of the system, 2) the timing library that is related to a given technology, and 3) the testbench or the data stimuli.
2. *Propagation-less Estimations* (Straight-forward): or also called initial estimates. It is the straight-forward method, in which power estimation is carried out using the average power consumption extracted per-component. The values of the average power consumption can be extracted using the cadence tool. T he system's total average power consumption, it can be derived by summing up the power consumption values associated to each of the system's components. Note that the interconnections between the system's components are not taken into account.
3. *Polynomial-regression-based Estimations* (Literature): The power estimations are delivered by the regression-based components models. The behavioral models, and the power ones are built using the polynomial-regression technique. The behavioral models are responsible for performing the feature propagation by means of average activity factor and static probability.

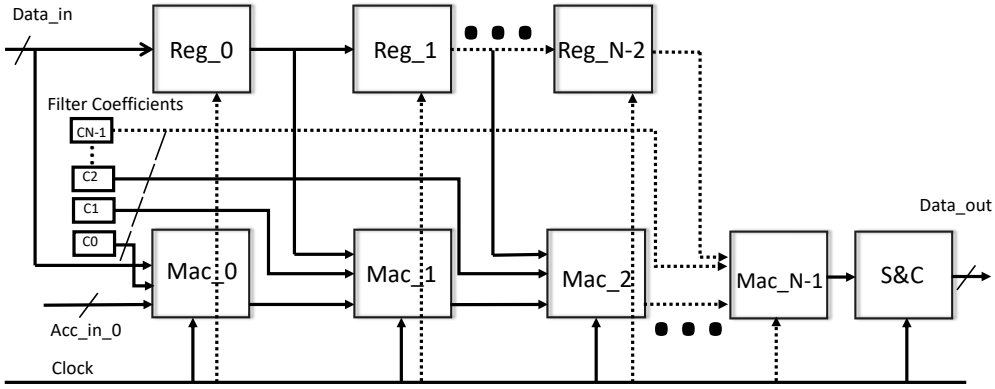



Figure 5.8 – Image filter (N layers/taps)

After that, the power models use the propagated features among all the components in order to get a system-level power estimation.

4. *NeuPow Power Estimations* (Our): The power estimations are carried out by NeuPow to exploit both: the power models per component ($f$) and the behavioral models per component ($g$), as described in Chapter 4. To perform these estimations, we only need to use the developed library of models that are associated with the components.

#### 5.3.5.1 Propagation-less (Straight-forward) Power Estimations

This constitutes the straightforward method to estimate the average power consumption of a composite system. This method is based on summing up all the average power consumption of the components. To this end, we may consider the N-tap image filter (See Figure 5.8)to start the comparison between the Gate-Level versus the Propagation-less (PL) estimations at the system-level. For instance, the total power consumption of an image filter that is composed of $N$ taps can be derived as in Eq. 5.11:

$$P_{PL}^{N}(Imagefilter) = N \times (P_{avg}^{ref}(Mac)) + (N-1) \times P_{avg}^{ref}(Reg) + P_{avg}^{ref}(S\&C) \quad (5.11)$$

where $N$ represents the number of taps of the filter, and $P_{avg}^{ref}(X)$ reflects the average power consumption of each component such as the Multiply-accumulate (Mac), Register (Reg) and the Shift and clip (S&C).

The results, presented in Table 5.9, show that the relative error of the *Propagation-less* estimations is always above 12% for the considered cases. Comparing this error against the reference, i.e., the *Gate-Level* estimations, this error is still high. The captured values demonstrate that the usage of the propagation-less estimation technique has limitations in terms of accuracy. However, it can still be used to get the composite or the system-level power estimations without using

Table 5.9 – System-level power estimations: on the impact of the signal propagation

| Scalability Configurations | Gate-Level (Cadence®Genus) ($\mu W$) | Propagation-less (Initial Estimates) ($\mu W$) | % (Relative Error) |
|---|---|---|---|
| Image Filter (4 Layers) | 746.2769 | 650.8124 | 12.7921 |
| Image Filter (6 Layers) | 1135.000 | 984.6224 | 22.0597 |
| Image Filter (8 Layers) | 1526.600 | 1318.4000 | 13.6304 |
| Image Filter (10 Layers) | 1916.000 | 1652.2000 | 13.7787 |

the behavioral model ($g$) described before. Hence, this saves the burden of the behavioral characterization and modeling for each component. Finally, we may also notice that the error in the 6-tap filter case is about 22%, which is not very accurate in this case. Consequently, the impact of the signal propagation on the power estimations is very crucial.

#### 5.3.5.2 Regression-based (Literature) Power Estimation

In this section, we present the results of the power estimations of a composite system, which mainly consists of 3 different components (Macs, S&C, and registers). These estimations are based on the regression-based power and the behavioral models. The power models of each component are developed and presented in Section 5.3.4. To evaluate the regression-based power estimations at system-level, behavioral models are used to estimate the power consumption of a composite system. Based on the work presented in [86], behavioral models are derived. In this section, we consider the behavioral models of 3 components: the Mac, S&C and Reg. The behavioral model ($g$) aims to map both, the average activity factor as well as the static probability of the primary inputs to the one that corresponds to the component's primary outputs (See the top right side of Figure 5.9).

The behavioral characterization (See Section 4.3) is performed, and the average activity factor and static probability at the component's primary outputs are calculated. After that, the regression-based power modeling method has been run, which uses these data (10000 samples) in order to build the models. Note that the degree of the polynomial regression models are select based on the minimum MSE achieved during the modeling phase. The models of each component are presented as follows:

1. For the Multiply-accumulate (Mac) component, we define the $g$ model by its two sub-models as follows:
   — $D_{out} = -0.0004869 + 5.755 \times D_{in} + 0.03244 \times P_{in} - 58.6 \times D_{in}^2 + 1.345 \times D_{in} \times P_{in} + 193.5 \times D_{in}^3 - 11.83 \times D_{in}^2 \times P_{in}$
   — $P_{out} = 0.4662 - 0.0008223 \times D_{in} + 0.06724 \times P_{in}$

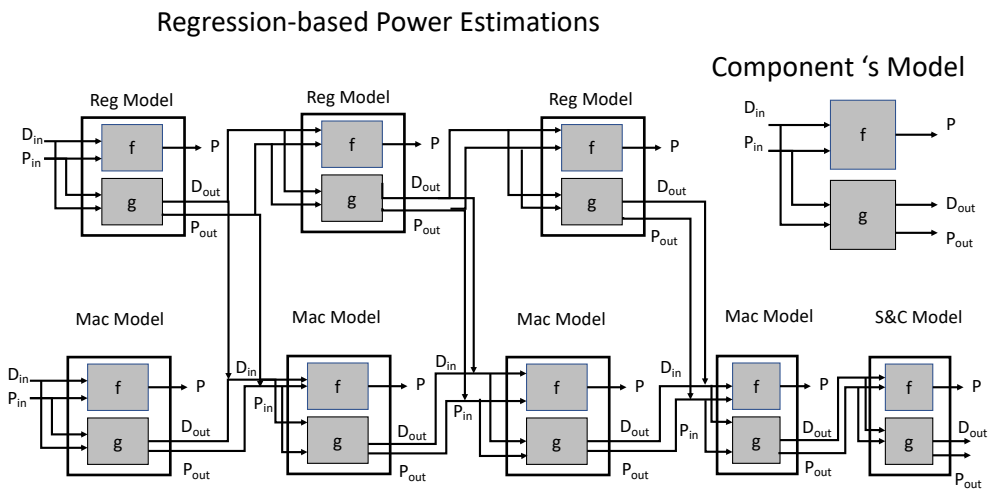



Figure 5.9 – Regression-based for system-level power estimation (i.e., 4- tap Image filter)

2. For the Shift& Clip (S&C) component:
   — $D_{out} = 0.1147 + 0.2464 \times D_{in} - 0.223 \times P_{in}$
   — $P_{out} = 0.8199 - 0.002396 \times D_{in} + 0.2986 \times P_{in}$
3. For the Register component:
   — $D_{out} = 0.9985 \times D_{in}$
   — $P_{out} = 0.9997 \times P_{in}$

The system-level power estimations are performed based on the developed power models proposed in Section 5.3.4. To illustrate this, Figure 5.9 shows the composition of a 4-tap image filter by means of both power and behavioral models. To perform a power estimation of the whole filter, the average activity factor and the static probability of the system's inputs are injected to the system. The average factor and the static probability are propagated, thanks to the behavioral models. Then, the power consumption of each component, is added to obtain the total power consumption for each input sample.

It is possible to estimate the power consumption of the 4-tap filter using the regression-based components model. The obtained average power, is about 856 $\mu W$. The collected total average power consumption of the same circuit (while considering the same bench of inputs stimuli) is about 746 $\mu W$. Note here that this value is obtained from the cadence ®Genus Solution, and is considered as the reference power. To assess the regression-based power estimation method, we may compute the relative estimation error. For instance, the percentage relative estimation error of the 4-tap filter is about 14.7%. This error is still too high relative to the straightforward or the propagation-less method (error of about 12.7%). The regression-based technique, based on average activity and static

Table 5.10 – System-level power estimation: Gate-level Vs Our Method, NeuPow

| Use Cases | Gate-Level (Cadence®Genus) ($\mu W$ ) | NeuPow ($\mu W$ ) | % (Relative Error) |
|---|---|---|---|
| Complex Multiplier | 186.0959 | 190.9553 | 2.6112 |
| Magnitude Calculator | 372.2309 | 384.5835 | 3.3185 |
| Image Filter (4 Layers) | 746.2769 | 706.8775 | 5.3619 |
| Image Filter (6 Layers) | 1135.0000 | 1142.9000 | 0.6167 |
| Image Filter (8 Layers) | 1526.6000 | 1576.6000 | 3.2765 |
| Image Filter (10 Layers) | 1916.0000 | 2011.2000 | 4.9582 |

probability, indicates a very low precision. This is due to the fact that we use the average activity and the static probability factors, which lead to loosing bit information. The results demonstrate that the bit information is very important when performing system-level power estimations. To this aim, we discuss in the rest of the sections the outcomes of our suggested technique.

#### 5.3.5.3 Neural-based Approach (our) Power Estimation

In this section, we evaluate NeuPow in terms of its estimation accuracy, while considering different case studies and analyzing the results. In Table 5.10 the power values estimated using our method are shown. As we can see, the upper bound for the error is equal to 5.5%, with respect to the target. In NeuPow, the behavioral model $g$ for each component makes it possible to propagate the switching activity among the different components. These components lead to more accurate power estimations at system-level. This then allows designers to explore different design choices and topologies.

While analyzing the numerical results, we can notice that the information related to the system architecture or the system topology is very important to accurately estimate the power consumption at the system-level. In fact, the power estimates that consider the topology of the composite system (the case when using behavioral models $g$) provide better outcomes than the case when the topology information is missing, such as the propagation-less estimates. For example, signals correlation in propagation-less method are not taken into considerations. In addition to this, keeping the activity factor and the static probability at the bit-level makes it possible to estimate the power consumption accurately.

To conclude, Figure 5.10 shows the power traces obtained using: 1) Cadence tool (reference), 2) Regression-based (method discussed above), and 3) our proposed method, NeuPow, while considering 10000 samples to monitor the average power consumption at system-level. As depicted in this figure, the power trace of NeuPow outperforms the regression-based trace for most of the samples. Based on the power average lines, it is clear that Neupow is relatively close to the

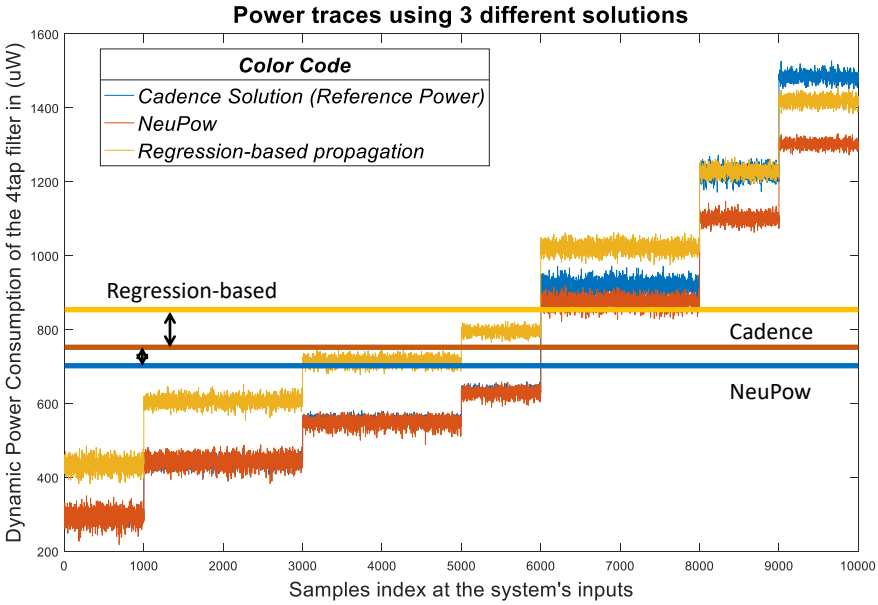



Figure 5.10 – Regression-based for system-level power estimation (i.e. 4 tap Image filter)

cadence solution.

### 5.3.6 Technology Verification

While all previous results for ASICs circuits refer to the 45 nm technology, an evaluation of a distinct and more recent technology has been performed: the 15 nm FinFET-based Open Cell Library [109]. The differences between this technology and the previous one appear in the size of the technology point (15nm instead of 45nm) as well as in the type of gates (FInFET instead of CMOS). Furthermore, the FinFET technology provides many advantages over CMOS, such as a higher drive current for a specified transistor, leading to a higher speed, and hence a reduced leakage. By this, the power consumption with no random doping fluctuations gets reduced.

The adoption of a different technology requires a complete execution of the whole characterization and the modeling flow. We consider the same case studies as adopted for the 45nm technology, especially the image filter. For this, we first perform the model dimensioning. Then, we provide the results for the three different components used in the image filter: Mac, Reg and S&C. The component-level results are shown in Table 5.11. These reported numbers show that the ANN models are still suitable for estimating the power of the synthesized components on the 15 nm technology, where the maximum relative error is about 3.7%. As for the 45 nm technology, this occurs in the S&C component case, that has the lowest average power $P_{avg}^{ref}$ among the library components.

Table 5.11 – Power Models on 15nm Technology.

| Models | MSE | $\pm RMSE$ $(\mu W)$ | $P_{avg}^{ref}$ $(\mu W)$ | %R-RMSE |
|---|---|---|---|---|
| $f(Mac)$ | 0.370611 | ±0.6088 | 87.1459 | ±0.6986 |
| $f(Reg)$ | 0.0000219762 | ±0.0047 | 10.3939 | ±0.0451 |
| $f(S\&C)$ | 0.0203668 | ±0.1427 | 3.8859 | ±3.6726 |

Table 5.12 – 15nm Technology Validation

| Test-Case | Gate-level (Cadence Genus) $(\mu W)$ | NeuPow $(\mu W)$ | %RE |
|---|---|---|---|
| Image Filter (4 Layers) | 424.5320 | 460.2543 | 8.4906 |
| Image Filter (6 Layers) | 664.1129 | 718.4044 | 8.1325 |
| Image Filter (8 Layers) | 896.7535 | 980.7820 | 9.3750 |
| Image Filter (10 Layers) | 1123.800 | 1225.600 | 9.0828 |

In addition, we also empirically verified our proposed method, NeuPow, on the 15nm technology. With this regard, we consider the image filter for 4 different sizes. Table 5.12 motivates the analysis at the system-level. In this table, a comparison among *Gate-Level Estimations* and *NeuPow Estimations* is reported for all the four design variants of the image filter use cases: *4 taps/layers*, *6 taps/layers*, *8 taps/layers*, *10 taps/layers*. Considering this new technology, the relative error is always between 8% to 9.5% for all the filter configurations. This error has increased with respect to the 45 nm technology (the maximum relative error in such case was 5% for the *4 taps/layers* design), but is still below 10%. With regard to the sources of error, two sources are possible: the error of estimation, where our models provide overestimation and the power in the interconnections, which are not taken into consideration. Modeling error is the common source of error for both technologies (45 nm and 15 nm), while the power of interconnection for newer technologies is less than for older technologies (45 nm). To this end, the difference between the power estimation obtained from both cadence (reference) and our proposed technique (NeuPow) is higher, and therefore the relative estimation error is higher.

Finally, we may also provide additional information about the total power consumption (dynamic+static). To this end, it is possible to define the $Ratio = (100 \times P_{static}/P_{dynamic})$, which is usually about 2% as shown in Table 5.13. Therefore, we can say that the static power is about 2% of the dynamic power consumption. Hence, it is possible to express the total power consumption as follows:

$$P_{total} = P_{Dynamic}^{NeuPow} + 0.02 P_{Dynamic}^{NeuPow} \tag{5.12}$$

Table 5.13 – Static power vs dynamic power 15nm Technology

| Design Layers | Static Power ($\mu W$) | Dynamic Power ($\mu W$) | Ratio (%) |
|---|---|---|---|
| *4 layers* | 7.9700 | 414.5320 | 1.9227 |
| *6 layers* | 12.007 | 624.1129 | 1.9239 |
| *8 layers* | 16.044 | 836.7535 | 1.9174 |
| *10 layers* | 20.079 | 1046.800 | 1.9181 |

### 5.3.7 Summary

In this section, we verified the proposed estimation method on both ASICs technologies (45nm CMOS and 15nm FInFET). Our estimation method proved an estimation accuracy of about 10%. To deliver more scalable, reliable, fast, and flexible power estimation approach, method assessments are needed. For this purpose, method scalability evaluation, estimation time study, and model extension are conducted. For this purpose, Section 5.4 presents a time assessment of the proposed estimation methodology. Then, Section 5.5 proves the scalability of *NeuPow*, so that it is possible to perform the power estimation for more considerable design size. Finally, Section 5.6 presents the frequency evaluation and the model upgrade, so that the new models may take the operating frequencies into consideration, for a frequency-aware power estimation method. Note that, in the rest of the assessments 45nm technology is considered.

## 5.4 NeuPow: Estimation Time Assessment

In this section, power estimation time factor is addressed. This metric is very important knowing that it is heavily linked to design productivity. The productive design methods contribute to less time-to-market in the digital design, allowing it to compete with other alternatives. In this respect, we can demonstrate the significance of this metric with a practical instance. For example, digital designers need to wait for a full design round (See Section 2.3.2) to estimate the power consumption of a given design choice. It is possible to express the time needed to estimate the power of a full design round, with a classical method as follows:

$$t_{Design-Round} = t_{RTL-Modeling} + t_{Synthesis} + t_{Implementation} + t_{Power-estimation} \quad (5.13)$$

where

- $t_{RTL-Modeling}$ denotes the time required to model the design using HDL;
- $t_{Synthesis}$ represents the time needed to synthesis a design (to produce the netlist);

– $t_{Implementation}$ corresponds to the total time, which requires to perform the physical connections;

– $t_{Power-estimation}$ reflects the required time to predict the power consumption. Note that, accurate power values are obtained using either a SAIF (Switching Activity Interchange format) file, which contains toggle counts (number of transitions) on the signals of the design or VCD (Value Change Dump) file that is an ASCII file, which containing header information, variable definitions, and value change details for each step of the simulation. These files are generated after a timing simulation that uses the structural timing netlist, SDF file, and test bench. To this aim, this time include the time needed to perform the timing simulation.

In relation to this, a number of different design architectures can be explored for a specified design. For example, if we assume that $n$ different design architectures exist for the same design, then $n$ design rounds are needed to explore all of these options, with a total time required that is equal to:

$$t_{Exploration-time} = n \times t_{Design-Round} \tag{5.14}$$

Note that, this formulation assumes that the time required to perform one design round for all choices are equal. However, the time per design round is proportional to the design size, and hence each design choice will have a different time per design round. For this purpose, the exploration time represented in Eq.(5.14) can now be generalized, as follows:

$$t_{Exploration-time} = \sum_{i=1}^{i=n} t^{i}_{Design-round} \tag{5.15}$$

where $i$ is the index of the $i$-th design round (design choice). It is evident that we need to minimize all the time elements associated with the exploration time expenses in order to minimize the exploration time for $n$ distinct design options. Hence, it avoids heavy simulation time.

In the rest of this section, numerical results are presented. The aim here is to compare the required time to run the power estimations using our method, NeuPow, with respect to that required using commercial CAD tool at the gate-level, as the Cadence®Genus Solution one. In the below, we assume that the time required to execute the RTL-Modeling, Synthesis and Implementation are negligible. We only compare the time needed to perform the power estimations on both solutions. In the exploration below, we are using the same operating conditions (operating frequency, target technology and input data vectors) for each considered system configuration for both *NeuPow Estimations* and *Gate-Level Estimations*. The power consumption for different image filter design sizes (*4 layers*, *6 layers*, *8 layers* and *10 layers*) is extracted from both methods, and the time traces are captured. Each design configuration is simulated to

Table 5.14 – Estimation time: Low-level gate level power simulations Vs NeuPoW power simulations

| Case studies | Gate-Level (Cadence®Genus) (Minutes) | NeuPow (Minutes) | Speed-up factor X |
|---|---|---|---|
| *4 Layers* | 35 | 0.34 | 102.9412 |
| *6 Layers* | 48 | 0.47 | 102.1277 |
| *8 Layers* | 62 | 0.64 | 96.8750 |
| *10 Layers* | 78 | 0.84 | 92.8571 |
| **Total time** | **223** | **2.3** | **96.9565** |

get 10000 power values corresponding to the 10000 different input data vectors. Table 5.14 shows the results in terms of the estimation time (in minutes) for each method. Hence, to explore the power consumption for all the 4 different considered designs, the gate-level power estimations require about 223 minutes, whereas NeuPow needs 2.3 minutes only. This corresponds to an acceleration factor of 96×. Note that the required time to characterize and build the models in NeuPow is not considered in the data reported in Table 5.14. As for the Gate-level power simulations, they are performed on an Intel(R) Xeon (R) CPU E5-2620 @2.00GHz, while *NeuPow* simulations are achieved on an Intel Core i5 @ 2.3GHz.

Despite the huge speed-up factor that NeuPow is capable of guaranteeing to its users when the library of components is available, the preparation of the library itself has a cost, and even the insertion of a new single component does not come for free. For any component addition, it is necessary:

— to characterize the component in terms of power and behavior, which implies performing several RTL synthesis, post-synthesis simulations and power estimations; and

— to train (specialize) the corresponding ANN models by feeding them with the provided characterization data.

However, the benefit in the use of NeuPow is still high. Recall that, in this comparison, the time needed to synthesize the design is not taken into consideration. Note that the synthesis time metric becomes very crucial when the size of the design rises. Using our advanced library of models and our proposed power estimation methodology, it is advantageous for designers to skip this step and thus to boost the design productivity. Indeed, if the designers manage to create a wide and heterogeneous component library, then he/she would always be able to get a substantial speed-up with respect to the classical approaches based on gate-level explorations. Just as an example, with a simple library of 8 components, we were able to use NeuPow over 6 use cases in this work.

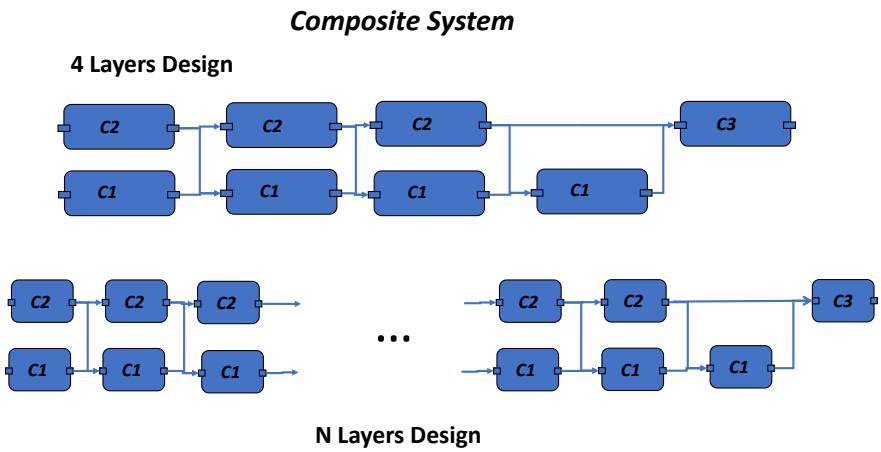



Figure 5.11 – Classical RTL description of a composite system

## 5.5 NeuPow: Scalability Study

The accuracy of NeuPow has been also analyzed from a different perspective. This section aims to assess the effect of the system depth and the number of cascaded layers on the accuracy. For this, the components have been ordered in layers following a dataflow approach. The $n$-th layer is then composed of components that receive values from the components in the previous layer (i.e, $n-1$) and that provide values to the components in the next layer (i.e., $n+1$). In the following exploration, we considered different versions of the image filter depicted in Figure 5.11 with different number of taps that correspond to different numbers of system layers: *4 taps/layers*, *6 taps/layers*, *8 taps/layers* and *10 taps/layers*. As an example, Figure 5.11 depicts the *4 taps/layers* image filter (top part of the figure) and a generic *N taps/layers* one to cover the cases from *6 taps/layers* to *10 taps/layers*. According to the image filter structure, it is possible to replace C1 by the Mac operator, C2 by the Reg, and C3 by S&C components.

Following our power estimation approach, it is possible to describe the image filter using the component's model $M$ (see Eq. 4.10) from our developed library of models. As an illustration for this, Figure 5.12 presents the global system making use of the component models. In this figure, $M1$, $M2$ and $M3$ correspond to the models of a Mac, a Reg and a S&C, respectively.

Table 5.15 presents the scalability analysis, while comparing NeuPow Estimations and Gate-Level Estimations. We can conclude from this table that the maximum error of NeuPow is about 5.3%. Moreover, it is clear that this error is not growing with the width or the design size/layers of the system. This means that NeuPow performance is independent from the number of layers. In fact, the randomness of the error in sign and magnitude committed by the different

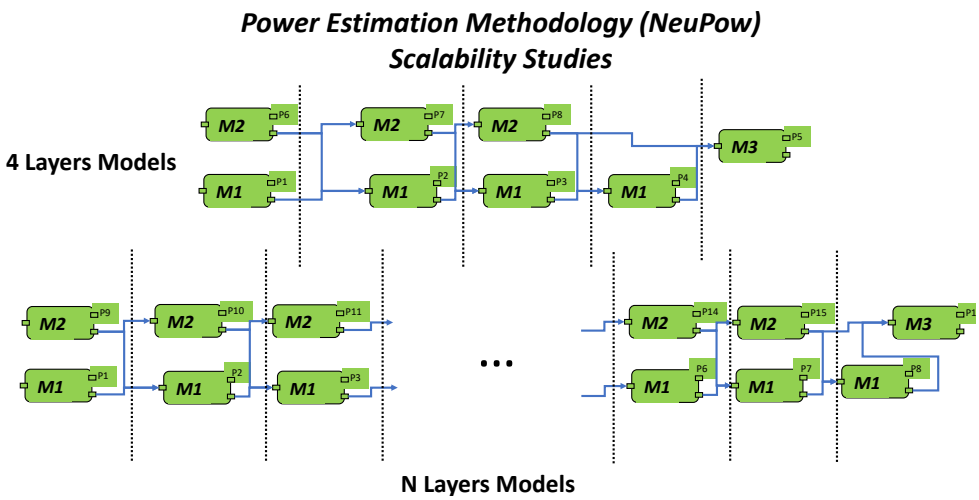



Figure 5.12 – NeuPow description of the digital filter based on our constructed library of models

ANN component models can lead to even smaller errors for deeper systems in the considered use case with respect to the smallest (4 layers) design. In summary, this exploration shows that NeuPow provides a good estimation accuracy independently of the design size. In addition, the results of the 15 nm technology presented in section 5.3.6 (See Table 5.12) show an error trend that does not depend on the system layers, as was the case with the 45 nm technology (See Table 5.15). This implies that NeuPow maintains the estimation scalability.

Table 5.15 – NeuPow: Scalability Studies

| Scalability Settings | Gate-Level (Cadence®Genus) ($\mu W$) | NeuPow ($\mu W$) | %RE (NeuPow) |
|---|---|---|---|
| *Image Filter (4 Layers)* | 746.2769 | 706.8775 | 5.3619 |
| *Image Filter (6 Layers)* | 1135.000 | 1142.900 | 0.6167 |
| *Image Filter (8 Layers)* | 1526.600 | 1576.600 | 3.2765 |
| *Image Filter (10 Layers)* | 1916.000 | 2011.200 | 4.9582 |

## 5.6 NeuPow: Extension Evaluation

Dynamic power consumption in digital circuits depends on several parameters, such as the activity factor, the static probability, the operating frequency and the considered technology. In this section, we study how dynamic power consumption changes with respect to the operating frequency. Generally speaking, the dynamic

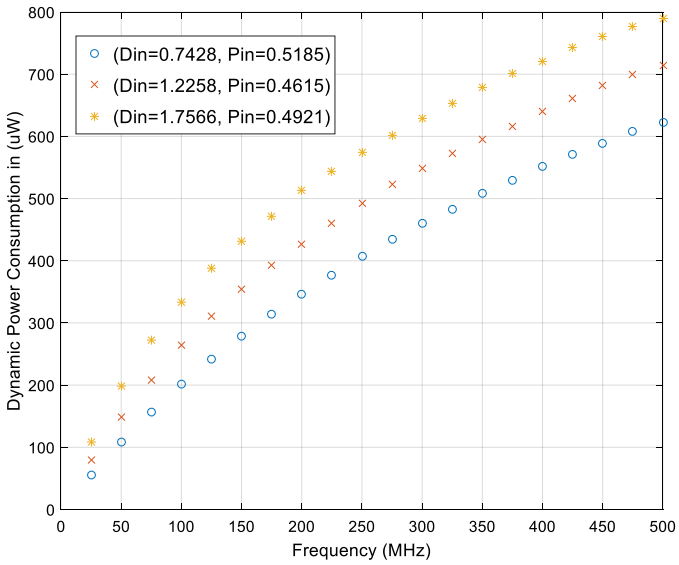



Figure 5.13 – Mac dynamic power consumption versus the frequency for 3 different input configurations

power is proportional to the frequency. In Figure 5.13, we plot the dynamic power consumption as function of the operating frequencies. As we can see, the power consumption trend for a Mac component is no longer linear anymore. These results are obtained after performing gate-level power simulations on 10000 input samples using the Cadence ®Genus solution. We plot here 3 samples only to show the shape of the dynamic power consumption. It is evident that the dynamic power consumption loses its linearity for a substantial value of the average activity factor. For this, we propose the same model template that is based on neural networks to model power consumption. The power estimates are then based on three variables: the activity factor, the static probability and the operating frequency.

Therefore, to build a Frequency-aware Power Model, we adopt the NeuPow baseline, while repeating all the required steps that are related to the characterization, the modeling and the library construction phases. Some modifications are also performed. In this case, the power characterization is performed at different frequency points, leading to different dynamic power consumption values. This frequency is then added as a new input to the power model. The characterization of the new ANN power model $h$ is made using the power values resulting from $f$ (which is trained with the same input features as previously used to characterize the power). Moreover, it is achieved using the frequency $f_{in}$, so that it can be possible to predict the new dynamic power consumption taking the frequency points into account (See Figure 5.14).

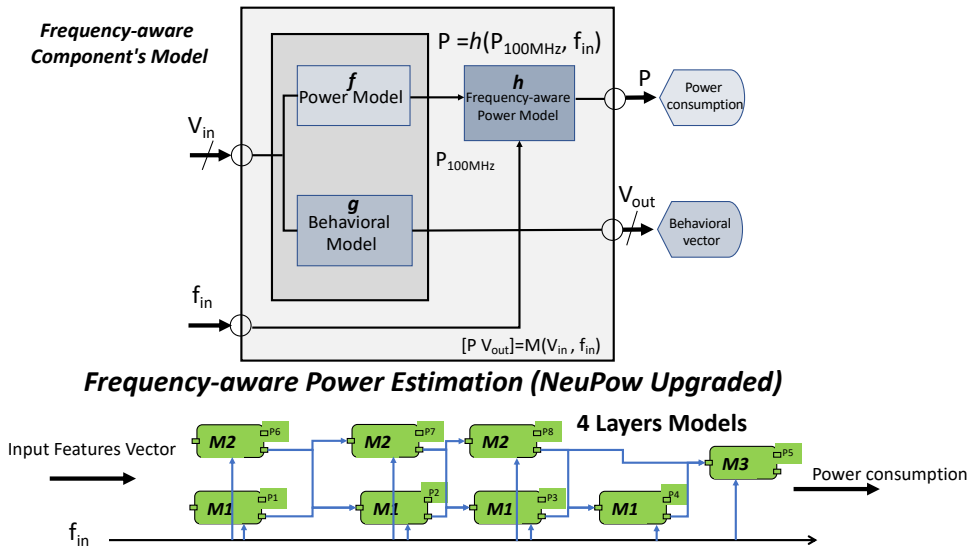



Figure 5.14 – Frequency-aware power estimation (i.e. 4 tap Image filter)

Below several numerical results performed at system-level are presented. In order to evaluate the efficiency of the frequency-aware power models related to each component, 4 case studies using the new power models are built (results are presented in Table 5.16). This table re-emphasizes that the relationship between the frequency and the dynamic power is not linear at the system-level. For example, we may notice that the power consumption of a composite system is about $746\mu W$ at $100MHz$, whereas it is equal to $2547\mu W$ at $500MHz$. This means that the power is not scaled by 5 (not equal to $3730\mu W$), as it would be the case of linear relationship. If such linear approximation had been adopted for frequency aware power estimation, an error of 46.4% would have occurred.

For this reason, we propose to evaluate the frequency effect on the system-level power estimations using our extended components models. This improvement offers to our method, NeuPow, an additional feature to explore the design space in terms of power, by taking the system operating frequency into account. To evaluate the Frequency-aware Power Model, we consider the 4-layers image filter as a test case at different frequencies (100, 200, 300 and 500 MHz).

Table 5.16 shows the comparison between the achievable results using the improved NeuPow and the target Gate-Level power estimations. The error of the frequency-aware power estimation method never exceeds 8.5%, but it is slightly increasing as the frequency grows. Building on this preliminary data, we can claim that if the frequency is extremely different from the starting one (the one adopted for training $f$ and $g$ ANN models), then the error becomes high. For this reason, Frequency-aware Power Model needs further investigations and some refinements in future works.

Table 5.16 – Frequency-aware power estimations: Gate-level Vs NeuPow (Upgraded)

| Frequency Configurations (MHz) | Gate-level (Cadence®Genus) ($\mu W$) | NeuPow (uW) | % RE |
|---|---|---|---|
| 100 | 746.2769 | 707.1985 | 5.2364 |
| 200 | 1331.400 | 1240.000 | 6.8650 |
| 300 | 1813.100 | 1663.000 | 8.2731 |
| 500 | 2547.800 | 2333.400 | 8.3994 |

## 5.7 Conclusion

In this chapter, we numerically verified our proposed estimation method that is based on artificial neural networks. First, a proof of concept study is conducted on FPGA. The results show that the proposed estimation method is capable of estimating the power consumption at both, component and system-level. The component-level estimation accuracy achieved is less than 1%. System-level power estimation achieved an accuracy of about 8%. Second, an estimation method verification on semi-custom ASIC platforms has been performed. Estimation accuracy on both 45nm CMOS and 15nm FInFET technology were investigated. The method showed an estimation accuracy of about 10%. Scalability studies were also performed and consist in exploring different design sizes. In addition, the estimation speed is also evaluated by assessing the time that is required to perform the estimation on a large amount of design points. Results show that the method is scalable and provides an estimation speed-up factor of 96× compared to the classical power estimation flow (reference). Moreover, some preliminary data for future directions building on this work were also given, especially to improve the method by making it aware of the system operating frequency. This shows that the frequency-aware power estimation is also accurate, with an accuracy of about 9%. A practical demonstration of the proposed estimation method is presented in Appendix B.

# Chapter 6

# Towards Accurate Power Models

## 6.1 Introduction

In this chapter, we present a power measurement methodology that helps in creating an accurate library of models. All of the results presented in the previous chapter (Chapter 5) were based on simulation results. For this purpose, we believe that real measurement data should help in validating our proposed library of models and our estimation method, NeuPow. In addition, we give more accurate and realistic library of models that model the actual power consumption of digital devices.

This work proposes a characterization methodology for low power hardware atomic modules. The characterization is based on a real measurement testbed methodology that helps to accurately measure the power consumption of basic low power components. The measurement considers both types of power sources: the dynamic component as well as the static one. Additionally, we provide a comparison between the results that are obtained from both, simulations and real measurements.

## 6.2 Power Modeling Method

The aim of the power modeling method is to develop components' models, so that it is possible to use them as off-the-shelf solutions for fast power exploration. The complete power modeling flow is depicted in Figure 6.1. The data stimuli generator is used to generate real data waveforms for the FPGA's inputs. The measurement system is connected to the FPGA's core supply in order to measure the FPGA's power consumption. The FPGA measurement system captures and collects the power consumption data during runtime. Using the data recorded by the FPGA measurement system and the controller connected to the data stimulus generator, power modeling can start modeling on the basis of predefined modeling techniques and models templates (using neural networks

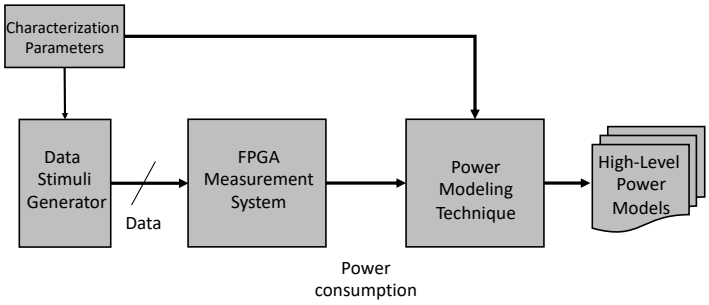



Figure 6.1 – Measurement-based power modeling flow

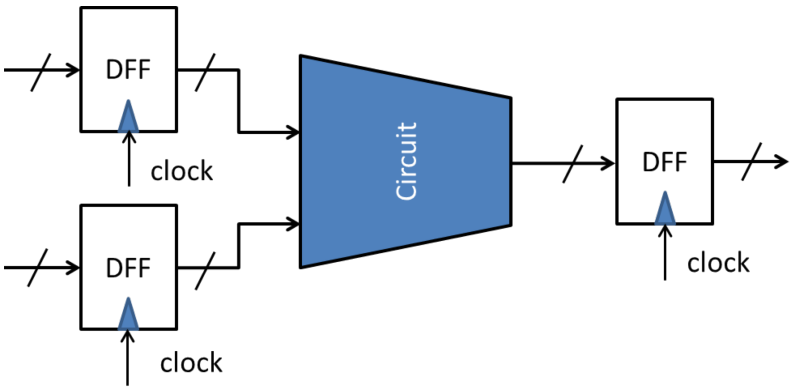



Figure 6.2 – Single Component's Architecture

or regression-based methods). The output of the power modeling algorithms is a high-level power model that maps the power consumption of the component under characterization to the input parameters.

Components models are very important to estimate the power consumption of a composite system, (i.e., a system composed of basic components). To build this library of models, it is very important to characterize the power consumption of single low power modules.

Moreover, our measurement methodology is automated so that efficient and productive power measurements may be performed. The rest of the contributions of this chapter can be summarized as follows:

— We propose a power characterization methodology, and apply it on single low-power components (See Figure 6.2) or IP;

— We quantify the dynamic of the low-power fine grain modules;

— We compare between simulation and measurement results.

In this work, we used the Artix7 (XC7A100T-2FTG256) FPGA, which is designed to address low power applications, and more specifically the battery-based

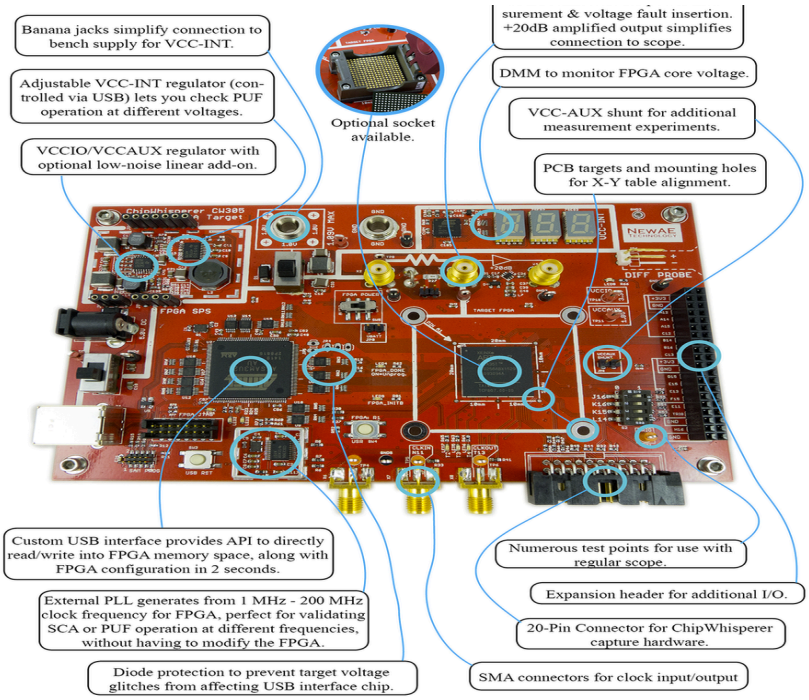



Figure 6.3 – FPGA Board Overview

applications (like portable devices). The FPGA target board (See Figure 6.3) delivers features including a simplified USB interface for communicating to the FPGA, an external phase-locked loop for adjusting frequency response, a programmable power supply, and diode protection. Also available with a special BGA socket, giving you the ability to perform tests on multiple FPGA devices. In addition, an external connectors make it simply to power the FPGA core from a low-noise power supply. This can improve power measurement analysis by reducing noise in the shunt measurement compared to the on-board switching supply. Finally, a shunt resistor is provided in the FPGA core supply. The current in the shunt resistor can be measured using a differential probe as depicted in Figure 6.4.

### 6.2.1 Measurement-based Power Characterization

The characterization process starts by the module implementation and the FPGA configuration step, followed by the measurement of the static power consumption ($P_s$) when data and clock signals are disabled. The next step is to compare the total power measured $P_t$ (when data and clock signals are enabled) and the static power consumption to consider the dynamic power. For this reason, in Figure 6.5, we illustrate the post-synthesized schematic of an exact design on FPGA. The implemented instances refer here to the same component. Note

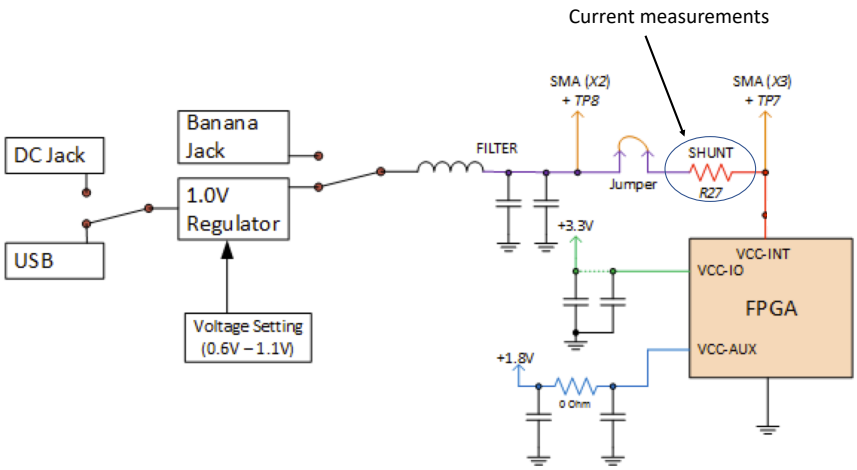



Figure 6.4 – FPGA Power Measurement Schematic

that it is difficult to measure the power consumption variations for low power hardware components. For this purpose, we propose the replication of $n$ times of the same component. This should help in quantifying the power consumption of a single low power component. It is very important to mention also that the input data are shared among all the $n$ replicated instances. As for the outputs, they are not connected to the FPGA's output package pins due to the presence of the constraint file with the $do - not - touch$ command that forces the synthesizer not to touch the unconnected components and signals. Sharing the data inputs among all the $n$ instances guarantees that each instance is equally treated in terms of switching activities, since they are exposed to the same input stimuli.

To get a significant difference in terms of power consumption on FPGA, the static power consumption should be exceeded. With this regard, maximizing the resource utilization on FPGA (e.g., up to 80%) is a way to obtain a sufficient amount of power consumption in order to achieve an important difference between the total, the static and the dynamic power consumption. Note that the total power consumption of the FPGA is measured after the implementation of the $n$ instances (replicated ones). The maximization of the resource utilization can be achieved by replicating $n$ times the same instances (as shown in Figure 6.5) to attain high power consumption, and hence more accurate values.

In order to evaluate the component's dynamic power consumption, the total dynamic power for $n$ instances (replicated components) $P_{Dyn}(total)$ is measured. Then, the component dynamic power $P_{Dyn}(PerComponent)$, which is our target power, is then determined by dividing the total dynamic power by $n$, as shown in Eq. (6.1a) and (6.1b).

$$P_{Dynamic}(Total) = P_{Total} - P_{Static} \tag{6.1a}$$

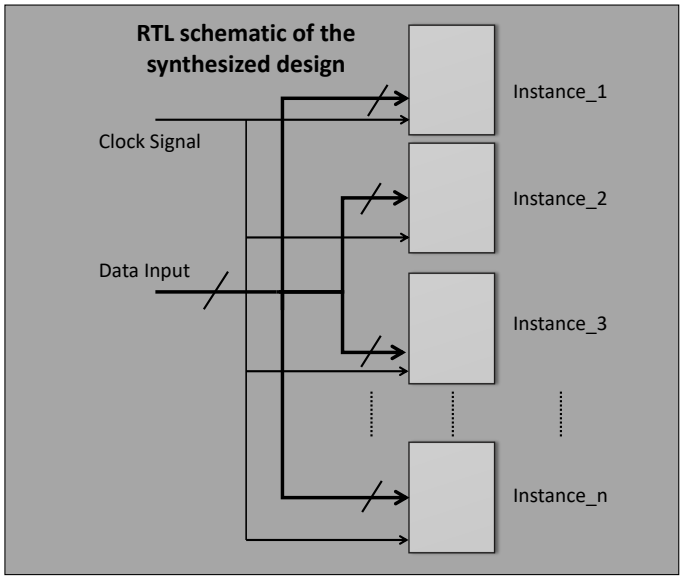



Figure 6.5 – Post-synthesis schematic of n instances

$$P_{Dynamic}(PerComponent) = \frac{P_{Dynamic}(Total)}{n} \tag{6.1b}$$

#### 6.2.1.1 Characterization Parameters

In this section, we briefly describe the factors used in order to characterize power consumption. To facilitate the study, we use the reduced features such as the average switching activity and the static probability. These parameters are good representatives but at the packet level. For this, $D_{in}$ (expressed in Eq. 5.7), and $P_{in}$ (shown in Eq. 5.8) are two factors that can be expressed at this level. Once these factors are quantified, the dynamic power consumption is then averaged during a time window $t$. For instance, we may present the data packet as in Table 6.1, where $K$ is the data packet length (number of the test vectors within a data packet), and $N$ is the data width (here $N = 8$ as an example). Note that, $N$ is highly dependent on the number of data input signals of the component under characterization. In the rest, we use $D_{in}$ and $P_{in}$ to validate the results of the power characterization, and to perform the comparison between the simulation tools, and to provide preliminary components power models.

#### 6.2.1.2 Measurement Process

In this section, we describe the FPGA's measurement system used to measure the power consumption of the FPGA core. As shown in Figure 6.6, our power measurement circuit consists of the following components:

— A $1.0V$ low noise power supply for the FPGA core voltage $V_{DD}$;

Table 6.1 – Data packet: characteristics extraction

| Packet | Bit | Window $t = K \times T_{clock}$ | | | | | | | $AF[n]$ | $P_1[n]$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Width N | bit 7 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0.34 | 0.57 |
| | bit 6 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0.17 | 0.14 |
| | bit 5 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0.34 | 0.14 |
| | bit 4 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0.34 | 0.14 |
| | bit 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.00 | 0.00 |
| | bit 2 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0.34 | 0.85 |
| | bit 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0.17 | 0.71 |
| | bit 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0.67 | 0.42 |
| | | | | | | | | | $D_{in}$ | $P_{in}$ |
| | | | | | | | | | **0.29** | **0.37** |

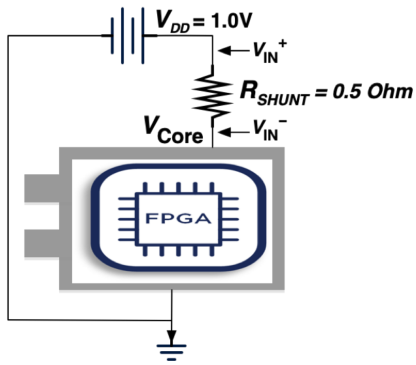



Figure 6.6 – FPGA Chip: Measurement circuit

— A shunt resistor $R_{shunt}$ of 0.5 *Ohms* in series with the FPGA to measure the current;

— A differential probe to measure a voltage difference in order to quantify the total power consumption.

The current that draws over the FPGA core $I_{Shunt}$, it is determined by $I_{Shunt} = V_{Diff}/0.5 = (V_{DD} - V_{Core})/0.5$. The total instantaneous power consumption of the FPGA core can be expressed by $P_{Core} = 2 \times (V_{Core} - V_{Core}^2)$.

A cycle accurate power measurement is then performed, where the dynamic power consumption value is captured at every clock cycle. With this regard, Eq. 6.2 shows how the average power consumption is calculated with respect to a time window $t$, and a sampling frequency that is equal to the clock frequency $f_{clock} = 1/T_{clock}$.

$$\overline{P_{Core}} = \frac{1}{K} \times \sum_{k=0}^{k=K-1} P_{Core}(kT_{clock}) \tag{6.2}$$

Note here that the power samples are acquired at each clock cycle so that the average power consumption is computed during the time window $t$, which is equal to the number of test vectors $K$ sent within a data packet. This number is then multiplied by the clock period $T_{clock}$, and thus time window $t = k \times T_{clock}$.

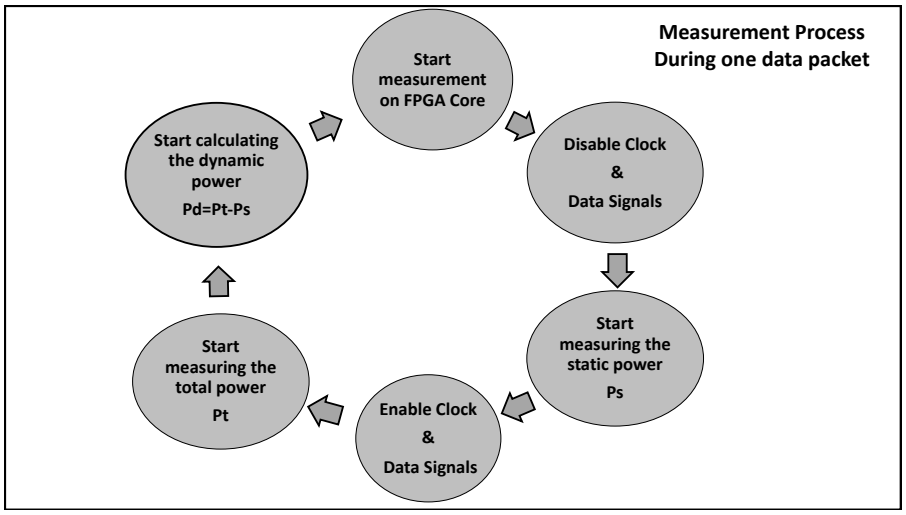



Figure 6.7 – Automated FPGA Measurement Process

An automated measurement process (see Figure 6.7) with respect to data packets has been developed. The measurement bench is composed of a controller responsible for sending the data stimuli to the hardware stimuli generator and targeting the FPGA chip. Then, capturing data samples up to 2.5 GS/s with 18 bits resolution while relying on a high speed digital oscilloscope. As shown in Figure 6.8, data stimuli generation phase based on given characteristics is prepared. For this purpose, a function is implemented to generate data packets with a given $D_{in}$ $P_{in}$. The hardware stimulus generator is used to send data stimulus and clock signals to the FPGA target. The data packets are stored in a specific format. The hardware stimulus system reads the data packets to be sent at the designated frequency. When these data packets begin to get inputs from the hardware block of the FPGA target, the trigger signal that makes the oscilloscope to begin capturing the current and calculate the total power consumed in FPGA.

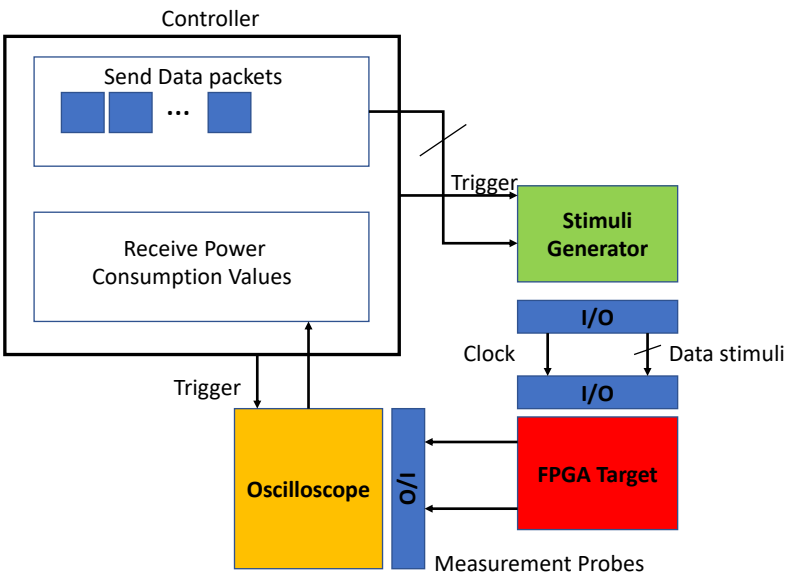



Figure 6.8 – Measurement bench

## 6.3 Experimental Results

In this section, we present the experimental results that help in validating the power characterization method for low power components. These results also show the difference between the simulation tools and the experimental measurements, while providing preliminary measurement-based power models for the basic components.

### 6.3.1 Power Characterization Results

We present several measurements results that validate our proposed methodology in quantifying the power consumption of low power modules implemented on FPGA. The considered cases here are simple 4x4 bits multipliers and adders. In this example, the data packet's width is equal to 8 (i.e., $N = 8$, for 4x4 bits multiplier with 8 inputs). In addition to this, the number of test vectors within the data packet is equal to 100000 (i.e, $K = 100000$). Note that 8-bit component inputs are considered due to hardware constraints (only 16 accessible inputs / outputs).

To illustrate, Figure 6.9 and Figure 6.10 summarize the results corresponding to the different modules (i.e., adders and multipliers blocks), the different number of implemented instances, and the different activity factor of the primary inputs is presented. These results are obtained under the following settings: 5 MHz for the clock frequency (maximum available clock frequency) and $P_{in} = 0.5$ for the average static probability. The results show that the module power consumption

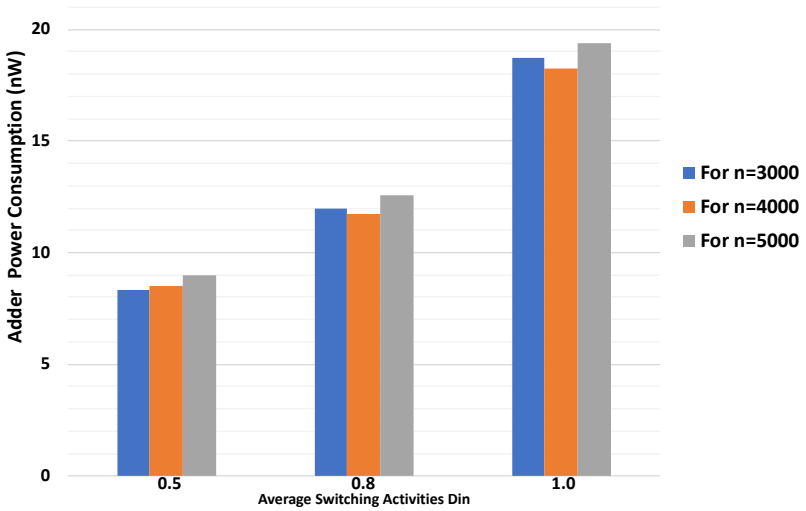



Figure 6.9 – Adder Power Consumption Per Component

can be characterized by only relying on maximizing the number of similar modules (instances) in all the FPGA pieces, independently of the module's size. For instance, the dynamic power of adders and multipliers is computed by dividing the total dynamic power consumption by the number of implemented instances. The static power is independent of the number of instances and achieves around $37mW$. Meanwhile the dynamic power increases with the number of instances, the activity factor and the frequency. Therefore, for a given scenario (in terms of frequency and activity factor) the total dynamic power consumption becomes dependent of the number of implemented instances. Thus, the total dynamic power consumption can be quantified and the dynamic power consumption of the component can be deduced.

Finally, Tables 6.2 and 6.3 summarize the results corresponding to the various modules (i.e., multipliers and adders blocks), the different number of implemented instances, and the distinct activity factor of the primary inputs. Note here that $P_{in}$ is set to 0.5 in order to attain a maximum average switching activity ($D_{in} = 1$). For instance, for $D_{in} = 1$, we have roughly the same dynamic power consumption per component independently of the number of instances. An example for this is $P_C(2500Multipliers) = 36.4nW$, $P_C\ (2800Multipliers) = 35.7nW$ and $P_C(3000Multipliers) = 35.7nW$. Moreover, we consider here the average component dynamic power consumption $AVGC_i$ that corresponds to the $i$-th $D_{in}$ for each module as a reference for computing the maximum error which is defined as follows:

$$\%e = \left(\frac{max|AVGC - P_C|}{AVGC}\right) \times 100 \tag{6.3}$$

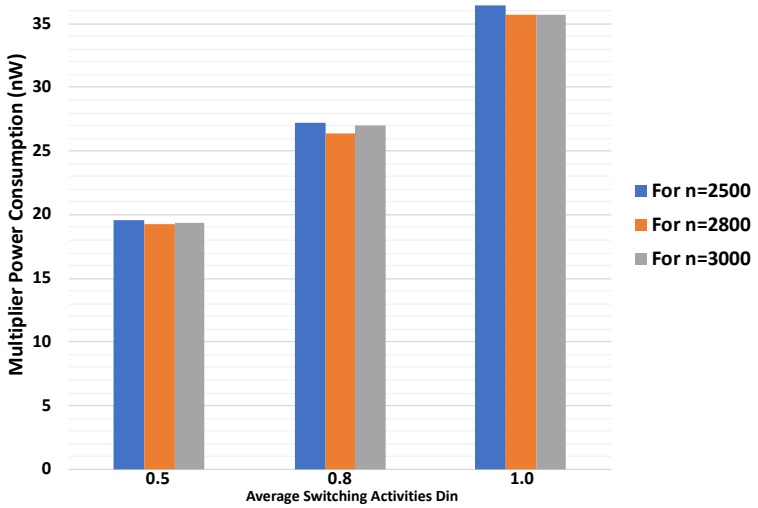



Figure 6.10 – Multiplier Power Consumption Per Component

Table 6.2 – Experimental results: Adder power characterization results

| | **ADDER** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $D_{in}$ | **0.5** | | | **0.8** | | | **1.0** | | |
| **n** | 3000 | 4000 | 5000 | 3000 | 4000 | 5000 | 3000 | 4000 | 5000 |
| **PT(mW)** | 25.0 | 34.0 | 45.0 | 36.0 | 47.0 | 63.0 | 56.0 | 73.0 | 97.0 |
| **PC(nW)** | **8.3** | **8.5** | **9.0** | **12.0** | **11.75** | **12.6** | **18.7** | **18.25** | **19.4** |
| **AVGC(nW)** | 8.6 | | | 12.12 | | | 18.79 | | |
| **%Error** | **4.65** | | | **3.96** | | | **3.24** | | |

Based on this metric (%), it may be noted that the highest relative error is approximately 5%. To this end, the power characterization technique, which is based on the replication of the same component (in order to maximize resource usage), may provide accurate values in this case.

### 6.3.2 Tool Vs Measurement Results

In this section, we present a comparison between the power consumption results that are obtained from the (Xilinx Xpower analyzer) tool and the measurement results presented above. For this, we use the adders and multipliers components. For the measurement setup, we can describe the setting as follows:

— Artix7 (XC7A100T-2FTG256) FPGA;

— $4 \times 4$ bit adder/multiplier;

— Clock frequency is 5 MHz;

Table 6.3 – Experimental results: Multiplier power characterization results

| | **MULTIPLIER** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $D_{in}$ | **0.5** | | | **0.8** | | | **1.0** | | |
| **n** | 2500 | 2800 | 3000 | 2500 | 2800 | 3000 | 2500 | 2800 | 3000 |
| **PT(mW)** | 49.0 | 54.0 | 58.0 | 68.0 | 74.0 | 80.0 | 91.0 | 100.0 | 107.0 |
| **PC(nW)** | **19.6** | **19.3** | **19.4** | **27.2** | **26.4** | **27.0** | **36.4** | **35.7** | **35.7** |
| **AVGC(nW)** | 19.44 | | | 26.87 | | | 35.94 | | |
| **%Error** | **0.82** | | | **1.75** | | | **1.28** | | |

— Number of implemented instances ( $n = 3000$ and $n = 4000$ for the multiplier and the adder, respectively);

— The average static probability is about 0.5 (50% of the time is logic high).

Based on this configuration, we use the average activity factor $D_{in}$ to conduct both, the measurement as well as the estimation. The same setup is used for both measurement and estimation. The power consumption values are obtained by varying the average activity factor of the primary inputs in both.

Figure 6.11 and Figure 6.12 indicate the power measurement and estimation values as a function of the average activity factor $D_{in}$. In order to calculate the relative error between the estimation and the measurement, we consider the measurement values to be our reference values. By this, the average dynamic power consumption can be marked by $P_{avg}^{ref}$, where the $ref$ is measured, and the average power consumption (denoted by $P_{avg}^{XPA}$, where $XPA$ refers to the power estimation tool) is based on the estimation.

Table 6.4 shows the average measured power obtained ($P_{avg}^{ref}(component)$), and the estimated ($P_{avg}^{XPA}(component)$) for both multiplier and adder components. In this regard, the overestimation error is more than +47% ($\%RME = +47.9574\%$) for the multiplier, while for the adder is about +43% ($\%RME = +43.6566\%$).

It is true that the estimation using the provided estimation tool, called XPA, is very fast. However, as previously shown, the power estimations that are based on the Xilinx Xpower Analyzer (XPA) provide less accurate results comparing to those obtained using our measurement testbed. As observed, the minimum relative error between the real values and the estimated ones is about 43%, which is no longer accurate.

Table 6.4 – Measurement Vs Estimation Tool (Xilinx Xpower Analyzer)

| **Component** | $P_{avg}^{ref}$ $(mW)$ | $P_{avg}^{XPA}$ $(mW)$ | **%Mean Relative Error** |
|---|---|---|---|
| Mult(4X4) | 50.97 | 75.42 | +47.95 |
| Add(4X4) | 30.33 | 43.58 | +43.65 |

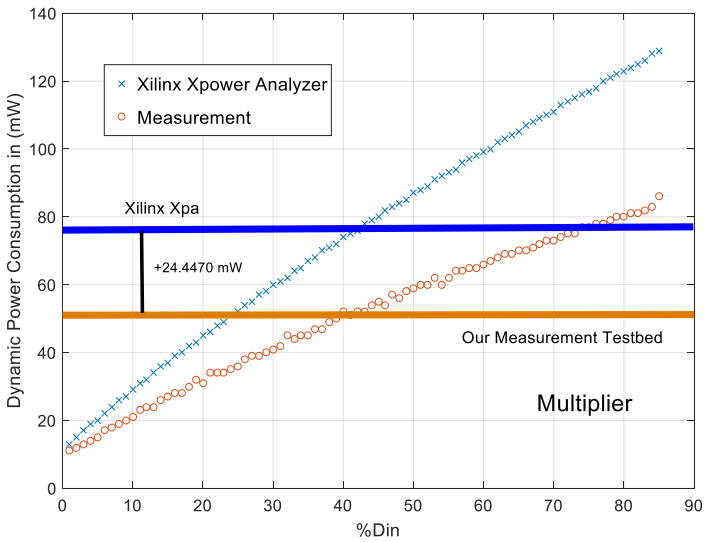



Figure 6.11 – Measurement-based Vs Xilinx Xpa for Multiplier

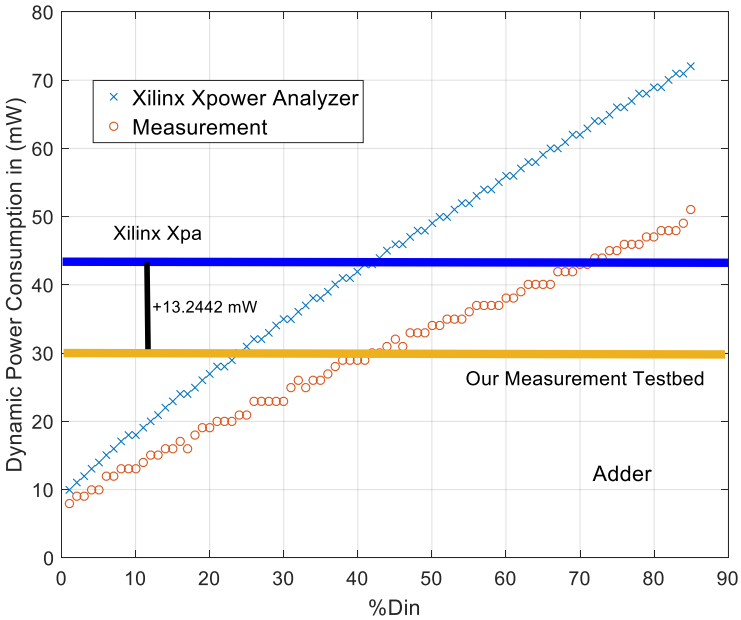



Figure 6.12 – Measurement-based Vs Xilinx Xpa for Adder

### 6.3.3 Preliminary Measurement-based Power Models

In this section, we provide preliminary power models for basic hardware components, such as the adders and the multipliers. Our preliminary models are approximated using a simple linear relationship as in [111]. These models may provide fast and accurate power estimations with more realistic power values,

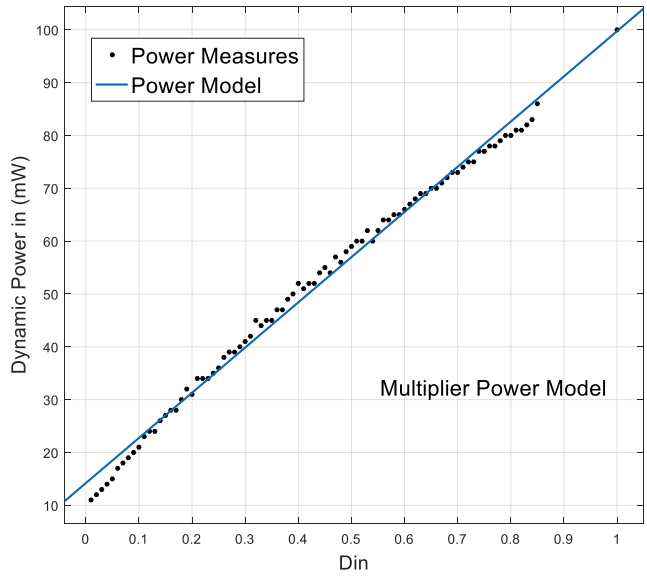



Figure 6.13 – Measurement-based Multiplier Power Model

even better than those obtained by the tool with a maximum error of about 5%.

Concerning the multiplier component, the obtained power model can be expressed as follows:

$$P_{Mult} = 85.55 \times D_{in} + 14.19 \tag{6.4}$$

where $D_{in}$ represents the average switching activity of the primary inputs, which ranges from 0 to 1. This model (See Figure 6.13) presents a root mean square error of about 1.824mW ($RMSE = 1.824mW$) and a relative root mean square error ($\%R-RMSE$) of about $\pm 3.5781\%$. Based on this, modeling results indicate that it is feasible to rely on this model to have a preliminary power approximation at component-level for hardware multipliers.

Regarding the adder component, it is possible to express the power model as follows:

$$P_{Add} = 49.56 \times D_{in} + 8.698 \tag{6.5}$$

The adder model (Figure 6.14) shows a root mean square error of about 1.3mW ($RMSE = 1.3mW$) and a relative root mean square error ($\%R - RMSE$) of about $\pm 4.2852\%$. Adder's results also prove that it is possible to use this model to have a preliminary power estimation at component-level.

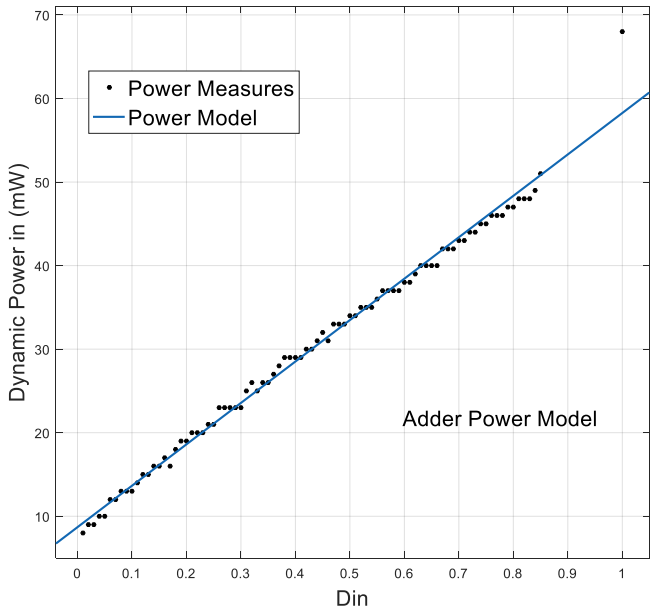



Figure 6.14 – Measurement-based Adder Power Model

## 6.4 Conclusion

In this work, we demonstrate a new methodology that allows designers to explore and quantify the power consumption of hardware modules on FPGAs. This methodology, which is design, frequency and data dependent, is based on maximizing the resource utilization on FPGAs. In this chapter, we validate our approach using real experiments that are based on measurements. Experimental results show that maximizing the resource utilization on FPGAs, and dividing the total dynamic power of the whole implemented design is enough to quantify the power consumption of an individual low power hardware module, with a maximum error of 5%. In addition, we benefit from this methodology to compare the results obtained from a tool such as Xilinx Xpower Analyzer (XPA) against our measurement method. Results have shown about 40% of relative error when using the tool to estimate the power consumption. Finally, we deliver basic power models that are based on real measurement. These models can be useful to estimate the power consumption for individual components accurately and quickly.

# Chapter 7

# Conclusion and Perspectives

## 7.1 Conclusion

Power consumption in digital systems has become a primary matter for both signal processing and hardware designers. Therefore, power reduction has become a major challenge, that necessitates more efficient power estimation methods. In addition, the development of high-level tools for digital signal processing and hardware design forces the proposal of high-level power estimation techniques. In this thesis, we present *NeuPow* as a system-level FPGA/ASIC power estimation methodology. The proposed methodology is capable of providing a fast and accurate power estimation at a very early design phase. The approach is based on the adoption of Artificial Neural Networks (ANNs) to model the power and the behavior of a set of digital components (adders, multipliers, registers etc.), that are commonly used in digital signal processing applications. The power estimation leverages on additional behavioral ANN models to propagate the signal activity throughout components. Both models are capable of providing extremely accurate power data at component- and system-level. Finally, we present how the designer can use our power estimation technique to assess different possible design alternatives in a very fast way, thanks to our developed library of power models that enable the re-use of these models, leading to a greater design productivity.

***Chapter 3*** reviewed the power measurement techniques, and delivered a detailed discussion while presenting the limitations and advantages for each of these techniques. Our literature review analysis showed that using an external power solution to measure the power consumption is better (in terms of accuracy) than measuring it using on-board solutions. However, measuring power consumption is still an expensive and time consuming solution. To this aim, we classified the power estimation techniques, that are usually used to characterize the power consumption of digital circuits. The analyzed power estimation techniques showed different level of accuracy and estimation speed. We noticed that although the simulation-based power estimation technique is more accurate than

the probabilistic-based one, but the latter showed a high estimation speed comparing to the former. Then, a review about the proposed solutions that enable fast and accurate power estimations is presented. In this regard, power modeling techniques such as Analytical, Table-based, Regression and Neural networks are reviewed, while defining different metrics for methods comparison. We set quantitative metrics that enable the evaluation of the modeling techniques. Our analysis proved that the regression-based and neural networks methods are the best in terms of accuracy and estimation speed. By comparing the neural networks to the regression-based techniques, most of the studied works showed that neural-based power estimations provide low relative errors (less than 5%). Consequently, we adopted the neural networks approach to model the power consumption, and these models are based on simulation-based power data. Finally, we clustered the literature review to facilitate the extraction of the common characteristics such as the accuracy, the estimation speed, the model scalability and the modeling effort. The clustering results proved that most of the modeling works lack the model scalability. Recall that the model scalability helps the designers in exploring the power consumption of different design choices at higher-level of abstractions. Based on this, we extended the work described in [103], and we investigated the model scalability of the neural networks modeling technique in order to provide system-level power estimations.

***Chapter 4*** presented a propagation-based power estimation methodology, by exploiting the neural networks modeling technique. Accurate power and behavioral characterization is described. After that, the detailed power and behavioral modeling processes are provided. Then, we outlined our dedicated component power model library. The objective of this library is to investigate the various design options in terms of power consumption. Consequently, we described our estimation method that is a component-based approach, where the design is considered as a set of interconnected components exchanging data. Each component here is independently characterized in terms of power, leveraging on two different models. Each model works on a set of features that are extracted from the component input binary data. The first model (power model) aims at determining power consumption from input features, whereas the second one (behavioral model) maps the input data features to the output data features. After a building step, which consists in interconnecting all the components of the design, a system-level simulation is performed to propagate the data features throughout the system, and determine the contribution of each component in the total consumed power. Finally, we expanded the power estimation technique, called NeuPow, to include the operating frequency of the system.

***Chapter 5*** evaluated our proposed estimation method, called NeuPow, on FPGA and ASIC platforms.

First, we proposed a new approach to estimate the dynamic power estimation on FPGA at system-level [112]. This is achieved by decomposing a digital system into basic components, propagating the statistical information among them, and

summing up the estimated power of each component. The feasibility of the propagation-based power estimations have been verified in a use case based on the implementation of a basic neural network [113, 114]. The presented method allows designers to perform early power estimation and efficient power reduction. We have shown that our method provides good results, since the error is less than 8%. In addition to the good accuracy, a speed-up of the design process is noted. After that, we have presented the estimation methodology, utilizing neural networks as key models that are used at high-level of abstractions. To this aim, two types of neural networks have been introduced to estimate both power and signal activity (signal rate and static probability). These models demonstrate a very good precision when considering single components with a modeling error of less than 0.01%. When using these models to assess more complex functions (composite system), the outcomes achieved are less accurate in the estimation of power owing to modeling errors and interconnecting power. Note that the highest errors of estimation achieved at the system-level is about 7.5%.

Second, we assessed *NeuPow* on the ASIC platform [110]. Our method has been improved by including the glitch effects within the ANN models. Then, the proposed methodology has been deeply assessed under several aspects: 1) in terms of the adopted modeling technique, by comparing ANNs against widely adopted techniques in power modeling, such as regression-based ones, 2) in terms of accuracy, by considering several use cases, 3) in terms of scalability, by exploring different design sizes, and 4) in terms of speed, by evaluating the time required to run the estimation of a large amount of design points. Moreover, some preliminary data on future directions of the work are given, especially with respect to the improvement of NeuPow by making it aware of the system operating frequency. NeuPow and the adopted ANN models demonstrated to be more precise than regression-based approaches in estimating power. The proposed methodology offers, in general, a very good estimation accuracy, with an error that is always less than 9% for different use cases, for different technology points, and when taking the operating frequency into account. Moreover, the approach proved to be scalable with an average error that is not depending on the system size. NeuPow is also fast, outperforming the classical Cadence®power estimation flow with a speed-up factor of about 96×. In addition, we evaluated *NeuPow* on the most recent technologies, such as the 15 nm FinFET-based Open Cell Library [109]. Results show that *NeuPow* offers a very good estimation accuracy (less than 9%), which is valid for both technologies (45nm CMOS and the 15nm FinFET), while taking a new metric into account, i.e., the operating frequency. This estimation accuracy is independent of the design sizes, making *NeuPow* immune to the design size. Consequently, the proposed library-based methodology allows designers to explore the design space in a very flexible way. Being generic, our approach may be then adopted to any black box semi-custom ASIC components and complex systems.

***Chapter 6*** presents our measurement methodology that allows designers to

accurately quantify the power consumption of low power hardware modules on FPGAs. This methodology is based on maximizing the resource utilization on FPGAs. Experimental results validate that maximizing resource utilization on FPGAs and dividing the total dynamic power of the entire implemented design is sufficient to quantify the power consumption of an individual low-power hardware module with a maximum error of 5%. Our methodology allows us to compare the findings acquired from the Xilinx Xpower Analyzer (XPA) tool and our measurement-based values. We have shown that, when using the estimation tool, the relative error is about 40% comparing to the measurement. Finally, we provide fundamental power models that are based on our real measurements, which can be helpful for the accurate estimation of the power consumption at component-level.

## 7.2 Future Work and Research Directions

This thesis presented a neural-based power estimation technique for high-level design tools based on a dedicated library. Future directions on our proposed estimation methodology can be presented for both FPGA and ASIC platforms.

At the FPGA-level, our estimation methodology is based on a dedicated library of models, which is highly target dependent. To this end, a possible extension is to provide generic power models for all FPGA targets disregarding the FPGA family. Based on this, an additional factor can be added to the generic power model. This factor may reflects many physical details such as the internal architecture and the FPGA technology. Additionally, next possible extension is to adapt the measurement methodology presented in Chapter 6 to perform real behavioral characterization. Then, power and behavioral models can be adopted based on neural networks. With this regard, measurement-based neural models will be delivered, and should lead to very accurate and realistic power estimations.

At the ASIC-level, our method is also verified on both 45nm and 15nm technology. We plan to further refine our technique by assessing in the efficiency of the suggested methodology for variable frequency and technology. In specific, we strive to bring the latter or other parameters based on it (e.g., operating voltage) as an extra input to the power model, as we have already suggested for the frequency, in order to provide a frequency-aware technology-aware power estimation.

Finally, we aim to build a system-on-chip power estimator platform that is based on our power estimation methodology. The power estimator can be incorporated with any high-level modeling instrument, such as MATLAB and LabView. By doing so, the gap between the high-level and the transistors-level tools will be smaller. To evaluate the different possible architectural scenarios in a fast way, it is possible to perform the exploration without the need to go deeply into the hardware design steps, which is a time consuming process. In addition, this will

be useful to help both signal processing and digital design researchers to quickly assess the power consumption and to choose the most power efficient design.

# Appendix A

# Motivations

## A.1 Motivations

As discussed in Chapter 4, the proposed power estimation strategy is based on two assumptions. The first one is that the power estimation can be obtained based on the signal statistics propagation among the connected components. As for the second assumption, it states that the power modeling can be performed using the neural networks that are considered as generic templates to model the component's power. To this aim, we numerically motivate the impact of the signal statistics propagation on the system-level power estimation. Then, we empirically prove that neural networks can be considered as a generic template model.

### A.1.1 Motivation To Propagation-based Power Estimation

In this section, we demonstrate the impact of the signal statistics propagation on the power estimation approach. The approach consists in simply adding the individual power consumption of each component by taking their switching activities (relative to a given input pattern) into account.

#### A.1.1.1 Evaluation Method

The results that are presented in Section 4.3.1 show that the signal activity of components, as well as the time of signals at logic high, constitute a representative information and a strong relationship to component's dynamic power consumption. These factors can be then used to model the power consumption of a given component. As it has been previously mentioned in Chapter 4 each component's model can be modeled by means of two sub-models ($f$ and $g$). The first sub-model ($f$) is used to estimate the dynamic power from the signal activity of the operator's inputs and from the percentage high parameter. $f$ provides an

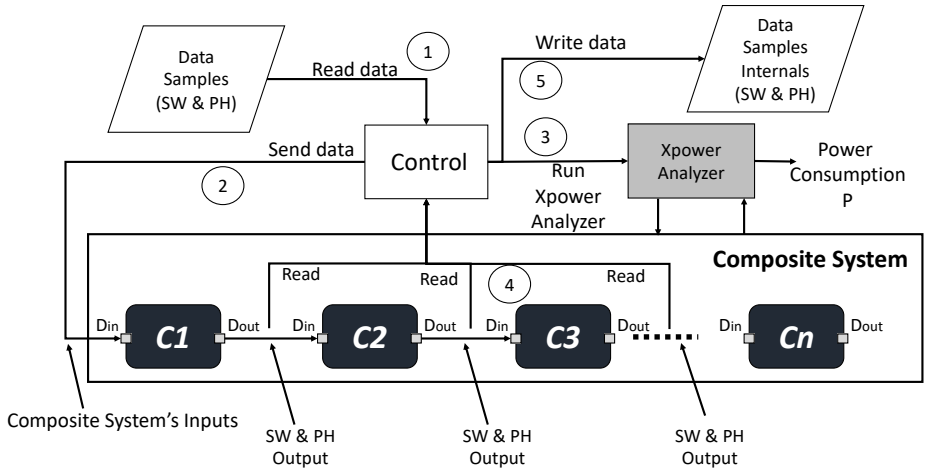



Figure A.1 – Power estimation of a composite system using the Vivado Xpower Analyzer

average of the energy consumed during a period of $1s$. As for the second sub-model ($g$), it helps to estimate the signal activity of the outputs (as well as the percentage high) according to the operator's inputs. In the results below, Xilinx Xpower analyzer [61] ($f$ and $g$) models are used to perform the propagation in order to obtain system-level power estimation.

A composite system has been described at RTL, and implemented on the FPGA target. Then, the composite system has been characterized in terms of power consumption using the Xilinx Xpower analyzer. During power characterization, scripts that have the following functionality (See Figure A.1) are performed:

— (1) Switching activities and percentage high are read by XPA from a file;

— (2) Data are asserted to the composite system's inputs.

— (3) Power estimation process (XPA) are executed.

— (4) Internal switching activities and the percentage high at the component's inputs are read. Note that, these data will be useful to individually estimate the power consumption of separately implemented components.

— (5) Switching activities and percentage high data (to be used in the assessment) are stored.

Then, we implement each of the described components (that are used to compose the system), and we perform the power characterization at the component-level. The extracted data (internal switching activities and percentage high) from the system-level power characterization are then exploited to individually estimate the power consumption, as depicted in Figure A.2.

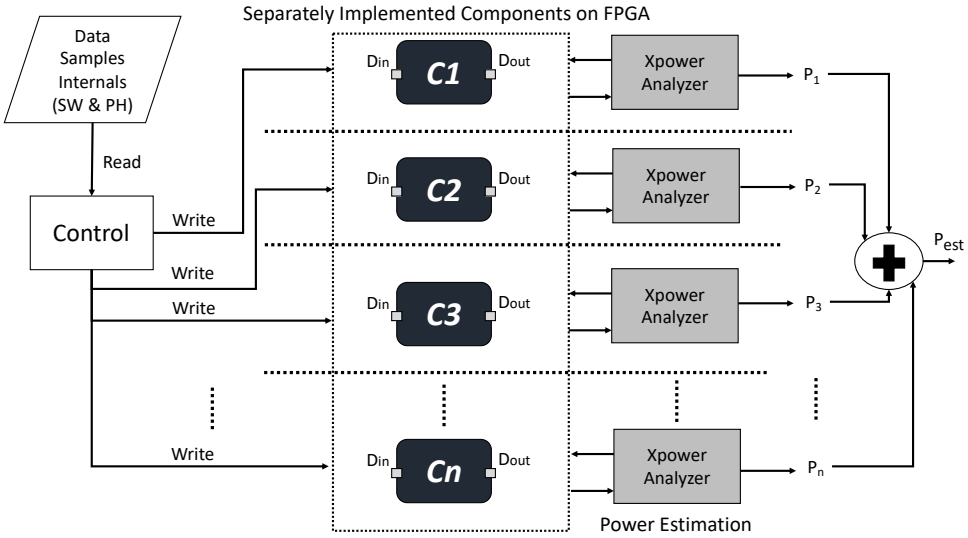



Figure A.2 – Power estimation for each component separately

To better describe our proposed methodology, we show here a simple example. For a system composed of $n$ components, the total average power consumption can be expressed as follows:

$$P_{est} = \frac{1}{M} \sum_{j=1}^{M} \sum_{i=1}^{n} P_{ij} \tag{A.1}$$

where $i$ and $j$ correspond to the $i$-th component and $j$-th input sample, respectively. M is the number of samples.

In the rest, numerical motivation is provided. This should help to assume that a composite system power estimation can only be achieved by adding the power consumption of each component, taking into account the corresponding signal and static probability of component's inputs.

#### A.1.1.2 Numerical Results

The purpose of this section is to demonstrate the feasibility of our approach on a use case that can be easily described. We have chosen here to consider two basic components (i.e., a basic multiplier accumulator (MAC) and a parallel to serial (P2S) converter). These two components are massively used in lots of applications, like signal and image processing, communications, and control. Starting with defining the MAC component, it consists in multiplying two operands and then accumulating the result in a dedicated register. Figure A.3 describes an architecture overview of the MAC component ( for example, the used MAC is a $16 \times 16$ bits), while illustrating a simple and generic representation of a MAC unit. The parallel to serial (P2S) component aims at receiving 4 inputs (coded

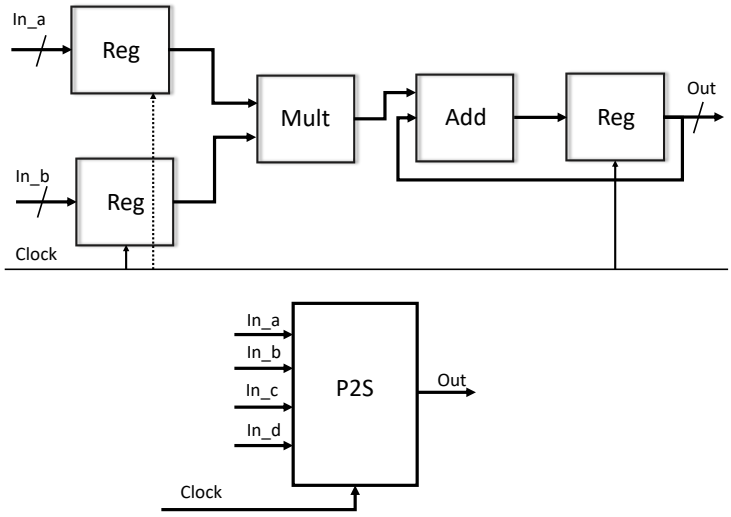



Figure A.3 – MAC and P2S components

into 16 bits) in parallel and transmitting each of these inputs in a serial way to the next block. Moreover, Table A.1 shows the implementation details in terms of hardware resources. Note that the resources used to implement the MAC and P2S components on the FPGA target are selected to be LUT-based (the DSP blocks are not used). Note here that this choice has been made to only take the logic elements into account. Regarding the hardware platform, the target FPGA, is a Virtex-7 FPGA: $xc7z045ffg900-2$, and the clock frequency is 200 MHz.

Table A.1 – Required resources for both MAC and P2S units at RTL (xc7z045ffg900-2 FPGA)

| Resource | MAC | P2S |
|---|---|---|
| LUT | 185 | 25 |
| D Flip-Flop | 107 | 26 |
| I/O | 84 | 87 |
| BUFG | 1 | 1 |

In our approach, we want to estimate the power consumption of a composite system that is a small Multi-layer perceptron neural network that can be made up of MAC and P2S components. A complete architectural view of a hardware implementation of such network is described in Figure A.4. As we can see in this figure, the case study deals with the implementation of a $(8 \times 4 \times 3)$ Multi-Layer Perceptrons (MLP). The $A$ inputs consist of 16 bits that are connected to a first MAC layer in order to compute the results of the hidden layer. The $W$ inputs, are used as second inputs of the MAC components to provide the weights

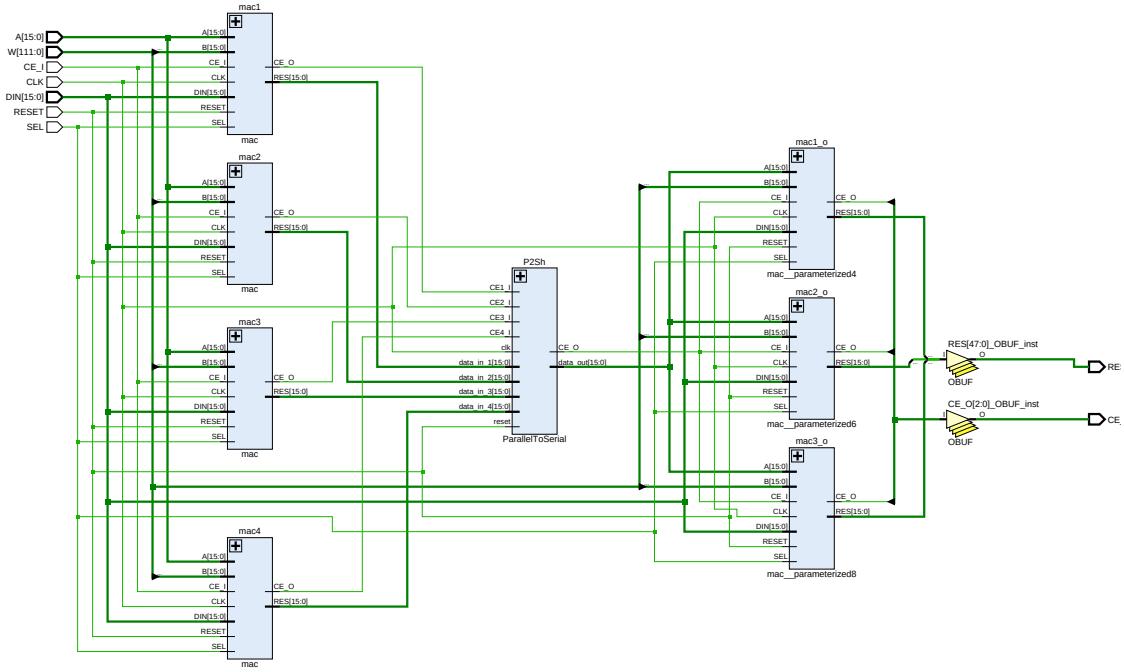



Figure A.4 – RTL schematic of a neural network implemented on FPGA

of the connections between the neurons. At the output of the first MAC layer, the results correspond to the outputs of the hidden layer. These results are then sent to the P2S module to be serialized and provided to the next MAC layer that is used to compute the 3 outputs of the neural network. A full implementation of the neural network has been performed on a xc7z045ffg900-2 FPGA target and Table A.2 shows the implementation results in terms of used resources. Data stimuli has been applied at the input of the system (neural network) and a timing simulation has been performed. All internal signals activity has been recorded in a trace file for further exploitation. In addition, power estimation results of the composite system (neural network) have been also obtained. A low-level power estimation tool has been used to estimate the power consumption of the complete neural network. The average dynamic power consumption has been obtained based on 10000 different system's inputs combinations of switching activities and percentage high. By this, the average dynamic power of the complete design (See Figure. A.1) obtained is about 239.0 mW, which we consider as the reference value $P$.

To study the propagation effect, we run the power estimations but now at the component-level. We separately implemented the MAC and the P2S components, and run the power estimations using the recorded internal switching activities and the percentage high obtained during the system-level estimations. After that, the

Table A.2 – Resource of the implemented neural network at RTL

| Resource | Utilization |
|---|---|
| LUT | 1313 |
| D Flip-Flops | 775 |
| I/O | 199 |
| BUFG | 1 |

power estimation at the component-level for the different components has been led. The dynamic power consumption has been recorded after each signal rate and percentage high injection to the components' inputs. By summing up the obtained power values for each component, it was then possible to get the total dynamic power estimation of the composite system (See Fig. A.2). The average dynamic power consumption obtained over 10000 samples is about 221.0 mW (i.e., $P_{est} = 221.0mW$). In order to measure the accuracy of the approach, the percentage of error (in terms of power) relative to the real implementation can be calculated as follows:

$$\%MRAE = \frac{|P - P_{est}|}{P} \times 100 \tag{A.2}$$

where $P$ is the power that results from the real implementation of the neural network, and $P_{est}$ stands for the total average power that has been obtained using our proposed method. Based on this, only 7.5 % of error is obtained using the proposed method. We believe that the expected error value of 7.5 % is due to the fact that we did not consider the power consumption of the interconnections among the components in the full design.

Numerical results show that propagating the signal activity rates only and % of logic high is enough to obtain a good level of estimation accuracy for a composite system, and to speed-up the design assessments. The aforementioned results show that it is possible to use the signal statistics to estimate the power consumption of a full design, while decomposing the full system into a set of components. Then, the total power consumption of the full design can be obtained by summing up the power consumption of each component. Finally, it is clear that this may allow designers to get a fast and quite accurate power estimation for their designs at high-level of abstraction.

### A.1.2 Why Neural Networks?

In this section, we numerically motivate the use of the neural networks technique as a generic modeling tool. More specifically, we propose the neural model as a fixed power model that can model the power consumption of any digital component without the need of knowing the internal architecture or to alter the model template. After that, the automatic power characterization with defined

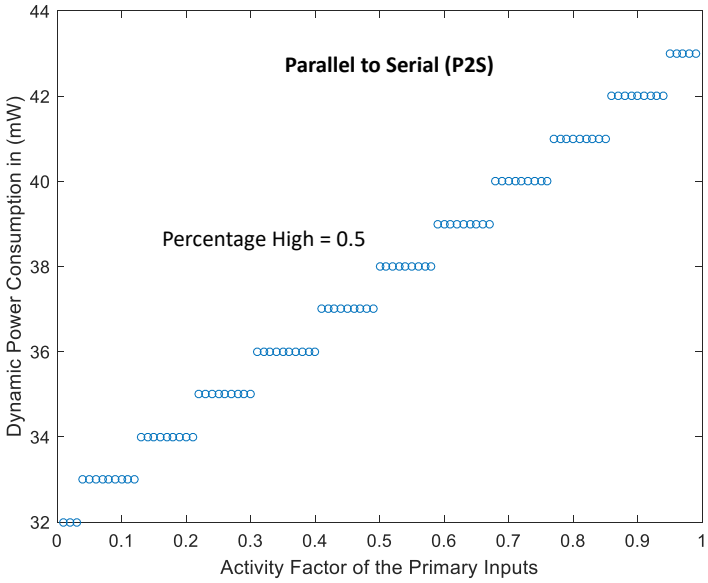
parameters, such as the signal statistics of the primary inputs, can be adopted to construct the power models.

To demonstrate the feasibility of our approach, we have implemented different components on FPGA platform. The selected platform is the Xilinx Virtex-7 FPGA ($xc7z045ffg900-2$), the operating frequency is 200 MHz, and the considered components are the Parallel to Serial (P2S) and the Multiply-accumulate (Mac). The power characterization, is performed according to the parameters presented in Section 4.3.1. Then, the activity factors and the percentage high of the primary inputs have been used to model the dynamic power consumption.

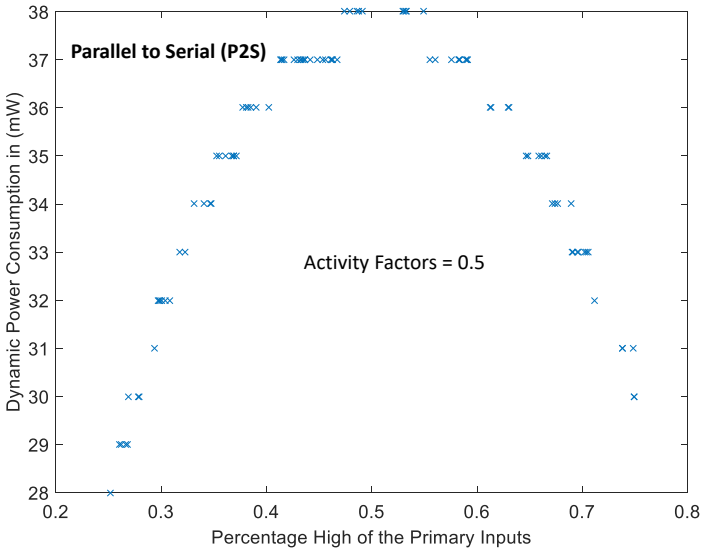
To illustrate the effect of these factors on power characterization and modeling, the corresponding results are presented in Figure A.5 and Figure A.6. For instance, Figure A.5 shows the dynamic power consumption data trend with respect to the average activity factor (See Figure. A.5a) and to the average percentage high (See Figure A.5b). We notice here that the relationship is nonlinear, and the power consumption data trend does not have a consistent pattern. According to the obtained results, this prevents the use of a simple linear relationship to map the dynamic power to the signal statistics. Hence, it forces us to rely on a robust modeling technique, such as the neural network. Despite, it is also possible, in some cases, to use simple linear relationship to model the power consumption as depicted in Figure A.6a: the power consumption here is linear with respect to the average activity factor only. However, we may notice that the dynamic power consumption also depends on the average percentage high, where the power data trend inconsistently behaves as shown in Figure A.6b. Consequently, neural networks may provide an off-the-shelf power modeling solution, which is capable to preform the modeling regardless of the data trend (linear or nonlinear), and can easily approximate the power behaviour of a digital component. This, in turn, helps to have a generic template model for all components.

### A.1.3 Conclusion

As shown above, the results related to the proposed method that is based on signal statistics propagation among the connected power models present a low relative error of about 7.5 %. Based on this preliminary result, it is possible to rely on the propagation of the signal statistics to quickly estimate the power consumption at high-level of abstraction. Additionally, the results that are related to the neural networks model clearly show that the component's power behavior has an inconsistent shape with respect to the signal statistics. With this regard, neural networks can be used to model the power consumption and can be considered as a generic power model.

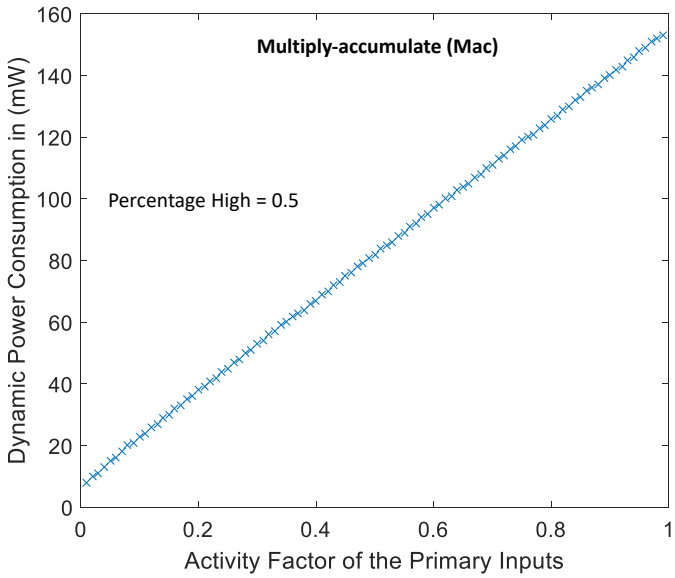



(a) Power consumption Vs Activity factor for percentage high=0.5

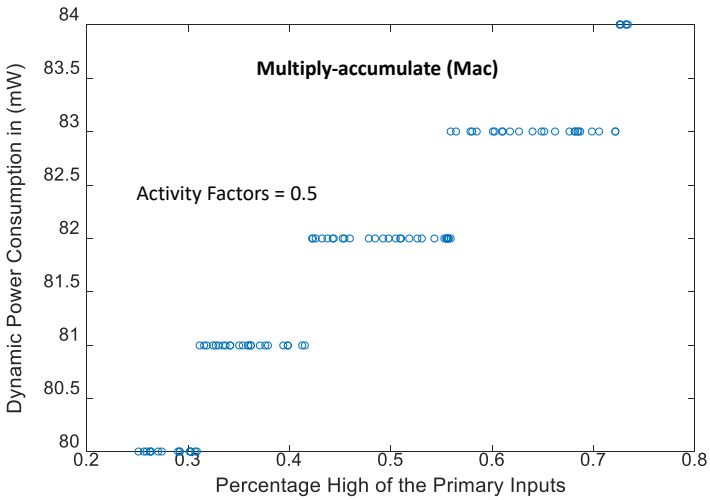



(b) Power consumption Vs Percentage high for activity factor=0.5

Figure A.5 – Parallel to serial power consumption Vs the signal statistical parameters




(a) Power consumption Vs Activity factor for percentage high=0.5




(b) Power consumption Vs Percentage high for activity factor=0.5

Figure A.6 – Multiply-accumulate power consumption Vs the signal statistical parameters

# Appendix B

# Demonstrations

## B.1 Introduction

In this section, a demonstration of our proposed estimation methodology is presented, while showing its practical use with a high-level simulation tool. The demonstration settings are listed as follows:

— High-Level Simulation Tool: MathWorks, Simulink;

— Use-case: a 4 tap image filter, implemented on the 45nm ASIC technology;

— Used Models: Register($9bits$), Multiply-accumulate($18\times18$), and Shift&clip ($18bits$);

— Operating frequency: 100 MHz;

— Number of samples: 10000 samples; sample time $t = 0.01ms$;

— Power Consumption: Average Dynamic power in mW.

## B.2 Power Estimation Description

In this part, a power simulation use-case is described. Figure B.1 shows an overview of the composite system, where the neural-based component's models are connected. Using our developed library of models in Simulink, it is then possible to quickly assess any digital signal processing algorithm in terms of power consumption. The power estimation is performed as function of both operating frequency and data stimuli that are depicted in Figure B.2. Regarding the wires that are connecting the models, the aim of them is to provide the signal statistics to the next models (power and behavioral). Additionally, some debugging nodes are presented. This, in turn, helps in the debugging process during the power simulations.

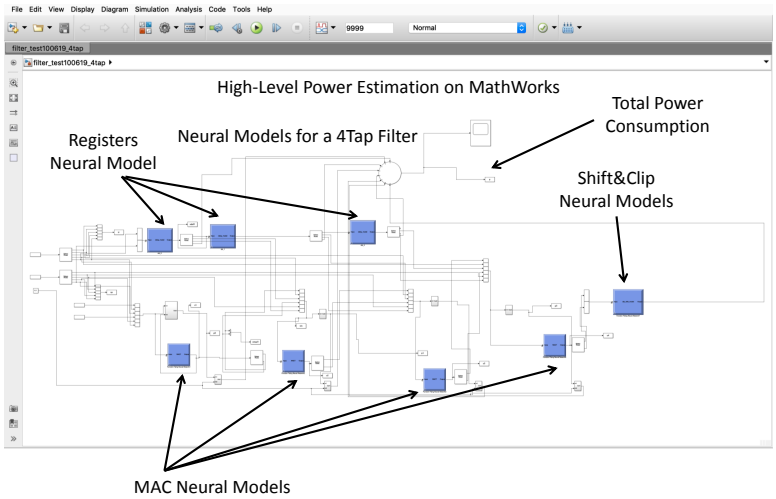



Figure B.1 – High-level power estimation for a 4tap filter composite system

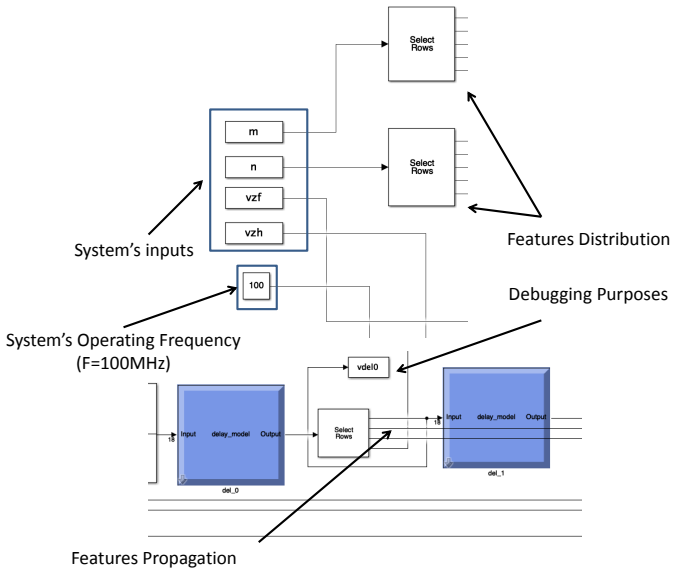



Figure B.2 – A Zoom on MAC and P2S components' models features propagation

## B.3 Image Filter Power Profile

Neural-based power simulation of a 4tap image filter at a high-level of abstraction is presented. The results of the simulation are captured in Figure B.3 and Figure B.4. As shown in these figures, the dynamic power consumption values are obtained as function of the sample's inputs and operating frequency. The dynamic power consumption at each time sample (0.01ms) is monitored. As for

the total time needed to explore the power consumption of 10000 samples, it is about 21 seconds on an Intel Core i5 working at 2.3 GHz.

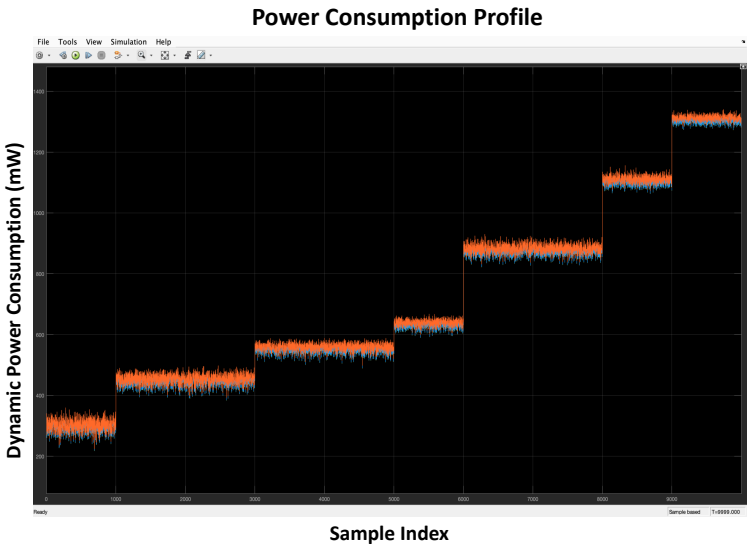



Figure B.3 – Dynamic power consumption profile for a 4tap image filter implemented on the 45nm ASIC technology

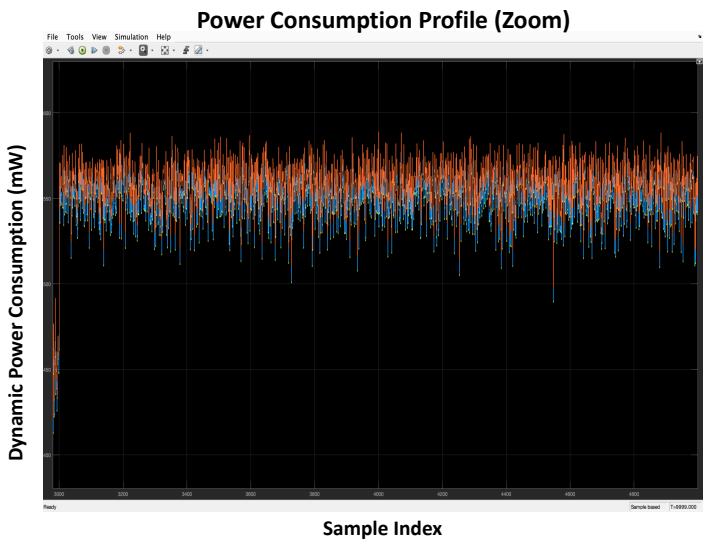



Figure B.4 – A Zoom on the dynamic power consumption

## B.4 Conclusion

The objective of this Appendix is to show that the power consumption can be estimated using neural-based power models. Therefore, the designers need only to build their digital signal processing scheme using our built-in component model library to perform power simulations. As previously demonstrated, a fast and flexible power estimation method can be made using a high-level tool (e.g., graphical based). Finally, this method may serve both digital signal processing and hardware designers.

# Nomenclature

**Abbreviations**

ANN Artificial Neural Network

ASIC Application Specific Integrated Circuit

CAD Computer Aided Design

CLB Configurable Logic Block

CMOS Complementary Metal Oxide Semiconductor

DSP Digital Signal Processing

HDL Hardware Description Languages

IP Intellectual Property

MLP Multi-Layer Perceptrons

PL Programmable Logic

RAM Random Access Memory

SAIF Switching Activity Interchange Format

TTM Time-to-Market

VCD Value Change Dump

VLSI Very Large Scale Integration

XPE Xilinx Power Estimator

# Bibliography


[1] Cisco, "Cisco: Internet of things at-a-glance." `https://www.cisco.com/c/dam/en/us/products/collateral/se/internet-of-things/at-a-glance-c45-731471.pdf`, 2018.

[2] T. Fanni *et al.*, "Power and Clock Gating Modelling in Coarse Grained Reconfigurable Systems," in *Conf. on Computing Frontiers*, 2016.

[3] F. Palumbo *et al.*, "Modelling and Automated Implementation of Optimal Power Saving Strategies in Coarse-Grained Reconfigurable Architectures," *JECE*, vol. 2016, p. 5, Nov. 2016.

[4] A. Raghunathan, S. Dey, and N. K. Jha, "Register-transfer level estimation techniques for switching activity and power consumption," in *Proceedings of International Conference on Computer Aided Design*, pp. 158–165, Nov 1996.

[5] S. Gupta and F. N. Najm, "Power macromodeling for high level power estimation," in *Proceedings of the 34th annual Design Automation Conference*, pp. 365–370, ACM, 1997.

[6] M. Barocci, L. Benini, A. Bogliolo, B. Riccó, and G. De Micheli, "Lookup table power macro-models for behavioral library components," in *Proceedings IEEE Alessandro Volta Memorial Workshop on Low-Power Design*, pp. 173–181, IEEE, 1999.

[7] S. Gupta and F. N. Najm, "Power modeling for high-level power estimation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 1, pp. 18–29, 2000.

[8] Y. A. Durrani, T. Riesgo, and F. Machado, "Statistical power estimation for register transfer level," in *Proceedings of the International Conference Mixed Design of Integrated Circuits and System, 2006. MIXDES 2006.*, pp. 522–527, IEEE, 2006.

[9] R. Jevtic and C. Carreras, "Power estimation of embedded multiplier blocks in fpgas," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, pp. 835–839, May 2010.

[10] T. Jiang, X. Tang, and P. Banerjee, "Macro-models for high level area and power estimation on fpgas," in *Proceedings of the 14th ACM Great Lakes*

*Symposium on VLSI*, GLSVLSI '04, (New York, NY, USA), pp. 162–165, ACM, 2004.

[11] L. Deng, K. Sobti, and C. Chakrabarti, "Accurate models for estimating area and power of fpga implementations," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1417–1420, March 2008.

[12] B. M. H. Chalbi Najoua, Boubaker Mohamed, "Accurate dynamic power model for fpga basedimplementations," in *IJCSI International Journal of Computer Science Issues*, vol. 9, March 2012.

[13] A. Borovyi, V. Konstantakos, V. Kochan, V. Turchenko, A. Sachenko, and T. Laopoulos, "Using neural network for the evaluation of power consumption of instructions execution," in *2008 IEEE Instrumentation and Measurement Technology Conference*, pp. 676–681, IEEE, 2008.

[14] L. Hou, X. Wu, and W. Wu, "Neural network based power estimation on chip specification," in *The 3rd International Conference on Information Sciences and Interaction Sciences*, pp. 187–190, IEEE, 2010.

[15] A. Association *et al.*, "Itrs-international technology roadmap for semiconductors, 2011 edition, system drivers," 2011.

[16] J.-P. Deschamps, *Hardware implementation of finite-field arithmetic.* McGraw-Hill, Inc., 2009.

[17] Duong Tran, Kyung Ki Kim, and Yong-Bin Kim, "Power estimation in digital cmos vlsi chips," in *2005 IEEE Instrumentation and Measurement Technology Conference Proceedings*, vol. 1, pp. 317–321, May 2005.

[18] D. Helms, R. Eilers, M. Metzdorf, and W. Nebel, "Leakage models for high-level power estimation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, pp. 1627–1639, Aug 2018.

[19] G. Paim, L. M. G. Rocha, T. G. Alves, R. S. Ferreira, E. A. C. da Costa, and S. Bampi, "A power-predictive environment for fast and power-aware asic-based fir filter design," in *2017 30th Symposium on Integrated Circuits and Systems Design (SBCCI)*, pp. 168–173, Aug 2017.

[20] N. Nasirian, R. Soosahabi, and M. A. Bayoumi, "Probabilistic analysis of power-gating in network-on-chip routers," *IEEE Transactions on Circuits and Systems II: Express Briefs*, pp. 1–1, 2018.

[21] J. Rabaey, *Low Power Design Essentials.* Springer Publishing Company, Incorporated, 1st ed., 2009.

[22] P.-E. Gaillardon, E. Beigne, S. Lesecq, and G. D. Micheli, "A survey on low-power techniques with emerging technologies: From devices to systems," *J. Emerg. Technol. Comput. Syst.*, vol. 12, pp. 12:1–12:26, Sept. 2015.

[23] P. Gupta and A. B. Kahng, "Quantifying error in dynamic power estimation of cmos circuits," in *Fourth International Symposium on Quality Electronic Design, 2003. Proceedings.*, pp. 273–278, March 2003.

[24] Xilinx, "About post-synthesis and post-implementation timing simulation." `https://www.xilinx.com/support/documentation/sw_manuals/xilinx2012_3/ug900-vivado-logic-simulation.pdf`, 2012.

[25] G. I. Webb, M. J. Pazzani, and D. Billsus, "Machine learning for user modeling," *User modeling and user-adapted interaction*, vol. 11, no. 1-2, pp. 19–29, 2001.

[26] P. Harrington, *Machine Learning in Action*. Greenwich, CT, USA: Manning Publications Co., 2012.

[27] H. N. Mhaskar, "Approximation properties of a multilayered feedforward artificial neural network," *Advances in Computational Mathematics*, vol. 1, pp. 61–80, Feb 1993.

[28] J.-G. Attali and G. Pagès, "Approximations of functions by a multilayer perceptron: a new approach," *Neural networks*, vol. 10, no. 6, pp. 1069–1081, 1997.

[29] J. J. Moré, "The levenberg-marquardt algorithm: implementation and theory," in *Numerical analysis*, pp. 105–116, Springer, 1978.

[30] L. Shang, A. S. Kaviani, and K. Bathala, "Dynamic power consumption in virtex$^{TM}$-ii fpga family," in *Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays*, pp. 157–164, ACM, 2002.

[31] Xilinx, "Ti power solutions for measuring power on xilinx evaluation kits," 2018.

[32] A. Nafkha and Y. Louet, "Accurate measurement of power consumption overhead during fpga dynamic partial reconfiguration," in *2016 International Symposium on Wireless Communication Systems (ISWCS)*, pp. 586–591, Sept 2016.

[33] M. A. Rihani, F. Nouvel, J. Prévotet, M. Mroue, J. Lorandel, and Y. Mohanna, "Dynamic and partial reconfiguration power consumption runtime measurements analysis for zynq soc devices," in *2016 International Symposium on Wireless Communication Systems (ISWCS)*, pp. 592–596, Sept 2016.

[34] M. Kim, J. Kong, and S. W. Chung, "Enhancing online power estimation accuracy for smartphones," *IEEE Transactions on Consumer Electronics*, vol. 58, pp. 333–339, May 2012.

[35] R. Jevtic and C. Carreras, "Power measurement methodology for fpga devices," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 1, pp. 237–247, 2011.

[36] D. Elleouet, N. Julien, and D. Houzet, "A high level soc power estimation based on ip modeling," in *Proceedings 20th IEEE International Parallel Distributed Processing Symposium*, pp. 4 pp.–, April 2006.

[37] J. Oliver, F. Veirano, D. Bouvier, and E. Boemo, "A low cost system for self measurements of power consumption in field programmable gate arrays," *Journal of Low Power Electronics*, vol. 13, no. 1, 2017.

[38] R. Bonamy, D. Chillet, S. Bilavarn, and O. Sentieys, "Power consumption model for partial and dynamic reconfiguration," in *Reconfigurable Computing and FPGAs (ReConFig), 2012 International Conference on*, pp. 1–8, IEEE, 2012.

[39] L. W. Nagel, "Spice2: A computer program to simulate semiconductor circuits," *Ph. D. dissertation, University of California at Berkeley*, 1975.

[40] C. X. Huang, B. Zhang, A.-C. Deng, and B. Swirski, "The design and implementation of powermill," in *In Proceedings ofthe International Symposium on Low Power Design*, Citeseer, 1995.

[41] M. Bushnell and V. Agrawal, *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*, vol. 17. Springer Science & Business Media, 2004.

[42] A. Boliolo, L. Benini, G. De Micheli, and B. Riccó, "Gate-level power and current simulation of cmos integrated circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 5, no. 4, pp. 473–488, 1997.

[43] A. Bogliolo, L. Benini, and B. Riccò, "Power estimation of cell-based cmos circuits," 1996.

[44] B. George, "Power analysis and characterization for semicustom design," in *Proc. Intl. Workshop on Low Power Design*, pp. 215–218, 1994.

[45] R. Burch, F. N. Najm, P. Yang, and T. N. Trick, "A monte carlo approach for power estimation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 1, no. 1, pp. 63–71, 1993.

[46] T.-L. Chou and K. Roy, "Statistical estimation of sequential circuit activity," in *Proceedings of the 1995 IEEE/ACM international conference on Computer-aided design*, pp. 34–37, IEEE Computer Society, 1995.

[47] L.-P. Yuan, C.-C. Teng, and S.-M. Kang, "Statistical estimation of average power dissipation in cmos vlsi circuits using nonparametric techniques," in *Proceedings of 1996 International Symposium on Low Power Electronics and Design*, pp. 73–78, IEEE, 1996.

[48] D. Marculescu, R. Marculescu, and M. Pedram, "Stochastic sequential machine synthesis targeting constrained sequence generation," in *33rd Design Automation Conference Proceedings, 1996*, pp. 696–701, IEEE, 1996.

[49] and, , and M. Pedram, "Stratified random sampling for power estimation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, pp. 465–471, June 1998.

[50] R. Burch, F. Najm, P. Yang, and T. Trick, "Mcpower: A monte carlo approach to power estimation," in *ICCAD*, vol. 92, pp. 90–97, 1992.

[51] M. G. Xakellis and F. N. Najm, "Statistical estimation of the switching activity in digital circuitsy," in *31st Design Automation Conference*, pp. 728–733, IEEE, 1994.

[52] Y. Park and E. Park, "Statistical power estimation of cmos logic circuits with variable errors," *Electronics Letters*, vol. 34, no. 11, pp. 1054–1056, 1998.

[53] Z. Chen, K. Roy, and T.-L. Chou, "Efficient statistical approach to estimate power considering uncertain properties of primary inputs," *IEEE Transactions on very large scale integration (VLSI) systems*, vol. 6, no. 3, pp. 484–492, 1998.

[54] C.-Y. Tsui, M. Pedram, and A. M. Despain, "Efficient estimation of dynamic power consumption under a real delay model," in *Proceedings of 1993 International Conference on Computer Aided Design (ICCAD)*, pp. 224–228, IEEE, 1993.

[55] R. Marculescu, D. Marculescu, and M. Pedram, "Switching activity analysis considering spatiotemporal correlations," in *Proceedings of the 1994 IEEE/ACM international conference on Computer-aided design*, pp. 294–299, IEEE Computer Society Press, 1994.

[56] P. H. Schneider and S. Krishnamoorthy, "Effects of correlations on accuracy of power analysis-an experimental study," in *Proceedings of 1996 International Symposium on Low Power Electronics and Design*, pp. 113–116, IEEE, 1996.

[57] S. Garg, S. Tata, and R. Arunachalam, "Static transition probability analysis under uncertainty," in *IEEE International Conference on Computer Design: VLSI in Computers and Processors, 2004. ICCD 2004. Proceedings.*, pp. 380–386, IEEE, 2004.

[58] synopsys, "Delivers accurate dynamic and leakage power analysis." `https://www.synopsys.com/content/dam/synopsys/implementation&signoff/datasheets/primepower-ds.pdf`, 2018.

[59] Cadence®, "Genus synthesis solution," 2018. `https://www.cadence.com/content/cadence-www/global/en_US/home/tools/digital-design-and-signoff/synthesis/genus-synthesis-solution.html`, 2018-08-28.

[60] Xilinx, "Xilinx power estimator user guide: Ug440," 2017.

[61] xilinx, "Xpower analyzer overview." `https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/isehelp_start.htm#xpa_c_overview.htm`, 2014.

[62] P. E. P. Estimators, "and power analyzer," *URL: https://www. altera. com/support/support-resources/operation-and-testing/power/pow-powerplay. html (visited on 04/09/2016).*

[63] J. Lamoureux and S. J. E. Wilton, "Activity estimation for field-programmable gate arrays," in *2006 International Conference on Field Programmable Logic and Applications*, pp. 1–8, Aug 2006.

[64] J. B. Goeders and S. J. E. Wilton, "Versapower: Power estimation for diverse fpga architectures," in *2012 International Conference on Field-Programmable Technology*, pp. 229–234, Dec 2012.

[65] X. Tang, E. Giacomin, G. D. Micheli, and P. E. Gaillardon, "FPGA-SPICE: A simulation-based architecture evaluation framework for FPGAs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2019.

[66] C.-Y. Tsui, M. Pedram, and A. M. Despain, "Exact and approximate methods for calculating signal and transition probabilities in fsms," in *31st Design Automation Conference*, pp. 18–23, IEEE, 1994.

[67] J. Monteiro, S. Devadas, and B. Lin, "A methodology for efficient estimation of switching activity in sequential logic circuits," in *31st Design Automation Conference*, pp. 12–17, IEEE, 1994.

[68] F. N. Najm, "Transition density: A new measure of activity in digital circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 2, pp. 310–323, 1993.

[69] F. N. Najm, "A survey of power estimation techniques in vlsi circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 446–455, 1994.

[70] T. Arslan, A. Erdogan, and D. Horrocks, "Low power design for dsp: methodologies and techniques," *Microelectronics Journal*, vol. 27, no. 8, pp. 731–744, 1996.

[71] S. Reda and A. N. Nowroz, "Power modeling and characterization of computing devices: A survey," *Found. Trends Electron. Des. Autom.*, vol. 6, pp. 121–216, Feb. 2012.

[72] P. Landman, "High-level power estimation," in *Proceedings of the 1996 International Symposium on Low Power Electronics and Design*, ISLPED '96, (Piscataway, NJ, USA), pp. 29–35, IEEE Press, 1996.

[73] B. S. Landman and R. L. Russo, "On a pin versus block relationship for partitions of logic graphs," *IEEE Transactions on Computers*, vol. C-20, pp. 1469–1479, Dec 1971.

[74] F. N. Najm, "A survey of power estimation techniques in vlsi circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, pp. 446–455, Dec 1994.

[75] F. N. Najm, "Transition density: a new measure of activity in digital circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, pp. 310–323, Feb 1993.

[76] H. A. Hassan, M. Anis, and M. Elmasry, "Total power modeling in fpgas under spatial correlation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, pp. 578–582, April 2009.

[77] J. A. Clarke, A. A. Gaffar, and G. A. Constantinides, "Parameterized logic power consumption models for fpga-based arithmetic," in *International Conference on Field Programmable Logic and Applications, 2005.*, pp. 626–629, Aug 2005.

[78] Zhanping Chen and K. Roy, "A power macromodeling technique based on power sensitivity," in *Proceedings 1998 Design and Automation Conference. 35th DAC. (Cat. No.98CH36175)*, pp. 678–683, June 1998.

[79] Z. Chen, K. Roy, and E. K. P. Chong, "Estimation of power sensitivity in sequential circuits with power macromodeling application," in *1998 IEEE/ACM International Conference on Computer-Aided Design. Digest of Technical Papers (IEEE Cat. No.98CB36287)*, pp. 468–472, Nov 1998.

[80] R. Jevtic, B. Jovanovic, and C. Carreras, "Power estimation of dividers implemented in fpgas," in *Proceedings of the 21st Edition of the Great Lakes Symposium on Great Lakes Symposium on VLSI*, GLSVLSI '11, (New York, NY, USA), pp. 313–318, ACM, 2011.

[81] X. Tang, L. Wang, and H. Xu, "An accurate dynamic power model on fpga routing resources," in *2012 IEEE 11th International Conference on Solid-State and Integrated Circuit Technology*, pp. 1–3, Oct 2012.

[82] J. Das, A. Lam, S. J. E. Wilton, P. H. W. Leong, and W. Luk, "An analytical model relating fpga architecture to logic density and depth," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, pp. 2229–2242, Dec 2011.

[83] K. K. W. Poon, S. J. E. Wilton, and A. Yan, "A detailed power model for field-programmable gate arrays," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 10, pp. 279–302, Apr. 2005.

[84] Y. Leow, A. Akoglu, and S. Lysecky, "An analytical model for evaluating static power of homogeneous fpga architectures," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 6, 12 2013.

[85] H. Mehri and B. Alizadeh, "Analytical performance model for fpga-based reconfigurable computing," *Microprocessors and Microsystems*, vol. 39, no. 8, pp. 796 – 806, 2015.

[86] Y. A. Durrani and T. Riesgo, "Power estimation for intellectual property-based digital systems at the architectural level," *Journal of King Saud University-Computer and Information Sciences*, vol. 26, no. 3, pp. 287–295, 2014.

[87] S. Gupta and F. N. Najm, "Power macro-models for dsp blocks with application to high-level synthesis," in *Proceedings. 1999 International Symposium*

*on Low Power Electronics and Design (Cat. No.99TH8477)*, pp. 103–105, Aug 1999.

[88] G. Bernacchia and M. C. Papaefthymiou, "Analytical macromodeling for high-level power estimation," *1999 IEEE/ACM International Conference on Computer-Aided Design. Digest of Technical Papers (Cat. No.99CH37051)*, pp. 280–283, 1999.

[89] L. Shang and N. K. Jha, "High-level power modeling of cplds and fpgas," in *Proceedings 2001 IEEE International Conference on Computer Design: VLSI in Computers and Processors. ICCD 2001*, pp. 46–51, Sept 2001.

[90] A. Bogliolo, L. Benini, G. De Micheli, G. De Micheli, and G. De Micheli, "Regression-based rtl power modeling," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 5, pp. 337–372, July 2000.

[91] A. Lakshminarayana, S. Ahuja, and S. Shukla, "High level power estimation models for fpgas," in *2011 IEEE Computer Society Annual Symposium on VLSI*, pp. 7–12, July 2011.

[92] G. Verma, V. Khare, and M. Kumar, "More precise fpga power estimation and validation tool (fpev_tool) for low power applications," *Wireless Personal Communications*, vol. 106, pp. 2237–2246, Jun 2019.

[93] J. Laurent, N. Julien, E. Senn, and E. Martin, "Functional level power analysis: An efficient approach for modeling the power consumption of complex processors," in *Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1*, DATE '04, (Washington, DC, USA), pp. 10666–, IEEE Computer Society, 2004.

[94] E. Senn, J. Guillot, D. Chillet, C. Belleudy, S. Niar, O. Zendra, and C. Samoyeau, "Open Power and Energy Optimization Platform and Estimator (Open-People) ANR Project," in *Design, Automation & Test in Europe (DATE 2011), University Booth*, (Grenoble, France), pp. –, Mar. 2011.

[95] N. Abdelli, A. Fouilliart, N. Julien, and E. Senn, "High-level power estimation of fpga," in *2007 IEEE International Symposium on Industrial Electronics*, pp. 925–930, June 2007.

[96] F. Scarselli and A. C. Tsoi, "Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results," *Neural Netw.*, vol. 11, pp. 15–37, Jan. 1998.

[97] W.-T. Hsieh, C.-C. Shiue, and C.-N. Liu, "A novel approach for high-level power modeling of sequential circuits using recurrent neural networks," in *2005 IEEE International Symposium on Circuits and Systems*, pp. 3591–3594, IEEE, 2005.

[98] X. Gao, Y. Yan, Y. Cao, and W. Qiang, "Neural network macromodel for high-level power estimation of cmos circuits," in *2005 International Conference on Neural Networks and Brain*, vol. 2, pp. 1009–1014, IEEE, 2005.

[99] K. Roy, "Neural network based macromodels for high level power estimation," in *International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007)*, vol. 2, pp. 159–163, IEEE, 2007.

[100] A. Suissa, O. Romain, J. Denoulet, K. Hachicha, and P. Garda, "Empirical method based on neural networks for analog power modeling," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 5, pp. 839–844, 2010.

[101] A. A. Sagahyroon and J. A. Abdalla, "Dynamic and leakage power estimation in register files using neural networks," *Circuits and Systems*, vol. 3, no. 02, p. 119, 2012.

[102] P. Ramanathan, B. Surendiran, and P. Vanathi, "Power estimation of benchmark circuits using artificial neural networks," *Pensee*, vol. 75, no. 9, 2013.

[103] J. Lorandel, J.-C. Prévotet, and M. Hélard, "Efficient modelling of fpga-based ip blocks using neural networks," in *2016 International Symposium on Wireless Communication Systems (ISWCS)*, pp. 571–575, IEEE, 2016.

[104] L. Shang, A. S. Kaviani, and K. Bathala, "Dynamic power consumption in virtex™-ii fpga family," in *Proceedings of the 2002 ACM/SIGDA Tenth International Symposium on Field-programmable Gate Arrays*, FPGA '02, (New York, NY, USA), pp. 157–164, ACM, 2002.

[105] K. K. Poon, A. Yan, and S. J. Wilton, "A flexible power model for fpgas," in *International Conference on Field Programmable Logic and Applications*, pp. 312–321, Springer, 2002.

[106] X. Liu and M. C. Papaefthymiou, "Incorporation of input glitches into power macromodeling," in *2002 IEEE International Symposium on Circuits and Systems. Proceedings (Cat. No. 02CH37353)*, vol. 4, pp. IV–IV, IEEE, 2002.

[107] B. C. Csáji, "Approximation with artificial neural networks," *Faculty of Sciences, Etvs Lornd University, Hungary*, vol. 24, p. 48, 2001.

[108] F. Scarselli and A. C. Tsoi, "Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results," *Neural networks*, vol. 11, no. 1, pp. 15–37, 1998.

[109] I. Nangate, "Nangate freepdk15 open cell library.," 2019. `https://http://www.nangate.com/?page_id=2328`, 2018-08-28.

[110] Y. Nasser, C. Sau, J.-C. Prévotet, T. Fanni, F. Palumbo, M. Hélard, and L. Raffo, "Neupow: artificial neural networks for power and behavioral modeling of arithmetic components in 45nm asics technology," in *Proceedings of the 16th ACM International Conference on Computing Frontiers*, pp. 183–189, ACM, 2019.

[111] C. Carreras Vaquer and R. Jevtic, "Analytical high-level power model for lut-based components," 2009.

[112] Y. Nasser, J.-C. Prevotet, and M. Hélard, "Power modeling on fpga: a neural model for rt-level power estimation," in *Proceedings of the 15th ACM International Conference on Computing Frontiers*, pp. 309–313, ACM, 2018.

[113] Y. Nasser, J.-C. Prévotet, M. Hélard, and J. Lorandel, "Dynamic power estimation based on switching activity propagation," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 1–2, IEEE, 2017.

[114] Y. Nasser, J.-C. Prévotet, M. Hélard, and J. Lorandel, "Power estimation on fpgas based on signal information propagation through digital operators," in *2017 Sensors Networks Smart and Emerging Technologies (SENSET)*, pp. 1–4, IEEE, 2017.

INSA de RENNES
Service des Formations

# AVIS DU JURY SUR LA REPRODUCTION DE LA THESE SOUTENUE

**Titre de la thèse:**
An efficient Computer-Aided Design Methodology for FPGA&ASIC High-Level Power Estimation Based on Machine Learning

**Nom Prénom de l'auteur : NASSER YEHYA**

Membres du jury :
- Monsieur PREVOTET Jean-Christophe
- Monsieur SHAFIQUE Muhammad
- Monsieur SOUDRIS Dimitrios
- Madame PALUMBO Francesca
- Madame HELARD Maryline
- Monsieur CHEVALIER Pascal

Président du jury : CHEVALIER Pascal

Date de la soutenance : 25 Novembre 2019

Reproduction de la these soutenue

☒ Thèse pouvant être reproduite en l'état
☐ Thèse pouvant être reproduite après corrections suggérées

Fait à Rennes, le 25 Novembre 2019

Signature du président de jury

Chevalier

Le Directeur,

M'hamed DRISSI

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

INSA RENNES

**Titre :** Une méthodologie efficace d'estimation de puissance au haut niveau sur FPGA et ASIC basée sur l'apprentissage statistique

**Mots clés :** Consommation d'énergie, FPGA, ASIC, Apprentissage statistiques, Modélisation, Estimation

**Résumé :** Aujourd'hui, des systèmes numériques avancés sont nécessaires pour mettre en œuvre des fonctionnalités complexes. Cette complexité impose au concepteur de respecter différentes contraintes de conception telle que la performance, la surface, la consommation électrique et le délai de mise sur le marché. Pour effectuer une conception efficace, les concepteurs doivent rapidement évaluer les différentes architectures possibles. Dans cette thèse, nous nous concentrons sur l'évaluation de la consommation d'énergie afin de fournir une méthode d'estimation de puissance rapide, précise et flexible. Nous présentons NeuPow qui une méthode s'appliquant aux FPGA et ASIC. Cette approche système est basée sur des techniques d'apprentissage statistique. Notamment, nous exploitons les réseaux neuronaux pour aider les concepteurs à explorer la consommation d'énergie dynamique. NeuPow s'appuie sur la propagation des signaux à travers des modèles neuronaux connectés pour prédire la consommation d'énergie d'un système composite à haut niveau d'abstraction. La méthodologie permet de prendre en compte la fréquence de fonctionnement et les différentes technologies de circuits (ASIC et FPGA). Les résultats montrent une très bonne précision d'estimation avec moins de 10\% d'erreur relative indépendamment de la technologie et de la taille du circuit. NeuPow permet d'obtenir une productivité de conception élevée. Les temps de simulation obtenus sont significativement améliorés par rapport à ceux obtenus avec les outils de conception conventionnels.

**Title:** An Efficient Computer-Aided Design Methodology for FPGA&ASIC High-Level Power Estimation Based on Machine Learning

**Keywords:** Power Consumption, FPGA, ASIC, Machine Learning, Modeling, Estimation

**Abstract:** Nowadays, advanced digital systems are required to address complex functionalities in a very wide range of applications. Systems complexity imposes designers to respect different design constraints such as the performance, area, power consumption and the time-to-market. The best design choice is the one that respects all of these constraints. To select an efficient design, designers need to quickly assess the possible architectures. In this thesis, we focus on facilitating the evaluation of the power consumption for both signal processing and hardware design engineers, so that it is possible to maintain fast, accurate and flexible power estimation. We present NeuPow as a system-level FPGA/ASIC power estimation method based on machine learning. We exploit neural networks to aid the designers in exploring the dynamic power consumption of possible architectural solutions. NeuPow relies on propagating the signals throughout connected neural models to predict the power consumption of a composite system at high-level of abstractions. We also provide an upgraded version that is frequency aware estimation. To prove the effectiveness of the proposed methodology, assessments such as technology and scalability studies have been conducted on ASIC and FPGA. Results show very good estimation accuracy with less than 10\% of relative error independently from the technology and the design size. NeuPow maintains high design productivity, where the simulation time obtained is significantly improved compared to those obtained with conventional design tools.