Contrôleurs reconfigurables ultra-faible consommation pour les réseaux de capteurs sans fil

Vivek Tovinakere Dwarakanath

N° d'ordre : - ANNÉE : 2013

THÈSE / UNIVERSITÉ DE RENNES 1

sous le sceau de l'Université Européenne de Bretagne

pour la grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Traitement du Signal et Télécommunications

École Doctorale : MATISSE

présentée par

Vivek TOVINAKERE DWARAKANATH

preparée à l'unite recherche : IRISA - UMR 6074

Institut de Recherche en Informatique et Systèmes Aléatoires - CAIRN

École Nationale Supérieure des Sciences Appliquées et de Technologie

---

Contrôleurs reconfigurables ultra-faible consommation pour les nœuds de réseaux de capteurs sans fil

Ultra-Low Power Reconfigurable Architectures for Controllers in Wireless Sensor Network Nodes

Composition du jury :

Ian O'CONNOR
Professeur, INL
Ecole Centrale de Lyon / Examinateur

Jean-Philippe DIGUET
Directeur de Recherche, CNRS, LAB-STICC
Université de Bretagne Sud / Examinateur

Patrick GIRARD
Directeur de Recherche, CNRS, LIRMM
Université de Montpellier / Rapporteur

Marc BELLEVILLE
Directeur de Recherche, CEA, LETI
Minatec, Grenoble / Rapporteur

Olivier SENTIEYS
Directeur de Recherche, INRIA
Lannion / Directeur de thèse

Steven DERRIEN
Professeur, Université de Rennes 1
Rennes / Co-directeur de thèse

Résumé

Les réseaux de capteurs sans fil représentent une convergence de trois aspects des systèmes intégrés pour le traitement de l'information, à savoir la perception de l'environnement, le traitement de données et leur communication par radio. Un nœud du réseau de capteurs sans fil peut avoir besoin de traiter dans ses unités de calcul les signaux de plusieurs types de capteurs. Sur l'aspect des communications sans fil, un nœud peut avoir à effectuer différentes de tâches liées à la transmission de l'information et à adapter les protocoles de communication. Un nœud lui-même peut donc avoir à changer son rôle de façon dynamique et doit donc inclure des tâches complexes de contrôle pour la gestion de ses ressources. Tous ces facteurs font que la flexibilité est un sujet primordial dans la conception de nœuds de réseaux de capteurs sans fil, en plus bien sur de l'énergie consommée. Les architectures reconfigurables, comme les FPGA qui utilisent des blocs logiques configurables et des réseaux d'interconnexion programmables ont été proposés pour répondre aux besoins de flexibilité. Ils sont milleurs en termes de coûts non récurrents par rapport à des circuits intégrés spécialisés (ASIC). Toutefois, le matériel flexible de ces processeurs reconfigurables n'est pas une solution alternative à la grande efficacité énergétique des circuits ASIC - la contrainte primordiale pour les nœuds des réseaux de capteurs sans fil.

Dans cette thèse, des contrôleurs flexibles de consommation ultra-faible pour les nœuds réseaux de capteurs sans fil basés sur des micro-tâches reconfigurables sont étudiées. Une micro-tâche est principalement une unité de contrôle numérique incluant une machine d'états finis (FSM) et un chemin de données assemblant des unités arithmétiques et logiques, des mémoires et des périphériques d'entrées-sorties. Pour introduire de la flexibilité dans les micro-tâches, des machines d'états finis flexibles et évolutives, et des additionneurs reconfigurables à précision variable sont considérés au prix d'une légère augmentation de la surface du matériel. De plus, un fonctionnement à ultra-faible consommation est recherché grâce à l'utilisation des techniques de *power gating* pour réduire la puissance statique. Il est bien connu que, dans les circuits CMOS nanométriques, l'augmentation de la densité de puissance statique dépasse de loin l'impact de la surface de Silicium en raison de la très forte intégration.

Le *power gating* (PG) comme technique de faible consommation est considéré aux niveaux architecture et circuit pour les machines d'état finis et les éléments du chemin de données. Une étude approfondie des problèmes liés à la conception et à la modélisation des circuits utilisant la technique de *power gating* est réalisée en suivant une approche ascendante. Des modèles au niveau porte pour l'estimation des paramètres de conception des circuits PG sont tout d'abord dérivés. Ensuite, les additionneurs et machines d'états finis reconfigurables proposées pour les micro-tâches sont étudiés pour tirer partie de l'efficacité des possibilités des techniques PG. Dans les additionneurs, la reconfigurabilité est utilisée pour faire varier dynamiquement la précision de l'opération et des économies significative d'énergie sont obtenues en éteignant les blocs logiques inutilisés. Pour les machines d'états finis reconfigurables, des architectures avec différents degrés de flexibilité et de complexité sont proposées. La technique PG au niveau de la *lookup table* (LUT) est proposée pour atteindre des réductions de puissance statique significatives. Différentes structures transverses et de rétroaction des machines d'états finis sont exploitées pour réduire les interconnexions programmables. Enfin, les modèles proposés sont appliqués à analyser l'économie d'énergie (ou de puissance) réalisée dans les contrôleurs et les chemins de données en raison de l'utilisation conjointe de la reconfiguration et du PG. Les micro-tâches reconfigurables proposées sont positionnées par rapport à

des microcontrôleurs de faible puissance et des micro-tâches câblées en tenant compte de différents paramètres.

Abstract

Wireless sensor networks (WSNs) represent a convergence of three aspects of evolutionary integrated systems for information processing *viz.*, sensing, computation and communications. A node in the wireless sensor network may need to process signals from different types of sensors in their sensing and computational units. On the communications side, a node may have to perform different transceiver tasks to adapt wireless communication protocols. A node itself may have to change its roles dynamically. Hence the controller in a node is required to execute different control tasks to manage its resources. All these factors imply that flexibility is of key concern in the design of WSN nodes. Reconfigurable hardware such as FPGAs that use configurable logic blocks and programmable interconnection networks have been proposed to address the need for flexibility. They also offer efficiency in terms of non-recurring engineering (NRE) costs compared to ASIC-based designs. However flexible hardware of reconfigurable processors are not energy efficient unlike dedicated circuits - a key constraint for WSN nodes.

In this thesis, ultra-low power flexible controllers for WSN nodes based on reconfigurable microtasks are explored. A microtask is essentially a digital control unit in a node with a finite state machine (FSM) and a datapath consisting of arithmetic-logic unit, memories and input/output interfaces. To introduce flexibility in microtasks, reconfigurable FSMs and adders are considered at the expense of hardware area. Scalable architectures for reconfigurable FSMs along with variable precision adders in the datapath are proposed in this work for flexible controllers based on a microtask model. Further, it is sought to achieve ultra-low power operation using power gating. It is well known that in nanoscale CMOS circuits, the increase in static power density as a cost far exceeds the impact of area due to increased logic integration.

Power gating as a low power technique is considered at architecture and circuit levels for FSM and datapath elements of microtasks. It involves an extensive study of design issues in power gating and modeling of design parameters of a power-gated circuit. A bottom-up approach is taken: gate level models for estimation of key design parameters of power-gated circuits are derived first. Next, reconfigurable adders and FSMs proposed for microtasks are studied at gate level for power gating opportunities. In adders, reconfigurability is used for dynamically varying the precision of operation and hence examined for potential energy savings by power gating unused logic. For reconfigurable FSMs, scalable architectures with varying degrees of flexibility and complexity are presented. Power gating at the level of lookup table (LUT) logic is proposed to achieve aggressive leakage power and energy reduction. The feedback and feedforward structures of a FSM are exploited to reduce programmable interconnections. Finally the proposed models are applied to analyze energy (or power) savings in the logic clusters of FSMs and adders in datapath due to power gating. The position of reconfigurable microtasks in the design space of controllers relative to a low power microcontroller and hardwired microtasks is discussed using different metrics.

Contents

List of Figures

List of Tables

Chapitre 0

Résumé étendu

0.1 Nœuds d'un réseau de capteurs

Les réseaux de capteurs (WSN) sont une infrastructure composée d'un large nombre de systèmes embarqués, appelés des nœuds, capables de capter des signaux physiques, de calculer et de communiquer entre eux au sein d'un réseau à la topologie variable. Ce type de fonctionnalité d'un WSN aboutit à un grand nombre d'applications potentielles [2, 3] telles que le monitoring de paramètres environnementaux, la surveillance dans un contexte de sécurité, la surveillance de santé personnelle [4], la gestion de trafic de véhicules, la gestion d'énergie dans les bâtiments, etc. Typiquement, un nœud peut contrôler d'autres nœuds, capter de l'information, transmettre, relayer ou recevoir des informations. Par conséquent, un nœud doit pouvoir changer son rôle dynamiquement au cours du temps, ce qui implique que leur omniprésence requiert de la flexibilité, tout en satisfaisant des contraintes fortes d'énergie et de performance.

L'architecture typique d'un nœud de capteur, comme le montre la Fig. 1, est composée des capteurs et de leur interface, d'une unité de calcul et de contrôle et d'un émetteur/récepteur radio-fréquence. En plus de ces trois unités, une alimentation électrique incluant une unité de gestion de la puissance est requise. La technologie la plus utilisée pour la réalisation matérielle de ces nœuds est la technologie de circuit intégré CMOS *(complementary metal-oxide-semiconductor)*. Cette intégration d'un large nombre de transistors MOS apporte de nombreux challenges et compromis entre puissance consommée, performance, taille du circuit intégré et coût du produit final. Une étude critique des relations entre la fonctionnalité des nœuds de WSN, leurs contraintes énergétiques et leurs méthodes de conception, est nécessaire pour explorer correctement l'espace de conception des solutions matérielles efficaces en énergie.

Tandis que les réseaux de capteurs évoluent rapidement, leur spectre d'applications évolue encore plus vite. Les unités de calcul traitent des signaux issus de plus en plus de types de capteurs. Les unités de communications doivent s'adapter aux évolutions

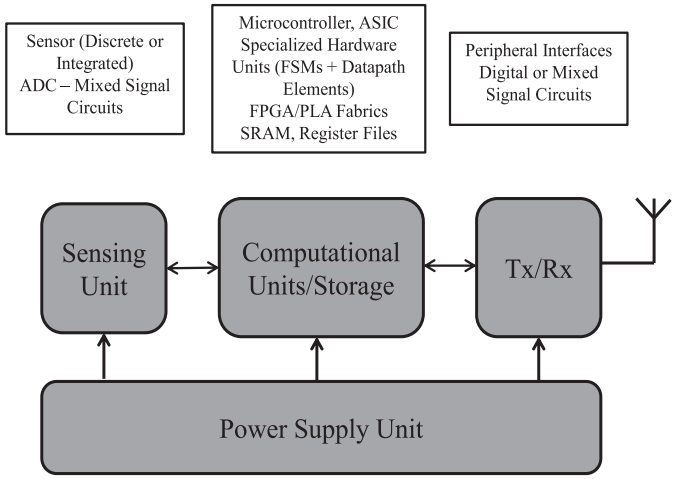


FIGURE 1: Un réseau de capteurs typique.

rapides des protocoles sans fil. Ceci implique donc de plus en plus de tâches diverses de contrôle et aussi que la flexibilité est en enjeu essentiel pour les unités de contrôle assurant le gestion des nœuds.

Un flot de conception complet assurant la synthèse depuis une spécification système de contrôleurs de nœuds d'un réseau de capteurs a été proposé dans [1]. Dans cette approche les parties calcul et contrôle d'un nœud de capteur sont constituées d'un ensemble de micro-tâches matérielles qui sont activées selon un principe événementiel, chacune étant dédiée à une tâche spécifique du système comme par exemple le relevé de paramètres, la couche MAC, le routage ou le traitement des données. Le flot prend en entrée une spécification de haut niveau du graphe flot de tâches associant le langage C et un langage spécifiquement conçu pour ce domaine. Il génère automatiquement la description architecturale de ces micro-tâches sous la forme de blocs matériels spécialisés, constitués d'une machine à états finis (FSM) et de chemin de données *(datapath)*, associés à une gestion système des tâches et des mémoires pour une exécution de l'ensemble du contrôle du nœud, comme le montre la Fig. 2. L'architecture est de plus optimisée en énergie, en particulier lors des périodes d'inactivité, grâce à l'utilisation des techniques de *power gating*. Même si le flot de conception permet un prototype rapide, le matériel généré est cependant spécifique au code synthétisé et la flexibilité est donc extrêmement limitée.

0.1.1 Espace de conception de contrôleurs flexibles pour les nœuds de capteurs

La fonction primaire d'un contrôleur de nœuds de capteurs est de gérer des ressources matérielles incluant des chemins de données de calcul, des horloges, des interruptions, des générateurs d'événements, des mémoires, des périphériques, des gestionnaires d'alimentation et des gestionnaires globaux. Il a été estimé que jusqu'à 25% de la consommation

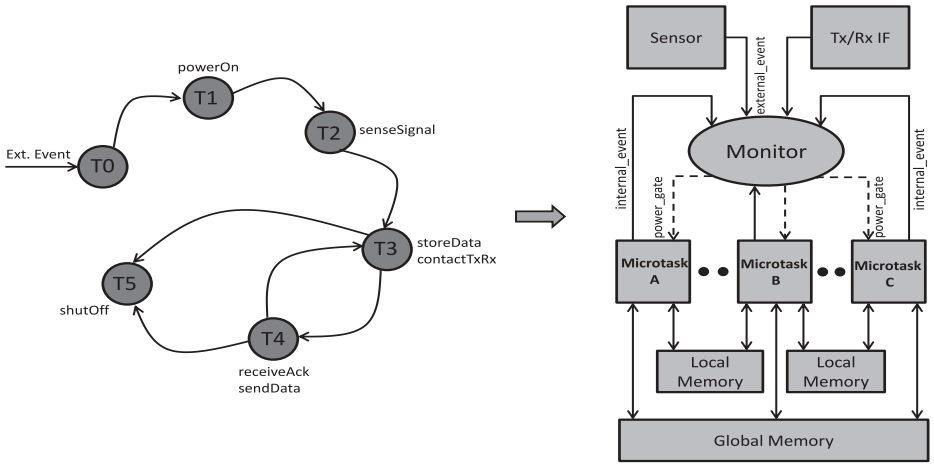


FIGURE 2: Un graphe de micro-tâches et la vue système de l'architecture générée par le flot de conception proposé dans [1].

globale d'un nœud est due au contrôle. Grâce à leur flexibilité, les micro-contrôleurs sont une cible privilégiée pour cette tâche, comme le montre à l'évidence leur utilisation dans de nombreuses plateformes [5, 6, 7]. Un large spectre de techniques de réduction de la consommation d'énergie allant de la coupure d'horloge *(clock gating)* à la conception sous le seuil en passant par la coupure d'alimentation *(power gating)*, [8, 9, 10] a été utilisé dans ce type de micro-contrôleurs. Cependant, ces cibles sont encore loin d'être des solutions optimales dans nombre d'applications, du fait que l'exécution séquentielle des différentes étapes du contrôle implique une forte consommation de la logique du processeur.

Dans les contrôleurs de WSN qui nécessitent de supporter de multiples graphes de tâches, les processeurs reconfigurables offrent à la fois use grande flexibilité et une réduction des coûts de prototype et de production par comparaison aux solutions ASIC. Une architecture reconfigurable communément utilisée, et nommée *Embedded FPGA* (eFPGA) dans ce travail, est une fabrique homogène contenant de nombreux blocs logiques configurables (CLB) associés à un réseau d'interconnexions programmable permettant de connecter les différents CLB entrée eux [11, 12]. Un CLB est constitué d'une table de correspondance *(lookup table)* à K entrées (K-LUT), de bits de configuration (CB) et d'une bascule que l'on peut ajouter pour mémoriser la sortie sur un cycle d'horloge. Une LUT est un ensemble de 2^K éléments mémoire sur un bit qui content la table de vérité d'une fonction décodée par les K variables d'entrées. Le CLB d'une LUT à 4 entrées et le bit de configuration utilisé dans un élément mémoire sont représentés à la Fig. 3. Une vue complète de la matrice du eFPGA et son élément de base sont quant à eux montrés Fig. 4. Une estimation de la consommation de puissance pour cinq FSM issues du benchmark SenseBench [13] implémentées dans une technologie CMOS 65nm est donnée dans le tableau 1. La puissance consommée est comparable à celle d'un micro-contrôleur faible

consommation [1] et, clairement, le matériel flexible des architectures reconfigurables de type FPGA n'est pas aussi efficace en énergie que les circuits dédiés, ce qui pose un problème pour les contraintes énergétiques requises par les applications des WSN.

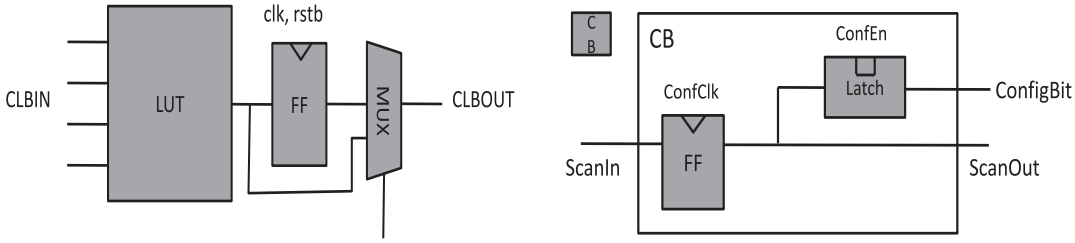


FIGURE 3: Structure d'un CLB et d'un CB dans le eFPGA.

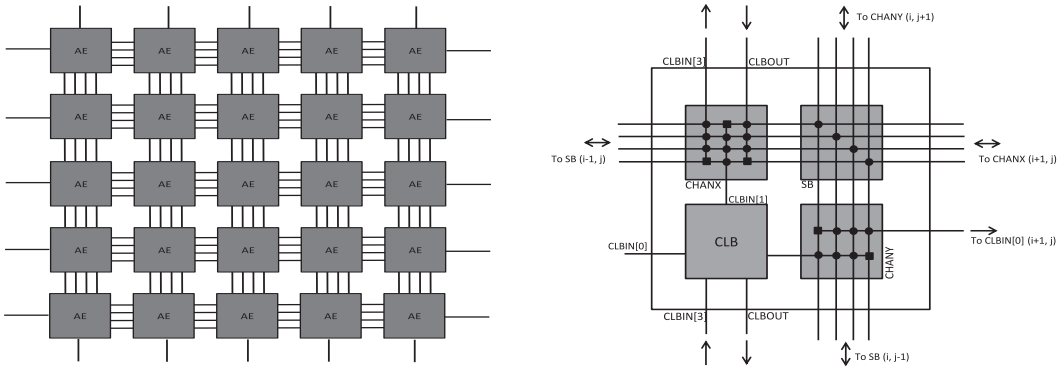


FIGURE 4: Vue système de la matrice du eFPGA et un élément de celle-ci contenant les canaux de routage et la matrice d'interconnexions programmables.

eFPGA	Total Power of FSM (mW)	
	f_{clk} = 20 MHz	f_{clk} = 100 MHz
abs	0.79	1.93
Crc8	1.39	3.32
receiveData	1.58	3.74
Crc16	2.70	6.77
firBasic	4.71	13.29

TABLE 1: Consommation de puissance (mW) pour cinq FSM issues de SenseBench synthétisées sur un eFPGA.

0.1.2 Contrôleurs reconfigurables à faible consommation

Dans cette thèse, des unités arithmétiques à précision variable pour les chemins de données et des architectures modulables pour des FSM reconfigurables à très fable consommation sont explorées comme cible matériel pour des micro-tâches flexibles. La coupure d'alimentation est utilisée comme technique de réduction d'énergie aux niveaux circuit et architecture pour les FSM et les chemins de données des micro-tâches telles que décrits

dans [1]. Ceci implique une étude extensive des problèmes liés à la conception d'un circuit utilisant la coupure d'alimentation. Des modèles au niveau porte pour l'estimation des paramètres clés des circuits à coupure d'alimentation sont tout d'abord dérivés. Ensuite, deux blocs matériels de micro-tâches, c'est-à-dire un additionneur et une machine d'états, sont optimisés en termes de consommation d'énergie. Dans les deux cas, la reconfigurabilité est considérée comme élément clé pour une conception matérielle flexible. Tandis que dans les additionneurs la reconfiguration est utilisée pour faire varier dynamiquement la précision, dans les FSM reconfigurables, des architectures modulaires avec plusieurs degrés de complexité et de flexibilité sont proposées. Finalement, les modèles sont appliqués pour analyser différents paramètres de conception dans la logique des FSM et des additionneurs utilisant le *power-gating*.

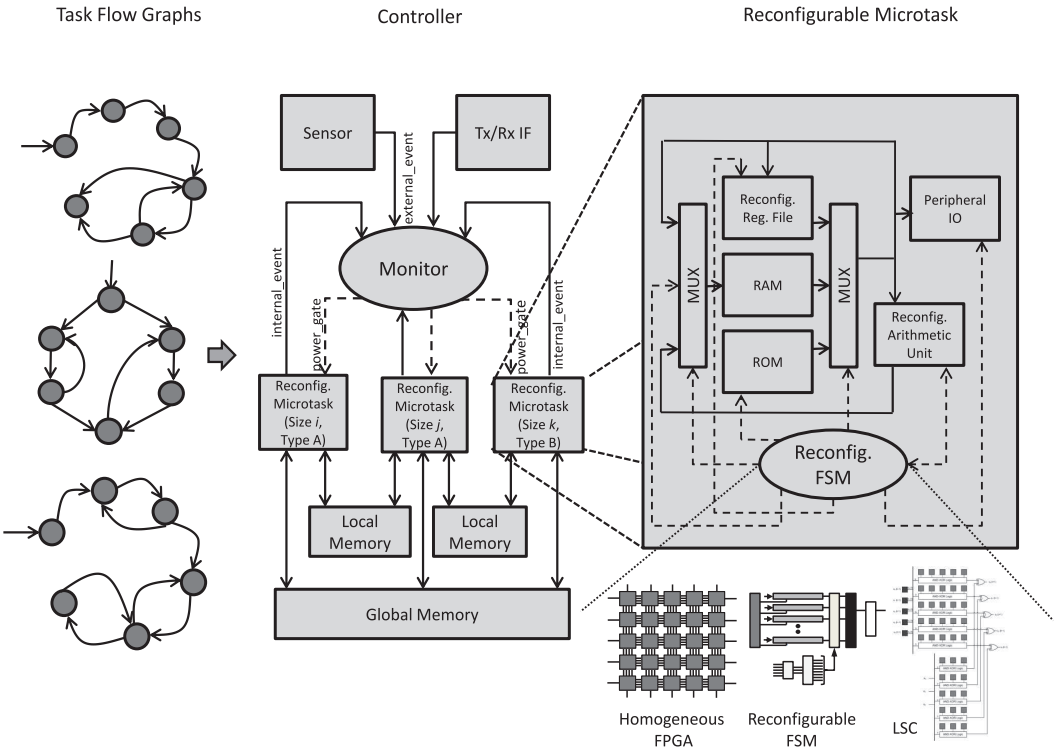


FIGURE 5: Structure d'un contrôleur flexible pour micro-tâche reconfigurable.

0.2 Modèles pour les circuits à coupure d'alimentation

0.2.1 Circuit et opération équivalents d'un circuit à coupure d'alimentation

Tandis que les transistors MOSFET voient leur dimension se réduire en dessous de 100nm, une augmentation exponentielle du courant de fuite est observée à cause de la réduction de la tension de seuil (V_{th}) opérée pour maintenir la commande grille à des valeurs correctes [14]. Une structure de *power-gating* coupe la tension d'alimentation des transistors

MOS d'une cellule logique de façon à ce que les courants de fuite soient réduits de façon significative pendant les phases de repos. Un exemple de circuit de coupure d'alimentation est donné à la Fig. 6. Il est constitué d'un transistor PMOS ayant une tension de seuil V_{th} haute, connecté entre le rail d'alimentation V_{dd} et le nœud de tension d'alimentation virtuelle V_{Vdd} du bloc logique. Un bloc logique est ici un ensemble de portes logiques dont la tension d'alimentation sera coupée par ce transistor de mise en veille, dont la grille est connectée à un signal de contrôle $SLEEP$ pour commander le bloc entre les états *on* et *off*. Un circuit à coupure d'alimentation opère donc dans trois modes : actif, veille et réveil dans un cycle typique de *power gating*, comme indiqué Fig. 7.

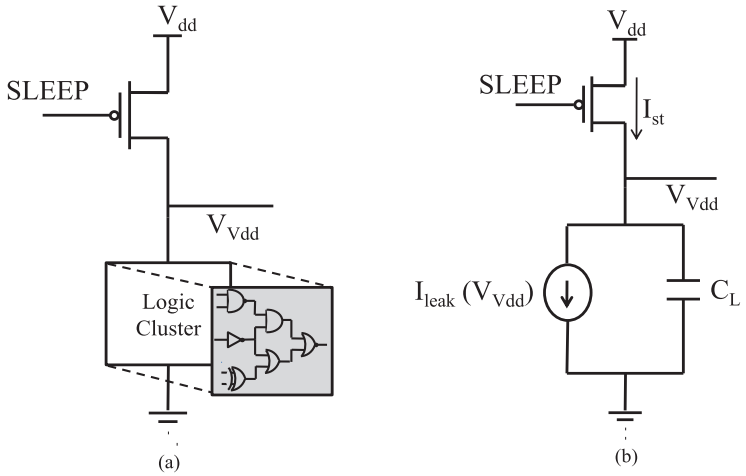


FIGURE 6: (a) Cluster logique à coupure d'alimentation. (b) Circuit équivalent.

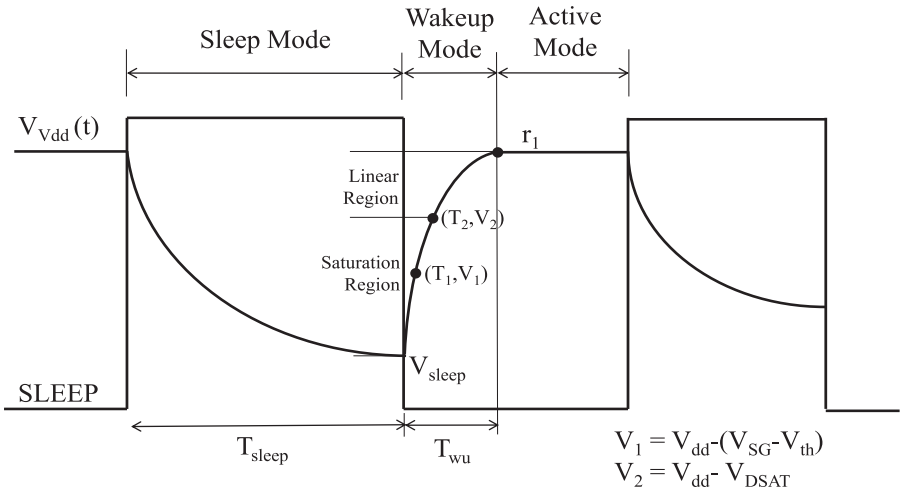


FIGURE 7: Un chronogramme temporel typique et les différents modes d'opérations dans un cycle typique de *power gating*.

0.2.2 Temps et énergie de réveil

Dans ce travail, nous avons proposé de dériver les profils des courants de fuite pour un circuit complet en utilisant une approche basée sur des approximations polynomiales. Soit un bloc logique à coupure d'alimentation composé de cellules logiques standards issues d'une bibliothèque, le courant statique total pour $n(S_i, j)$ occurrences de chaque cellule S_i est

$$I_{leak} = \sum_{i=0}^{P-1} \sum_{j=0}^{R_i-1} n(S_i, j) I_{leak}(S_i, j)$$

où $I_{leak}(S_i, j) = \sum_{k=0}^{N} b_k(S_i, j) V_{Vdd}^k$ est use approximation polynomiale obtenue à partir de simulations au niveau transistor plus précises, P et R_i sont le nombre de types de cellules et le nombre de combinaisons potentielles de la cellule S_i respectivement. La capacité totale du bloc logique est obtenue comme la somme des capacités de toutes les entrées des cellulose standards qui le constituent $C_L = \sum_{i=0}^{P-1} \sum_{j=0}^{R_i-1} n(S_i, j) C_{ij}$. La résolution des équations

$$\frac{dV_{Vdd}}{dt} = -\frac{1}{\tau} \sum_{i=0}^{N} c_i V_{Vdd}^i$$

et

$$\frac{dV_{Vdd}}{dt} = -\frac{1}{\tau}(V_{Vdd} - r_1)(V_{Vdd} - r_2)$$

avec $\tau = R_{lin} C_L$, $c_i = f_i(V_{dd}, R_{lin}, b_i, V_{th})$, ainsi que d'autres heuristiques pour le courant dans la région de saturation, aboutissent à une expression pour $V_{Vdd}(t)$ (avec un comportement comme montré sur la Fig. 8) à partir de laquelle le temps de réveil peut être déterminé comme $T_{wu} = T_{(V_{Vdd}=r_1)}$ en prenant l'hypothèse que $V_{Vdd} = V_{sleep}$ à $t = 0$. R_{lin} et r_1 représentent ici la résistance du transistor de mise en veille en région linéaire et l'état de V_{Vdd} au début du mode actif, respectivement. De plus, les gains en énergie dans le bloc logique avec *power gating* par rapport à celui sans coupure d'alimentation sont donnés par

$$E_s = V_{dd} I_{leak}(V_{dd}) T_{sleep} - \int_0^{T_{sleep}} V_{Vdd} I_{leak}(V_{Vdd}) dt.$$

Le surcoût en énergie dû au réveil peut être estimé en sommant les N_{wu} intervalles sur la courbe $I_{st}(t)$ de taille Δt in $[0, T_{wu}]$ selon

$$E_{wu} \approx V_{dd} \sum_{i=0}^{N_{wu}-1} I_{st_i} \Delta t$$

où $I_{st}(t)$ est le courant dans le transistor de mise en veille. Ainsi, plusieurs paramètres de conception sont obtenus depuis des caractérisations polynomiales simples du courant

statique dépendant de la tension d'alimentation dans des portes logiques. Des résultats expérimentaux montrent que les temps de réveil et l'énergie de réveil sont estimés avec une erreur moyenne de 16% et 13% respectivement par rapport à des simulations SPICE sur des circuits issus des benchmark ISCAS85.

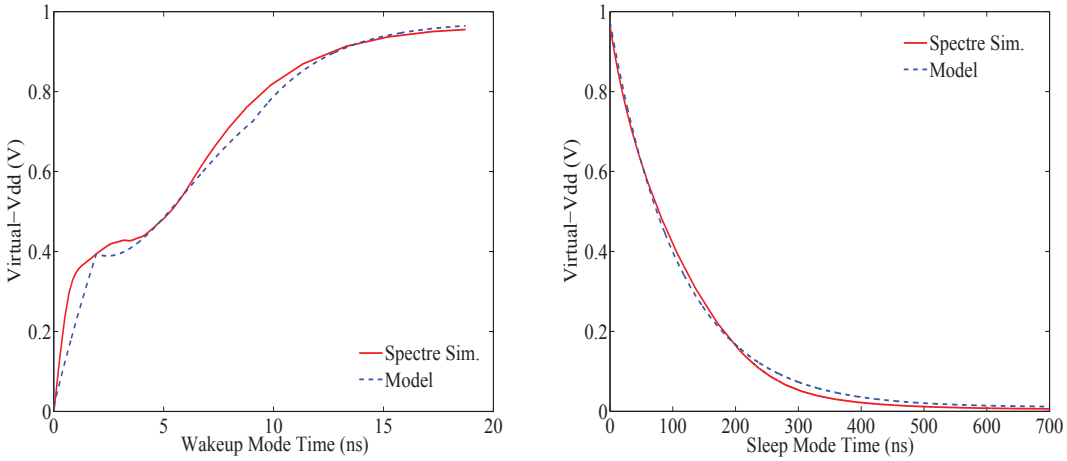


FIGURE 8: Virtual-Vdd en modes réveil et veille (W=1.2μm) pour le benchmark c7552.

0.3 Additionneur faible consommation à précision variable

Les circuits arithmétiques ayant des tailles d'entrée fixes sont des sources de gaspillage d'énergie lorsque des données de précision plus faibles doivent être calculées pendant de longues périodes de temps. Sachant que ces opérateurs arithmétiques sont des structures denses mais régulières pouvant être implémentées avec des tailles de mots et du parallélisme à des granularités diverses, ils pourraient être configurés pour des précisions plus faibles conjointement avec une suppression des courants de fuite inutiles dans les portes logiques inutilisées. Les additionneurs sont utilisés dans la logique d'adressage et de séquencement (boucles) des microprocesseurs tandis que les multiplieurs font partie intégrante des chemins de données de calcul. Dans ce travail nous nous sommes concentrés sur les additionneurs comme composant des chemins de données intégrés aux micro-tâches matérielles de [1]. Une approche générale est considérée pour partitionner la logique d'un additionneur d'une taille donnée en différents blocs logiques supportant des précisions intermédiaires, et en appliquant des coupures d'alimentation pour éteindre les blocs inutilisés en fonction de la précision considérée. La méthode est utilisée sur deux additionneurs à préfix parallèle de Brent-Kung (BK) [15] et de Kogge-Stones (KS) [16]. Deux additionneurs 32 bits pouvant être configurés comme des additionneurs 8 bits, 16 bits ou 32 bits, sont décrits. Une comparaison de leur puissance consommée avec celle de l'additionneur 32 bits non flexible montre des potentiels de réduction de la puissance

par un facteur allant de 8 à 13 pour des augmentations en surface de 15% et 9.2% respectivement.

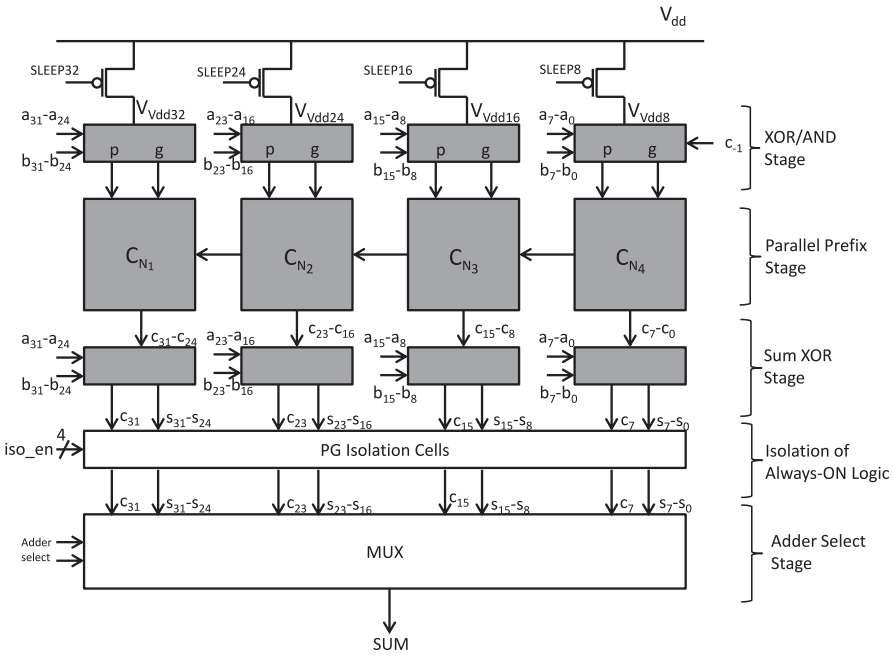


FIGURE 9: Additionneur reconfigurable à coupure d'alimentation.

Function	$I_{max,ac}$ (mA)		$P_{active,av}(\mu W)$		$P_{idle,av}(\mu W)$	
	BK	KS	BK	KS	BK	KS
addition sur 32 bits	4.92	4.61	222.3	240.7	55.5	62.9
addition sur 24 bits	4.22	3.86	185.2	200.6	55.8	62.8
addition sur 16 bis	2.92	2.63	142.6	152.8	55.4	62.3
addition sur 8 bits	1.80	1.59	102.4	104.8	55.1	61.9

TABLE 2: Consommation de puissance pour des additionneurs BK et KS sans coupure d'alimentation.

Adder	$I_{max,ac}$ (mA)		$P_{active,av}$ (μW)		$P_{idle,av}$ (μW)		$P_{sleep,av}$ (μW)	
	BK	KS	BK	KS	BK	KS	BK	KS
additionneur 32 bits	1.91	2.02	223.8	286.2	60.5	81.7	8.6	8.7
additionneur 24 bits	1.52	1.56	171.8	218.9	47.7	62.5	7.8	8.2
additionneur 16 bits	1.17	1.09	118.3	139.5	34.8	40.1	7.3	7.6
additionneur 8 bits	0.73	0.75	65.1	66.1	20.9	20.7	6.8	7.0

TABLE 3: Consommation de puissance pour des additionneurs reconfigurables BK et KS avec coupure d'alimentation.

0.4 Machines à états reconfigurable avec *power-gating*

La mise en œuvre de machines à états (Finite State Machine) consiste en deux blocs combinatoires notés **F** et **G** ainsi qu'un ensemble d'éléments de mémorisation (registres) **S** synchronisés sur un signal d'horloge. La valeur des n entrées de la machine à états à un instant t est représentée par le vecteur $\mathbf{x}(t) = [x_0(t), x_1(t), ..., x_{n-1}(t)]$, la valeur des m sorties de cette machine à états par $\mathbf{y}(t) = [y_0(t), y_1(t), ..., y_{m-1}(t)]$. Enfin, la valeur des N bits du registre d'états est noté $\mathbf{s}(t) = [s_0(t), s_1(t), ..., s_{N-1}(t)]$. Soit f_i une fonction booléenne dans **F** et g_j une fonction booléenne dans **G**.

L'ensemble de fonctions de transition $s_i(t+1)$ et de commande y_j représentées par $s_i(t+1) = f_i(\mathbf{x}(t), \mathbf{s}(t)), i = 0, 1, ..., N-1$ et $y_j(t) = g_j(\mathbf{s}(t)), j = 0, 1, ..., m-1$ décrivent une FSM de Moore [17] tandis que celles représentées pas $s_i(t+1) = f_i(\mathbf{x}(t), \mathbf{s}(t)), i = 0, 1, ..., N-1$ and $y_j(t) = g_j(\mathbf{x}(t), \mathbf{s}(t)), j = 0, 1, ..., m-1$ représentent des FSM de Mealy [18]. La réécriture de la fonction de transition en utilisant sa décomposition de Shannon nous permet d'obtenir $s_i(t+1) = \sum_{k=0}^{2^{(n+N-K)}-1} m_k f_i(n(m_k), ..., s_{N-1})_k$ avec K correspondant au nombre de variables dont dépend $f_i(.)_k$ après cette décomposition. Le *minterm* généré par les $n+N-K$ premières variables d'entrées de la séquence $x_i, ..., s_i(t+1)$ s'écrit m_k, par exemple $m_1 = x'_{n+N-K-1} x'_{n+N-K-2} ... x_0$. Le vecteur motif binaire correspondant au *minterm* m_k est représenté par $n(m_k)$, par exemple on aura $n(m_1) = 000...01$. Pour un f_i donné, la fonction peut-être réalisée comme un assemblage de porte logiques, cependant si l'on souhaite permettre la reconfigurabilité, il est nécessaire de réaliser $f_i(.)_k$ comme une LUT (Look-Up-Table ou mémoire tabulée) qui pourra être reconfigurée en fonction de la machine à état à implanter.

0.4.1 Utilisation de la technique de coupure d'alimentation

Les possibilités de *power gating* sont utilisées comme illustré dans la figure Fig. 10, qui est dérivée de la décomposition de Shannon d'une fonction de transition à un niveau de granularité de type LUT combiné à un ensemble de porte AND.

Le réseau de transistors de coupure nécessite autant de signaux de contrôle que de LUT. L'opération de *power gating* peut-être expliquée comme suit. Quand la FSM reconfigurable est configurée en une FSM spécifique, son fonctionnement dépend d'un sous ensemble de ses entrées qui varie en général assez peu dans le temps, de ce fait, suivant les valeurs de ces entrées, tous les *minterms* s'évaluent à 0. Il est également possible que certaines entrées n'interviennent pas dans l'équation logique définissant certains bits du registre d'état, là encore les *minterm* correspondant s'évaluent à 0.

Il donc possible d'exploiter ces propriétés afin de couper l'alimentation de certaines LUT même lorsque la FSM est active, tant que le *minterm* qui leur est associé ne change pas de valeur (sa valeur dépendant elle même des entrées de la FSM). Ces sorties de

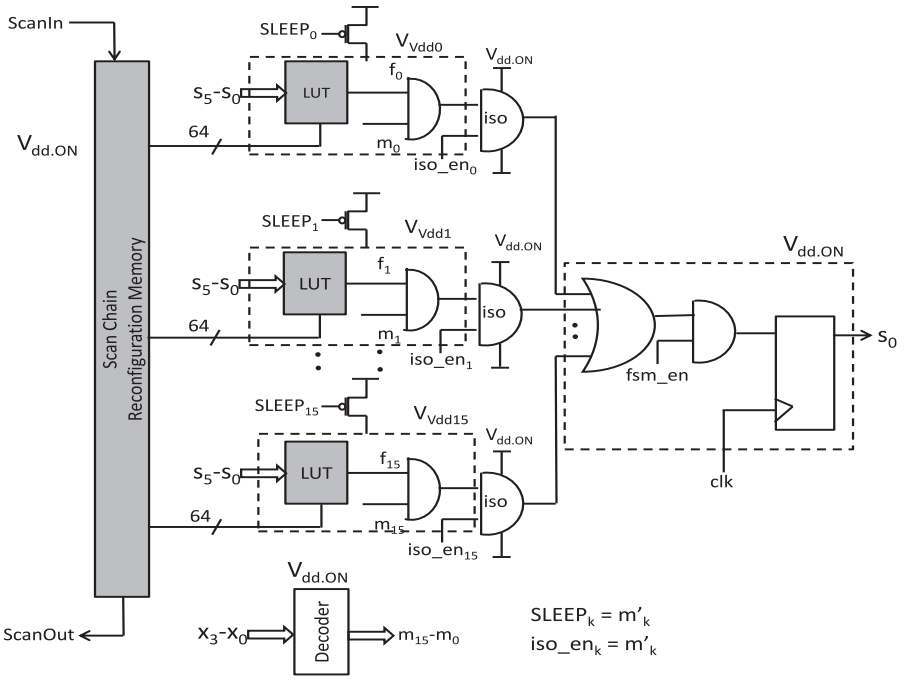


FIGURE 10: Modèle architectural de FSM reconfigurable, tel qu'il a été utilisé pour la validation expérimentale.

décodage peuvent fonctionner comme des signaux de contrôle qui serviront à couper l'alimentation de ces LUT évitant ainsi le recours à un contrôleur externe. Cette technique de *power gating* induit un coût plus important en termes de surface du fait de l'utilisation de cellules d'isolation [19] mais aussi en termes de réactivité (délais et dissipation énergétique causés par les phases de réveil [20]. Il est important d'indiquer que des technique similaires (opérant à des niveaux de granularité divers) peuvent être utilisées pour les fonctions de sortie.

Pour des architectures basées sur une granularité de *power gating* au niveau LUT, telles que proposées plus haut, on peut remarquer qu'à un instant donné, il n'y a qu'une seule LUT par bit d'état (ou de sortie) en mode actif, les autres pouvant être mises en veille. En d'autres termes, la consommation électrique à un instant donné ne dépend que de ces $N + m$ LUT, ceci indépendamment du coût (en nombre de LUT) de la FSM qui varie (dans le cas conservatif) comme une fonction exponentielle du nombre d'entrées, d'états et de sorties. Il faut néanmoins également comptabiliser les sources de dissipation énergétiques liées aux bits de configuration et aux cellules d'isolation.

Il faut également remarquer que pour chaque changement de l'entrée du décodeur, le *minterm* actif change également, et par conséquent, tous les *minterms* actifs peuvent être désactivés (par *power gating*). Par conséquent, un changement des valeurs en entrée induit des surcoûts en termes de temps de réaction et d'énergie causés par la phase de réveil correspondant à l'activation d'un *minterm*. Des gains en dissipation énergétique seront toutefois possibles lorsque la fréquence de fonctionnement de la FSM est peu élevée. De

plus, pour les FSM dont le nombre d'états et de sorties est inférieur au seuil maximum de l'architecture proposée, les LUT inutilisées peuvent également être désactivées par *power gating*.

0.4.2 Estimation de puissance

Une estimation de la consommation électrique pour un ensemble de machines à états utilisant la technique proposée est fournie dans le tableau 4. Les résultats ont été obtenus à partir de simulations SPICE. Une comparaison avec les estimations de puissance et d'énergie pour des FSM implantées sur un eFPGA indiquent une amélioration de la consommation électrique pour les FSM de taille importante en faveur de notre approche.

	Puissance dissipée (mW)		Energie par opération (pJ)	
	f_{clk} = 20 MHz	f_{clk} = 100 MHz	f_{clk} = 20 MHz	f_{clk} = 100 MHz
abs	0.35	0.75	17.76	7.54 (7.92)[a]
Crc8	0.51	1.13	25.46	11.30 (11.45)
receiveData	0.64	1.46	32.23	14.56 (14.54)
Crc16	0.59	1.32	29.38	13.21 (13.22)
firBasic	0.63	1.41	31.31	14.14 (14.10)

[a]Estimation extrapolée à partir de simulations SPICE

TABLE 4: Puissance et énergie moyennes par opération pour des FSM implantées sur une architecture à base de *power gating*.

0.5 Contributions

Les contributions de ce travail sont résumées ci-dessous.

1. Nous avons proposé un modèle semi-empirique pour estimer le temps de réveil pour des clusters logiques exploitant la coupure d'alimentation *power gating*. Étant donnée une description de circuit au niveau porte et une caractérisation compacte de son organisation en termes de portes logiques et de transistors de coupure, notre approche permet d'estimer très rapidement le temps de réveil de chaque bloc avec un taux d'erreur avoisinant les 16%. Elle permet donc d'obtenir des estimations précises tout en évitant de recourir à des simulations au niveau transistor, bien trop longues pour pouvoir être utilisées.

2. La puissance statique d'un bloc logique quelconque peut-être estimée avec une précision de 3% en utilisant un modèle analytique par approximation polynomiale du courant statique au niveau porte. Par ailleurs, d'autres paramètres clés des circuits exploitant le *power gating*, à savoir la valeur de la tension d'alimentation en régime permanent en mode actif et veille, le coût en énergie des phases de réveil ainsi que les gains globaux s'obtiennent directement à partir de ce modèle ou à

l'aide de simples extensions. Ces modèles, lorsque utilisés ensemble, peuvent servir en tant qu'outils d'estimation rapide qui pourront servir à guider le reste du flot de conception.

3. Nous proposons une approche de partitionnement appliquée à l'exploitation du *power gating* pour des circuits arithmétiques denses et réguliers (p. ex. : additionneurs ou multiplieurs). Notre approche permet leur utilisation comme des opérateurs à précision variable. Nos résultats indiquent que des gains en consommation allant jusqu'à 30% peuvent être obtenus. Nous avons également utilisé la possibilité de déterminer un seuil de précision au dessus duquel il est possible de réduire la puissance dynamique. Celui ci peut ensuite aider à la dérivation d'un ordonnancement du signal de coupure d'alimentation en vue d'obtenir une consommation minimale.

4. Nous avons proposé un modèle extensible d'architecture reconfigurable pour la mise en œuvre de FSM. Ce modèle se base sur la décomposition de Shannon des fonctions de transition et de décodage. Nous avons identifié des possibilités de réduire significativement la puissance statique dissipée (en mode actif) grâce a l'utilisation du *power gating* directement au niveau de la couche de reconfiguration. Ce modèle permet d'obtenir un niveau de consommation électrique intermédiaire entre une approche complètement programmable à base de micro-contrôleur et une approche complètement dédiée à base de micro-tâches matérielles spécialisées. Notre modèle d'architecture est donc une alternative intéressante combinant flexibilité et efficacité énergétique.

0.6 Conclusion

Dans ce travail, nous avons proposé l'utilisation d'architectures reconfigurables exploitant la technique de *power gating*, afin d'offrir une approche flexible à la mise en œuvre de nœuds de réseaux de capteurs basés sur le principe de micro-tâche matérielle. Nous avons exploité les propriétés structurelles des automates à mettre en œuvre afin de réduire le coût en interconnexion et la logique de sélection à un niveau minimal tout en garantissant un bon passage à l'échelle pour des machines à états plus complexes. Nous avons également proposé des modèles pour l'estimation rapide des surcoût liés aux temps de réveil ainsi que pour d'autres paramètres importants pour une conception au niveau porte logique exploitant les coupures d'alimentation. Nous avons ainsi montré que le délai de réveil est un élément clé qui contribue grandement dans le bilan énergétique global (en mode actif) des FSM reconfigurables à base de *power gating*. Enfin, nous discutons l'efficacité énergétique de notre approche en la mettant en regard du surcoût en surface induit par l'utilisation d'une forme de reconfiguration.

0.6.1 Efficacité énergétique

Une bonne métrique permettant d'évaluer l'efficacité énergétique des différentes réalisations est l'énergie moyenne dissipée par instruction. Le tableau 5 évalue cette métrique pour différentes tâches et stratégies de réalisation matérielle utilisées dans ce travail. Ainsi, l'énergie équivalente par instruction pour une micro-tâche E_{eff} s'obtient par

$$E_{eff} = \frac{E_{task}}{N_{inst}} \tag{1}$$

où l'énergie par tâche E_{task} est obtenue selon

$$E_{task} = N_{states} E_{op,MT}. \tag{2}$$

et où N_{states} est le nombre d'opérations (ou transitions d'états de la FSM) nécessaires à l'exécution de la tâche et $E_{op,MT}$ correspond à l'énergie moyenne par opération d'un micro-tâche.

Les résultats du tableau indiquent que l'efficacité énergétique de l'approche à base de micro-tâches reconfigurables se situe entre celle d'une approche à base de micro-contrôleur programmable très faible consommation et celle à base de micro-tâches spécialisées, ce qui était l'objectif à l'origine de ces travaux.

Micro-tâche	Energie équivalente par instruction (pJ/Inst.)					
	openMSP430 [1][a]		Micro-tâche Reconfigurable[b]		Micro-tâche Câblée 16 bits [1][c]	
	N_{inst}	E_{eff}	N_{states}	E_{eff}	N_{states}	E_{eff}
Crc8	30	163	71	31.60	71	8.1
receiveData	66	230	332	83.53	332	15.7
Crc16	27	170	73	41.27	73	9.3
firBasic	58	179	168	46.90	168	26.1

[a] 130nm, @16MHz
[b] 65nm, multiple V_{th} cells, @100MHz, 16×16 register file, no SRAM
[c] 65nm, std.-V_{th} cells

TABLE 5: Coût en énergie par instruction pour trois réalisation de contrôleurs.

0.6.2 Le coût de la flexibilité

Un revers important de la flexibilité est le surcoût en surface comparé à une approche à base de micro-tâches spécialisées. En se basant sur les résultats de Pasha et al. [1], il peut être montré que le surcoût en surface d'une micro-tâche reconfigurable est jusqu'à 19 fois plus élevé que sa version spécialisée, comme indiqué Table 6. Ce surcoût en surface

doit-être envisagé sous deux angles différents. Dans ce travail, les architectures proposées garantissent que toute FSM satisfaisant certaines contraintes (en terme de nombre d'états, de commandes, etc.) pourra être implantée sur une micro-tache reconfigurable. Ce n'est pas le cas d'approches basées sur un eFPGA ou sur micro-tâches spécialisées.

Microtask	Spécialisées Microtask [1](μm^2)	Reconfigurable Microtask (μm^2)	eFPGA (μm^2) (217 CLBs)	openMSP-430(μm^2)
Crc8	3097	140522.2	1076871	22141
receiveData	2858			
Crc16	3102			
firBasic	7164			

TABLE 6: Comparaison des coûts en surface des FSM spécialisées par rapport aux FSM reconfigurables (pour une chemin de données sur 16 bits).

De plus, le nombre de micro-tâches dans une plateforme pour nœud de réseau de capteurs peut varier et aller jusqu'à 40 ou 50. La coût en surface correspondant à une telle implémentation est donc significatif, et est susceptible de dépasser celui d'une approche à base de micro-tâches reconfigurables. En principe, une unique micro-tâche reconfigurable peut-être utilisée en lieu et place d'un ensemble de micro-tâches spécialisées en utilisant une forme de multiplexage temporel. Le contrôleur du nœud nécessite encore dans ce cas un moniteur système pour assurer la reconfiguration et le contrôle global de la plateforme.

Chapter 1

Introduction

Convergence of computing and communications has led to development of new portable, handheld, battery-powered products for multimedia and wireless applications. A testimony to this fact is the advent of smartphones which, in association with advanced wireless communication networks, are able to deliver comprehensive user experience of nearly anywhere, anytime telephony and multimedia applications. The onboard software driving these embedded systems process multimedia content requiring intensive computations. Other consumer electronic products based on such a convergence include tablets, portable computers and wireless peripheral devices.

Another class of applications that has gained importance are broadly based on what are referred to as Wireless Sensor Networks (WSNs). The fundamental unit of a WSN is called a node. Typically, but not necessarily, nodes have an additional function of sensing along with computing and communications [21]. Such an integration of functions leads to several applications [2, 3] - weather monitoring, surveillance for security, personal health care, vehicular traffic management, energy efficient buildings, etc. An anticipated evolution of mobile devices into WSN nodes would merge the two sets of applications into one (e.g., 'Cloud Health Care' [4]). A characteristic feature of WSNs is the possibility of using a large number of nodes in a number of untethered ways. For example, a node can be a controller of other nodes, a communicating agent with sensing, a transmitter, repeater, or a receiver. They may be configured to operate in different network topologies. Further, nodes may change their behaviour dynamically over time. The ubiquity of nodes requires that they offer flexibility while also satisfying power and performance constraints.

Wireless Sensor Network nodes have been realized on various technological platforms. One of the most common technologies used in WSN nodes has been the complementary metal-oxide-semiconductor (CMOS) integrated circuit (IC) technology as is also the case with a large number of electronic systems in the last fifty years. A key driver behind the success of CMOS circuits has been the continuous scaling of devices to nanoscale dimensions and consequent high density of device integration. The predictability of

scaling has been captured by Moore's 'law' [22] which states that the integration density of MOS transistors doubles approximately every two years. The International Technology Roadmap for Semiconductors (ITRS) [23] had indicated downscaling of physical gate length of MOS transistors from 65nm in the year 2002 to 22nm in 2012. The technological feasibility of integrating a very large number of MOS devices has brought about challenges in identifying trade-offs in power consumption, performance, sizes of integrated circuits and cost of final products. A critical study of interrelationships between functional convergence in WSN nodes, energy requirements and design automation is necessary to explore the design space for energy efficient solutions.

In Section 1.1, a broad overview of low power design is given. The sources of power constraints for integrated circuits are stated. The different components of total energy consumption in an arbitrary digital logic circuit are described. The section ends with an enumeration of various low power techniques developed and the support provided by electronic design automation (EDA) tools for synthesis and analysis of low power integrated circuits. Section 1.2 describes briefly the architecture and constraints of a WSN node. Some examples of different platforms on which the nodes have been realized are also given. The context of this work explained in Section 1.3 focuses on automated generation of energy efficient architectures for controllers in WSN nodes as it forms the starting point for the current work. Possible solutions in the design space are briefly discussed while also reviewing previous work. The contributions of the thesis are outlined in Section 1.4. Finally, the chapter concludes with organization of the thesis in Section 1.5.

1.1 Overview of Low Power Design

1.1.1 Sources of Power and Energy Constraints

Power and energy consumption[1] in CMOS integrated circuits have received significant attention for three important reasons - thermal management, battery lifetimes and autonomy of self-powered integrated systems. In microprocessors, as performance increased, efforts were made to reduce power consumption to lower the cost of heat sinks and packaging for thermal management [24]. Temperature variations due to power dissipation in microprocessors, Application Specific Integrated Circuits (ASICs), System-on-Chips (SoCs) and Field Programmable Gate Arrays (FPGAs), cause significant on-chip parameter variations that result in unreliable operation [25] and is therefore another reason for power dissipation control.

The evolution of ASICs into SoCs due to increase in integration density has reduced cost per unit function but has increased standby power dissipation considerably. FPGAs

[1] In this thesis, the terms energy and power are used interchangeably as long as it does not create any confusion. In contexts where the difference is significant, it will be explained to clarify the usage of a particular term.

on the other hand offer flexibility at low non-recurring engineering (NRE) costs. However the additional hardware in relation to its ASIC counterpart is a source of significant energy consumption in both active and standby modes of operation. Such SoCs and FPGAs when used in battery powered devices impose severe strain on battery lifetimes, as for a given form factor, the battery can supply only a finite amount of energy specified by its capacity.

The third reason for power constraints is due to autonomous nature of WSN nodes for certain applications. Some of the nodes may operate in such conditions that replacement of batteries is impossible. Hence nodes must depend on ambient energy sources to harvest power. The harvested power is significantly less than that provided by batteries thus imposing a constraint on the power consumed by the circuit at a given point of time. For example, weather monitoring nodes distributed in widely separated geographical areas require power consumption of less than 100μW to retain autonomy.

1.1.2 Energy Consumption in CMOS Circuits

Given an arbitrary digital system, the total energy consumed over a period T can be determined as follows. The total power consumption P_{total} in a CMOS circuit is given by

$$P_{total} = P_{sc} + P_{sw} + P_{static} \tag{1.1}$$

where $(P_{dyn} =)P_{sc} + P_{sw}$ is the dynamic component of total power and P_{static} is the static power due to leakage currents. P_{sc} is the power due to short circuit current through transistors and P_{sw} is the switching power due to charging and discharging of the capacitive load of each gate. Further, with supply voltage V_{dd}, short circuit current I_{sc}, load capacitance C_{load}, clock frequency f, transition density α (dependent on input patterns) and leakage current I_{leak}, Eq. (1.1) can be written in the most general form as [26]

$$P_{total} = V_{dd}I_{sc} + \alpha f V_{dd}^2 C_{load} + V_{dd}I_{leak}. \tag{1.2}$$

Let T_a denote the time for which the circuit is in active state and T_s the time for which it remains in standby state. Then the total energy consumption E_{total} of a circuit over a period of time $T = T_a + T_s$ is given by

$$E_{total} = [V_{dd}I_{sc,av} + \alpha f V_{dd}^2 C_{load} + V_{dd}I_{leak}]T_a + V_{dd}I_{leak}T_s \tag{1.3}$$

where $I_{sc,av} = \sum_i \frac{I_{sc_i}T_{sc_i}}{T_a}$ for each switching event i of duration T_{sc_i} is used for notational simplicity.

1.1.3 Low Power Techniques

A number of techniques have been used for reducing each component of power in Eq. (1.2). Dynamic power consumption due to charging and discharging of capacitive loads of logic gates can be reduced by shutting off clock (clock gating [27]). Architectural transformations like pipelining and parallelism can potentially reduce dynamic power [28]. Parts of integrated circuits with lower performance requirements can be operated at lesser supply voltages. Dynamic voltage and frequency scaling have also been used for power reduction in advanced ICs at circuit level for dynamic trade-offs between power and performance. Short circuit currents due to switching of gate outputs are reduced by balancing input and output transition times using logical effort techniques [24, 29].

A technique being actively studied for low power circuits is their operation under subthreshold or near-threshold supply voltages [30]. While dynamic power, being proportional to square of the supply voltage, is reduced drastically due to voltage scaling, MOS devices in such circuits have a low I_{on}/I_{off} ratio, degraded performance and lower noise margins due to high relative variations in device parameters. Current research is directed towards increasing the reliability of subthreshold devices and modeling for design automation [31, 32]. A detailed treatment of subthreshold design for ultra low-power systems is given in [33].

In sub-100nm CMOS technologies, the focus has shifted to problems due to large leakage power. As an example, the static power in 45nm technology is about 6 times more than that in 90nm technology. While leakage currents exist in transistor stacks of CMOS circuits at all times due to bias voltages, they are primarily responsible for energy consumption in standby states when no switching activity exists. This is particularly true for circuits that remain in standby states for significantly longer times than in active states as in nodes for some WSN applications and have a direct impact on battery lifetimes. Power gating has emerged as a promising technique for suppressing leakage currents and has found applications in a number of industrial IC designs [34]. In this technique bias voltages are cut off by a series transistor so that bias-dependent leakage current becomes negligibly small in standby state. In active state, the series transistor operates in linear region and the virtual supply (or ground) node provides necessary bias voltages for the logic gates to operate at their normal drive strengths. Another technique used to reduce subthreshold leakage current is to use multi-threshold voltage (V_{th}) CMOS circuits [35] due to the fact that high-V_{th} MOS devices have lesser leakage current than their low-V_{th} counterparts.

Recently, inexact circuits that use probabilistic techniques for pruning logic in dense arithmetic circuits to trade accuracy of outputs for energy savings have been proposed in [36]. Algorithmic modeling of digital systems with fixed-point realizations of computational units with energy constraints is an area of active research [37].

1.1.4 Design Automation for Low Power

Power closure is a relatively new term used in the design of VLSI circuits meeting specified constraints of power consumption before design sign-off. A design flow that supports power closure depends extensively on automated deployment of low power techniques for synthesis as also on power estimation methods for analysis. Commercial EDA tools have progressively incorporated these techniques into their tool suites. Logic synthesis tools [38] provide support for clock gating and are able to use multiple threshold voltage cell libraries for optimizing delay-leakage power trade-off. While the Unified Power Format (UPF) [39] has been framed for specification of multiple supply voltage and power-gated domains for verification, Common Power Format (CPF) [40] has been used for physical synthesis of such power domains with specified power intents. Power analysis tools [41, 42] are able to provide estimates of dynamic and static power along with thermal variation profiles at different process, voltage and temperature (PVT) corners in presence of noise and crosstalk. Further, state-of-the-art physical design tools provide capabilities to verify chip-wide IR drop and analyze power-rail electromigration for reliability in complex multi-million gate designs [43]. Despite such developments in the EDA domain, low power design remains a custom or semi-custom approach with an ensemble of *ad hoc* techniques.

1.2 WSN Nodes: Architecture and Realizations

1.2.1 Structure of WSN Nodes

The architecture of a typical WSN node is shown in Fig. 1.1. It consists of a sensing unit, computational and storage unit and a communication unit for transceiver (Tx/Rx) frontend. Associated with the three units is a power supply unit that houses power management circuits, batteries or energy scavenging units. The sensing unit interfaces with sensors and has associated signal conditioning circuits. The communication unit interfaces to a Tx/Rx antenna. The choice of technologies for implementation of nodes is usually determined by design constraints imposed by the application. Important constraints include energy consumption and form factor. Data processing capabilities and error-free communications are two other constraints nodes have to deal with and they further impact the first two factors. Sensing and communication units include mixed signal circuits whereas computational units are typically digital blocks. The units could be physically independent as is the case when the node is designed with off-the-shelf components or they could be integrated on to a SoC.

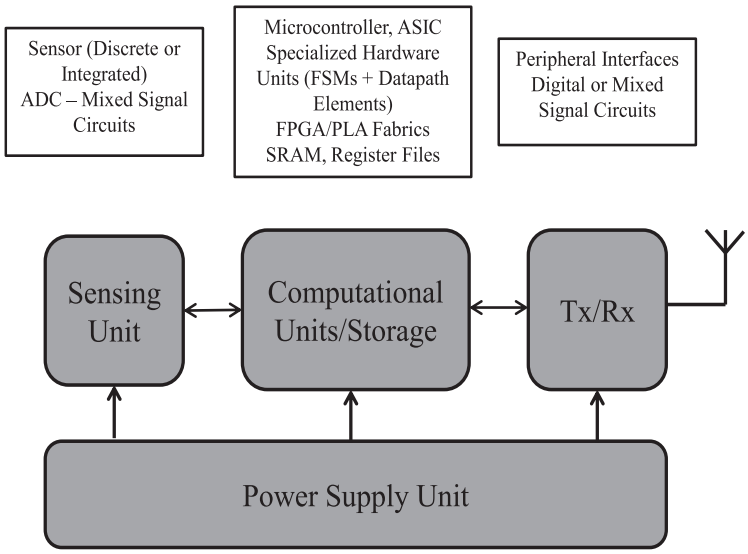


FIGURE 1.1: A typical wireless sensor network node.

1.2.2 WSN Node Realizations

Wireless sensor network nodes have been realized on different platforms. A number of industrial solutions exist for several applications. Being an active area of research, several experimental platforms have also been developed as part of broader study of node and network design. A few examples that demonstrate the breadth of WSN node implementations with their design methodologies are given next. In [44], a WSN node consisting of a MEMS motion sensor and CMOS wireless circuit for human physiological activity monitoring has been described. It represents an example of an application specific node designed with heterogenous technological processes. A full-custom or a mixed-signal IP integration flow is usually used for such a design. At the other end of the spectrum are nodes with independent sensing, computational and communication units. Such an architecture allows for flexibility in node functions and can be targeted towards a wider range of applications. A number of nodes have been based on low power microcontrollers (e.g., MSP430 of Texas Instruments, ATmega128L of Atmel) as computational units. Examples include BTNodes [7], Telos [5] and PowWow [6]. Since a microcontroller is a general system, a simple software compilation approach forms the basis for application mapping. Application Specific Instruction Processors (ASIPs) with dedicated hardware for specialized instructions have also been used to supplement the computational power of a processor. The Xtensa class of processors of Tensilica belong to this category [45]. Development tools provide direct programming support with customized instructions for application mapping. High throughput computational units are mostly designed as dedicated ASICs using standard or semi-custom VLSI design flows involving low power techniques. A logical extension of ASICs in the family of computational units is with reconfigurable circuits. They are discussed in the next section.

1.3 Context of the Work

While WSNs are evolving fast, the scope of their applications is also expanding tremendously. On the sensing side, signals from different types of transducers are required to be processed by signal conditioning circuits and computational units at varying signal-to-noise ratios (SNRs). Similarly on the communications side, wireless protocols are under constant evolution so that they require different levels of SNR, data throughput and error tolerance to be supported. Further, a part of the node needs to perform housekeeping functions thus incurring some energy. A node itself may have to change its roles dynamically over time. All these factors imply that flexibility is of key concern in the design of WSN nodes.

A complete design flow involving a system-level synthesis of ultra low-power WSN node controllers was proposed in [1]. The design flow takes as inputs a high level description of the WSN node controller represented by a task flow graph (e.g., in C or a domain specific language) in terms of smaller, independent control tasks and generates their hardware description in form of specialized microtasks [46]. The hardwired microtasks, which are a combination of finite state machines (FSMs) and datapath units consisting of arithmetic-logic units (ALU), storage devices and peripheral IO interfaces are finally integrated along with a system monitor and global memories to form a complete hardware mapping of the controller as shown in Fig. 1.2. A key contribution of the work in [1] has been rapid generation of hardware description of specialized controllers for WSN applications. It was proposed that the architecture could be exploited for power gating. While the design flow supports rapid prototyping, the final hardware thus generated still lacks flexibility.

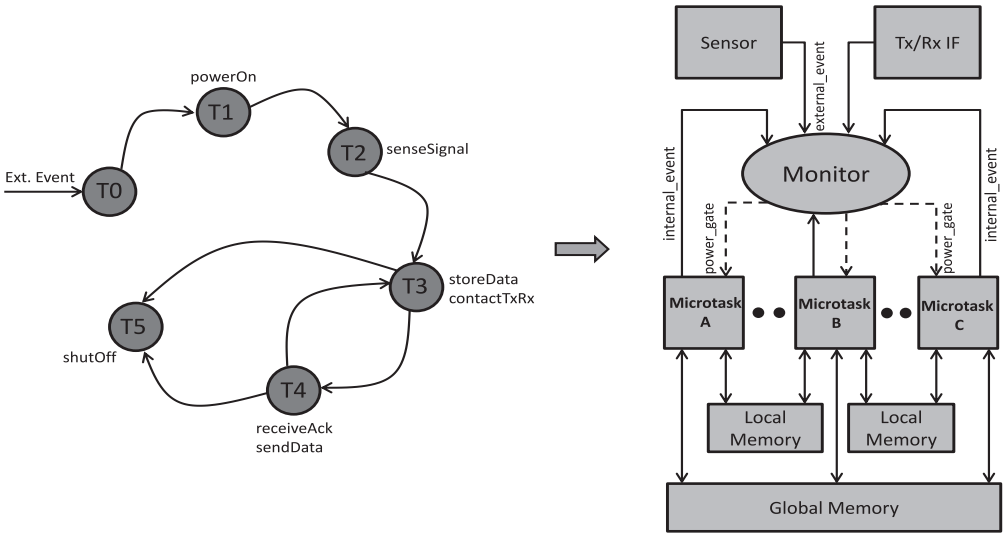


FIGURE 1.2: A generalized task flow graph and system level view of generated architecture as proposed in [1].

Reconfigurable hardware that use configurable logic blocks (CLBs) and programmable interconnection networks as in FPGAs, have been proposed as a solution to address the need for flexibility in general. In wireless applications that need to support multiple standards and be able to adapt transceiver algorithms, the programming-in-space approach of reconfigurable processors offers flexibility as also efficiency in terms of NRE costs compared to ASIC-based designs [47]. However flexible hardware of reconfigurable processors are not energy efficient unlike dedicated circuits and hence pose power and energy constraints for WSN nodes.

In this thesis, variable precision arithmetic units for datapath and scalable architectures for reconfigurable FSMs with ultra-low power consumption are explored in the context of flexible microtasks. Power gating as a low power technique is studied at architecture and circuit levels for FSM and datapath elements of microtasks described in [1]. It involves an extensive study of design issues in power gating and modeling design parameters of a power-gated circuit. A bottom-up approach is taken: gate level models for estimation of key design parameters of power-gated circuits are derived first. These models are useful in fast analysis of power-gated circuits. Next, two hardware blocks integral to microtasks *viz.*, arithmetic units of a datapath and finite state machine for control, are studied for low power optimizations. In both cases reconfigurability is considered a key aspect of flexible hardware design. In the context of arithmetic units like adders, reconfigurability is used for dynamically varying precision of the operation and hence examined for potential energy savings from unused logic. For reconfigurable FSMs, scalable architectures with varying degrees of flexibility and complexity are proposed and power gating opportunities are identified to achieve aggressive leakage power and standby mode energy reduction. Finally, the models are applied to analyze various design parameters in power-gated logic clusters of FSMs and adders in datapath due to power gating. An additional interest in the derivation of models is the possibility of using them as cost functions when synthesis of power-gated logic clusters is viewed as an optimization problem with design constraints to be satisfied.

Our approach has been biased towards power reduction at the cost of increased hardware area for two reasons. It is intended to introduce a high degree of flexibility in the hardware through programming-in-space approach in the first instance. Secondly, as technological nodes scale down, the increase in static power density as a cost far exceeds the impact of area due to flexible hardware.

1.4 Contributions

The specific contributions of this work are enumerated as follows.

1. A semiempirical model for estimation of wakeup time at gate level in power-gated logic clusters is proposed. Given an arbitrary netlist and compact characterization

of constituent logic gates and sleep transistor, wakeup time may be estimated rapidly within an average error margin of 16%[2] while avoiding transistor level simulations of complete netlist. It can in turn be used as a cost function for wakeup time in wakeup scheduling of power-gated logic clusters.

2. The static power of an arbitrary logic block can be estimated within an accuracy of 3% using simple polynomial based characterizations of static current at logic gate level. Further key design parameters of power-gated circuits, namely, steady-state virtual supply voltage in active and sleep modes, wakeup energy and energy savings due to power gating are obtained either as part of derivation of the model or its simple extensions. These models when used together have the potential to serve as a rapid analysis engine in a design flow.

3. A general approach to partitioning regular and dense arithmetic circuits like adders and multipliers to exploit variable precision operation is presented. It is shown that energy savings of upto 30% can be achieved during variable precision operation of adders and unused logic being power-gated. Precision scaling by power gating, when applied to data processing in a microtask can enhance the scope of its applications that is now limited to simple control tasks. Determination of a cut-off precision beyond which dynamic power saving also results helps define a power gating schedule to maximize power (and energy) savings.

4. Scalable architectures for reconfigurable FSMs based on Shannon decomposition of next-state functions and output functions are proposed. Power gating opportunities at the level of a configuration and for aggressive leakage reduction within active mode of a configuration are identified. Power estimation based on the models proposed show that the energy efficiency in terms of an equivalent energy per instruction metric with respect to a low power microcontroller core lies between that of the core and hardwired microtasks. The architecture represents a potential alternative in design space for flexible controllers especially as technological nodes scale down.

1.5 Organization of the Thesis

The thesis is organized as follows. A background of controllers in wireless sensor network nodes is given in Chapter 2. Reconfigurable circuits in controllers are discussed from the perspective of introducing flexibility in microtasks. A general homogeneous FPGA for FSMs in controllers is discussed along with power estimation at a first order of approximation for comparison with low power architectures proposed later in the thesis.

[2]As applied to ISCAS85 benchmark circuits.

It also serves as a motivation for problems addressed in this work. The chapter ends with the state-of-the-art on low power reconfigurable circuits.

In Chapter 3, previous work on design of power-gated circuits is reviewed in detail. The different types of power gating networks and the challenges involved in designing them are described. The various factors required to be taken into account for power-gated circuit design are enumerated and form the basis for contributions described later in the chapter. A power gating example is given to illustrate the aspects discussed. Hence a semiempirical model for wakeup time and its extensions to determine wakeup energy and energy savings in a power-gated logic cluster are presented. A key aspect of derivation of these models is the polynomial representation of static current in logic gates and its use in estimation of steady state virtual supply voltage at the end of wakeup mode. Potential applications are described to highlight the utility of models. The limitations of the model are also indicated.

In Chapter 4, power analysis of two types of power-gated adders that are configurable for different precisions is presented. The benefits and costs of power gating unused logic when reconfigurable adders operate at a lower precision are studied. A general method for logic partitioning and clustering for power gating based on desired precisions that can be extended to other arithmetic units like multipliers is described. The models of Chapter 3 and a classical approach for dynamic power estimation are used for power estimation in power-gated adders along with SPICE simulations. Such an analysis is useful in the context of microtasks since an adder forms an integral part of the datapath.

In Chapter 5, lookup table-based reconfigurable architectures that exploit feedback and feedforward logic structures in finite state machines are proposed. A thorough study of resources required for various flavours of fully reconfigurable FSMs is carried out. Hence trade-offs between FSM specifications and resources leading to FSM realizations with limited reconfigurability are discussed. Power gating opportunities are identified in these architectures at a granularity of lookup table logic clusters. Power estimation in architectures proposed for reconfigurable FSMs is described and the results are validated by transistor level SPICE simulations. A restricted class of sequential circuits referred to as linear sequential circuits are also studied in the context of reconfigurable FSMs.

The thesis concludes with Chapter 6 by elaborating on inferences derived from the work described in various chapters. Different solutions for WSN node controllers in design space are re-examined with the objectives of comparing energy efficiency and costs incurred for their realization. Finally some perspectives on future work are also outlined.

Chapter 2

Controllers for Wireless Sensor Network Nodes

2.1 Introduction

Given the tight energy constraints that a WSN node needs to satisfy, every aspect of its function is targeted for low power optimizations. Controllers are an important part of WSN nodes and are responsible for management of resources of nodes at a specified level of power budget and performance. In this chapter, the issues involved in trading flexibility and power constraints in controllers of wireless sensor network nodes are examined.

An overview of a wireless sensor network and its nodes is given in the next section with an illustrative example. In Section 2.3, the various ways in which controllers are realized in WSN nodes are surveyed. The trade-offs with respect to flexibility, performance and power in each are examined. Controllers based on reconfigurable microtasks are introduced in Section 2.4 with an enumeration of different reconfigurable logic blocks within the microtask. In Section 2.5, an embedded FPGA that represents a general reconfigurable system is described with reference to a physical implementation of its architecture. Bottlenecks encountered in mapping FSMs representing typical control tasks on to the FPGA architecture is described. The demands on FPGA resources and power consumption in the reconfigurable array are highlighted. A review of low power flexible hardware is presented in Section 2.6. The chapter ends with a discussion of problems addressed in this work.

2.2 Wireless Sensor Network Nodes

A node is a fundamental unit of wireless sensor network. Several nodes (from tens or hundreds to tens of thousands) form wireless sensor networks [21]. A node is in general

controlled by a controller of nodes but often, the control may be distributed. A node can be a transmitter, repeater, receiver or any combination of these apart from being able to do computation, sensing and data storage. Also, a node may change its behaviour dynamically over time. A group of nodes may communicate with another group of nodes in a cooperative way sharing communication and computational resources. A node may also be a broadcast node by just transmitting signals for all nodes within its communication reach. Thus nodes can be networked in a number of ways and each networking topology is governed by various factors like total computational capacity, energy budget, channel conditions, communication protocols and methods, intended application, etc.

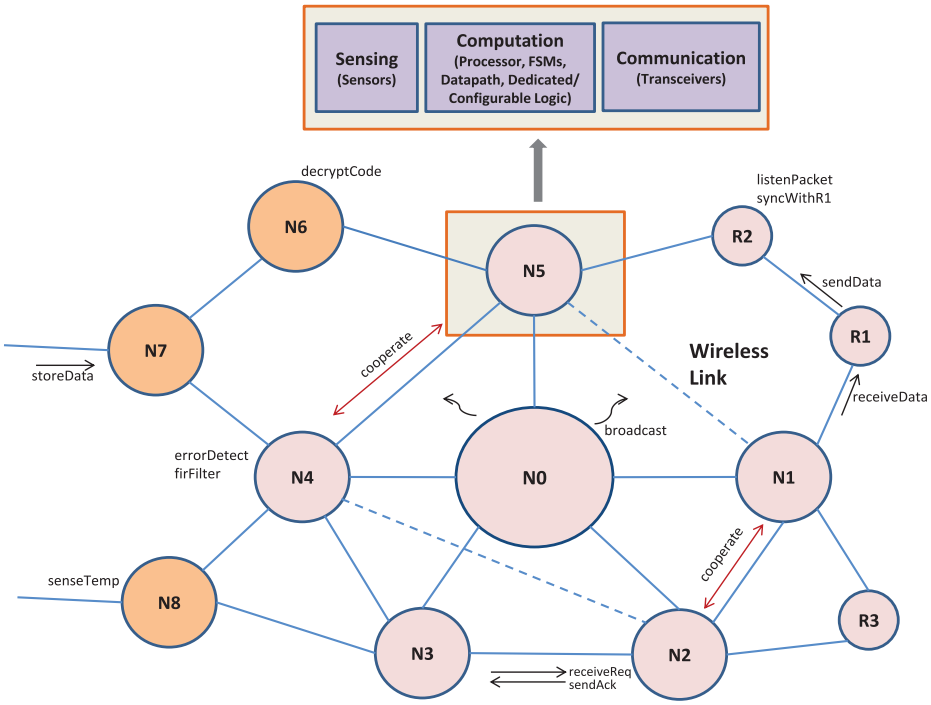


FIGURE 2.1: Functional representation of a wireless sensor network.

A highly simplified functional representation of a wireless sensor network with its nodes Ni, $i = 0, 1, 2, ..., 8$ and Ri, $i = 1, 2, 3$, is shown in Fig. 2.1. It shows an indicative set of tasks that nodes may be expected to perform within constraints of power, performance and ubiquity. For example, nodes N2 and N3 communicate by an asynchronous handshake protocol requiring execution of *send(receive)Req* and *receive(send)Ack* tasks by controllers within them. Nodes Ri act as repeaters requiring *receive(send)Data* tasks to be executed. Further, the pairs of nodes (N1, N2) and (N4, N5) may cooperate among themselves by dynamically sharing data and communicating with the other pair. Therefore functionally, node and network design are application dependent. However the evolution of WSNs and communication protocols [48] including traffic-aware [49] and dynamically adaptable ones makes it preferable to have flexible nodes so that they may

be used across applications. Further the cost and time for design and production of nodes may also be reduced for large volumes.

The basic structure of a node was described in Chapter 1, Section 1.2 to broadly consist of four units. Among them, while sensing and communication units interface with the external world, the computational unit is responsible for computational and control tasks. An important sub-unit of a computational unit is the node controller[1]. Alternatively, a group of generic controllers may constitute a computational unit. Some of the controllers may have an overlap with sensing and communication units as well. The focus of this chapter is two fold: (1) to survey the various ways in which control tasks are handled in a WSN node and (2) to motivate a search for low power reconfigurable architectures that are both flexible and are close to ASIC-based controllers in the design space.

2.3 Controllers for WSN Nodes

The primary function of controllers in nodes is to manage resources for required performance with low-power (and energy) demands. Resources in the node include datapath elements for arithmetic and logic operations on data, timers, interrupt controllers and event generators, memory for data storage, peripheral IOs for external interfacing, power supply and system monitoring circuits. It is estimated that upto 25% of total power budget in a node may be consumed by controllers. Microcontrollers have been an important choice for controllers in nodes due to their flexibility. Several WSN platforms have used low-power microcontrollers as their driving engines for control tasks. A popular microcontroller, MSP430 of Texas Instruments [50], has found application in nodes like Telos [5], PowWow [6] and HydroWatch. Some of the other low power microcontrollers for WSN nodes have been CoolRISC [51] (in WiseNet), ATMega series (in BTNodes [7]) and Intel's StrongARM. A spectrum of low power techniques - from clock gating to subthreshold design including power gating [8, 9, 10] have been implemented in microcontrollers proposed for nodes.

Some of the solutions for controllers have been based on customizable core generators like Xtensa family of cores of Tensilica [52]. The cores can be generated with application-specific instructions that use custom-designed accelerators. While the cores have to be integrated into the system and hence the circuit fabricated, a rapid core generation engine and tool support ensures fast turnaround at low NRE costs. Application Specific Instruction Processors (ASIPs) are an useful extension of microcontrollers with

[1]The term controller appears in different contexts in this thesis. A controller in a node is different from controller of nodes (e.g., N0 in Fig. 2.1) in a wireless sensor network. A controller of nodes may have a controller within itself. Some authors use the term controller synonymously with the control part when discussing control and datapath model of a digital system. In our work, a controller consists of one or more microtasks that have an FSM-oriented control part and a datapath.

customized instructions. Nevertheless microcontrollers are far from optimal solutions in several applications as sequential execution of steps in control tasks exercise significant logic of a microcontroller to cause high power consumption. Examples of such operations include memory access for reading and writing instructions or data, frequent branching and unsystematic indexing. Further full-sized datapath and instruction decoder logic are completely used even for control tasks of lower complexity.

A number of systems for digital control can be abstracted by a model based on a control unit and a datapath. Different behavioural models have been used to describe the control unit. However from the physical implementation viewpoint two major styles of control logic realizations are recognized: (1) microcode oriented control and (2) finite state machine oriented control. Different flavours of implementations exist in both forms of control unit realizations. A datapath on the other hand, consists of arithmetic, logic and storage units.

An intermediate representation for compile-time reconfigurable controllers in chip generators based on partial evaluation was proposed in [53] with an example of a cache controller. Partial evaluation is a technique used to specialize a generic microcode sequence (program) based on compile-time inputs. For partial evaluation of reconfigurable controllers in chip generators, table-based control is converted into efficient logic implementation for synthesis using state propagation and state folding.

Previous work in our group proposed a complete design flow for system-level synthesis of low power WSN node controllers in [1]. The function of a controller in a node is specified in terms of a task flow graph. The task flow graph is an input to the design flow at a high level of abstraction (e.g., in C or a domain specific language) and functionally it represents a set of tasks the controller has to execute. From the perspective of hardware implementation, the synthesized controller consists of a set of specialized hardware units referred to as microtasks shown in Fig. 2.2. From the figure it can be inferred that the microtask is based on a FSM-datapath model. The microtask consists of a dedicated FSM and datapath for control and computations respectively. Register files, memories for local storage and peripheral IO interfaces form a part of datapath along with arithmetic-logic units. In other words, a microtask, once generated by the design flow, is specialized to execute a specific control task. An implementation of a node controller involves several such microtasks specialized for tasks in the task flow graph. A system monitor manages scheduling of microtasks according to the task flow graph. A domain-specific language (DSL) was also described to specify system-level execution model of a WSN node to generate a system monitor for a given task flow graph.

In order to exploit low-power techniques in generated controllers, the tasks are mapped on to microtasks in such a way that during the operation of the controller based on run-to-completion semantic, all microtasks than the ones required may be

power-gated to suppress leakage power. Additionally, the hardware specialization of microtask is an ASIC-like feature in that the dynamic power consumption characteristics are similar to ASICs. This approach leads to low power realizations of controllers in WSN nodes. In the design flow, the outputs are obtained as a hardware description (in VHDL) of controllers at register transfer level (RTL). The structural description of controller in VHDL is then used to perform logic synthesis for technology mapping. The synthesizable code in VHDL is obtained using model driven engineering and retargetable compilation methods. The approach is in principle similar to core generators described above, but with differences in flexibility offered by the complete systems generated by the two design flows and low power features.

Power estimation results show a reduction in power by factors in the range of 64 to 185 for typical controller tasks as compared to their execution on openMSP430 [54] microcontroller. Similarly, the controllers based on hardware-specialized microtasks are about 20 times more efficient than openMSP430 microcontroller in terms of energy per operation (or instruction) metric. While ASIC implementations of generated microtasks are power-efficient and that their architectures can be generated rapidly using the proposed design flow, the utility is limited by the specificity of application. A controller based on hardwired microtasks cannot be programmed for a different task flow graph either in space or in time to execute control tasks other than those designed for.

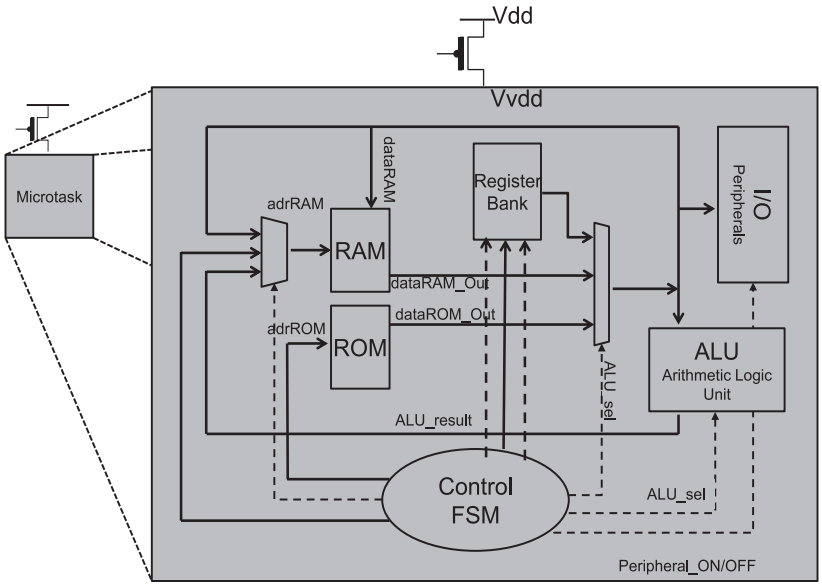


FIGURE 2.2: Structure of a microtask (as proposed in [1]).

2.4 Reconfigurable Microtasks

The subject of this thesis is to explore possibilities of introducing flexibility in microtasks and yet at the same time retaining optimizations for low power at a certain trade-off with respect to area. Reconfigurable architectures provide a way to incorporate flexibility by being able to map logic circuits on to basic configurable logic blocks. In this work we consider reconfigurable architectures for Control FSM and Arithmetic Unit of ALU in the microtask of Fig. 2.2. The term reconfigurability broadly refers to configuration of certain basic logic blocks and control of interconnections between them to realize several possible logic functions. It is also referred to as a programming-in-space approach in contrast to the programming-in-time approach of microprocessors.

The structure of a generated controller with reconfigurable microtasks and a typical architecture of a microtask is shown in Fig. 2.3. For a microtask to be flexible, it is envisaged that the control finite state machine, the arithmetic unit and the register file of the microtask shown in Fig. 2.2 be replaced with their reconfigurable counterparts. Clearly, a family of task flow graphs can be implemented by merely reconfiguring the same configurable logic blocks in the microtask. A flexible controller in its simplest form may have only one reconfigurable microtask and a system monitor with associated memories. The different tasks of a task flow graph under consideration may then be implemented as consecutive configurations of the flexible microtask. The overhead due to reconfiguration time and reconfiguration memory needs to be considered. Alternatively, a controller may also have multiple reconfigurable microtasks of different sizes and types as shown in Fig. 2.3 and also be managed by a system monitor. This allows a possibility of interaction among microtasks or executing parallel tasks. From a hardware perspective two important costs, area and static power, are incurred and have to be quantified. Further there exists a one-time cost in development of architecture mapping tools for the reconfigurable architectures. To control power dissipation due to increased logic integration, it is necessary to explore opportunities for power gating in the spatial dimension at a finer granularity (unused logic at any given time) apart from power shut-off during standby modes.

Reconfigurable systems have typically been realized with PLAs, FPLDs and FPGAs. In this thesis the term reconfigurability (or reconfigurable systems) is used in three contexts:

1. FPGA mapping of finite state machines,
2. Variation of precision in adders for power gating unused logic at lower precisions of operands, and
3. Realization of different finite state machines by programming the hardware fabric for next-state functions and output functions.

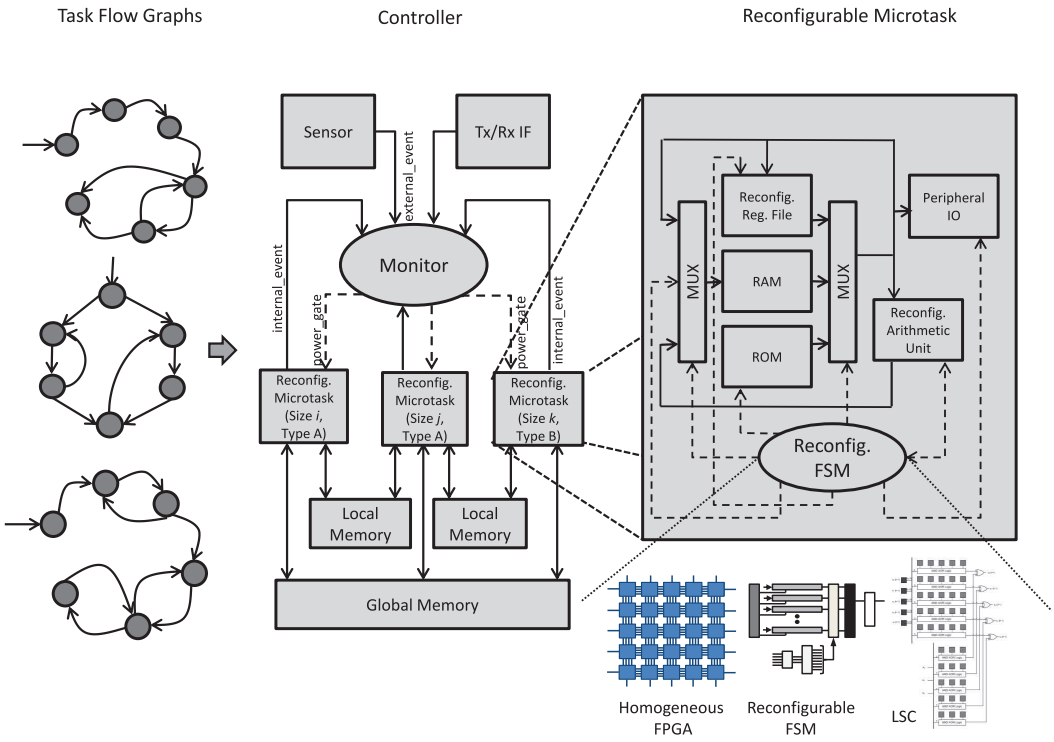


FIGURE 2.3: Structure of flexible controller with reconfigurable microtasks.

The last two contexts address certain specific classes of logic systems and each class has a set of parameters that may be reconfigured to address flexibility in that class of logic system. For example, reconfigurability with respect to adders may involve precision and type of carry generation whereas reconfigurability with respect to FSMs may include number of state register bits, number of primary inputs and outputs as parameters. Therefore the last two contexts are similar to each other in the sense that the reconfigurable hardware fabric is intended for specific classes of logic systems (FSMs, adders) whereas the first is a general reconfigurable system. In general reconfigurable systems, flexibility is quantified by configurable logic blocks (CLBs) available, programmability of interconnections between CLBs, number of routing channels for signals and number of inputs and outputs for external interfacing. These systems are not directed towards any particular class of logic systems.

In the rest of the chapter, a general FPGA is reviewed for mapping of FSMs typical of tasks executed by node controllers. In Chapter 4 and Chapter 5, architectures that are reconfigurable for certain parameters and optimized for specific functions *viz.*, arithmetic units and finite state machines that constitute a controller, and that lead to low power implementations are studied.

2.5 Embedded FPGA

One of the most common reconfigurable systems is a hardware fabric with several configurable logic blocks and a programmable interconnection network of switches for routing inputs and outputs across a matrix of CLBs. It is referred to as an island-style or a homogeneous FPGA [11, 12]. A CLB consists of a K-input lookup table (K-LUT), configuration bits (CB) and possibly a register for programmable delay of one clock cycle. A LUT is essentially a set of 2^K storage elements (memory or registers) that contains the truth table of a function to be implemented and is decoded by K input variables as a K-bit address. LUTs have also been widely used in compact realizations of decoders and compute intensive logic like adaptive digital filters. The CLB based on a 4-LUT and the configuration bit used in an array element is shown in Fig. 2.4 and Fig. 2.5 respectively. The configuration bit consists of two storage elements. The latch stores the configuration bit corresponding to the current or active configuration whereas the flip-flop shown in the figure is used to store the subsequent configuration. Provided that the current configuration is operational for a time longer than the time for reconfiguration, the overall CB structure ensures a zero-latency reconfiguration from a system point of view. In other words, dynamic reconfiguration can be achieved at the cost of an additional storage register.

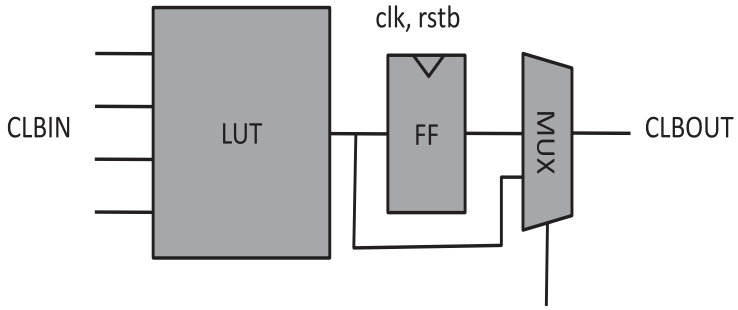


FIGURE 2.4: Configurable Logic Block (CLB) structure in eFPGA.

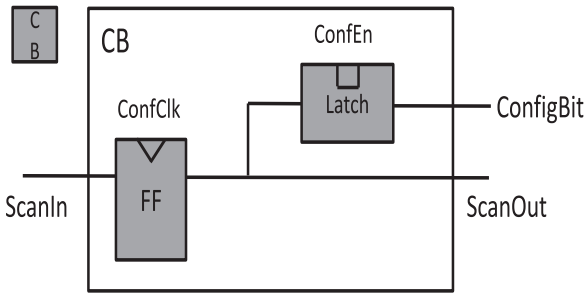
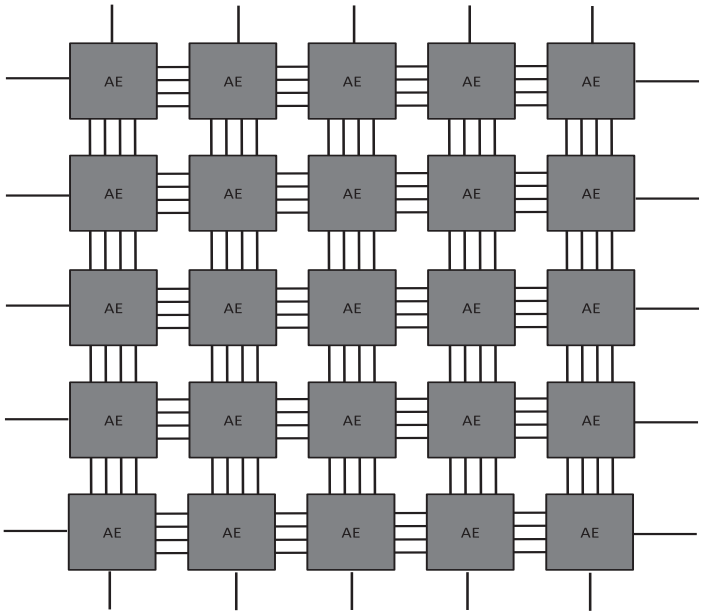


FIGURE 2.5: Configuration Bit (CB) in eFPGA.

In this section, an array of such CLBs and a routing network is described to serve as a basis for comparison with proposed reconfigurable FSM architectures. For brevity it is referred to as embedded FPGA (eFPGA) in this thesis. The architecture was developed as part of previous work in the team to support efficient dynamic reconfiguration [55]. A top level view of the homogeneous array of CLBs and interconnections of eFPGA is shown in Fig. 2.6. In Fig. 2.7, the basic element of the overall array consisting of CLB



AE: Array Element

FIGURE 2.6: Top level view of eFPGA.

and two connection boxes (CHANX and CHANY) that provide access to neighbouring array elements and a switch box (SB) that acts as a crossbar routing switch between two segments of adjoining routing channels are shown. The switches are designed using transmission gates whose function is controlled by configuration bits as shown in Fig. 2.8. A transmission gate consists of a parallel connection of a PMOS and NMOS pass transistors. An NMOS pass transistor faithfully passes a logic 0 (low) while a PMOS pass transistor passes a logic 1 (high). Hence both states of a CLB output is transmitted without any attenuation along the routing lines. In this work a 4-channel routing network is assumed for an array of 22 $\times$ 22 blocks as baseline architecture.

2.5.1 Resource Utilization

The resource utilization for a logic circuit, with respect to the reconfigurable array on to which it is mapped, is defined as the number of CLBs utilized to realize the specified logic, with the requirement of minimum number of channels for interconnections between CLBs being satisfied. It is assumed that the array has sufficient number of input and

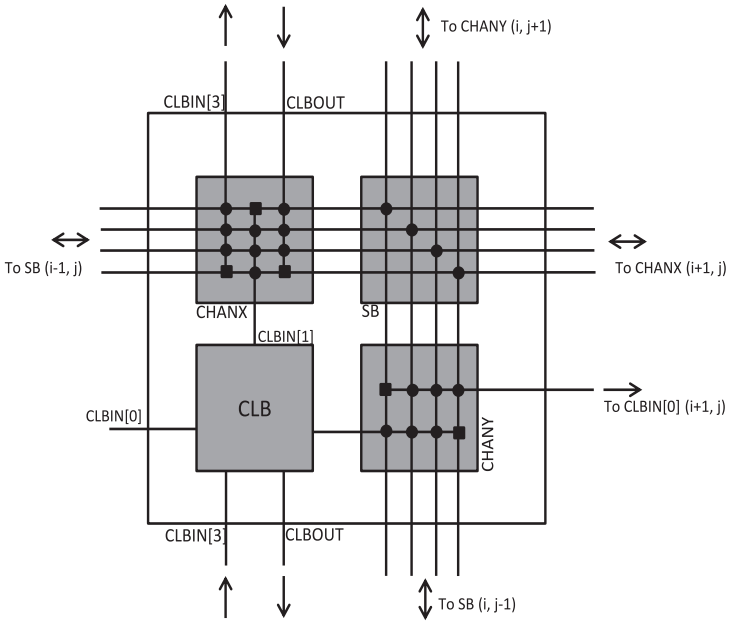


FIGURE 2.7: An array element of eFPGA with routing channels and interconnection network.

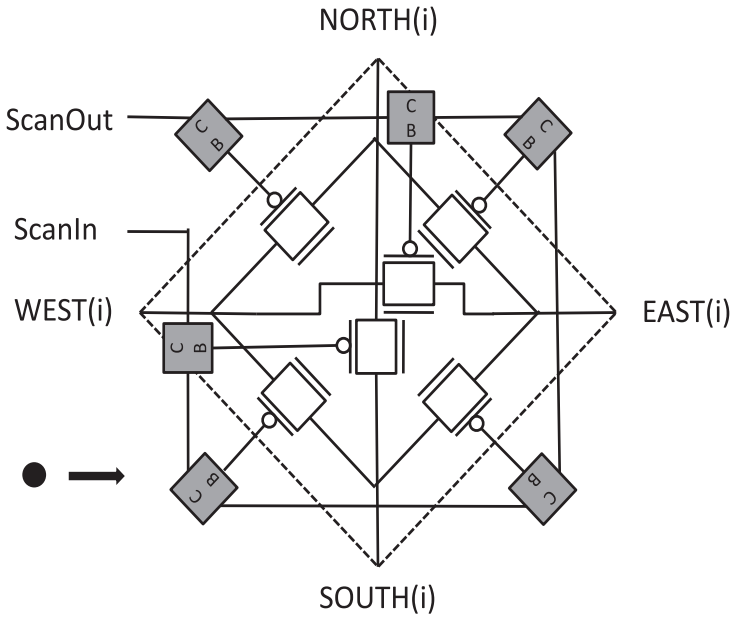


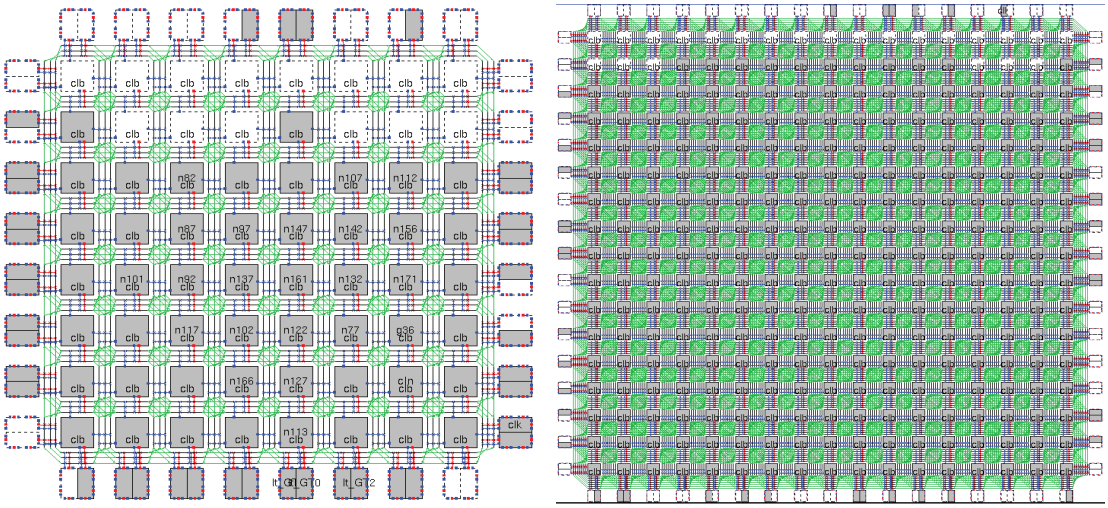
FIGURE 2.8: Switch box in eFPGA.

output ports to interface the logic circuit externally. To study the resource utilization characteristics of some of the representative FSMs used in controllers, the finite state machines listed in Table 2.1 were mapped, placed and routed using the Verilog-to-Routing (VTR) tool [56] with eFPGA array as the targeted architecture. The FSMs were chosen from benchmark descriptions in SenseBench [13] that includes tasks such as arithmetic absolute value, 8-bit and 16-bit cyclic redundancy check and FIR filtering. The RTL

FSM	N	N_{CLB}	MCW	L_{av}
abs	5	50	3	3.55
Crc8	6	84	4	4.80
receiveData	6	94	4	4.58
Crc16	7	143	5	7.25
firBasic	7	217	7	7.89
calcNeighbor	8	266	7	8.05

TABLE 2.1: Resource utilization N_{CLB}, minimum channel width (MCW) and average interconnection length (L_{av}) required for FSMs on eFPGA-like array.

descriptions of the FSMs were obtained from the work of [46]. The table shows the number of CLBs (N_{CLB}) and minimum channel width (MCW) required for implementation of each FSM with N state register bits along with average length of an interconnection segment in terms of number of CLBs spanned (L_{av}). Clearly for the targeted eFPGA architecture with channel width of 4, only the first three FSMs listed in the table can be mapped whereas the last three FSMs face constraints with respect to channel width, although the number of CLBs are sufficient for logic function mapping. This reinforces a well known contention that in a FPGA, the complexity of interconnection network for signal routing creates a bottleneck as the complexity of logic circuit increases [57]. To proceed with a study of resource utilization and power estimation, the architecture is scaled to accommodate larger FSMs of Table 2.1. A visual representation of interconnection networks required for two FSMs in an eFPGA-like architecture with required MCW as obtained from a mapping and PnR with VTR tool is shown in Fig. 2.9.



(a) abs (b) calcNeighbor

FIGURE 2.9: Interconnection network complexities in mappings of two FSMs.

2.5.2 Power Estimation

Power estimation in a general homogeneous FPGA is a challenging task. Since most of the components of total power consumption in a CMOS circuit show state dependence and signal activity in nodes, power consumed may only be estimated statistically for a range of variations in states of circuit nodes. Added to the difficulty of enumeration of circuit node states is the configurability of the array for logic systems of varying complexities leading to different resource utilization and activity patterns. A comprehensive survey of techniques for power estimation in FPGAs is given in [58]. Two key approaches for power estimation in configurable systems have been based on simulation and probabilistic methods. As the complexities of commercially available FPGAs increased, more empirical methods have been adopted. A linear regression based power model for Xilinx FPGAs to predict switching activities was proposed in [59]. Curve fitting theories and empirical formulae were used for size and activity extrapolation. A power estimation methodology based on estimation of switching activities using pseudorandom inputs for different benchmark circuits and metrics to validate their accuracy was proposed in [12] and has been included in the VPR tool flow.

A polynomial-based approach for leakage power estimation [60] (Chapter 3, Section 3.4) and a simulation-based switched capacitance estimation with a large set of random inputs for average dynamic energy estimation is followed for eFPGA and rest of the reconfigurable architectures in this thesis. Statistical parameters are assumed to be identical across mapped CLBs. It is reported in [58] that the assumption of spatial independence of statistical parameters in FPGAs results in an error margin of about 20% relative to those with spatial correlation of signals in benchmark circuits. The objective of power estimation in logic systems mapped on to eFPGA architecture is to obtain optimistic average power estimates of various components for comparison against that of reconfigurable FSM architectures with low-power techniques applied.

In order to estimate power consumption in FSM logic when mapped on to eFPGA architecture, the three routable FSMs in Table 2.1 are considered first. Estimation of power consumption involves four components *viz.,* (1) static power consumption in CLBs (2) dynamic power in CLBs (3) static power consumption in the interconnection network and (4) dynamic power in the interconnection network. Static power in CLBs is due to leakage currents in constituent logic gates and configuration bit registers whereas in the switching fabric of the array, static power is mostly due to leakage currents in configuration bits for switches and buffers along routing lines and clock networks and is present at all times. From Fig. 2.7 it can be inferred that the configuration bits in the switching network will contribute to the static power considerably more than a CLB in an array element. Further since configuration bits must remain in powered-on state all the time irrespective of the configuration, the only possibility of reducing its static

power is to use power optimized standard cells. For this architecture we estimate static power of configuration bits using standard cells (flip-flops and latches) of high threshold voltage devices in two families of the same technology library, namely,

1. general purpose ("Family A") cells with high-V_{th} devices and
2. low power ("Family B") cells with high-V_{th} devices.

Typically, standard cells of Family B have higher V_{th} than those of Family A and hence have a higher cell delay. The two types of cells represent two extremes of possible variation in static power and delay among cells with high-V_{th} devices when used under identical conditions. In as far as configuration bit scan chain is concerned, since the configuration clock does not have to be of high frequency, longer clock-to-output delays are not significant.

Dynamic power consumption depends on the circuit capacitances being switched when inputs to CLBs and logic states along routing lines in the channels change. Therefore only those CLBs that are mapped with the logic function consume power in the active mode.

2.5.2.1 Model for Optimistic Power Estimation

The total static power consumed by the FSM (mapped CLBs) is given by

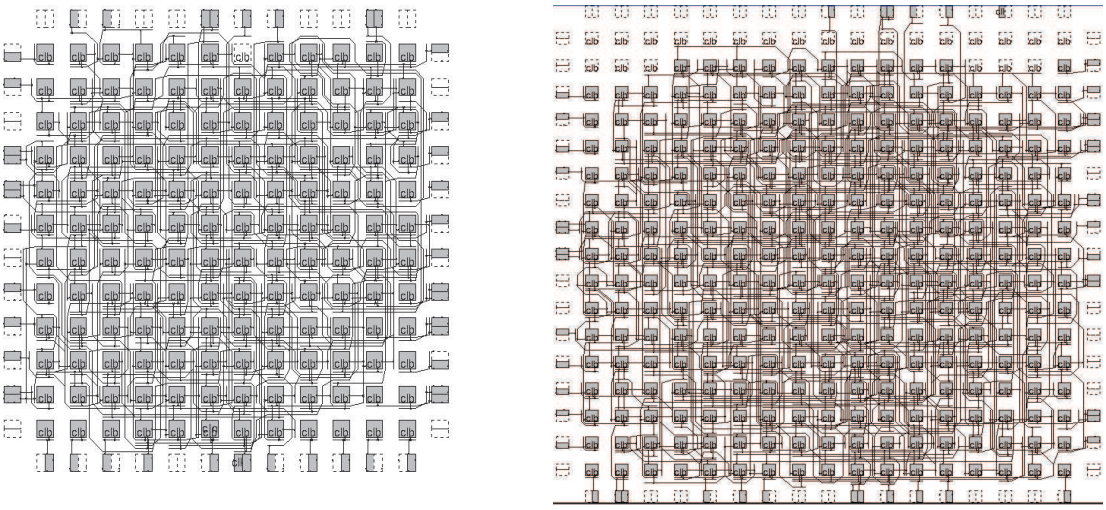
$$P_{static,FSM} = N_{CLB} P_{static,CLB} + P_{static,SW(CLB)} \tag{2.1}$$

where N_{CLB} denotes the number of CLBs mapped to FSM logic in the array, $P_{static,CLB}$ denotes the static power in one CLB and $P_{static,SW(CLB)}$ represents the leakage power in interconnection network of mapped CLBs consisting of transmission gate switches and configuration bit registers for switch control. Unmapped CLBs also contribute to static power when additional static power reduction techniques have not been applied. Denoting by N_{ARRAY} the number of CLBs in the complete array, the total static power of the array is given by

$$P_{static,ARRAY} = N_{ARRAY} P_{static,CLB} + P_{static,SW(ARRAY)}. \tag{2.2}$$

The assumptions made in the optimistic power estimation described next are enumerated below.

1. The mapped CLBs are assumed to be compactly placed adjacent to each other and that half of the interconnection resources in an array element are shared by adjacent CLBs. Figure 2.10 shows two examples of FSMs being placed and routed with their minimum channel widths satisfied.



(a) Crc16 (b) calcNeighbor

FIGURE 2.10: Compact placement and routing in mappings of two FSMs.

2. The dynamic energy due to routing lines along clock networks and static energy of buffers for both routing channels and clock networks are not considered.

3. The dynamic energy due to reconfiguration of eFPGA by shifting configuration bits in a scan chain fashion is also neglected. This is due to the fact that the eFPGA is usually programmed to be in a particular configuration for a long time so that average power can be considered small.

4. The bias and transistor size dependent leakage current of NMOS and PMOS transistors in switches are ignored. Such currents are mostly sneak currents and are negligible in the context of transmission gates.

5. Spatial independence of dynamic energy consumption in mapped CLBs across the eFPGA is assumed. As will be shown in the next subsection dynamic energy consumption within a LUT is significantly lower than that of interconnection network and will have negligible impact on estimation of overall energy consumption [61].

The energy consumed due to transition of signal states in the logic of CLB and metal lines of the interconnection network due to changing inputs and outputs is given by

$$E_{dyn_{FSM},av} = N_{CLB} E_{dyn_{CLB},av} + E_{dyn_{SW},av} \tag{2.3}$$

where $E_{dyn_{FSM},av}, E_{dyn_{CLB},av}, E_{dyn_{SW},av}$ denote average energy consumed in all the blocks used, one CLB and the switching network respectively. In order to estimate $E_{dyn_{CLB},av}$ 10000 random inputs are applied to CLB logic and switching capacitance is determined

from the classical model by gate level simulation as

$$E_{dyn_{CLB},av} = \frac{V_{dd}^2}{N_{input}} \sum_{i=1}^{N_{input}} C_{sw_i,CLB} \tag{2.4}$$

where $C_{sw_i,CLB}$ is the total switched capacitance per input pattern in CLB due to logic transitions in the internal nodes. Average dynamic power of the FSM is then obtained as

$$P_{dyn_{FSM},av} = N_{CLB} E_{dyn_{CLB},av} f_{clk} \tag{2.5}$$

where f_{clk} is the frequency of operation of the array.

To determine the capacitance of the switches and routing lines the model described in [11] is used. The path between the output of a CLB to the routed inputs of other CLBs is divided into segments, each terminated by the corresponding switch. The capacitance of each segment is given by

$$C_{seg} = mC_{diff} + nC_{wire}A_{wire,seg} \tag{2.6}$$

where m represents the number of switches, C_{diff} is the diffusion capacitance of the transmission gate switch, n is the number of routed wires, C_{wire} is the capacitance of metal wire per unit area and $A_{wire,seg} = W_{wire}L_{seg}$ is the area of wire in the segment, W_{wire} and L_{seg} are width and length of the routed wire in a segment respectively. Further

$$C_{wire} = C_{plate} + 2C_{fringe} \tag{2.7}$$

where the metal plate capacitance and fringe capacitance are denoted by C_{plate} and C_{fringe} respectively. In Eq. (2.6), it is assumed that the FSM is compactly placed and routed in the array that every segment of a group of adjacent routing channel and CLBs are used. This represents an ideal scenario that serves as a baseline estimate for comparisons. The energy consumed for signal transitions along four channels of eFPGA between two adjecent CLBs is given by

$$E_{dyn_{SW}} = (129C_{diff} + 4C_{wire}A_{wire,seg})V_{dd}^2. \tag{2.8}$$

To a first order of approximation, the energy consumed in the array for an FSM implementation can be estimated to be

$$E_{dyn_{SW},FSM} = \frac{N_{CLB}}{2} E_{dyn_{SW}}. \tag{2.9}$$

Hence the total power of an FSM is given by

$$P_{FSM} = P_{static,FSM} + P_{dyn_{FSM},av}. \tag{2.10}$$

2.5.2.2 Results

To estimate the static power and dynamic energy consumption in the eFPGA-routable FSMs of Table 2.1 the following experiment was performed. A CLB shown in Fig. 2.4 was synthesized into a gate level netlist using a 65nm industrial technology library. Further the basic array element shown in Fig. 2.7 was placed-and-routed to obtain typical wire lengths in each segment. Some of the important parameters of the array element in eFPGA are given in Table 2.2. The transmission gate switches were designed as standard cells for use in connection box and switch box. The total static power of the CLB at supply voltage of 1V was determined from the model as $V_{dd}I_{leak}$ from Eq. (3.11). We use $N_{ARRAY} = 22 \times 22$ and N_{CLB} is given in Table 2.3 for each FSM considered in this experiment. The results of static power and dynamic energy estimation using models described above is given in Table 2.3. In all the following tables, the static power estimate due to configuration bits with Family B register cells is indicated in parenthesis.

Parameter	Value
Technology Library	65nm CMOS (SVT, HVT Cells)
Area of an Array Element	$4963\mu m^2$
Relative Area of Interconnection Network in an Array Element	88.26%
CLB-CLB Routing Metal Length and Width (per channel)	70μm and 0.1μm
$P_{static,CLB}$	9.3750μW
$P_{static,SW,AE}$ 4 Channels 5 Channels 7 Channels	 43.68 (0.442) μW 56.01 (0.565) μW 80.44 (0.812) μW

TABLE 2.2: Important eFPGA parameters as obtained from physical design. (Estimates due to 'Family B' low power registers are in parenthesis.)

eFPGA	N_{CLB}	$P_{static,CLB}$ $\times N_{CLB}$ (μW)	$P_{static,SW}$ (μW)	$E_{dyn_{CLB},av}$ $\times N_{CLB}$ (pJ)	$E_{dyn_{SW},FSM}$ (pJ)
abs	50	468.7	2188.5 (22.11)	0.85	13.49
Crc8	84	787.5	3676.7 (37.15)	1.43	22.67
receiveData	94	881.2	4114.4 (41.57)	1.60	25.37

TABLE 2.3: Resource utilization, power and energy estimation for routable FSMs on eFPGA.

Further the experiment is extended to include non-routable FSMs and a scaled architecture similar to eFPGA with channel width equal to the highest minimum channel width is assumed. The models are evaluated again for the scaled architecture with seven routing channels. The place and route tool is assumed to perform compact routing so

eFPGA	Total Power of Mapped CLBs (mW)	
	f_{clk} = 20 MHz	f_{clk} = 100 MHz
abs	2.94 (0.78)	4.09 (1.92)
Crc8	4.95 (1.31)	6.87 (3.23)
receiveData	5.54 (1.46)	7.69 (3.62)

TABLE 2.4: Total power for routable FSMs on eFPGA. (Static power of unused array elements is not considered.)

that Eq. (2.9) is valid. Clearly such an energy estimate for switching and routing network is highly optimistic and should only serve as a lower bound. Table 2.5 shows power estimation with the scaled 7-channel architecture.

eFPGA	N_{CLB}	$P_{static,CLB}$ $\times N_{CLB}$ (μW)	$P_{static,SW}$ $\times N_{CLB}$(μW)	$E_{dyn_{CLB},av}$ (pJ)	$E_{dyn_{SW},FSM}$ (pJ)
abs	50	459.37	4022.6 (40.64)	0.85	13.49
Crc8	84	843.75	6757.8 (68.28)	1.43	22.67
receiveData	94	965.62	7562.2 (76.41)	1.60	25.37
Crc16	143	1565.62	11504 (116.24)	2.43	48.48
firBasic	217	2390.6	17458 (176.4)	3.69	103.58

TABLE 2.5: Resource utilization, power and energy estimation for FSMs on 7-channel scaled eFPGA.

eFPGA	Total Power of FSM (mW)	
	f_{clk} = 20 MHz	f_{clk} = 100 MHz
abs	4.77 (0.79)	5.92 (1.93)
Crc8	8.08 (1.39)	10.01 (3.32)
receiveData	9.07 (1.58)	11.22 (3.74)
Crc16	14.09 (2.70)	18.16 (6.77)
firBasic	21.99 (4.71)	30.58 (13.29)

TABLE 2.6: Total power for FSMs on 7-channel scaled eFPGA.

2.5.3 Observations on Embedded FPGA

The eFPGA described above is a general reconfigurable fabric with an architecture not optimized to take advantage of feedback and feedforward logic structures in a FSM. As an example, a reconfigurable FSM would not require more registers than N, the number of state register bits of the most complex FSM supported for cycle delay. Correspondingly, the complexity of interconnection network would be significantly lower.

No leakage current reduction techniques like power gating were applied, even at configuration time to suppress leakage current in unmapped CLBs. Assuming that power gating could be employed, the insertion of sleep transistor structures cannot be optimized for a particular FSM as CLB mapping characteristics vary among different FSMs.

Further all mapped CLBs need to be powered-on during active mode in general and hence active mode leakage current cannot be suppressed. Even if an attempt is made to have active mode power gating in eFPGA, problems arise in effectively separating power-gated domain and always-on domain. Due to dense interconnection network in an array element, it is difficult to design coarse grain power gating structures. Fine-grain power gating of LUTs would require a large number of sleep transistor control circuits and they are required to remain in always-on state. The use of as many isolation cells to separate always-on logic from power-gated logic will also add a cost in terms of static power and area.

As the number of CLBs are increased to support complex logic in nanoscale technologies, the static power due to leakage current in CLBs becomes significant. The results on static power consumption in logic-mapped CLBs in Table 2.5 are presented with a view that leakage power in unused CLBs may be suppressed using some of the techniques like power gating. In the case where it is not feasible, the static power consumption is almost constant, except for input dependent leakage currents in mapped CLBs for the array and is given by Eq. (2.2). For a 22×22 array it is estimated to be 11.8mW.

Another important observation about eFPGA architecture is that it can easily be scaled due to regularity of CLBs but the interconnection network may not always ensure that projected scalability is achieved as was determined from VTR place and route. Also, timing is not always guaranteed to be the same as interconnections for different FSMs may span across the array.

2.6 Low Power Reconfigurable Hardware

As was shown in the previous section, even an optimistic estimate of total power consumption in eFPGA can be compared with that of a microcontroller like openMSP430 [1]. The interconnection network and the configuration bits contribute upto about 71% of the total power consumption. Further, even for a modest increase in the complexity of a FSM, a significant increase in the complexity of switching and routing network is required leading to increased power consumption. Clearly such a high power consumption is not suitable for low power applications like WSN controllers.

In [11], FPGA architectures were explored for potential low power optimizations. It was also investigated if the high energy consumption was essentially an unavoidable feature of FPGAs. The problem has particularly aggravated in nanometer CMOS technologies due to an exponential increase in power due to leakage currents. It was suggested that circuit techniques could be explored to reduce power consumption. Pass transistor logic design techniques have been studied for look-up table implementations. In the eFPGA context low power optimizations were difficult to achieve without modifying the architecture itself. Being as general as it is, the uncertainty in mapping of CLBs for

applications makes such modifications difficult. FPGA routing problem is known to be NP complete [62]. Therefore an optimal solution may not be found by the routing tool in reasonable time.

Reducing supply voltage has by far been considered the most effective method for reducing power consumption in CMOS circuits. FPGAs with multiple supply voltages and threshold voltages have been proposed in [63, 64]. In [65], Ishihara *et al.* describe a low-power FPGA based on fine grain power gating at the level of 2-LUTs with autonomously generated sleep transistor control signals. Recently low-power FPGAs based on subthreshold design techniques have been proposed in [31]. With supply voltages in subthreshold or near-threshold region, the proposed flexible architectures are reported to provide good energy-delay characteristics sufficient for ultra-low energy WSN circuits. A significant outcome of studies in this topic is to restrict the configurability to classes of logic systems relevant to intended applications like wireless sensor networks. In other words, evolution of reconfigurable systems is more domain specific in nature.

Models for reconfigurable FSMs used for digital control have been proposed in [66, 67]. However the proposed models are not specifically targeted for low power architectures. The architectures also have a non-zero latency with respect to primary clock. In this thesis reconfigurable architectures for finite state machines that are optimized for low power are explored. In the first instance the complexity of switching and routing network is sought to be reduced by taking advantage of locality of routing and simple selector logic structures. Secondly power gating as a low power technique to reduce leakage power in active modes of operation is explored. The benefits of energy savings due to power gating in standby modes of operation are obvious.

2.7 Conclusion

A case for design of reconfigurable microtasks to support flexibility in microtask based design of controllers for WSN nodes was presented. Based on power estimation of FSMs mapped on to a general homogeneous FPGA and previous works it was argued that for reconfigurable systems to be power-efficient it is necessary to exploit the structure or function of classes of logic systems under consideration. This may involve for example, exploiting the locality of interconnection networks in FSMs or regular structures of logic gates for scalable precision in adders. Further, for reconfigurable architectures with a large number of configuration bits a separate family of standard cells optimized for low power must be used to achieve ultra-low power operation. These aspects of reconfigurability are discussed in detail in the subsequent chapters.

Chapter 3

Design Considerations in Power-Gated Circuits

3.1 Introduction

Power gating is an invasive low power technique. Inserting a power gating structure on-chip has a design cost that must be weighed against the benefits derived with respect to power consumption. In the first half of this chapter, existing work on design considerations in power-gated circuits is reviewed. Hence gate-level models for some of the design parameters of cluster-based power-gated circuits that forms one of the contributions of this work, is presented. The sources of leakage current in static CMOS circuits is briefly described in Section 3.2. Section 3.3 enumerates different power gating structures and ways of deploying them. The impact of power gating on some of the design parameters are explained along with an overview of related previous work. An example to illustrate the design parameters of interest is also given. In Section 3.4, semiempirical models for estimation of wakeup overheads are proposed. Further, logic clustering for wakeup scheduling as an application of wakeup time estimation is discussed. Section 3.5 concludes the chapter.

3.2 Leakage Currents in MOS Devices

Leakage currents exist in MOS devices due to various reasons. A detailed survey of different leakage current mechanisms in MOSFETs is presented in [14]. An important source of leakage current in MOSFETs is the subthreshold or weak inversion current that exists when the applied gate voltage is below threshold voltage ($|V_{gs}| \leq |V_{th}|$) and a bias exists across the drain-source channel ($|V_{ds}| \geq 0.1$V). Under such bias conditions, a conducting path between power supply V_{dd} and ground facilitates flow of leakage current in CMOS circuits. In short channel devices (channel length $L \leq 90$nm and oxide thickness

$t_{ox} \leq 2$nm), the effect of subthreshold leakage current is aggravated by Drain Induced Barrier Lowering (DIBL) leading to increased leakage current during standby states [26]. The scaling of MOS devices for constant field requires that the supply voltage also be scaled. The on-state current $I_{on} = I_{ds}|_{V_{gs}=V_{dd}}$ is given by the alpha power law model [68],

$$I_{on} = \mu_{eff} C_{ox} \frac{W}{L} (V_{dd} - V_{th})^{\alpha} \tag{3.1}$$

where μ_{eff} is effective carrier mobility, C_{ox} is oxide capacitance and W/L is the device ratio. From the simplest Pao-Sah model, $\alpha = 2$. It is therefore obvious that V_{th} needs to be reduced for higher I_{on} and lower delay. From the model for subthreshold current given by

$$I_{ds,sub} = \mu_{eff} C_{ox} \frac{W}{L} (m-1) V_T^2 e^{\frac{(V_{gs}-V_{th})}{mV_T}} \left(1 - e^{\frac{-V_{ds}}{V_T}}\right) \tag{3.2}$$

where V_T is thermal voltage kT/q and $m = 1 + \frac{3t_{ox}}{W_{dm}}$ is the body effect coefficient, W_{dm} is the maximum depletion layer width, the off-state current $I_{off} = I_{ds}|_{V_{gs}=0, V_{ds}=V_{dd}}$ can be determined as

$$I_{off} = \mu_{eff} C_{ox} \frac{W}{L} (m-1) V_T^2 e^{\frac{-V_{th}}{mV_T}}. \tag{3.3}$$

Optimizing different device parameters in a linear scaling regime does not offset an exponential increase in I_{off} due to a lower V_{th} for better performance. The subthreshold current density defined as the subthreshold leakage current per unit area of a MOS transistor and given by $J_{ds,sub} = \frac{I_{ds,sub}}{WL}$ is proportional to $1/L^2$. Clearly, for the same total integration area of MOS transistors, the total subthreshold leakage current is significantly higher in an advanced technology (smaller L) node. Table 3.1 shows the subthreshold leakage current density of a PMOS transistor in three sub-65nm technology nodes as obtained from the predictive technology models of Arizona State University [69]. Further, from $\frac{\partial I_{ds,sub}}{\partial T} = \frac{\partial I_{ds,sub}}{\partial V_{th}} \frac{\partial V_{th}}{\partial T}$, it can be shown that

$$\frac{\partial I_{ds,sub}}{\partial T} = -\frac{I_{ds,sub}}{mV_T}(-K_T) \tag{3.4}$$

for some positive K_T [26], thus implying that leakage current increases with temperature as shown in Table 3.1.

Technology Node	45nm ($V_{dd} = 1V$)		32nm ($V_{dd} = 1V$)		22nm ($V_{dd} = 0.9V$)	
Temperature	25°C	100°C	25°C	100°C	25°C	100°C
$J_{ds,sub}(nA/\mu m^2)$	0.968	8.114	3.440	25.156	14.056	84.391
Relative Increase	1	1	3.55	3.10	14.52	10.40

TABLE 3.1: Leakage current density in nanoscale PMOS transistors.

Other sources of leakage current include Gate Induced Drain Leakage (GIDL) proportional to $e^{-t_{ox}}$, oxide tunneling current and leakage current due to hot carrier injection. In nanoscale devices, the effects of leakage current are particularly important because at a certain level of circuit complexity, they dominate dynamic power consumption. Methods to reduce leakage currents have also been described in [14] at both process and circuit levels. In this work only bias-dependent leakage current is considered as it is the dominant source of leakage current.

3.3 Power Gating

3.3.1 Sleep Devices and Power Gating Networks

Sleep devices are a fundamental part of power gating network. A high-V_{th} PMOS transistor (header switch) between V_{dd} and Virtual-V_{dd}(V_{Vdd}) supply lines or an NMOS transistor (footer switch) between the Gnd (or V_{ss} in general) and Virtual-V_{Gnd}($V_{V_{Gnd}}$) rails with a control signal constitutes a power gating circuit in its simplest form. It is possible to use both header and footer switches to power-gate the logic circuit. The merits and demerits of each basic form is enumerated in [34]. The purpose of sleep transistors is to provide a very high ohmic connection between the supply/ground rails and the gated logic during standby modes. In general, the logic between V_{Vdd} and V_{Gnd}/V_{Vss} is implemented with standard cells of low-V_{th} (including std.-V_{th}) transistors.

A set of sleep devices, isolation cells, associated control signals and virtual supply/-ground rails with a specific placement and routing structure constitutes a power gating network. Various structures for power gating networks have been proposed. An important criterion for classification of power gating networks is the granularity of design at which sleep transistors are inserted. A number of such topologies have been proposed ranging from fine to coarse granularities. In fine grained power gating, each library cell as small as a basic logic gate has a sleep transistor, typically of high threshold voltage. These cells are referred to as multi-threshold CMOS (MTCMOS) circuits. A weak pull-up/pull-down transistor controlled by a sleep signal is also added to avoid floating outputs when the cell is in the sleep mode. While the virtual power nets are short and simple to route, this topology leads to an increase in area by a factor of 2 to 4 [70]. Apart from inflicting a high area penalty, the power gating network becomes extremely sensitive to Process-Voltage-Temperature (PVT) variations. A problem with this approach is that sneak currents flow along paths between supply and ground during sleep mode, and such paths may exist through a set of OFF devices or a combination of devices in ON and OFF states.

In contrast, coarse grained implementations have sleep transistors connected between power supply networks and logic clusters. The sleep transistors can share charge and

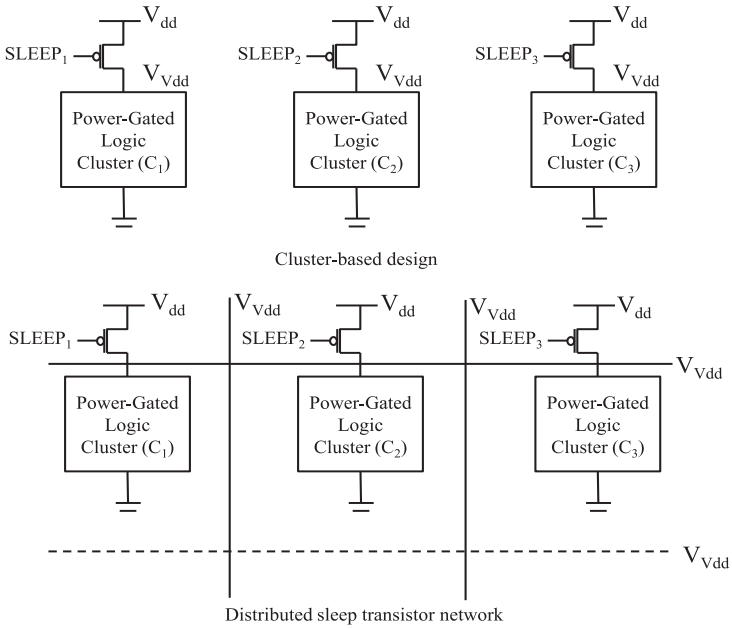


FIGURE 3.1: Cluster-based power gating and distributed sleep transistor network.

discharge currents with a low area overhead. Different forms of coarse grained power gating networks have been proposed. In a cluster-based approach [71] a sleep transistor is provided for an ensemble of logic gates whereas in a Distributed Sleep Transistor Network (DSTN), several sleep transistors are provided for power gating of a block of clusters [72], referred to as a module, with a common virtual supply/ground node. A schematic representation of the two types of coarse-grained power gating is shown in Fig. 3.1. The virtual supply or ground nets of each cluster are connected together in a module to reduce the maximum instantaneous current (MIC) in active mode of operation. However DSTN is not suitable when independent clusters have to be isolated from other clusters or always-on blocks as opposed to a cluster based design. Further, power-gated logic clusters are simple to implement compared to a DSTN. In this thesis we consider cluster-based design as the primary topology for all our experiments and design methodology. A schematic representation of a power-gated logic cluster with header type of sleep transistor is shown in Fig. 3.2.

3.3.2 Design Parameters

While sleep transistors are conceptually simple, optimal design of sleep transistors and power gating networks is challenging [73]. Some of the design problems addressed so far have been sleep transistor sizing [71], selection of appropriate power gating topology [72, 71], estimation and control of in-rush current and delay degradation in power-gated circuits [74]. Other design considerations that have been studied are effects of

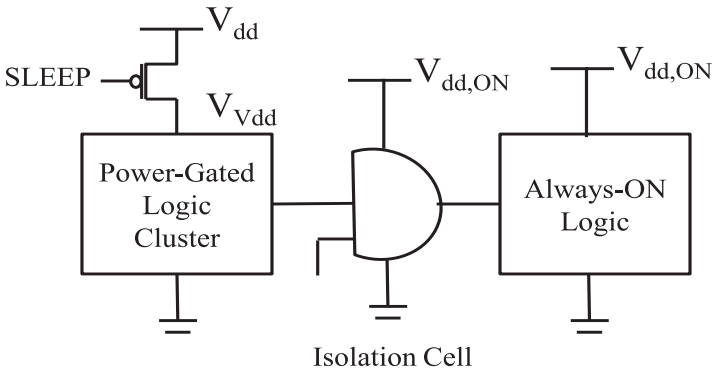


FIGURE 3.2: A power-gated logic cluster with header type of sleep transistor and isolation cell.

supply/ground bounce on reliability of power grid [75], design of isolation cells and state retention registers, and automated synthesis of power gated circuits [76, 77].

Sleep transistor width W of a power-gated logic cluster is a critical design parameter as it impacts both area due to power gating and circuit performance due to delay degradation. In active mode when inputs to the logic gates change switching currents are generated depending on the number and types of gates that switch outputs. The maximum current that is generated in the worst case is referred to as maximum instantaneous current (MIC). In order to minimize delay degradation of logic cluster the sleep transistor must be able to source (or sink) the MIC. Therefore the size of sleep transistor W should be large enough to balance the MIC and have minimum IR drop across the sleep transistor in active mode of operation. Let $V_{st} = I_{st}R_{st}$ be the IR drop across the sleep transistor in active mode with $I_{st} = MIC$, the current through the sleep transistor and R_{st} the channel resistance. It is shown in [71] that the propagation delay of a power gated logic circuit increases to

$$T_{d,pg} = \frac{K_d C_L V_{dd}}{(V_{dd} - V_{st} - V_{th})^{\alpha}} \tag{3.5}$$

from

$$T_d = \frac{K_d C_L V_{dd}}{(V_{dd} - V_{th})^{\alpha}} \tag{3.6}$$

of an non power-gated circuit resulting in a performance loss (δ) given by

$$\delta = 1 - \frac{T_d}{T_{d,pg}} \tag{3.7}$$

where K_d is some positive constant. The sleep transistor area is then calculated from its width given by

$$\frac{W}{L} = \frac{I_{st}}{\delta \mu_n C_{ox} (V_{dd} - V_{th})^2} \tag{3.8}$$

Typically, V_{st} is specified as a constraint, determined by the allowable delay degradation and hence I_{st} is estimated. This leads to a minimum required size $(W/L)_{min}$ for the sleep transistor.

Another important consideration in the design of power gated circuits is supply and ground bounce noise as a result of in-rush current. During wakeup mode transitions instantaneous current passing through the sleep transistor creates current surges resulting in inductive $L(di/dt)$ voltage drops along the power rails. This current is due to the saturation current I_{on} of sleep transistor that has a quadratic relationship with V_{ds} and the charging current due to capacitances of the switching gates getting charged as the voltage across them increases from a steady-state sleep mode value to a steady-state active mode value. Hence, if the magnitude of such voltage drops on power rails is greater than the noise margin of a circuit, erroneous values may be latched in adjacent always-on logic as shown in Fig. 3.2. Further it causes reliability issues due to electromigration on power rails. Clearly, this leads to a requirement of maximum transistor size $(W/L)_{max}$ to limit current I_{max} drawn at wakeup. The logic gates have to be partitioned and clustered in such a way that $(W/L)_{min} \leq (W/L) \leq (W/L)_{max}$ while satisfying MIC, I_{max} and δ.

In [75], methods to reduce ground bounce noise is presented along with techniques to stabilize data-retention voltage of power gating structures. In the first approach proposed, V_{gs} and V_{ds} of NMOS sleep transistor are controlled until V_{ds} is significantly reduced, when V_{gs} is increased in non-uniform steps to turn on the transistor. the technique is referred to as 'staggered turn-on' and it requires a complex control circuitry. In a similar approach, the effective transistor size is changed dynamically with $V_{gs} = 0$ until its V_{ds} is significantly reduced. In the third approach proposed, two PMOS sleep transistors are stacked between V_{dd} and V_{Vdd} with a metal-to-metal capacitor between them and are turned ON/OFF with pseudorandom pulses to reduce the ground bounce noise. However, greater the number of transistors stacked between main and virtual supply rails, higher would be the IR drop across sleep transistors during active mode [34] and higher the delay degradation of the circuit.

SoC design with coarse-grained power gating has largely remained a semi-custom design approach because of custom nature of circuit blocks. Attempts have been made in [76] to adapt physical design based on standard-cell methodology to power gating structures by creating specific power gating cells. In [77] a high-level synthesis flow for power gated circuits has been proposed with focus on minimizing the size of retention storage for a typical SoC-like architecture. A complete framework for power gating that covers scheduling, allocation, controller synthesis, timing closure, and placement and routing is described. A practical approach to power gating is provided in [34] with design recommendations at different levels of SoC design using a technology demonstrator chip (SALT: Synopsys-ARM Low Power Technology) designed in 90nm process technology. A framework for power gating functional units in datapaths of embedded processor

architectures to avoid leakage reduction at compiler level is proposed in [78]. A sleep transistor insertion methodology for row-based power gating has been proposed in [79].

Most approaches to coarse grained power gating have been directed towards partitioning the logic to be power-gated into several clusters to restrict the in-rush current. After defining a power gating topology, other design parameters like sleep transistor size are determined with constraints of peak MIC of the cluster, allowable IR drop across the sleep transistor and tolerable wakeup time. Wakeup time refers to the time taken by the virtual supply/ground rail to reach a steady state value at the onset of active mode from its value in sleep mode. In most designs the design parameters are inter-related as discussed above and therefore require to be optimized in a combined way. A comprehensive analysis and control of design parameters for power gated circuits is presented in [80]. The work analyzes relationships between wakeup time, peak current and sleep transistor area for different sizes of cluster and their wake-up schedule. The above design problem is solved by formulating it into repeated applications of finding a maximal clique in a graph. However the approach involves long SPICE simulations in the absence of compact models for rapid parameter estimation.

3.3.3 Power-Gating Example

In this section a 32-bit array multiplier given by the ISCAS85 benchmark circuit [81] c6288 is used to demonstrate energy savings due to power gating. In order to evaluate different design parameters, the multiplier was synthesized with a set {nand2, nor2, inv, xor2} of low-V_{th} gates from an industrial 65nm technology library. The low-V_{th} gates have very high leakage current compared to higher threshold voltage gates and are generally used for high speed logic paths. The SPICE netlists of resulting gate netlists were obtained and a high-V_{th} sleep transistor of PMOS type was inserted in the power gating network of multiplier. The sleep transistor size was set to W=12μm without loss of generality. Sleep transistor sizing is generally seen as an optimization problem based on in-rush current and IR drop, but it is not addressed in this example. A netlist of c6288 without the sleep transistor was used as a reference for a circuit without power gating.

In order to estimate the average power consumption of both circuits, 10000 input vectors randomly generated with uniform distribution were applied to the multiplier at an interval of 3ns and was simulated using Eldo SPICE simulator at a nominal supply voltage of 1V and 100°C temperature. Transient analysis was carried out for a total time of 70μs without the loss of generality. The ungated multiplier remained in standby state for a time of 40μs whereas in the power-gated case, power gating was applied for 10μs and it continued in idle state (powered-on, but without changes in inputs) for the remaining time. The current drawn by the power-gated multiplier is shown in Fig. 3.3 and Fig. 3.4.

The virtual supply voltage in active, wakeup and sleep modes are shown in Fig. 3.5 and Fig. 3.6 respectively. Table 3.2 shows results obtained from circuit simulation of the two multipliers. $P_{total,av}$ denotes the average power across a time interval of [0, 70μs]. $P_{idle,av}$ and $P_{sleep,av}$ denote average power in idle state and sleep mode respectively.

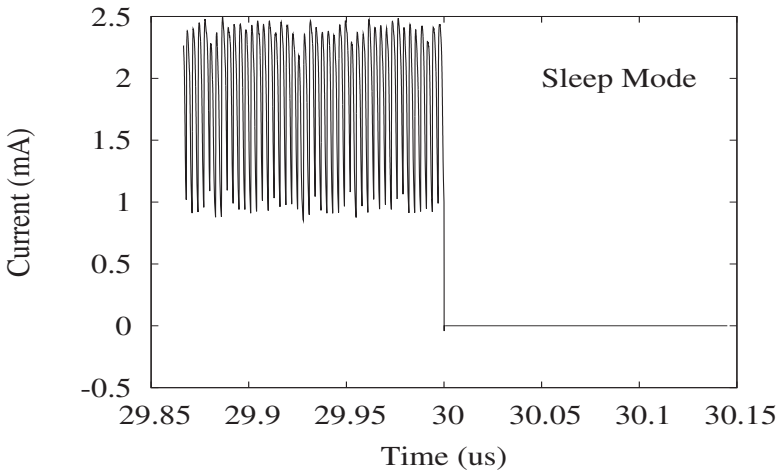


FIGURE 3.3: Active mode and sleep mode current in power-gated c6288 (W=12μm).

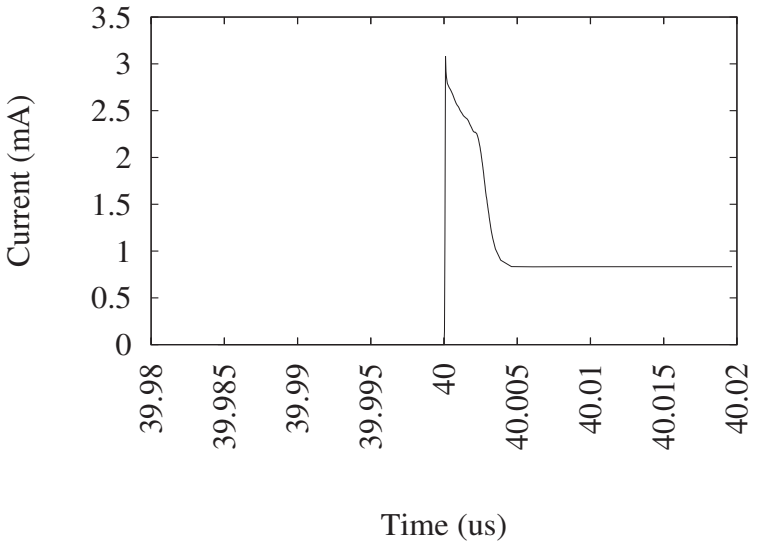


FIGURE 3.4: Wakeup mode current in power-gated c6288 (W=12μm)

c6288	$P_{idle,av}$ (μW)	$P_{sleep,av}$ (μW)
Ungated case	1100	-
Power-gated case	833	0.19

TABLE 3.2: Power consumption in ungated and power-gated array multiplier (c6288).

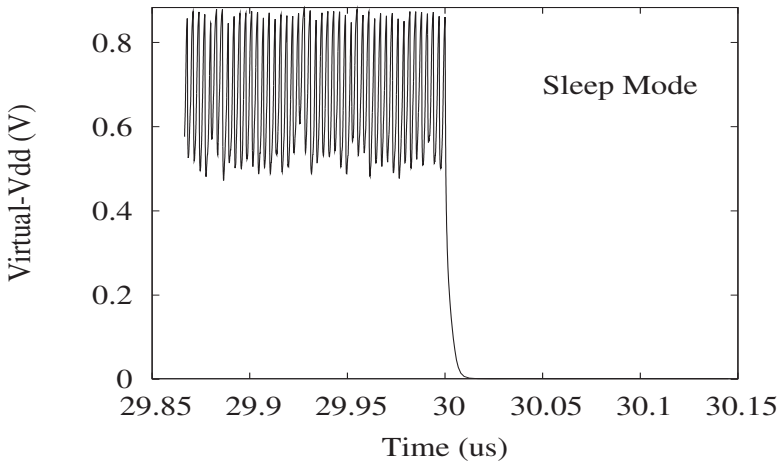


FIGURE 3.5: Active mode and sleep mode Virtual-Vdd in power-gated c6288 (W=12μm).

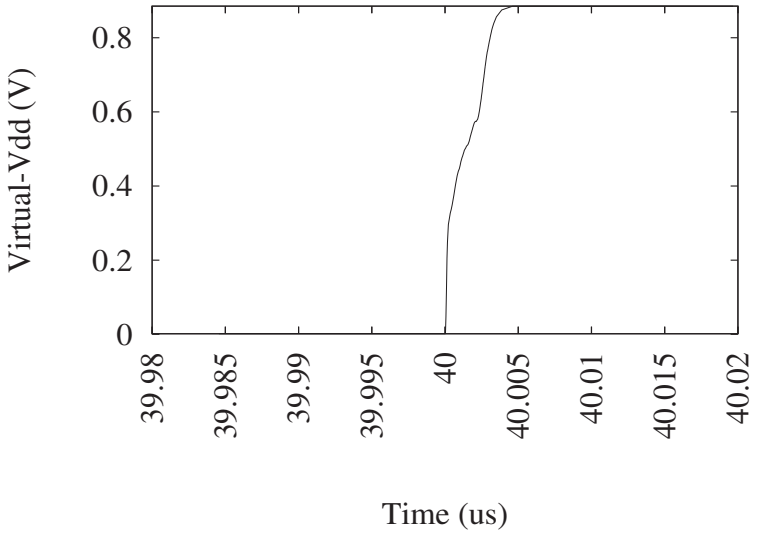


FIGURE 3.6: Wakeup mode Virtual-Vdd in power-gated c6288 (W=12μm).

The average power of the multiplier in active mode only can be determined as

$$P_{av,active} = \left[\frac{P_{av,total}T - (P_{av,idle}T_{idle} + P_{av,sleep}T_{sleep})}{T_{active}} \right]. \tag{3.9}$$

It can be inferred that the power consumption of circuit in sleep mode is significantly less than that in idle state resulting in power savings due to power gating. Further a significant drop in V_{Vdd} in steady state to 0.85V from V_{Vdd} at 1V is observed. This is on account of both IR drop in the sleep transistor and the leakage current in the logic circuit in active mode and idle state. In the active mode, V_{Vdd} drops additionally by about 250mV due to short-circuit currents in the logic gates. With a logic circuit consisting of higher threshold voltage cells the drop in V_{Vdd} from V_{dd} is lower due to

lesser leakage current. This is discussed in detail in Section 3.4.

3.4 Models for Estimation of Wakeup Time and Wakeup Energy

Wakeup time estimation is fundamental to logic clustering algorithms for cluster-based power gating design that require constraints of peak current and wakeup time to be satisfied [82, 71, 80]. In this context, a simple analytical model for estimation of wakeup time is useful especially when it needs to be determined iteratively during an optimization run for a number of candidate clusters. The problem of wakeup time estimation arises in other scenarios as well. In [75] and [83], a need for wakeup latency estimation arises to quantify the effectiveness of proposed ground bounce reducing techniques and intermediate strength power gating techniques respectively under a wakeup time constraint. In [84], Xu *et al.,* have proposed a numerical approach for estimation of V_{Gnd} as a function of time in sleep mode. To extend the same method to wakeup mode, it is necessary to incorporate size dependent sleep transistor current characteristics. In this case an analytical model for V_{Gnd} would be highly desirable. This analysis identically applies to Virtual-Vdd (V_{Vdd}) in a power-gated cluster with a header type of sleep transistor. In [85] the role of wakeup latency in state-retentive power gating of register files in multicore processors is studied. Most works have used time consuming SPICE simulations [80] or constant current source model [83] for sleep transistors to determine wakeup time.

In the subsequent sections, a model for V_{Vdd} based on polynomial representation of leakage current in a logic cluster and linear region resistance of sleep transistor is derived. A method to estimate steady-state Virtual-Vdd after wakeup mode using leakage current profiles of constituent logic gates is described. Hence, a closed-form expression is derived for estimation of wakeup time of the power-gated circuit. Further, this can be extended to estimate wakeup energy. The model for Virtual-Vdd in sleep mode can be used to determine energy savings due to power gating. In other words, some of the key design parameters have been captured in a single model [86, 60].

3.4.1 Power-Gated Circuit Operation

A power gating structure cuts-off bias voltages for MOS devices so that bias-dependent leakage current in the logic circuit reduces significantly in standby state. A simple power-gated circuit shown in Fig. 3.7 and used in this work consists of a high-V_{th} PMOS sleep transistor connected between power supply rail (V_{dd}) and virtual power supply node, Virtual-Vdd (V_{Vdd}) of the logic cluster. A cluster refers to an ensemble of connected logic gates power-gated by a sleep transistor. The gate terminal of the sleep transistor

is connected to a control signal $SLEEP$, to switch the sleep transistor between ON and OFF states. A power-gated circuit operates in three modes in a typical power gating cycle as shown in Fig. 3.8. When $SLEEP$ is high, power supply to the logic is cutoff; $|V_{gs}| < |V_{th}|$ and V_{Vdd} decreases. The circuit is said to be in sleep mode. The leakage current decreases exponentially with V_{Vdd} resulting in energy savings. When $SLEEP$ is low, $|V_{gs}| > |V_{th}|$ and therefore current flows through the sleep transistor to charge circuit capacitances. Due to charging effect, V_{Vdd} increases until it reaches a steady state value less than V_{dd} due to IR drop across the sleep transistor. The sleep transistor is typically in saturation region at wakeup and moves to linear region as $V_{ds}(t) = V_{dd} - V_{Vdd}(t)$ decreases. This mode of operation is referred to as wakeup mode and the mode of operation after wakeup is referred to as active mode in this thesis.

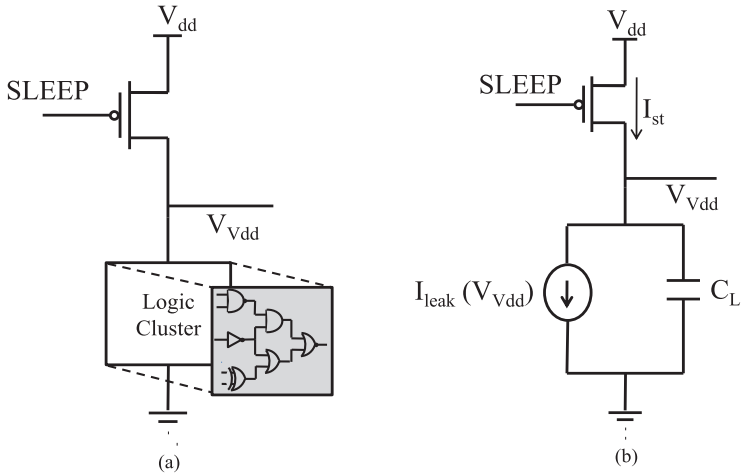


FIGURE 3.7: (a) Power-gated logic cluster of header type. (b) Equivalent circuit of logic cluster.

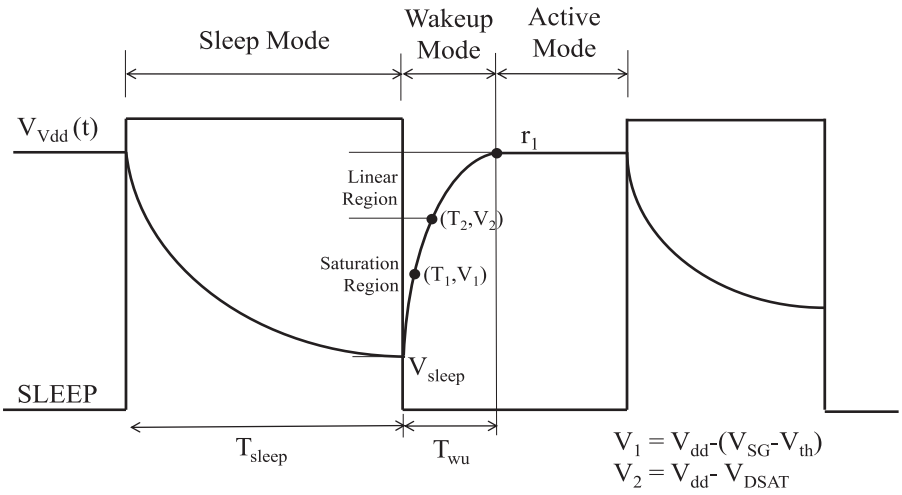


FIGURE 3.8: Typical timing instants and modes of operation in a power gating cycle.

3.4.2 Power-Gated Logic Cluster Model

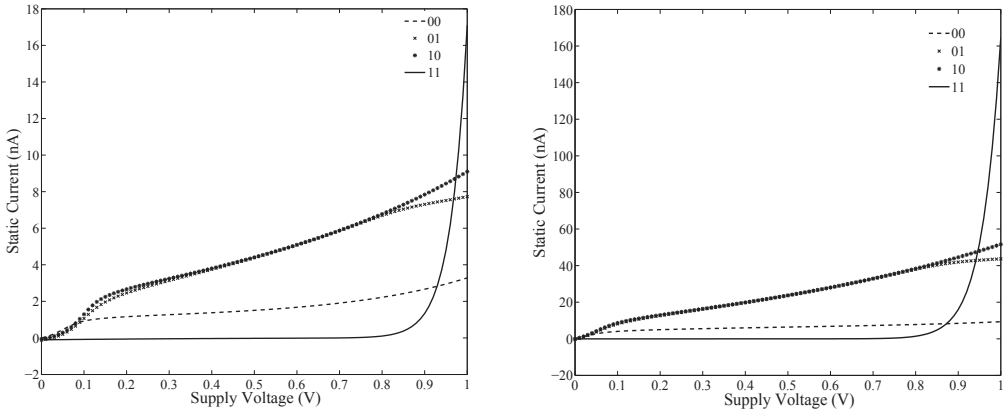
Models for subthreshold leakage current that capture its exponential behaviour with bias voltages at device level have been described in [26]. In [84], compact models for leakage current have been derived at gate and circuit levels in a hierarchical way. It was shown that the leakage current can be represented by a voltage controlled current source (VCCS) as in Fig. 3.7(b). In this work, we take a polynomial based approach to derive leakage current profile for the complete circuit. For each type of cell S_i and input pattern j, leakage current is determined at several voltages and the resulting profile is fitted with a polynomial of degree N in V_{Vdd} as given by

$$I_{leak}(S_i, j) = \sum_{k=0}^{N} b_k(S_i, j) V_{Vdd}^k \tag{3.10}$$

where $\{b_k(S_i, j)\}$ represents coefficients of the polynomial. We assume a standard-cell based design approach for implementation of the cluster. The static current profiles for a single 2-input NAND gate of high-V_{th} and std.-V_{th}[1]devices is shown in Fig. 3.9 for all input patterns of the gate. Therefore, the total static current for $n(S_i, j)$ occurrences of each cell and each input pattern is obtained as

$$I_{leak} = \sum_{i=0}^{P-1} \sum_{j=0}^{R_i-1} n(S_i, j) I_{leak}(S_i, j) \tag{3.11}$$

where P and R_i are number of types of cells and number of possible input combinations for cell S_i respectively. As an example, if a logic cluster is composed of a set $S = \{$nand2,



High-V_{th} Cell Std.-V_{th} Cell

FIGURE 3.9: Static current profile of a 2-input NAND gate of high-V_{th} and std.-V_{th} devices for all input patterns.

[1]An industrial 65nm library of with three families of logic gates, with high-V_{th}(HVT), std.-V_{th}(SVT) and low-V_{th}(LVT) devices is used. LVT devices have highest leakage current and HVT devices have the lowest leakage current.

nor2, inv, xor2} of gates, then $P = 4$. For a 2-input NAND gate $R_i = 4$, whereas for an inverter, $R_i = 2$. For notational simplicity, the total leakage current profile of the logic cluster is represented by

$$I_{leak} = \sum_{i=0}^{N} b_i V_{Vdd}^i \tag{3.12}$$

in the rest of the chapter. Equation (3.12) has the form of nonlinear resistance. The physical units of b_i can be derived to be A/V^i. A limitation of this behavioral model is that process and device parameters are not explicitly captured but it is sufficiently simple to model design parameters at the level of a logic cluster.

The total capacitance of the logic cluster is derived as the sum of capacitances of all the inputs of all constituent standard cells.

$$C_L = \sum_{i=0}^{P-1} n(S_i) \sum_{l=0}^{R_i-1} C_{il} \tag{3.13}$$

In physical implementations with CMOS technologies, a decoupling capacitance (decap) C_D is generally included between the supply voltage rail or Virtual-Vdd node of the power-gated domain and ground to suppress bounces on supply rails during switching of gate outputs. In the model described in this work, a decap is not explicitly included. For the latter case the total circuit capacitance including a decoupling capacitance would be $C_L + C_D$ since capacitances appear in parallel between Virtual-Vdd and ground.

The gate capacitances of MOS transistors are input dependent. At wakeup, as V_{Vdd} increases some of the logic gates switch to logic 1 while the rest remain at logic 0. Gate inputs in the fan-out of gate outputs that switch to logic 1 will present a higher output capacitance to the switching gate than to the driving gate remaining at logic 0. In the logic clusters used here, the outputs of each gate is determined from the primary inputs based on the gate function (NAND, XOR etc.) and hence appropriate value of capacitance obtained from SPICE characterization of standard cell for each of its input is used to determine total capacitance in Eq. (3.13). Further gate terminal capacitances of all MOSFETs in standard cells include parasitic capacitances (fringe and overlap) referred to the gate terminal.

Parasitic capacitances along interconnect lines have been neglected in determining total circuit capacitance considering that in cluster based power-gated circuit design, independent clusters have a local distribution of interconnects unlike a distributed sleep transistor network (DSTN) based power gating.

3.4.3 Virtual-Vdd Model

3.4.3.1 Determination of Steady-State Virtual-Vdd Voltage

Consider the equivalent circuit model in Fig. 3.7(b). In the wakeup mode, the operating point on the I_{sd} vs. V_{sd} characteristics of sleep transistor moves from saturation region to linear region until V_{Vdd} reaches a steady-state value. The virtual supply node is said to be in steady state when $dV_{Vdd}/dt = 0$, *i.e.*, when there are no changes in V_{Vdd} either on account of short circuit currents due to changing logic states of internal nodes or due to charging effect. Let the current through the sleep transistor during wakeup and in non-saturation region be denoted by $I_{st,ns}$, the total leakage current at the output of VCCS by I_{leak} and the capacitive load charging current by I_{load}. Then,

$$I_{st,ns} = I_{leak} + I_{load}. \tag{3.14}$$

The current through the sleep transistor in non-saturation region is given by the quadratic model

$$I_{st,ns}(t) = \frac{1}{R_{lin}}\left[(V_{dd} - V_{Vdd}(t)) - \frac{(V_{dd} - V_{Vdd}(t))^2}{2(V_{dd} - V_{th})}\right] \tag{3.15}$$

where R_{lin} is the resistance in linear region. The determination of R_{lin} is described in subsection 3.4.3.4. From Eq. (3.12), Eq. (3.15) and $I_{load} = C_L(dV_{Vdd}/dt)$, Eq. (3.14) becomes

$$\frac{dV_{Vdd}}{dt} = -\frac{1}{\tau}\sum_{i=0}^{N} c_i V_{Vdd}^i \tag{3.16}$$

where $\tau = R_{lin}C_L$ and $c_i = f_i(V_{dd}, R_{lin}, b_i, V_{th})$ are expressions derived from Eq. (3.12) - Eq. (3.15). Although threshold voltage depends on applied bias voltages, a nominal V_{th} is assumed as it does not lead to significant loss of accuracy. To solve for V_{Vdd} in Eq. (3.14), a numerical procedure to find the roots is required. However to obtain solutions in closed form, the Nth degree polynomial in Eq. (3.16) is reduced to a quadratic polynomial by least-squares approximation and is expressed in terms of its roots r_1 and r_2 as

$$\frac{dV_{Vdd}}{dt} = -\frac{1}{\tau}(V_{Vdd} - r_1)(V_{Vdd} - r_2). \tag{3.17}$$

Both r_1 and r_2 are steady state points of Eq. (3.17). One of the roots r_1 satisfying the interval of validity $V_{sleep} < r_1 < V_{dd}$, is determined to be the steady state Virtual-Vdd voltage. Here V_{sleep} denotes the value of V_{Vdd} at the wakeup transition.

In a RC circuit the steady state as defined above is reached at $t = \infty$. However the error in assuming value of V_{Vdd} at onset of active mode to be r_1 is negligible as demonstrated in Section 3.4.4. The conditions of validity for one of the roots r_1 can

be intuitively explained to be $V_{sleep} < r_1 < V_{dd}$ as follows. For each value of V_{Vdd}, the VCCS outputs a current given by Eq. (3.12). Therefore for each value of V_{Vdd} it can be inferred that the resistance of circuit is given by

$$R_s(V_{Vdd}) = \frac{V_{Vdd}}{\sum_{i=0}^{N} b_i V_{Vdd}^i}. \tag{3.18}$$

We refer to R_s as pseudo-resistance in the rest of the chapter [20]. Let at steady state, the pseudo-resistance of VCCS be given by some $R_{ss} = V_{Vdd,ss}/I_{leak}(V_{Vdd,ss})$ where $V_{Vdd,ss}$ denotes Virtual-Vdd in steady state. Then the circuit at that instant can be represented by Thévenin's equivalent resistance $R_{TH} = \frac{R_{ss}R_{lin}}{(R_{ss}+R_{lin})}$ and Thévenin's equivalent voltage $V_{TH} = \frac{R_{ss}V_{dd}}{(R_{ss}+R_{lin})}$ with total circuit capacitance C_L in series with R_{TH} and V_{TH}. Clearly, $V_{TH} < V_{dd}$. C_L is charged to V_{TH} in steady state which we determine to be r_1 as a solution of Eq. (3.17). For non-zero C_L the inference that $V_{sleep} < r_1$ is trivial.

3.4.3.2 Wakeup Time Estimation

In order to obtain a model for $V_{Vdd}(t)$ in wakeup mode, the ordinary differential equation in Eq. (3.17) is solved in the non-saturation region and hence, is extended to saturation region by means of piecewise approximations. Let at time $t = 0$ the operating point move to non-saturation region so that $V_{Vdd}(0) = V_{initial}$. The solution of Eq. (3.17) satisfying the interval of validity and moving towards r_1 can be written as

$$[V_{Vdd}(t)]_{ns} = \frac{r_1 - r_2 K e^{-at}}{1 - K e^{-at}} \tag{3.19}$$

where $K = (V_{initial} - r_1)/(V_{initial} - r_2)$, $a = 1/A\tau$ and $A = 1/(r_1 - r_2)$. From Fig. 3.8, $V_{initial} = V_{dd} - V_{DSAT}$ where V_{DSAT} is the saturation voltage.

To extend the model to saturation region, the time instant $t = 0$ is moved to sleep-to-wakeup mode transition so that $V_{Vdd}(0) = V_{sleep}$. Let T_{wu} denote the wakeup time defined as the time taken for V_{Vdd} to evolve from V_{sleep} to $0.99r_1$. Further, let V_1 and V_2 be two voltage levels attained by V_{Vdd} at T_1 and T_2 respectively as shown in Fig. 3.8. The solution Eq. (3.19) does not represent V_{Vdd} in the saturation region, $V_{Vdd} < V_2$ accurately. Therefore corrections are applied to Eq. (3.14) in the first two segments as

$$I_{st}(t) = I_{leak} + I_{load} - \Delta I_0(t) + \Delta I_1(t). \tag{3.20}$$

In Eq. (3.20) the time instant $t = 0$ corresponds to sleep-to-wakeup mode transition and $V_{initial} = V_{sleep}$. Let U_T denote the time shifted unit step function $u(t - T)$. We define

$$\Delta I_0(t) = I_0 \left[U_0 e^{-at} - U_{T_1} e^{-a(t-T_1)} \right] \tag{3.21}$$

$$\Delta I_1(t) = I_1 \left[U_{T_1} e^{-a(t-T_1)} - U_{T_2} e^{-a(t-T_2)} \right] \tag{3.22}$$

based on heuristics for I_0 and I_1 described in Section 3.4.3.5. In the third interval Eq. (3.19) alone is satisfied and hence no correction is required. Using Eq. (3.19) to Eq. (3.22), the model for Virtual-Vdd in wakeup mode can be derived as

$$\begin{aligned} V_{Vdd}(t) = {} & \frac{r_1 - r_2 K e^{-at}}{1 - K e^{-at}} + A I_0 R_{lin} \left[U_0 \left(1 - e^{-at} \right) \right] \\ & - A R_{lin} (I_0 + I_1) \left[U_{T_1} \left(1 - e^{-a(t-T_1)} \right) \right] \\ & + A R_{lin} I_1 \left[U_{T_2} \left(1 - e^{-a(t-T_2)} \right) \right]. \end{aligned} \tag{3.23}$$

In compact form, Eq. (3.23) can be written as $Xe^{-2at} + Ye^{-at} + Z = 0$. The solution for t is given by

$$t = \frac{1}{a} \ln \left(\frac{2X}{-Y - \sqrt{Y^2 - 4XZ}} \right). \tag{3.24}$$

At $t = T_1$, $V_{Vdd} = V_{dd} - V_{sg} + V_{th}$ corresponding to the criterion $V_{sd} = V_{sg} - V_{th}$. Applying this condition and $V_{sg} = V_{dd}$, X, Y and Z are determined to be

$$\begin{cases} X = A R_{lin} K I_0, \\ Y = K(V_{th} - r_2) - (1 + K) X/K, \\ Z = A I_0 R_{lin} + r_1 - V_{th}. \end{cases} \tag{3.25}$$

Similarly for $t = T_2$, $V_{Vdd} = V_{dd} - V_{DSAT}$ corresponding to the condition $V_{sd} = V_{DSAT}$, which gives

$$\begin{cases} X = -A R_{lin} K [I_0 (e^{aT_1} - 1) + I_1 e^{aT_1}], \\ Y = K(V_{dd} - V_{DSAT} - r_2) + (X/K) + A R_{lin} I_1 K, \\ Z = -A R_{lin} I_1 + r_1 - V_{dd} + V_{DSAT}. \end{cases} \tag{3.26}$$

For wakeup time T_{wu}, $V_{Vdd} = 0.99 r_1$, which gives

$$\begin{cases} X = A R_{lin} K [-I_0 (e^{aT_1} - 1) + I_1 (e^{aT_2} - e^{aT_1})], \\ Y = K(0.99 r_1 - r_2) - (X/K), \\ Z = 0.01 r_1. \end{cases} \tag{3.27}$$

The values of r_1 and r_2 are unaffected due to ΔI_0 and ΔI_1 as Eq. (3.23) satisfies $V_{sd} = V_{DSAT}$ at $t = T_2$ as shown in Fig. 3.8. For clusters with $0.99 r_1 \leq (V_{dd} - V_{DSAT})$, wakeup time $T_{wu} = T_2$ determined from Eq. (3.26) and with the condition $V_{Vdd} = 0.99 r_1$.

3.4.3.3 Sleep Mode Virtual-Vdd Model

To calculate T_1, T_2, and T_{wu} using Eq. (3.25) to Eq. (3.27), it is necessary to determine V_{sleep}. If the cluster is in sleep state for a time interval T_{sleep}, then $V_{sleep} = V_{Vdd}(T_{sleep})$.

It should be noted that for simplicity both mode transitions are assumed to occur at $t = 0$, so that the initial condition for sleep mode can be denoted by $V_{Vdd}(0)$ as for wakeup mode. In sleep mode, the sleep transistor is cut-off so that only a leakage current $I_{st,leak}$ flows through it. I_{load} in Eq. (3.14) is now a discharging current. Hence Eq. (3.14) for sleep mode can be written as

$$-C_L\frac{dV_{Vdd}}{dt} = -(b_0 - I_{st,leak}) - \sum_{i=1}^{N} b_i V_{Vdd}^i. \tag{3.28}$$

Neglecting $I_{st,leak}$ and rewriting Eq. (3.28) similar to Eq. (3.16),

$$\frac{dV_{Vdd}}{dt} = -\frac{1}{R_s(V_{Vdd})C_L}\left[-R_s(V_{Vdd})\sum_{i=0}^{N} b_i V_{Vdd}^i\right]. \tag{3.29}$$

A numerical solution to Eq. (3.29) is of the form [84]

$$V_{Vdd,j+1} = V_{Vdd,j}e^{-\frac{\Delta t}{R_s(V_{Vdd,j})C_L}} \tag{3.30}$$

where j denotes a time interval in $[0,\ T_{sleep}]$ of size Δt. To develop an approximation, we consider a heuristic for choice of R_s as explained in Section 3.4.3.5. Denoting R_{sp} as the pseudo-resistance chosen by applying the heuristic, the model for Virtual-Vdd in sleep mode can be derived as

$$\frac{dV_{Vdd}}{dt} = -\frac{1}{R_{sp}C_L}\prod_{i=1}^{N}(V_{Vdd} - r_i^s) = 0 \tag{3.31}$$

where r_i^s represents roots of the polynomial in sleep context. Let r_1^s satisfy $r_1 < r_1^s < 0$. Then the approximate solution that moves towards r_1^s from its initial value is given by

$$V_{Vdd}(t) = r_1^s + e^{-\frac{t}{R_{sp}C_L}+K^s}. \tag{3.32}$$

At the end of active mode, the value of Virtual-Vdd satisfies $r_1 - \Delta V_{Vdd,max} \leq V_{Vdd} \leq r_1$ where $\Delta V_{Vdd,max}$ is the maximum degradation of V_{Vdd} due to dynamically changing inputs of logic cluster. In this work, it is assumed that the power-gated logic cluster remains in active mode for a duration long enough with appropriate input conditions that $V_{Vdd} = r_1$ at the end of active mode. This assumption is mostly true in circuits with adequate positive timing slack and is justified with an experiment next. Let $\Delta V_{Vdd,max}$ denote maximum degradation of V_{Vdd} due to dynamically changing inputs of logic cluster. Further, let t_i denote the time instant of end of cycle i in active mode. Clearly, $V_{Vdd}(t_i)$ satisfies $V_{Vdd}(t_{i-1}) - \Delta V_{Vdd,max} \leq V_{Vdd}(t_i) \leq r_1$. We consider the conditions under which $V_{Vdd}(t_{i-1}) = r_1$. A logic cluster was synthesized for a maximum path delay of 1.0ns and was power-gated by a header type of sleep transistor. In practice, the size of

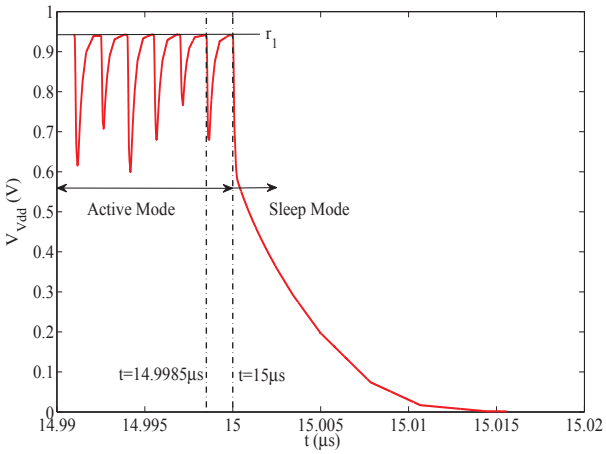


FIGURE 3.10: Virtual-Vdd in active and sleep modes.

the sleep transistor is chosen for a fixed performance loss or $\Delta V_{Vdd,max}$ in active mode. The circuit was simulated with Spectre circuit simulator with random inputs applied to the circuit at a clock period of 1.5ns, i.e., with a positive slack of 0.5ns in active mode. The variation of Virtual-Vdd voltage in active and sleep modes is shown in Fig. 3.10. It can be seen that for a maximum duration of path delay, V_{Vdd} degrades by about $\Delta V_{Vdd,max} = 0.3V$ and at the end of active mode time slot of 1.5ns, V_{Vdd} attains a value of r_1. With a sufficient and constant clock cycle period $T = t_i - t_{i-1}$, $V_{Vdd}(t_{i-1}) = r_1$ for all i. The assumption of sufficient positive slack holds for low power, low performance circuits. Therefore $V_{Vdd} = r_1$ can be specified as the initial value of V_{Vdd} in sleep mode in Eq. (3.32). Hence, applying the initial condition that $V_{Vdd}(0) = r_1$, we have

$$V_{Vdd}(t) = r_1^s + (r_1 - r_1^s)e^{-\frac{t}{R_{sp}C_L}}. \tag{3.33}$$

The value of V_{Vdd} at the end of sleep mode, V_{sleep}, is obtained by substituting $t = T_{sleep}$ in Eq. (3.33).

The energy savings E_s of the power-gated logic cluster in sleep mode with respect to an ungated cluster can be determined by

$$E_s = V_{dd}I_{leak}(V_{dd})T_{sleep} - \int_0^{T_{sleep}} V_{Vdd}I_{leak}(V_{Vdd})dt. \tag{3.34}$$

3.4.3.4 Determination of R_{lin}

To determine the resistance of sleep transistor in linear region, the method proposed in [87] for extraction of series resistance (R_{sd}) of MOS device is followed. It is described for completeness here. Two operating points $(I_{sd}^{(1)}, V_{sg}^{(1)}, V_{th}^{(1)})$ and $(I_{sd}^{(2)}, V_{sg}^{(2)}, V_{th}^{(2)})$ with $V_{sd} = 0.05V$ are determined from I_{sd} vs. V_{sg} characteristics for a specific width W_{sp} of the transistor. All V_{sg} are chosen such that they satisfy constant mobility condition [87][26]

while V_{th} is determined by g_m/I_D method. The drain current $I_{sd}^{(i)}$, for $i = 1, 2$, including the effects of R_{sd} is given by

$$I_{sd}^{(i)} = \mu C_{ox} \frac{W_{eff}}{L_{eff}} \left(V_{sg}^{(i)} - V_{th}^{(i)} - 0.5 V_{sd} \right) \left(V_{sd} - R_{sd} I_{sd}^{(i)} \right). \tag{3.35}$$

Here μ is the constant carrier mobility, $C_{ox} = \epsilon_{ox}/t_{ox}$ is the oxide capacitance, W_{eff} and L_{eff} are effective width and channel length of sleep transistor. From the pair of equations (3.35), R_{sd} is determined. Further μ is determined from one of the equations of drain current in Eq. (3.35). Let R_{ch} denote the intrinsic channel resistance. Then $R_{lin} = R_{ch} + R_{sd}$. From [26],

$$R_{lin} = R_{sd} + \left(\frac{L_{eff}}{\mu C_{ox} W_{eff} (V_{sg} - V_{th} - 0.5 V_{sd})} \right). \tag{3.36}$$

Table 3.3 shows linear region resistances for PMOS sleep transistors of different sizes in an industrial 65nm bulk CMOS technology library with nominal $V_{dd} = 1V$ at 100°C, $L = 0.06\mu m$, $V_{sg} = 1V$, $V_{sd} = 0.05V$.

W (μm)	0.54	1.2	2.4	4.8	9.6	12
R_{lin} $(k\Omega)$	2.57	1.203	0.612	0.322	0.167	0.134

TABLE 3.3: Linear region resistance of PMOS transistors.

3.4.3.5 Heuristics for I_0, I_1 and R_{sp}

Correction terms in Eq. (3.21) and Eq. (3.22) were applied in Eq. (3.20), to account for saturation region of sleep transistor operation. From Eq. (3.15) the current in saturation region is underestimated by $I_0 = I_{on,sat} - I_{st}(V_{Vdd} = V_{sleep})$ where $I_{on,sat}$ is the saturation drain current. Fig. 3.11 shows the variation of error in width-normalized estimated drain current $\frac{I_{error}}{W} = \frac{1}{W}(I_{sd} - I_{st})$ with V_{sd} where I_{st} is as determined from Eq. (3.15) for all V_{Vdd}. Similarly for I_1, we choose error in current corresponding to one of the values of V_{Vdd} in the interval $[(V_{dd} - V_{sg} + V_{th}), (V_{dd} - V_{DSAT})]$. From our experiments, we empirically choose $V_{sd} = 0.6V$, at which the error determined from Fig. 3.11 is $-I_1 = 0.174 I_0$.

The voltage dependent pseudo-resistance changes as V_{Vdd} evolves with time according to Eq. (3.18). Hence it can be inferred that the time constant $R_s C_L$ also varies with time. In our experiments, we have observed that in large logic clusters, the values of pseudo-resistance and its dynamic range are less than that for small logic clusters as leakage currents are higher in the former case. A typical variation of pseudo-resistance with V_{Vdd} is shown in Fig. 3.12 in the next section. The effect of a larger value of pseudo-resistance on V_{Vdd} is that it takes a longer time to change V_{Vdd} levels than with smaller values. Typically, higher values of pseudo-resistance determine V_{Vdd} after about

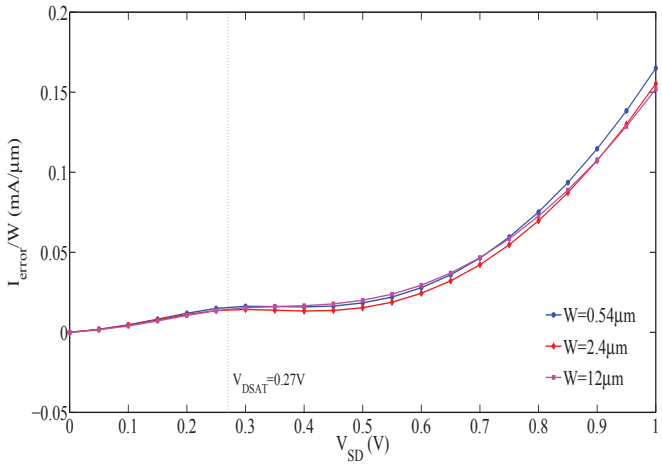


FIGURE 3.11: I_{error}/W vs. V_{sd} for 65nm PMOS transistors.

four time constants of sleep time. Considering these observations, we choose R_{sp} as the pseudo-resistance at $V_{Vdd} = r_1$.

3.4.4 Experimental Results

The model was applied to ISCAS85 benchmark circuits [81] listed in Table 3.8 to validate the approximations proposed. The results were compared with simulations using Spectre circuit simulator of Cadence Virtuoso ICFB. Detailed results are reported for c7552, c6288, c2670 and c432 and a summary of results is provided for all circuits in Table 3.8.

The circuits were synthesized with two sets of logic gates, {nand2, nor2, xor2, and2, fa, ha, inv} in high-V_{th}(HVT) and {nand2, nor2, xor2, inv} in standard-V_{th}(SVT) process options of an industrial 65nm CMOS technology library. The two sets of circuits present a wide range of variation in leakage current and total circuit capacitance profiles for evaluation. For each logic gate, leakage currents were determined for supply voltage varying between 0 and 1V for all input patterns at an operating temperature of 100°C using Spectre circuit simulator. Each of these profiles were then fitted with polynomials of degree 7 using MATLAB. The maximum error between evaluated leakage current and simulated leakage current was less than 3% except near $V_{Vdd} = 0$, where absolute values of leakage current are negligible. Further, the leakage current profile of the complete circuit was determined by weighting the polynomials with number of occurrences in the gate netlist and adding them together to form I_{leak} in Eq. (3.12). A leakage current profile for c6288 is shown in Fig. 3.12. From this curve, pseudo-resistance is determined at each point in the Virtual-Vdd segment.

One set of Spectre simulations of high-V_{th} PMOS transistor is required for each technology library to determine threshold voltages, constant mobility, saturation voltage and saturation currents. To establish these parameters, I_{sd} vs. V_{sd} characteristics at

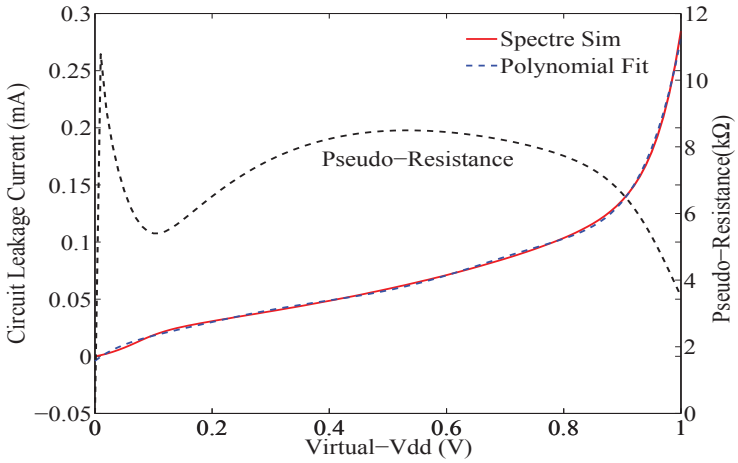


FIGURE 3.12: Leakage current and pseudo-resistance profile in c6288.

$V_{sg} = 1V$ and I_{sd} vs. V_{sg} characteristics at $V_{sd} = 0.05V$ and $V_{sd} = 1V$ with $W_{sp} = 0.54\mu m$ were obtained using Spectre.

To compare wakeup time estimation using models with circuit simulations in Spectre, V_{dd} was set to 1V. Without loss of generality, all primary inputs of the circuit were set to logic 0. The evolution of Virtual-Vdd during wakeup and sleep modes in c7552 is shown in Fig. 3.13 and Fig. 3.14 respectively. In Table 3.4 and Table 3.6, the maximum voltage levels attained by Virtual-Vdd and the wakeup times with sleep transistors of different sizes are given. Table 3.8 shows average errors (μ_{error}) in estimation of the two quantities for all circuits considered in this work. The wakeup time is estimated by equations (3.25)-(3.27) within an average error margin of 16.3% for a variation of 22× in sleep transistor sizes. The steady-state Virtual-Vdd is determined within 1.8% on an average from the corresponding results of Spectre simulations. Further a significant reduction in computation time is achieved for wakeup time estimation using the model compared to a SPICE level circuit simulator. For example, model calculations in c6288 using MATLAB took 21ms compared to 4 minutes in SPICE level circuit simulations.

In logic clusters that do not satisfy wakeup dependency [71, 80], short-circuit currents are generated due to changing logic states of internal nodes as V_{Vdd} increases towards r_1 in wakeup mode. They create the effect of altering effective resistance of the circuit and hence wakeup time. In other words, the accuracy of wakeup time estimation is reduced when the effects of short-circuit currents are not taken into account as is shown for c499 in Table 3.4 and Table 3.6. To address this problem it is necessary to model individual cells for short-circuit currents when both supply voltage and its rise time are varying. The cluster definitions and sleep transistor widths considered in this work are not designed to satisfy wakeup dependency or meet a particular peak current constraint [80] as the problem of logic clustering was not addressed in this work.

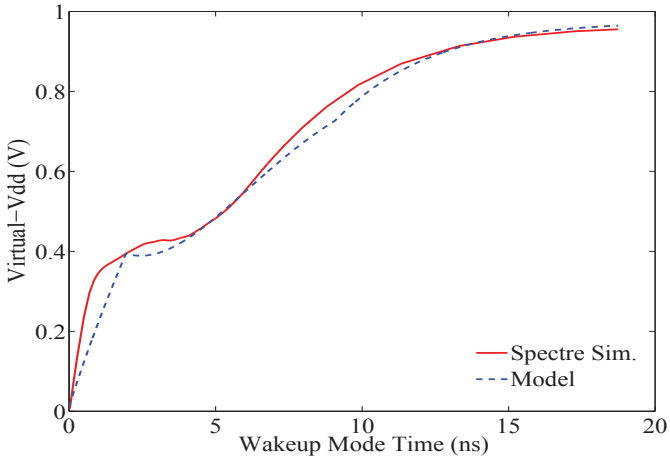


FIGURE 3.13: Virtual-Vdd in wakeup mode (W=1.2μm) in c7552.

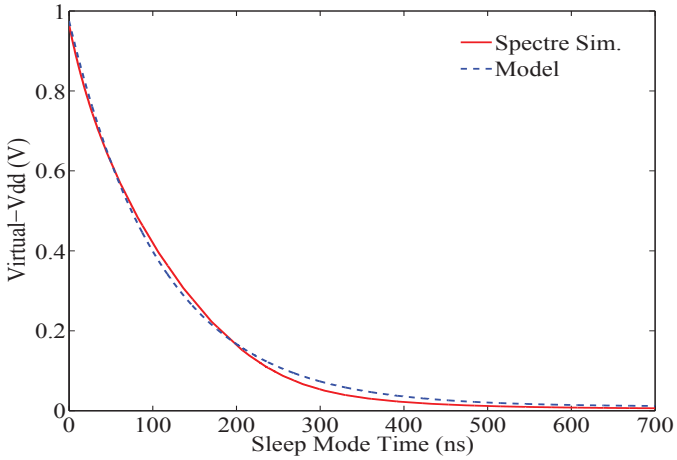


FIGURE 3.14: Virtual-Vdd in sleep mode (W=1.2μm) in c7552.

$W(\mu m)$	r_1 (V) Model [Eq. (3.17)] (Spectre)			
	c7552	c6288	c2670	c432
0.54	0.94 (0.93)	0.96 (0.94)	0.97 (0.96)	0.99 (0.99)
1.2	0.97 (0.96)	0.98 (0.97)	0.98 (0.98)	0.99 (0.99)
2.4	0.98 (0.98)	0.99 (0.98)	0.99 (0.99)	0.99 (0.99)
4.8	0.99 (0.99)	0.99 (0.99)	0.99 (0.99)	0.99 (0.99)
9.6	0.99 (0.99)	0.99 (0.99)	0.99 (0.99)	0.99 (0.99)
12	0.99 (0.99)	0.99 (0.99)	0.99 (0.99)	0.99 (0.99)

TABLE 3.4: Maximum Virtual-Vdd after wakeup in ISCAS85 benchmark circuits with HVT cells.

$W(\mu m)$	Wakeup Time (ns) Model [Eq. (3.24),(3.27)] (Spectre)			
	c7552	c6288	c2670	c432
0.54	40.69 (38.22)	43.24 (46.74)	15.55 (15.04)	4.57 (3.89)
1.2	18.94 (18.72)	20.27 (22.24)	7.38 (7.25)	2.20 (1.89)
2.4	9.64 (9.82)	10.35 (11.97)	3.79 (3.75)	1.14(0.92)
4.8	5.08 (5.27)	5.46 (6.44)	2.00 (2.02)	0.60 (0.59)
9.6	2.64 (2.83)	2.84 (3.52)	1.05 (1.11)	0.32 (0.38)
12	2.13 (2.32)	2.29 (2.95)	0.85 (0.92)	0.26 (0.33)

TABLE 3.5: Wakeup time in ISCAS85 benchmark circuits with HVT cells.

$W(\mu m)$	r_1 (V) Model [Eq. (3.17)] (Spectre)			
	c7552	c6288	c2670	c432
0.54	0.72 (0.67)	0.68 (0.60)	0.77 (0.84)	0.97 (0.96)
1.2	0.84 (0.81)	0.81 (0.77)	0.87 (0.91)	0.98 (0.98)
2.4	0.91 (0.89)	0.89 (0.86)	0.93 (0.95)	0.99 (0.99)
4.8	0.95 (0.93)	0.94 (0.92)	0.96 (0.97)	0.99 (0.99)
9.6	0.97 (0.96)	0.96 (0.95)	0.98 (0.98)	0.99 (0.99)
12	0.98 (0.97)	0.97 (0.96)	0.98 (0.98)	0.99 (0.99)

TABLE 3.6: Maximum Virtual-Vdd after wakeup in ISCAS85 benchmark circuits with SVT cells

$W(\mu m)$	Wakeup Time (ns) Model [Eq. (3.24),(3.27)] (Spectre)			
	c7552	c6288	c2670	c432
0.54	36.66 (37.10)	62.79 (63.24)	18.06 (13.89)	4.65 (3.49)
1.2	20.90 (17.16)	34.59 (29.97)	8.17 (6.89)	2.21 (1.74)
2.4	10.01 (9.07)	16.65 (15.88)	4.01 (3.70)	1.14 (0.96)
4.8	5.19 (5.02)	8.45 (9.10)	2.07 (2.04)	0.60 (0.57)
9.6	2.65 (2.77)	4.3 (5.27)	1.06 (1.13)	0.32 (0.37)
12	2.13 (2.28)	3.45 (4.43)	0.86 (0.94)	0.26 (0.33)

TABLE 3.7: Wakeup time in ISCAS85 benchmark circuits with SVT cells

3.4.5 Wakeup Energy Estimation

The models developed for wakeup time estimation can be used to determine the energy consumed due to wakeup mode transition. The wakeup energy is given by

$$E_{wu} = \int_0^{T_{wu}} V_{dd} I_{st} dt. \tag{3.37}$$

The current through the transistor is determined using Eq. (3.15) by substituting Eq. (3.23) for $V_{Vdd}(t)$. The nature of $I_{st}(t)$ is such that the integral cannot be evaluated to obtain a closed form expression. Hence a numerical approach is taken to determine the wakeup energy by summing N_{wu} incremental areas under the $I_{st}(t)$ curve for several

Circuit	C_L (pF)		Max. V_{Vdd} μ_{error} (%)		T_{wu} μ_{error} (%)	
	HVT	SVT	HVT	SVT	HVT	SVT
c7552	2.892	2.993	0.7	2.8	4.7	7.6
c6288	3.171	4.833	0.6	4.3	14.8	11.1
c5315	1.966	2.826	0.6	2.2	12.5	22.3
c3540	1.466	2.037	0.5	1.5	11.2	17.5
c2670	1.148	1.202	0.3	3.0	3.0	7.8
c1908	0.606	0.633	0.2	0.3	21.8	15.4
c499	0.601	0.478	0.2	0.1	24.6	33.7
c432	0.351	0.360	0.1	0.3	16.4	15.1
Mean			0.4	1.8	13.6	16.3

TABLE 3.8: Average relative errors in estimation of maximum V_{Vdd} and wakeup time in ISCAS85 benchmark circuits.

intervals of size Δt in $[0, T_{wu}]$ as

$$E_{wu} \approx V_{dd} \sum_{i=0}^{N_{wu}-1} I_{st_i} \Delta t. \tag{3.38}$$

Table 3.9 shows a summary of relative errors in estimation of wakeup energy for ISCAS85 benchmark circuits using the models proposed above.

Circuit	Wakeup Energy (pJ)		Error
	Spectre	Model	(%)
c7552	3.078	3.062	9.0
c6288	3.695	3.741	5.0
c5315	2.271	2.811	24.8
c3540	1.728	2.264	31.1
c2670	1.086	1.306	20.4
c1908	0.725	0.676	7.0
c499	0.737	0.719	4.4
c432	0.392	0.438	11.7
Mean			12.6

TABLE 3.9: Average relative error in estimation of wakeup energy in ISCAS85 benchmark circuits.

3.4.6 Logic Clustering for Wakeup Scheduling

As stated in Section 3.3.2, the design of power-gated circuits has been viewed as an optimization problem of partitioning logic into clusters satisfying constraints of peak current, delay degradation, sleep transistor area, wakeup time and energy savings. Wakeup time is an important overhead that must be determined for effective power gating, particularly

in logic clusters that undergo frequent mode transitions for run-time leakage power reduction. Run-time leakage reduction has been explored in [78], [88] and [89], where only parts of the overall circuit are put to sleep during short periods of inactivity. Along with a high wakeup energy overhead a large wakeup delay in a cluster can result in reduced energy savings. Hence these two parameters have to be carefully considered in the design of power-gated circuits.

The model for wakeup time estimation presented in this chapter may be applied in scheduling of power-gated logic clusters as part of a larger optimization problem. Consider a combinational circuit **C**. Let $C_i, i = 1, 2, ..., N$ denote N logic clusters obtained by partitioning **C** such that they satisfy constraints of minimum sleep transistor area, peak current, maximum delay degradation and minimum wakeup time. The optimization problem referred to wakeup time constraint is stated as follows. Let $T_{wu,i}$ denote the wakeup time of logic cluster C_i and $T_{wu,max}$ the maximum acceptable wakeup time of the overall circuit **C**. Then,

$$max\left(\sum_{j=1}^{P} T_{wu,j}, \sum_{k=P+1}^{Q} T_{wu,k}, ..., \sum_{l=R+1}^{N} T_{wu,l}\right) \leq T_{wu,max} \tag{3.39}$$

for some P,Q,R,... such that $P \geq 1$, $Q \geq P + 1$,.... Hence a wakeup schedule for the N logic clusters may be derived. The model presented here may be used to determine each $T_{wu,i}$ during the optimization run.

3.4.7 Logic Clustering for Wakeup Energy Control

Partitioning larger power-gated circuits can also be viewed as a way to reduce wakeup energy at a given time and hence reducing average power in a temporal window. Logic circuits that are partitioned into clusters such that only one of them need to wake up depending on its input states contribute to lower instantaneous power consumption. This leads to choice of power gating granularity in logic circuits at the cost of control circuits that need to remain in always-on state. This aspect is elaborated in Chapter 5.

3.5 Conclusion

In this chapter, various considerations in design of power-gated circuits were discussed apart from reviewing previous work. Further, a semiempirical approach for estimation of wakeup time of a power-gated logic cluster was presented. The technique relies on only a few basic circuit parameters and one time SPICE level simulations per technology library due to approximations used. A simple method to determine steady state Virtual-Vdd after wakeup as a function of sleep transistor size and leakage current was fundamental to development of rest of the model. In sleep mode, the model can be used to determine

leakage energy savings in inactive states of the circuit. In other words, some of the key parameters used as optimization criteria for logic clustering have been captured in closed-form expressions. Specifically, the model may be applied in logic clustering for wakeup scheduling optimizations. It should be noted that, short circuit currents (I_{sc}) that are generated in the internal nodes during wakeup mode are neglected in Eq. (3.14). In this work I_0 and I_1, which are assumed to be constants based on heuristics developed in Section 3.4.3.5, must be replaced with time and V_{Vdd} dependent models. Modeling short circuit currents in logic gates when both supply voltage and its rise time are varying leads to a unified model. A further limitation of these models is that process and temperature variations cannot be accounted for as the models are not parameterized with respect to bias and temperature dependent threshold voltage and temperature as an independent variable.

Chapter 4

Variable Precision Arithmetic Units for Low Power

4.1 Introduction

Arithmetic units such as adders and multipliers find application in almost all digital computational hardware. In the design flow for generation of hardware microtasks, data precision has been used as a parameter for adapting precision of adder in the datapath. For hardware microtasks to have flexibility by dynamically changing the precision just enough to that of incoming data, it is essential to have arithmetic units like adders to be reconfigurable for variable precision. In this chapter arithmetic units that are reconfigurable for different precisions of data are explored. An analysis of power gating as a low power technique to suppress leakage current in unused logic is performed.

A review of related work on energy efficient arithmetic circuits is presented in Section 4.2. In Section 4.3, the proposed design approach and an analysis of energy savings in active mode of power-gated arithmetic units is presented. In Section 4.4, a brief description of Brent-Kung and Kogge-Stones adders is given while focusing on the parallel-prefix tree structure of carry generation and propagation. The design of power-gated reconfigurable adders is explained in Section 4.5. In Section 4.6, the experimental setup for the proposed design and power estimation flow is described and results are discussed. Section 4.7 concludes the chapter.

4.2 Variable Precision Arithmetic Units: A Review

Arithmetic units are dense but regular structures that can be implemented with varying granularity of word slices and parallelism. While this leads to hardware realizations with speed-area tradeoffs [90], another parameter of significant interest in the design space is power consumption. For arithmetic units in processor or ASIC-like implementations

that support dynamically variable precision arithmetic, it is even more important to control power consumption in unused logic of an arithmetic unit to achieve dynamic and standby energy efficiency [91, 37]. Adders are used in addressing logic for instruction and data sequencing in microprocessors whereas multipliers are an integral part of Multiply-Accumulate (MAC) operations in datapath. In this work we focus on adders and extend the analysis to multipliers as FSM controlled datapaths in microtasks of [1] can be potentially upgraded to include a multiplier. Arithmetic units are also ubiquitously used in fixed-point and variable-precision implementations of signal processing algorithms, hardware for numerical computations in optimization procedures and linear algebra routines. Further, they are also available as part of reconfigurable circuits like FPGAs [92] and Floating-Point Units [93, 94] used for variable precision, high accuracy arithmetic.

4.2.1 Low Power Optimizations

Several fixed-precision parallel-prefix trees in adders have been studied for energy and delay properties using different circuit techniques in [95]. A methodology based on logical effort and energy models has been proposed to determine gate types and sizing in parallel-prefix trees. In [96], a decimal floating-point adder for variable precision addition that uses Kogge-Stones parallel-prefix tree structure is described. Floating-point adders represent one end of the accuracy vs. hardware complexity design space while a fixed-precision adder lies at opposite end of the same space. Reconfigurable precision adders also offer a trade-off with respect to accuracy and hardware complexity and occupy a position in between the two ends of the space.

Being a regular logic circuit, the generation of such adders (and other arithmetic circuits in general) has been a topic of design space exploration in datapaths. One such example is a high-level synthesis procedure involving bit-level reuse for variable precision adders is described in [97]. Recent works on low power adders have focussed on power optimized implementations of a full adder unit at gate level and its repeated use to form a N-bit carry ripple adder. In [98], the leakage power in a full adder is sought to be controlled by selectively stacked inverters. The problem of optimal sizing of sleep transistors for power gating adders at fine granularity is presented in [99]. N such full adders are required to form a N-bit cascaded adder. While sleep transistor widths are optimized based on statistical properties of inputs and in-rush current analysis, area overhead due to sleep transistors and other associated control circuits in fine grained power gated circuits is, in general larger than those for cluster based coarse grain power gating [19]. An analysis of various overheads and design parameters of power-gated execution units is described in [88]. A specific study of twin-precision arithmetic circuits with coarse grain power gating has been presented in [100] but it does not provide

a method to generalize it to other precisions and analyze the overheads. In [101], a data-width-driven power gating method for integer arithmetic circuits close to the work described in this chapter is presented.

In this work, a general approach for power gating unused parts of an arithmetic unit to achieve configurable variable-precision operation and reduced leakage power is presented. The method is then applied to two reconfigurable power-gated adders, based on Brent-Kung (BK) [15] and Kogge-Stones (KS) [16] parallel-prefix trees for carry generation. Two 32-bit adders with the flexibility of being configured as a 8-bit, 16-bit and 24-bit adder are described. The power consumption in the above adders in power-gated context is compared with 32-bit adders performing respective precision addition. This method can be extended to power-gated multipliers as, in general, multipliers can be designed to be made of a partial products stage and a stage of adders to sum the partial products [102]. A typical flow for design of power-gated variable-precision circuits is shown in Fig. 4.1. An essential part of the design flow is power estimation of both ungated and power-gated circuits as it is necessary to determine if energy savings effectively result due to insertion of power gating structures.

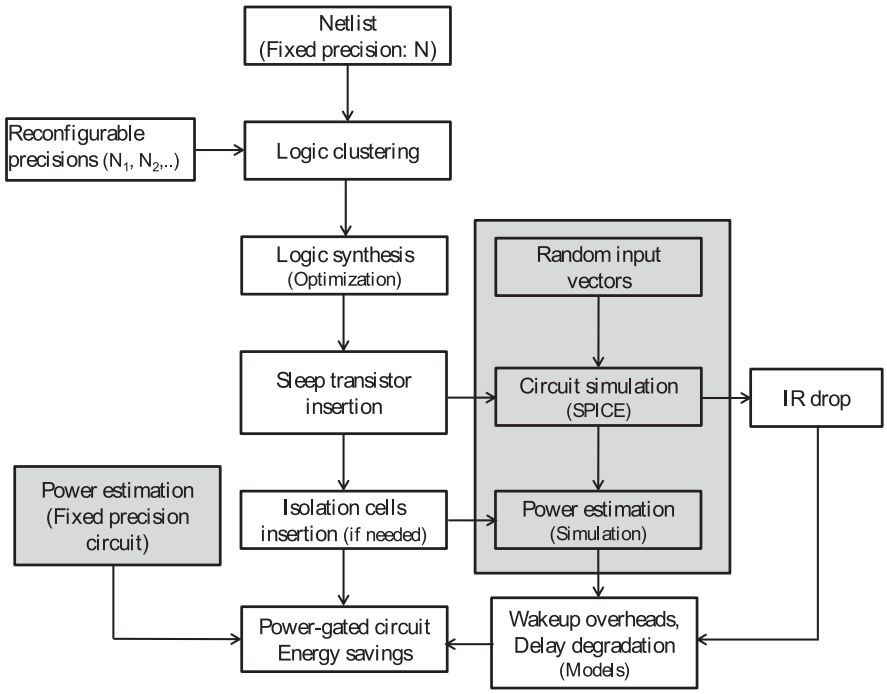


FIGURE 4.1: Typical flow for design of variable-precision power-gated arithmetic circuits.

4.3 Logic Clustering Method and Energy Savings

4.3.1 Logic Clustering

Aggressive leakage power reduction in an arbitrary logic circuit by power gating involves exercising sleep structures for part of the circuit that is not used in a schedule. Given a N-bit arithmetic unit in general, the design of a power-gated reconfigurable variable-precision unit can be viewed as a task of partitioning N-bit logic cluster C_N of size $S(C_N)$ into subclusters C_{N_1} whose size $S(C_{N_1})$ is a function of lower precision $N_1 < N$ and a power-gateable cluster $C_{N_1'}$ of size $S(C_{N_1'})$ where $N_1' = N - N_1$. To reduce the precision further consider a partition of C_{N_1} into C_{N_2} and $C_{N_2'}$ such that $N_2 < N_1$, $N_2' = N_1 - N_2$ and $S(C_{N_2'}) < S(C_{N_1'})$. For a cluster-based, coarse-grain power gating design, $C_{N_2'}$ can be power-gated to achieve leakage power savings when the N-bit logic circuit is configured for N_1-bit precision. In general, $C_{N_k'}$ may be repeatedly partitioned until the desired precision is achieved at the cost of increased power gating blocks. Let $A = (a_{N-1}, a_{N-2}, ..., a_1, a_0)$ and $B = (b_{N-1}, b_{N-2}, ..., b_1, b_0)$ denote two operands of an arithmetic unit. The logic clustering method described above as applied to the logic gates in the transitive fanout of A and B is given in Algorithm 1.

Algorithm 1 Partition N-bit Arithmetic Logic Circuit

Require: precisions N_k, N such that $N > N_1 > ... > N_{k-1} > N_k$, cluster C_N, input bits $a_m, b_m, m = 0, 1, ..., N-1$ of C_N
Ensure: Clusters C_{N_j}
  $N_{init} = N$
  for $(j = 1, j \leq k, j = j + 1)$ **do**
    for all $(a_m, b_m,\ m = [N_{init}, N_{init} - 1, ..., N_j + 1])$ **do**
      $N_j' = N_{init} - N_j$
      if $(N_{init} = N)$ **then**
        /* FO(x,y) denotes transitive fanout of inputs x, y */
        $C_{N_j'} \leftarrow$ getGates in FO(a_m, b_m) in $C_{N_{init}}$
      else
        $C_{N_j'} \leftarrow$ getGates in FO(a_m, b_m) in $(C_{N_{init}} - C_{N_{j-1}'})$
      end if
      $C_{N_j} \leftarrow$ getGates in $(C_{N_{init}} - C_{N_j'})$
    end for
    $N_{init} = N_j$
  end for

It should be noted that for systems that remain in standby mode for long periods of time compared to their active mode duration, power gating results in significant energy savings. However in applications that use dynamically variable precision operation with frequent active mode to sleep mode transitions and vice versa, the energy savings due to power gating is reduced by mode transition energy overheads.

4.3.2 Energy Savings in Active Mode

Let the time taken by a cluster C_{N_i} to wakeup from sleep mode to active mode be given by its wakeup time $T_{wu_i}(C_{N_i}) = f(I_{st_i}, C_{load_i}, I_{leak_i})$, where f is a function of sleep transistor current I_{st_i}, load capacitance of the cluster C_{load_i} and bias-dependent leakage current I_{leak_i}. The wakeup energy is given by

$$E_{wu_i} = \int_0^{T_{wu_i}} V_{dd} I_{st_i}(t) dt \tag{4.1}$$

where V_{dd} is the power supply voltage. Let four logic clusters C_{N_i}, $i = 1, 2, 3, 4$ remain in sleep mode for times T_{sleep_i}, $i = 1, 2, 3, 4$, respectively in any duration of time $T > max(T_{sleep_i}, i = 1, 2, 3, 4)$. Note that T_{sleep_i} is the total time for which C_{N_i} is in sleep mode and is not necessarily continuous due to mode transitions. Assume that the non-reconfigurable arithmetic circuit is in active mode for all the time T and that due to the variable precision nature of operation, at least one cluster of power-gated reconfigurable arithmetic circuit is in active mode at any given time during T. The energy consumed by the fixed-precision circuit in time T with average active mode power $P_{active,av}$ is given by

$$E_{fp} = P_{active,av} T. \tag{4.2}$$

Let $P_{active_i,av}, P_{sleep_i,av}$ and T_{sleep_i} be the average active mode power, average sleep mode power and total sleep time of cluster C_{N_i} respectively. Given a reconfigurable schedule the energy consumed by the power-gated circuit is given by

$$E_{pg} = \sum_i \left[P_{active_i,av}(T - T_{sleep_i}) + E_{sleep_i} + n_i E_{wu_i} + E_{iso} \right] \tag{4.3}$$

where

$$E_{sleep_i} = P_{sleep_i,av} T_{sleep_i} \tag{4.4}$$

n_i is the number of wakeup transitions of cluster C_{N_i} in time T, and E_{iso} is the energy consumed by isolation cells that are an overhead. The energy savings of power-gated arithmetic unit in comparison with fixed-precison circuit is given by $E_s = E_{fp} - E_{pg}$. Assuming that drop in the virtual supply voltage across a cluster from V_{dd} is negligible, $P_{active,av} \approx \sum_i P_{av,active_i}$ for the same set of input data. Hence

$$E_s = \sum_i \left[(P_{active_i,av} - P_{sleep_i,av}) T_{sleep_i} - n_i E_{wu_i} - E_{iso} \right]. \tag{4.5}$$

From Eq. (4.5), it can be inferred that higher the $P_{av,active_i}$ due to nature of input patterns, higher is the energy savings in power-gated circuit even at lower T_{sleep_i}. The circuit will have reduced energy consumption as long as the number of mode transitions are such

that wakeup energy and isolation cells do not offset the gains. Given a reconfiguration schedule, Eq. (4.5) can be used to estimate average energy savings in active mode of the power-gated circuit.

4.4 Logic Clustering in Arithmetic Circuits

4.4.1 Parallel-Prefix Trees

An N-bit adder $S = A + B$ with input carry c_{-1} can be represented as follows. Let m denote the index of a bit in the N-bit word such that $i = 0, 1, ..., N-1$. Then the sum bit s_m and carry bit to the next stage c_m is given by [103],

$$\begin{cases} s_m = a_m \oplus b_m \oplus c_{m-1}, \\ c_m = a_m b_m + c_m(a_m \oplus b_m). \end{cases} \tag{4.6}$$

Numerous adder architectures have been proposed based on the way carry of each stage in Eq. (4.6) is generated. Since the worst case critical path between inputs and outputs of an adder starts from c_{-1} and ends in c_N, a carry ripple adder would have the slowest path. A parallel adder referred to as carry look ahead (CLA) adder exploits parallel computations of partial terms leading to the carry of each stage in to reduce the critical path. Parallel-prefix trees have tree-based structure for the generation of the carry in Eq. (4.6) as shown in Fig. 4.2 and Fig. 4.3. A detailed description of their derivation is given in [15] and [16]. Assuming $p_i = a_m \oplus b_m$ and $g_m = a_m b_m$, with

$$(G_m, P_m) = \begin{cases} (g_m, p_m), & m = 0, \\ (g_m + p_m G_{m-1}, p_m P_{m-1}), & m = 1, .., N-1 \end{cases} \tag{4.7}$$

the carry of each stage $c_m = G_m$ and the sum bit $s_m = p_m \oplus c_m$. Therefore the adders have three stages: 1) the input AND, XOR stage to generate g_m and p_m, 2) the intermediate terms G_m and P_m for generation and propagation of carry of each stage referred to as parallel prefix and 3) the final XOR stage for sum bits of the adder. The size of the adder for a particular precision is directly proportional to its precision. Since the data flow takes place along only one direction (right to left), application of the algorithm in Section 4.3.1 is simply equivalent to cutting the directed graph by a straight line as shown in the figures. The BK adder has fewer logical elements and adjacent cluster connectivity compared to a KS adder whereas the carry signal in the latter is realized in fewer logical stages and with reduced fanout compared to the former.

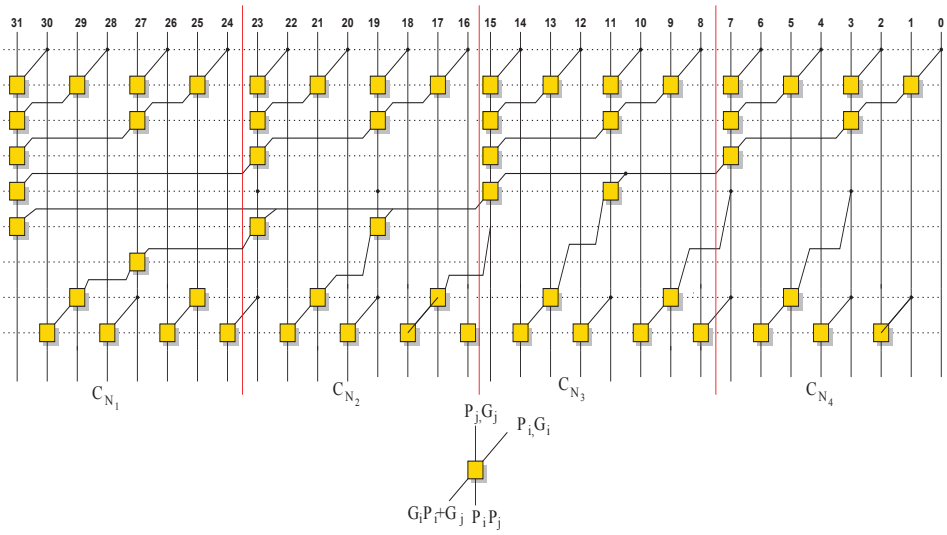


FIGURE 4.2: Parallel-prefix tree structure for carry generation in BK adder.

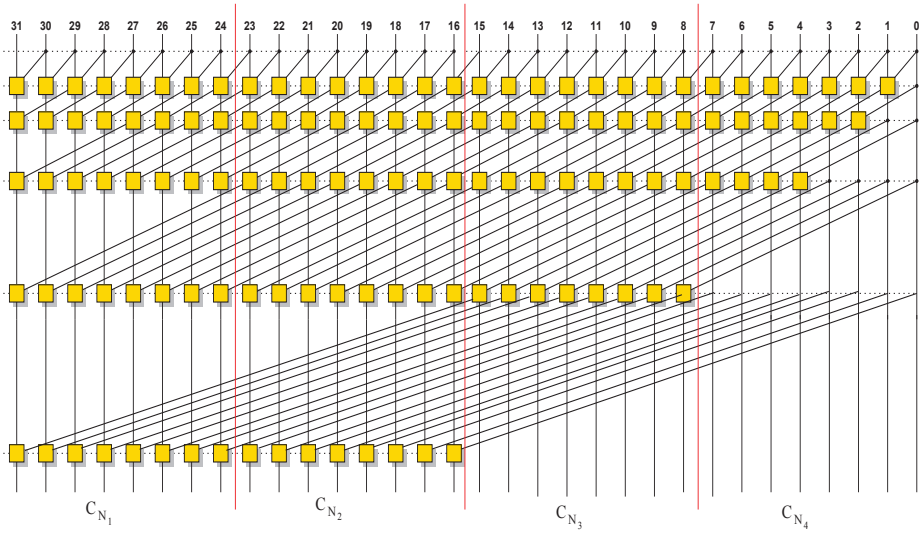


FIGURE 4.3: Parallel-prefix tree structure for carry generation in KS adder.

4.4.2 Partial Products in Multiplier

A multiplier has a more complex structure than an adder. However the partial products in a multiplier presents similar opportunities for clustering for variable-precision operation. An $M \times N$ multiplier generator based on Booth algorithm is described in [102]. In this work, for simplicity, we consider a 4×4 multiplier whose partial products can be clustered for 2×2, 3×3 multiplication as an example. In Fig. 4.4, AND terms are clustered based on the dependency of input bits that are part of the lower precision word. It should be noted that, just as AND/XOR stage outputs are dependent on exactly one pair of bits irrespective of carry signal, partial products are easily clustered. The sum stage however presents a complex picture for power gating and is discussed in the next section.

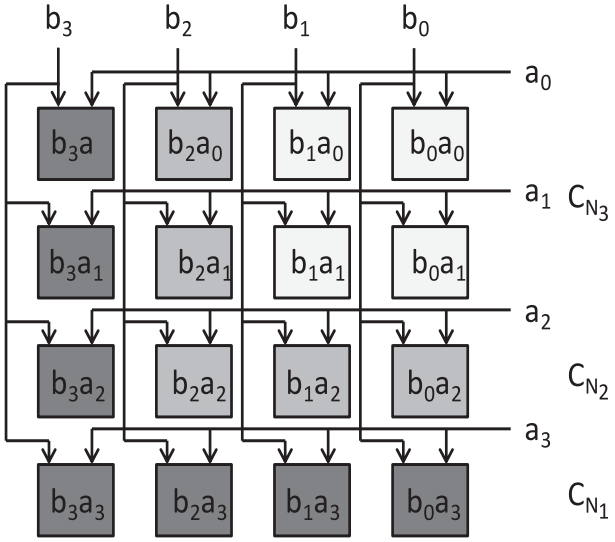


FIGURE 4.4: Partial product structure in a binary multiplier clustered for variable precision and power gating.

4.5 Power-Gated Reconfigurable Circuits

4.5.1 Variable-Precision Adders

In order to design a power-gated 32-bit adder that can be configured as a 32-bit, 24-bit, 16-bit or an 8-bit adder, the parallel-prefix tree of the adder is partitioned into four blocks as shown in Fig. 4.2 and Fig. 4.3 and are a part of clusters represented by C_{N_i}, $i = 1, 2, 3, 4$ in Fig. 4.5. It should be noted that XOR/AND and Sum XOR stages are also part of C_{N_i}. Further a multiplexer is introduced after the sum XOR stage to select the (zero-padded) outputs of sum and carry bits as a function of adder select bits for configuration. Four sleep transistors of PMOS type (header sleep transistors) are inserted between the supply voltage (V_{dd}) and the virtual supply voltage $V_{Vdd,j}$ where $j = 32, 24, 16, 8$ for each of the four blocks. It represents a cluster based coarse-grain power gating scheme. The mode transitions of the four sleep transistors are controlled by $SLEEPi$ where $i = 32, 24, 16, 8$. The schedule for operating the adder in different configurations to realize variable-precision addition with the same hardware is given in Table 4.1. A block is in ON state or active mode when the corresponding $SLEEP$ signal is 'low' and it is in sleep mode or OFF state when the $SLEEP$ signal has a 'high' value.

4.5.2 Power Gating in Multipliers

Figure 4.6 shows the way partial products are used in the sum stage. It can be seen that partial product terms in different clusters need to be input to adder clusters and that the data flow between depends on the clustering criterion used. Figure 4.6 shows a logic clustering method as described in Section 4.3.1 which involves data flow in only

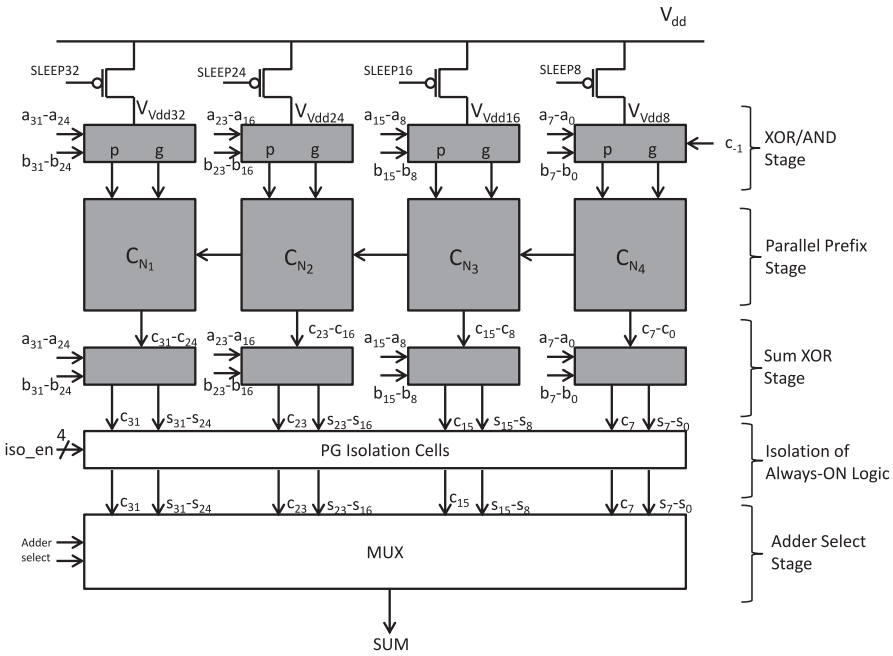


FIGURE 4.5: Power-gated reconfigurable adder.

Adder	32-bit	24-bit	16-bit	8-bit
Select bits	11	10	01	00
C_{N_1}	ON	ON	ON	ON
C_{N_2}	ON	ON	ON	OFF
C_{N_3}	ON	ON	OFF	OFF
C_{N_4}	ON	OFF	OFF	OFF

TABLE 4.1: On-Off schedule for operation of different adders in the power-gated reconfigurable adder.

one direction across the clusters avoiding isolation cells. However unlike the adder, the number of product bits that get generated out of each cluster increases with precision and therefore the clusters have different sizes.

4.6 Power Estimation and Analysis

4.6.1 Experimental Setup

Two 32-bit reconfigurable adders of BK and KS types were designed to implement 32-bit, 24-bit,16-bit or 8-bit addition. Architecturally, the logic was partitioned as described in Section 4.3.1, such that, unused gates in any particular configuration (for 24, 16 or 8-bit addition or complete shut-off) could be power-gated using sleep transistors. Further, fixed-precision 32-bit BK and KS adders were designed with respective parallel-prefix tree type architectures except that the logic for reconfigurability and sleep transistors were eliminated for comparison with power-gated ones. The four adders were synthesized into gate-level netlists using std.-V_{th} (SVT) cells of an industrial 65nm bulk CMOS technology

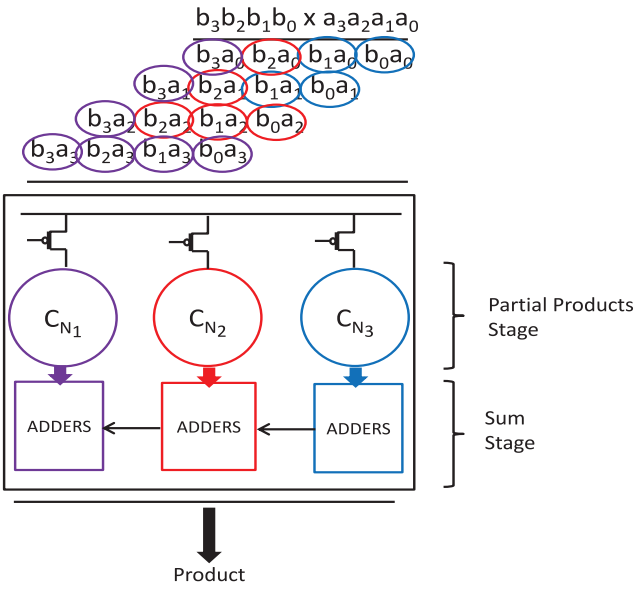


FIGURE 4.6: Power-gated reconfigurable multiplier.

library using Synopsys Design Compiler applying a maximum path delay constraint of 1ns from all inputs to all outputs. The standard cells used in both cases were restricted to a set {nand2, nor2, xor2, inv} of logic gates without loss of generality. Identical wire load models and operating conditions were used for synthesis. Care was taken not to alter the parallel-prefix tree structure during logic synthesis. Table 4.2 shows the areas of the four adders and the overhead in reconfigurable adder over fixed-precision adder due to the multiplexer.

Adder	Fixed 32-bit		Variable Precision		Area Overhead	
	BK	KS	BK	KS	BK	KS
Area (μm^2)	915.2	1493.5	1053.5	1631.8	15%	9.2%

TABLE 4.2: Area overhead in power-gated reconfigurable adders over non-reconfigurable adders.

The SPICE netlists of resulting gate netlists were obtained using Cadence Virtuoso ICFB environment. High-V_{th} sleep transistors of PMOS type were inserted as headers of power gating circuit of the reconfigurable adders. The sleep transistors were sized to W=1.2μm without the loss of generality. Sleep transistor sizing is generally seen as an optimization problem [80] based on in-rush current and IR drop, but it is not addressed in this work. In order to estimate average power consumption of both circuits for addition of different word sizes, 10000 input words randomly generated with uniform distribution were applied to the circuit at an interval of 1.5ns and simulated using Eldo SPICE simulator at a nominal supply voltage (V_{dd}) of 1V at 100°C temperature. Transient analysis was carried out for a total time of 55μs. The non-reconfigurable adder was in standby state for a time of 40μs whereas the reconfigurable adder was power-gated

for 10μs and it remained in idle state (ON, but without changes in inputs) for rest of the time. Further, in configurations for 24, 16 and 8-bit addition, unused parts of the circuit were power-gated even during the active period of the [0, 55μs] interval. When the adder was configured in reduced precision, the input vectors were zero padded to form a 32-bit word. The current drawn by a power-gated reconfigurable KS adder in 16-bit configuration is shown in Fig. 4.7 and the virtual supply voltages for clusters C_{N_4} and C_{N_2} are shown in Fig. 4.8 and Fig. 4.9 respectively. Table 4.3 to Table 4.5 show results obtained from the circuit simulation of two adders. $I_{max,ac}$ represents the peak current drawn from the supply in active mode. In the active mode the unused clusters are shut-off. $P_{total,av}$ denotes the average power across a time interval of [0, 55μs]. $P_{idle,av}$ and $P_{sleep,av}$ denote average power in idle state, when the respective clusters are ON but inputs do not change and in sleep mode when all clusters are switched-off, respectively. The average power of any configuration in active mode only can be calculated as

$$P_{active,av} = \left[\frac{P_{total,av}T - (P_{idle,av}T_{idle} + P_{sleep,av}T_{sleep})}{T_{active}}\right]. \tag{4.8}$$

Further $P_{active,av} = P_{dyn,av} + P_{idle,av}$ where $P_{dyn,av}$ denotes the average dynamic power due to switching activity in active mode.

Function	$I_{max,ac}$ (mA)		$P_{active,av}(\mu W)$		$P_{idle,av}(\mu W)$	
	BK	KS	BK	KS	BK	KS
32-bit addition	4.92	4.61	222.3	240.7	55.5	62.9
24-bit addition	4.22	3.86	185.2	200.6	55.8	62.8
16-bit addition	2.92	2.63	142.6	152.8	55.4	62.3
8-bit addition	1.80	1.59	102.4	104.8	55.1	61.9

TABLE 4.3: Power consumption in non-reconfigurable/non-power gated BK and KS adders.

Adder	$I_{max,ac}$ (mA)		$P_{active,av}$ (μW)		$P_{idle,av}$ (μW)		$P_{sleep,av}$ (μW)	
	BK	KS	BK	KS	BK	KS	BK	KS
32-bit adder	1.91	2.02	223.8	286.2	60.5	81.7	8.6	8.7
24-bit adder	1.52	1.56	171.8	218.9	47.7	62.5	7.8	8.2
16-bit adder	1.17	1.09	118.3	139.5	34.8	40.1	7.3	7.6
8-bit adder	0.73	0.75	65.1	66.1	20.9	20.7	6.8	7.0

TABLE 4.4: Power consumption in power-gated reconfigurable BK and KS adders.

4.6.2 Results

The total average power in active mode and average power in standby (or sleep) mode consumed by the two adders of non-reconfigurable and reconfigurable types are shown in Table 4.3 and Table 4.4 respectively. For each type, the power consumed across BK and

Adder	$V_{Vdd32,max}$ (mV)		$V_{Vdd24,max}$ (mV)		$V_{Vdd16,max}$ (mV)		$V_{Vdd8,max}$ (mV)	
	BK	KS	BK	KS	BK	KS	BK	KS
32-bit	987	982	986	977	984	981	994	993
24-bit	xx	xx	986	976	985	984	994	993
16-bit	xx	xx	xx	xx	984	981	995	993
8-bit	xx	xx	xx	xx	xx	xx	993	992

TABLE 4.5: Maximum Virtual-V_{dd} in active mode operation of BK and KS adders.

KS adders is consistent with their respective parallel-prefix tree structures. The power-gated reconfigurable BK and KS adders show leakage power reduced by a factor of about 7 and 8 respectively in comparison with non-reconfigurable BK and KS adders without any power gating. Further the average active power in 8-bit and 16-bit power-gated BK and KS adders actually shows a reduction by about 37% (8-bit) and 8% (16-bit) when compared to ungated adders despite 9% and 15% increase in area of logic gates in the latter. It can be inferred that to that extent, considerable power is consumed in active mode at lower precisions due to unnecessary switching activity in unused logic of 32-bit adder. Table 4.5 shows maximum virtual supply voltage levels in steady state of active mode. In other words, the results show a higher IR drop across the sleep transistor in KS adder than BK adder despite identical sleep transistors. This can be attributed to higher leakage current [20] in KS adder due to higher hardware complexity than in BK adder.

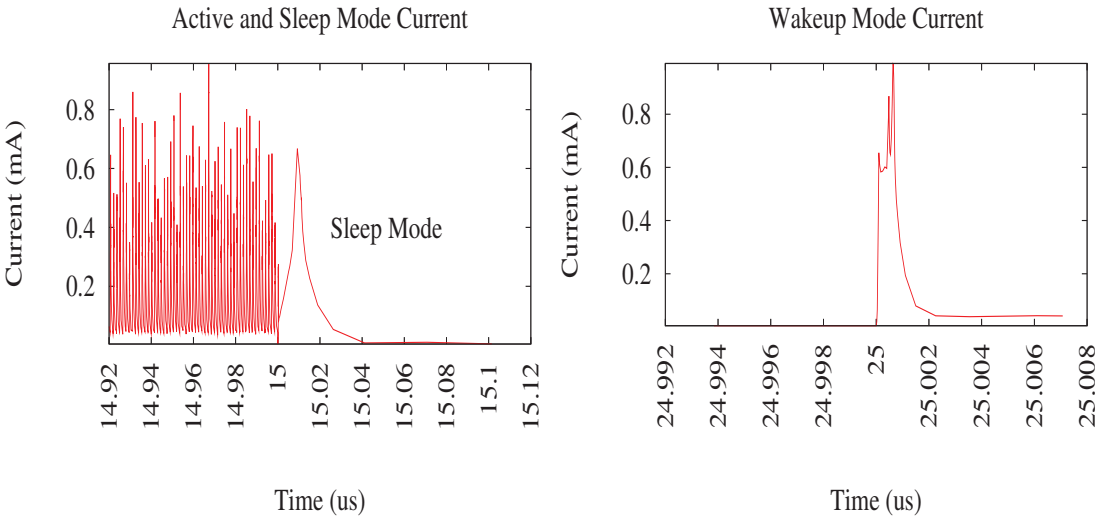


FIGURE 4.7: Current drawn by power-gated KS adder (16-bit precision).

4.6.3 Reducing Simulation Time

To determine average static power in idle mode and other design parameters of power-gated circuit in sleep mode and wakeup mode transitions rapidly, the models proposed in

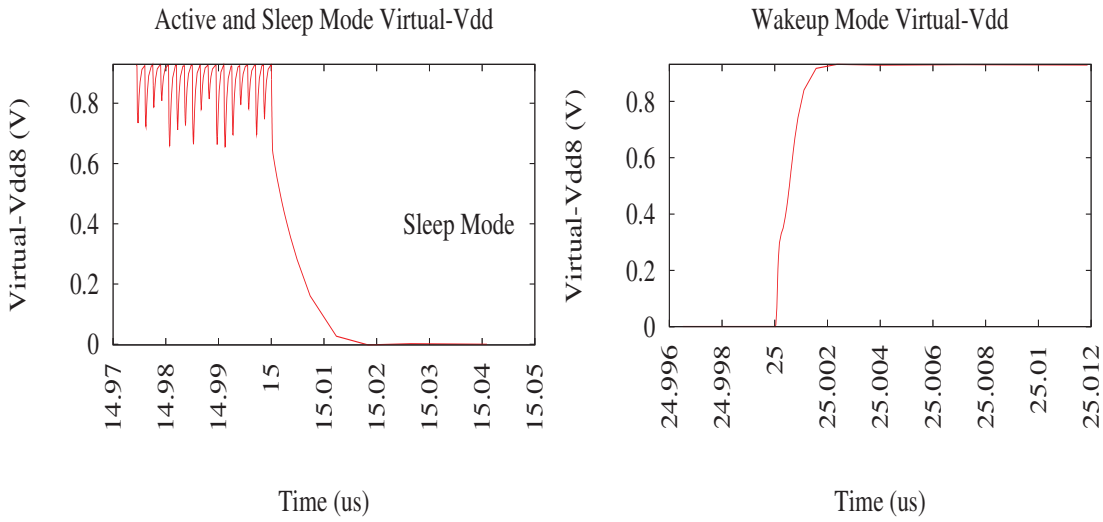


FIGURE 4.8: Virtual-Vdd of C_{N_4} in power-gated KS adder (16-bit precision).

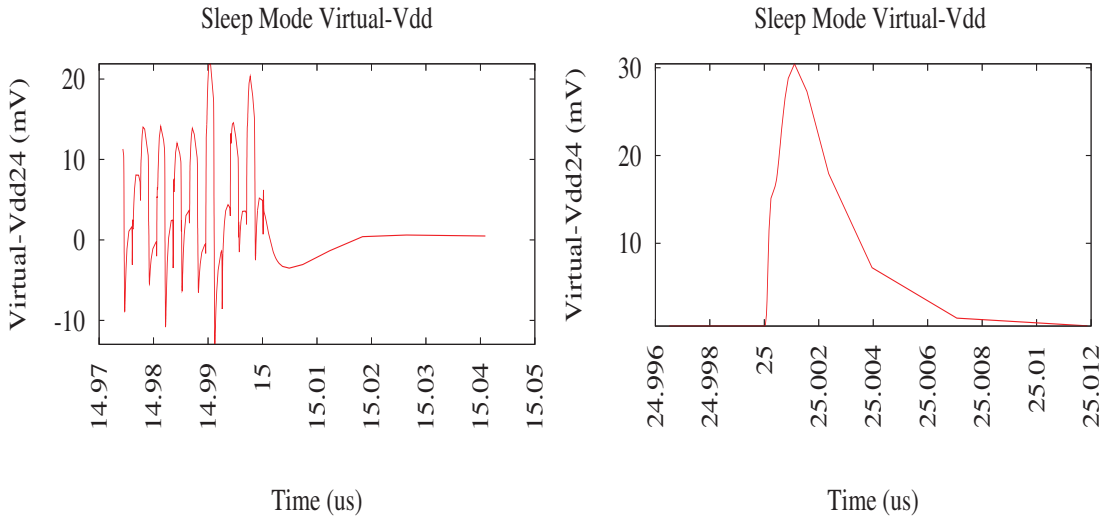


FIGURE 4.9: Virtual-Vdd of C_{N_2} in power-gated KS adder (16-bit precision).

Chapter 3 and Section 3.4 can be used. However the steps for dynamic power estimation shown in Fig. 4.1 involve transistor level simulations with SPICE. For the experimental setup described before, the simulation time using Eldo circuit simulator running a multi-processor simulation was about six hours for each context and 10000 input vectors. In order to reduce time for dynamic power estimation we attempt the classical approach for a comparison. For each input vector, the gate-level netlist is parsed for logic-level transitions at the output of each gate from its value for the previous vector. The total switched capacitance for an input set is calculated as the sum of capacitances of gate nodes that switch logic levels. For N_{input} inputs, let C_{sw} be the total switched capacitance. Then

the average dynamic power in active mode is given by

$$P_{dyn,av} = \frac{C_{sw} V_{dd}^2}{N_{input} T_d} \tag{4.9}$$

where T_d represents the critical path delay from inputs to outputs. The estimated dynamic power using this approach is within 10% of the estimated value using SPICE simulations. An example is given in Table 4.6. This routine was realized in Tcl using PrimeTime Static Timing Analysis tool. Even with a single processor based analysis, the time for power estimation reduced by half. $P_{idle,av}$ was estimated using models proposed in Section 3.4

	SPICE	Model
$P_{dyn,av}$	197μW	177μW
$P_{idle,av}$	62.9μW	63μW

TABLE 4.6: Comparison of power estimation results between SPICE simulations and models in 32-bit KS adder.

4.6.4 Energy Savings Example

Consider the active time period T as defined in Section 4.3.2 and a reconfiguration sequence $\{32(T_1) \rightarrow 24(T_2) \rightarrow 16(T_3) \rightarrow 8(T_4)\}$ where $\sum_i T_i = T$ and $X(T_i)$ indicates X-bit precision configuration for time T_i. From Table 4.1, C_{N_1} is in active mode for T_1 and sleep mode for $T - T_1$. Similarly for $C_{N_2}(T_2$ and $T - (T_1 + T_2))$,.. etc. C_{N_4} is ON for all of T. From Eq. (4.5),

$$E_s = \sum_{i=1}^{4} P_{active_i,av} T_i - \sum_{i=1}^{4} P_{sleep_i,av} \left(T - \sum_{k=1}^{i} T_k \right) - 3E_{wu} - E_{iso}. \tag{4.10}$$

Substituting the values of each term from Table 4.4 for BK adder, using Eq. (4.8) and assuming $T = 60\mu$s, $T_i = 15\mu$s and $E_{wu} = 1$pJ, $E_{iso} = 3.2$pJ, we have $E_s = 28$nJ (36%). For a schedule of $\{16(30\mu s) \rightarrow 8(30\mu s)\}$, average energy savings is 13.5nJ (30%). Typical times for a configuration can range from hundreds of microseconds to several milliseconds.

4.7 Conclusion

A generic method for the design of power-gated arithmetic circuits based on reconfigurable input data widths for dynamically variable-precision arithmetic was proposed in this chapter. The regularity and unidirectional data flow properties of these circuits were exploited to derive a simple logic clustering technique. Results of application of this method to two types of adders indicate that leakage power in unused logic gates

may be reduced by a factor of about 8 despite 9% to 15% increase in area and negligible delay degradation. Substantial savings in dynamic power at lower precisions in power-gated circuits justifies the use of power gating. The power gating overheads on energy savings in active mode were indicated to be used as cost function in the design flow of Fig. 4.1.

Chapter 5

Low Power Reconfigurable Finite State Machines

5.1 Introduction

A finite state machine (FSM) is an abstract representation of a sequential system that works in synchronization with a timing reference signal called clock. An FSM specifies behaviour of the synchronous system in response to its inputs and clock. The system represented by an FSM can be in one of the finite number of states at any given time. A realization of FSM involves mapping behavioural representation of its synchronous system to a network of basic logic gates and storage elements. Modern digital VLSI systems typically involve complex synchronous systems with large networks of logic gates and storage devices. Reconfigurable FSMs are sequential systems whose behaviour may be configured in time.

In this chapter, realizations of reconfigurable FSMs are explored with focus on power gating as a low power technique. The next section describes models of FSM in their reconfigurable form. The next-state and output functions that constitute a finite state machine are defined. Architectures optimized for reconfigurable FSMs are presented in Section 5.3. The hardware resources required to implement a fully reconfigurable FSM are derived in terms of its basic parameters. The complexity of the logic involved is presented from the perspective of reconfigurability of behaviour and scalability of parameters. Power gating opportunities in the proposed architectures are identified for operation of FSM in active mode. Section 5.4 shows various optimizations to reduce logic complexity by supporting only a limited set of reconfigurable FSMs and achieve aggressive leakage power reduction in active mode. An analysis of power consumption of power-gated reconfigurable FSMs is presented in Section 5.5. A restricted yet useful set of well known sequential systems referred to as Linear Sequential Circuits (LSCs)

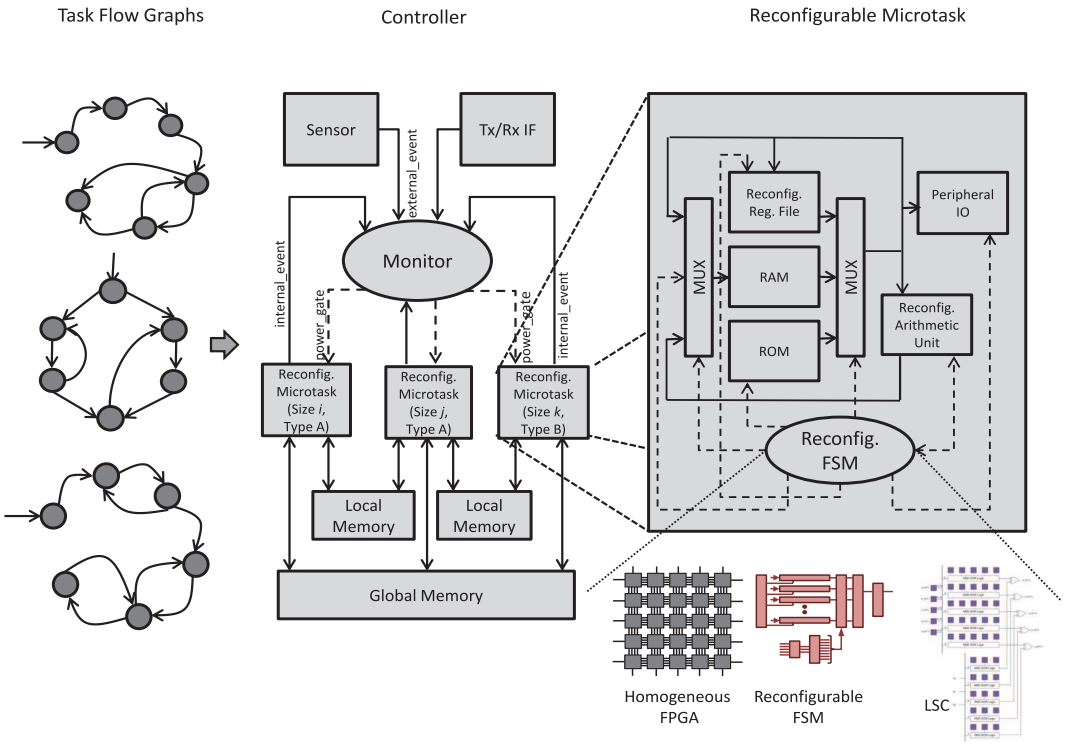


FIGURE 5.1: Microtasks with reconfigurable FSMs.

are presented from a reconfigurability viewpoint in Section 5.7 for completeness. Figure 5.1 shows two sequential circuits, the reconfigurable FSM and LSCs, explored in this chapter in the context of reconfigurable microtasks. Together, the various reconfigurable architectures for FSMs along with those for datapaths, hardwired microtasks and microcontrollers represent a design space for choice of controllers.

5.2 Reconfigurable Finite State Machines

A realization of finite state machine consists of two networks of combinational logic gates denoted by two sets of boolean functions **F** and **G** and a set of storage elements (registers) **S** synchronized by a clock as shown in Fig. 5.2. Let at time unit t, the n primary inputs to the FSM be denoted by the vector $\mathbf{x}(t) = [x_0(t), x_1(t), ..., x_{n-1}(t)]$, the m outputs of FSM by $\mathbf{y}(t) = [y_0(t), y_1(t), ..., y_{m-1}(t)]$ and the state vector of N-bit state register by $\mathbf{s}(t) = [s_0(t), s_1(t), ..., s_{N-1}(t)]$. A time unit is typically the index of a reference edge of the clock. The state vector at time unit $t+1$ is then denoted by $\mathbf{s}(t+1)$ and the notation applies identically to inputs and outputs. In digital logic, since $x_i(t)$, $y_i(t)$ and $s_i(t)$ take values from the set {0,1}, the sets of all inputs, outputs and states consist of 2^n, 2^m and 2^N possible distinct patterns respectively.

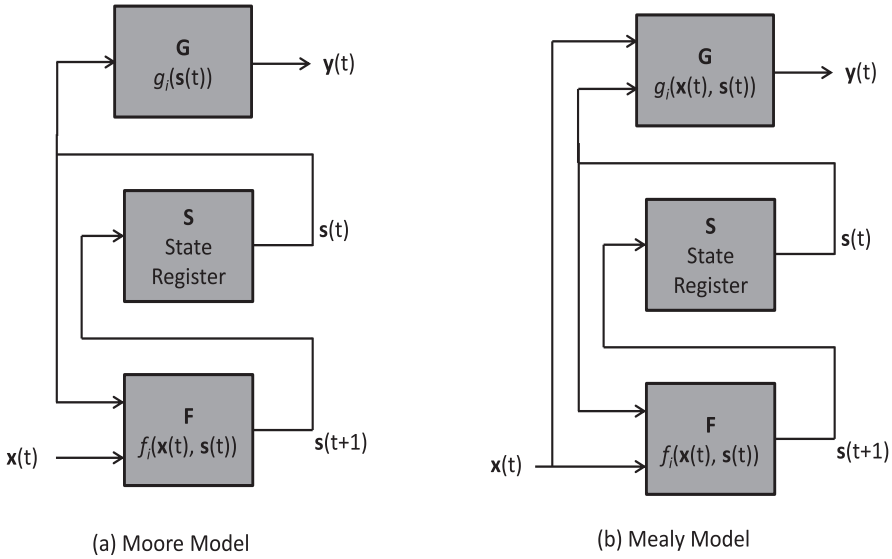


FIGURE 5.2: Moore and Mealy models of finite state machines.

Let f_i denote a boolean function in **F** and g_j denote a boolean function in **G**. The set of next-state functions $s_i(t+1)$ and output functions y_j represented by

$$s_i(t+1) = f_i(\mathbf{x}(t), \mathbf{s}(t)) \quad i = 0, 1, ..., N-1 \tag{5.1}$$

$$y_j(t) = g_j(\mathbf{s}(t)) \quad j = 0, 1, ..., m-1 \tag{5.2}$$

denotes a finite state machine of Moore type [17] (Fig. 5.2(a)) and those represented by

$$s_i(t+1) = f_i(\mathbf{x}(t), \mathbf{s}(t)) \quad i = 0, 1, ..., N-1 \tag{5.3}$$

$$y_j(t) = g_j(\mathbf{x}(t), \mathbf{s}(t)) \quad j = 0, 1, ..., m-1 \tag{5.4}$$

represents a finite state machine of Mealy type [18] (Fig. 5.2(b)). With $n+N$ input variables for f_i, there are a total of $N \times 2^{2^{n+N}}$ possible functions in **F** for N state registers[1]. Similarly, the number of possible output functions for each g_j is 2^{2^N} or $2^{2^{n+N}}$ depending on whether the FSM is of Moore type or Mealy type respectively. Hence **G** can contain a total of $m \times 2^{2^N}$ or $m \times 2^{2^{n+N}}$ functions. The FSM is reconfigurable if the realization of FSM can be configured to support more than one set of boolean functions f_i and g_j across time. The FSM is fully reconfigurable[2] if any of the functions in **F** and **G** can be realized across time. Therefore the logic complexity of a fully reconfigurable FSM is of the order $O(2^{n+N})$.

[1]Although $2^{2^{n+N}}$ functions are the same for each state register bit we index each set with the state register bit so that total number of functions to which next-state logic can be reconfigured is $N \times 2^{2^{n+N}}$.

[2]We use the term full reconfigurability to distinguish from limited reconfigurability as discussed in Section 5.4.

5.3 Architectures Optimized for Reconfigurable FSMs

5.3.1 Next-State Functions

Rewriting the next-state function in Eq. (5.1) and Eq. (5.3) as

$$s_i(t+1) = f_i(x_0, x_1, ..., x_{n-1}, s_0, s_1, ..., s_{N-1}) \tag{5.5}$$

and expressing Eq. (5.5) in terms of well known Shannon expansion [104] of its variables gives

$$s_i(t+1) = x_0' f_i(0, x_1, ..., x_{n-1}, s_0, ..., s_{N-1}) + x_0 f_i(1, x_1, ..., x_{n-1}, s_0, ..., s_{N-1}). \tag{5.6}$$

The binary operation '+' denotes a logical-OR or disjunction. The logical-AND operation or conjunction of two variables (or functions) a and b is denoted by ab. Inversion of a boolean variable a is denoted by a'. By extension to second variable, Eq. (5.6) can be written as

$$\begin{aligned} s_i(t+1) = {} & x_0' x_1' f_i(0, 0, ..., x_{n-1}, s_0, ..., s_{N-1}) + x_0' x_1 f_i(0, 1, ..., x_{n-1}, s_0, ..., s_{N-1}) \\ & + x_0 x_1' f_i(1, 0, ..., x_{n-1}, s_0, ..., s_{N-1}) + x_0 x_1 f_i(1, 1, ..., x_{n-1}, s_0, ..., s_{N-1}) \end{aligned} \tag{5.7}$$

and more generally as

$$s_i(t+1) = \sum_{k=0}^{2^{(n+N-K)}-1} m_k f_i(n(m_k), ..., s_{N-1})_k. \tag{5.8}$$

K corresponds to number of variables on which $f_i(.)_k$ depends after Shannon decomposition. The minterm generated by first $n+N-K$ input variables of the sequence $x_i, ..., s_i(t+1)$ is denoted by m_k. For instance $m_1 = x'_{n+N-K-1} x'_{n+N-K-2} ... x_0$. The binary pattern vector corresponding to minterm m_k is represented by $n(m_k)$. Hence $n(m_1) = 000...01$. For a fixed f_i, the function can be realized physically by a mapping of logic gates but for the FSM to be fully reconfigurable it is necessary to realize $f_i(.)_k$ as a lookup table (LUT) which can be reloaded with function values for reconfiguration. Assuming that each $f_i(.)_k$ can be realized with K-LUTs it can be inferred that the next-state vector $\mathbf{s}(t+1)$ requires the following resources:

1. $N \times 2^{(n+N-K)}$ K-LUTs for N-bit state register,
2. One $(n+N-K)$-to-$2^{(n+N-K)}$ decoder to generate minterms m_k, and
3. $N \times 2^{(n+N-K)}$ 2-input AND gates and N, $2^{(n+N-K)}$-input OR gates or an equivalent combination.

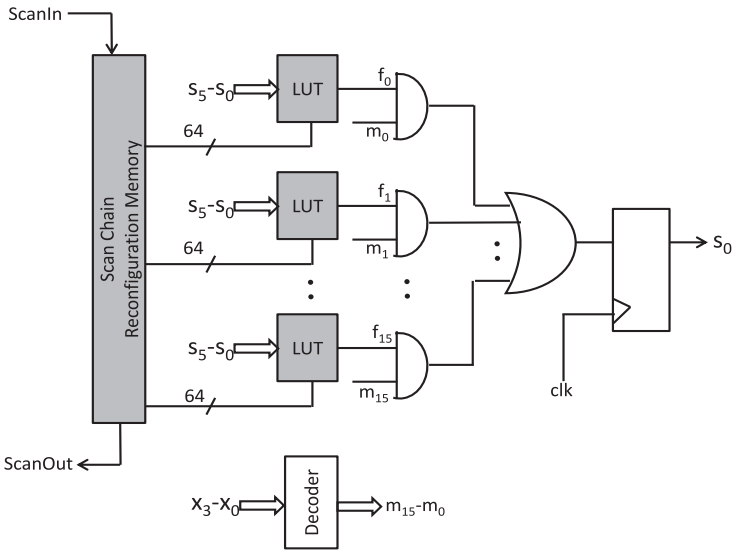


FIGURE 5.3: Next-state function realization for one state-register bit $(N = 6, n = 4, K = 6)$.

The above expressions are valid for $n + N \geq K$. For the case, $n + N < K$, $2^{(n+N-K)}$ must be replaced by 1. A schematic diagram of the architecture is shown in Fig. 5.3 for one state register bit. It is assumed that the FSM has $n = 4$ primary inputs and $N = 6$ state register bits in the figure. The decoder shown in the figure is common for all state register bits whereas the rest of the logic and memory shown must be replicated for other state register bits.

5.3.2 Output Functions

Let the set of boolean functions for FSM outputs be denoted by

$$y_l = g_l(s_0, s_1, ..., s_{N-1}) \tag{5.9}$$

and

$$y_l = g_l(x_0, x_1, ..., x_{n-1}, s_0, s_1, ..., s_{N-1}) \tag{5.10}$$

for Moore type and Mealy type of FSMs respectively, where y_l, $l = 0, 1, ..., m-1$, denote the output variables. Let K_{op} denote the number of inputs of a K_{op}-LUT. Writing Eq. (5.9) similar to Eq. (5.8), we have for the Moore type

$$y_l = \begin{cases} \sum_{k=0}^{2^{N_s}-1} m_k g_l(n(m_k), ..., s_{N-1})_k & K_{op} < N \text{ (Case 1)}, \\ g_l(s_0, ..., s_{N-1})_k & K_{op} \geq N \text{ (Case 2)} \end{cases} \tag{5.11}$$

with $N_s = N - K_{op}$ and writing Eq. (5.10) similar to Eq. (5.8), we have for the Mealy type

$$y_l = \sum_{k=0}^{2^{(n+N-K)}-1} m_k g_l(n(m_k), ..., x_{n-1}, s_0, ..., s_{N-1})_k. \tag{5.12}$$

The resources for FSM outputs in Moore machine can be derived by considering the number of outputs m as

1. $m \times 2^{N_s}$ K_{op}-LUTs for N-bit state register where $K_{op} = N - N_s$ (Case 1) and m K_{op}-LUTs (Case 2)
2. One $(n + N - K)$-to-$2^{(n+N-K)}$ decoder shared with next-state functions (Case 1) and
3. $m \times 2^{(n+N-K)}$ 2-input AND gates and N, $2^{(n+N-K)}$-input OR gates or an equivalent combination (Case 1).

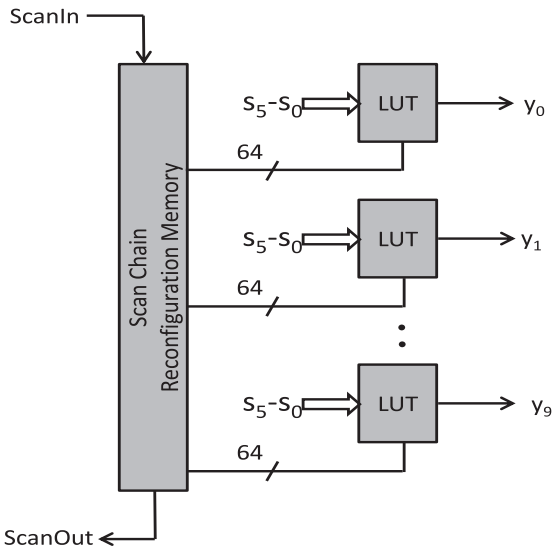


FIGURE 5.4: Output function realization in Moore type FSM (Case 2. $N = 6, m = 10, K = 6$).

Similarly the outputs in a Mealy FSM require

1. $m \times 2^{(n+N-K)}$ K-input LUTs for N-bit state register
2. One $(n + N - K)$-to-$2^{(n+N-K)}$ decoder and
3. $m \times 2^{(n+N-K)}$ 2-input AND gates and N, $2^{(n+N-K)}$-input OR gates or an equivalent combination.

The above expressions are valid for $n+N \geq K$. For the case, $n+N < K$, $2^{(n+N-K)}$ must be replaced by 1. Schematic diagrams of the architecture for FSM outputs in Moore-type

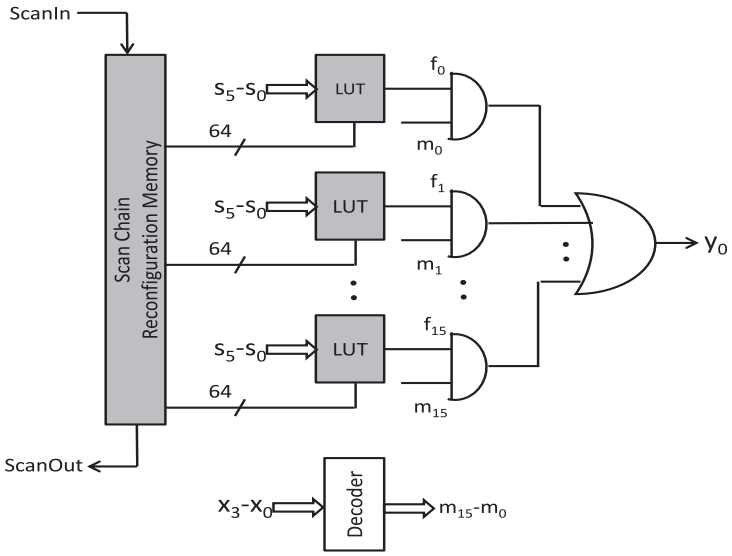


FIGURE 5.5: Output function realization in Mealy type FSM ($N = 6, m = 1, K = 6$).

and Mealy-type FSMs are shown in Fig. 5.4 and Fig. 5.5. It is assumed that the FSM has $n = 4$ primary inputs and $N = 6$ state register bits in the figures. It should be noted that there is a requirement of only one decoder in FSMs of Mealy type and Moore type (Case 2) for all of outputs whereas the rest of the logic and memory structures have to be replicated for other outputs.

Table 5.1 shows the number of LUTs required for full reconfigurability of next-state functions for indicated number of inputs and state register bits. The resources required for reconfigurable logic can be tabulated similarly for output functions of the FSMs. The parameterization of resources in terms of number of inputs, outputs, state register bits and LUT sizes forms the basis for finding power-gating opportunities discussed in Section 5.3.4 and estimating power (and energy) consumption in Section 5.5.

# LUTs	$K = 4$			$K = 6$		
N	$n = 2$	$n = 3$	$n = 4$	$n = 2$	$n = 3$	$n = 4$
1	1	1	2	1	1	1
2	2	4	8	2	2	2
3	6	12	24	3	3	6
4	16	32	64	4	8	16
5	40	80	160	10	20	40
6	96	192	384	24	48	96
7	224	448	896	56	112	224

TABLE 5.1: Number of LUTs required for next-state function for full reconfigurability.

5.3.3 Configuration Bits for Reconfiguration

A number of implementations exist to realize the scan chain reconfiguration memory ranging from simple chain of flip-flops or latches to complex SRAM and scan chain based architectures. Two simple examples based on flip-flops and latches are shown in Fig. 5.6. In eFPGA, the scan chain shown in Fig. 5.6(b) was used so that the subsequent configuration could be stored in the flip-flops for negligible overhead in reconfiguration time. In the proposed architectures the scan chain in Fig. 5.6(a) is used to save leakage power from an additional register at the expense of reconfiguration latency. The total

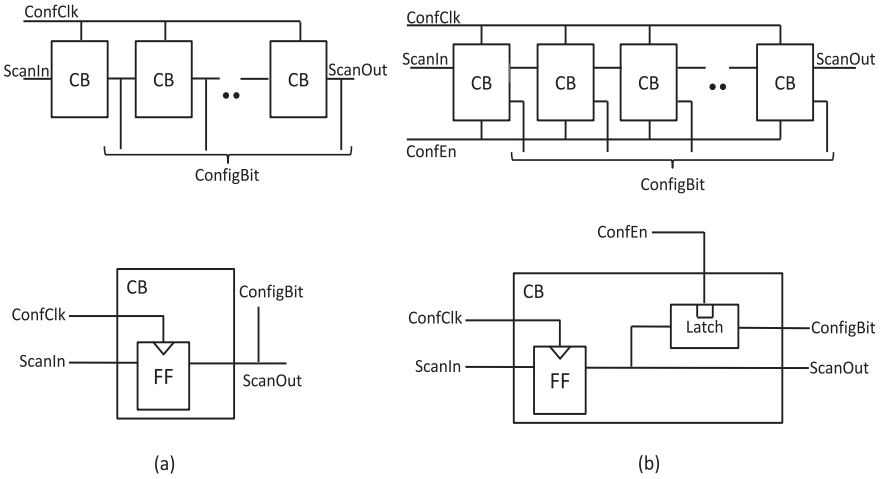


FIGURE 5.6: Scan chain reconfiguration memory of LUTs.

number of configuration bits required for reconfiguration memory for the FSM types discussed before is given in Table 5.2.

Logic Function and FSM Type	Size (bits)
Next-State Functions (Moore and Mealy Types)	$N \times 2^{n+N}$
Output Function Moore Type: Mealy Type:	 $m \times 2^{N}$ $m \times 2^{n+N}$

TABLE 5.2: Number of configuration bits.

5.3.4 Power Gating Opportunities

Low power optimizations of FSMs have generally included clock gating and FSM partitioning based on state transition probabilities to reduce dynamic power. Various state encoding schemes have been investigated to reduce logic switching transitions when the FSM changes its states depending on its current state and primary inputs. In [105, 106] both techniques have been used along with power gating to reduce dynamic and static power in fixed FSMs. Shannon decomposition has been widely used in logic synthesis

algorithms, variable assignment and function mapping for LUTs in FPGAs. Further the separation of input variables into logical complements gives rise to potential of power gating one or the other combinational path in a synthesized circuit. This leads to fine-grained power gating opportunities [107] which is reported to save about 40% of total power in ISCAS85 benchmark circuits. In reconfigurable circuits, power gating needs to be explored at a logic cluster level or in other words, at coarse-grain granularity if benefits are to outweigh costs. This in turn requires careful evaluation of design paramaters at cluster level. In this work we seek to identify power gating opportunities in reconfigurable FSMs at a granularity of LUTs or a group of LUTs to reduce leakage power.

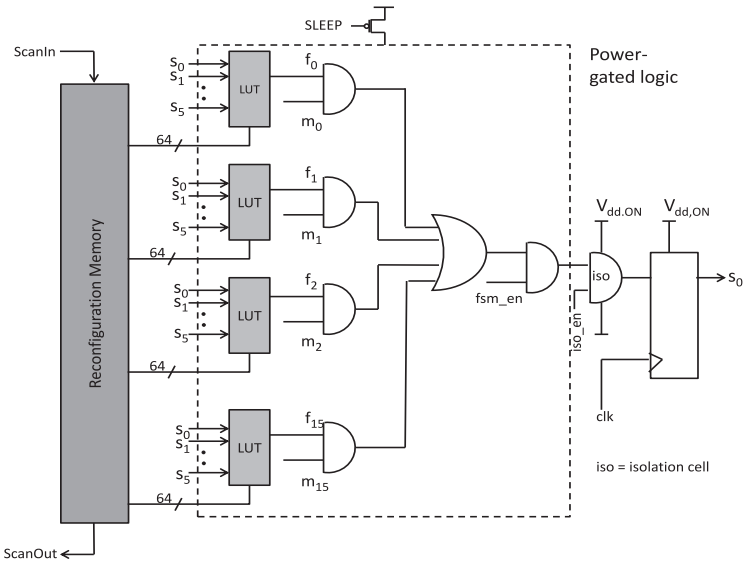


FIGURE 5.7: Power gating opportunity for active mode energy savings in a reconfigurable FSM.

In Section 5.2 it was shown that for a fully reconfigurable FSM architecture the complexity of logic increases exponentially as a function of number of state register bits and primary inputs. This increases total area of the circuit. When such an expansion in area is accepted for flexibility and by logic integration metrics, power consumption due to leakage current becomes a prohibitive factor (Table 3.1). It was shown in the power gating example of Section 3.3.3 that significant power (and energy) savings result from power gating when the complete system is in sleep mode. In this section, power gating opportunities are explored to shut-off power supply to parts of the reconfigurable FSM with the proposed architecture in active modes of operation. In other words we take advantage of the fact that parts of the logic circuit may remain in sleep mode in a particular configuration at a given time. For example, when a reconfigurable FSM with resources for six state register bits and fifteen outputs is configured with an FSM of five state register bits and ten outputs, then the logic clusters related to one of the

state register bits and five outputs may be power-gated to suppress leakage power during that configuration. It is clear from Fig. 5.3 to Fig. 5.5 that the architecture is highly regular and the combinational logic of LUTs provide a power gating opportunity. A typical schematic of the architecture with a power gating opportunity identified is shown in Fig. 5.7. A single control signal ($SLEEP$) manages the power-up and power-down modes of a set of LUTs and associated gates. An isolation cell is inserted between power-gated domain and always-on logic. In case of the reconfigurable FSM the decoder, control signals and state register bits lie in the always-on power domain.

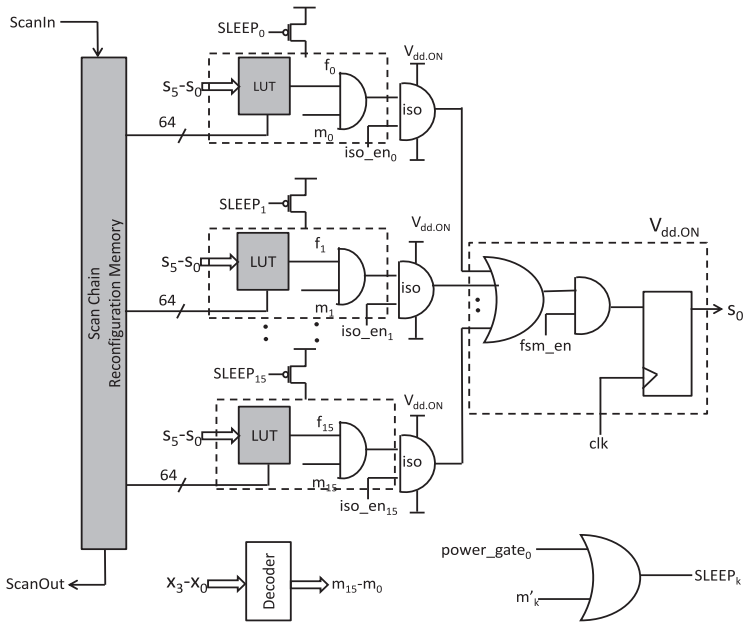


FIGURE 5.8: Power gating opportunity for aggressive active mode energy savings in a reconfigurable FSM.

Further power gating opportunities have been identified in Fig. 5.8 at the granularity of a LUT and associated AND gates. This network of sleep transistors requires as many control signals as the number of LUTs. The power gating operation can be explained as follows. When the reconfigurable FSM is configured as a particular FSM, its operation depends on some of the primary inputs that possibly vary slowly and therefore, depending on the values of those inputs all but one of the minterms evaluate to logic 0. It is also possible that some of the primary inputs are unused for certain state register bits in which case the corresponding minterms always evaluate to 0. In other words some of the LUTs may remain in power-gated states even in the active modes as long as the minterm does not change its value which in turn depends on values of primary inputs. Such decoder outputs can function as control signals ($SLEEP_k = power\_gate_i + m'_k$ in Fig. 5.8 and Fig. 5.9) to power-gate the respective LUT while also eliminating the need for a separate controller. This power gating method entails a higher cost in terms of area due to isolation cells [19] and wakeup overheads like wakeup time and energy [20]. A

quantitative treatment of energy consumption and energy savings due to power gating in active and sleep modes of operation of the complete circuit is given in Section 5.5.

Similar power gating schemes with varying granularities can be designed for output functions. We consider an example of a Moore type of FSM with $m = 10$ outputs, number of state register bits $N = 8$ and LUT input size $K_{op} = 6$. This corresponds to Case 1 in Eq. 5.11. A schematic of the intended architecture for one output is shown in Fig. 5.9. It should be noted that $K_{op} < N$ and therefore $N - K_{op}$ state register bits have been borrowed as inputs to the decoder in the same architecture for power gating as for next-state functions.

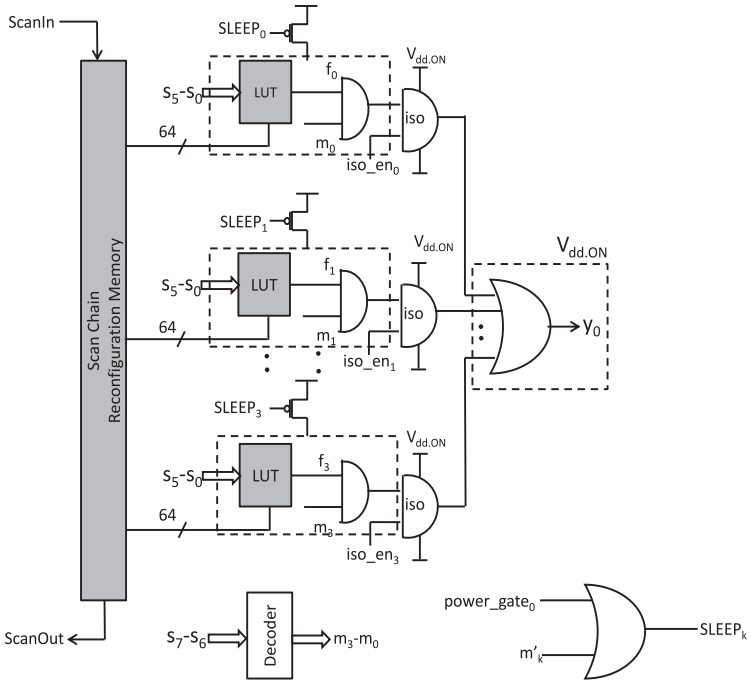


FIGURE 5.9: Power gating opportunity for aggressive active mode energy savings in reconfigurable FSM output logic.

5.3.5 Observations on Power-Gated Architectures

A key observation on the architectures with power gating at LUT level of granularity proposed above is that at any time instant only one LUT per state register bit and output is active while the rest are in sleep mode. In other words the total power consumption at any given time depends on only $N + m$ LUTs and associated logic irrespective of exponential dependency of number of LUTs on number of inputs or decoder outputs. The additional sources of power dissipation are static power of configuration bits and isolation cells.

It should be noted that for every change in inputs to the decoder the active minterm changes and hence all LUTs but the one associated with this minterm is turned off. Hence a change in inputs poses an overhead in terms of wakeup time and wakeup energy.

Energy savings will result if the inputs change at a lower frequency so as to offset the wakeup energy overhead. Further, for FSMs whose number of state registers and outputs are less than the maximum possible with the architecture, the unmapped LUTs can be completely powered-off to reduce static power dissipation. These observations are quantified in Section 5.5.

5.4 Limited Reconfigurability in FSMs

5.4.1 Motivation

It was seen in Section 5.2 that the logic complexity increases exponentially with the number of primary inputs and state register bits. As the number of inputs (primary inputs and current state register bits) of a next-state function increased the number of minterms required to be ANDed with LUT outputs increased and consequently the number of LUTs also increases. In reality however multiple boolean functions in an FSM realization may be dependent on same number of inputs but the inputs themselves may be different. Hence it is very likely that the number of required minterms may be much less than the total number of minterms required for full reconfigurability. In other words, if for a class of FSMs, the minterms m_k in the Shannon decomposition evaluate to logic 0 due to primary inputs or state register bits, then the conjunction $m_k f_i(.)_k$ is not required to be evaluated, thus eliminating need for LUT or the minterm itself. The realization then reduces to having LUTs whose f_i may evaluate to logic 1 and those minterms that are required for conjunction with f_i. This reduces to the problem of selecting only those inputs whose minterms are required for each boolean function (next-state or output function).

Example 1: Consider a boolean function of four variables x_0, x_1, x_2, x_3 given by

$$f = x_0' x_1' x_2' x_3' + x_0 x_1' x_2 x_3' + x_0 x_1 x_2 x_3' + x_0' x_1' x_2 x_3. \tag{5.13}$$

Using Shannon's expansion Eq. (5.13) can be written as

$$f = x_0' x_1' (x_3) + x_0 x_1' (x_2 x_3') + x_0' x_1 (0) + x_0 x_1 (x_2 x_3'). \tag{5.14}$$

It is clear that the LUT associated with minterm $x_0' x_1$ does not influence the value of the function f. Therefore the LUT may be used in conjunction with another minterm. Next consider a scenario (e.g., a different configuration) with $x_0 = 0$ always. In other words, in a different configuration f does not depend on variable x_0. Clearly although the minterms $x_0' x_1'$ and $x_0' x_1$ may evaluate to 1 depending on x_1, the output of LUTs associated with minterms $x_0 x_1$ and $x_0 x_1'$ does not affect the value of f. Therefore, in both

scenarios, either the associated LUTs may be power-gated or the concerned minterms can be avoided for conjunction.

Example 2: Three FSMs with different parameter level specifications are given in Table 5.3. We assume 6-LUTs being used for realization of the FSM. The entries in third column of the table indicates the following: in the example FSM of firBasic the state register function $s_0(t+1)$ has a fanin of three primary inputs x_0, x_1, x_2 and a fanin of seven state register bits $s_0, ..., s_6$. Assuming that the inputs of associated LUTs are $s_0, ..., s_5$, the 16 minterms for full reconfigurability are given by s_6, x_0, x_1, x_2. Similarly, $s_3(t+1)$ has no fanin of primary inputs but a fanin of seven state register bits $s_0, ..., s_6$. The realization of $s_3(t+1)$ would then require only two minterms generated by s_6. Inferences may be made on the same lines for other example FSMs and fanin properties of state registers and outputs. In order that the three FSMs be supported by the reconfigurable architecture the specification corresponding to maximum number of resources needs to be identified for each state register bit and output. Once the resources are identified, a power gating schedule may be derived.

The maximum number of minterms required for each state register bit is given by

FSM	N	n for each $s_i(t+1)$	N for each $s_i(t+1)$	[m, Fanin of y_l]
firBasic	7	(3,3,3,0,0,0,0)	(7,7,7,7,7,7,7)	(21,7)
Crc16	7	(4,4,4,0,0,3,0)	(7,7,7,7,7,7,7)	(17,7)
receiveData	6	(3,3,0,0,0,0)	(6,6,6,6,6,6)	(23,6)

TABLE 5.3: Limited reconfigurability examples

$\{s_0(32),\ s_1(32),\ s_2(32),\ s_3(2),\ s_4(2),\ s_5(16),\ s_6(2)\}$ where $s_i(P)$ denotes that for the function $s_i(t+1)$, P minterms are required. Similarly, for each of the outputs $y_j(Q)$ $j = 0, 1, ..., 22$, $Q = 2$ for a Moore type of FSM. This in turn decides LUTs, decoder, memory and power gating resources for the reconfigurable FSM that supports limited number of FSMs. In other words a tradeoff is derived between implementations of most likely FSMs and flexibility of fully reconfigurable FSMs. While this tradeoff reduces the number of LUTs and hence power-gated logic clusters, it introduces input selector logic for each next-state function in a Moore type FSM.

5.4.2 Input Selector-Decoder Design and Overheads

The example described above represents a top-level resources identification method when specifications of a set of FSMs are given at a parameter level. This procedure may be utilized for microtask generation given a task flow graph where control flow in tasks are treated as FSMs. In this section a reconfigurable FSM architecture for control of datapath in a microtask is derived for a limited reconfigurability specification. In the context of limited reconfigurability the primary inputs are denoted by n_I and the number of outputs of the input selector by n. We fix the number of state registers $N = 7$ and

number of primary inputs $n_I = 5$ as an example for discussion and consider a scenario where the next-state functions depend on only three inputs but not identically. Therefore the number of minterms required for conjunction would be 8 reducing the number of LUTs per state register bit. Similarly the outputs are fixed at $m = 23$. We consider a Moore type of FSM. The objective of this section, along with Section 5.5, is to show scalability of architecture from limited reconfigurability with an upper bound of specifications to full reconfigurability, both in terms of implementation and power estimation.

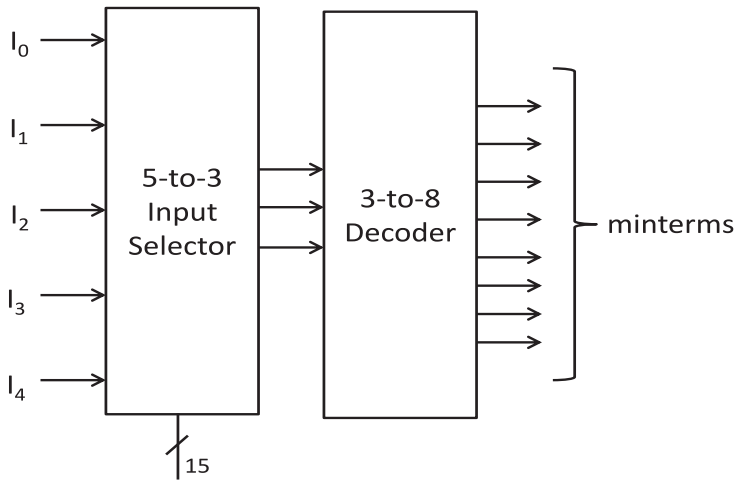


FIGURE 5.10: Input selector-decoder logic to select minterms of dependent inputs.

A cascade of 5-to-3 input selector and 3-to-8 decoder with eight minterms as outputs is shown in Fig. 5.10. The purpose of this circuit is to select only eight minterms for eight LUTs per next-state function or output function. The input selector is able to route three out of five inputs to any of the three outputs to have eight minterm outputs from the decoder. Clearly only next-state functions that have minterms that may evaluate to 1 and can be chosen by the limited selection available can be implemented by the reconfigurable FSM. Additionally for this specification a next-state function may have no more than eight minterms that can evaluate to logic 1 to be implementable by the proposed reconfigurable FSM. One such cascade of input selector and decoder cascade is required per state register bit while the number of LUTs can be reduced by 75% compared to full reconfigurability FSM and also have the same level of granularity in power gating.

The output functions also require $(N - K_{op})$-to-$2^{N-K_{op}}$ decoders when $N > K_{op}$ (Moore type of FSM, Case 2) and $(n + N - K_{op})$-to-$2^{n+N-K_{op}}$ decoders when $(N + n) > K_{op}$ (Mealy type of FSM).

5.4.3 Overall Architecture

The overall architecture for a power-gated reconfigurable FSM is shown in a schematic form in Fig. 5.11. It can be inferred that the architecture is regular and scalable based

on the parameters discussed in the previous sections. Three distinct advantages result

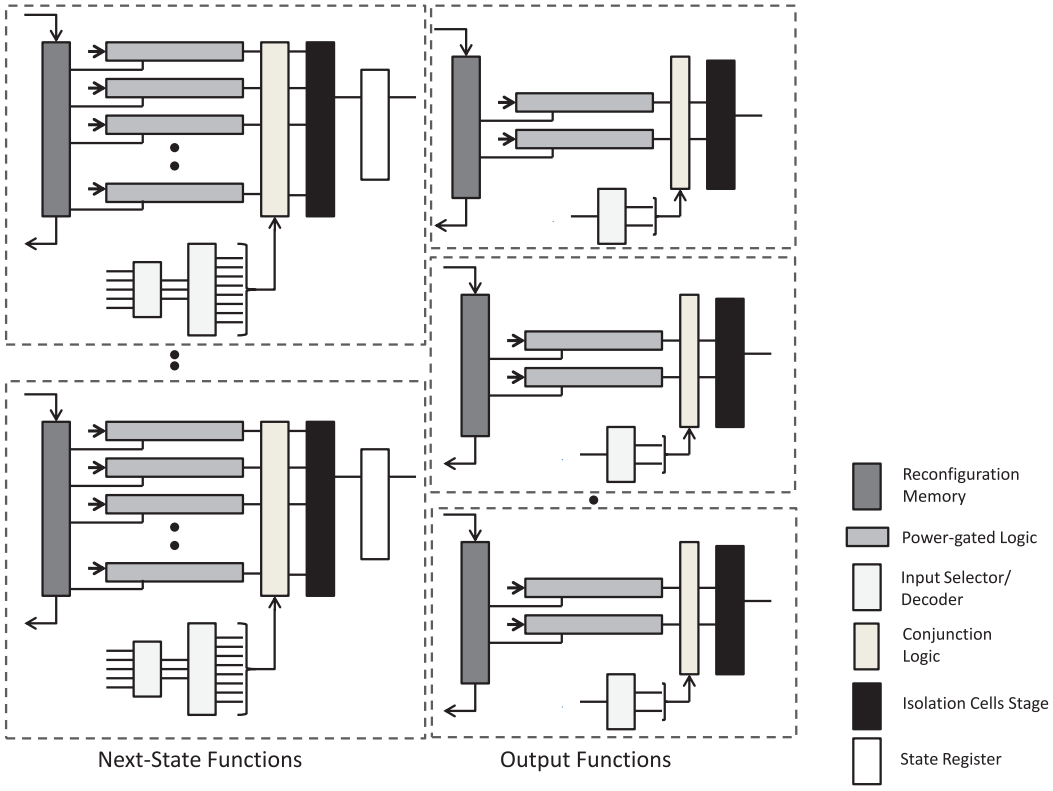


FIGURE 5.11: Schematic diagram of the overall architecture of scalable power-gated reconfigurable FSM.

with the proposed architecture when compared to the homogeneous reconfigurable array based FSM described in Chapter 2.

1. The complexity of routing required for each boolean function implemented is considerably reduced as the switches are localized to input selector-decoder cascade. This results in better utilization of hardware area.

2. The unused LUT logic at any time is always power-gated thus resulting in power savings. The power gating network is similar for all LUTs resulting in scalability of sleep transistor networks. In the homogeneous array of CLBs, optimal location of sleep transistor networks cannot be decided for a configuration as it is solely based on mapping of resources.

3. There is better control over functional mapping of FSM to resources available. This also leads to development of simpler mapping tools and FSM partitioning techniques.

5.5 Power Estimation in Reconfigurable FSMs

In order to estimate power consumption in power-gated FSMs with limited reconfigurability we use the method followed in Section 2.5. The basic units of the architecture are identified and are characterized for various parameters like static power, average dynamic energy, design parameters of power-gated circuits, etc. The configuration bits, LUT logic with AND gates, input selector-decoder cascade, isolation cells and OR logic, and state register bit form the basic units of architecture apart from the sleep transistor. They are then used to estimate power consumption of the FSM. The total average power consumption of the power-gated architecture for a particular mapping of FSM is given by

$$\begin{aligned} P_{total,FSM} = (N_{FSM} + m_{FSM})P_{static,LUT} + N_{CB}P_{static,CB} + \\ 2^N(N_{FSM}2^{n-K} + m_{FSM}2^{-K_{op}})P_{static,iso} + \\ N_{FSM}(P_{static,IPD} + P_{static,SR}) + \\ N_{FSM}E_{dyn,IPD}f_{inp,av} + (N_{FSM} + m_{FSM})(E_{wu} + E_{dyn,LUT})f_{inp,av} \end{aligned} \tag{5.15}$$

where N_{FSM} and m_{FSM} are number of mapped state register bits and outputs, N_{CB} denotes the total number of configuration bits, $P_{static,LUT}, P_{static,CB}, P_{static,SR}, P_{static,iso}$ and $P_{static,IPD}$ denote the static power of K-LUT loigc, a configuration bit register, state register bit, isolation cell and input selector-decoder respectively. Further, $E_{dyn,LUT}$ and $E_{dyn,IPD}$ represent the average dynamic energy components of LUT logic and input selector-decoder due to changes in inputs. Here both state register bits and primary inputs are combined and an average activity factor for transitions is assumed by means of $f_{inp,av}$, the average rate of change of inputs. Since there may be a wakeup transition of at most one per state register bit and output, we consider the worst case with wakeup transition for all state register bits and outputs to account for wakeup energy E_{wu}. Equation (5.15) can be rewritten to show scalability in power estimation relative to high level parameters of the FSM defined in Section 5.2 as below:

$$\begin{aligned} P_{total,FSM} = N_{FSM}[P_{static,LUT} + P_{static,IPD} + P_{static,SR} + 2^{n+N-K}P_{static,iso} + \\ (E_{wu} + E_{dyn,LUT})f_{inp,av}] + N_{CB}P_{static,CB} \\ +m_{FSM}[P_{static,LUT} + 2^{N-K_{op}}P_{static,iso} + (E_{wu} + E_{dyn,LUT})f_{inp,av}]. \end{aligned} \tag{5.16}$$

In this estimation, the dynamic power due to reconfiguration is not considered as it is assumed that the time span of each configuration is long enough to ignore the contribution of energy consumption due to reconfiguration compared to other components. Further energy due to capacitances of routed interconnect lines normally estimated at the physical design level have been ignored. Unlike in the architecture of eFPGA, the changes in logic

levels due to function evaluation are not propagated along such long interconnects within a single configuration time unit. In the architectures proposed the interconnect routing is localized to a regular and small area and 87% of the logic and consequently the connected wires is power-gated at any given time. Hence the contribution of wire capacitances to dynamic power is negligible, which is a clear advantage with respect to FPGA-based solutions.

5.5.1 Characterization of LUTs, Input Selector and Decoders

We consider a 6-LUT for lookup table implementations in reconfigurable FSM [108]. The combinational logic of 6-LUT is synthesized with a set {nand2, nor2, xor2, ivx4} of gates without loss of generality and their areas and path delay are determined. Since the configuration bits are in the always-on power domain, they are considered separately from the LUT logic even though they form inputs to the LUT logic. The leakage power is determined at $V_{dd} = 1.0$V by applying the polynomial model in Section 3.4 and taking into account the value at each input of constituent logic gates. The various design parameters of power-gated circuits are also determined from the gate level models presented in Chapter 3. Switching energy for a single LUT is determined by applying 10000 sets of random inputs and computing the average total switched load for one input set at a supply voltage of 1.0V as

$$E_{dyn_{LUT}} = \frac{V_{dd} r_1}{N_{input}} \sum_{i=1}^{N_{input}} C_{sw_i,LUT} \tag{5.17}$$

where $C_{sw_i,LUT}$ is the total switched capacitance of the LUT per input set and r_1 is the steady state Virtual-V_{dd} as obtained from Eq. (3.17). The parameters are listed in Table 5.4.

Similar estimation of parameters for decoders of several sizes and input selector-decoder logic is shown in Table 5.5. Since decoders and input selector logic lie in the always-on power domain the design parameters related to power-gated circuits are not determined.

5.5.2 Static Power in State Register, Configuration Bits and Isolation Cells

For N state registers the total leakage power is simply $N_{FSM} P_{static,SR}$. Similarly for the number of configuration bits N_{CB} as determined from Table 5.2, the static power consumption is given by $N_{CB} P_{static,CB}$. The nominal leakage power of one such flip-flop from power-optimized Family B (cf. Chapter 2, Section 2.5.2) cells has been determined to be 2.185nW at $V_{dd} = 1.0$V and 100°C. The area of a register is $7.8\mu m^2$. The static

Power-Gated K-LUT Parameter	Value K=4		Value K=6	
Area	$94.6\mu m^2$		$354.6\mu m^2$	
$P_{static}(V_{dd} = 1V)$	3.48μW		12.56μW	
Average Switching Energy per State or Input Transition	0.017pJ		0.059pJ	
Sleep Transistor Width (W)	0.54μm	12μm	0.54μm	12μm
Steady State Virtual-$V_{dd}(r_1)$	994mV	999mV	980mV	998mV
V_{sleep}(150ns)	15mV	14.1mV	21mV	15mV
V_{sleep}(500ns)	10mV	10.6mV	10.6mV	10.6mV
I_{leak}(150ns)	20.45nA	15nA	169nA	72nA
I_{leak}(500ns)	0.08pA	0.03pA	5.3pA	0.25pA
Wakeup Time T_{wu}	1.26ns	0.06ns	4.8ns	0.23ns
Wakeup Energy E_{wu}	0.09pJ	0.09pJ	0.28pJ	0.28pJ

TABLE 5.4: K-LUT parameters for power estimation (K=4, K=6).

Decoder	Area (μm^2)	$P_{leak}(\mu$W)	$E_{dyn,IPD}$ (fJ)	t_d (ns)
2 to 4	11.4	0.489	1.555	0.04
3 to 8	35.87	1.278	7.068	0.16
4 to 16	56.16	1.644	7.893	0.15
5 to 32	112.32	2.852	12.063	0.22
6 to 64	202.28	4.361	14.664	0.36

TABLE 5.5: Area, leakage power consumption (P_{leak}), switching energy (E_{sw}) and critical Path (t_d) comparisons against decoders of different sizes.

	Area (μm^2)	$P_{leak}(\mu$W)	$E_{dyn,IPD}$ (pJ)	t_d (ns)
5 × 4 to 16	226	7.97	0.051	0.45

TABLE 5.6: Input selector-decoder logic.

FSM	N_{FSM}	n	m
abs	5	3	9
Crc8	6	3	16
receiveData	6	3	23
Crc16	7	4	19
firBasic	7	3	21

TABLE 5.7: Parameter level specifications of FSMs.

power of an isolation cell in the standard cell library is similarly determined to be 1.482nW at V_{dd}=1.0V. The area of an isolation cell as obtained from the library is $2.6\mu m^2$

5.5.3 An Analysis of Power Estimation

The different components of power and energy consumption in the power-gated architectures proposed for reconfigurable FSMs are obtained as described above for FSM

examples in Table 5.7 as in the case of eFPGA (Table 2.5) and are presented in Table 5.8 to Table 5.10. Two examples of circuits with limited reconfigurability are taken by varying number of inputs and hence number of LUTs per next-state function. It should be noted that in the former case the FSM example Crc16 is not implementable. However in both cases that static power and average dynamic energy in active mode depends on only one LUT per next-state function and ouput function irrespective of increase in area. The contribution of isolation cells and input selector-decoder logic to static power and dynamic energy is negligible compared to other sources of power and energy consumption.

FSM Specification $N = 7, n = 3, m = 23$	$P_{static,FSM}$ (μW)			
	CB and SR[a]	LUT Logic	Input Selector-Decoder	Isolation Cells
abs	13.88	175.84	42.14	0.15
Crc8	13.88	276.32	42.14	0.15
receiveData	13.88	364.24	42.14	0.15
firBasic	13.88	351.68	42.14	0.15

[a]using power optimized registers

TABLE 5.8: Static power in power-gated FSM architecture with limited reconfigurability ($N = 7, n_I = 3, m = 23$).

FSM Specification $N = 7, n = 4, m = 23$	$P_{static,FSM}$ (μW)			
	CB and SR[a]	LUT Logic	Input Selector-Decoder Logic	Isolation Cells
abs	21.68	175.84	55.72	0.23
Crc8	21.68	276.32	55.72	0.23
receiveData	21.68	364.24	55.72	0.23
Crc16	21.68	326.56	55.72	0.23
firBasic	21.68	351.68	55.72	0.23

[a]using power optimized registers

TABLE 5.9: Static power in power-gated FSM architecture with limited reconfigurability ($N = 7, n_I = 4, m = 23$)

An estimate of total power dissipation and energy consumption per operation for each FSM is computed at two frequencies (f_{clk}) of 20 MHz and 100 MHz and given in Table 5.11. In this analysis a pessimistic assumption of $f_{inp,av} = f_{clk}$ is made. The total average power is determined as

$$P_{total,FSM} = P_{static,FSM} + E_{dyn,FSM} f_{clk} \tag{5.18}$$

FSM Specification $N = 7, n = 4, m = 23$	$E_{dyn,FSM}$ (pJ)		
	LUT Logic	Input Selector-Decoder Logic	Wakeup Energy
abs	0.826	0.255	3.92
Crc8	1.298	0.306	6.16
receiveData	1.711	0.306	8.12
Crc16	1.534	0.357	7.28
firBasic	1.652	0.357	7.84

TABLE 5.10: Dynamic energy estimation in power-gated reconfigurable FSM architecture

and the total average energy consumption per operation (clock cycle) is obtained from

$$E_{op,FSM} = \frac{P_{static,FSM}}{f_{clk}} + E_{dyn,FSM}. \tag{5.19}$$

A comparison with an optimistic power estimation in eFPGA given in Table 2.6 shows that the total power for mapped FSMs is significantly lesser with optimized and power-gated architectures. As an example, for 'firBasic' the estimated power is about 13.29mW on an eFPGA architecture with optimistic assumptions whereas it is about 1.4mW with the proposed power-gated reconfigurable architecture at an operating clock frequency of 100 MHz.

	Total Power of FSM (mW)		Energy per Operation (pJ)	
	f_{clk} = 20 MHz	f_{clk} = 100 MHz	f_{clk} = 20 MHz	f_{clk} = 100 MHz
abs	0.35	0.75	17.76	7.54 (7.92)[a]
Crc8	0.51	1.13	25.46	11.30 (11.45)
receiveData	0.64	1.46	32.23	14.56 (14.54)
Crc16	0.59	1.32	29.38	13.21 (13.22)
firBasic	0.63	1.41	31.31	14.14 (14.10)

[a]Scaled estimates from SPICE simulations

TABLE 5.11: Total average power and energy per operation for FSMs on power-gated architecture

5.5.4 Experimental Setup and Validation

In Section 5.5.3, power and energy in FSMs mapped onto reconfigurable architectures were estimated using gate-level and cell-level characterizations of basic architectural units described before. By integrating models for design parameters in power-gated circuits, the method provides a way to estimate power (and energy) in both always-on and power-gated parts of the overall architecture in a unified way rapidly. Except for a simple characterization of average energy per operation in a 6-LUT, the power estimation method was essentially spatial in nature so that Eq. (5.16) could be used. Power estimation in

the reconfigurable FSM using a fully temporal approach would require significantly long circuit simulation times as the number of nodes in the complete logic circuit increase exponentially with basic parameters. Hence, a spatio-temporal approach involving transistor level simulations of architectural components using a commercial circuit simulator is followed here.

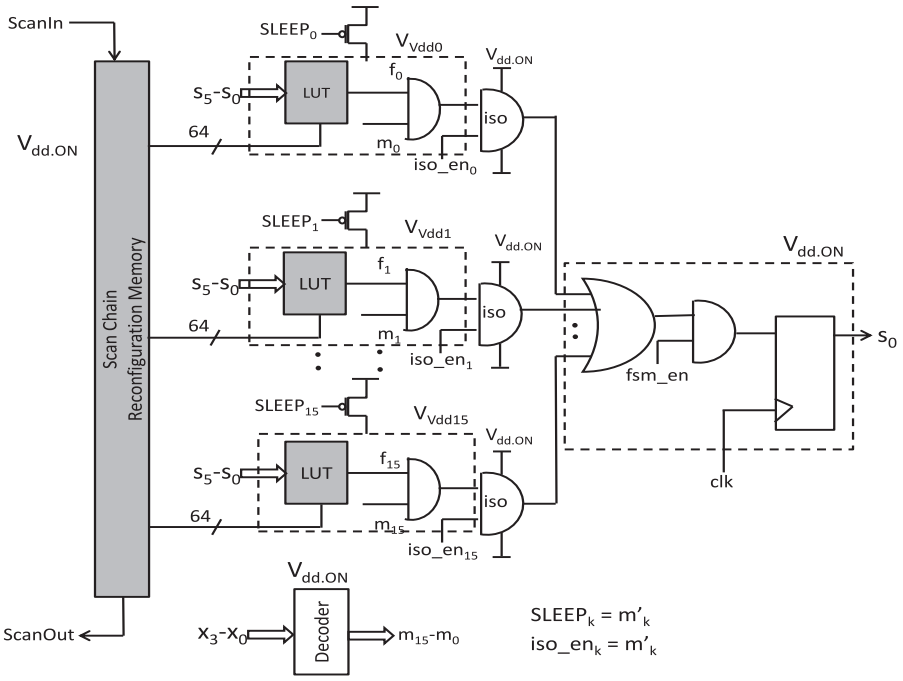


FIGURE 5.12: An architectural module of overall power-gated reconfigurable FSM used in experimental validation.

The gate-level netlists along with sleep transistor network are converted into a transistor-level schematic and a transient analysis is performed using Eldo circuit simulator and BSIM4 SPICE models provided by the 65nm technology library. A power profile is obtained by simulation of an architectural unit for one next-state function shown in Fig. 5.12. The configuration bit registers are loaded with arbitrary data in scan chain fashion in the first part of simulation. In the second part, present-state inputs to LUTs and primary inputs to the decoder are applied at a clock period of 10ns. This represents FSM operation at a clock frequency f_{clk} = 100MHz.

To determine average power and total energy consumption over the period of transient analysis, the current drawn from supply nodes for always-on power domain and power-gated domains are determined. Hence average energy per input pattern (or operation) is obtained. The total current drawn in the two domains at any instant of time includes all sources of energy consumption *viz.*, charging current, short circuit current and leakage currents. The estimates obtained from a temporal analysis for one architectural unit is scaled by the number of such units determined by the top-level parameters of FSM to be implemented with the reconfigurable fabric leading to the spatio-temporal

approach discussed above. Further the evolution of virtual supply voltage for each of the power-gated clusters (V_{Vdd0} to V_{Vdd15} in Fig. 5.12) as they transition through sleep - wakeup - active modes of operation is shown in Fig. 5.13. From the figure it can be verified that at any given time only one of the LUT logic clusters is in active mode of operation.

5.5.5 Sources of Errors in Power Estimation

The results of SPICE level simulations as scaled for each FSM is given in Table 5.11. It should be noted that power estimation in both approaches is pessimistic unlike a fully temporal approach because the correlation in power dissipation due to state transitions described by the FSM is not considered. Also, short circuit currents due to switching of outputs of CMOS logic gates described in Eq. (1.3) are not considered in the first approach based on models. Provided that the transition times of inputs and outputs of a logic gate are balanced, the dynamic power due to short circuit currents is found to be within 10%-15% of dynamic power due to charging and discharging of capacitive nodes [24]. Further variations in estimation of wakeup energy for an arbitrary power-gated cluster as shown in Table 3.9 can have an impact on determination of energy per operation metric for the overall architecture and needs to be addressed further.

5.5.6 Effects of Wakeup Overheads and Performance-Power Trade-offs

An important objective of using a lower technology node is to gain hardware integration area for more flexibility (or functions) and higher performance. Since this entails increased leakage power, a trade-off needs to be derived. In nodes that rely on energy harvesting partially or completely, performance must be traded for lower power by reducing f_{clk} and addressing $P_{static,FSM}$ while in nodes where energy efficiency or battery lifetimes are important, higher clock speeds and eventual shutdown of logic into sleep mode is an appropriate choice. This leads to determination of maximum clock frequency supported by the logic circuit. In the architectures proposed above, the critical path for FSM clock is given by

$$T_{clk} = T_{LUT} + T_{IPD} + T_{wu} + T_{iso} + T_{su} \tag{5.20}$$

where T_{clk} denotes the minimum clock period, $T_{LUT}, T_{IPD}, T_{iso}, T_{wu}$ and T_{su} represent logic delay in LUT, input selector-decoder logic, isolation cell and setup time of the state register respectively. For logic clusters of small sizes and adequately sized sleep transistors T_{wu} can be made small. Also, each next-state function output is delayed by an amount equal to wakeup time of LUT logic cluster and represents a latency in terms of clock cycles.

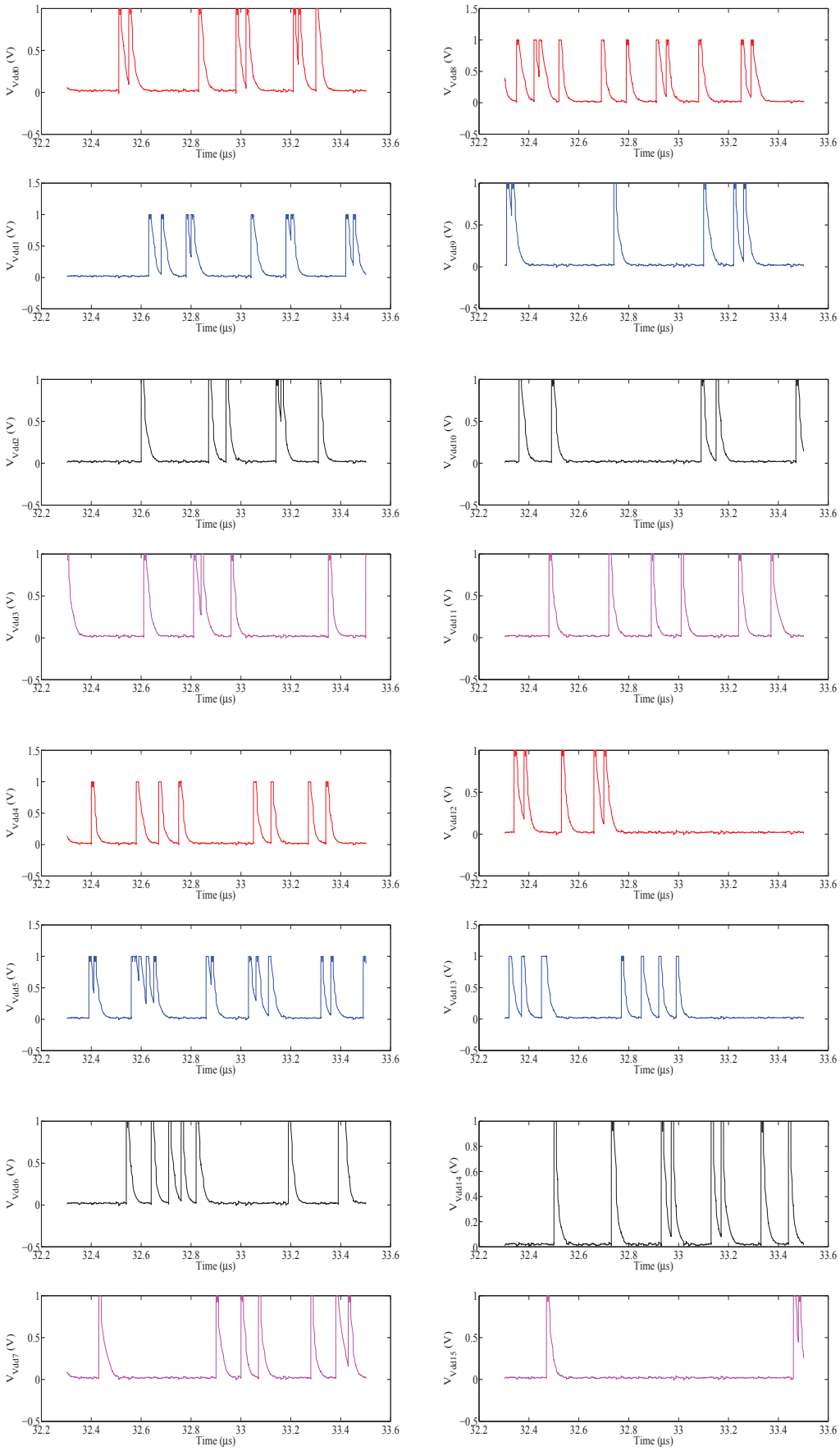


FIGURE 5.13: Snapshot of Virtual-V_{dd} nodes of 16 LUT logic clusters from SPICE simulations for random input data at 100 MHz.

An important aspect of the proposed architectures that take advantage of aggressive power gating in active mode is the wakeup energy due to sleep-to-active mode transition as inputs change. Any change in the inputs will cause a change in one of the minterms leading to wakeup of only one LUT logic cluster. Even so, frequent changes in inputs causes an increased wakeup energy overhead. It can be inferred from Table 5.10 and Table 5.11 that at a high frequency of operation, upto 39% of energy consumed per operation is for wakeup transition. Power gating is useful when the overhead incurred due to wakeup energy is less than the achievable energy savings due to power gating. Clearly this puts a limit on the frequency of changes in the inputs. Wakeup energy can be reduced by reducing the size of clusters. In the proposed architectures, the size of the LUT may be reduced leading to power gating at a finer granularity. Smaller clusters also require sleep transistors of smaller sizes. However this increases the number of power-gated clusters as also increasing the size of input selector-decoder block and hence T_{IPD} and $P_{static,IPD}$, but decreasing T_{wu}. In the proposed power-gating scheme the input selector-decoder block can be seen as an active mode power gating controller. Thus the granularity of power gating determines energy and delay trade-off. Clustering the LUT logic can then be seen as an optimization problem as described in Sections 3.4.6 and 3.4.7. We point out here that the models proposed in Section 3.4 are useful for fast estimation and have been used here.

5.6 Cost of Area

The power-gated architectures for reconfigurable FSMs had a bias for power reduction and therefore an expansion in area was allowed for maximum parallelism in evaluation of Shannon expansion functions. This also ensured that wakeup energy is restricted to one LUT per state register bit and output function. Area estimation is important to compare with alternate realizations of FSMs and its realization in advanced technology nodes. To estimate the total area the method used for power estimation is followed. The area of the basic components of the architecture are determined by actual design with standard cells from the library. Table 5.12 gives the area of individual blocks for Moore type of FSM with N=7, n_I=5, n=4 and m=23.

Reconfig. FSM Architecture Component	Area (μm^2)
Configuration bits and State Registers	79578.3
LUT Logic and Associated Gates	56026.8
Input Selector-Decoder	1582
Isolation Cells	410.8
Sleep Transistor Area	113.76
Total Area	137710.8

TABLE 5.12: Area of power-gated reconfigurable FSM

5.7 Linear Sequential Circuits

In Section 5.2, finite state machines were specified by two sets of functions *viz.*, next-state functions and output functions of variables on which boolean algebra applied. Each boolean variable was assumed to take values from the set {0,1} and the functions were obtained by means of finite number of binary operations, conjunction (logical-AND) and disjunction (logical-OR), and an unary operation complement (inversion) of the boolean variables. The functions were realized physically by digital logic gates whose behaviours could be specified by boolean algebraic functions. In this section we consider a class of sequential circuits where input, output and state variables take values from the field $GF(2)$ and the vectors $\mathbf{s}(t) \in GF(2^N)$, $\mathbf{x}(t) \in GF(2^n)$ and $\mathbf{y}(t) \in GF(2^m)$. Any extension field $GF(2^p)$ is a vector space of dimension p over field $GF(2)$. If the next-state functions and output functions can be denoted as linear transformations given by the mappings $\mathbf{s}(t+1) : GF(2^N) \times GF(2^n) \rightarrow GF(2^N)$ and $\mathbf{y}(t) : GF(2^N) \times GF(2^n) \rightarrow GF(2^m)$ or $\mathbf{y}(t) : GF(2^N) \rightarrow GF(2^m)$ the two sets of functions can be written as [109, 110]

$$s_i(t+1) = \sum_{j=0}^{N-1} a_{ij} s_j(t) + \sum_{j=0}^{n-1} b_{ij} x_j(t) \tag{5.21}$$

$$\begin{cases} y_i(t) = \sum\limits_{j=0}^{N-1} c_{ij} s_j(t) + \sum\limits_{j=0}^{n-1} d_{ij} x_j(t) & \text{(Mealy Type)} \\ y_i(t) = \sum\limits_{j=0}^{N-1} c_{ij} s_j(t) & \text{(Moore Type)} \end{cases} \tag{5.22}$$

where $a_{ij}, b_{ij}, c_{ij}, d_{ij} \in GF(2)$ and the binary operation $+$ denotes addition modulo 2 and products of the form $a_{ij}s_j(t)$ denotes multiplication modulo 2 of $s_j(t)$ by constant $a_{ij} \in GF(2)$. In this section, the operation $+$ is equivalent to logical exclusive-OR (XOR)[3] whereas the product ab is equivalent to logical-AND boolean operation. We choose to use this convention consistent with the existing literature. The sequential circuits represented by Eq. (5.21) and Eq. (5.22) are referred to as Linear Sequential Circuits (LSCs). The theory of linear sequential circuits has foundations in Linear Algebra and Matrix Theory. Since our primary interest is in physical realizations of LSCs, we do not describe connections between vector spaces and boolean algebra as used here and in Section 5.2. Some of the original work can be found in [111, 112]. The two equations Eq. (5.21) and Eq. (5.22) can be written in matrix form as

$$\mathbf{s}(t+1) = \mathbf{A}\mathbf{s}(t) + \mathbf{B}\mathbf{x}(t) \tag{5.23}$$

$$\begin{cases} \mathbf{y}(t) = \mathbf{C}\mathbf{s}(t) + \mathbf{D}\mathbf{x}(t) & \text{(Mealy Type)} \\ \mathbf{y}(t) = \mathbf{C}\mathbf{s}(t) & \text{(Moore Type)} \end{cases} \tag{5.24}$$

[3]It should be noted that $+$ in Section 5.2 denoted logical-OR.

where the size of state transition matrix $\mathbf{A}$ is $N \times N$ and the size of $\mathbf{B}$ is $N \times n$. Similarly the sizes of matrices $\mathbf{C}$ and $\mathbf{D}$ are given by $m \times N$ and $m \times n$ respectively.

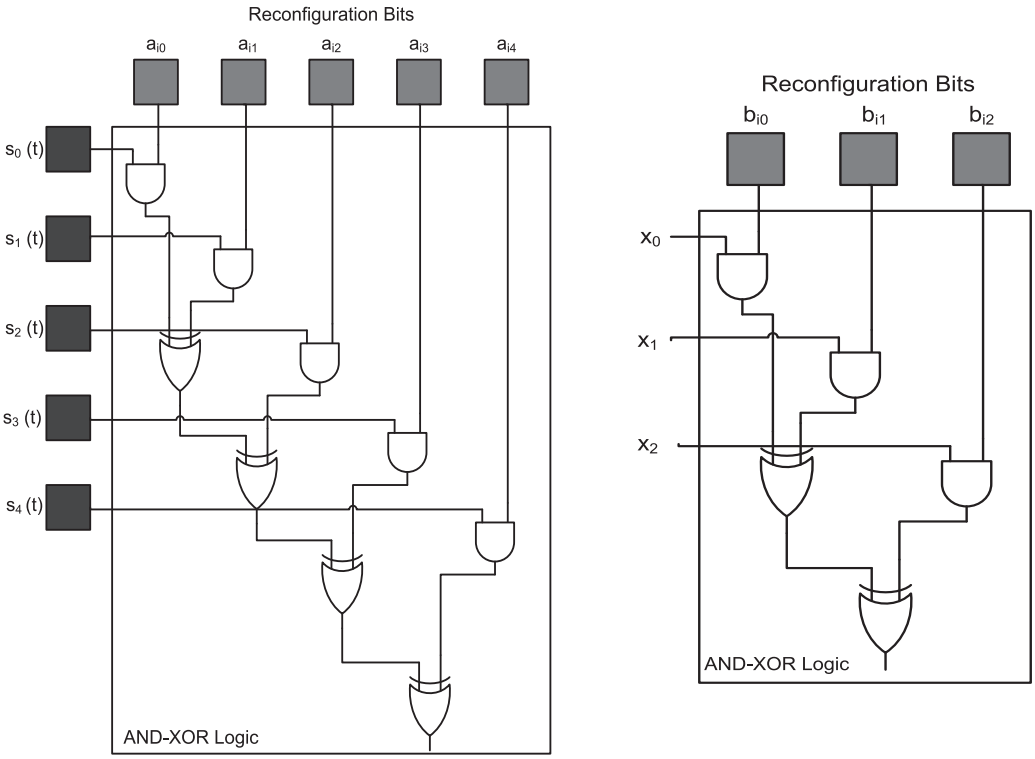


FIGURE 5.14: Basic logic structures to evaluate partial matrix multiplications in Eq. (5.23).

The next-state and output functions in Eq. (5.21) and Eq. (5.22) can be realized with simple XOR and AND gates if it is assumed that the entries of the matrices are a part of configuration scan chain of size $N^2 + N(n + m)$ for Moore type of LSCs and $N^2+N(n+m)+mn$ for Mealy type of LSCs. No power gating is proposed to be employed at a fine granularity of gate level. The basic logic structures for realizations of next-state functions in a LSC are shown in Fig. 5.14 for $N = 5$ and $n = 3$. An architectural schematic for next-state vector is given in Fig. 5.15. Similar architectures can be derived for output functions of both Moore and Mealy models. The number of 2-input AND gates and XOR gates required to realize the direct form of equations (5.21) and (5.22) can be trivially determined to be $N^2+N(n+m)$ and $N^2+(m+n-2)N-1$ respectively. Table 5.13 gives the resources required to implement an LSC and an estimate of area and power.

LSCs have applications in error control coding, pseudorandom number generation, polynomial arithmetic and design of counters. Two of these examples namely, CRC encoder/decoder and counters to realize 'wait' states, were identified in microtasks considered in previous sections. However, further work is required to identify potential applications of LSCs and use them in the context of microtask-based controllers in WSN nodes.

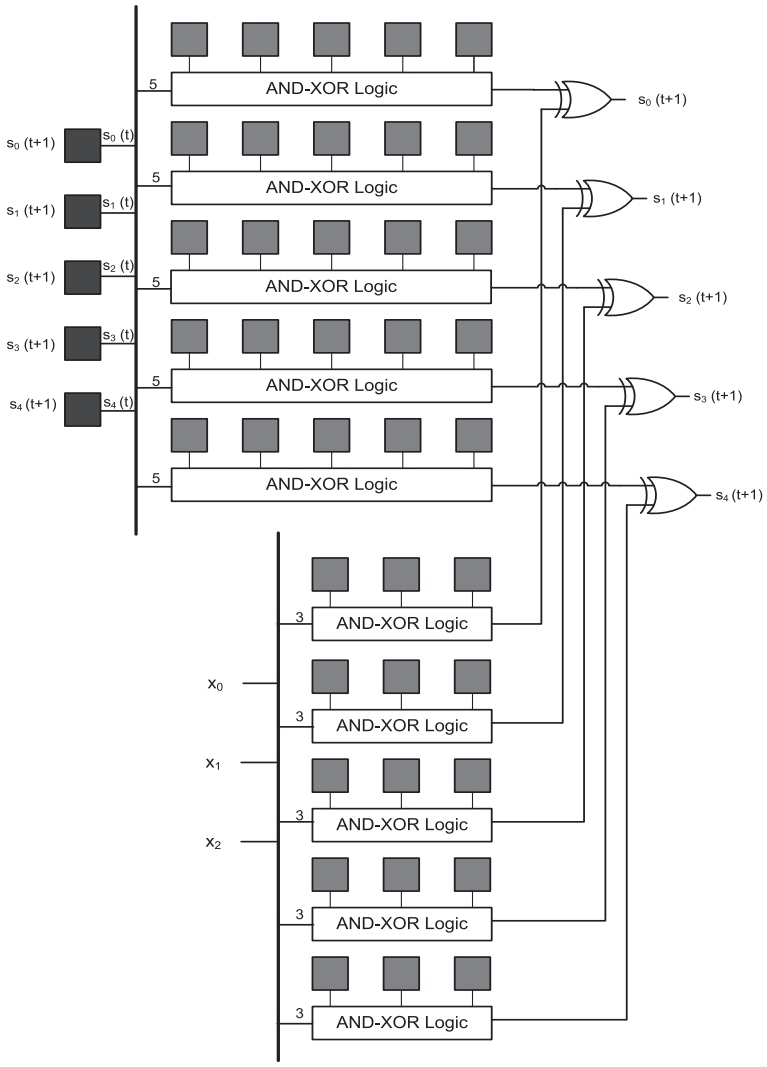


FIGURE 5.15: Schematic for next-state function in a linear sequential circuit.

N=7, n=4, m=23 Resources	#	Area (μm^2)	Static Power
Matrix Entries (Configuration Bits)[a]	238	1856.4	520nW
AND, XOR Gates	468	1695.2	153.41μW
State Register Bits[a]	7	72.8	0.595nW
Total		3624.4	154μW

[a] using power optimized registers

TABLE 5.13: Resources required for LSC and its area and static power estimates.

5.8 Conclusion

Architectures for reconfigurable finite state machines that were optimized based on Shannon decomposition of next-state and output functions of FSMs were presented in this

chapter. Power gating opportunities were explored and limited reconfigurability architectures with power consumption significantly less than that of eFPGA were presented. It was noted that the architectures are easily scalable and have a static and dynamic power consumption of only one LUT logic per next-state function and output irrespective of the total number of LUTs required for the reconfigurable FSM. Also, aggressive leakage reduction in active mode could be achieved by power gating without the need of a separate power gating controller. From characterizations of basic blocks of the architecture, it was inferred that the static power of configuration bit and wakeup energy of power-gated clusters are critical in achieving overall power and energy savings. Finally a brief discussion of linear sequential circuits for a class of FSMs that can be linearized was presented. A key aspect to be addressed is the extent to which FSMs for reconfigurable microtasks can be linearized.

Chapter 6

Conclusions and Perspectives

6.1 Overview

In this thesis, reconfigurable hardware was explored to introduce flexibility in microtask based controllers for WSN nodes. General reconfigurable systems like FPGAs are beset with high power (and energy) consumption problems due to interconnection networks. Switching and long routing networks are primarily responsible for poor area utilization in FPGAs. They are also not amenable to low power techniques like coarse grain power gating to reduce leakage power. A common thread in our exploration of reconfigurable architectures for low power was to identify and power-gate unused logic for aggressive leakage power savings in active mode of operation. In reconfigurable FSMs, the logic structure was exploited to keep the complexity of interconnections and selector logic to the minimum required. The architectures are scalable and the modular architectural units can be easily scaled for larger designs. Power estimation is equally scalable. In adders for datapath, the essential structure of parallel carry and sum generation was retained across several precisions, but the circuit was partitioned for variability in precision and power gating unused logic.

Power gating is an invasive technique. The design parameters must be carefully considered in the design of power-gated circuits. Models were proposed for rapid estimation of wakeup overheads and certain other parameters of interest at gate level. The models are helpful especially in an iterative context where the design parameters have to be evaluated repeatedly for candidate logic clusters as in an optimization procedure. Models for design parameters can serve as cost functions in a synthesis engine. While most of the commercial EDA tools incorporate support for insertion of power gating structures, to the best of our knowledge there is minimal support for automated exploration of power gating opportunities.

To conclude, we examine energy efficiency of the proposed power-gated architectures and control techniques *vis-à-vis* the cost of flexibility at a microtask level. We highlight

some of the limitations and propose ways to address them in future work.

6.2 Energy Efficiency

A metric to measure energy efficiency of different realizations is the energy per instruction. A comparison between low power microcontroller-based realizations of controllers and hardwired implementation of microtasks was presented in [1] as part of a previous work in the CAIRN lab. In order to determine the position of reconfigurable microtasks in the design space the metric is computed for the microtasks used in this work. We consider a 16-bit datapath and hence a 16-bit power-gated adder of Chapter 4. The energy per operation of a microtask $E_{op,MT}$ is given by

$$E_{op,MT} = \frac{1}{f_{clk}}(P_{static,FSM} + P_{static,adder} + P_{static,RF}) + E_{dyn,FSM} + E_{dyn,adder} + E_{dyn,RF} \tag{6.1}$$

where $P_{static,adder}$, $P_{static,RF}$, $E_{dyn,adder}$, and $E_{dyn,RF}$ represent static power of adder, register file and energy per operation of adder and register file respectively. Other quantities have their usual meanings. The total energy for execution of a task is then

$$E_{task} = N_{states} E_{op,MT} \tag{6.2}$$

where N_{states} is the number of operations or state transitions required to execute the task. Then, the equivalent energy per instruction is determined as

$$E_{eff} = \frac{E_{task}}{N_{inst}}. \tag{6.3}$$

Table 6.1 shows the metric for microtasks under consideration. In this exercise, a register file of size 16 × 16 has been considered while RAM and ROM blocks are not used. It can be inferred from the table that the energy efficiency of reconfigurable microtasks in terms of energy per instruction lies in between that of a low power microcontroller and hardwired microtask under the stated conditions.

The reconfigurable microtasks considered in this context can function at more than 200 MHz in the 65nm technology node. The operation of microtasks at such a frequency would reduce the energy per operation but at the same time increase the average power. For an architecture to evolve towards autonomous function the power demand should be typically less than 0.1-1mW for a sensor node. In general, average power and energy per operation can be traded depending on the application.

Microtask	Equivalent Energy per Instruction (pJ/Inst.)					
	openMSP430 [1][a]		Reconfigurable Microtasks[b]		16-bit Hardwired Microtasks [1][c]	
	N_{inst}	E_{eff}	N_{states}	E_{eff}	N_{states}	E_{eff}
Crc8	30	163	71	31.60	71	8.1
receiveData	66	230	332	83.53	332	15.7
Crc16	27	170	73	41.27	73	9.3
firBasic	58	179	168	46.90	168	26.1

[a] 130nm, @16MHz
[b] 65nm, multiple V_{th} cells, @100MHz, 16×16 register file, no SRAM
[c] 65nm, std.-V_{th} cells

TABLE 6.1: Equivalent energy per instruction in three realizations of node controllers.

6.3 Cost of Flexibility

A significant cost of flexibility is the increase in area compared to hardwired microtasks. From a comparison with results in [1] it can be inferred that the increase in area of a reconfigurable microtask is about 19 times compared to its hardwired counterpart. Table 6.2 gives a comparison of area estimates of hardwired microtasks, the proposed reconfigurable microtask, eFPGA-like array with 217 CLBs and openMSP430 microcontroller. Increase in area needs to be seen with two viewpoints. In this work, the proposed architectures are such that any FSM that meets the upper bound of specifications of the reconfigurable FSM can always be mapped unlike in eFPGA or hardwired microtasks. Secondly, from a controller's perspective, several hardwired microtasks may need to be

Microtask	Hardwired Microtask [1](μm^2)	Reconfigurable Microtask (μm^2)	eFPGA (μm^2) (217 CLBs)	openMSP-430(μm^2)
Crc8	3097	140522.2	1076871	22141
receiveData	2858			
Crc16	3102			
firBasic	7164			

TABLE 6.2: Comparison of areas of 16-bit hardwired and reconfigurable microtasks.

integrated and controlled by a system monitor to be able to realize a task flow graph. A WSN application would typically require about 40 to 50 tasks. The total area of a controller designed using the approach of [46] would be the sum of areas of all microtasks and associated system monitor and memories. In principle, a reconfigurable microtask can be used in place of hardwired microtasks by time-multiplexing tasks at the controller level. The node controller would still require a system monitor for reconfiguration and overall control. The advantages of power gating are present in both active and standby modes of operation.

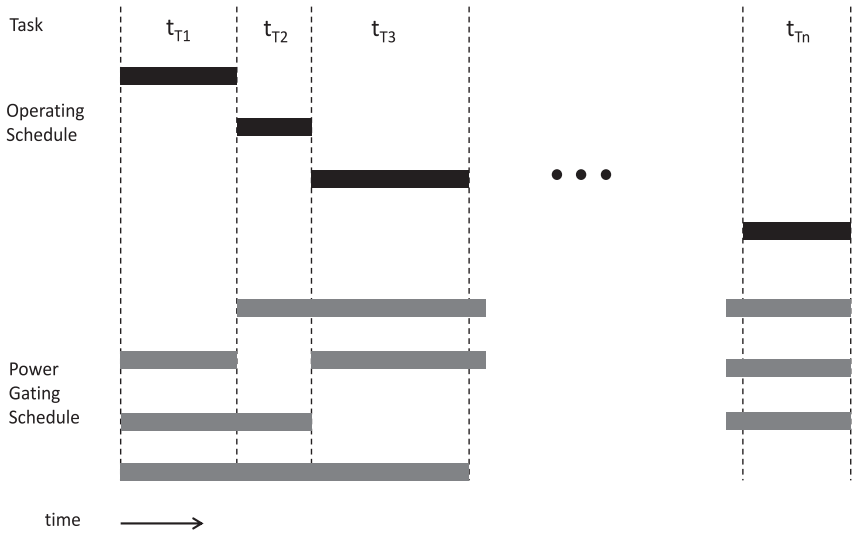


FIGURE 6.1: Operating and power gating schedules for hardwired microtasks as proposed in [1].

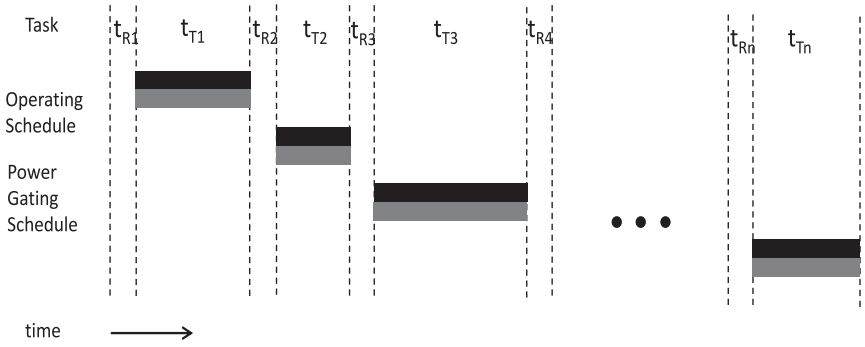


FIGURE 6.2: Operating and power gating schedules for reconfigurable microtask.

The operating schedule and power gating schedule for mapped tasks in both scenarios are shown in Fig. 6.1 and Fig. 6.2 respectively. A constraint to satisfy would be that the FSM corresponding to a task must meet the specifications of the reconfigurable FSM. Alternately, the tasks in the task flow graph must be partitioned in such a way that the related FSMs can always be mapped on to the reconfigurable FSM. A system-level analysis would be required to examine if it would be feasible. The work described in this thesis has been restricted to analysis at a microtask level. Further a thorough analysis of linear sequential circuit implementations from a reconfigurability viewpoint could throw light on classes of useful FSMs that may be implemented with much lesser complexity.

The objective of technology migration in CMOS SoCs has been to enhance integration density for more features and higher performance. Therefore an area cost in one technology node may not be an important cost in more advanced technology nodes. On the contrary, static power that may not be a serious problem in older technologies

may become important in nanoscale circuits. This guides the priorities for design of architectures.

6.4 Future Work

6.4.1 Power Efficient Reconfiguration Mechanisms

The primary drawback of LUT-based flexible circuits is the large reconfiguration memory required to store function values exhaustively. As seen from Table 5.12 it contributes to highest area. Inspite of power optimized registers used for power estimation, substantial power (and energy) was attributed to configuration register bits as they must always remain in ON state. A parallel implementation of the FSM was considered that warranted a full-length configuration chain. However a useful approach at architecture level is to investigate alternate reconfiguration mechanisms based on on-chip memories. This would have a cost with respect to latency and finally on performance as next-state and output function evaluations have to be pipelined.

6.4.2 Circuit-Level Optimizations

LUT decoding logic contributed to the second highest area in the proposed architectures. The logic was implemented with CMOS gates. Pass transistors as an alternate logic style can be studied to implement the logic. It would require less than half the number of MOS devices as that of CMOS logic gates. Also pass transistors are in principle power-efficient with negligible leakage current. However as the number of LUTs increase exponentially with state register bits and outputs for flexibility, it is important to quantify leakage currents due to sneak paths that may be caused by configuration bit patterns. In the case of power-gated architectures discussed in this work, any power (and energy) consumption in LUT decoding logic was only linearly dependent on number of state registers and output functions. A disadvantage with pass transistors is that regenerative buffers are required in large LUT logic circuits to overcome logic degradation effects due to transistor resistance drops. A specific logic style to consider could be sense amplifier-based pass transistor logic [113].

Publications

Journal Publications

1. V. D. Tovinakere, O. Sentieys, and S. Derrien, "A polynomial based approach for wakeup time and energy estimation in power-gated logic clusters," *Journal of Low Power Electronics*, vol. 7, no. 4, pp. 482–489, December 2011.

Conference Publications

1. T. D. Vivek, O. Sentieys, and S. Derrien, "Wakeup time and wakeup energy estimation in power-gated logic clusters," in *Proceedings of the 24th International Conference on VLSI Design*, Chennai, India, January 2011, pp. 340–345.

2. V. D. Tovinakere, O. Sentieys, and S. Derrien, "A semiempirical model for wakeup time estimation in power-gated logic clusters," in *49th ACM/IEEE Design Automation Conference*, 2-7 June 2012, pp. 48–55.

Bibliography

[1] M. A. A. Pasha, *System-Level Synthesis of Ultra Low Power Wireless Sensor Network Node Controllers: A Complete Design Flow.* PhD Thesis, 2010.

[2] W. Weber, J. Rabaey, and E. Aarts, *Ambient Intelligence.* Springer, 2005.

[3] B. H. Calhoun, J. Lach, J. Stankovic, D. D. Wentzloff, K. Whitehouse, A. T. Barth, J. K. Brown, Q. Li, S. Oh, N. E. Roberts, and Y. Zhang, "Body sensor networks: A holistic approach from Silicon to users," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 91–106, January 2012.

[4] C.-L. Yang, C.-L. Tsai, K.-T. Cheng, and S.-H. Chen, "Low-invasive implantable devices of low-power consumption using high-efficiency antennas for cloud health care," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 2, no. 1, pp. 14–23, March 2012.

[5] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in *Proceedings of the 4th International Symposium on Information Processing in Sensor networks*, Picastaway, NJ, 2005, pp. 364–369.

[6] INRIA, *Protocol for Low Power Wireless Sensor Network.* INRIA Tech. Report, 2010.

[7] J. Beutel, O. Kasten, and M. Ringwald, "BTnodes - a distributed platform for sensor nodes," in *Proceedings of the 1st ACM International Conference on Embedded Networked Sensor Systems*, New York, 2003, pp. 292–293.

[8] M. Hempstead, D. Brooks, and G.-Y. Wei, "Reliability analysis and optimization of power-gated ICs," *IEEE Transactions on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 2, pp. 193–202, June 2011.

[9] M. Seok, S. Hanson, Y.-S. Lin, Z. Foo, D. Kim, Y. Lee, N. Liu, D. Sylvester, and D. Blaauw, "The Phoenix processor: A 30pW platform for sensor applications," in *Proceedings of the IEEE Symposium on VLSI Circuits*, 2008, pp. 188–189.

[10] M. Sheets, F. Burghardt, T. Karalar, J. Ammer, Y. Chee, and J. Rabaey, "A power-managed protocol processor for wireless sensor networks," in *Proceedings of the IEEE Symposium on VLSI Circuits*, 2006, pp. 212–213.

[11] V. George and J. M. Rabaey, *Low-Energy FPGAs - Architecture and Design.* Kluwer Academic Publishers, 2001.

[12] V. Betz, J. Rose, and A. Maequardt, *Architecture and CAD for Deep-Submicron FPGAs.* Springer, 1999.

[13] L. Nazhandali, M. Minuth, and T. Austin, "SenseBench: Toward an accurate evaluation of sensor network processors," in *Proceedings of the IEEE International Workload Characterization Symposium*, 2005, pp. 197–203.

[14] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits," *Proceedings of the IEEE*, vol. 91, no. 2, pp. 305–327, June 2003.

[15] R. P. Brent and H. T. Kung, "A regular layout for parallel adders," *IEEE Transactions on Computers*, vol. 31, no. 3, pp. 144–147, March 1982.

[16] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Transactions on Computers*, vol. C-22, no. 8, pp. 786–793, August 1973.

[17] E. F. Moore, "Gedanken-experiments on sequential machines," *Automata Studies*, pp. 129–153, 1956.

[18] G. H. Hardy, "Method for synthesizing sequential circuits," *Bell System Technical Journal*, vol. 34, pp. 1045–1079, September 1955.

[19] A. Niedermeier, K. Svarstad, F. Bouwens, J. Hulzink, and J. Huisken, "The challenges of implementing fine-grained power gating," in *Proceedings of the 20th Great Lakes Symposium on VLSI.* New York, USA: ACM Press, May 2010, p. 361.

[20] V. D. Tovinakere, O. Sentieys, and S. Derrien, "A semiempirical model for wakeup time estimation in power-gated logic clusters," in *49th ACM/IEEE Design Automation Conference*, 2-7 June 2012, pp. 48–55.

[21] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, pp. 393–422, 2002.

[22] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics Magazine*, vol. 38, no. 8, 19th April 1965.

[23] ITRS, "International Technology Roadmap for Semiconductors: 2011 Overall Roadmap Technology Characteristics Tables," http://www.itrs.net, 2011.

[24] J. Rabaey, *Low Power Design Essentials*. Springer, 2009.

[25] V. Janakiraman, A. Bharadwaj, and V. Vishvanathan, "Voltage and temperature aware statistical leakage analysis framework using artificial neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 7, pp. 1056–1068, July 2010.

[26] Y. Taur and T. Ning, *Fundamentals of Modern VLSI Devices*. Cambridge University Press, 2009.

[27] Q. Wu, M. Pedram, and X. Wu, "Clock-gating and its application to low power design of sequential circuits," *IEEE Transactions on Circuits and Systems - I: Fundamental Theory and Applications*, vol. 47, no. 3, pp. 415–420, March 2000.

[28] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. Wiley-Interscience, 1999.

[29] L. Bisdounis and O. Koufopavlou, "Short circuit energy dissipation modeling for submicrometer CMOS gates," *IEEE Transactions on Circuits and Systems - I: Fundamental Theory and Applications*, vol. 47, no. 9, pp. 1350–1361, September 2000.

[30] D. Markovic, C. C. Wang, L. P. Allarcon, T.-T. Liu, and J. M. Rabaey, "Ultralow power design in near-threshold region," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 237–252, February 2010.

[31] B. H. Calhoun, J. F. Ryan, S. Khanna, M. Putic, and J. Lach, "Flexible circuits and architectures for ultralow power," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 267–282, February 2010.

[32] J. Kwong, Y. K. Ramadass, N. Verma, and A. P. Chandrakasan, "A 65nm sub-Vt microcontroller with integrated SRAM and switched capacitor DC-DC converter," *IEEE Journal of Solid State Circuits*, vol. 44, no. 1, pp. 115–126, January 2009.

[33] A. Wang, B. H. Calhoun, and A. P. Chandrakasan, *Sub-threshold Design for Ultra Low-Power Systems*. Springer, 2006.

[34] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi, *Low Power Methodology Manual for System-on-Chip Design*. Springer, 2008.

[35] S. Shigematsu, S. Mutoh, Y. Matsuya, Y. Tanabe, and J. Yamada, "A 1-V high-speed MTCOMS circuit scheme for power-down application circuits," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 6, pp. 861–869, June 1997.

[36] A. Lingamneni, C. Enz, J.-L. Nagel, K. Palem, and C. Piguet, "Energy parsimonious circuit design through probabilistic pruning," in *Proceedings of the 14th Design, Automation and Test in Europe*, March 2011, pp. 764–769.

[37] H. Nguyen, D. Menard, and O.Sentieys, "Design of optimized fixed-point WCDMA receiver," in *Proceedings of the XVII European Signal and Image Processing Conference*, August 2009, pp. 993–997.

[38] Synopsys Inc., *Design Compiler User Guide*, 2010.

[39] IEEE, *1801-2009 IEEE Standard for Design and Verification of Low Power Integrated Circuits*. IEEE, 2009.

[40] Cadence Design Systems Inc., *Encounter Foundation Flows: CPF-Based Low Power Implementation Flow Guide*, 2009.

[41] Synopsys Inc., *PrimeTime SI User Guide*, 2006.

[42] ——, *PrimeTime PX: Methodology for Power Analysis*, 2006.

[43] Cadence Design Systems Inc., *Encounter User Guide*, 2009.

[44] A. Sadat, H. Qu, C. Yu, J. S. Yuan, and H. Xie, "Low-power CMOS wireless MEMS motion sensor for physiological activity monitoring," *IEEE Transactions on Circuits and Systems - I: Regular Papers*, vol. 52, no. 12, pp. 2539–2551, December 2005.

[45] N. Potlapally, S. Ravi, A. Raghunathan, R. Lee, and N. Jha, "Configuration and extension of embedded processors to optimize IPSec protocol execution," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 5, pp. 605 –609, May 2007.

[46] M. A. Pasha, S. Derrien, and O. Sentieys, "A complete design flow for generation of ultra-low power WSN node architectures based on micro-tasking," in *Proceedings of the 47th ACM/IEEE Design Automation Conference*, Anaheim, USA, June 2010.

[47] J. Rabaey, "Silicon platforms for the next generation wireless systems. What role does reconfigurable hardware play?" in *Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing*, ser. Lecture Notes in Computer Science, R. Hartenstein and H. Grünbacher, Eds. Springer Berlin / Heidelberg, 2000, vol. 1896, pp. 277–285.

[48] Zigbee Alliance, "Zigbee standards overview," http://www.zigbee.org, 2012.

[49] M. M. Alam, O. Berder, D. Menard, and O. Sentieys, "TAD-MAC: Traffic-aware dynamic MAC protocol for wireless body sensor networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 2, no. 1, pp. 109–119, March 2012.

[50] Texas Instruments Inc., *MSP430 User Guide Technical Report*, 2010a.

[51] C. Piguet, J.-M. Masgonty, C. Arm, S. Durand, T. Schneider, F. Rampogna, C. Scarnera, C. Iseli, J.-P. Bardyn, R. Pache, and E. Dijkstra, "Low-power design of 8-b embedded CoolRISC microcontroller cores," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 7, pp. 1067–1078, July 1997.

[52] Tensilica Inc., "Xtensa customizable processors," http://www.tensilica.com, 2012.

[53] K. Kelley, M. Wachs, A. Danowitz, P. Stevenson, S. Richardon, and M. Horowitz, "Intermediate representations for controllers in chip generators," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2011, pp. 1–6.

[54] Opencores, "The openMSP430 User Guide," http://www.opencores.org, 2009.

[55] J. Lallet, *MOZAIC: Plate-forme générique de modélisation et de conception d'architectures reconfigurables dynamiquement.* PhD Thesis, 2008.

[56] J. Rose, J. Luu, C. W. Yu, O. Densmore, J. Goeders, A. Somerville, K. B. Kent, P. Jamieson, and J. Anderson, "The VTR project: Architecture and CAD for FPGAs from Verilog to Routing," in *Proceedings of the 20th ACM/SIGDA International Symposium on Field-Programmable Gate Arrays.* ACM, 2012, pp. 77–86.

[57] F. de Dinechin, "The price of routing in FPGAs," INRIA, Tech. Rep. RR-3772, September 1999.

[58] H. Hassan and M. Anis, *Low-Power Design of Nanometer FPGAs.* Elsevier Inc., 2010.

[59] J. Anderson and F. Najm, "Power estimation techniques in FPGAs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 10, pp. 1015–1027, October 2004.

[60] V. D. Tovinakere, O. Sentieys, and S. Derrien, "A polynomial based approach for wakeup time and energy estimation in power-gated logic clusters," *Journal of Low Power Electronics*, vol. 7, no. 4, pp. 482–489, December 2011.

[61] K. Poon, A. Yan, and S. Wilton, "A flexible power model for FPGAs," in *Proceedings of the ACM International Symposium on Field Programmable Gate Arrays*, 2002, pp. 312–321.

[62] Y.-L. Wu and D. Chang, "On the NP-completeness of regular 2-D FPGA routing architectures and a novel solution," in *IEEE/ACM International Conference on Computer-Aided Design*, November 1994, pp. 362–366.

[63] F. Li, Y. Lin, and J. Cong, "FPGA power reduction using configurable dual-Vdd," in *Proceedings of the IEEE/ACM Design Automation Conference*, 2004, pp. 735–740.

[64] ——, "Low-power FPGA using predefined dual-Vdd/dual-Vt fabrics," in *Proceedings of the ACM International Symposium on Field Programmable Gate Arrays*, 2004, pp. 42–50.

[65] S. Ishihara, M. Hariyama, and M. Kameyama, "A low-power FPGA based on autonomous fine-grain power gating," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 8, pp. 1394–1406, August 2011.

[66] V. Sklyarov, "Reconfigurable models of finite state machines and their implementation in FPGAs," *Journal of Systems Architecture*, vol. 47, 2002.

[67] ——, "Hierarchical finite state machines and their use for digital control," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, no. 2, pp. 222–228, February 1999.

[68] T. Sakurai and R. Newton, "Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 2, pp. 584–594, April 1990.

[69] Arizona State University, "Predictive Technology Models," http://ptm.asu.edu, 2012.

[70] B. H. Calhoun, F. A. Honore, and A. P. Chandrakasan, "A leakage reduction methodology for distributed MTCMOS," *IEEE Journal of Solid State Circuits*, vol. 39, no. 5, pp. 818–826, May 2004.

[71] M. Anis, S. Areibi, and M. Elmasry, "Dynamic and leakage power reduction in MTCMOS circuits using an automated gate clustering technique," in *Proceedings of the 39th ACM/IEEE Design Automation Conference*, New Orleans, June 2002, pp. 480–485.

[72] C. Long and L. He, "Distributed sleep transistor network for power reduction," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 9, pp. 937–946, September 2004.

[73] K. Shi and D. Howard, "Challenges in sleep transistor design and implementation in low-power designs," in *Proceedings of the 43rd ACM/IEEE Design Automation Conference*, San Francisco, July 2006, pp. 113–116.

[74] H. Kriplani, F. N. Najm, and I. N. Hajj, "Pattern independent maximum current estimation in power and ground buses of CMOS VLSI circuits: Algorithms, signal correlations and their resolution," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 8, pp. 998–1012, August 1995.

[75] S. Kim, C. J. Choi, D. K. Jeong, S. V. Kosonocky, and S. B. Park, "Reducing ground-bounce noise and stabilizing data-retention voltage of power gating structures," *IEEE Transactions on Electron Devices*, vol. 55, no. 1, pp. 197–205, January 2008.

[76] H.-O. Kim, Y. Shin, H. Kim, and I. Eo, "Physical design methodology of power gating circuits for standard cell based design," in *Proceedings of the 43rd ACM/IEEE Design Automation Conference*, San Francisco, July 2006, pp. 109–112.

[77] E. Choi, C. Chin, and Y. Chin, "HLS-pg: High level synthesis of power gated circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 3, pp. 451–456, March 2009.

[78] S. Roy, N. Ranganathan, and S. Katkoori, "A framework for power-gating functional units in embedded microprocessors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 11, pp. 1640–1649, November 2009.

[79] A. Sathanur, L. Benini, A. Macii, E. Macii, and M. Poncino, "Row-based power-gating: A novel sleep transistor insertion methodology for leakage power optimization in nanometer CMOS circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 3, pp. 469–482, March 2011.

[80] Y. Lee, D.-K. Jeong, and Taewhan, "Comprehensive analysis and control of design parameters for power gated circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 3, pp. 494–498, March 2011.

[81] X. Lu, "Layout and parasitic information for ISCAS circuits," http://dropzone.tamu.edu/~xiang/iscas.html, 2004.

[82] A. Abdollahi, F. Fallah, and M. Pedram, "An effective power mode transition technique in MTCMOS circuits," in *Proceedings of the 42nd ACM/IEEE Design Automation Conference*, Anaheim, June 2005, pp. 27–32.

[83] H. Singh, K. Agarwal, D. Sylvester, and K. J. Nowka, "Enhanced leakage reduction techniques using intermediate strength power gating," *IEEE Transactions on Very*

Large Scale Integration (VLSI) Systems, vol. 15, no. 11, pp. 1215–1224, November 2007.

[84] H. Xu, R. Vemuri, and W.-B. Jone, "Dynamic characteristics of power gating during mode transition," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 2, pp. 237–249, February 2011.

[85] S. Roy, N. Ranganathan, and S. Katkoori, "State-retentive power gating of register files in multicore processors featuring multithreaded in-order cores," *IEEE Transactions on Computers*, vol. 60, no. 11, pp. 1547–1560, November 2011.

[86] T. D. Vivek, O. Sentieys, and S. Derrien, "Wakeup time and wakeup energy estimation in power-gated logic clusters," in *Proceedings of the 24th International Conference on VLSI Design*, Chennai, India, January 2011, pp. 340–345.

[87] D.-W. Lin, M.-L. Cheng, S.-W. Wang, C.-C. Wu, and M.-J. Chen, "A constant-mobility method to enable mosfet series-resistance extraction," *IEEE Electron Device Letters*, vol. 28, no. 12, pp. 1132–1134, December 2007.

[88] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural techniques for power gating of execution units," in *Proceedings of International Symposium on Low Power Electronics Design*, Newport Beach, USA, August 2004, pp. 32–37.

[89] J. Seomun, I. Shin, and Y. Shin, "Synthesis of active-mode power-gating circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 3, pp. 391–403, March 2012.

[90] B. Ramkumar and H. Kittur, "Low-power and area-efficient carry select adder," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 2, pp. 371–375, February 2012.

[91] A. Lingamneni, K. K. Muntimadugu, C. Enz, R. M. Karp, K. Palem, and C. Piguet, "Algorithmic methodologies for ultra-efficient inexact architectures for sustaining technology scaling," in *Proceedings of the 9th ACM International Conference on Computing Frontiers*, 15-17 May 2012, pp. 3–12.

[92] Y. J. Chong and S. Parameswaran, "Configurable multimode embedded floating-point units for FPGAs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 11, pp. 2033–2044, November 2011.

[93] C. Tsen, S. Gonzalez-Navarro, and M. Schulte, "Hardware design of a binary integer decimal-based floating-point adder," in *25th International Conference on Computer Design*, October 2007, pp. 288–295.

[94] S. Jain, V. Erraguntla, S. Vangal, Y. Hoskote, N. Borkar, T. Mandepudi, and V. Karthik, "A 90mW/GFlop 3.4GHz reconfigurable fused/continuous multiply-accumulator for floating-point and integer operands in 65nm," in *23rd International Conference on VLSI Design*, January 2010, pp. 252–257.

[95] B. R. Zeydel, D. Baran, and B. G. Oklobdzija, "Energy-efficient design methodologies: High-performance VLSI adders," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 6, pp. 1220–1233, June 2010.

[96] L.-K. Wang and M. Schulte, "Decimal floating-point adder and multifunction unit with injection-based rounding," in *18th IEEE Symposium on Computer Arithmetic*, June 2007, pp. 56–68.

[97] M. C. Molina, J. M. Mendías, and R. Hermida, "High-level synthesis of multiple-precision circuits independent of data-objects length," in *Proceedings of the 39th Design Automation Conference*, 2002, pp. 612–615.

[98] S. Eratne, P. Nair, and E. John, "Leakage control in full adders with selectively stacked inverters," in *53rd IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, August 2010, pp. 833–836.

[99] P. Nair, S. Eratne, and E. B. John, "Probability-based optimal sizing of power-gating transistors in full adders for reduced leakage and high performance," *Journal of Low Power Electronics*, vol. 8, no. 4, pp. 464–471, August 2012.

[100] M. Sjalander, M. Drazdziulis, P. Larsson-Edefors, and H. Eriksson, "A low-leakage twin-precision multiplier using reconfigurable power gating," in *IEEE International Symposium on Circuits and Systems*, May 2005, pp. 1654 – 1657.

[101] T. Hoang and P. Larsson-Edefors, "Data-width-driven power gating of integer arithmetic circuits," in *IEEE Computer Society Annual Symposium on VLSI*, 19-21 August 2012, pp. 237–242.

[102] J. Fadavi-Ardekani, "M × N booth encoded multiplier generator using optimized Wallace trees," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 1, no. 2, pp. 120–125, June 1993.

[103] J. Cavanaugh, *Digital Computer Arithmetic: Design and Implementation*. McGraw-Hill, 1984.

[104] C. E. Shannon, "A symbolic analysis of relay and switching circuits," *Transactions of AIEE*, vol. 57, pp. 713–723, 1938.

[105] P. Choudhury and S. N. Pradhan, "An approach for low power design of power gated finite state machines considering partitioning and state encoding together," *Journal of Low Power Electronics*, vol. 8, no. 4, pp. 452–463, August 2012.

[106] S. N. Pradhan, M. T. Kumar, and S. Chattopadhyay, "Integrated power-gating and state assignment for low power FSM synthesis," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, 2008, pp. 269–274.

[107] L. Leinweber and S. Bhunia, "Fine-grained supply gating through hypergraph partitioning and Shannon decomposition for active power reduction," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2008, pp. 373–378.

[108] E. Ahmed and J. Rose, "The effect of LUT and cluster size on deep-submicron FPGA performance and density," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 3, pp. 288–298, March 2004.

[109] A. Gill, *Linear Sequential Circuits - Analysis, Synthesis, and Applications*. McGraw-Hill, Inc., 1966.

[110] S. S. Yau and K. C. Wang, "Linearity of sequential machines," *IEEE Transactions on Electronic Computers*, vol. 15, no. 3, pp. 337–354, June 1966.

[111] M. Cohn, "Properties of linear machines," *Journal of the ACM*, vol. 11, no. 3, pp. 296–301, July 1964.

[112] B. Elspas, "The theory of autonomous linear sequential networks," *IRE Transactions on Circuit Theory*, vol. 6, pp. 45–60, 1959.

[113] L. Alarcon, T. T. Liu, M. D. Pierson, and J. Rabaey, "Exploring very low-energy logic: A case study," *Journal of Low Power Electronics*, vol. 3, no. 3, pp. 223–233, December 2007.

Résumé en français : Un nœud d'un réseau de capteurs sans fil traite dans ses unités de calcul les signaux issus de plusieurs types de capteurs et effectue différentes tâches liées aux protocoles de communication. Devant exécuter plusieurs types de contrôle, sa flexibilité est un paramètre très important. Les solutions à base de microcontrôleurs ou de FPGA ont été proposées pour aborder le besoin de flexibilité, mais au prix d'une efficacité énergétique réduite. Dans cette thèse, des contrôleurs flexibles à ultra-faible énergie basés sur un contexte de micro-tâches reconfigurables sont explorés comme alternative. Des architectures modulaires pour des machines d'états finis (FSM) et des chemins de données (DP) reconfigurables sont proposées. Les techniques de coupure de l'alimentation (PG pour *power gating*) sont utilisées pour adapter la consommation aux besoins et réduire la puissance statique. Dans un premier temps, des modèles pour l'estimation des paramètres clés d'un circuit avec PG sont proposés au niveau porte. Ensuite, les opportunités des techniques PG sont déterminées sur les FSM et DP reconfigurables pour en réduire l'énergie. Dans les chemins de données, la reconfiguration fait varier la précision des opérateurs et le PG permet d'éteindre les blocs logiques inutilisés. Une gestion de l'alimentation au niveau *lookup table* (LUT) est proposée pour réduire les courants de fuite en mode actif et en veille dans les FSM reconfigurables. Des résultats montrent les très bonnes performances des architectures proposées par rapport aux processeurs et FPGA.

Mots clés : Circuits intégrés à faible consommation, commande numérique, microcontrôleurs, réseaux de capteurs, réseaux sans fil

Résumé en anglais : A wireless sensor network (WSN) node may need to process signals from various sensors and perform different transceiver tasks apart from being able to change its functions dynamically. A controller in the node is therefore required to execute different control tasks to manage its resources implying that flexibility is a key concern. Microcontrollers and FPGAs have been proposed to address the need for flexibility at the cost of reduced energy efficiency. In this thesis, ultra-low power flexible controllers for WSN nodes based on reconfigurable microtasks are explored. A reconfigurable microtask is a digital control unit with a reconfigurable finite state machine (FSM) and datapath. Scalable architectures for reconfigurable FSMs along with variable precision adders in datapath are proposed for flexible controllers in this work. Power gating is considered for FSMs and adders for low power operation. First, the design issues in power gating are studied extensively. Models for estimation of key design parameters of power-gated circuits are derived at gate level. Next, power gating opportunities are determined in reconfigurable adders and FSMs proposed for microtasks. In adders, reconfigurability is used for varying the precision of operation and saving energy by power-gating unused logic. Power gating at the level of lookup table logic is proposed to achieve active leakage power reduction in reconfigurable FSMs. The proposed models are then applied to analyze energy savings in logic clusters due to power gating. Power estimation results show good performance of proposed architectures on different metrics when compared with other solutions in the design space of controllers.

Keywords: Digital control, low power integrated circuits, microcontrollers, wireless sensor networks