Gestion hiérarchique de la reconfiguration pour les équipements de radio intelligente fortement hétérogènes

Xiguang Wu

N° d'ordre : 2016-02-TH

CentraleSupélec

Ecole Doctorale MATISSE

« Mathématiques, Télécommunications, Informatique, Signal, Systèmes Electroniques »

Laboratoire de Signal, Communication et Electronique Embarquée

THÈSE DE DOCTORAT

DOMAINE : STIC
Spécialité : Electronique

Soutenue le 21 mars 2016

par :

Xiguang WU

Hierarchical Reconfiguration Management for Heterogeneous Cognitive Green Radio Equipments

Composition du jury :

Président du jury :	M. Guy GOGNIAT	Professeur à Université de Bretagne-Sud
Rapporteurs :	Mme. Lirida NAVINER	Professeur à Télécom ParisTech
	M. Tanguy RISSET	Professeur à l'INSA de Lyon
Examinateurs :	M. Christophe MOY	Professeur à CentraleSupélec
	M. Dominique NOGUET	Ingénieur à CEA-LETI
	M. Xun ZHANG	Professeur assistant à ISEP
Directeur de thèse :	M. Jacques PALICOT	Professeur à CentraleSupélec
Co-directeur de thèse :	M. Pierre LERAY	Professeur à CentraleSupélec

“殊途同归。”

— <<周易•系辞下>>

“Tous les chemins mènent à Rome.”

“All roads lead to Rome.”

Acknowledgements

First and foremost, I express my most sincere gratitude to my supervisors, Professor Jacques Palicot and Professor Pierre Leray, for giving me the opportunity to do this work. Thanks to Professor Jacques Palicot for his great guidance, patience and support all along the past three years. Thanks to Professor Pierre Leray for his precious technical guidance and help throughout my Ph.D. Without their guidance and encouragement, this work would not have been successful.

I deeply appreciate the rest of the staff in SCEE team for their encouragement and thoughtful suggestions. I would like to thank all the members of SCEE team for their friendship and help during my time in Rennes. Thanks especially to Malek for the discussions and valuable suggestions on the OFDM scenario of HDCRAM.

I would like to thank Professor Lirida NAVINER and Professor Tanguy RISSET for agreeing to be the rapporteurs of this dissertation. Your valuable suggestions and critical comments are important and helpful for revising and improving the thesis. I am also grateful to Professor Guy GOGNIAT for accepting to serve as the president of the dissertation committee, and Professor Christophe MOY, Dr. Dominique NOGUET, Dr. Xun ZHANG, for accepting to be my committee members. Your feedback and discussion are valuable for guiding and improving the current work.

I would like to acknowledge all my friends for their warm support, care and precious friendship through these difficult years.

It is my pleasure to express my gratitude to all those people who have supported and helped me during this thesis.

Finally, I would like to dedicate this work to my family for standing behind me with their love, concern, constant support and limitless patience.

WU Xiguang

Rennes, France

Contents

Résumé

Introduction

Cette thèse s'intéresse à la mise en œuvre d'équipements sur plateformes hétérogènes.

Le contexte principal est celui de l'Ecoradio. A savoir l'étude et le développement de systèmes de radiocommunications économes en énergie, qui de ce simple fait auront une empreinte carbone beaucoup plus faible que les systèmes actuels. Plus précisément nous nous intéressons au domaine de l'Ecoradio Intelligente au niveau électronique d'un équipement. L'Ecoradio est rapidement présentée au chapitre 1-1. Nous (équipe SCEE) avons montré depuis plusieurs années que la Radio Intelligente (RI) peut être un outil très efficace pour réussir à atteindre une Ecoradio. La RI est résumée au chapitre 1-2. Dans ce contexte de RI, les équipements sont considérés comme intelligents car ils obéissent au cycle intelligent proposé pour la Radio Intelligente. Utiliser la RI sous contrainte de consommation d'énergie pour atteindre une Ecoradio, est proposé au chapitre 1-3 et cela aboutit au concept d'Ecoradio Intelligente Les équipements RI étudiés étant complexes et adaptatifs (par principe de la RI) il est nécessaire de les gérer de manière automatique et autonome : c'est précisément le but du gestionnaire développé par l'équipe SCEE depuis une dizaine d'années. Ce gestionnaire, nommé HDCRAM pour Hierarchical and Distributed Cognitive Radio Architecture Management est utilisé pour gérer les équipements étudiés dans cette thèse. Celui-ci est présenté au chapitre 1-4.

Après avoir présenté dans ce premier chapitre le contexte et les outils de base qui serviront à la mise en œvre des équipements étudiés, le second chapitre propose d'implanter le gestionnaire HDCRAM sur des plates formes hétérogènes. En particulier, l'apport et l'intérêt de la Reconfiguration Partielle (RP) de FPGA sera étudié dans ce contexte. Dans le troisième chapitre, les métriques, plus particulièrement celles relatives à l'état de

la plate-forme d'un point de vue électronique, nécessaires à une prise de décision sous contrainte d'économe d'énergie sont étudiées. Elles sont identifiées, leur accessibilité est précisée et leur utilisation dans notre contexte est présentée. Dans le quatrième chapitre la mise en œuvre de l'ensemble des techniques étudiées lors de cette thèse est réalisée pour un système de type émission/réception. Les scénarios, les métriques utilisées lors de ces scénarios, les algorithmes de décision ainsi que le déploiement d'HDCRAM sont détaillés. L'implantation temps réel sur plate-forme du système permet de conclure sur les gains attendus et offre une possibilité de démonstration de l'ensemble. Celle-ci sera présentée lors d'un Workshop ETSI en mars et lors de la soutenance.

1 Contexte et Position du problème

1.1 L'Eco Radio

Il y a quelques dizaines d'années, le développement durable (DD) n'était la préoccupation que de quelques groupes écologistes. Maintenant, depuis l'assemblée générale des Nations Unies de décembre 1987 et la résolution 42/187 [1], ce problème est devenu une préoccupation de la société. La commission Bruntland a défini le DD comme étant un développement qui : "meets the needs of the present without compromising the ability of future generations to meet their own needs". Depuis plusieurs conférences, organisées sous l'égide des Nations Unies, ont confirmé l'importance du DD (de Rio de Janeiro-1992 à Copenhague-2009 et tout récemment la COP 21 à Paris en décembre 2015). L'un des problèmes le plus important que doit prendre en compte le DD est le changement climatique et l'émission de CO_2...

Même s'il est clair que les principaux contributeurs en émission de CO_2 sont la production d'électricité, le transport et l'industrie, les Technologies de l'Information et de la Communication (TIC) y contribuent pour une part non négligeable. En effet, actuellement, 3 % de l'énergie mondiale sont consommées par les TIC, ce qui est à l'origine de 2 % des émissions de CO_2 (ce qui est comparable à l'émission de CO_2 de l'aviation civile mondiale !), ces chiffres continuent de croitre régulièrement malgré les efforts mis en œuvre par les différents acteurs du domaine.

Réduire le niveau d'émission des ondes électromagnétiques est un autre aspect du DD pour les radios communications. Cette réduction offrira une meilleure coexistence entre tous les systèmes et réduira le niveau d'exposition des utilisateurs. Le premier papier relatif à l'écoradio (sous l'angle de la réduction du niveau des ondes électromagnétiques), grâce au concept de radio intelligente, a été présenté lors d'une assemblée générale de l'URSI [2].

Mais, à cette époque, ce type de préoccupation n'était pas à la mode.

L'écoradio (1) est souvent limité à l'aspect efficacité énergétique, mais nous l'envisageons, dans cette thèse, dans un sens plus large. Dans [3], les différentes implications du DD dans le domaine des radiocommunications ont été décrites. Ces implications vont de l'émission de CO_2 (à cause de la consommation électrique) au recyclage des équipements et des ondes transmises, en passant par la pollution électromagnétique (avec les conséquences de l'exposition aux ondes des utilisateurs). Parmi l'énorme activité sur le sujet quelques projets sont (ou ont été) particulièrement importants et productifs. Nous nous limiterons donc à la présentation de ceux-ci. Plus de projets sont présentes en section 1.1.2 dans le corps du document.

1.1.1 Au niveau international

1. GREENTOUCH

Il s'agit d'un projet très ambitieux piloté par Alcatel, dont l'objectif est de décroître d'un facteur 1000 la consommation énergétique du réseau [4]. Cette décroissance est analysée segment par segment avec des objectifs différents suivant les segments et cela malgré l'augmentation des débits. Parmi les nombreux résultats de ce projet, des architectures, des technologies, des composants, des algorithmes ont été proposés.

2. EARTH

Le projet EARTH pour Energy Aware Radio and NeTwork TecHnologies a été un projet financé lors du programme FP7 de la Commission Européenne [5]. Ce projet a été

(1). Le concept anglais de Green Radio pourrait être traduit radio verte. Mais le terme technique Green a récemment été étudié et la traduction éco a été adoptée au journal officiel, c'est pour cela que nous utilisons la formulation écoradio.

un des premiers à s'intéresser au problème de l'Ecoradio avec un objectif ambitieux de réduire de 50% la consommation des systèmes de télécom mobiles. Ce projet a été à l'origine de nombreuses idées, définitions et de nombreux algorithmes aujourd'hui reconnus et utilisés par de nombreux autres projets. Parmi d'autres citons les idées d'allumage/exctinction des Stations de Base en fonction du nombre d'usagers, d'allumage/exctinction de l'amplificateur de puissance en fonction des périodes sans transmission, d'algorithmes pour augmenter ces périodes, les protocoles coopératifs, le cell breathing, etc...

3. C2POWER

Ce projet est intéressant, car il est exactement dans la lignée de ce que nous appelons l'Ecoradio Intelligente (voir section suivante) [6]. Il se propose d'étudier comment l'intelligence et les stratégies de coopération permettent de réduire globalement la consommation énergétique. Les résultats de ce projet ont été considérés comme très positifs, ce qui nous conforte dans cette thèse dont le contexte est justement l'Ecoradio Intelligente au niveau électronique d'un équipement.

1.1.2 Au niveau Français

1. SOGREEN

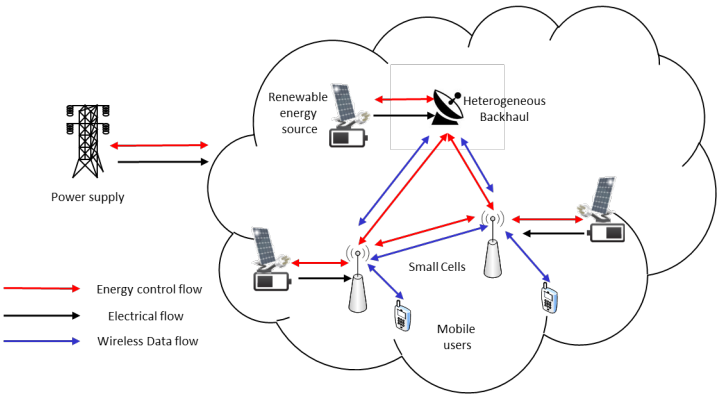


FIGURE 1 – SOGREEN.

Suivant une approche multidisciplinaire, SOGREEN propose un système de gestion intelligente de l'énergie basé sur une intégration étroite entre réseau cellulaire et smart grid, escomptant une amélioration considérable de l'efficacité éco-énergétique [7]. Comme cela

est indiqué sur la Figure 1 le réseau de télécommunications cellulaire et le Réseau Electrique Intelligent (REI) sont interconnectés, de manière à globalement optimiser l'énergie consommée. Dans ce schéma, nous pouvons distinguer trois flux différents : celui correspondant aux données du réseau de communications, celui correspondant au REI et enfin celui correspondant aux communications spécifiques du REI. Dans ce projet sont menées des études d'algorithme de prise de décision globale, au niveau de chaque sous réseau. L'application du gestionnaire HDCRAM (voir section suivante) est aussi proposée dans ce projet.

2. TEPN

TEPN est un projet du laboratoire d'excellence Breton CominLabs Le but de ce projet est d'adapter la consommation du réseau à la charge réelle de celui-ci [8]. Parmi les sujets étudiés, figurent la définition de métriques prenant en compte la globalité du problème, l'étude de solutions permettant de diminuer la consommation des amplificateurs de puissance au niveau des stations de base et l'étude d'algorithme de prise de décision sous différentes contraintes et métriques, notamment en se focalisant sur les algorithmes d'apprentissage qui apprennent le comportement du réseau afin de l'optimiser.

1.2 La Radio Intelligente

Après avoir, en 1995, proposé le nouveau concept de radio logicielle (RL) ou Soft-Ware Radio en anglais [9] , Joe Mitola lors de son travail de thèse s'est intéressé à l'utilisation du spectre. Il a constaté que celui-ci était très mal utilisé, en grande partie sous-utilisé. Il en a déduit qu'une gestion locale, intelligente du spectre permettrait d'augmenter considérablement son taux d'utilisation. Mitola a compris qu'il fallait mettre de l'intelligence à la fois dans le réseau et dans les équipements pour être au plus près des besoins et de la ressource donc au final pour augmenter l'efficacité spectrale : c'est la raison pour laquelle il a proposé la RI (Cognitive Radio en anglais) [10, 11]. Il a montré que celle-ci serait plus efficace si elle était associée à la technologie RL.

Suivant la description de la Figure 2, un système RI pourra adapter son comportement (fonctionnement) à son environnement grâce à :

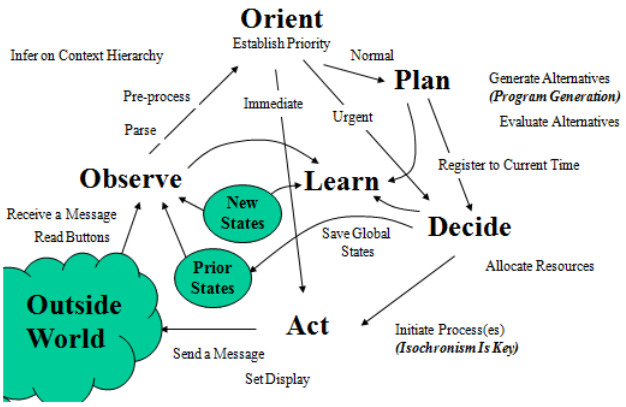


FIGURE 2 – Le cycle intelligent de J.Mitola. [10]

- ses capacités d'analyse à travers ses capteurs. La notion de capteurs est, dans notre vision, très large. Elle correspond à tout moyen de fournir de l'information au moteur intelligent qui prendra les décisions. Cette information proviendra de capteurs physiques réels, d'algorithmes de traitement du signal, d'échanges d'information avec les différents nœuds d'un réseau, etc.

- son intelligence qui lui permet de prendre les décisions adéquates (basées sur de l'apprentissage et/ou des bases de connaissance). La connaissance utilisée par la prise de décision est, comme l'information fournie par les capteurs, une notion très large, cela va des paramètres fournis par les capteurs aux considérations technico-économiques, en passant par les règles réglementaires d'utilisation du spectre. Dans le contexte de cette thèse, une contrainte particulière est associée à cette fonction de prise de décision. Il s'agit de la contrainte DD, sous les déclinaisons contraintes de consommation minimale, non pollution électromagnétique...

- ses capacités d'auto reconfiguration (offertes par la technologie support : la RL) pour modifier son fonctionnement.

Un schéma simplifié représentant ce fonctionnement est donné sur la Figure 3.

Le mot capteur doit être pris au sens large. Il s'agit de tout moyen donnant de l'information de toute nature pouvant être mise à profit dans le cycle intelligent pour optimiser le lien radio afin d'améliorer le service rendu.

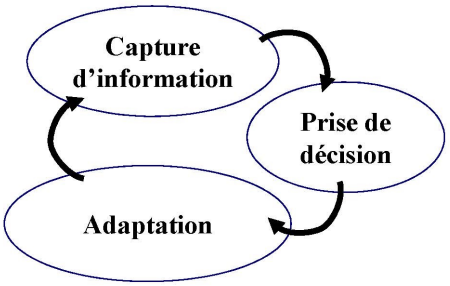


FIGURE 3 – Cycle intelligent simplifié en trois étapes.

Ces différents moyens vont des capteurs au sens classique du terme (microphone, etc.) aux capteurs appelés intelligents dans la littérature et fournissant une information qui résulte d'un traitement évolué (par exemple la réponse impulsionnelle d'un canal).

Classiquement on peut faire la liste de ces capteurs en fonction de l'environnement considéré comme dans le tableau 1 suivant.

TABLE 1 – Classification d'une liste (non exhaustive) de capteurs en fonction de l'environnement. [12]

Capteurs	Environnent
occupation spectrale, trous ou blancs dans le spectre rapport Signal à Bruit, réponse impulsionnelle du canal, etc...	Électromagnétique
nombre et positions des Hot Spot, et stations de base, des utilisateurs Standards utilisables à proximité, Opérateurs et services à proximité charge sur un lien radio, etc...	Réseau
niveau de batterie, consommation énergétique taux d'utilisation des circuits (FPGA), de la ressource de calcul taux d'occupation de la mémoire température du matériel	Matériel
micro, caméra, appareil photo, identification de l'usager Position spatiale, vitesse, heure, intérieur/extérieur préférences, profil de l'utilisateur détection, reconnaissance de visage, reconnaissance de voix, etc...	Utilisateur

Le chapitre 3 de cette thèse discutera clairement des métriques (capteurs) relatifs au matériel dans le tableau.

1.2.1 La gestion du spectre

Contrairement à une idée reçue, le spectre est une ressource publique, seule son utilisation est privée. Ce qui fut le cas lors de la vente des licences UMTS.

Le spectre est une ressource naturelle finie. En effet une fréquence n'existe que parce qu'elle peut être générée. De ce point de vue, il est nécessaire d'avoir une quantité d'énergie suffisante pour générer la fréquence et la diffuser. Nous pouvons donc parler de ressource finie puisqu'elle dépend elle-même de ressources énergétiques finies.

Cette ressource finie peut être utilisée indéfiniment (tant que la ressource énergétique est disponible pour générer l'onde électromagnétique).

Les règles d'attribution actuelles des fréquences obéissent à un processus très compliqué et long à mettre en œuvre. L'allocation des fréquences est aujourd'hui fixe et attribuée sur la base de services suivant des règles internationales rigides, elles- mêmes discutées tous les 5 ans (lors de la Conférence Mondiale Administrative) [(2)]. Une telle allocation aboutit à une situation dans laquelle il apparait clairement que l'ensemble du spectre est alloué. Une première conclusion hâtive serait de dire qu'il n'y a plus de place disponible dans ce spectre.

Or les études ont montré que le spectre pouvait être alloué mais non utilisé (cas des bandes réservées aux militaires). Une analyse de l'occupation spectrale telle que celle présentée dans la figure 4 pour la bande à 2.4 GHz et la figure 5 pour la bande TV montre qu'à un instant précis (le 1er septembre 2004) et dans un lieu donné (à New-York) le spectre est sous-utilisé (une utilisation de l'ordre de quelques %). Ce constat a donné naissance à la notion de "Hic et Nunc", qui veut dire, qu'indépendamment de l'attribution des fréquences, le spectre peut être disponible en un lieu et à un instant donné. Par conséquent dans ce lieu et à l'instant considéré, si l'équipement est capable de connaitre le spectre utilisé, il pourra établir une communication dans les bandes spectrales sous-utilisées. C'est ce qui est aussi appelé, dans la littérature, une communication opportuniste.

Les techniques mises en œuvre pour identifier l'occupation spectrale sont grossièrement de deux types comme décrit ci-après :

(2). WARC process (World Administrative Radio Conference)

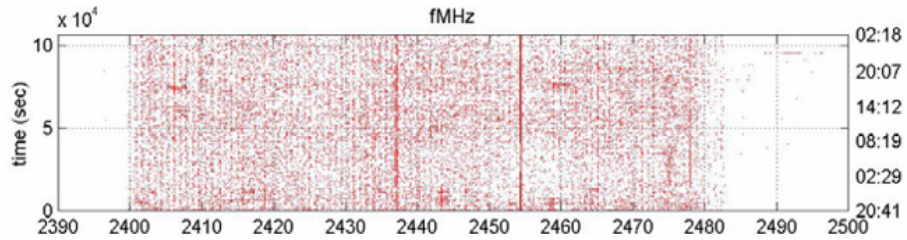


FIGURE 4 – Les mesures d'occupation de la bande 2.4 GHz. [13]

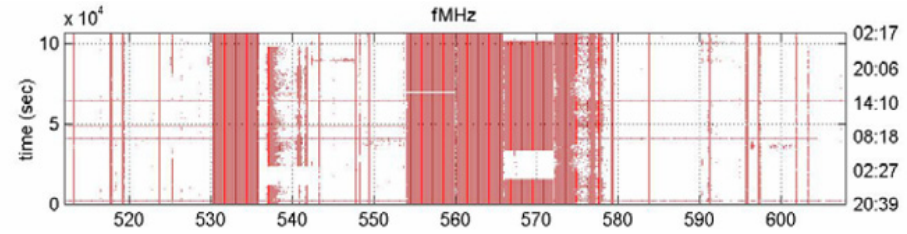
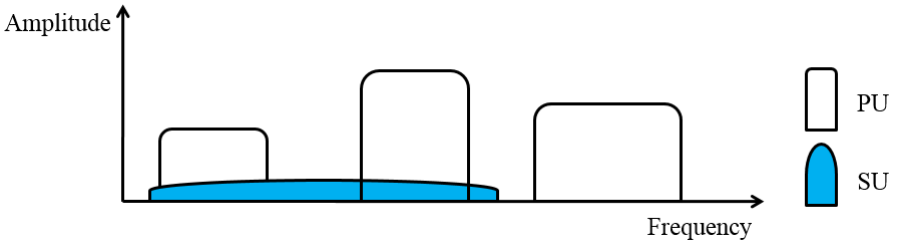
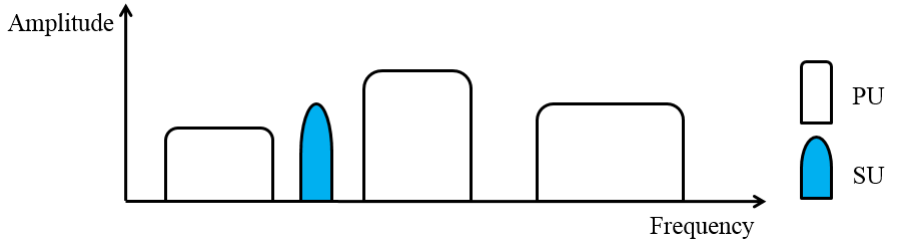


FIGURE 5 – Les mesures d'occupation de la bande TV. [13]



(a) underlay



(b) overlay

FIGURE 6 – Dynamic spectrum access modes.

La technique underlay

Comme son nom le laisse supposer cette technique consiste à insérer un nouveau signal

dans le même spectre et en même temps que les signaux d'origine. La contrainte évidente est que le signal additionnel ne perturbe en rien la qualité des signaux d'origine. C'est une contrainte très forte et très peu de systèmes la remplissent.

Dans ce contexte Haykin [14] a défini la notion de température d'interférence.

La technique overlay

Lorsque l'on parle d'accès opportuniste au spectre, de détection de blancs du spectre, de trous dans le spectre et d'insertion du signal d'un utilisateur secondaire, c'est généralement par une technique "overlay". Cette technique nécessite 5 étapes successives.

- un filtrage
- une détection de présence ou d'absence d'un utilisateur primaire dans la bande considérée.
- une qualification de la qualité de cette bande
- une prise de décision quant à l'utilisation par l'utilisateur secondaire, grâce aux différentes informations : présence, qualité,...
- une insertion du signal dans le spectre (cette insertion doit se faire de manière très précautionneuse, de façon à ne pas perturber les bandes adjacentes,...)Des modulations avec des DSP présentant des affaiblissements importants dans les bandes adjacentes seront préférées (cas par exemple de l'OFDM/OQAM)

Chacune des ces étapes a des contraintes très spécifiques et nécessite des algorithmes de traitement du signal avancés.

1.2.2 Une vision plus globale

L'équipe SCEE (pour rappel, qui accueille cette thèse), a proposé un modèle en trois couches pour expliciter sa vision de la RI. (voir figure 7)

- une couche de haut niveau, qui regroupe essentiellement la couche application, ainsi que les interfaces de type homme-machine, appelée *couche supérieure* ;
- une *couche intermédiaire* dans laquelle on retrouve les couches Transport et Réseau,
- une couche de bas niveau dans laquelle on retrouve les couches MAC et physique, appelée *couche inférieure*.

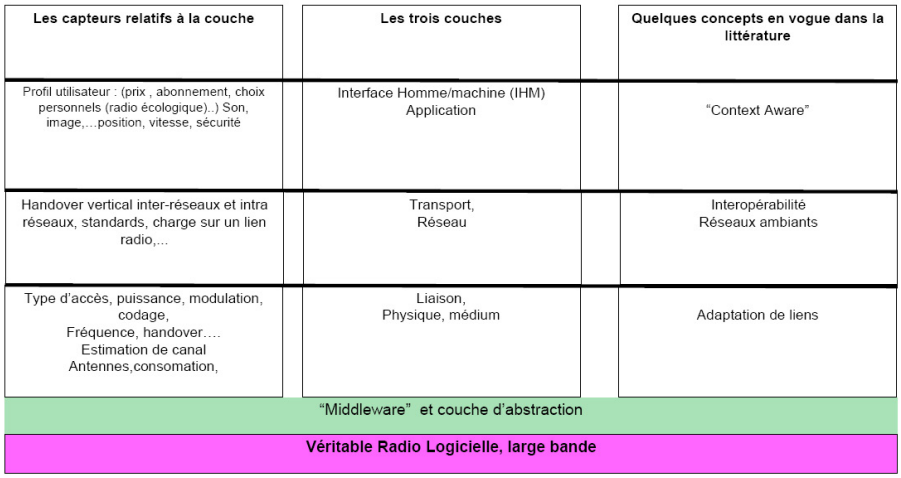


FIGURE 7 – Une vision multicouches de la RI. [12]

L'ensemble de ces couches fonctionne sur une plate-forme RL (si possible idéale), mais ce modèle fonctionne aussi avec une plate-forme radio logicielle restreinte. Ces plates-formes radio logicielle reposent sur une architecture matérielle d'exécution, qui en toute généralité est hétérogène. Cette plate-forme est idéalement abstraite à travers une couche d'abstraction, qui offre une transparence en termes d'implantation de composants logiciels de traitement du signal que l'on y exécute. Dans le modèle de la figure 7, nous avons dans la colonne de gauche fait figurer certains capteurs. Dans la colonne de droite sont cités les domaines de recherche relatifs à la couche en question avec lesquels la RI entretient des liens très étroits. Bien entendu comme notre objectif est d'optimiser le fonctionnement de ces trois couches de manière intelligente, la RI aura aussi un lien très fort avec le domaine de l'optimisation intercouches. Ce que l'on trouve dans la littérature sous la dénomination ≪ radio opportuniste ≫ est, suivant le modèle précédemment présenté, la restriction à la sous-partie de la couche physique de la RI concernée par la gestion du spectre.

1.3 l'Ecoradio Intelligente

L'EcoRadio Intelligente (ERI) est une radio intelligente (RI) qui prend en compte le développement durable (en particulier l'efficacité énergétique) comme une contrainte additionnelle dans le processus de décision du cycle intelligent. L'ERI consiste à : ≪ décroitre le niveau des ondes électromagnétiques en envoyant le signal adéquat dans la direction

désirée, avec la puissance suffisante, quand cela est nécessaire, tout en conservant la même qualité de service ≫. Il s'agit du concept d'≪ ondes utiles ≫. Pour cela, la RI grâce à ses capteurs, qui permettent d'avoir une vision locale de l'environnement, permettra de répondre efficacement à ce concept d'≪ ondes utiles ≫.

D'un point de vue théorique, le gain en efficacité spectrale, quelle que soit la manière d'obtenir ce gain, pourrait être utilisé pour diminuer la puissance des ondes transmises. Cependant, d'un point de vue pratique, l'ensemble des acteurs des télécommunications préfère utiliser ce gain pour accroitre le débit transmis (donc le nombre d'utilisateurs) à puissance constante plutôt que de diminuer la puissance à débit constant. Par conséquent, notre approche pourrait sembler en contradiction avec les considérations économiques des acteurs des télécommunications. Or, il n'en est rien, car d'une part diminuer la facture énergétique est devenu une préoccupation majeure de ces différents acteurs et d'autre part l'ERI consiste à : ≪ décroitre le niveau des ondes électromagnétiques en envoyant le signal adéquat dans la direction désirée, avec la puissance suffisante, quand cela est nécessaire, tout en conservant la même qualité de service ≫. Il s'agit du concept d'≪ondes utiles≫

Pour cela, la RI grâce à ses capteurs, qui permettent d'avoir une vision locale de l'environnement, permettra de répondre efficacement à ce concept d'≪ ondes utiles ≫. Cela devrait éviter la pollution de certaines bandes, comme la bande de radioastronomie. En effet, l'écoute passive dans cette bande est très perturbée par le niveau de plus en plus élevé des ondes des signaux de radiocommunications.

Comme déjà expliqué précédemment, nous aimerions obtenir cet éco-comportement dans un sens le plus large possible (diminution de la consommation d'énergie pour réduire l'empreinte carbone, équilibre entre l'efficacité énergétique et l'efficacité spectrale, contrôle de la pollution électromagnétique, impact sur les personnes, cycle de vie des équipements, etc.). Nous avons déjà identifié qu'une intelligence répartie dans le réseau est une condition nécessaire. Par conséquent nous proposons d'utiliser la Radio Intelligente comme une technologie potentielle pour atteindre l'objectif. Cette solution pourrait être implémentée soit côté terminal mobile ou côté station de base, partout dans le réseau radio hétérogène.

1.4 HDCRAM

Cette section présente une architecture de gestion, initialement proposée pour gérer un équipement de RI. Son acronyme est HDCRAM ce qui signifie en anglais Hierarchical and Distributed Cognitive Radio Architecture Management. HDCRAM peut être ajouté à tout système existant afin de transformer ce dernier en un système intelligent capable de prendre et de gérer des décisions autonomes. Par exemple HDCRAM a été récemment appliqué au Réseau Electrique Intelligent (REI) ou smart grid.

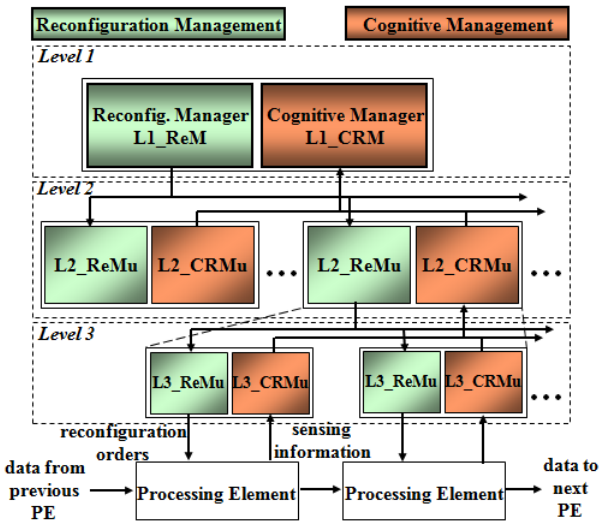


FIGURE 8 – A schematic example of HDCRAM architecture.

HDCRAM est présenté sur la Figure 8. Le cycle intelligent de la Figure 3 montre qu'un équipement a trois activités principales, capture de l'information, prise de décision et reconfiguration du système. Dans HDCRAM, la reconfiguration et la gestion de l'intelligence (capture des métriques et prise de décision) suivent deux chemins séparés. HDCRAM est composé de 3 niveaux hiérarchiques ainsi qu'un niveau opérateur qui exécute l'ensemble de la chaîne de transmission.

Cette architecture comprend deux sous-gestionnaires :

Le gestionnaire de l'intelligence noté Cognitive Radio Management Units (CRMu) : Un CRMu échange de l'information seulement d'un niveau inférieur à un niveau supérieur. Cette entité possède l'intelligence et peut prendre des décisions. Dans ce cas, elle envoie ses ordres liés à la décision au gestionnaire de reconfiguration de même niveau.

Le gestionnaire de re-configuration noté Reconfiguration Management Units (ReMu) : Un ReMu échange de l'information seulement d'un niveau supérieur à un niveau inférieur. Comme indiqué précédemment, il existe aussi un échange d'information possible entre un CRU et un CRMu de même niveau.

Les 3 niveaux se comportent de la façon suivante :

- Le niveau 1 est composé d'un seul couple CRM/L1 ReM et est le gestionnaire général du système. C'est à ce niveau que se prennent les décisions globales qui ont un impact sur l'ensemble du système.
- Le niveau 2 est composé d'un certain nombre de couples (L2 CRMu/L2 ReMu). Ils fournissent au niveau L1 l'information utile pour qu'il puisse prendre une décision. Il transmet l'information du niveau L3 sous une forme compressée, il s'agit d'abstraire l'information. S'il possède toute l'information nécessaire, une décision peut être prise à ce niveau.
- Le niveau 3 est composé d'un certain nombre de couples (L3 CRMu/L3 ReMu). Chacun de ces couples est associé à un opérateur. L3 ReMu est l'entité qui est en charge de la reconfiguration de son opérateur et L3 CRMu est en charge de traiter l'information provenant de l'opérateur (une métrique par exemple) et il peut si l'information dont il dispose est suffisante prendre une décision locale.
- Un opérateur est un composant (une fonction du système) qui est soit reconfigurable soit une mesure de métrique (par exemple un filtre ou un SNR).

L'intelligence est distribuée dans les 3 niveaux hiérarchiques et à différents emplacements à chaque niveau. De ce fait, il est possible de prendre des décisions à plusieurs niveaux et ainsi de générer des cycles de décision plus ou moins rapides.

- Une décision locale , simple et rapide au niveau 3 (voir le petit cercle de la figure 9).
- Si la décision est plus complexe à prendre et met en jeu plusieurs opérateurs gérés par le même niveau 2, alors elle est prise au niveau 2 (cycle intermédiaire sur la figure 9).
- Enfin, si la décision implique de nombreux opérateurs qui ne sont pas tous gérés par le même L2 alors la décision sera prise au niveau L1 (grand cycle sur la figure 9).

En résumé HDCRAM possède les caractéristiques suivantes :

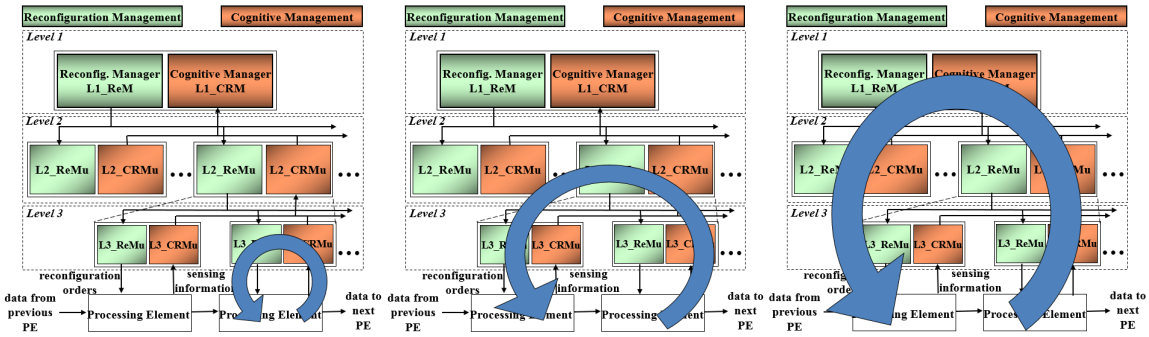


FIGURE 9 – Scale of the cognitive cycle : small (left), medium (middle), and large (right).

- Deux chemins totalement indépendants, un pour la remontée et la gestion de l'intelligence et un second pour redescendre les ordres de reconfiguration.
- HDCRAM est un modèle d'architecture indépendant des traitements réalisés dans les boîtes interconnectées.
- HDCRAM est un squelette d'une architecture de gestion.
- Pour un scénario donné le modèle HDCRAM est implémenté de manière spécifique.
- Il y a 3 niveaux de décision possible, ce qui correspond à 3 cycles intelligents de taille différente dans l'architecture.
- HDCRAM peut être appliqué à n'importe quel système complexe.
- Les règles et les connections entre les boites permettent de déployer des scénarios intelligents et de spécifier tous les éléments nécessaires, ainsi que leurs connections pour implémenter ce scénario.
- Une implémentation spécifique de HDCRAM peut être émulée, simulée, de façon à pouvoir prédire le fonctionnement d'un système pour un scénario donné.
- Une implémentation spécifique de HDCRAM peut intégrer n'importe quel algorithme de prise de décision.

Pour réaliser des équipements de RI, il est nécessaire d'utiliser une plate-forme reconfigurable, qui soit capable de s'adapter à n'importe quel type de traitement et aux différentes contraintes de ces traitements. Ces contraintes peuvent être la flexibilité, la puissance de calcul... . Pour répondre à ces contraintes, une plate-forme hétérogène comportant des composants différents de type GPP, DSP, FPGA, chacun ayant une réponse spécifique à une contrainte particulière, est la solution la plus adaptée.

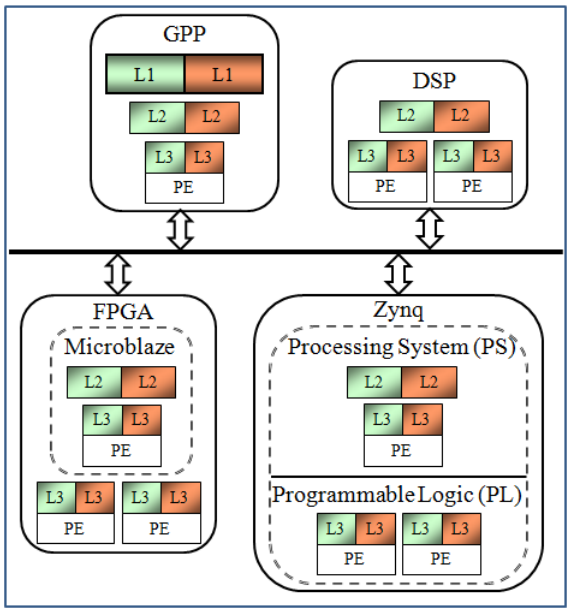


FIGURE 10 – A schematic example of HDCRAM architecture.

Lors d'une thèse précédente le déploiement de HDCRAM sur une cible logicielle de type GPP a été abordé avec succès. Dans cette thèse, notre objectif est de déployer HDCRAM sur une cible matérielle de type FPGA sur une plate-forme hétérogène, c'est précisément l'objet du chapitre 2 suivant que d'étudier ce déploiement.

2 Implantation de HDCRAM sur plateformes hétérogènes

Comme nous l'avons indiqué précédemment HDCRAM a déjà, lors de travaux précédents, été implémenté sur des ressources logicielles. Le but de ce chapitre est donc d'implémenter HDCRAM sur des ressources matérielles, plus précisément sur des FPGA en tirant profit de la Reconfiguration Partielle de FPGA.

2.1 Reconfiguration Partielle de FPGA

La Reconfiguration Partielle de FPGA permet de modifier dynamiquement des fonctions dans certaines zones du FPGA, permettant à l'application de continuer à fonctionner sur les autres zones, sans aucune interruption des données, du service. En d'autres termes la RP apporte une souplesse équivalente à celle du logiciel dans le monde du matériel.

Pour les circuits Virtex la RP est réalisée au travers du port ICAP, qui lit le bitstream partiel correspondant à la nouvelle fonction. Pour les circuits Zynq-7000, celle-ci est réalisée soit au travers du port ICAP ou par l'intermédiaire du port PCAP.

2.2 Implémentation de HDCRAM sur plate-forme Virtex5

La carte Xilinx ML506 (voir l'annexe B qui décrit cette carte d'évaluation) est connectée à un PC. Le niveau 1 de HDCRAM est implanté sr le PC, donc sur le FPGA sont implantés les niveau 2 et 3 tel que cela est représenté sur la figure 11. L'ensemble des connections entre les différents éléments utilise le protocole UDP, ce qui donne une très grande souplesse car les éléments sont repérables par leurs adresses IP. Un UDP Core a été spécifiquement développé pour les FPGA dans le cadre de cette thèse. Celui-ci est complètement décrit en Annexe A.

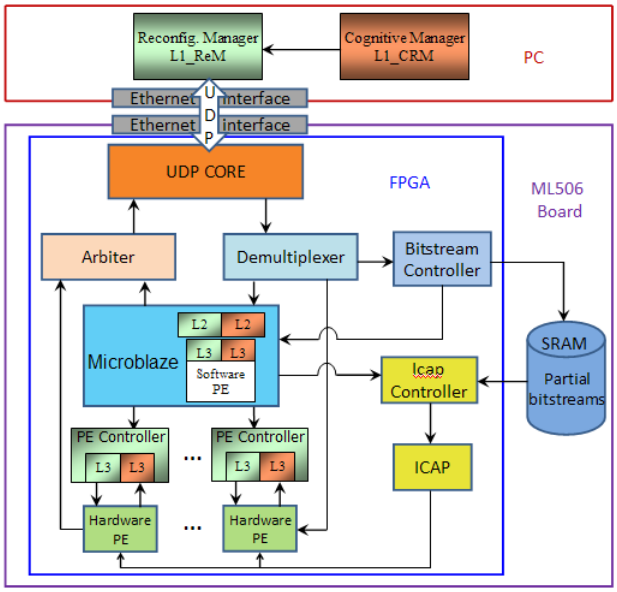


FIGURE 11 – The block diagram of the management platform.

2.3 Implémentation de HDCRAM sur plate-forme Zynq 7000

La carte d'évaluation ZC702 est totalement décrite en Annexe C. Elle comprend un dual core ARM CORTEX pour la partie Processing System (PS) et un FPGA Xilinx

Artix-7 pour la partie Programable Logic (PL). La figure 12 présente l'implémentation de HDCRAM sur cette plate-forme.

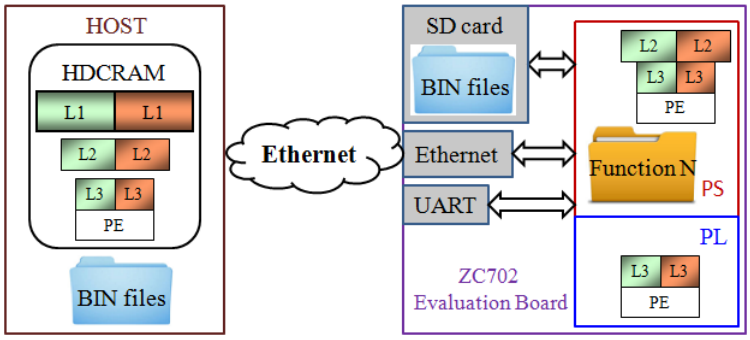


FIGURE 12 – The HDCRAM implementation on the ZC702 evaluation board.

Le niveau L1 est implanté sur le PC. Un niveau L2 est implanté sur la partie PS de la carte. Un opérateur peut être implanté soit en logiciel soit en matériel, par conséquent le niveau L3 associé sera soit sur PS soit sur PL.

Les bitstreams de configuration peuvent être mémorisés soit sur le PC soit sur la mémoire de la carte.

3 Etude des métriques liées à la plate-forme dans un contexte d'Ecoradio Intelligente

Dans ce chapitre, nous nous intéressons aux métriques permettant de caractériser le fonctionnement d'un équipement et d'optimiser ce fonctionnement d'un point de vue de l'efficacité énergétique. Les différentes métriques accessibles sur une plate-forme sont identifiées, en particulier, celles liées à l'électronique de l'équipement. Chacune est discutée pour savoir comment elle peut être obtenue, et comment elle peut être utilisée pour prendre une décision. Ensuite, ces métriques sont discutées sous différents aspects : accessibilité, statique/dynamique, facilité d'utilisation... Finalement, l'implémentation d'un filtre FIR en série ou parallèle, est discuté sous ses aspects métriques.

3.1 Les différentes métriques

Dans cette section, toutes les métriques identifiées et discutées l'ont été à partir de la plate-forme Xilinx Virtex-5 ML506. Rappelons que cette plate-forme est totalement décrite en Annexe B. Il est donc tout à fait possible que certaines métriques ne s'appliquent pas facilement à d'autres plates-formes.

Parmi les métriques identifiées, nous avons la tension, le courant, la fréquence,... Nous décrivons certaines plus en détail ci-après.

3.1.1 La température

La température est une métrique très utile. Elle peut-être obtenue par les outils Xilinx, tel System Monitor, mais sera, dans ce cas très difficile, voire impossible, à utiliser en fonctionnement. Une autre façon de l'obtenir indirectement est d'utiliser un thermomètre numérique basé sur un Ring Oscillator, dans ce cas il sera possible de l'utiliser en fonctionnement. Nous savons qu'il existe une relation linéaire entre la fréquence de l'oscillateur et la température de la zone. Ce thermomètre utilise très peu de ressources et peut être placé à différents endroits du circuit. Son schéma est donné à la figure 13.

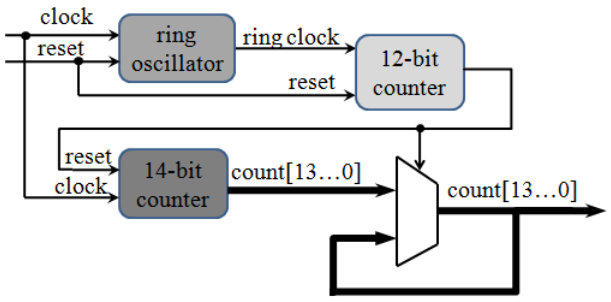


FIGURE 13 – The digital thermal sensor.

Dans d'autres travaux [15], nous avons montré qu'il existe une relation entre la température et la consommation statique (voir figure 14), cette métrique peut donc servir à connaitre la consommation statique et prendre les décisions adéquates. Elle peut aussi être utilisée pour des décisions de sauvegarde comme diminuer la fréquence, la charge de travail, mettre en œuvre le refroidissement pour éviter une surchauffe.

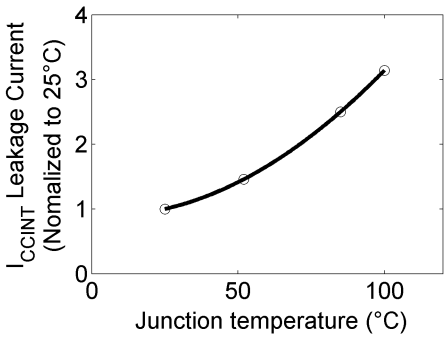


FIGURE 14 – Leakage current variations with Temperature.

3.1.2 La ressource disponible, la surface et la position d'une fonction

Ces métriques sont connectées les unes aux autres. Elles peuvent être obtenues par le même outil Xilinx PlanAhead. Lors de la phase de design un opérateur peut être dans une certaine position, mais il peut être nécessaire de le modifier en cours de fonctionnement grâce à la RP. Ce cas de figure se produira si plusieurs opérateurs occupent la même position et sont instanciés à différents moments grâce à la RP.

3.1.3 Le taux d'activité

Le taux d'activité d'un opérateur est donné par l'équation (1) :

$$Activity\,rate = \frac{en \times N}{c} \times 100\% \tag{1}$$

Avec c : le durée de la mesure exprimée en nombre de cycles d'horloge.

en : le nombre de cycles d'horloge autorisant l'entrée des données pendant la durée de la mesure.

N : une constante qui donne le nombre de cycles nécessaire pour le traitement d'une donnée en entrée pour obtenir une donnée en sortie.

La Fig. 15 donne un exemple. Dans ce cas si N = 1, pendant c = 10 cycles, le taux d'activité est = 20 % ; si N = 5, alors taux activité = 100 %. Bien sûr ceci est un exemple très simple. c doit être choisi soigneusement, plus c sera grand plus le taux d'activité sera précis.

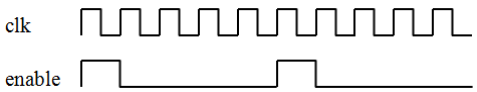


FIGURE 15 – A timing diagram example.

3.1.4 Implantation série/parallèle

De nombreux opérateurs peuvent être implantés soit en mode série soit en mode parallèle. Prenons l'exemple du calcul de c donné par l'équation (2).

$$c = \sum_{i=0}^{N-1} a[i] \times b[i] \tag{2}$$

Celle-ci peut-être implantée en mode parallèle (figure 16) ou en mode série (figure 17). Cette métrique peut être définie lors de la phase de design ou modifiée dynamiquement en fonction d'autres métriques (voir l'exemple suivant dans ce chapitre).

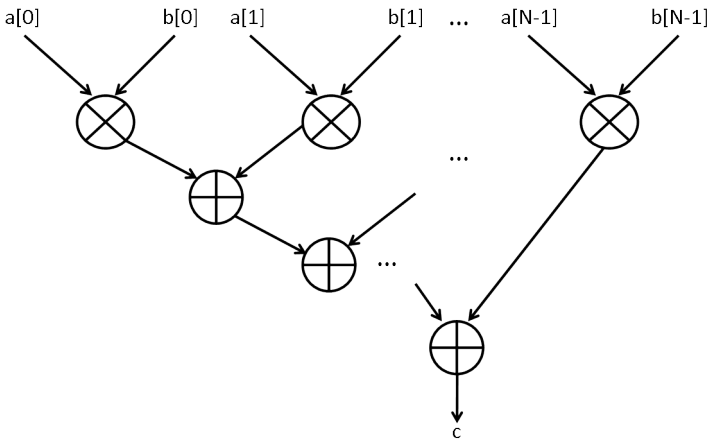


FIGURE 16 – Parallel method.

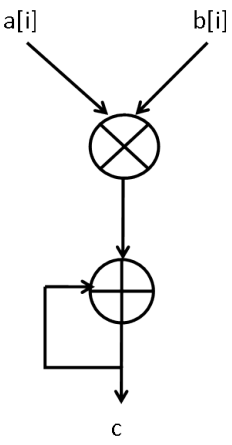


FIGURE 17 – Serial method.

3.1.5 La consommation

Cette métrique est évidemment de la plus grande importance dans notre contexte d'Ecoradio Intelligente. Elle peut être obtenue à partir de la connaissance de la tension

et du courant. Elle peut l'être aussi, à partir des outils Xilinx Power Estimator (XPE) et plus particulièrement Xilinx Power Analyzer (XPA), que nous utiliserons par la suite.

Suivant la valeur de cette métrique, l'organe de décision peut décider de modifier tel ou tel paramètre pour diminuer cette consommation.

3.2 Discussion sur les différentes métriques

Certaines métriques sont fixes alors que d'autres peuvent évoluer dans le temps, elles sont alors dynamiques. Certaines s'obtiennent facilement alors que d'autres sont beaucoup plus difficiles à obtenir, c'est la notion d'accessibilité. Certaines sont reconfigurables, d'autres non, alors que d'autres peuvent se modifier suite à une reconfiguration de certaines. C'est le niveau de reconfigurabilité. Est aussi identifié le niveau d'impact de la métrique sur la consommation énergétique.

Le tableau 2 ci-dessous résume cette discussion :

TABLE 2 – Consideration of the metrics.

Metrics	Self-changeability	Configurability	Green impact	At which Level	Susceptibility
Voltage	static	medium	strong	System	Low
Current	dynamic	unconfigurable	strong	system	medium
Frequency	static	easy	strong	PE	Low
Temperature	dynamic	unconfigurable	strong	system	high
Area	static	medium	medium	PE & system	Low
Position	static	medium	weak	PE	Low
Resource	static	difficult	strong	PE & system	Low
Activity rate	dynamic	unconfigurable	medium	PE	medium
Serial / parallel	static	easy	medium	PE	low
Power consumption	dynamic	unconfigurable	strong	PE & system	medium
Performance to power consumption ratio	dynamic	unconfigurable	strong	PE	medium
Working mode	static	easy	strong	system	low

La fréquence de fonctionnement d'un opérateur ou PE peut être reconfigurée par l'intermédiaire d'un Digital Clock Manager (DCM), par conséquent, il s'agit d'une métrique reconfigurable pour cette plate-forme. Le mode Série/Parallèle est similaire si l'on a la

possibilité de commuter entre différentes options existantes notamment par RP. La surface et la position sont reconfigurables notamment grâce à la RP. Mais la ressource ne l'est pas car celle-ci est définie lors de la configuration de l'opérateur.

En ce qui concerne l'impact Eco, certaines métriques telles que la tension, le courant, la fréquence, la température,la ressource, la consommation... ont un impact important et certaines ont une influence directe sur la consommation. La position d'un opérateur n'a pas d'impact alors que le mode série/parallèle a un impact complexe et indirect. C'est précisément cette métrique que nous étudions en détail dans la section suivante.

3.3 Etude d'un cas particulier : implantation série ou parallèle d'un filtre

Le filtrage est une fonction classique et nécessaire dans tout équipement de radiocommunications. Il peut s'agir de filtrer une bande de fréquence pour éviter de polluer les bandes adjacentes, ou de filtrer une bande d'intérêt pour optimiser le convertisseur analogique/numérique, ou de réaliser un filtre de Nyquist,...

Très classiquement, ces filtres sont réalisés à partir de filtre à Réponse impulsionnelle finie (FIR). Un FIR comporte des retards et des coefficients (multiplieurs). Il peut être implanté sous forme parallèle ou série chacune ayant ses avantages. La figure de la section précédente présente ces deux possibilités. De manière évidente la forme parallèle sera plus rapide mais consommera plus de ressources que le mode série. Nous pourrions donc en déduire que la forme parallèle consommera plus, mais cela n'est pas si simple. Lors d'une implémentation d'un filtre FIR à 32 coefficients avec 32 MAC (multiplieurs /accumulateurs) pour le mode parallèle et un seul MAC pour le mode série le tableau 3 confirme que le mode parallèle consomme plus de ressource :

TABLE 3 – Resources used by the two implementation architectures.

Architecture	#FF	#LUTs	#DSPs
Parallel	3051	1067	64
Serial	1192	542	2

Pour étudier l'influence sur la consommation, nous avons implanté ce filtre sur le FPGA, sans aucune autre fonction. La consommation est estimée en utilisant l'outil XPA.

TABLE 4 – Power consumption of the FIR filter.

Frequency (MHz)	Power consumption (W)					
	parallel			serial		
	Dynamic	Quiescent	Total	Dynamic	Quiescent	Total
40	0.058	0.545	0.603	0.019	0.544	0.564
50	0.069	0.545	0.614	0.022	0.544	0.566
66.67	0.087	0.545	0.632	0.025	0.544	0.570
75	0.096	0.545	0.641	0.027	0.544	0.572
100	0.123	0.545	0.668	0.033	0.545	0.578
125	0.150	0.545	0.695	0.039	0.545	0.583
133.33	0.159	0.545	0.704	0.040	0.545	0.585
150	0.176	0.546	0.722	0.044	0.545	0.589
166.67	0.194	0.546	0.740	0.048	0.545	0.592

On peut constater que la consommation dynamique est plus faible pour le mode série à fréquence identique. Mais comparer à fréquence identique n'est pas une comparaison intéressante. En effet, la comparaison doit se faire à débit identique, ce qui nécessitera d'augmenter la fréquence de fonctionnement du mode série et, comme nous le verrons, inversera la conclusion sur la consommation. Notons aussi sur ce tableau, que la consommation totale pour les 2 modes est très proche. Comme l'outil XPA ne donne qu'une consommation totale du FPGA, il est très difficile de conclure sur la consommation statique du filtre suivant le mode d'implantation. En effet cette surface, quel que soit le mode, est faible relativement à l'ensemble du FPGA. Elle aura donc peu d'influence sur la consommation totale.

Nous nous focalisons maintenant sur la consommation dynamique. Nous implantons dans cette étude les deux modes de manière à obtenir le même débit en sortie du filtre. La figure 18 présente la consommation dans ce contexte.

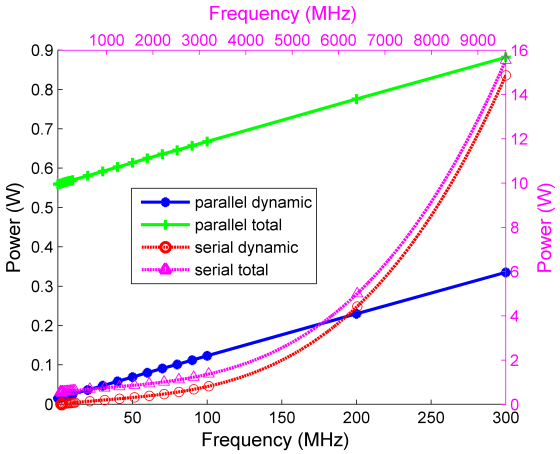


FIGURE 18 – The power consumption.

A cause de la différence importante en consommation entre les deux modèles les axes concernant l'architecture parallèle sont ceux de gauche et bas alors que pour l'architecture série ce sont ceux de droite et haut.

En mode parallèle la consommation croît presque linéairement et reste sous 1W, mais en mode série, pour garder le même débit, la fréquence doit aller de 3.2 MHZ à 9600MHz. En dessous de 3200MHz la consommation est sous 1,5 W mais croît très rapidement ensuite jusqu'à croiser la courbe parallèle. Cela s'explique facilement par le fait que l'horloge est l'élément qui consomme le plus à l'intérieur du FPGA, à partir d'une certaine fréquence elle devient prépondérante.

Comme la Figure 18 a des axes différents, il est difficile de comparer les détails des 2 architectures. Les 2 figures (Figure 19 et Figure 20) suivantes font un zoom sur la région 0.1 MHz à 1MHz.

On constate que le mode parallèle consomme plus que le mode série mais que l'inverse se produit pour une fréquence série supérieure à 25.6 MHz.

Une première conclusion (surprenante) de cette étude consiste à dire que sous une certaine fréquence il est préférable d'utiliser le mode série, qui utilise moins de ressource et consomme moins, et qu'au-delà il est préférable d'utiliser le mode parallèle. Cette conclusion sera très utile pour prendre les bonnes décisions dans HDCRAM.

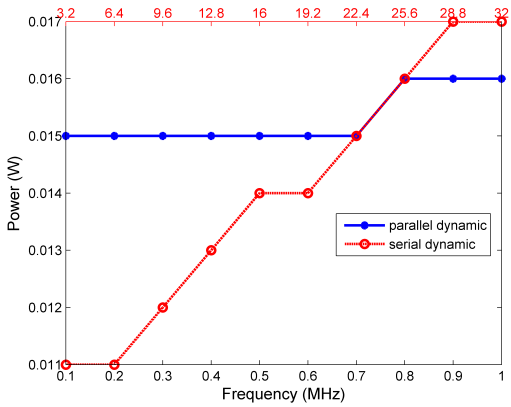


FIGURE 19 – The dynamic power.

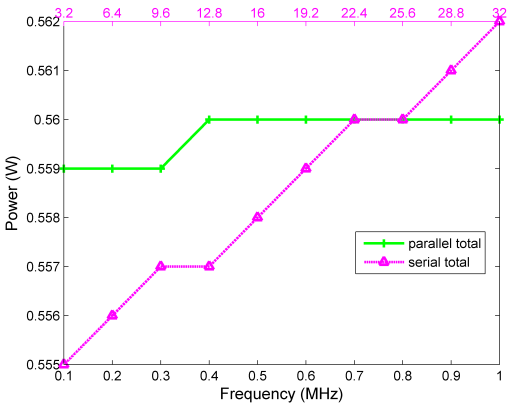


FIGURE 20 – The total power.

3.3.1 Influence du nombre de coefficients

Nous souhaitons étudier dans cette section l'influence du nombre de coefficients sur la consommation. Pour cela nous avons implanté des filtres de 32, 64, 128 coefficients. La Fig. 21 donne la consommation, en fonction de la fréquence pour les 3 longueurs de filtre. Comme attendu le filtre comprenant le plus grand nombre de coefficients consomme le plus, lorsqu'ils travaillent à la même fréquence. A mesure que la fréquence augmente, la consommation croît plus rapidement avec le filtre le plus long.

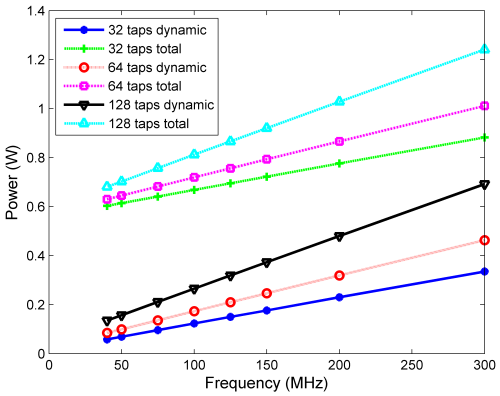


FIGURE 21 – Power consumption of the filter with three different numbers of taps when the frequency increases.

La Fig. 22 montre la consommation en fonction du nombre de coefficients. A 40MHz, la consommation croît faiblement quand le nombre de coefficients augmente, mais quand la fréquence passe à 300MHz, la consommation croit nettement plus vite.

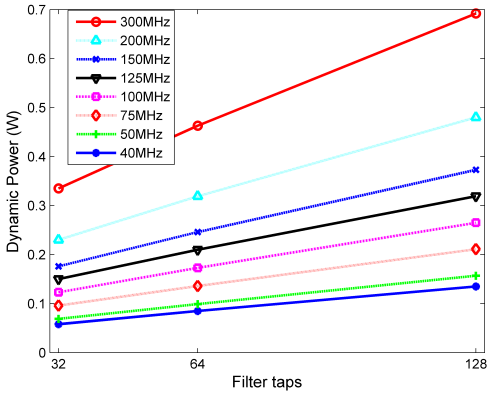


FIGURE 22 – Dynamic power consumption according to the number of taps.

Ce qui veut dire que plus le filtre est long et plus le débit souhaité sera élevé plus la consommation sera élevée. Ce résultat n'est pas surprenant.

3.3.2 Gestion de ces métriques par HDCRAM

Les métriques utilisées dans cette analyse sont :

- Série/parallèle
- Fréquence
- Consommation
- Ressource

Lorsque celles-ci ont été obtenues elles sont alors utilisées par les algorithmes de décision de HDCRAM. La Fig. 23 donne un exemple d'utilisation de celles-ci. Les trois opérateurs (générateur de fréquence DCM, filtre, calcul de la ressource) sont gérés par leurs gestionnaires respectifs de niveau 3.

- Si le débit demandé est faible (< 0.8 MHz) alors le gestionnaire de niveau 3 va prendre la décision de fonctionner en mode série.
- Si le débit demandé est élevé, alors il est possible de l'atteindre en augmentant la fréquence de travail ou en augmentant le nombre de MACs. Le niveau L3 ne peut

pas prendre de décision seul, l'information remonte donc au niveau L2 qui pourra prendre une décision. Par exemple pour doubler le débit, si l'opérateur ressource indique que celle-ci est suffisante, L2 peut décider de multiplier par 2 le nombre de MAC sinon, il décidera d'augmenter la fréquence de fonctionnement. Cette décision de L2 se répercutera sur les L3 de reconfiguration de DCM et du filtre.

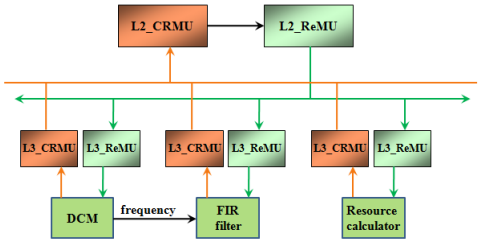


FIGURE 23 – An example of level 2 HDCRAM management.

4 Implantation d'un système émission/réception OFDM

Afin de valider les différentes études des chapitres précédents, nous avons implanté sur une plate-forme une transmission réelle qui intègre l'ensemble de nos propositions. Cette transmission est basée sur une modulation de type OFDM largement utilisée aujourd'hui dans de nombreux standards et qui a fait ses preuves, notamment pour lutter contre les évanouissements sélectifs.

La figure 24 présente la gestion avec HDCRAM du système OFDM. Afin de démontrer l'aspect distribué de HDCRAM et l'aspect plate-forme hétérogène de notre réalisation, l'émetteur est implanté sur un PC (GPP) alors que le récepteur est implanté sur la plate-forme Zynq. L'émetteur est considéré comme la station de base alors que le récepteur est considéré comme le terminal. Le lien entre le PC et la plate-forme se fait par le protocole UDP à travers Ethernet.

Plusieurs scénarios ont été étudiés, parmi ceux-ci trois sont présentés dans ce résumé.

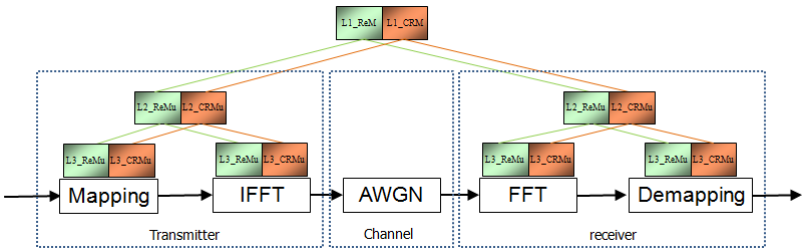


FIGURE 24 – The block diagram of a simplified OFDM system model.

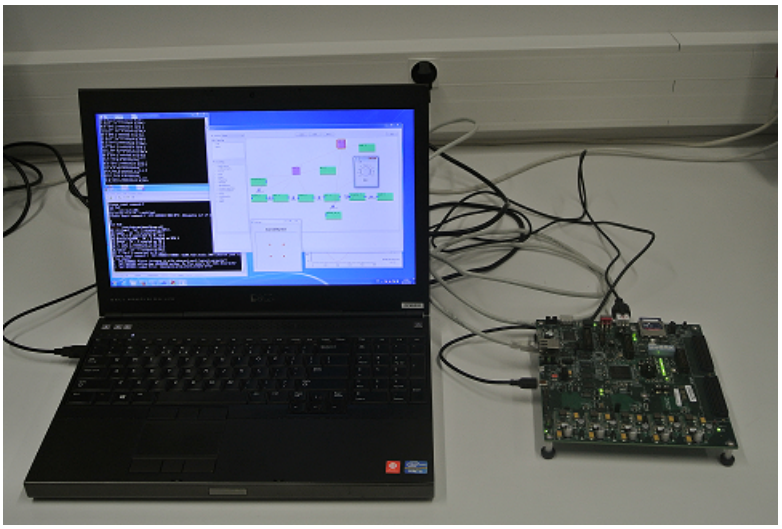


FIGURE 25 – Implementation platform.

4.1 Implantation de la FFT par RP

Comme la FFT est la fonction la plus coûteuse en termes de calculs, elle peut avantageusement être réalisée en matériel sur PL (voir Figure 26), même s'il est possible de l'implanter en logiciel sur PS (voir Figure 27).

Dans un but d'optimisation de la surface et de la consommation, nous proposons d'implanter la FFT par reconfiguration partielle, ce qui nous permettra, au lieu d'implanter une grosse FFT reconfigurable entre toutes les tailles nécessaires, de choisir et modifier à la volée la FFT de bonne taille pour le scénario envisagé. Deux types d'implantation sont considérées dans ce travail : une architecture de type pipe-line et une autre de type Radix2 (voir Annexe D).

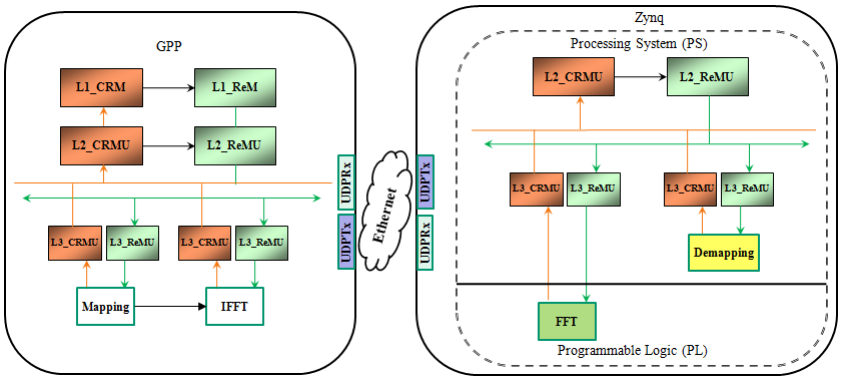


FIGURE 26 – The hardware implementation of FFT.

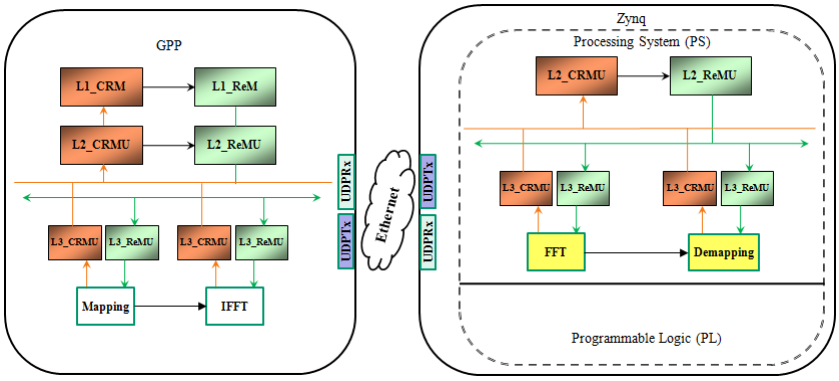


FIGURE 27 – The software implementation of FFT.

La figure 28 présente l'implantation de la FFT Radix2 pour différentes tailles en utilisant la RP.

Le tableau 5 suivant dresse une comparaison des ressources nécessaires pour les différents cas envisagés. Ces résultats doivent être comparés avec ceux du tableau 6 correspondant au cas d'une FFT reconfigurable entre toutes les tailles envisagées. On constate que la ressource nécessaire est supérieure à celle de la FFT de plus grande taille : cela est du à la logique de contrôle pour la reconfiguration entre toutes les tailles.

Les différents temps de transformation sont listés dans les tableaux 7. Il apparaît que ce temps est le plus faible avec l'architecture pipe-line hardware au prix d'une ressource supérieure comparée à l'architecture Radix 2. Le temps supplémentaire lié à la RP doit aussi être pris en compte il est donné tableau 8.

TABLE 5 – Resources available and used by different FFT implementations in the reconfigurable region.

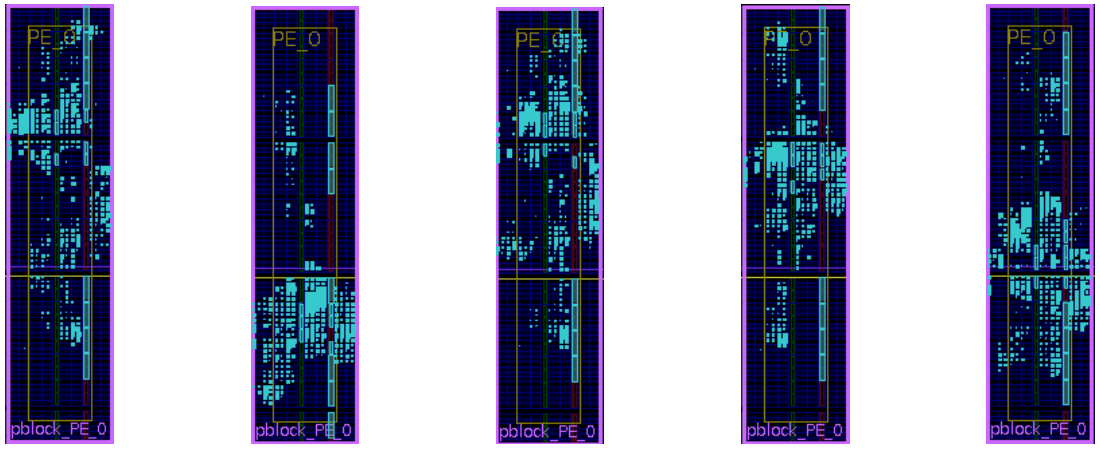
Transform length	Resource	LUT	Register	SLICE	DSP48E1	BRAM
	Available	5184	10368	1359	32	48
128	pipelined	2806	3196	702	9	9
	radix-2	1067	1316	268	3	11
256	pipelined	3175	3578	795	9	10
	radix-2	1101	1361	276	3	11
512	pipelined	3589	4113	898	12	12
	radix-2	1154	1392	289	3	11
1024	pipelined	3993	4507	999	12	14
	radix-2	1153	1425	289	3	11
2048	pipelined	4455	5086	1114	15	19
	radix-2	1194	1491	299	3	13

TABLE 6 – Resources used by traditional reconfigurable FFT implementation with pipelined architecture.

LUT	Register	SLICE	DSP48E1	BRAM
5741	6056	1435	15	19

TABLE 7 – The transform time of different FFT implementations.

Transform length	Software (μs)	Hardware (μs)		Traditional reconfigurable FFT (μs)
		pipelined	radix-2	
128	166	4.81	8.29	4.92
256	364	8.71	16.77	8.93
512	798	16.49	34.85	16.61
1024	1751	31.91	73.41	32.13
2048	3867	62.73	155.49	62.85



(a) FFT128 (b) FFT256 (c) FFT512 (d) FFT1024 (e) FFT2048

FIGURE 28 – Implementation of FFT with single radix-2 architecture using partial reconfiguration.

TABLE 8 – Full and partial configuration time of the FFT design.

Type	Size (bytes)	Time (μs)
Full	4 045 564	211 413
Partial	384 512	35 122

De même les tableaux 9 et 10 comparent la consommation des différentes architectures.

TABLE 9 – The power consumption of different FFT implementations of the DPR approach.

Transform length	Power consumption (W)	
	pipelined	radix-2
128	0.103	0.096
256	0.105	0.097
512	0.108	0.098
1024	0.113	0.099
2048	0.121	0.101

TABLE 10 – The power consumption of software FFT and traditional reconfigurable FFT.

	Software FFT	Traditional reconfigurable FFT
Power consumption (W)	0.12	0.135

En conclusion, le meilleur compromis sur tous ces critères est l'implantation de la FFT avec l'architecture pipe-line en utilisant la RP.

4.2 Différents scénarios d'Ecoradio Intelligente

4.2.1 Adaptation de la constellation

Dans ce scénario, la constellation est modifiée en fonction du SNR du signal reçu. Ce scénario implique qu'il y ait une communication entre le terminal et à la station pour lui demander de modifier la constellation émise.

Remarque : Ce scénario, dans une version figée a déjà été implanté au laboratoire et une démonstration existe déjà. Cependant, dans cette nouvelle version le scénario est complètement géré par HDCRAM, ce qui offre une très grande souplesse.

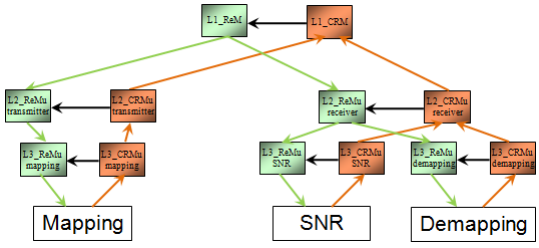


FIGURE 29 – Scenario 1.

L3 CRMu SNR

Le capteur SNR mesure le niveau de bruit et envoie cette information au niveau L2 CRMu supérieur.

L3 CRMu demapping

Informe le L2 de la modulation en cours.

L2 CRMu receiver

En se basant sur la valeur de SNR remontée par le niveau L3, L2 CRMu receiver prend une décision très simple pour adapter la modulation :

- Si 5dB < SNR <= 10dB, la QPSK sera choisie.

- Si SNR > 10dB, la 16QAM sera utilisée.

En fonction de la modulation en cours, L2 CRMu receiver informe ou non le niveau L1 d'une reconfiguration de la modulation.

L1 CRM

Si L1 ReM reçoit une commande de son L1 CRM associé, alors il envoie l'ordre aux L2 ReMu transmitter et L2 ReMu receiver concernés.

L2 ReMu transmitter

Si le L2 ReMu transmitter reçoit l'ordre du L1 ReM il execute l'action et envoie l'ordre de reconfiguration au L3 ReMu mapping.

L3 ReMu mapping

Le L3 ReMu mapping gère la reconfiguration de son opérateur Mapping associé.

Le déroulement est complètement identique entre L2 ReMu receiver L3 ReMu demapping côté récepteur et n'est pas rappelé ici.

4.2.2 Gestion de la FFT en fonction du niveau de batterie

Comme nous l'avons vu précédemment la FFT peut être réalisée en logiciel, matériel architecture pipeline, matériel architecture radix2. C'est cette dernière possibilité qui sera choisie si le niveau de batterie est faible.

4.2.3 Gestion de la taille de la FFT en fonction du standard à utiliser

Ce scénario montre comment il est possible de modifier la taille de la FFT en fonction, par exemple, d'un changement de standard à démoduler. Il s'agit par exemple de modifier la bande de 1.25MHz to 2.5MHz pour le LTE donc modifier la taille de FFT de 128 à 256.

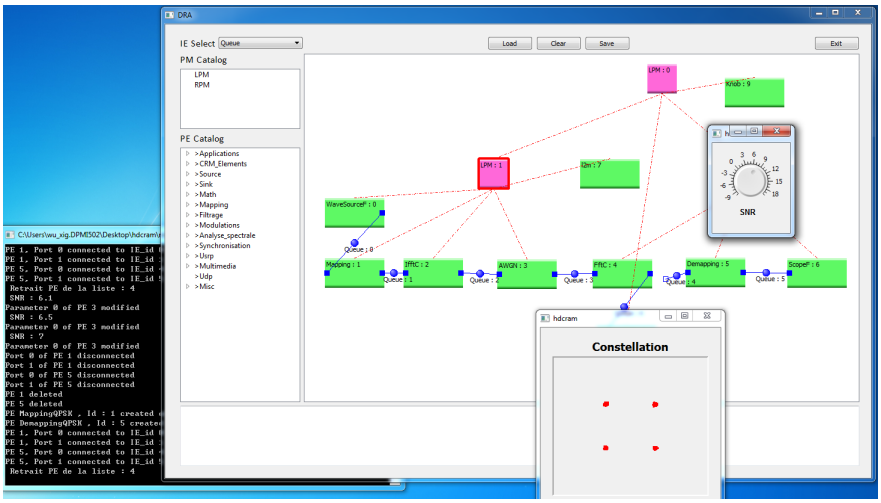


FIGURE 30 – Adaptation to QPSK when SNR < 10.

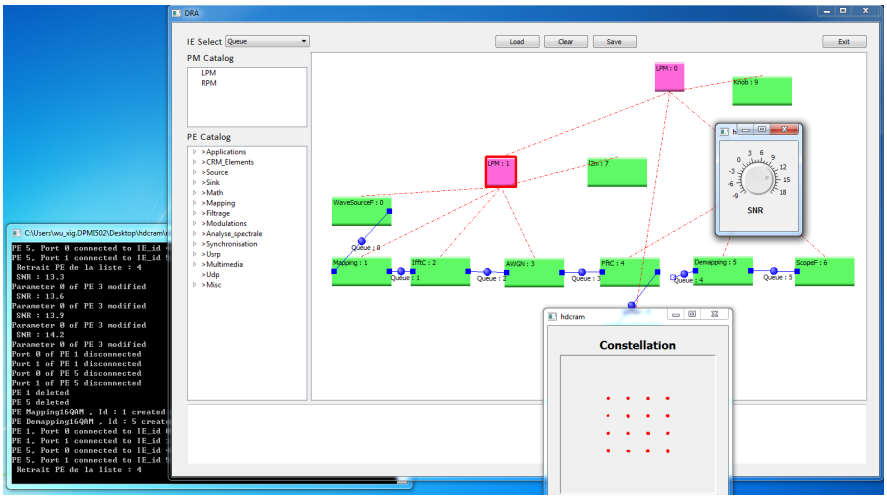


FIGURE 31 – Adaptation to 16QAM when SNR > 10.

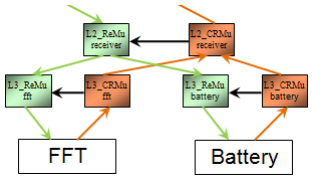


FIGURE 32 – Scenario 3.

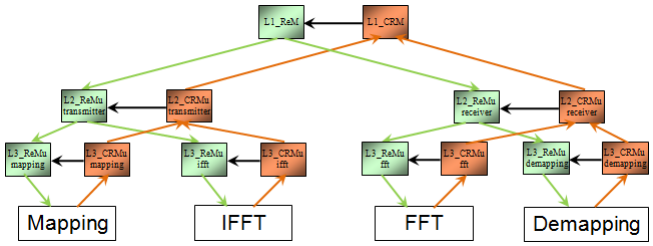


FIGURE 33 – Scenario 4.

5 Conclusion et Perspectives

L'objectif de cette thèse était l'étude et l'implantation sur plate-forme hétérogène du gestionnaire de RI HDCRAM, en se focalisant sur la ressource matérielle (notamment FPGA) et en tirant profit de la RP de FPGA. Cela dans un contexte d'Ecoradio intelligente, donc des équipements qui obéissent au cycle intelligent classique sous contrainte d'économie d'énergie. Une partie du travail a consisté à identifier et étudier, les métriques qui sont utiles dans un tel contexte. L'implantation de scénarios répondant à l'ensemble de cette thèse a permis de valider ce qui a été proposé, et cela a finalement aboutit à un démonstrateur qui sera démontré lors de diverses occasions et en particulier lors de la soutenance.

Bien entendu, ce travail laisse ouvert un certain nombre de pistes, parmi lesquelles l'étude d'algorithmes de prise de décision, plus performants que les machines d'états utilisées dans les scénarios, serait très intéressante. Une autre évolution intéressante serait d'interfacer HDCRAM avec des bibliothèques d'opérateurs en open-source, de manière à pouvoir utiliser facilement et rapidement d'autres opérateurs.

Abstract

As the digital communication systems evolve from GSM and now toward 5G, the supported standards are also growing. The desired communication equipments are required to support different standards in a single device at the same time. And more and more wireless Internet services have been being provided resulting in the explosive growth in data traffic, which increase the energy consumption of the communication devices thus leads to significant impact on global CO_2 emission. More and more researches have focused on the energy efficiency of wireless communication. Cognitive Radio (CR) has been considered as an enabling technology for green radio communications due to its ability to adapt its behavior to the changing environment. In order to efficiently manage the sensing information and the reconfiguration of a cognitive equipment, it is essential, first of all, to gather the necessary metrics so as to provide enough information about the operating condition thus helping decision making. Then, on the basis of the metrics obtained, an optimal decision can be made and is followed by a reconfiguration action, whose aim is to minimize the power dissipation while not compromising on performance. Therefore, a management architecture is necessary to be added into the cognitive equipment acting as a glue to realize the CR capabilities. We introduce a management architecture, namely Hierarchical and Distributed Cognitive Radio Architecture Management (HDCRAM), which has been proposed for CR management by our team. This work focuses on the implementation of HDCRAM on heterogeneous platforms. One of the objectives is to improve the energy efficiency by the management of HDCRAM. And an example of a simplified OFDM system is used to explain how HDCRAM works to efficiently manage the system to adapt to the changing environment.

Introduction

Energy efficiency has attracted more and more attention due to the fact that the information and communication technology (ICT) industry consumes 2% to 10% of the world's overall energy [16] and is becoming one of the major contributors to the world-wide CO_2 emission [17]. It has been predicted that the CO_2 emissions of mobile communication systems will increase by a factor of three between 2007 and 2020 [18]. Therefore, the ICT industry is playing an increasingly important role in reducing greenhouse gas emissions, and has a profitable opportunity to foster energy efficiency in other sectors, thus helping to decrease the carbon footprint at the global level [19]. All sides, including users, operators, governments, and academia etc., have driven the research on energy efficiency.

Cognitive Radio (CR) [10, 11, 14] has been considered as an enabling technology for green radio communications due to its ability to adapt its behavior to the changing environment [3, 20].

A cognitive equipment, which applies the cognitive cycle into an equipment, is able to sense the surrounding environment, make decisions depending on the information it has obtained, and then take actions to adapt its behavior to the changing environment (including the updated constraints and requirements). In order to achieve this goal, it needs to gather the necessary metrics so as to provide enough information pertaining to its operating condition thus enabling decision making. Therefore, it is important to select the proper and useful metrics in order to evaluate the system's performance and to reconfigure the system. In [21], the authors have reviewed the performance metrics at the node, network, and application levels. We focus on the equipment level and introduce some possible metrics on a FPGA platform and the ways to measure them. All kinds of information about the environment of the cognitive equipment can be considered as me-

trics. These metrics cover many aspects : performance, power consumption, temperature, and resources, etc.

After obtaining the metrics, we introduce a management architecture, namely Hierarchical and Distributed Cognitive Radio Architecture Management (HDCRAM) [22, 23, 12] into an equipment, which has been proposed for CR management by our team, to efficiently manage the cognitive equipments.

In order to design cognitive equipments, flexible and efficiently reconfigurable hardware platforms are necessary. Many hardware platforms can be used to design cognitive equipments. Those include General Purpose Processors (GPPs), Digital Signal Processors (DSPs), Field Programmable Gate Arrays (FPGAs), Application Specific Integrated Circuits (ASICs), etc.

GPP is the most flexible platform, but has poor performance. DSP is flexible like GPP and has its advantage when dealing with signal processing applications, but its performance is still not good enough. ASIC provides high performance but with less flexibility.

FPGA becomes a favorable choice since it has some kind of flexibility and its performance is close to that of ASIC. Besides, modern FPGA integrates embedded processors to provide more flexibilities. Recently, some FPGA families have provided a Dynamic Partial Reconfiguration (DPR) technique. DPR is the ability to dynamically reprogram a subset of the logic within an operating FPGA. This is done thanks to the download of a partial configuration file while the remaining logic continues to operate without interruption [24]. Benefiting from these features, FPGA is more suitable for developing cognitive equipment.

By taking advantage of the DPR, it is possible to dynamically change the functionality of part of the FPGA , which makes the hardware software-like. This capability enables different functionalities to be implemented in the same portion of the device. Therefore, the same system can be implemented in small devices featuring less resource, meanwhile, saving cost and reducing power consumption. This is especially useful for Software Defined Radio (SDR) and CR. It is possible to implement multi-mode multi-band radios in the same device.

HDCRAM can be implemented on heterogeneous platforms, different platforms are connected by Ethernet and communicate using UDP protocol. By this approach, it is

easy to add new platforms and remove old unused platforms, which makes the system scalable.

We take the HDCRAM management of a simplified Orthogonal Frequency Division Multiplexing (OFDM) [25] system as an example to offer the possibility to glue almost all the aspects of the work introduced in this thesis. In this example, we introduce some metrics and their corresponding cognitive cycles. Then, how HDCRAM manages these sensed metrics, decision making, and the reconfiguration of the system to adapt to the changing environment is explained.

The thesis is organized as follows.

Chapter 1 presents the trend and motivations toward green communications and some relevant projects. Cognitive radio, as an enabling technique for green communications, is introduced. We treat CR in a general vision in this thesis. In order to efficiently manage the CR features, a management architecture, HDCRAM proposed by our team, is also described.

Chapter 2 explains the implementations of HDCRAM first on Virtex 5 platform and then on a more flexible Zynq-7000 platform. We first implemented HDCRAM on Xilinx Virtex 5 platform, and developed a hardware UDP core to provide a high speed data transmission. The software processing elements (PE) are implemented on a soft microprocessor core Microblaze, and the hardware PEs are implemented in hardware. The management units of the hardware PEs can be implemented in hardware or in software, or part of in software executed on Microblaze and another part in hardware. Due to some limitations of the Virtex 5 platform, we then implemented HDCRAM on a more flexible platform, Xilinx Zynq-7000 platform, which integrates a dual-core ARM Cortex-A9 as PS and a Xilinx's 7 series FPGA Artix-7 as PL in a single device.

As discussed above, the management architecture needs proper metrics to sense the surroundings and efficiently reconfigure the system thus adapting to the working environment. Chapter 3 mainly introduces some metrics on a FPGA platform that are useful for the HDCRAM management architecture to efficiently manage the equipment, as well as some measurement approaches of the metrics. We study the power consumption of a FIR filter when it is implemented in parallel and serial modes and works in different

frequencies as a use case. The results are useful for HDCRAM to make decisions, which suggest that it is better to work in serial mode when the frequency is low, otherwise, the parallel method is recommended. We also analyze the power consumption when the filter is implemented with three different numbers of taps, which shows that there is a trade off between the power consumption, the performance, and the resources.

In Chapter 4, we employ a simplified OFDM system model, and discuss a HDCRAM management scenario of the OFDM transmitter and receiver. Because of several advantages over the traditional reconfigurable FFT, dynamic partial reconfiguration technique is used to reconfigure the hardware FFT. The implementation is based on Xilinx Zynq-7000 platform described in Chapter 2. Some metrics introduced in Chapter 3 are used in this example. The OFDM transmitter and receiver is managed by HDCRAM architecture presented in Chapter 1. It shows that the HDCRAM can easily plan all scenarios presented in this example.

Finally, Chapter 5 concludes this thesis and also discusses about the future research directions.

Appendix A explains the hardware UDP core developed on Xilinx ML506 board used in Chapter 2. Appendix B and Appendix C briefly introduce the Xilinx ML506 evaluation board and ZC702 evaluation board utilized in Chapter 2, respectively. Appendix D presents two implementation architectures of FFT exploited in Chapter 4.

Chapter 1

Background and motivation

1.1 Energy Efficiency

1.1.1 Motivation

Wireless communication plays an increasingly important role in modern world and in people's social lives. More and more people all over the world use mobile devices as the most important tool to communicate with each other. It has been estimated by International Telecommunication Union (ITU) that there are more than 7 billion mobile cellular subscriptions by end 2015. This is almost equal to 95.5% of the world population [26]. This brings new challenges to the wireless communication system. It requires the radio system to be upgraded and updated easily to provide new Internet services and higher speed with little cost. Recently, energy-efficient system design has received much attention in Information and Communication Technology (ICT) sector. There are some reasons :

1. Nowadays, mobile phones are used not only to communicate with friends by voice and message, people tend to create content, share interesting things in their lives, upload and download content using social media and other Internet-based applications [27]. They prefer to access the Internet by their mobile phone rather than by computer. The subscribers of mobile Internet increase rapidly. Further more, in the information society, various mobile Internet services emerge. The services have been shifted from mobile voice to mobile Internet data transmission, resulting in the explosive mobile data traffic growth.

2. The increasing mobile data traffic leads to the rise of energy costs and consequently a significant growth of carbon emission. 3% of the world-wide energy is consumed by the ICT infrastructure accounting for 2% of the global CO_2 emissions, which is comparable to the world-wide CO_2 emissions by aviation or one quarter of the globalCO_2 emissions by cars, resulting in a global CO_2 equivalent emission of 1.3% [28].

3. The continuously rising energy consumption of mobile communication results in steadily increasing energy costs and the operating expenditure (OPEX). From an operator's perspective, there is a strong economic driver to reduce the energy consumption thus reducing the cost and keeping a desired service level.

4. Mobile phone users frequently use a wide variety of mobile Internet services, such as music, video, game, and TV applications. They suffer "two days of battery life during active use" and think that the battery life in their phones are pretty poor.

5. Governments are also seeking strategies to build a greener industry, both in the perspective of a sustainable long-term development, and a shorter perspective of economic growth [29].

These drivers also motivate the research on energy efficiency in academia, and some international projects on energy efficiency involving academic, industry, and some international non-governmental research organizations have been started in recent years.

It is encouraging to see that a lot of efforts have been made from many perspectives to address the challenges and provide possible ways to save energy. The enabling technologies and challenges on energy-efficient wireless networks have been surveyed by [30, 31, 32, 33]. Moreover, [31, 32] have addressed that most promising solutions on energy-efficiency are the hybrid techniques in multi-user and multi-cell cases. [33] has presented the joint optimization of component, link and network levels that entails tangible fundamental improvements in terms of energy efficiency when compared to the sole optimization of some specific aspects or components. [34] has described novel approaches to reduce the energy consumption of future base stations. [35] has proposed a framework for green radio featuring four fundamental trade-offs. [36, 29] have surveyed the research efforts on energy efficiency for fixed networks. [37] has provided an overview of a network-based model of power consumption in the Internet infrastructure. [38] has analyzed energy consumption in cloud computing. Cognitive Radio (CR) [10, 11, 14] has also been considered as an

enabling technology for green radio communications due to its ability to adapt its behavior to the changing environment [3]. However, it is still lack of efforts to investigate power reduction via efficient management at the equipment level, which is one of the objectives of this work.

1.1.2 Projects

We non-exhaustively list some projects below and simply classify them in 6 classes depending their focuses. First we would like to introduce the projects dealing with the energy efficiency of mobile radio networks, and then followed by the projects focus on data center, wired network devices, optical networks, wireless mobile devices, and finally the projects with a global vision.

Mobile Radio Networks

EARTH [5]

Energy Aware Radio and NeTwork TecHnologies (EARTH) was an European funded Seventh Framework Programme (FP7) project, whose duration was from January 2010 until July 2012. The reader can find the details on the website of the project [5]. We now summarize the important aspects connected to our work. This project was one of the first to be interested in green radio with an ambitious goal to reduce the energy consumption by a factor of 50 % of mobile telecommunications systems. This project had many original ideas, definitions, and many algorithms now recognized and used by many other projects. Among other ideas include the turning on/off base stations according to the number of users, turning on/off the power amplifier based on the periods without transmission, algorithms to increase these periods, cooperative protocols, cell breathing, etc ...

Mobile VCE Green Radio [39]

The reader can find the details on the website of the project [39]. We now summarize the important aspects connected to our work. Mobile VCE Green Radio aims to reduce the power consumption by 100-fold of the wireless communication networks while not compromising the Quality of Service (QoS) as well as the cost of network deployment, by optimizing the network architectures and developing new techniques.

OPERA-Net 2 [40]

The Optimizing Power Efficiency in Mobile Radio Networks 2 (OPERA-Net 2) project, duration from December 2011 until November 2014, aimed to improve the power efficiency by optimizing the network access technique, and to improve the material efficiency and reduce the environmental impact of mobile radio networks by designing low power cooling systems and hybrid power systems. More details can be found on the website of the project [40].

Data Center

Green Grid [41]

Green Grid aims to improve the energy efficiency of data centers by developing metrics. These metrics can be used to measure the productivity of a data center. Based on the metrics, smarter decisions can be made when deploying new data centers. The reader can find the details on the website of the project [41].

FIT4Green [42]

FIT4Green, a 30-month EU project started in January 2010, aimed to reduce the energy consumption of data centers by designing an energy-aware plug-in. Energy savings were achieved by several ways : setting the unused servers to standby mode ; turning off the unused servers ; dynamic migration of virtual machines, etc. More details can be found on the website of the project [42].

Wired Network Devices

ECONET [43]

ECONET (low Energy COnsumption NETworks), duration from October 2010 to September 2013, was an European Commission FP7 co-funded project, whose aim was to reduce the energy consumption of wired network devices by 50% in the short to mid-term, and by 80% in the long run. Energy savings were achieved by standby and performance scaling when a part of a device is unused. The reader can find the details on the website of the project [43].

Optical Networks

CHRON [44]

The Cognitive Heterogeneous Reconfigurable Optical Network (CHRON) project aims to improve the resource efficiency and energy efficiency of the optical networks by taking advantage of cognitive radio technologies. CHRON has proposed a novel architecture to integrate the control and management plane (CMP), the data plane, and the Cognitive Decision System (CDS). More details can be found on the website of the project [44].

Wireless Mobile Devices

C2POWER [6]

Cognitive Radio and Cooperative Strategies for POWER saving in multi-standard wireless devices (C2POWER), duration from January 2010 until December 2012, was a project supported by European FP7. C2POWER aimed to reduce energy consumption of wireless mobile devices, as its name implies, by using the cognitive radio technologies and by the cooperation of wireless mobile devices. The reader can find the details on the website of the project [6].

Global Vision

GreenTouch [4]

This is a very ambitious project led by Alcatel, which aims to decrease the energy consumption of the network by a factor of 1000. This decrease is analyzed segment by segment with different objectives according to the segments regardless of traffic growth.

Among the results of this project, architectures, technologies, components, algorithms have been proposed for the mobile access networks, fixed access networks, and core networks. Two tools has been developed and are publicly available :

•GWATT [45] : A web-based, interactive application that provides a complete view of the technologies of GreenTouch and the energy impact from an end to end viewpoint.

•Flexible Power Model [46] : An power model and software tool that provides power consumption values for cellular base stations, configurations and scenarios.

More details can be found on the website of the project [4].

TREND [47]

TREND (Towards Real Energy Efficient Network Design) is an European funded FP7 project, whose aim is to design energy-efficient networks, by integrating the European research activities and using a holistic approach considering all segments in networking. The Trend-meter tool, available on-line [48], has been developed to monitor and control the power consumption of networking infrastructures. An on-line database of power consumption values, Powerlib, was created to provide and collect power consumption values [49]. The reader can find the details on the website of the project [47].

SCEE team also involved in two projects : Smart pOwer Grid for Energy Efficient small cell Networks (SOGREEN) project funded by French National Research Agency (ANR), and Toward Energy Proportional Network TEPN project.

SOGREEN [7]

Following a multidisciplinary approach, SOGREEN offers intelligent management of energy system based on the integration of wireless networks and smart grid, expecting a considerable improvement in energy efficiency. As shown in Figure 1.1, wireless networks and smart grid are interconnected, so as to optimize overall energy consumption. In this scheme, we can distinguish three different types of flows : data flow of wireless communications networks, electrical flow, and energy control flow. In this project, decision-making algorithms, both at global level and at each subnet level, are studied. HDCRAM (will be introduced section 1.2) has been also proposed as the manager in this project.

TEPN [8]

The Cominlab TEPN (Toward Energy Proportional Network) project aims at adapting the network energy consumption to the actual load of this network, which can be achieved by taking decision-making algorithms (based on various constraints and metrics) into the network, in particular learning algorithms that can learn behaviors of the network, in order to adapt the energy consumption to the users' needs.

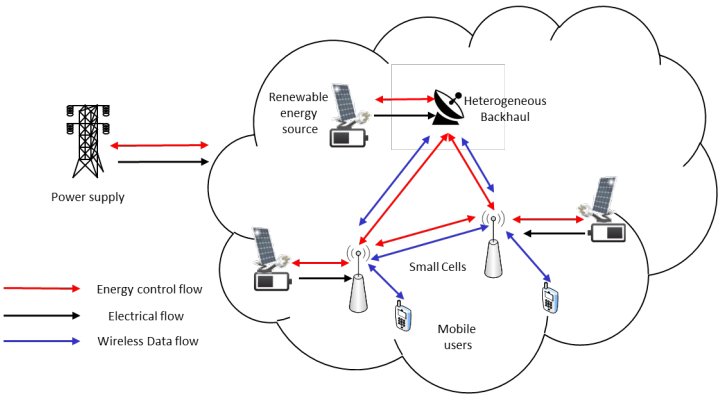


FIGURE 1.1 – SOGREEN.

1.1.3 Comparison of our work with the state of the art

Cognitive radio has been considered as an enabling technology for the green radio communications due to its ability to adapt its behavior to the changing environment [3]. Therefore, in this thesis we take advantage of cognitive radio technologies. Among these projects, only CHRON and C2POWER use CR as a tool to reach green radio. CHRON project mainly focuses on the improvement of energy efficiency of the optical networks, in this thesis, our work is at the electronic level of a hardware device. C2POWER aimed at reducing energy consumption of wireless mobile devices, which is interesting because it is exactly in line with what we call cognitive green radio. The results of this project were considered very positive, which confirms our thesis that this is precisely the cognitive green radio context at electronic level of an equipment. Even C2POWER is also dealing with the energy reduction of hardware equipments, it did not implement the cognitive cycle inside a device. In this thesis, we not only implement cognitive cycles inside hardware devices, but also use a management architecture HDCRAM to efficiently manage the CR features. Moreover, we also introduce and study some useful metrics on hardware device that can be used in the cognitive cycle to make better decisions to efficiently manage the equipment.

1.2 Cognitive Radio

As more and more radio standards are being developed to provide various communication applications, a single radio device is required to support multi-mode multi-band radios. Software radio (SR) [9, 50] has been considered as a solution to provide the flexibility, which has been defined that all of its functionalities can be defined or configured by software. Furthermore, in order to efficiently use the resources of the communication system, the concept of cognitive radio (CR) has been first proposed by Mitola [10], and has soon become a hot research topic. A CR system can adapt its behavior to the changing environment to efficiently use the available resources based on the sensing information from its internal states or its surroundings by dynamic reconfiguring its functionalities. Figure 1.2 explains how a cognitive radio agent interacts with its environment. Such a cognitive radio continually observes the environment, orients itself, creates plans, decides, and then acts [11].

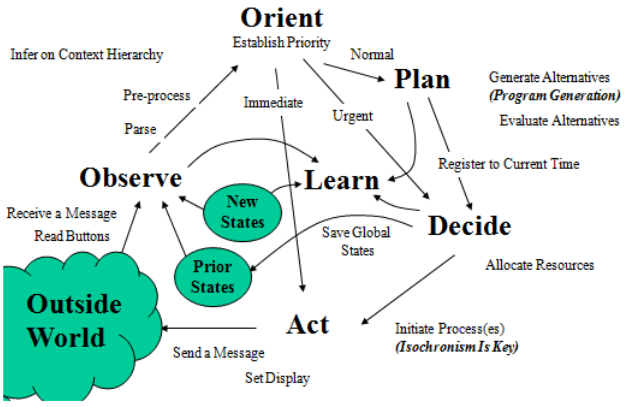


FIGURE 1.2 – The Cognitive cycle proposed by Mitola. [10]

1.2.1 Spectrum Utilization

The radio spectrum is considered as an exclusive property of a country. Therefore, traditionally the use of radio spectrum is nationally regulated by a government agency. And frequency bands are fixedly allocated for different radio services.

Some measurement reports have revealed that most radio frequency spectrum was inefficiently utilized [51, 52, 53, 54]. Some bands are heavily used(e.g., those bands used

by cellular base stations) while many other bands are not in use or are used only part of the time [52]. Federal Communications Commission (FCC) has measured the spectrum occupancy below 1 GHz in Atlanta, New Orleans, and San Diego. Figure 1.3 and 1.4 show the percentage of idle frequencies for two nonadjacent 7 megahertz blocks of spectrum below 1 GHz. The measurements show that some frequencies are heavily or partly used in Figure 1.3, while the frequencies on another band are almost completely idle in Figure 1.4.

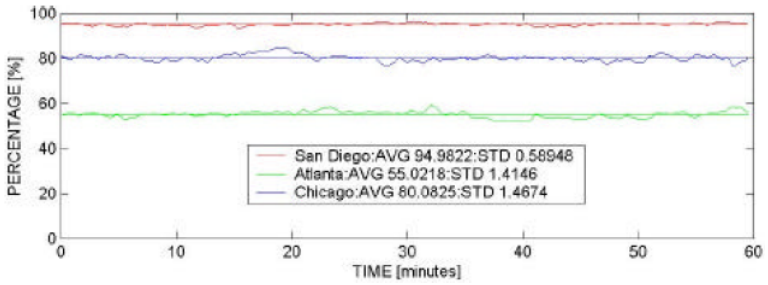


FIGURE 1.3 – Percentage of idle frequencies on a 7 megahertz band below 1 GHz. [51]

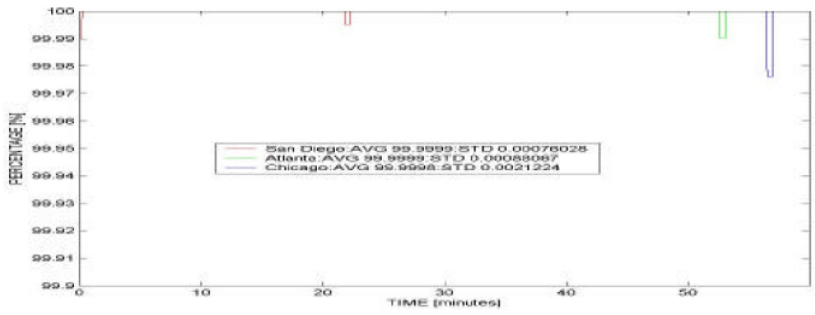


FIGURE 1.4 – Percentage of idle frequencies on another 7 megahertz band below 1 GHz. [51]

In [54], authors have investigated the spectrum usage in two countries, Czech Republic and France, in three regions : 1) northern suburb of Brno, Czech Republic ; 2) eastern suburb of Paris (ESIEE Paris), France ; and 3) city of Paris, near "Place de la Nation", France. The regional spectrum utilization is summarized in Figure 1.5. The overall spectrum utilization in the band 400 MHz - 3 GHz in regions 1, 2 and 3 is 6.5%, 10.7% and 7.7% respectively [54].

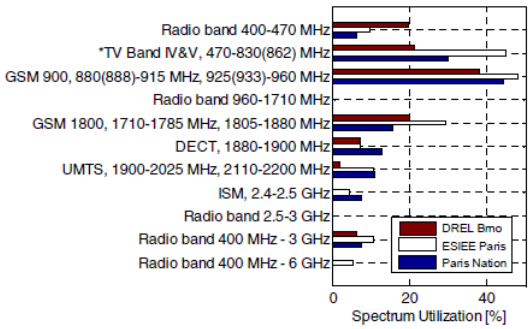


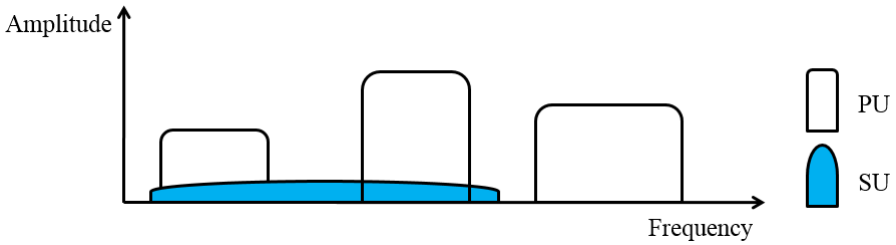
FIGURE 1.5 – Comparative summary on regional spectrum utilization. [54]

These measurements have demonstrated the underutilization of the spectrum. With the increasing demand of wireless application, the insufficiency of spectrum is more and more serious. However, on the other hand, the radio spectrum is considered scarce. Therefore, there exists the opportunities to reuse the unoccupied spectrum by dynamic spectrum access (DSA) [55]. This has become one of the most hot research domain on CR to increase the efficient use of spectral resources. There are many research works on spectral optimization [56]. That is why CR is often reduced to this vision of Spectrum-Sensing Cognitive Radio, in which only the radio-frequency spectrum is considered [57] [58].

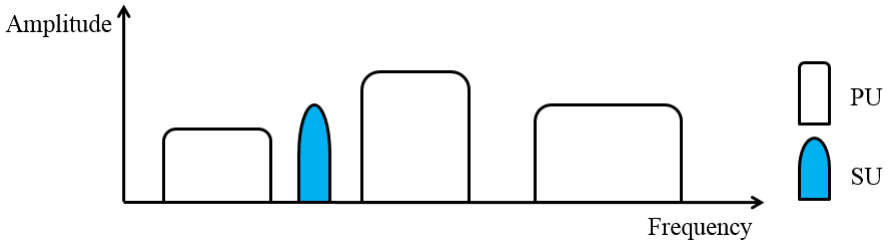
There are different modes of spectrum sharing : interweave, underlay, and overlay. Figure 1.6 illustrates the underlay and overlay dynamic spectrum access.

The underlay mode allows for simultaneous transmission of PUs and SUs. SUs may share the spectrum by transmitting at the same time as the PUs but at very low power to ensure that the interference noise level at the PUs side does not exceed a predefined limit even if the PUs are idle. the underlay mode is represented in Figure 1.6a.

The overlay mode, similarly to the underlay mode, also allows for co-existence of SUs and Pus in the same band. SUs detect the white spaces / holes in the spectrum and then insert their own transmissions on the white space / holes as shown in Figure 1.6b, making sure not to interfere with other PUs. Therefore, SUs share the spectrum with PUs, but the PUs have priority, and SUs can transmit without power limit.



(a) underlay



(b) overlay

FIGURE 1.6 – Dynamic spectrum access modes.

1.2.2 General Vision

There is a more general vision that known as full Cognitive Radio (Mitola radio), in which every possible parameter observable by a wireless node (or network) is considered [10]. SCEE team has proposed a multi-layered model of CR, which is presented as [12] :

- A high-level layer, which contains the application layer and man/machine interfaces. Some kinds of sensors are specific for this layer, such as sound, image, velocity, and position etc. These sensors could be used in context aware communications [59].

- An intermediate layer, which contains transport layer and network layer.

- A low-level layer, which contains medium access control layer and physical layer.

A simplified version of Mitola's cognitive cycle is shown in Figure 1.7, which is composed of three essential parts : sensing, decision, and action.

- Sensing : senses and perceives any kinds of useful information of the environment (surroundings or internal states etc.) ;

- Decision : makes decision based on the observed information with some kind of intelligence (including learning, planning, decision making etc.) ;

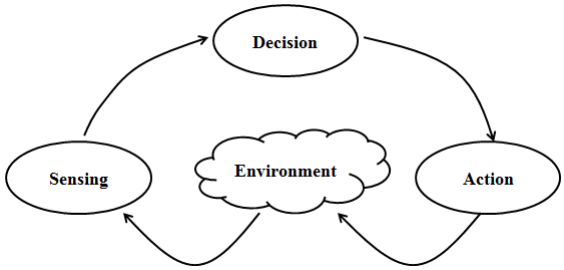


FIGURE 1.7 – The simplified cognitive cycle.

- Action : reconfigures the radio to adapt to the changing environment. It requires a flexible platform, therefore, SR is an ideal tool of reconfiguration for the CR.

These three parts should work coordinately. It is a good idea to add a managament architecture to glue them together and efficiently manage the CR features.

1.2.3 Sensing

Thanks to the five human sensors as shown in Figure 1.8, we can perceive the surrounding environment. Each sense can have a vector of parameters, e.g., human vision could have three parameters : resolution, wavelength, and range.

Similarly, in the CR domain, CR sensors are also needed to gather information about the internal working state and outside environment, e.g., sensor to monitor the battery level, sensor to detect the communication standard, etc. Figure 1.9 gives a multi-dimension representation of CR sensors. Each sensor represents a dimension in the CR domain. Like the human analogy each dimension has also a set of parameters [60]. By analogy with the human sensors, Frequency Hopping (FH)/Direct Sequence (DS) sensor, used by the Standard Recognition sensor, has 3 parameters : time, frequency and power.

Table 1.1 gives a non-exhaustive list of the sensors based on simplified three layers model introduced in the previous subsection. Rather than the classical sensors, the sensors in the list are in a broad sense, taking into account all means that can give information of the environment [60]. These sensors are then used to gather information of the environment, so as to help understand the working situation and make appropriate decisions.

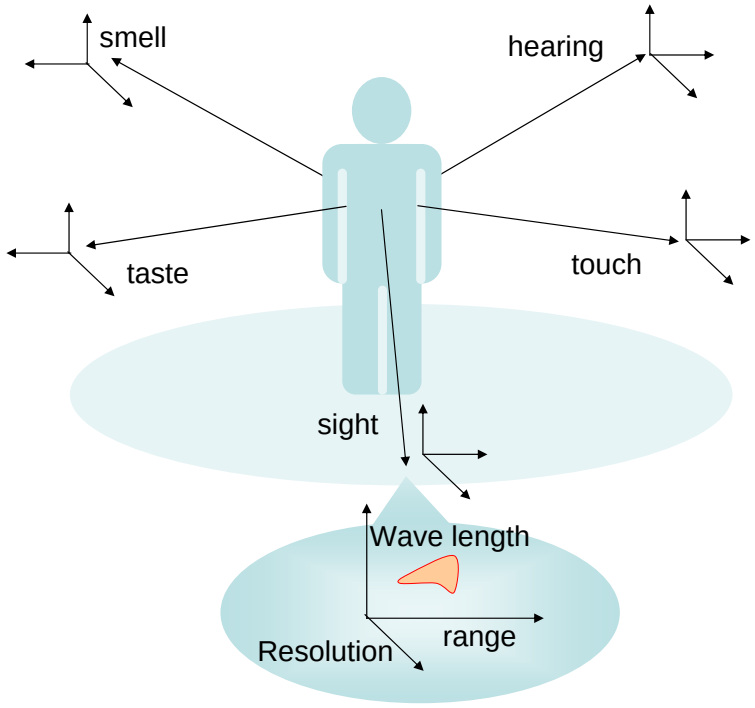


FIGURE 1.8 – Human sensors. [60]

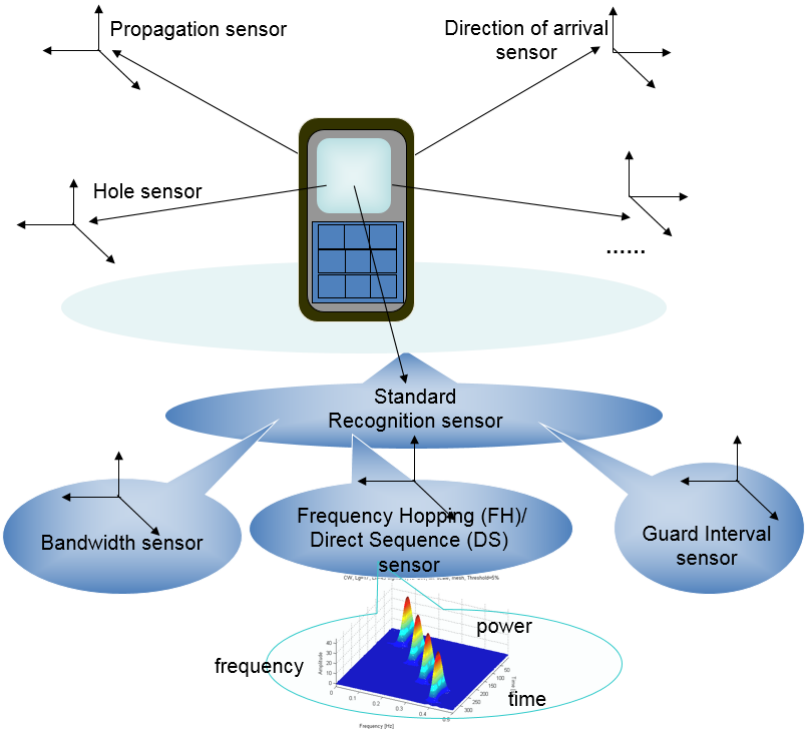


FIGURE 1.9 – Cognitive Radio sensors. [12]

TABLE 1.1 – Classification of sensors based on simplified three layers model. [60]

Sensors	Layers
User profile : Price,, Operator, Personal choices, etc. Sound, Video, Speed, Position, Security, etc.	Application and man/machine interfaces
Vertical handover Inter/intra networks Standards Load on a link, etc.	Transport, Network
Access mode, Power, Modulation, Channel coding, Carrier frequency, Symbol frequency, Horizontal handover, Channel estimation, Antennas beams, Consumption, etc.	Physical, link

SCEE team has also done some research work on sensing, including : cyclostationarity-based test for detection of vacant frequency bands [61], blind standard recognition sensor [62], blind spectrum detection using compressed sensing [63], video sensor [64], energy detection under sensing uncertainty with sensing errors [65].

1.2.4 Decision Making

After getting the information of the environment through the sensors, the CR system should make decisions based on the obtained information.

Decision making approaches have been classified depending on the degree of *a priori* knowledge provided to the cognitive engine, which is depicted in Figure 1.10 [66]. The *a priori* knowledge is defined as a set of assumptions made by the designer on the amount of the available information to the decision making engine when it first deals with the environment [66]. In Figure 1.10, on the left side, a priori knowledge is complete, the expert approach is sufficient ; on the right side, the knowledge is totally unknown, the CR system has to learn the knowledge from the environment.

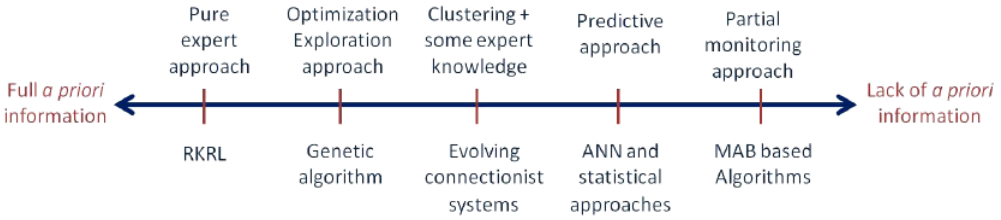


FIGURE 1.10 – Suggested decision making techniques depending on the assumed *a priori* knowledge. [66]

1.2.4.1 Expert approach

The expert approach requires a large amount of expert *a priori* knowledge provided by researchers and engineers. Decision rules are inferred by intensive off-line simulations and then applied on-line to adapt to the environment. These rules are supposed to satisfy all the cases that the CR system will meet. The more *a priori* knowledge acquired, the better the CR system can adapt itself to the environment. If the knowledge is represented as a set of rules, the decision making process becomes very simple. Mitola has represented the knowledge using radio knowledge representation language (RKRL) [11]. The expert approach is sufficient when the situation is well designed, but when the situation evolves, the system's performance might be poor, new rules should be added.

1.2.4.2 Exploration based decision making : Genetic Algorithms

According to the obtained environmental information, Genetic Algorithms (GA) have been proposed in the decision making engine of CR to find the best parameters to meet the users' needs, which is an optimization problem [67, 68]. *A priori* knowledge about the objective functions and the fitness functions is required.

Based on the evolutionary theory of Charles Darwin and genetics, Genetic Algorithms (GA) mimic processes of natural selection and are used to solve optimization problems. A common GA starts with a set of randomly generated solutions, which is called a population. Solutions from one population are selected and used to form a new population for next generation. Solutions are selected according to a fitness function. The higher the fitness, the more chances to be reproduced. This repeats until a suitable condition (e.g., a certain number of generations have passed, or best solution has been found) is satisfied.

GAs are promising to solve CR problems due to their ability to adapt to the changing environment. But GAs have also some limitations. The environment-related analytical models are idealized and not practical. Sometimes GAs run for quite a long time and therefore are not always feasible for real time cases.

1.2.4.3 Learning approaches : exploration and exploitation

In the case when obtaining little environmental information and models, the CR decision making engine has to implement the learning process. Some learning approaches have been proposed : Artificial Neuronal Networks (ANN) [69, 70], Evolving connectionist systems (ECS) [71, 72], statistical learning [73], regression models, etc. All of these learning approaches react with the environment and try to infer the decision making rules. These learning approaches have been classified depending on the way they learn and exploit their rules [74].

1) separate exploration and exploitation phases

ANN and statistical learning are in the class. They have a pure exploration phase in which the CR decision making engine learns from the environment and infers decision making rules. The exploration phase needs a large amount of data and computational power in order to extract reliable knowledge. However if the first phase is well achieved the second phase is usually very simple and does not require much time or energy

2) combine exploration and exploitation phases

ECS based decision making engine can change its structure by learning new knowledge without "forgetting" previously learned knowledge [71, 72]. When there is no a priori knowledge is provided, the CR decision making engine has to try different configurations to estimate the performance, which is belong to the Multi-Armed Bandit (MAB) problem. One solution to this problem is to use Upper Confidence Bound (UCB) [75] algorithms. The main advantage of UCB algorithms is that they offer a balance between exploration and exploitation phases without interrupting the operation.

SCEE team has exploited Upper Confidence Bound (UCB) algorithms based on MAB framework for dynamic configuration adaptation (DCA) [75], dynamic spectrum access [76], and decision making under sensing uncertainty with sensing errors [77].

1.3 HDCRAM Architecture

1.3.1 Introduction

A radio equipment consists of a set of functional components that are connected with each other, illustrated as processing elements (PEs) at the bottom of Fig. 1.11. In traditional radio devices, These PEs are fixed functions that cannot be modified once designed. However, with the evolution of the communication technologies, a radio device is required to support different standards, thus these PEs should be reconfigurable. The design of a PE supporting multiple functionalities normally involves not only software but also hardware toward software / hardware co-design. These PEs can either be software or hardware elements.

According to the cognitive cycle in Figure 1.7, a cognitive equipment should be at least composed of three parts :

- sensors ;
- decision making means (learning) ;
- autonomous adaptation (reconfiguration).

Only if these three parts coordinate together, the cognitive equipment can efficiently work. Therefore, a management architecture is necessary to be added into the cognitive equipment acting as a skeleton to realize the CR capabilities.

Our team has proposed a management architecture for Cognitive Radio in a previous work. This architecture, named HDCRAM, is an abbreviation for Hierarchical and Distributed Cognitive Radio Architecture Management [22, 23, 12]. A diagram of HDCRAM architecture featuring three levels is depicted in Fig. 1.11.

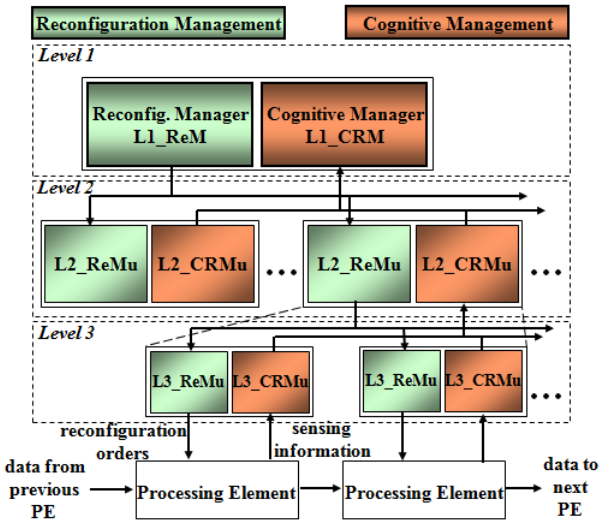


FIGURE 1.11 – A schematic example of HDCRAM architecture.

HDCRAM consists of two aspects : Cognitive Radio Management (CRM) and Reconfiguration Management (ReM) [78]. The CRM part is responsible for gathering sensing information and making decision based on the metrics obtained from PEs. The ReM part is in charge of taking actions to reconfigure the system. Sensing information is submitted to the upper level from the lower level. Once a CRM unit has made a decision, it sends the reconfiguration parameters to its associated ReM unit at the same level. The reconfiguration commands are sent from the upper level to the lower level.

HDCRAM has three hierarchic levels.

- level 1 : a central manager L1_CRM/L1_ReM, which is unique ;
- level 2 : intermediate manager L2_CRMu/L2_ReMu ;
- level 3 : local manager L3_CRMu/L3_ReMu of a PE.

At level 1, only one cognitive radio manager and one reconfiguration manager can exist, because this is the top level. At level 2 and level 3, there are multiple couples of

Cognitive Radio Management units (CRMu) and their associated Reconfiguration Management units (ReMu).

The architecture featuring three levels is sufficient. The level 1 manages the exchange of different standards ; the level 2 manages the reconfiguration of the middle scale functions ; and the level 3 manages the PEs.

According to this hierarchical management, a cognitive cycle can be on three different scales as shown in Figure 1.12 : 1) a local small cycle, in which the sensing, decision making, and reconfiguration action are finished, only includes the PE and its associated level 3 management ; 2) a medium cycle that involves multiple PEs and a level 2 management, the reconfiguration of a PE needs the cooperation with other PEs ; 3) or a large cycle that concerns all the three levels of management. More detailed explanation of HDCRAM can be found in [23] and [12].

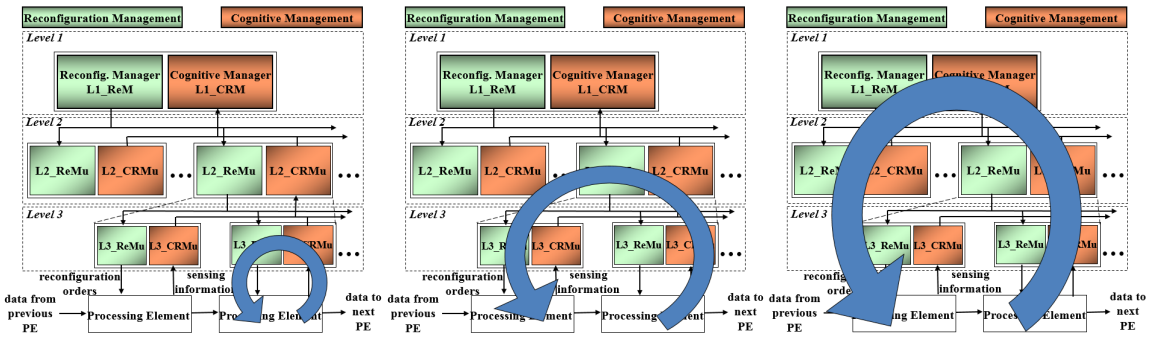


FIGURE 1.12 – Scale of the cognitive cycle : small (left), medium (middle), and large (right).

In order to efficiently manage the sensing information and the reconfiguration of a cognitive equipment, including the HDCRAM management architecture is the necessary price, which could turn a non-intelligent legacy system into a smart system. Any decision making algorithms can be embedded in HDCRAM. However, it does not mean to add the HDCRAM managing all PEs all at once but step by step depending on the real needs to minimize the additional overhead. Depending on what kind of a PE is, the level 3 management of the PE differs as illustrated in Figure 1.13. If the PE is neither reconfigurable nor an sensor, there is no need to have a level 3 management. If the PE is reconfigurable but without sensing information, only a L3_ReMu is necessary. If the PE is an sensor but not reconfigurable, only a L3_CRMu is necessary. If the PE is both

reconfigurable and has sensing information, therefore both L3_CRMu and its associated L3_ReMu are needed.

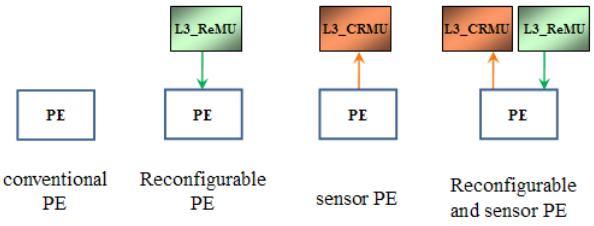


FIGURE 1.13 – Level 3 management depending on the role of the PE.

Although in this thesis, HDCRAM is used for cognitive radio system, it can also be applied to any other complex systems. HDCRAM has also been proposed to manage the smart grid [79].

1.3.2 Heterogeneous Deployment

1.3.2.1 Hardware Platforms

In order to design cognitive equipments, flexible and efficiently reconfigurable hardware platforms are necessary. Many hardware platforms can be used to design cognitive equipments. Those include General Purpose Processors (GPPs), Digital Signal Processors (DSPs), Field Programmable Gate Arrays (FPGAs), Application Specific Integrated Circuits (ASICs), etc.

General Purpose Processors

GPPs are suited for generic applications and normally are not designed for any particular applications (real-time applications etc.). Programs are written in easily understandable high-level programming languages, such as C and C++. Although some modern GPPs have parallel units, the instructions are still mainly executed in a sequential fashion. GPPs are usually running a operating system (OS) thus have a level of abstraction of the hardware. Hence GPPs are very flexible, but with the cost of low performance and high energy consumption.

Digital Signal Processors

DSPs, similar to GPPs, can be programmed with high-level languages, but the architecture of the DSP is specially designed with optimized arithmetic logic for the high speed computations needs of digital signal processing. Therefore, DSPs provide good flexibility with improved performance and low power consumption.

Field-Programmable Gate Arrays

FPGAs are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. Different from GPPs and DSPs, the development of FPGAs uses Hardware Description Languages (HDL), such as Very-high-speed integrated circuit HDL (VHDL) and Verilog. One advantage of the FPGAs is the high degree of parallelism, which provides a high-level computational capacity. Hence their performance is close to that of ASICs. Compared with ASICs, FPGAs are reconfigurable thus have some kind of flexibility at the price of higher power consumption than the ASICs. Traditional FPGAs cannot change the functionality during operation once it has been configured. An FPGA has to stop running and reprogram the entire logic even if a very small part of the logic needs to be updated. Recently, some FPGA families have provided a Dynamic Partial Reconfiguration (DPR) [78, 80, 81] technique. DPR is the ability to dynamically reprogram a subset of the logic within an operating FPGA. This is done thanks to the download of a partial configuration file while the remaining logic continues to operate without interruption [24].

Application Specific Integrated Circuit

An ASIC, as the name indicates, is an integrated circuit customized for a specific application. In contrast to a general purpose circuit, an ASIC is highly optimized for a particular use purpose, therefore has high performance and low power consumption, but at the expense of no flexibility. ASICs are not reconfigurable, if new features are required, the entire ASIC must be redesigned. Therefore, ASIC is not a very good choice in the CR domain.

Figure 1.14 concludes the above-mentioned hardware platforms from the perspectives of performance and flexibility.

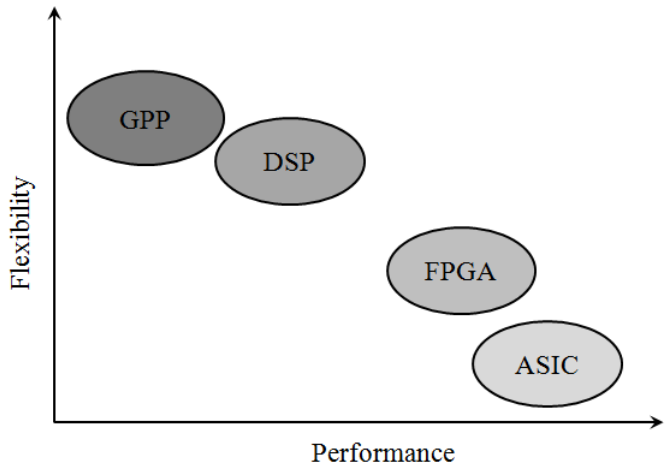


FIGURE 1.14 – Conclusion of several different hardware platforms from the perspectives of performance and flexibility.

We do not try to compare which hardware platform is advantageous over others. Each hardware platform has its own advantages and disadvantages. In fact, a heterogeneous approach to combine different hardware platforms is a better choice. Actually, some vendors have already taken this approach. For example, Xilinx Zynq-7000 All Programmable SoC (AP SoC) integrates a ARM and a FPGA in a single device, thus taking the advantage of the flexibility of the GPP and the performance of the FPGA at the same time.

1.3.2.2 Deployment Example

There may be many different choices to deploy HDCRAM. In this section, we only take one possible HDCRAM deployment method as an example, to introduce the deployment of HDCRAM, as shown in Figure 1.15. It comprises a GPP, a DSP, a FPGA, and a Zynq based device. A straightforward way is placing the level 1 manager on the GPP, and multiple level 2 and level 3 management units on it. A level 2 management unit and multiple level 3 management units are deployed on DSP, FPGA, as well as Zynq. An embedded processing core Microblaze is employed on the FPGA with the level 2 management unit on it. A PE could either be hardware in logic or software on Microblaze. Therefore, a level 3 management unit that is in charge of managing a PE could also be hardware or software, or part of it is software executed on Microblaze and another part is

hardware. The choice is dependent on the given scenarios, and predefined at the beginning design stage of the system. On Zynq, similar to the PFGA, a level 2 management unit is on processing system (PS), and a PE could also either be hardware on programmable logic (PL) or software on PS. A level 3 management unit could also be hardware or software, or part of it is software executed on PS and another part is hardware on PL.

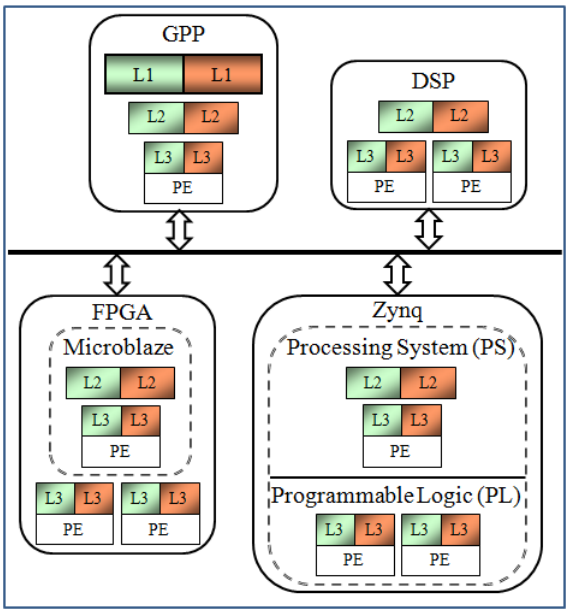


FIGURE 1.15 – A schematic example of HDCRAM architecture.

1.3.3 Software Radio Engines

1.3.3.1 GNU Radio

GNU Radio [82] is a well known and widely used free software development toolkit for the design of software defined radio. GNU Radio Companion provides a graphical user interface (GUI) to make it easy to use in a drag-and-drop way. One shortcoming of GNU Radio is that it needs to stop running and recompile the application even only a parameter is reconfigured. It does not well support the hardware development.

1.3.3.2 RFNoC

Thereforer, RFNoc [83] has been developed to support the FPGA development. RFNoc can be integrated into GNU radio. The blocks of RFNoC can be used in the same way as those in GNU Radio, but the data of these processing blocks are offloaded to FPGA. The RFNoC has the same problem that sometimes it requires to pause the application when reconfiguring a parameter.

1.3.3.3 IRIS

Iris [84] is a software architecture for development of cognitive radio systems. Iris has defined three levels of reconfiguration : the reconfiguration of a parameter of a component ; or structural reconfiguration, e.g., changing components ; and reconfiguration of the entire application. Although it supports runtime reconfiguration but it is mainly at the software level. It is using high level synthesis written in C++ and does not go deep into hardware.

All these three engines do not support the runtime dynamic partial reconfiguration of the hardware. Our HDCRAM approach not only supports the reconfiguration of parameters, adding and deleting components, and the reconfiguration of the entire application, but also supports the runtime dynamic partial reconfiguration of FPGA. This is why we implement the HDCRAM on hardware platforms, which will be introduced in the next chapter. Table 1.2 summarizes these SDR engines.

1.4 Conclusion

With the explosive growth of data traffic in wireless communication, ICT industry is facing more and more serious challenge of increasing energy consumption. Energy efficiency has drawn increasing attention. In section 1.1, the motivations of the research on energy efficiency have been discussed. And we introduce a non-exhaustive list of relevant projects. Compared with all these projects, only a few of them use CR as a tool reach green radio, and only our approach implement the cognitive cycle in hardware equipments.

TABLE 1.2 – Software Radio Engines.

Engines	**Development language**	**FPGA support**	**Runtime reconfiguration**	**Dynamic partial reconfiguration**
GNU Radio	C++ & Python	Not supported	Parameter	Not Supported
GNU Radio + RFNoC	C++ & Python & Verilog	Supported	Parameter	Not Supported
Iris	C++	Supported	Parameter & component & application	Not Supported
HDCRAM	**C++ & VHDL**	**Supported**	**Parameter & component & application**	**Supported**

As an enabling technology for green radio communications, cognitive radio has been introduced in section 1.2. CR is often reduced to the vision of spectrum-sensing cognitive radio, in this thesis, we treat CR in a general vision that known as full cognitive radio.

In order to efficiently manage the CR features, a management architecture HDCRAM, has been presented in section 1.3 to be integrated into CR equipment, to glue sensing, decision, and action together to efficiently manage the CR features. HDCRAM supports heterogeneous hardware platforms working together to take advantages of the merits of different platforms. As described in section 1.3.3, HDCRAM is well adapted for runtime dynamic reconfiguration of both software and hardware.

Chapter 2

HDCRAM on FPGA Platform

2.1 Introduction

As discussed in section 1.3, a management architecture is necessary to efficiently manage the CR features and functionalities. Taking into account the capability of dynamic partial reconfiguration of FPGA equipments, in this chapter, we introduce the implementation of HDCRAM on two FPGA platforms.

2.2 Partial Reconfiguration on FPGA Platform

FPGA devices have provided the flexibility to do on-site device reprogramming, but a drawback of traditional FPGA is that it has to stop running and reprogram the entire logic even if a very small part of the logic needs to be updated. Recently, some FPGA families have provided a Dynamic Partial Reconfiguration (DPR) [24] technique, which extends the inherent flexibility of the traditional FPGA. DPR allows designers to change the functionality of specific regions in an operating FPGA by dynamically downloading a partial configuration bitstream while the remaining logic continues to operate without interruption.

Our SCEE team has worked on DPR since the work of [78], and then developed a more efficient DPR controller than Xilinx provided one and applied to a Network on Chip (NoC) structure [80]. DPR has also been applied to a video application [81].

A Partial Reconfiguration system on a Xilinx Virtex FPGA is mainly implemented by using the Internal Configuration Access Port (ICAP). The ICAP reads a partial bitstream from a nonvolatile memory or form a memory cache (e.g., Block RAM, SRAM), and then reconfigures the specific portion of the FPGA. On Xilinx Zynq-7000 platform, DPR can be implemented by ICAP, or through the processor configuration access port (PCAP). We have learned the Partial Reconfiguration design Flow as well. A partially reconfigurable FPGA design project is more complex than an average FPGA design project.

The logic in the FPGA design is divided into two different types, reconfigurable logic and static logic. Reconfigurable logic is any logical element that is part of a reconfigurable region. These logical elements are modified when a partial bitstream is loaded. Static logic is any logical element that is not part of a reconfigurable region. These logical elements are never partially reconfigured and always active when a partial bitstream is loaded [24].

As shown in Figure 2.1, the block portion labeled Reconfigurable Region represents reconfigurable logic and the light gray area of the FPGA block represents static logic. The function implemented in Reconfigurable Region is modified by downloading one of several available partial BIN files, PR1.bin, PR2.bin, PRn.bin, etc.

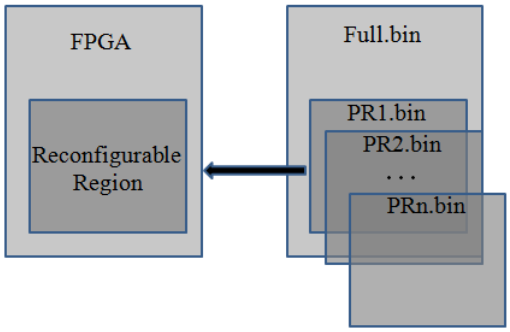


FIGURE 2.1 – Reconfigurable logic and static logic.

There are many reasons why the DPR is advantageous over traditional full configuration.

✓Flexibility. The functionality of part of the FPGA can be updated at any time by locally or remotely loading the partial bitstream that is needed on the fly, which makes the hardware software-like.

✓Reduce reconfiguration time. Because a partial bitstream is smaller than the full bitstream, and the configuration time is proportional to the size of the bitstream, the reconfiguration time of DPR is shorter. Especially when the partial bitstream is quite small, compared with the reconfiguration of the entire device, DPR can significantly reduce the reconfiguration time, which is quite useful to applications requiring strict timing constraints.

✓Improve performance. Only a portion of the device is reconfigured, the static logic remains functioning and is completely unaffected by the loading of a partial BIN file. There is no need to stop running and reprogram the entire device, therefore, it does not affect the performance of the rest of the device.

✓Hardware sharing. DPR can realize the hardware reuse, which enables different functionalities to be implemented in the same portion of the device.

✓Save space and resources. By taking advantage of the DPR, the same system can be implemented in smaller devices featuring less resource thus reducing the size of the FPGA.

2.3 HDCRAM Implementation

2.3.1 Virtex 5 Platform

The management architecture comprises one PC and one FPGA, as shown in Figure 2.3. The level 1 HDCRAM is unique and implemented on a PC. Therefore, on the FPGA side, the highest level is level 2. The Xilinx ML506 board (the brief introduction can be found in Appendix B) is connected to PC by an Ethernet cable. The communication among different platforms of HDCRAM follows the User Datagram Protocol (UDP), which makes the communication easier and flexible. By this method, different devices do not have to be placed together very near to each other. They are connected with each other via Ethernet only requiring their IP addresses. It makes the system scalable so that we can add new devices easily, and need not change those devices that have already existed. Various components are necessary, and all these components work together, enabling the implementation of reconfiguration management on FPGA platform.

The following explains how the different components work together that enables the reconfiguration management functionality in Figure 2.2.

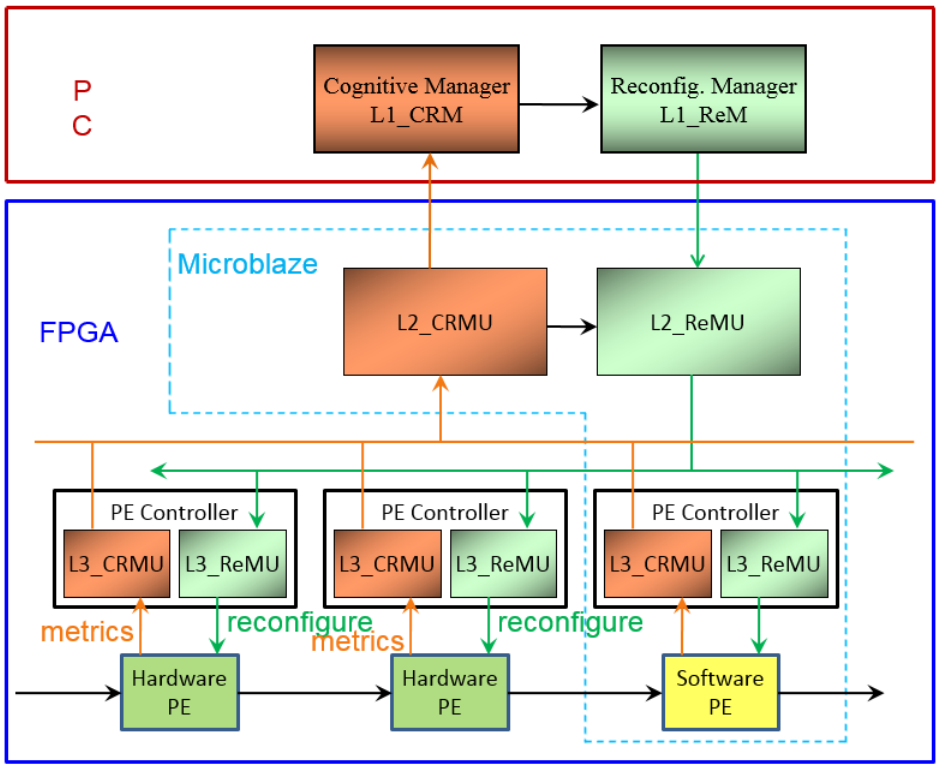


FIGURE 2.2 – An example of management functionality.

A. Hardware UDP Core

Based on the Embedded Hard Tri-Mode Ethernet MAC [85] provided by Xilinx, we have developed a hardware UDP CORE, which works at 1Gbits/s and thus provides a high speed transmission of data and partial bitstreams [86]. In addition to UDP protocol, it also supports Address Resolution Protocol (ARP). The reason of including the ARP protocol is that it allows FPGA to change its IP address thus to dynamically build up communication with different devices. When receiving a packet, the UDP CORE extracts the effective data from the incoming packet by trimming the headers, and then sends the effective data to the corresponding component. On the contrary, when transmitting data,

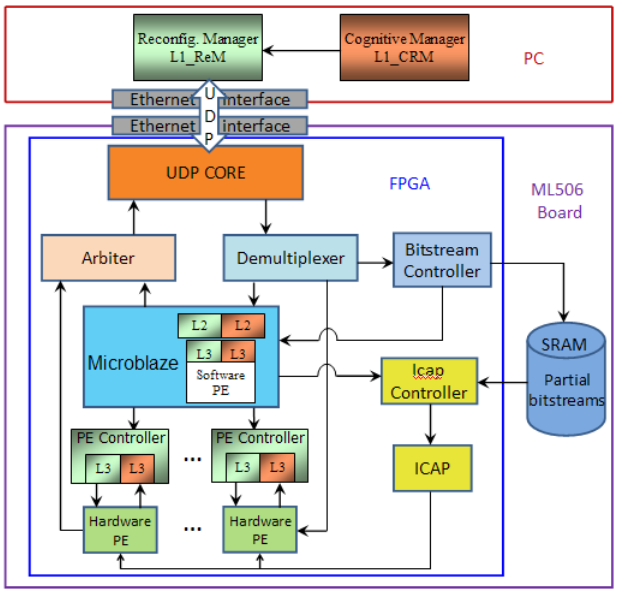


FIGURE 2.3 – The block diagram of the management platform.

the UDP CORE adds the headers in front of the data before sending a frame. The details of the hardware UDP CORE can be found in Appendix A.

The interface of the UDP core is as below, which mainly has 3 parts : receive part, transmit part, and the connection to Embedded Hard Tri-Mode Ethernet MAC.

```
UDP_core_inst: UDP_core
    port map (
                        -- UDP Layer signals
                        --rx
                        udp_rx_start                =>
                            rx_usr_data_start,  --user data start
                        data_rx_out             => data_rx_out,
                        data_len_rx             => data_len_rx,
                        src_port_rx             => src_port_rx,
                        dst_port_rx             => dst_port_rx,
                        src_ip_rx               => src_ip_rx,
                        --tx
```

```
            tx_start                    =>
                tx_start,
            data_input_bus      => tx_data_bus,
            data_length         => data_len_tx,
            src_port            => dst_port_rx,
            dst_port            => dest_port_tx,
            dst_ip_addr         => dst_ip_addr,
            tx_data_out_ready   =>
                tx_usr_data_start,          --
                start user data
            -- system signals
            clk_emac                    =>
                emac_clk,
            reset_emac                  =>
                emac_reset,
            our_ip_address          =>
                local_ip_address,
            our_mac_address         =>
                local_mac_address,
    -- Clock Signals - EMAC0
    -- SGMII Interface - EMAC0
            TXP_0                       => TXP_0,
            TXN_0                       => TXN_0,
    RXP_0                       => RXP_0,
    RXN_0                       => RXN_0,
    PHYAD_0                     => PHYAD_0,
    -- unused transceiver
    TXN_1_UNUSED                => TXN_1_UNUSED,
    TXP_1_UNUSED                => TXP_1_UNUSED,
    RXN_1_UNUSED                => RXN_1_UNUSED,
    RXP_1_UNUSED                => RXP_1_UNUSED,
```

```
        -- SGMII RocketIO Reference Clock buffer inputs
        MGTCLK_P                    => MGTCLK_P,
        MGTCLK_N                    => MGTCLK_N,
        -- Asynchronous Reset
        RESET                       => RESET,
                PHY_RESET                       => PHY_RESET
    );
```

B. Demultiplexer and Arbiter

There are several different types of data, which should be correctly sent to the corresponding components. Figure 2.4 shows the different data paths. Depending on the destination port of the incoming UDP packet, we classify the incoming data into three kinds : command, processing data, and partial bitstream. But the UDP core has only one receiver, so a demultiplexer is necessary to switch the data path depending on the incoming data type. If the incoming packet is a command, it should be sent to level 2 ReMU implemented in Microblaze ; if the incoming packet is processing data, it should be transmitted to processing elements (PEs) ; if the incoming package is a partial bitstream, it should be stored in SRAM. Likewise, an arbiter decides what kind of data should be sent to transmitter when the FPGA sends data to PC.

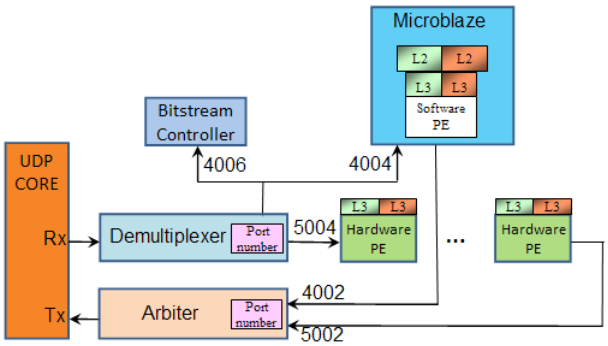


FIGURE 2.4 – Demultiplexer and Arbiter.

C. Microblaze

A level 2 management is implemented in Microblaze, which is a soft processor core embedded in FPGA. The level 2 management controls the level 3 management units, both software and hardware management units. Multiple software CRMUs and ReMUs and software PEs can be created in Microblaze. They are software coded with C or C++ language, for instance a function in a class.

D. Hardware PE Controller

It is easy to implement such software management units and PEs. Generally, software is flexible, but hardware has good performance. Therefore, we expect to have the hardware PE that is as flexible as software and at the same time keeps its performance. With this aim in mind, we have developed hardware PE controller (namely hardware level 3 CRMU and ReMU), which is connected to Microblaze, as shown in Figure 2.5.

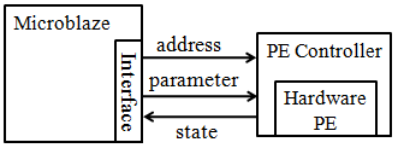


FIGURE 2.5 – Hardware PE controller.

The interface between PE controller and Microblaze has several signals such as address, input, and output. In this way, we can have enough parameters only using these signals. As shown in Table 2.1, address 0 corresponds to parameter 1 or state 1 ; address 4 corresponds to parameter 2 or state 2, and so on. Depending on the different values of the address signal, we can easily change the parameters of the hardware PE or read the states of the hardware PE. For example, when L2_ReMU sends a command to reconfigure a parameter of the hardware PE, Microblaze first writes the address value corresponding to this parameter into the address signal, then writes the new value of the parameter into the input signal of the PE controller (L3_ReMU). And when L2_CRMU reads a state value of the hardware PE, Microblaze first writes the address value corresponding to this parameter into the address signal, then reads the state value of the hardware PE from the output signal of the PE controller (L3_CRMU).

TABLE 2.1 – Relations between address, parameter, and state.

Address	Parameter	State
0	P_1	S_1
4	P_2	S_1
8	P_3	S_3
...	...	...
4(x-1)	P_x	S_x

There is one address signal, and in order to avoid conflict, only Microblaze can write values into this address signal. The hardware PE controller is not allowed to write values to the address signal, but it can read the value of the address signal. When the L2_CRMU wants to read the metrics of the hardware PE, Microblaze writes the address value corresponding to the metric into the address signal. Then PE controller writes the value of the metric (L3_CRMU) into the output signal, and waits Microblaze to read it.

The hardware PE supports Dynamic Partial Reconfiguration. When it needs only to change the general parameters, it uses the method discussed above. When it needs to change the overall functionality of the hardware PE, it is better to choose the Dynamic Partial Reconfiguration approach. Besides, it is also possible to delete the hardware PE by downloading its corresponding black partial bitstream. The DPR feature makes our platform more flexible.

E. Bitstream Controller

When the incoming data is a partial bitstream, the Demultiplexer switches the datapath to the Bitstream Controller. The Bitstream Controller reads the partial bitstream from UDP CORE and writes it into SRAM. Meanwhile, the Bitstream Controller sends the length of the incoming partial bitstream to Microblaze. Because Microblaze is more flexible than hardware logic, we choose Microblaze to manage the base address and the length of the partial bitstream. The base address of the first partial bitstream begins with 0, which is the beginning address of SRAM. In this way, we can calculate the base address of the next partial bitstream if we know the length of the current partial bitstream. In order to efficiently manage the partial bitstreams, Microblaze makes the base address and

the length of a partial bitstream as a pair so that it can find the corresponding partial bitstream correctly when performing a partial reconfiguration.

F. Icap Controller and ICAP

Internal Configuration Access Port (ICAP) [24] is responsible for reconfiguring the specific portion of the FPGA. The Icap Controller is connected to Microblaze. All reconfigurable PEs share the Icap Controller and ICAP. When a hardware PE needs to perform a partial reconfiguration, Microblaze sends its corresponding base address and the length of the partial bitstream to the Icap Controller, then, according to the base address and the length, the Icap Controller reads the partial bitstream from the SRAM and sends it to ICAP. Finally ICAP reconfigures the region of this hardware PE.

G. SRAM

Partial bitstreams are downloaded and stored in a 1MB SRAM so that it can reduce the reconfiguration time, because we can reuse the partial bitstreams many times after downloading and storing them in SRAM thus do not need to download them every time. The functionality of SRAM is similar to a local software library.

2.3.1.1 Data transfer between UDP core and Microblaze

The IP address is connected to Microblaze by General Purpose Input/Output (GPIO), so that we can change the IP address by software.

UDPrx :

When the data path is switched to Microblaze, the incoming data are cached into a FIFO, and the connection between FIFO and Microblaze is via GPIO, the interface is shown in the following codes. The data transmission is controlled by rd_data_start, rd_clk, rd_en, and rd_len. When a packet is sending to Microblaze, the signal rd_data_start activates an interrupt of Microblaze, then the Microblaze controls signals mentioned above to read the incoming data from the FIFO.

```
udprx_command_inst : udprx_command
port map
```

```
        (
--UDP_core interface
                clk_emac     => emac_clk,
                reset_emac   => emac_reset,
                udp_rx_start => rx_usr_data_start,
                data_rx_out  => data_rx_out,
                data_len_rx  => data_len_rx,
                dst_port_rx  => dst_port_rx,
                src_ip_rx    => src_ip_rx,

--Microblaze interface
                rd_start => rd_data_start_GPIO_IO_I_pin,
                rd_clock => rd_clk_GPIO_IO_O_pin,
                rd_en    => rd_en_GPIO_IO_O_pin,
                rd_data  => rd_data_GPIO_IO_I_pin,
                rd_len   => rd_len_GPIO_IO_I_pin
        );
```

UDPtx :

Although we can use the same way as receiving method (namely via GPIO and then FIFO) to send data from Microblaze to UDP core, we would like to find a better way, which makes it easier for Microblaze to write data and has a higher speed. Because the data width in UDP core is 8 bits and the Microblaze is slower than the hardware, we would like to write 32-bit data each time by Microblaze and split the 32-bit data into 4 bytes in the hardware to increase the speed. Therefore, the dual port block RAM (BRAM) is a good choice.

Figure 2.6 explains how the BRAM connects to the Microblaze. The BRAM (bram_block_0 in Figure 2.6) has 2 ports : PORTA and PORTB. PORTA is connected to the BRAM controller, and the BRAM controller is connected to the Data-side Local Memory Bus

(DLMB) of the Microblaze, so that the Microblaze can directly write the data into the BRAM. PORTB is connected to the hardware logic by "Make External" as shown in Figure 2.7.

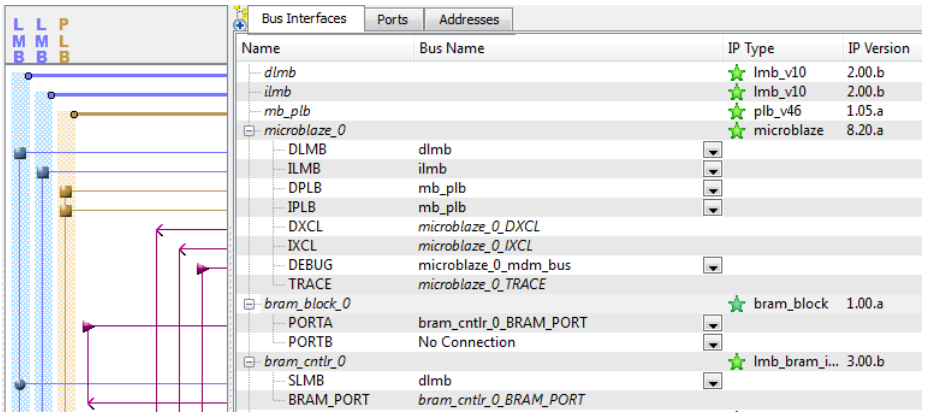


FIGURE 2.6 – Connection between Block RAM and Microblaze.

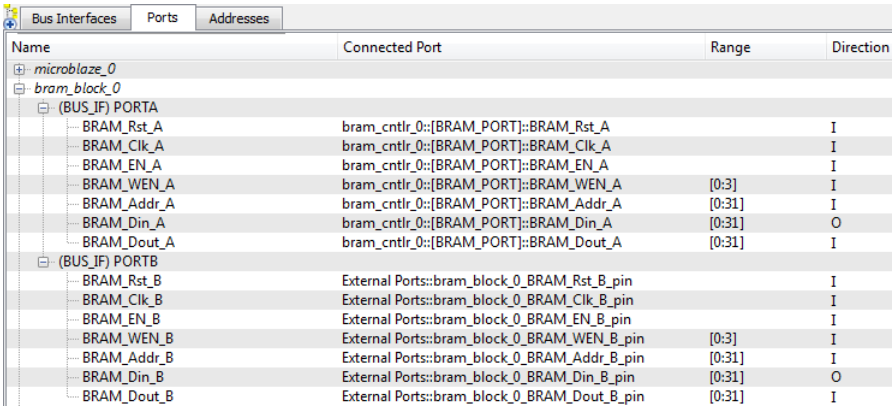


FIGURE 2.7 – Connection between Block RAM and hardware logic.

The interface between UDP core and Microblaze is shown in the following codes. The Microblaze writes the 32-bit sending data into BRAM, and then the bram_udptx block reads the data from BRAM and converts the 32-bit data to 8-bit data then sends them to the UDP core.

```
bram: bram_udptx
port map
```

```
(
 Udp_tx_Data => tx_data_bus,
 clkin       =>      emac_clk,
 rst         => emac_reset,
 udp_valid   =>      tx_usr_data_start,
 BRAM_Rst_B  => bram_block_0_BRAM_Rst_B,
 BRAM_Clk_B  => bram_block_0_BRAM_Clk_B,
 BRAM_EN_B   => bram_block_0_BRAM_EN_B,
 BRAM_WEN_B  => bram_block_0_BRAM_WEN_B,
 BRAM_Addr_B => bram_block_0_BRAM_Addr_B,
 BRAM_Din_B  => bram_block_0_BRAM_Din_B,
 BRAM_Dout_B => bram_block_0_BRAM_Dout_B
);
```

2.3.1.2 The Speed of Downloading FPGA Partial Bitstreams through UDP

As described above, the hardware UDP CORE works at 1Gbits/s, namely the data rate is 125MBytes/s. But, we should find out the actual data rate by taking into account the overhead when transmitting the bitstreams, because each UDP packet has a preamble and several headers.

Ethernet data are encapsulated in frames. Figure 2.8 illustrates the format of a standard Ethernet frame [85]. Because we adopt UDP protocol, our partial bitstreams, as well as IP headers and UDP headers, are inserted into the data field of the Ethernet frames. The length of the data field can vary from 0 to 1500 bytes for a normal frame. The IP header has a length of 20 bytes, while the length of UDP header is 8 bytes. Therefore, in addition to the headers, the maximum length of the effective data is 1500 - 20 - 8 = 1472 bytes in a standard Ethernet frame, the maximum length of which is 1518 + 7 + 1 = 1526 bytes. The overhead of each standard Ethernet frame is 7 + 1 + 6 + 6 + 2 + 20 + 8 + 4 = 54 bytes, no matter whether the frame is with the maximum length or not. If a partial bitstream has a bigger size than 1472 bytes, it requires the reconfiguration

manager to send multiple frames. As a matter of fact, normally the lengths of the partial bitstreams are larger than 1472 bytes.

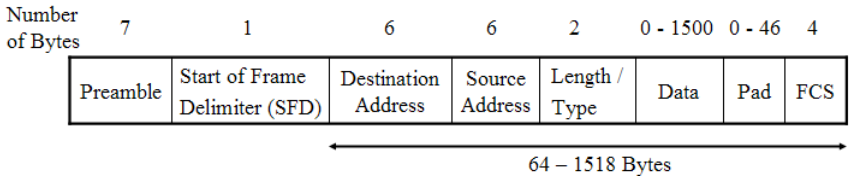


FIGURE 2.8 – Standard Ethernet Frame Format.

Therefore, in order to reduce the total overhead, each time, we should send the frame with the maximum length as much as possible.

The ratio of the length of the partial bitstream to the bytes totally transmitted determines the actual data rate. As presented in subsection II.B, the hardware UDP core works at 125 MHz. Therefore, we can get the actual data rate (Bytes/s) from (2.1).

The numerator n is the length of the partial bitstream, and the denominator is the bytes totally transmitted. The function rem (n, 1472) calculates the remainder of dividing 1472 into n.

We can calculate the limit data rate from (2.1), which is nearly 120.6Mbytes/s.

$$R_{UDP} = \frac{n}{\lfloor \frac{n}{1472} \rfloor \times 1526 + rem(n, 1472) + 54} \times 125 \times 10^6 \tag{2.1}$$

In the following subsection, we would like to compare our method with the fastest partial bitstreams downloading approach through Ethernet that we could find so far.

lwIP is an open source networking stack for embedded systems [87]. Xilinx Embedded Development Kit (EDK) provides the Ethernet MAC IP xps_ll_temac to send and receive packets. It supports lwIP to add networking capability to a Xilinx embedded system.

The approach proposed in [88] can download partial bistreams with a sustained rate of 80 Mbits/s over Ethernet 100 Mbit/s. The xps_ll_temac application on Virtex-5 provided in [87] works at 125 MHz. The maximum throughput of Xilinx Virtex-5 xps_ll_temac in the RAW mode is 100 Mbps, namely 12.5MB/s, without considering the overhead of headers.

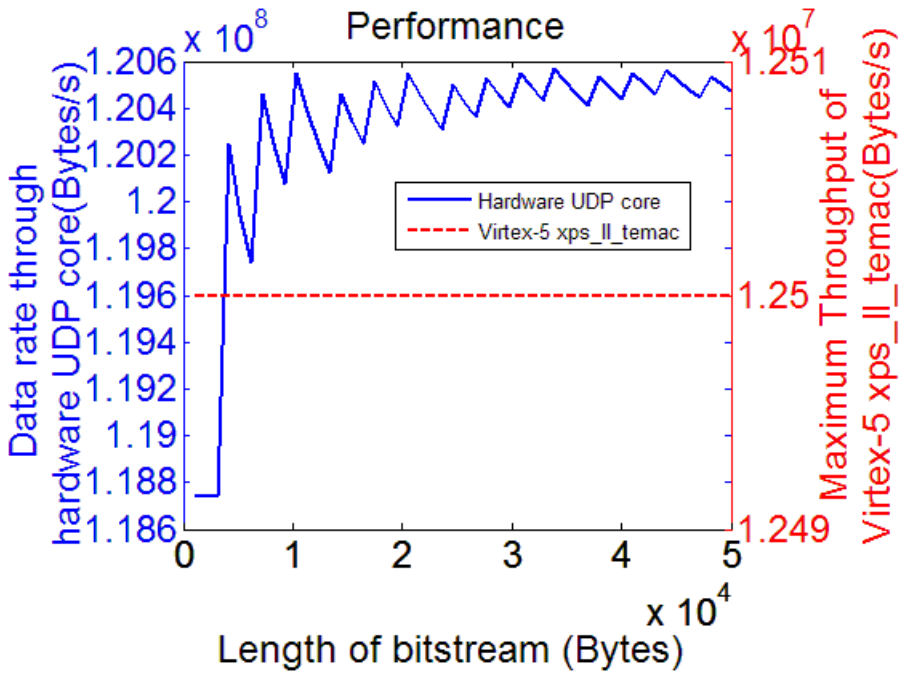


FIGURE 2.9 – The performance of hardware UDP core and Xilinx Virtex-5 xps_ll_temac.

Therefore, we would like to compare our method with the maximum throughput of Xilinx Virtex-5 xps_ll_temac. Even though our method takes into account the overhead, it is much faster than Xilinx Virtex-5 xps_ll_temac.

Figure 2.9 illustrates the comparison of our method with the maximum throughput of Xilinx Ethernet MACs using lwIP. The length of the partial bitstream ranges from 1K bytes to 50K bytes, with the step size of 1K bytes.

We can see that our method is about 10 times faster than the maximum throughput of Xilinx Virtex-5 xps_ll_temac.

2.3.1.3 Discussion on the Reconfiguration time

We hope we can change the functionality of a SDR equipment immediately without any delay, but normally it is impossible in reality.

In a way, the process of downloading a partial bitstream, itself, could be considered as the overhead of a system. We denote t as the time consumed to download a partial bitstream with the length of n bytes. We can calculate t more accurately by (2.2) using

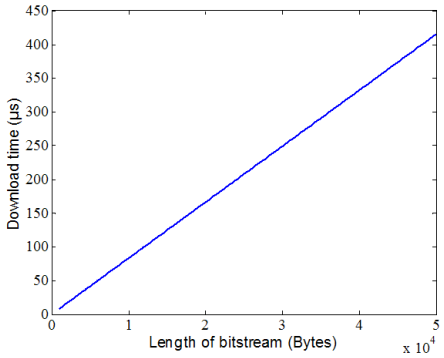


FIGURE 2.10 – The download time vs length of partial bitstream.

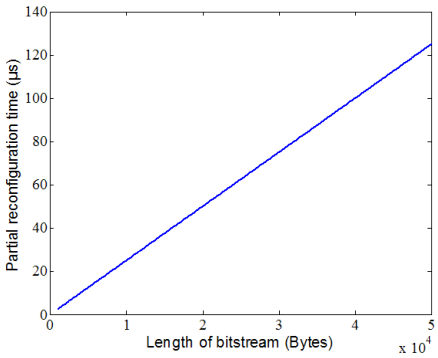


FIGURE 2.11 – The partial reconfiguration time vs length of partial bitstream.

the bytes totally transmitted instead of the length of the partial bitstream by taking into account of the headers.

$$t = \frac{\left\lfloor \frac{n}{1472} \right\rfloor \times 1526 + rem(n, 1472) + 54}{125} \ \mu s \tag{2.2}$$

Similarly, performing a process of partial reconfiguration takes time, therefore, the time consumed by partial reconfiguration could also be considered as the overhead. As discussed in Section III, the throughput of the partial reconfiguration is 400Mbytes/s, therefore, we can calculate the time it requires to perform a partial reconfiguration.

Figure 2.10 and Figure 2.11 illustrate the download time and partial recofiguration time respectively with the length of partial bitstream ranging from 1K bytes to 50K bytes.

Although we can download the partial bitstreams at a high speed, it still consumes more time in contrast to partial reconfiguration. In order to reduce the overhead for a SDR system when changing its functionality, it is a good choice to store the partial bitstreams in a local memory. The memory should be close to the ICAP and can be accessed directly by the Icap Controller. In this way, the reconfiguration time can be reduced by eliminating the download time. For example, if the length of a partial bitstream is 30K bytes, the first time it needs 254.8 + 76.8 = 331.6 μs to change the functionality, but after that it needs only 76.8 μs to do the same thing, because we store the partial bitstream in the local memory and can reuse it many times.

A study of parallel / serial implementation of FIR filter has employed Virtex 5 platform, which is detailed in section 3.4.

2.3.2 Zynq-7000 Platform

Although we have implemented HDCRAM on Xilinx Virtex 5 Platform, there are still some limitations :

- software is standalone application without OS ;
- codes on Microblaze are hardware dependent ;
- hard to migrate ;
- high power consumption, etc.

Therefore, when we have the Xilinx Zynq-7000 platform, we decided to implement HDCRAM on the new platform because of several benefits :

- software is running in Linux on ARM ;
- thus easy to upgrade ;
- portable ;
- low power consumption, etc.

2.3.2.1 HDCRAM implementation on ZC702 Evaluation Board

The ZC702 evaluation board (refer to Appendix C for this board) utilizes a Xilinx Zynq-7000 All Programmable SoC (AP SoC), which integrates a dual-core ARM Cortex-A9 as the processing system (PS) and a Xilinx's 7 series FPGA Artix-7 as the programmable logic (PL) in a single device [89].

On Zynq, there are two ways for DPR to reconfigure the PL, i.e., either by the internal configuration access port (ICAP) primitive on PL, or through the device configuration (DevC) / processor configuration access port (PCAP) interface on PS [90]. ICAP can only perform partial reconfiguration on PL, but PCAP supports both full and partial reconfiguration of the PL from the PS, which provides more flexibilities. Furthermore, the bitstreams are transferred to the PCAP interface by a Direct Memory Access (DMA)

approach, which frees the processor to execute other tasks. Therefore, we utilize the PCAP method.

Different functions can be designed to share the hardware PL by dynamic full and partial reconfiguration in the field. The generated full and partial bitstreams can be stored in a database. Each function has a full bitstream and several partial bitstreams depending on the real needs. Figure 2.12 illustrates the storage organization of the BIN files database.

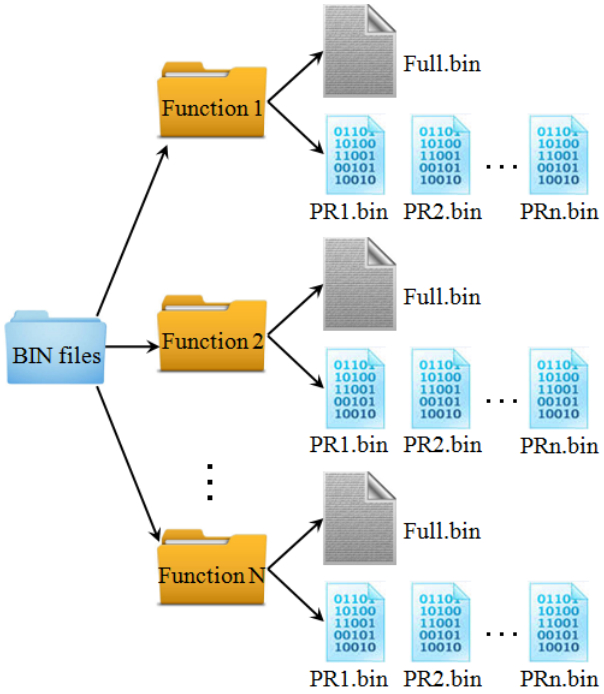


FIGURE 2.12 – The storage organization of the reconfiguration bitstreams.

As shown in Figure 2.13, the main form of connection between the PS and PL elements of Zynq is via AXI (Advanced eXtensible Interface) interfaces, which provide high bandwidth, low latency links between both parts of the device.

We can create a hardware PE as a custom peripheral on PL, and communicate with PS via AXI interface. This is done by "Create and Import Peripheral Wizard" in XPS (Xilinx Platform Studio).

We choose the AXI4-Stream interface, which is designed for the transmission of high-speed streaming data. Connection is from master to slave only, so if bidirectional transfers are required both peripherals must be of type master/slave.

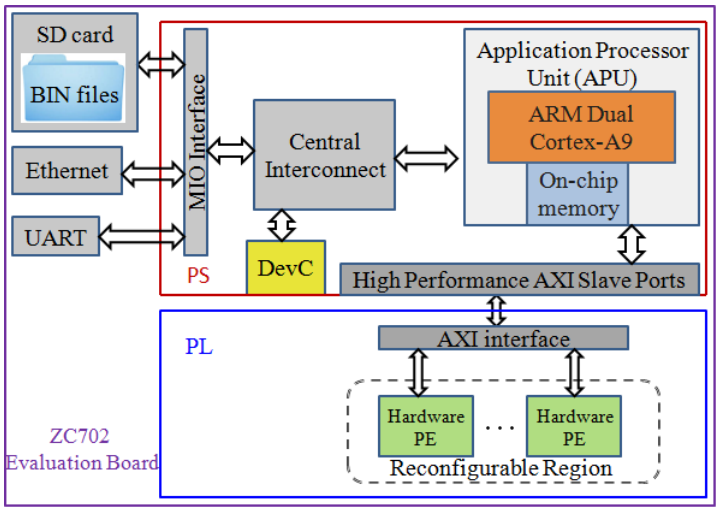


FIGURE 2.13 – A simplified architecture of the ZC702 evaluation board.

But we can not directly connect the AXI4 streaming interface to the AXI interconnect. So we use an AXI DMA Engine to convert AXI4 Streaming to AXI interconnect, which is then connected to an AXI_HP (High Performance) interface. The AXI_HP interfaces provide PL bus masters with high bandwidth data paths to PS memories including the DDR memory and OCM (On-Chip Memory). The interfaces are illustrated in Figure 2.14. The AXI_MM2S (Memory-Mapped to Streaming) and AXI_S2MM (Streaming to Memory-Mapped) are memory-mapped AXI4 buses, which are connected to PS, while the AXIS_MM2S and AXIS_S2MM are AXI4 streaming buses, which are connected to the custom PE. Further information is available in [91].

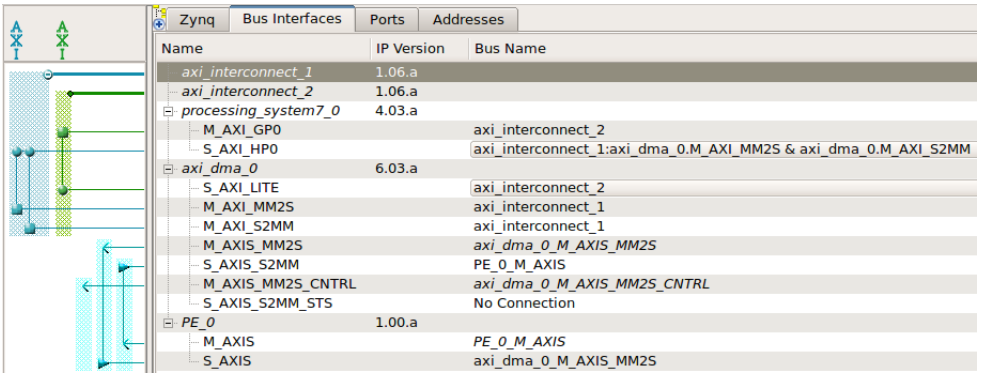


FIGURE 2.14 – The interfaces between PE and PS.

The HDCRAM manages the full and partial reconfiguration. Figure 2.15 illustrates the HDCRAM implementation on the ZC702 evaluation board. The level 1 manager is implemented on the host computer. On the ZC702 evaluation board, a level 2 management unit is implemented on PS. A PE may either be hardware on PL or software on PS. Therefore, a level 3 management unit that is in charge of managing a PE may also be hardware or software, or part of it is software executed on PS and another part is hardware on PL.

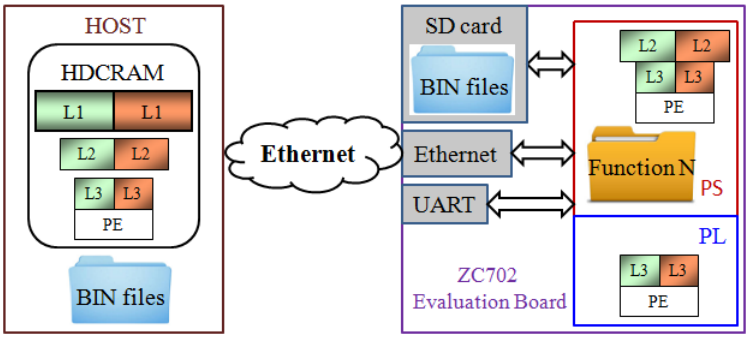


FIGURE 2.15 – The HDCRAM implementation on the ZC702 evaluation board.

There are different ways to store the reconfiguration bitstreams :

* All the reconfiguration bitstreams can be stored in the database on the host computer. The full or partial bitstreams can be remotely downloaded through Ethernet to change the functionality of the complete or pre-defined regions of PL on the fly as needed.
* They can also be stored on the SD card on the ZC702 evaluation board if the level 2 management works standalone. It is also possible to dynamically download new full and partial bitstreams through Ethernet to update the database.
* Some partial bitstreams are able to be read into the on-chip memory on PS if they are frequently used.

2.3.2.2 Case study

A finite impulse response (FIR) filter is a commonly used processing element in digital signal processing. It could be implemented either in software mapped onto the PS or in hardware mapped onto PL. Therefore, we would like to investigate the benefit and cost of

the FIR filter implementation on PS and PL respectively, and then the results will provide helpful information for CRMu to make an appropriate decision.

Evaluation of performance and power consumption of FIR filter implementations

Take a 32-tap FIR filter as an example, which is implemented on PS and on PL respectively. The operations are executed in serial on PS, but on PL, the FIR filter could be implemented in serial or in parallel.

And the hardware serial and parallel implementations of the FIR filter reuse the PL logic by taking advantage of the PR. After generating the full and partial bitstreams for the PL following the PR design flow, we store them in the database on the host as shown in Figure 2.16. A blank full bitstream is also generated to clear the PL to save power if the PL part is not needed, which is stored in NOPL folder. Table 2.2 shows the resource available in the reconfigurable region and used by the FIR filter. The serial implementation consumes less resource, and it uses 2 DSP48E1s, which is 32 times less than the parallel implementation. But the serial way consumes more memory than the parallel approach.

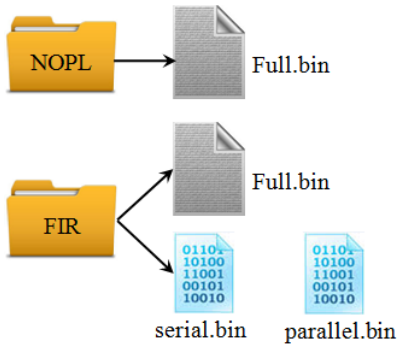


FIGURE 2.16 – The full and partial bitstreams of the design.

The timing overhead of full and partial reconfiguration should also be considered. Because downloading a bitstream remotely from the host computer consumes longer time than that from the local memory, if we can benefit from remote reconfiguration, undoubtedly we can also benefit from local reconfiguration. The sizes of full and partial bitstreams, and the time consumed of remote full and partial configuration are listed in Table 2.3.

TABLE 2.2 – Resources available and used by the FIR filter.

Resource	Available	Serial	Parallel
LUT	10304	868	1096
FD_LD	20608	1516	3108
SLICEL	1564	141	288
SLICEM	1012	91	187
DSP48E1	72	2	64
RAMBFIFO36E1	36	8	4

TABLE 2.3 – Full and partial configuration time.

Type	Size (bytes)	Time (μs)
Full	4 045 564	215 736
Partial	707 712	51 865

We have also measured the power consumption of both PS and PL. The most convenient and simplest way to monitor the power consumption on ZC702 board is to use Texas Instruments' (TI) Fusion Digital Power Designer, which is a Graphical User Interface (GUI) used to monitor and display the real-time voltage and current of selected power rails of the board [89, 92]. Table 2.4 lists the power consumption of PL for blank design and the FIR filter.

TABLE 2.4 – Power consumption of PL.

Function	NOPL	Serial	Parallel
Power(W)	0.06	0.095	0.101

In order to clearly and visibly observe the results, we have sent amount of data to the implemented software and hardware FIR filter. Each time we sent 4096 32-bit integers and then repeat 2000 times. When the hardware approach is chosen, the data are transferred between PS and PL by DMA approach. Table 2.5 gives the total time consumed by software and hardware implementations of the FIR filter.

TABLE 2.5 – Execution time of the FIR filter.

Software (μs)	Hardware (μs)	
	Serial	Parallel
12 229 279	281 315	279 026

We can see that although the hardware approaches consume much less time than the software way, the hardware parallel implementation is not as fast as expected more than 32 times faster than the serial implementation, which is because the overhead of data transmission between PS and PL. It takes some time when the data and commands are transmitted from user space to Linux driver and then to the hardware. Therefore, if only offloading the FIR filter from the PS onto the PL, it is better to choose the serial implementation, which occupies less resource and consumes less power while not losing much performance.

The reason why we repeat 2000 times is that we cannot catch the power changes by TI Fusion Digital Power Designer when the execution time is too short. And even so, sometimes we still cannot catch PR and hardware FIR filter operations. For the sake of comparison and analysis, we put the operations of software FIR filter, PR, and hardware FIR filter together in Figure 2.17. At time 41 :00, the software FIR filter are started execution, at around 41 :25 PR is performed to reconfigure the PL, and at time 41 :36, the hardware FIR filter operations are executed. The power risings at around 41 :25 and at 41 :36 are because the data transmission from PS to PL. We can see that the power increases from 0.33W to 0.44W during software FIR filter operations, which lasts about 12.23s. But the additional power increase of the hardware serial and parallel implementations is around 0.04W on PL, which is less than 0.11W on PS.

Management of FIR filter by HDCRAM

Based on the above results, it is possible to benefit both performance and power consumption by offloading the FIR filter from the PS onto the PL. Another advantage is that it frees the PS to execute other tasks.

Therefore, we choose to implement the level 3 management of the FIR filter on the PS. The L2_CRMu makes the decision to implement the FIR filter on PS or on PL in serial

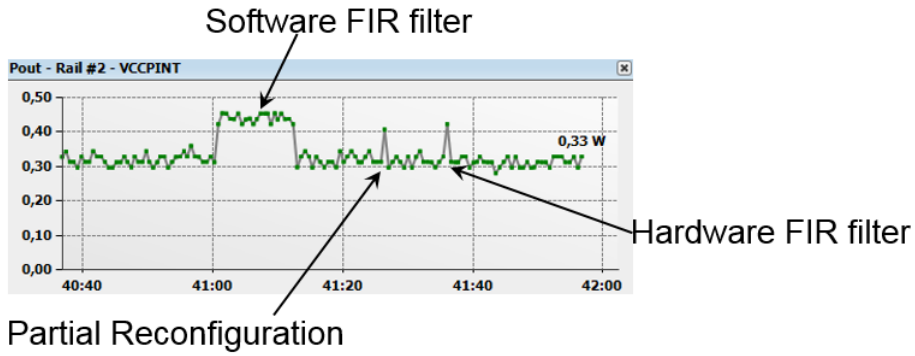


FIGURE 2.17 – Power consumption of PS.

or in parallel based on the information obtained from other L3_CRMus. And then the L2_ReMu sends the corresponding reconfiguration command to the L3_ReMu of the FIR filter, who then maps the FIR filter onto PS by calling the software FIR filter function or onto PL by dynamic full or partial reconfiguration.

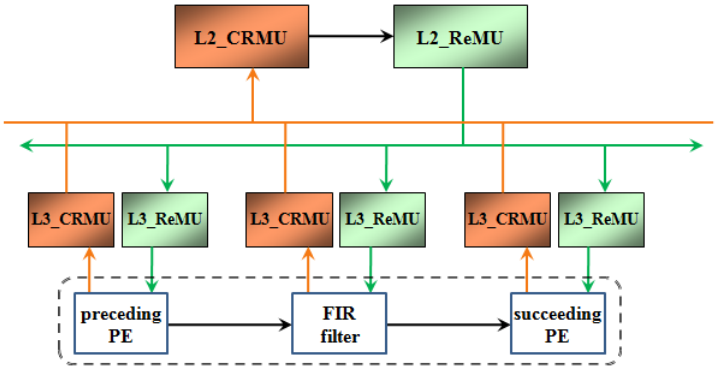


FIGURE 2.18 – Management of FIR filter.

If the PL is occupied by other computation intensive PEs and has no more space for the FIR filter, there is no choice and the L2_CRMu decides to implement the FIR filter in software on PS, which consumes 0.11W more power and has a longer execution time.

Else if the preceding PE and the succeeding PE of the FIR filter is implemented on PS, the L2_CRMu decides to implement the FIR filter on PL in serial mode, because it uses less resource with additional 0.035W power consumption and the performance is close to

the parallel way (see Table 2.5) due to the overhead of data transmission between PS and PL.

Else if the preceding PE or the succeeding PE of the FIR filter is implemented on PL, the L2_CRMu decides to implement the FIR filter on PL in parallel mode, because the speed is more than 32 times faster than the serial way and the data transmission is in hardware, which does not slow down the data processing. This way consumes 0.041W more power but has a higher performance.

2.4 Conclusion

In this chapter, we have briefly introduced partial reconfiguration, and mainly explained how the HDCRAM could be implemented on two FPGA platforms, Virtex 5 and Zynq-7000, what kinds of components are developed and used, and how they work together to achieve the functionality of reconfiguration management. We have studied the commonly used FIR filter and the benefit and cost when it is implemented on PS and PL on Zynq-7000 platform. To process the same amount of data, the software FIR filter needs about 12.23s and consumes 0.11W, the hardware parallel FIR filter needs about 281μs and consumes around 0.041W, and the hardware serial FIR filter needs about 279μs and consumes around 0.035W. The results show that we can win both performance and power consumption by offloading the FIR filter from the PS onto the PL. But it also shows that the hardware parallel implementation is not as faster as expected than the serial implementation because of the overhead of data transmission between PS and PL. The time consumption is not only the process time, but also includes the time for data upload and offload. These information are then provided to the HDCRAM to make appropriate decisions.

Chapter 3

Metrics on FPGA Platform

3.1 Introduction

In this chapter, we mainly introduce some metrics that are useful for the management architecture to efficiently manage the equipment. In order to efficiently use these metrics when some of them are employed in certain scenarios, we discuss these metrics in many aspects, such as self-changeability, configurability, green impact, working level, and susceptibility.

We study the FIR filter as a use case of some of the metrics. The FIR filter is implemented in parallel and in serial respectively, and at the same time, changing the working frequency of the filter. The results show that, although the serial mode uses fewer resources and consumes less power at lower frequencies, in order to keep the same performance, it consumes more power than the parallel mode when it works at frequencies higher than 25.6MHz.

We also estimate the relation between power consumption and the number of taps of the FIR filter. There is a trade off between the power consumption, the performance, and the resources. The system has to make a optimal choice depending on its working environment.

3.2 Useful Metrics on FPGA Platform

For a cognitive equipment, it should sense the surrounding environment and its operating states, and according to the information obtained, make decision and adapt itself to the changing environment by reconfiguring part of or all functionality of the system. As described in previous chapters, designers should carefully select proper and effective metrics, because different scenarios require different kinds of metrics.

These metrics can be used as necessary operating information inside or outside the device for decision making and system reconfiguration (e.g. change functionality).

In the following subsections, we introduce some metrics that can be useful for the cognitive management architecture on a FPGA platform, as well as some measurement approaches of the metrics.

In this chapter, we consider the Xilinx Virtex-5 ML506 board hereinafter as the reference FPGA platform. For other platforms, the methods described in this chapter can be the references. Depending on the platforms, these methods may be used directly, or there are similar methods or alternatives.

3.2.1 Voltage

Voltage is a basic parameter for a system. Normally, for a FPGA platform, there are several power supply voltages for different resources.

For a Xilinx Virtex-5 ML506 board, V_{CCINT} is the primary power supply for the FPGA. It is the internal core supply voltage, which supplies all internal logic functions, such as Configurable Logic Blocks (CLBs), block Random Access Memory (RAM), and DSP blocks [93].

The auxiliary supply voltage V_{CCAUX} powers the auxiliary logic, including the configuration logic, some internal and I/O resources, clock management tiles (CMTs), some dedicated configuration pins, and the Joint Test Action Group (JTAG) interface.

The V_{CCO} powers the I/O resources, and has separate rails for each bank of I/O for maximum flexibility. All of the V_{CCO} connections to a specific I/O bank must be connected to the same voltage.

3.2.1.1 How to Get It

There are several ways to get the value of voltage, we provide here two available methods :

Stored in a Status Register

When a system is developed on a platform, the voltage used by the system is well known. If the voltage is considered as a metric, whose value is usually already known, it is then possible to keep it in a status register.

The voltages of the current platform we used are fixed at certain values, so we cannot change the voltage of a specific region of the FPGA. We hope to be able to have a flexible platform in the future that supports programmable voltage, so that the voltage of a part of the FPGA will be adjustable during operation. This kind of FPGAs will help the development of SDR and CR, and make the implementation of SDR and CR more practical.

Measured by System Monitor

The voltage can also be measured by System Monitor, which is a component provided by Xilinx and located in the center of the die.

The System Monitor function is achieved mainly by a 10-bit, 200-kSPS (kilo samples per second) Analog-to-Digital Converter (ADC), and on-chip voltage and temperature sensors [94]. When they are working together, the System Monitor can provide the on-chip power supply voltages and the die temperature. Furthermore, additional external analog inputs, i.e., a dedicated analog-input pair (VP/VN), and 16 user-programmable analog input pairs ($V_{AUXP}[15:0]$, $V_{AUXN}[15:0]$), are also available to allow the users to access to external signals.

To access the information measured by the System Monitor, there is not only a single way, we have multiple choices.

Use the ChipScope Pro Tool :

The System Monitor offers a useful feature since the measurement information can be accessed via the JTAG TAP at any time thanks to the ChipScope Pro tool, which gives an easy access and a graphical display of the measurement data. The ChipScope Pro

tool also provides the ability to record the measurement data along with the time stamp information in a log file. Thus, further analysis can be done at a later date if needed.

Use an embedded processor :

A limitation of using the ChipScope Pro Tool to get the measurement data is that it needs the help of an additional PC, which is not so flexible. If we want to measure these metrics by the FPGA itself to avoid using a PC, there is an alternative way, which takes advantage of the System Monitor IP. The Xilinx Embedded Development Kit (EDK) provides the System Monitor IP, which can be connected to a Microblaze processor via the Processor Local Bus (PLB), allowing the Microblaze processor to control the System Monitor and access the measurement data.

The System Monitor contains on-chip power-supply sensors, which are used to sense voltages in the range 0V to 3V with a resolution of approximately 3 mV. Once it has been sampled and digitized by the ADC, the measurement information is stored in the data registers.

We have implemented the measurement of V_{CCINT} and V_{CCAUX} using the System Monitor, which is controlled by a Microblaze on a ML506 platform. The System Monitor is connected to a Microblaze processor by the PLB so that the Microblaze can easily access the data registers of the System Monitor. After reading the ADC codes from the data registers, we can then calculate the voltages by (3.1).

$$Supply\,Voltage\,(Volts) = \frac{ADCCode}{1024} \times 3V \tag{3.1}$$

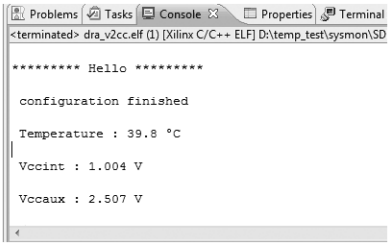


FIGURE 3.1 – The measured results of V_{CCINT} and V_{CCAUX}.

Fig. 3.1 shows the screenshot of the measured results of V_{CCINT} and V_{CCAUX} from Xilinx Software Development Kit (SDK).

3.2.1.2 How to Use It

It can be directly used as information to monitor the working state, making sure the system is under proper state.

Because we take the aforementioned Xilinx Virtex-5 ML506 board as the reference FPGA platform in this chapter, the voltage supplies are fixed and not reconfigurable.

For other platforms, if there are several voltage supplies, and the voltage can be switched among several levels during operation, it is possible to dynamically change the working voltage according to the power budget or performance requirement.

3.2.2 Temperature

Temperature is normally considered as a parameter of thermal constraint in a system. Therefore, it can be a useful metric.

3.2.2.1 How to Get It

Measured by System Monitor

Similar to the measurement of voltage described in subsection 3.2.1.1, it can also be measured by System Monitor. We can choose a visible way to access the System Monitor through JTAG, and display the measured die temperature in the ChipScope Pro tool on a PC. The ChipScope Pro tool provides a window, in which we can observe the variation of the die temperature curve.

We can also measure the die temperature independently by an embedded processor at runtime to avoid using an additional PC.

The System Monitor includes a temperature sensor, which is used to measure the die temperature. The relationship between the sensor output voltage and the die temperature is written in (3.2), which is proportional.

$$Voltage = 10 \times \frac{kT}{q} \times \ln(10) \tag{3.2}$$

Where :

k : Boltzman's constant = 1.38×10^{-23}.

T : Temperature $^{\circ}K$ (Kelvin).

q : Charge on an electron = $1.6 \times 10^{-19}C$.

Then, once the sensor output voltage has been digitized into a 10-bit digital output code (ADC code) by the ADC, we get a more simple function, which can be used to measure the die temperature, and is expressed in (3.3). The on-chip temperature sensor has a maximum-measurement error of $\pm 4^{\circ}C$.

$$Temperature(^{\circ}C) = \frac{ADCcode \times 503.975}{1024} - 273.15 \tag{3.3}$$

We have measured the die temperature in the same way as the measurement of V_{CCINT} and V_{CCAUX} in subsection 3.2.1.1 on the same platform. To be simple and brief, the Microblaze reads the ADC code from the temperature register of the System Monitor, and calculates the temperature by using (3.3). The screenshot of the measured temperature can be found in Fig. 3.1 in subsection 3.2.1.1.

Can be Indirectly Measured by Digital Thermal Sensor (Ring Oscillator)

From [95] we know that the temperature has a linear relationship with the frequency of the ring oscillator. Therefore, a digital thermal sensor, which is mainly based on a ring oscillator [96, 97, 98, 15], can be used to measure the temperature due to the linear relationship. If we can obtain the frequency of the ring oscillator, we are able to measure the temperature accordingly. Moreover, it uses few resources, and has the flexibility to be placed in different locations. Thus, the digital thermal sensor is able to measure the temperatures in different places.

In this subsection, we will explain how to use a digital thermal sensor to measure the temperature.

The digital thermal sensor, as shown in Fig. 3.2, is mainly made up of three parts : a ring oscillator, a 12-bit counter, and a 14-bit counter.

The ring oscillator is a feedback loop that should contain an odd number of inverters, because a signal passing through an even number of inverters does not change and thus does not produce an oscillation.

The frequency of the ring oscillator is defined by (3.4).

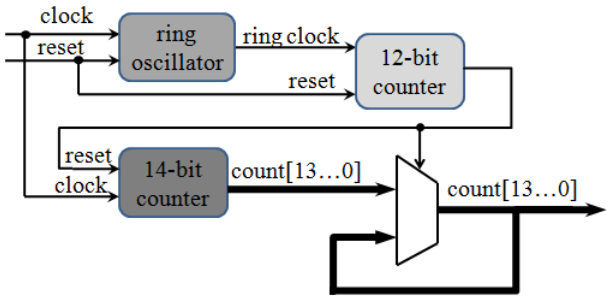


FIGURE 3.2 – The digital thermal sensor.

$$f = \frac{1}{2N\tau} \tag{3.4}$$

where N is the odd number of inverters, and τ the propagation delay of one inverter, assuming that the delays of all the inverters in the loop are the same.

In a CMOS technology circuit, higher temperatures result in larger propagation delays, thus in lower frequencies. Theoretically, we can find the relation between the frequency of the ring oscillator and the temperature, by counting the amount of times the oscillator fluctuates.

The 12-bit counter, which is clocked by the ring oscillator, is used to generate a Boolean signal for the 14-bit counter. This Boolean signal equals '1' if the value of the 12-bit counter is equal to 2^{12}, otherwise it is '0'.

The 14-bit counter computes the number of rising edges of the reference clock between two Boolean '1' from the 12-bit counter. Using the counted number from the 14-bit counter, we can calculate the frequency of the ring oscillator, and along with the temperature, we can get the relationship between the frequency of the ring oscillator and the temperature as expressed in (3.5).

$$f = a \times T + b \tag{3.5}$$

where

f : the frequency of the ring oscillator in MHz.

T : the temperature in degree Celsius ($°C$).

a : the negative slope, which means that higher temperatures result in lower frequencies.

b : a calibration constant, which can be easily calculated by a given initial temperature and its corresponding frequency of the ring oscillator.

3.2.2.2 How to Use It

The temperature varies as the system activity changes. It can be used to monitor the working condition of the system, providing the necessary information for decision making, to ensure that the system operates properly and does not infringe the thermal constraint.

If the temperature increases and gets close to the maximum safe operating temperature, the system has to take actions (e.g., turn on the cooling fan, scale down the frequency or voltage, decrease the workload), to cool down the equipment.

3.2.3 Current

Current is also a common parameter for a system. It has a great impact on the power consumption.

3.2.3.1 How to Get It

Of course, we can measure the current by a multimeter or an oscilloscope with the help of a shunt resistor. But we prefer to measure the current dynamically and independently during operation rather than to measure it with additional instruments.

Measured by System Monitor

As described above, the System Monitor provides 16 user-selectable analog inputs, known as auxiliary analog inputs ($V_{AUXP}[15:0]$, $V_{AUXN}[15:0]$). Taking advantage of this available tool, we can use a small shunt resistor to indirectly measure the current by measuring the voltage drop V_R over the shunt resistor, as shown in Fig. 3.3. A shunt resistor can be placed in series between power supply and voltage input. Then we can measure the voltage drop over the shunt resistor by the System Monitor at analog inputs $V_{AUXP}[0]$ and $V_{AUXN}[0]$. If the resistance of the shunt resistor is R, the current I can be calculated by Ohm's law from (3.6).

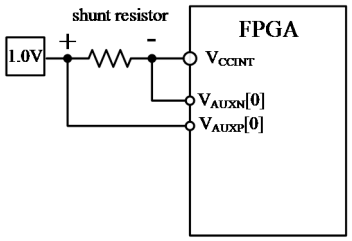


FIGURE 3.3 – Measure the current by a shunt resistor.

$$I = V_R \,/\, R \tag{3.6}$$

The Leakage Current Can be Indirectly Measured by Digital Thermal Sensor

As the semiconductor technology scales down, leakage current increases. And the leakage power dissipation is expected to exceed the dynamic power consumption in the sub-65nm geometries [99, 100].

As explained in subsection 3.2.2.1, a digital thermal sensor can be used to measure the temperature. From the experiment presented in [95] we know that it is a linear relationship between the frequency of the ring oscillator and the temperature expressed as (3.5).

On the basis of the Xilinx white paper [101] and the experiment in [95], we can conclude that the leakage current and the junction temperature have an approximately quadratic relationship, as shown in Fig. 3.4. We can express it as (3.7).

$$I_{CCINT} = a \times T^2 + b \times T + c \tag{3.7}$$

where T is the junction temperature in degree Celsius ($^\circ C$), and I_{CCINT} is the leakage current in milliamp (mA).

According to the above analysis, we can get the relationship between the leakage current and the frequency of the ring oscillator as shown in Fig. 3.5. In this way, we can indirectly measure the leakage current by a Digital Thermal Sensor (a ring oscillator).

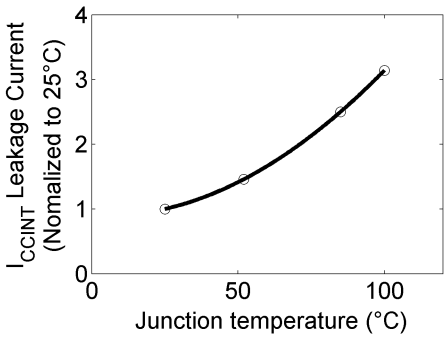


FIGURE 3.4 – Leakage current variations with Temperature.

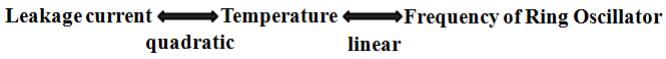


FIGURE 3.5 – The relationship between the leakage current and the frequency of the ring oscillator.

However, there is a limitation by this method, that is, it can only measure the leakage current but not the overall current. It is also a little complex and not so straightforward. But at least, it provides a solution to get the leakage current.

3.2.3.2 How to Use It

The current varies as time goes on. It can be used to show the relation between the workload and the power consumption of the system thus providing the necessary information for decision making. The leakage current can be used as a parameter to decide the implementation of a hardware PE with different area occupations.

3.2.4 Frequency

Frequency is also an important parameter and a useful metric. At the time of design, normally we know the frequency of a PE. If the working frequency of a PE is configurable, it can then be scaled up or scaled down during operation. Even though the frequency is scalable, we know the PE works at one of the available frequencies at one time. Therefore, we can store the frequency in a status register. When the frequency of the PE is changed

during operation, the value stored in the status register is changed as well, so that we can get the operating frequency at runtime.

Frequency can be a useful piece of information as regards decision making. It is also possible to take an action to change the frequency at runtime when the frequency is scalable.

3.2.5 Area, Position, and Resource

3.2.5.1 How to Get Them

We put these three metrics together, because they are related to each other. They can be obtained by using the same tool, PlanAhead, which is a software provided by Xilinx.

At the time of design, a PE can be placed at a particular position. Once the design is finished, we can get the position of the PE, the area occupied by the PE, and the resources used by the PE from PlanAhead software. For example, in a project, we have put a hardware PE on the upper left side of the FPGA. The hardware PE can be found as the pink rectangle in Fig. 3.6, to which the red arrow is pointing. We can also get the position of the PE, which is a rectangle from slice X8Y110 to slice X11Y114, as illustrated in Fig. 3.7. Furthermore, we can easily compute the area the PE occupies, which should comprise 20 slices (4×5).

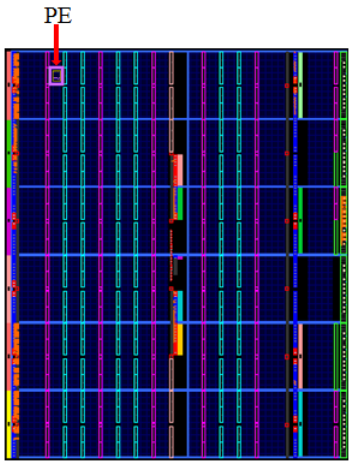


FIGURE 3.6 – The plan of the FPGA in Device view.

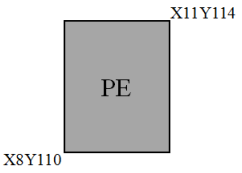


FIGURE 3.7 – The position information.

Table 3.1 gives the available resources inside the area, the resources the PE requires, and the percentages of the resources the PE uses.

These three metrics can be stored in registers.

TABLE 3.1 – Resources available and required.

Site Type	Available	Required	% Utility
LUT	80	57	72
FD_LD	80	32	40
SLICEL	10	8	80
SLICEM	10	8	80

3.2.5.2 How to Use Them

These three metrics give the necessary information for decision making. When the metrics of all PEs are available, we can effectively manage the system implementation. It is possible to implement different PEs within the same part of the device by taking advantage of dynamic partial reconfiguration technique. Different functionalities reuse the same resources, which are time multiplexed, thus saving space and resources.

More interesting scenarios deal with the displacement of a PE from one place to another. For instance, let's consider that a part of the FPGA is damaged (due to heat, radiation, etc.), in order to make the system continue to work properly, the functionality of the damaged part should be moved to another place. The decision maker searches a place that is both available and suitable, so as to meet the requirements in terms of area and resources needed, according to the necessary information provided by other parts.

The functionality of the damaged part is then moved to the new available position by means of the dynamic partial reconfiguration.

3.2.6 Activity Rate

When a PE is running, sometimes we would like to know how often it acts, i.e., its activity rate. Taking the clock signal as the reference, the activity rate can be defined as in (3.8).

$$Activity\,rate = \frac{en \times N}{c} \times 100\% \tag{3.8}$$

Where :

c : the number of clock cycles.

en : the number of clock cycles the enable signal lasts during c clock cycles.

N : a constant that indicates, given an input, how many clock cycles are required to generate an output.

Fig. 3.8 gives an example of a timing diagram. In this case, if $N = 1$, during $c = 10$ clock cycles, the activity rate = 20 %; if $N = 5$, the activity rate = 100 %. Of course, this is only an example for the sake of explanation, a value of $c = 10$ is too small, in practice, c must be properly selected, the larger is it, the more accurate is the activity rate.

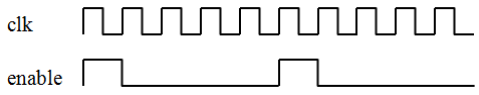


FIGURE 3.8 – A timing diagram example.

3.2.6.1 How to Get It

In order to calculate the activity rate, it needs two additional counters and a register. One is used to count the number of clock cycles c, while the other computes en, and N is stored in the register.

3.2.6.2 How to Use It

This metric is used to evaluate the performance of a PE and provide this information for decision making. The optimum activity rate is 100%, i.e., the clock resource is fully used without any waste, the PE processes data every clock cycle. If the activity rate is quite low, which means that the frequency is too high, it is then better to scale down the frequency.

3.2.7 Serial / Parallel

Due to the high degree of computational similarities [102], some PEs, e.g. MAC (multiply-accumulate) based PEs, can be implemented in serial for resource efficiency. But sometimes high performance requires the PE to be implemented in parallel mode. We take an N MAC operation for example, which is expressed in (3.9).

$$c = \sum_{i=0}^{N-1} a[i] \times b[i] \tag{3.9}$$

We can implement (3.9) in parallel, as shown in Fig. 3.9. N multipliers and N-1 adders are needed to perform the operation. We assume that c can be calculated within τ clock cycles.

Or (3.9) can be implemented in serial, as illustrated in Fig. 3.10. It needs only one multiplier and one adder, but takes $N\tau$ clock cycles to compute c.

This metric can be stored in a status register. At the time of design, the designer decides the PE to work in parallel, or in serial, or interchangeably between these two modes. This metric can work together with other metrics, to change the implementation of a hardware PE to have a trade off between performance, power consumption, and resource occupation.

3.2.8 Power Consumption

Power consumption is an important parameter we should consider when designing a system. As discussed in chapter 1, one of our objectives, which is also a motivation, is to reduce the power consumption. Therefore, the first thing is to measure the power consumption. Because power P equals voltage V times current I, $P = V \times I$, the me-

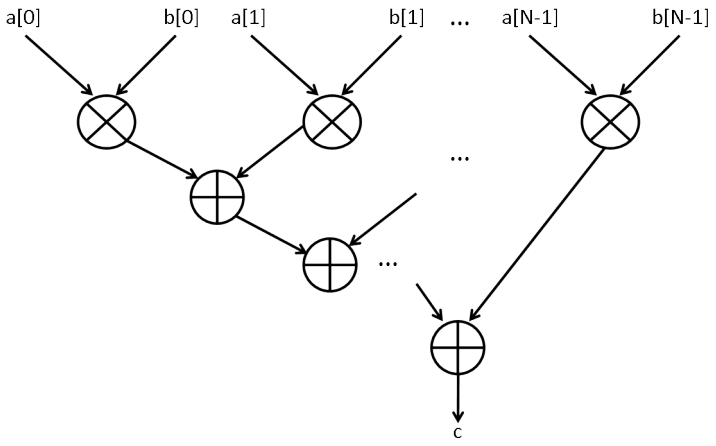


FIGURE 3.9 – parallel method.

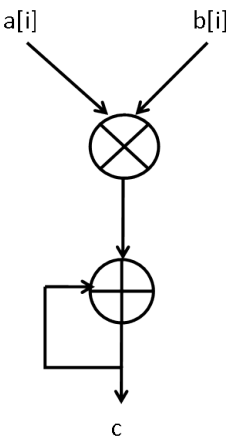


FIGURE 3.10 – Serial method.

thods used to measure voltage and current introduced in the previous subsections are also effective here to measure the power consumption.

3.2.8.1 How to Get It

There are several approaches to measure the power consumption, which will be introduced in the following subsections.

Estimated by XPower Analyzer

Xilinx provides two useful tools, Xilinx Power Estimator (XPE) and Xilinx Power Analyzer (XPA), to estimate and analyze the power consumption and the junction temperature of Xilinx devices.

The XPower Estimator spreadsheet is normally used in the early stages of a design, such as the pre-design and pre-implementation phases, with limited and incomplete information about the design [103]. After Place and Route, the complete real design data are available in the database, based on which, the XPA tool can then be used for more accurate power estimates and analysis [103]. It is the most accurate tool since it can read from the implemented design database the exact logic and routing resources used[104]. And Xilinx suggests to use the XPE for the pre-design power estimation, and the XPA for the post-implementation design power optimization.

Therefore, between these two Xilinx Power Tools, here, we prefer to choose XPA as the power consumption estimation tool.

Once the XPA has finished to run the power analysis, it provides the detailed power consumption information via a comprehensive graphical user interface (GUI).

Different views are available to navigate the power consumption of the design, either the Summary view, or the Details view : By Hierarchy, By Clock Domain, and By Resource Type.

- The Summary view displays the On-Chip power, the Supply power and the Thermal Properties.

- The By Resource Type view provides the power consumption for each type of resources that is used in the design, and also gives more details about the power dissipation at the resource level.

- The By Clock Domain view indicates the clock frequencies used by the design and the power they consumed.

- The By Hierarchy view lists the design hierarchy and power dissipated in each component.

With the helpful information provided by the XPA, we can have a detailed analysis of power consumption, and find out the most power hungry parts or components in the design, thereby offering efficient data-based reference for power optimization.

Measured by System Monitor

We have introduced the methods of measuring the voltage and current by means of the System Monitor in the previous subsections. Since we can get the voltage and the current at the same time with the same tool, it is natural that we can measure the power consumption by multiplying the voltage by the current.

Indirectly Measured by Digital Thermal Sensor (Ring Oscillator)

As discussed in subsection 3.2.3.1, the digital thermal sensor can only measure the leakage current. Therefore, in this way, only leakage power can be measured, moreover, it requires another tool to measure the voltage.

For this reason, although it is a favorable approach to measure the temperature, it may not be so good when it comes to measuring the power consumption.

3.2.8.2 How to Use It

Power consumption is quite a useful piece of information for decision making, and allows to consider the constraint of power budget. According to the power consumption, the system can choose an optimal solution based on other available information (voltage, frequency, activity rate, etc.), and take an action to reconfigure part of or the overall system to keep higher performance and at the same time reduce power consumption.

3.2.9 Performance to Power Consumption Ratio (PTCR)

We always hope that the system has good performance while consuming less energy. Therefore, we need such a metric that can provide a trade off between maximizing the performance and minimizing the power consumption. This metric can be the performance to power consumption ratio. We can reuse the metrics in the previous subsections, thus it can be defined as the ratio of Activity Rate to the Power Consumption as expressed in (3.10).

$$PTCR = \frac{Activity\ Rate}{Power\ Consumption} \times 100\% \tag{3.10}$$

For a given PE, the larger the value of the metric is, the better the PE works.

3.2.10 Working Mode

If the platform supports several different kinds of working modes, such as the wake-up, suspend, sleep, hibernation, and power down modes, the system can then switch from one mode to another at run-time. This metric can be stored in a status register, the value of which changes when the system switches between different working modes.

3.3 Discussion About the Metrics

These metrics can be considered under many angles so as to improve their use efficiency.

Depending on whether it is fixed, or self-changing over time during operation, a metric can be static or dynamic. If a metric is self-changing over time during operation, we can

TABLE 3.2 – Consideration of the metrics.

Metrics	Self-changeability	Configurability	Green impact	At which Level	Susceptibility
Voltage	static	medium	strong	System	Low
Current	dynamic	unconfigurable	strong	system	medium
Frequency	static	easy	strong	PE	Low
Temperature	dynamic	unconfigurable	strong	system	high
Area	static	medium	medium	PE & system	Low
Position	static	medium	weak	PE	Low
Resource	static	difficult	strong	PE & system	Low
Activity rate	dynamic	unconfigurable	medium	PE	medium
Serial / parallel	static	easy	medium	PE	low
Power consumption	dynamic	unconfigurable	strong	PE & system	medium
Performance to power consumption ratio	dynamic	unconfigurable	strong	PE	medium
Working mode	static	easy	strong	system	low

consider it as a dynamic metric, otherwise, it is a static metric, i.e., the metrics can be classified according to "self-changeability".

Some metrics are configurable while others are not, and among the configurable metrics, some are easy to configure, some are difficult, and the others are in the intermediate position. Some metrics, such as the Current, the Temperature, the Activity Rate, the Power Consumption, and the Performance to Power Consumption Ratio, are not directly configurable, but their values may change when some other metrics are reconfigured, e.g., scaling up the working voltage may increase the current thus the power consumption. The frequency of a PE can be configured by using a Digital Clock Manager (DCM), therefore, we think it is an easily configurable metric. Metrics Serial/Parallel and the Working Mode have the similar approach that switching between several available options. We mark the Voltage as being medium, because it is usually fixed, but if there exist several optional supplies it is then configurable. The Area and the Position are configurable if taking advantage of the DPR technique. But the Resource is difficult to configure even if using DPR, because the resource used by a PE is determined once it is designed.

With respect to the green impact, some metrics, such as the Voltage, the Current, the Frequency, the Temperature, the Resource, the Power Consumption, the Performance to Power Consumption ratio, and the Working Mode, have a great impact on the ambient environment, and some of them even have a direct influence on power dissipation and thermal emission. The Area, the Activity Rate, and the Serial/Parallel affect the power consumption but not so directly and obviously. The Position has a relatively weak effect on the working state of the system.

It is necessary to know at which level we use these metrics. Considering the specificity of the FPGA, we think that a metric should be at system level or at PE level. Usually, the Voltage, the Current, the Temperature, and the Working Mode are at system level ; the Frequency, the Activity Rate, the Serial/Parallel, and the Performance to Power Consumption Ratio are at PE level. The Area, the Resource, and the Power Consumption can be the metrics of a PE, or the metrics of the overall FPGA.

We should also consider that if a metric is easy to be influenced by the working state of the system and the ambient. The Temperature, it is highly susceptible to the ambient temperature and heat emission of the system. The Current, the Activity Rate, the Power Consumption, and the Performance to Power Consumption Ratio are affected by the running state of the system. The Voltage, the Frequency, the Area, the Position, the Resource, the Serial/Parallel, and the Working Mode are usually fixed values and the environment has little effect on these metrics.

Table 3.2 summarizes the discussion.

3.4 Case Study

We have proposed an application to show the use cases of the metrics. A FIR filter was employed as a hardware PE. We used a DCM to dynamically generate several different frequencies at the CLKFX output, which provides the input clock to the FIR filter. In this way, the working frequency of the FIR filter can be dynamically changed. A system monitor was used to measure the temperature of the FPGA. And the Microblaze worked as a controller to manage the DCM and the system monitor.

The experimental results were not ideal, since changing the working frequency of the FIR filter does not significantly influence the temperature of the system, which can be concluded that the FIR filter occupies only a small part of the FPGA and is not the most energy intensive component in the system.

Therefore, we focus on the PE level and implement only a FIR filter on the FPGA to analyze the power consumption of the filter. Some metrics : Serial / Parallel, Frequency, Power Consumption, and Resource, are involved in this study.

3.4.1 Parallel vs. Serial

For the sake of comparison and analysis, we choose two different implementation architectures : parallel architecture with 32 MACs (Multiply-accumulate), and serial architecture with only one MAC.

TABLE 3.3 – Resources used by the two implementation architectures.

Architecture	#FF	#LUTs	#DSPs
Parallel	3051	1067	64
Serial	1192	542	2

Table 3.3 gives the resources used by the two implementation architectures of the FIR filter. It is as expected that the parallel architecture consumes more resources than the serial one, which uses only two DSPs, while the parallel method consumes 32 times more, i.e. 64 DSPs.

It is generally thought that the serial architecture takes fewer resources and consumes less power than the parallel one. We have to try and see if this is true through the experiments. In order to minimize the influences from other components and make the results more accurate, we implement only a FIR filter on the FPGA with parallel architecture and serial way respectively, and estimate the power consumption of each architecture by using XPA.

The results are listed in Table 3.4, from which we can see that the serial architecture does consume less power than the parallel one when they are working at the same frequency. We have to point out that the quiescent power in Table 3.4 is the overall quies-

TABLE 3.4 – Power consumption of the FIR filter.

Frequency (MHz)	Power consumption (W)					
	parallel			serial		
	Dynamic	Quiescent	Total	Dynamic	Quiescent	Total
40	0.058	0.545	0.603	0.019	0.544	0.564
50	0.069	0.545	0.614	0.022	0.544	0.566
66.67	0.087	0.545	0.632	0.025	0.544	0.570
75	0.096	0.545	0.641	0.027	0.544	0.572
100	0.123	0.545	0.668	0.033	0.545	0.578
125	0.150	0.545	0.695	0.039	0.545	0.583
133.33	0.159	0.545	0.704	0.040	0.545	0.585
150	0.176	0.546	0.722	0.044	0.545	0.589
166.67	0.194	0.546	0.740	0.048	0.545	0.592

cent power of the FPGA. We also notice that the quiescent power between these two architectures have only a slight difference and is roughly the same, and what is more, it predominates the total power consumption, which is because the XPA provides only the overall quiescent power of the whole FPGA but the filter occupies a little part of the chip thus has little influence on the overall quiescent power. Therefore, we focus our main attention on the dynamic power.

Now we know that, working at the same frequency, the parallel approach consumes more power than the serial way, but the more interesting thing is to see what if these two implementation architectures have the same throughput, i.e., they can finish the same amount of computations during the same time duration.

We adjust the working frequency of the FIR filter in parallel architecture from 0.1MHz to 300MHz, so its corresponding working frequency in serial mode should be 32 times faster if they provide the same throughput, and we estimate the power consumption of both of these two architectures to analyze the pros and cons of the two methods.

Because of the large differences of the power consumption between the parallel and serial modes, in Fig. 3.11 we plot the parallel architecture using the bottom and left axes, and the serial one with the top and right axes, respectively.

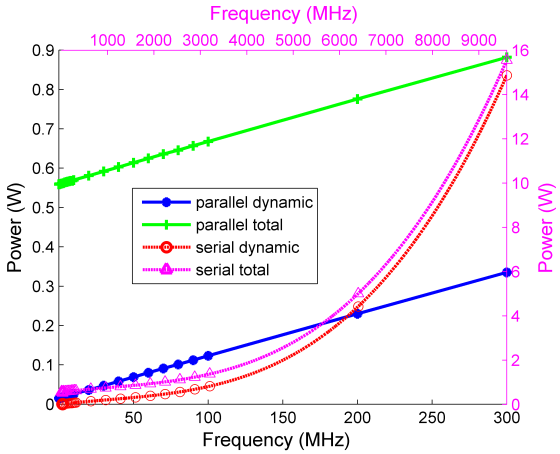


FIGURE 3.11 – The power consumption.

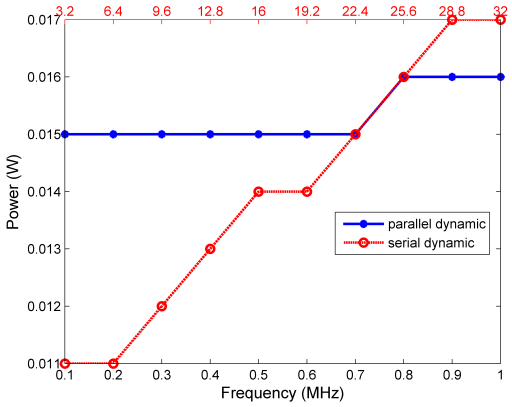


FIGURE 3.12 – The dynamic power.

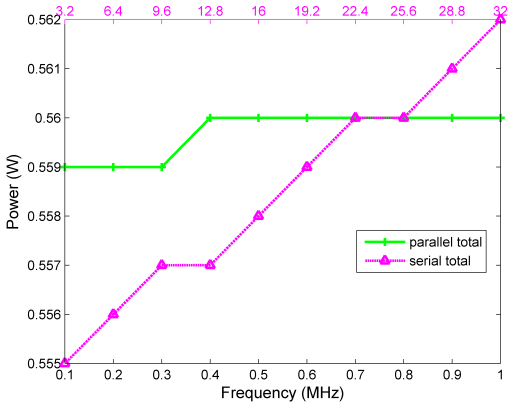


FIGURE 3.13 – The total power.

In parallel mode, when the frequency going from 0.1MHz to 300MHz, the power consumption of the FIR filter increases almost linearly and remains always under 1W. But in serial architecture, in order to keep the same throughput, the frequency should be changing from 3.2 MHZ to 9600MHz. When the frequency is less than 3200 MHz, the power consumption is under 1.5 W, but when the frequency is becoming higher, the curve becomes steeper and steeper, i.e., the power consumption increases more and more significantly and is becoming much more than that of the parallel architecture. This can be explained by the fact that the clock is a power consuming element inside the FPGA,

while the FIR filter only takes a little part of the FPGA. In particular, when the frequency goes higher, the clock dominates the total power consumption.

In both of the two architectures, the quiescent power, which is the gap between the dynamic power and total power in Fig. 3.11, is almost a static value. The reason, as discussed above, is that the quiescent power refers here to the overall quiescent power of the whole FPGA, and the filter only occupies a little part of the chip thus has little effect on the overall quiescent power.

Fig. 3.11 uses different vertical axes, we cannot see the details when these two architectures have comparable power consumption, therefore, we zoom into the region when the frequency in parallel mode is ranging from 0.1 MHz to 1MHz. The dynamic power and total power are illustrated in Fig. 3.12 and Fig. 3.13 respectively.

We can see that when the frequency is quite low, the parallel method consumes more power than the serial one, as the frequency goes higher, the power is equivalent when the frequency in parallel mode is 0.7 MHz to 0.8 MHz (22.4 MHz to 25.6 MHz for the serial mode), and the power consumption in serial mode overtakes that of the parallel method when the frequency is higher than 25.6 MHz (higher than 0.8 MHz in parallel mode). Therefore, it is better to adopt the serial architecture when the working frequency is low, since we can benefit from it with less resources as well as lower power consumption. But when the frequency goes higher, the clock will consume more power than the FIR filter itself, and in this case, the parallel architecture becomes the preferable choice. This provides useful information for the management architecture to make a proper decision.

3.4.2 Power Consumption with Different Number of Taps

After analyzing the power consumption when the filter is implemented in parallel and serial modes, we would like to estimate how the number of taps affects the power consumption.

Therefore, we implement the FIR filter with 32 taps, 64 taps, and 128 taps, respectively. And then we estimate the power consumption of the filter when the working frequency varies from 40MHz to 300MHz. Fig. 3.14 gives the dynamic power consumption and the total power consumption of the filter implemented with three different numbers of taps when the frequency increases. As expected, the filter with larger number of taps consumes

more power than the one with fewer taps when they work at the same frequency, and when the working frequency grows higher and higher, the more taps the filter has, the more rapidly the power consumption increases.

Fig. 3.15 shows the dynamic power consumption according to the number of taps. When working at 40MHz, the power consumption increases slowly when the number of taps grows from 32 to 128. But when the frequency is changing from 40 MHz to 300MHz, the lines become steeper and steeper, which means that when the number of taps increases, the higher frequency the filter works at, the faster the power consumption increases.

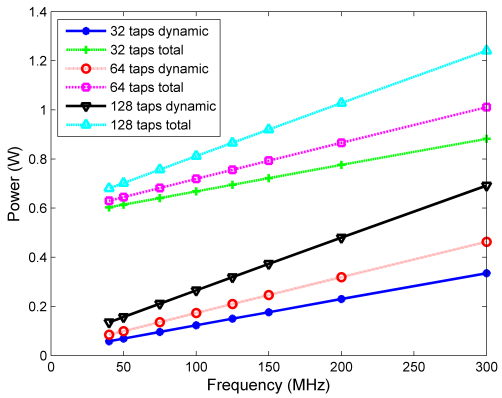


FIGURE 3.14 – Power consumption of the filter with three different numbers of taps when the frequency increases.

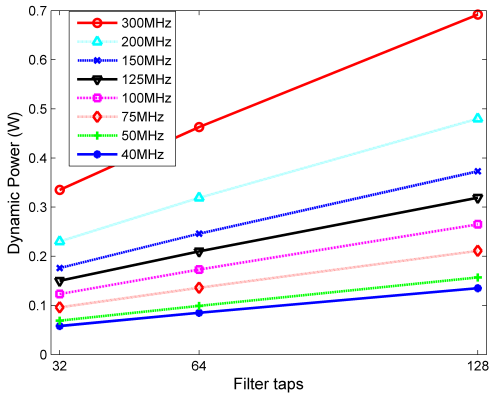


FIGURE 3.15 – Dynamic power consumption according to the number of taps.

TABLE 3.5 – The Relationship between Power Consumption, Performance and Resources.

Frequency	32 MACs	64 MACs	128 MACs
40 MHz	1	1.47	2.33
50 MHz	1.19	1.71	2.71
75 MHz	1.66	2.34	3.64
100 MHz	2.12	2.98	4.57
125 MHz	2.59	3.62	5.50
150 MHz	3.03	4.24	6.43
200 MHz	3.97	5.50	8.28
300 MHz	5.78	7.98	11.93

3.4.3 Evaluation of the Relationship between Power Consumption, Performance and Resources

In order to make the results clearer, we take the dynamic power consumption of the filter when it is working at 40MHz with 32MACs as a reference, and then normalize the power consumption of the other cases to the reference power. The normalized power consumption is listed in Table 3.5, from which we can see that the power consumption varies quite differently, the maximum difference can reach 10.93 times. Increasing the number of MACs will consume more resources but at the same time improve the performance. A more direct way to improve the performance is to scale up the working frequency. For example, when it is working at 50MHz with 32MACs, if it is required to double the performance, there are two choices : one is increasing the number of MACs to 64 resulting in the use of more resources, which will consume about 44% more power ; while the other way is to scale up the frequency to 100MHz, in which case the power consumption will be about 78% more. Therefore, a trade off between the power consumption, the performance, and the resources needs to be found. According to the scenarios, the decision maker has to select an optimal choice according to the working state, available resources, and the environment. In next subsection, two management cases about these metrics will be discussed.

3.4.4 Metrics Management by HDCRAM

Based on the analysis of the FIR filter in the above subsections, the metrics involved in the above studies are Serial / Parallel, Frequency, Power Consumption, and Resource. Once these metrics have been obtained, they can be managed and used by the HDCRAM architecture. Fig. 3.16 gives a simple example of a use case of these metrics.

According to these metrics, we introduce three PEs, namely the DCM, the FIR filter, and a resource calculator as shown in Fig. 3.16. The DCM generates different clock frequencies for other PEs. The resource calculator computes the resources the system uses, and it is better to be a software PE that resides in an embedded processor (e.g. Microblaze). The metrics of these three PEs, managed by their corresponding level 3 managers, are then submitted to L2_CRMU. Two management cases are discussed as below.

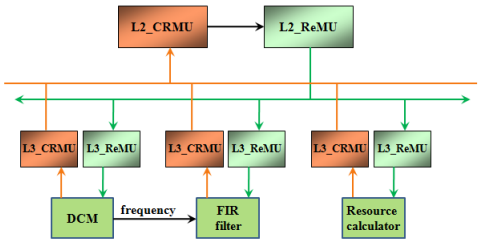


FIGURE 3.16 – An example of level 2 HDCRAM management.

3.4.4.1 Case 1

Depending on the results in the above subsections, the FIR filter consumes less power and resources in serial mode when the throughput is lower than 0.8 MHz. Therefore, it is better to work in serial when the throughput is lower than 0.8 MHz. The L3_CRMU of the FIR always makes the decision to work in serial mode when the required throughput of the FIR filter is lower than 0.8 MHz, without the help of other PEs.

3.4.4.2 Case 2

There are two ways to improve the performance of the FIR filter : increasing the working frequency or increasing the number of MACs. The L3_CRMU of the FIR filter can not finish the work itself, because it needs the collaboration with other PEs, i.e., the

DCM and the resource calculator. Therefore, the right of decision making is submitted to the L2_CRMU to decide how to reconfigure the PEs to improve the performance. Depending on the metrics the L2_CRMU obtained from the L3_CRMUs of the three PEs, the L2_CRMU makes the decision to select the best strategy and sends the corresponding reconfiguration commands to the L2_ReMU.

For example, given the reference when the FIR filter is working at 50MHz with 32MACs, in order to double the performance, depending on the working situation, there are two solutions :

- The L2_CRMU has the information of how many resources are occupied and how many are still available. If the required resources are available, the L2_CRMU makes the decision to increase the number of MACs to 64 and sends the reconfiguration command to L2_ReMU. And finally the reconfiguration command comes to the L3_ReMU of the FIR filter, who performs the reconfiguration action. This method will consume more resources and about 44% more power.

- But if no more resource is available, the L2_CRMU has to decide to increase the working frequency of the FIR filter to 100MHz, and sends the reconfiguration command to the L2_ReMU and then to the L3_ReMU of the DCM, who reconfigures the DCM to generate a 100MHz output clock frequency instead of a 50MHz one. This method will consume about 78% more power but without additional resources consumption.

3.5 Conclusion

An efficient architecture is required to manage the cognitive equipment. The management architecture needs proper metrics to sense the surroundings and efficiently reconfigure the system thus adapting to the working environment. In this chapter, we take the Xilinx Virtex-5 ML506 board as the reference FPGA platform, and introduce some useful metrics that can be used by the HDCRAM architecture, as well as some measurement approaches of the metrics. For other platforms, some methods described in this chapter should be adjusted accordingly. As an example of the use case of the metrics, we study the power consumption of a FIR filter when it is implemented in parallel and serial modes and works in different frequencies. The results are useful for decision making, which sug-

gest that it is better to work in serial mode when the frequency is low, otherwise, the parallel method is recommended. We also analyze the power consumption when the filter is implemented with three different numbers of taps, which shows that there is a trade off between the power consumption, the performance, and the resources. The system is then able to make a proper decision based on the information it has obtained.

Chapter 4

OFDM transmitter and receiver example

4.1 Introduction

The Orthogonal Frequency Division Multiplexing (OFDM) technique is one of the most important methods of digital modulation. OFDM can transmit large amounts of digital data simultaneously at different frequencies by splitting a signal into several closely spaced orthogonal narrow-band channels at different frequencies in the available bandwidth. Moreover, one of the advantages of OFDM over Frequency Division Multiplexing (FDM) is the efficient use of spectrum by spacing the channels much closer together allow. This is achieved by choosing all the sub-carriers that are orthogonal to each other, thus enabling the sub-carriers to be spaced very close.

Therefore, OFDM has been adopted for various standards in wireless communications, such as Wireless Local Area Network (WLAN) [105], Digital Audio Broadcasting (DAB) [106], Digital Video Broadcasting (DVB) [107], and Long-Term Evolution (LTE) [108].

In this chapter, we would like to introduce some management scenarios of an OFDM system with software/hardware co-design.

4.2 OFDM system model

A simplified OFDM system model has been employed in our studies as shown in Figure 4.1.

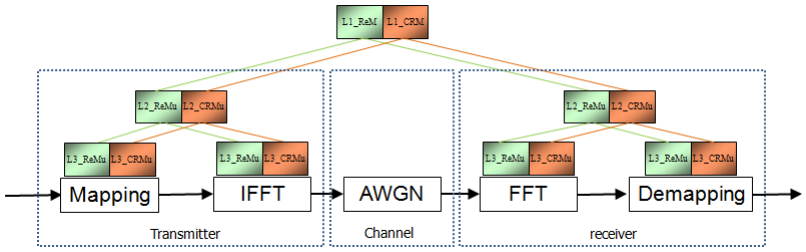


FIGURE 4.1 – The block diagram of a simplified OFDM system model.

It consists of three parts : a transmitter, a receiver, and an additive white Gaussian noise (AWGN) channel. The transmitter has two blocks : Mapping and Inverse Fast Fourier Transform (IFFT), and the receiver has also two corresponding blocks : Fast Fourier Transform (FFT) and Demapping. These blocks are described as below.

The transmitter :

Mapping :

The input data are converted into groups of n bits depending on the digital modulation techniques used (e.g., 2 bits -QPSK, 4 bits -16QAM), and then mapped on to required modulation format (i.e., complex values (I+jQ) representing the mapped constellation point that specify the amplitude or phase or both amplitude and phase of the sinusoid for their associated subcarriers).

IFFT :

The complex symbols are then input to the IFFT, which provides an efficient and simple way to superimpose the complex data points onto the required orthogonal sub-carriers. The output samples from the IFFT make up a single OFDM symbol.

Channel :

An additive white Gaussian noise (AWGN) channel model is then applied to the transmitted signal. The model is used to simulate the radio channel, which allows for the

signal to noise ratio (SNR) to be controlled to change the channel condition. The SNR is set by adding a known amount of AWGN to the transmitted signal.

The receiver :

FFT :

After the signal is transmitted across the radio channel, at the receiver, a FFT block is used to process the received signal and transform it into the frequency domain which is used to recover the original data bits.

Demapping :

The signal of each sub-carrier is then evaluated and demodulated back to the data bits. The data bits are then combined back to the same word size as the original data.

4.3 Implementation Platform

The whole OFDM system can be implemented on a GPP. Or we can put some elements on an embedded system such as Zynq platform. In order to show the heterogeneous and distributed management, we implement the transmitter on a PC and the receiver on a Zynq platform, which is introduced in subsection 2.3.2. Figure 4.2 illustrates the implementation platform consisting of a PC and a Zynq board. For the sake of clarity, we treat the transmitter as the base station, and the receiver as the terminal. As described in the previous chapters, the link between the PC and Zynq platform is through Ethernet, and they communicate by using UDP protocol.

4.4 FFT implementation using partial reconfiguration

Since the FFT is one of the most computationally intensive elements of the OFDM system, it is a good choice to offload the FFT in hardware on PL (Figure 4.3) to alleviate the workload of the PS ; of course it can also be implemented in software on PS (Figure 4.4).

The FFT has been considered as a common operator for many classical telecommunications operations [109, 110, 111]. In order to support multi communication standards, the FFT size should be reconfigurable to adapt to the operating standard.

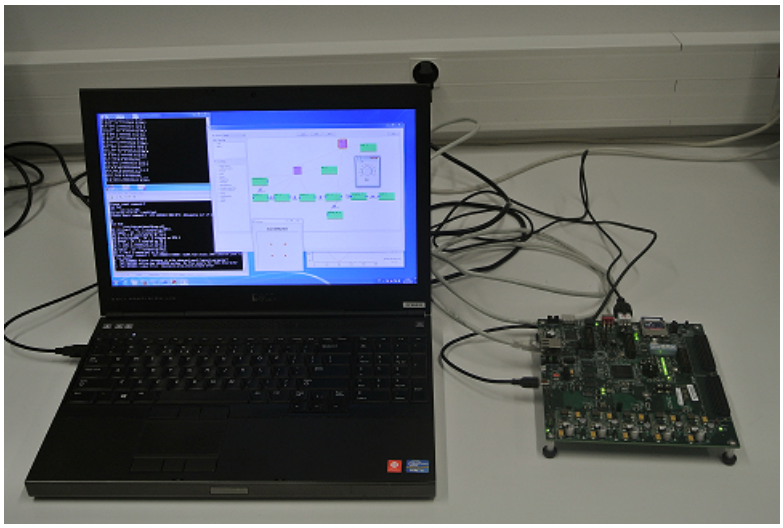


FIGURE 4.2 – Implementation platform.

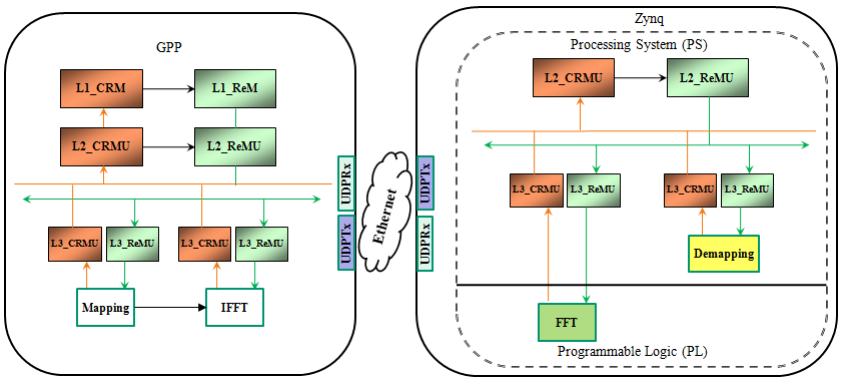


FIGURE 4.3 – The hardware implementation of FFT.

Traditional reconfigurable FFT has to implement the maximum transform length that the FFT can support, even it is not frequently used. Hence, this method uses more resources and consequently consumes more power.

Instead, we would like to implement the FFT by taking advantage of dynamic partial reconfiguration. The FFT implementations with different transform lengths share the resource in the same reconfigurable region. Each FFT implementation corresponds to a transform length. Moreover, the approach using DPR not only supports the reconfiguration of the transform length, but also the implementation architecture, e.g., pipelined

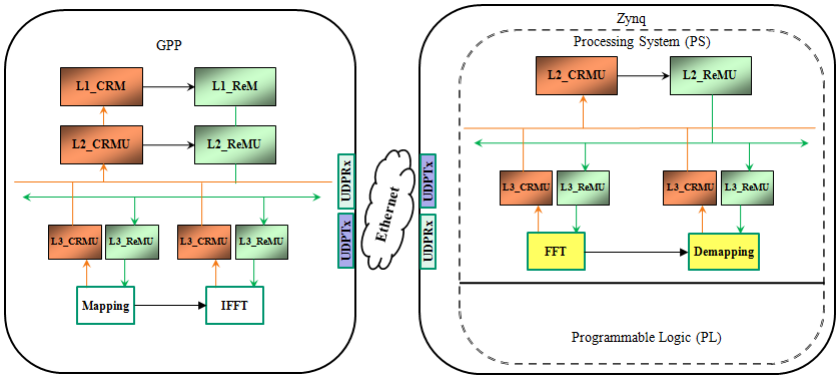


FIGURE 4.4 – The software implementation of FFT.

architecture or single radix-2 architecture (the introduction of the pipelined architecture and single radix-2 architecture can be found in Appendix D). Therefore, these options offer a trade-off between resource utilization and transform time. Depending on the scenarios, the FFT can be easily reconfigured by choosing either performance or resource efficiency.

These implementations of FFT with different architectures and transform lengths are generated by using Xilinx FFT core [112]. The FFT is implemented in the reconfigurable region on the upper right side of the FPGA on Zynq platform, which can be found as the pink rectangle in Figure 4.5.

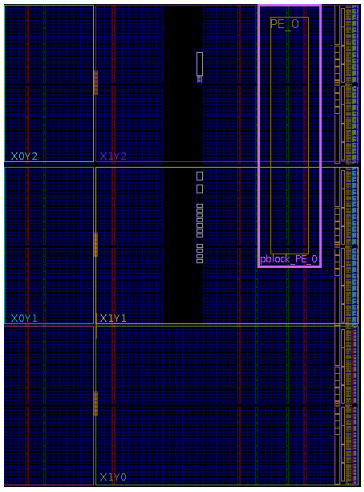
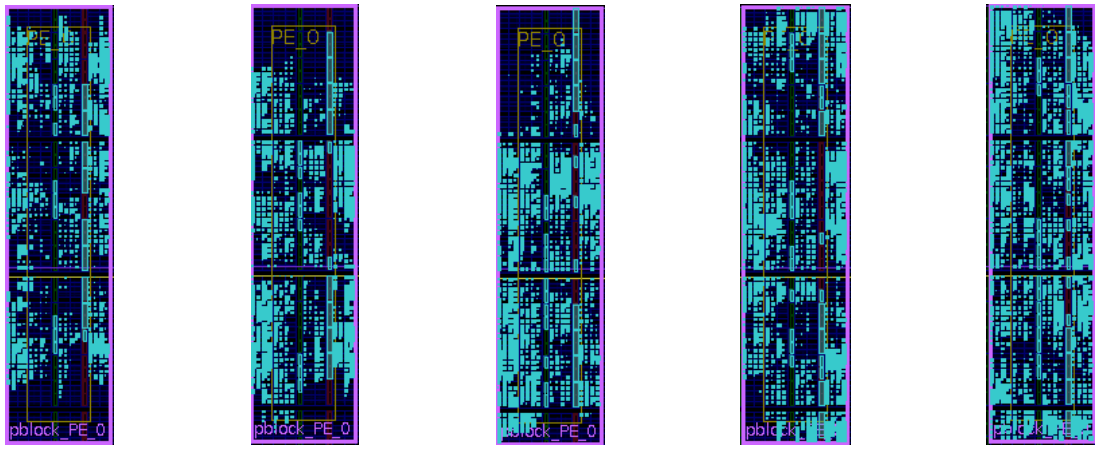


FIGURE 4.5 – Implementation of FFT using partial reconfiguration.

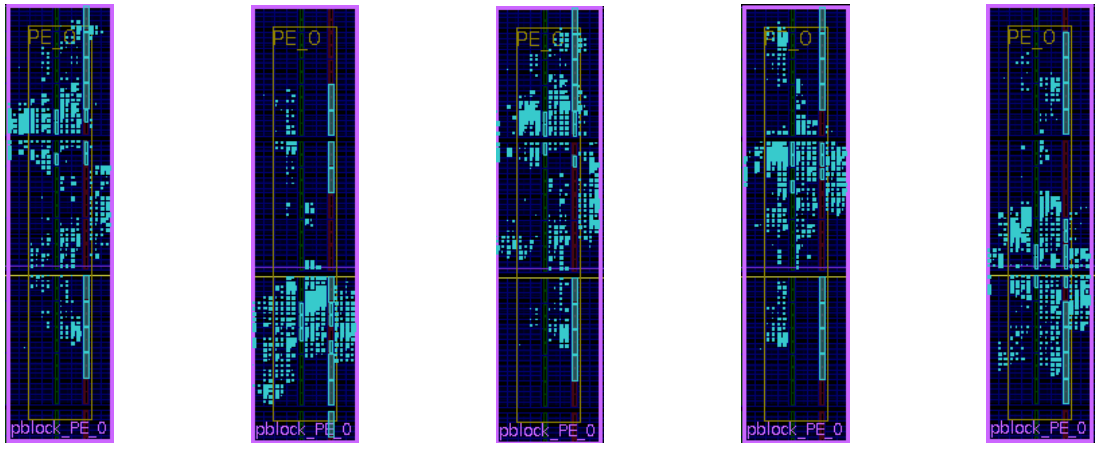
4.4.1 Resource Utilization

Figure 4.6 and Figure 4.7 show the implemented FFT of different transform lengths using partial reconfiguration with pipelined architecture and single radix-2 architecture respectively.



(a) FFT128 (b) FFT256 (c) FFT512 (d) FFT1024 (e) FFT2048

FIGURE 4.6 – Implementation of FFT with pipelined architecture using partial reconfiguration.



(a) FFT128 (b) FFT256 (c) FFT512 (d) FFT1024 (e) FFT2048

FIGURE 4.7 – Implementation of FFT with single radix-2 architecture using partial reconfiguration.

As can be seen from these figures, it is apparent that the single radix-2 architecture consumes less resource than the pipelined architecture. Table 4.1 lists the resource available in the reconfigurable region and used by the FFT with different transform lengths and implementation architectures.

TABLE 4.1 – Resources available and used by different FFT implementations in the reconfigurable region.

Transform length	Resource	LUT	Register	SLICE	DSP48E1	BRAM
	Available	5184	10368	1359	32	48
128	pipelined	2806	3196	702	9	9
	radix-2	1067	1316	268	3	11
256	pipelined	3175	3578	795	9	10
	radix-2	1101	1361	276	3	11
512	pipelined	3589	4113	898	12	12
	radix-2	1154	1392	289	3	11
1024	pipelined	3993	4507	999	12	14
	radix-2	1153	1425	289	3	11
2048	pipelined	4455	5086	1114	15	19
	radix-2	1194	1491	299	3	13

In Table 4.1 we can see that resource use of the single radix-2 architecture of different transform lengths has small difference while resource occupation of the pipelined architecture varies distinctly, which is because the single radix-2 architecture uses only one radix-2 butterfly processing engine and the pipelined architecture pipelines several radix-2 butterfly processing engines to offer the ability of continuous data processing. Therefore, the pipelined architecture has a better performance and the single radix-2 architecture is more resource efficient.

The resource used by the traditional reconfigurable FFT with the pipelined architecture enabling reconfigurable transform length from 128 to 2048 is also listed in Table 4.2. It shows that the traditional reconfigurable FFT consumes more resource than the

maximum transform length of the partial reconfiguration approach, which is because it needs additional control logic.

TABLE 4.2 – Resources used by traditional reconfigurable FFT implementation with pipelined architecture.

LUT	Register	SLICE	DSP48E1	BRAM
5741	6056	1435	15	19

4.4.2 Transform time

Now we would like to see the performance of different FFT implementations. The transform time is the time used by the FFT to compute a transform. The transform time of different FFT implementations is listed in Table 4.3. The hardware implementations have better performance, while the software implementations consume more time. And the hardware pipelined architecture has the best performance with the price of more resource occupation than the radix-2 architecture. The transform time of the traditional reconfigurable FFT is longer than the pipelined architecture using DPR and shorter than the radix-2 architecture using DPR, because it also employs the pipelined architecture.

TABLE 4.3 – The transform time of different FFT implementations.

Transform length	Software (μs)	Hardware (μs)		Traditional reconfigurable FFT (μs)
		pipelined	radix-2	
128	166	4.81	8.29	4.92
256	364	8.71	16.77	8.93
512	798	16.49	34.85	16.61
1024	1751	31.91	73.41	32.13
2048	3867	62.73	155.49	62.85

4.4.3 Reconfiguration time

The timing overhead of full and partial reconfiguration should also be considered. The sizes and the time consumption of full and partial bitstreams of the FFT design

are listed in Table 4.4. The reconfiguration time of the traditional reconfigurable FFT is generally in several clock cycles, thus we consider it is negligible compared with the partial reconfiguration time.

TABLE 4.4 – Full and partial configuration time of the FFT design.

Type	Size (bytes)	Time (μs)
Full	4 045 564	211 413
Partial	384 512	35 122

4.4.4 Power consumption

The power consumption of the FFT implementations of the DPR approach is listed in Table 4.5. And the power consumption of software FFT and traditional reconfigurable FFT is also included in Table 4.6. The DPR approach consumes less power than the traditional reconfigurable FFT. And within the DPR approach, the pipelined architecture consumes more power than the radix-2 architecture. The software FFT consumes comparable power to the 2048 point pipelined architecture of DPR approach, but considering the transform time, the total energy consumption of the software FFT would be higher than the DPR approach.

TABLE 4.5 – The power consumption of different FFT implementations of the DPR approach.

Transform length	Power consumption (W)	
	pipelined	radix-2
128	0.103	0.096
256	0.105	0.097
512	0.108	0.098
1024	0.113	0.099
2048	0.121	0.101

We have tried to measure the power consumption during the partial reconfiguration process using TI Fusion Digital Power Designer [92], but because of the partial reconfigura-

TABLE 4.6 – The power consumption of software FFT and traditional reconfigurable FFT.

	Software FFT	Traditional reconfigurable FFT
Power consumption (W)	0.12	0.135

tion time is too short, we cannot catch the power changes during partial reconfiguration. Then instead, we tried to measure the power consumption of the full reconfiguration, which is shown in Figure 4.8. Even so, sometimes we still cannot catch the power changes and had to try several times. We take this measurement result as the reference of the power consumption of the partial reconfiguration, which is around 0.07W.

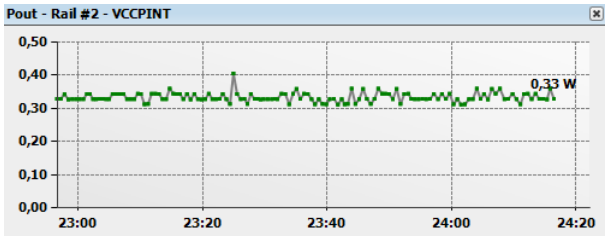


FIGURE 4.8 – Power consumption of reconfiguration.

The benefits of DPR are three-fold : 1) each option uses less resource and consumes less power than the traditional reconfigurable FFT ; 2) the implementation of these options can be dynamically changed which provides further flexibility and possibly power reduction ; and 3) a blank partial bitstream can be loaded to clear the reconfigurable region. Of course this is achieved with the cost of additional reconfiguration time and power consumption.

Theoretically, the energy consumption includes the configuration energy and operating energy as expressed in (4.1).

$$E = P_{config} \times t_{config} + P_{run} \times t_{run} \tag{4.1}$$

Where :

E : energy consumption.

P_{config} : power consumption of configuration.

P_{run} : power consumption during operation.

t_{config} : configuration time.

t_{run} : operating time.

The energy consumed by DPR approach can be expressed as (4.2) :

$$E_{pr} = P_{config} \times t_{config} + P_{pr\_run} \times t_{run} \tag{4.2}$$

We consider the configuration energy of traditional reconfigurable FFT is negligible. Hence the energy consumption can be expressed as (4.3) :

$$E_{tradition} = P_{tradition\_run} \times t_{run} \tag{4.3}$$

To make sure the DPR approach consumes less energy, i.e., $E_{pr} < E_{tradition}$, the operating time should be under the constraint condition in (4.4).

$$\begin{aligned} P_{config} \times t_{config} + P_{pr\_run} \times t_{run} &< P_{tradition\_run} \times t_{run} \\ t_{run} &> \frac{P_{config} \times t_{config}}{P_{tradition\_run} - P_{pr\_run}} \end{aligned} \tag{4.4}$$

This results in tens of milliseconds. But the traditional approach is running all the time. Moreover, the DPR approach can clear the reconfigurable region by loading a blank partial bitstream, which could achieve further energy reduction.

In conclusion, the DPR approach is advantageous over the traditional reconfigurable FFT in terms of resource utilization, performance (with the same architecture), power consumption and flexibility, except for the reconfiguration time. Therefore, in this chapter, we adopt the DPR approach to implement the FFT.

In the following subsections, we introduce some management scenarios using the simplified OFDM system model described in section 4.2.

4.5 Scenario 1 : Modulation Adaptation

SCEE team has developed an application demonstrating the modification of modulation scheme of the transmission channel according to the SNR level [113]. But this

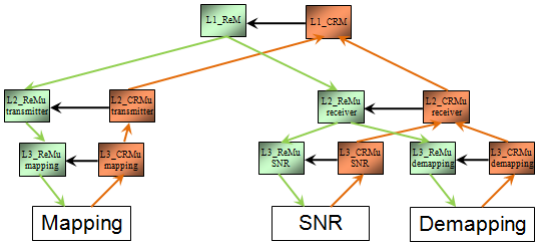


FIGURE 4.9 – Scenario 1.

application did not include the HDCRAM management. In this section, the scenario illustrates the HDCRAM management of modulation adaptation depending on the channel conditions. When the radio channel condition is good, 16 quadrature amplitude modulation (16QAM) is used to achieve high bit rates. On noisy channels, the OFDM system adapts to provide reliable communications using quadrature phase shift keying (QPSK), which is more robust. It involves three PEs, Mapping, SNR, and Demapping. The metric SNR is used in this scenario.

L3_CRMu SNR

The PE SNR is a channel condition sensor. The metric SNR is managed by the L3_CRMu SNR and then is sent to the upper level manager L2_CRMu receiver.

L3_CRMu demapping

The demodulation scheme of PE Demapping is managed by the L3_CRMu demapping, and then is sent to the upper level manager L2_CRMu receiver.

L2_CRMu receiver

Based on the value of SNR received from L3_CRMu SNR, L2_CRMu receiver makes decisions to adapt the modulation scheme to the noise levels, for example :

- If 5dB < SNR <=10dB, which is interpreted as the channel condition is poor, QPSK is chosen to provide reliable communications and to improve robustness.

- If SNR > 10dB, which means the channel condition is good, 16QAM is employed to increase throughput and to achieve high bit rates.

Then, L2_CRMu receiver compares its decision with the current running demodulation scheme received from L3_CRMu demapping, if they are the same, no reconfiguration is needed, else if they are different, L2_CRMu receiver should not change the demodulation scheme of the receiver itself, instead, it has to inform the transmitter through L1_CRM. Therefore, it sends the required modulation scheme to L1_CRM. For example, if the SNR obtained is 6dB, the modulation scheme should be QPSK, but the element Demapping is running at 16QAM, then L2_CRMu receiver sends the required modulation scheme QPSK to L1_CRM through Ethernet to reconfigure both transmitter and receiver to run at QPSK.

L1_CRM

If L1_CRM receives the new demodulation scheme from L2_CRMu receiver, which means the channel condition has been changed. Therefore, it is necessary to reconfigure the modulation scheme of both the transmitter and the receiver to adapt to the channel condition. So L1_CRM sends reconfiguration command to its associated L1_ReM.

L1_ReM

If the L1_ReM receives the command from L1_CRM, it then sends the command to its target lower level managers, namely L2_ReMu transmitter and L2_ReMu receiver.

L2_ReMu transmitter

In this example, if the L2_ReMu transmitter receives the command from its upper level manager L1_ReM, it then takes action to execute the command and sends reconfiguration command to L3_ReMu mapping.

L3_ReMu mapping

L3_ReMu mapping manages the reconfiguration of its associated PE Mapping. If L3_ReMu mapping receives command from its upper level manager L2_ReMu transmitter, it then executes the command. In this example, L3_ReMu mapping reconfigures the

modulation scheme of PE Mapping from 16QAM to QPSK.

L2_ReMu receiver

In this example, if the L2_ReMu receiver receives the command from its upper level manager L1_ReM, it then takes action to execute the command and sends reconfiguration command to L3_ReMu demapping.

L3_ReMu demapping

L3_ReMu demapping manages the reconfiguration of its associated PE Demapping. If L3_ReMu mapping receives command from its upper level manager L2_ReMu receiver, it then executes the command. In this example, L3_ReMu demapping reconfigures the modulation scheme of PE Demapping from 16QAM to QPSK, in accordance with its corresponding PE Mapping in the transmitter.

After the above processes managed by HDCRAM, both transmitter and receiver are properly reconfigured, and finally the OFDM system self-adapts itself to the changing channel condition.

This scenario can visually demonstrate the modulation adaptation when SNR changes using constellation diagram, as shown in Figure 4.10 and Figure 4.11.

4.6 Scenario 2 : Management of FFT implementation type depending on the hardware resource utilization

When the receiver is implemented on Zynq platform, the PE FFT could be implemented either in software on PS or in hardware on PL. Furthermore, by taking advantage of dynamic partial reconfiguration, the hardware implementation of FFT can use different architecture options, e.g., pipelined architecture or single radix-2 architecture, to offer a trade-off between resource utilization and transform time. Therefore, the FFT has three implementation options :

- Software
- Hardware Pipelined
- Hardware Radix-2

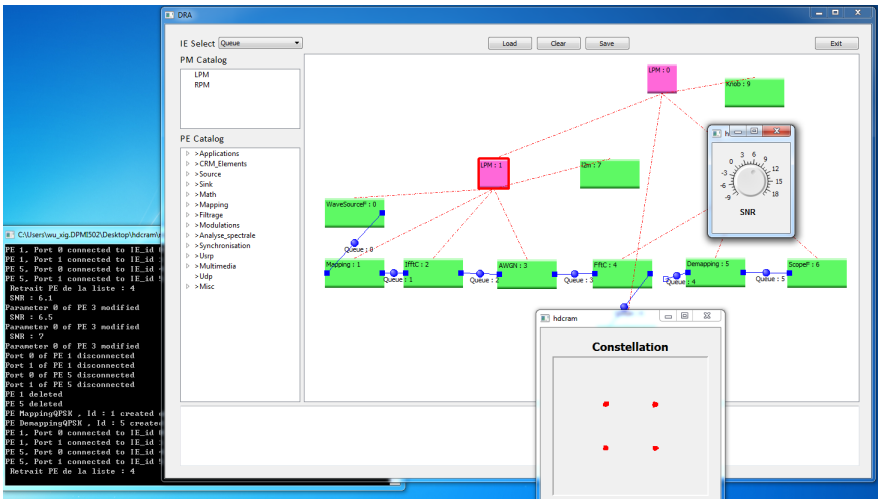


FIGURE 4.10 – Adaptation to QPSK when SNR < 10.

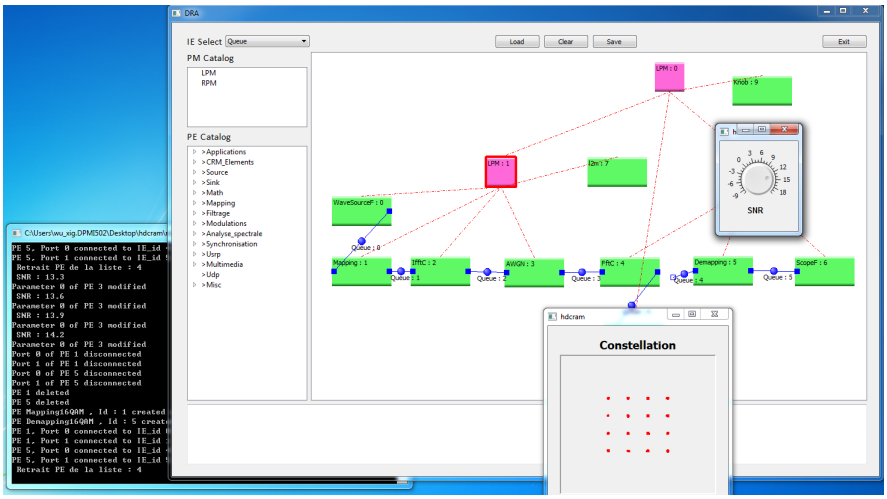


FIGURE 4.11 – Adaptation to 16QAM when SNR > 10.

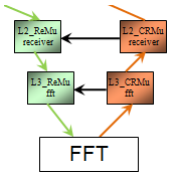


FIGURE 4.12 – Scenario 2.

The resource consumed by the hardware implementation of FFT can be calculated at the time of design. This example shows the management of FFT implementation, de-

pending on the hardware resource utilization, the implementation type of FFT can be dynamically changed between software, hardware pipelined, and hardware radix-2. As discussed in section 4.4, the DPR approach is advantageous. Hence, the hardware implementation of the FFT takes advantage of DPR. The metrics involved are FFT_type (similar to the metric Serial / Parallel introduced in subsection 3.2.7) and Resource (introduced in subsection 3.2.5). This example takes FFT size 256 as a reference.

L3_CRMu fft

Theoretically, the resource utilization of hardware FFT should be managed by the L3_CRMu fft. But currently we calculate the resource utilization of hardware FFT at the time of design instead of dynamic measurement. Therefore, only the metric FFT_type (software, hardware pipelined, hardware radix-2) is sent to the upper level manager L2_CRMu receiver.

L2_CRMu receiver

L2_CRMu receiver manages the hardware utilization of all the hardware PEs. The available resource (R_left) equals 100% - total hardware utilization. It is preferred to implement the FFT in hardware because of the high performance and lower power consumption.

Because we can get the resource utilization at the time of design, the metrics Resource and FFT_type are made one pair and stored in a table, except for the case when FFT_type = software, because no hardware resource is used.

If L2_CRMu receiver receives the metric FFT_type from L3_CRMu fft, it updates the value of R_left based on the table, and then compares R_left with the metric Resource in the table and decides if it needs a reconfiguration operation :

- If FFT_type = software, and R_left > Resource (hardware pipelined), the FFT should be implemented in hardware with pipelined architecture, L2_CRMu receiver sends reconfiguration command to its associated L2_ReMu receiver with the parameter FFT_type (=hardware pipelined). This decision will save 0.015W.

- If FFT_type = software and Resource (hardware radix-2) < R_left < Resource (hardware pipelined), L2_CRMu receiver sends reconfiguration command to its associated

L2_ReMu receiver with the parameter FFT_type (=hardware radix-2). This decision will save 0.023W.

- If FFT_type = software and R_left < Resource (hardware radix-2), there is not enough space in hardware, therefore it is not possible to implement the FFT in hardware.

- If FFT_type = hardware pipelined or hardware radix-2, regardless of the value of R_left, no change is needed, since the FFT is already in hardware.

L2_ReMu receiver

If the L2_ReMu receiver receives the command from L2_CRMu receiver, it then sends reconfiguration command to L3_ReMu fft.

L3_ReMu fft

L3_ReMu fft manages the reconfiguration of its associated PE FFT. If L3_ReMu fft receives command from its upper level manager L2_ReMu receiver, it then executes the command.

This management always tends to implement the FFT in hardware, not only because the FFT block is computationally intensive, it is a good choice to offload the FFT in hardware on PL to alleviate the workload of the PS, but also the hardware implementations have better performance and lower power consumption.

Because the processes run in background, this scenario is not easy to show visually as the modulation adaptation scenario, which can use constellation diagram as the demonstrator. This is what CR should normally do, to adapt to the changing environment, without human intervention. The processes are not visual on the surface but in background. The situation is similar for the following scenarios.

4.7 Scenario 3 : Management of FFT implementation type depending on the battery level

As described in the previous section, the FFT has three implementation options : Software, Hardware Pipelined, and Hardware Radix-2. The power consumption of both software FFT and hardware implementations of FFT can be measured at the time of

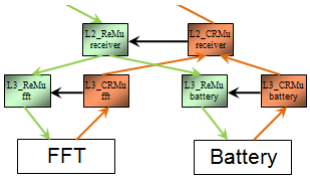


FIGURE 4.13 – Scenario 3.

design. This example shows the management of FFT implementation depending on the battery level. If the battery level is high, hardware pipelined architecture is used to achieve higher performance. If the battery level is low, hardware radix-2 architecture is chosen to save power.

The metrics involved are FFT_type, Battery Level, and Power Consumption (introduced in subsection 3.2.8). The hardware implementation of the FFT takes advantage of DPR.

L3_CRMu fft

The power consumption of FFT is managed by the L3_CRMu fft. In this study, we define :

- Power Consumption = high, if FFT_type = software ;
- Power Consumption = medium, if FFT_type = hardware pipelined ;
- Power Consumption = low, if FFT_type = hardware radix-2.

The metric Power Consumption is then sent to the upper level manager L2_CRMu receiver.

L3_CRMu battery

The PE Battery is a sensor monitoring the battery level. The metric Battery Level is managed by the L3_CRMu battery. Its value could be defined as below :

- Battery Level = high, if the battery level >= 60% ;
- Battery Level = medium, if 30% <= the battery level < 60% ;
- Battery Level = low, if the battery level < 30% ;

The metric Battery Level is then sent to the upper level manager L2_CRMu receiver.

L2_CRMu receiver

Based on the metric Battery Level received from L3_CRMu battery and metric Power Consumption received from L3_CRMu fft, L2_CRMu receiver makes decisions according to the battery levels :

- If Battery Level = high and FFT_type != hardware pipelined, higher performance can be achieved. L2_CRMu receiver sends reconfiguration command to its associated L2_ReMu receiver with the parameter FFT_type (=hardware pipelined).

- If Battery Level = low and FFT_type != hardware radix-2, the FFT should be implemented with low power consumption architecture. L2_CRMu receiver sends reconfiguration command to its associated L2_ReMu receiver with the parameter FFT_type (=hardware radix-2).

L2_ReMu receiver

If the L2_ReMu receiver receives the command from L2_CRMu receiver, it then sends reconfiguration command to L3_ReMu fft.

L3_ReMu fft

L3_ReMu fft manages the reconfiguration of its associated PE FFT. If L3_ReMu fft receives command from its upper level manager L2_ReMu receiver, it then executes the command. If FFT_type received from L2_ReMu receiver != current running FFT_type, L3_ReMu fft performs a partial reconfiguration operation to reconfigure the PE FFT.

4.8 Scenario 4 : Modify the FFT size according to the network/user order

This example shows the adaptation to a network order or a user decision, e.g., a user changes the standard from LTE to WIFI, or the network changes the channel bandwidth of LTE. This should be managed by L1_CRM.

For the sake of clarity, here, we take the network changing the channel bandwidth from 1.25MHz to 2.5MHz as an example. In this case, following the LTE standard, the FFT size should be changed from 128 to 256. Therefore, HDCRAM manages the reconfi-

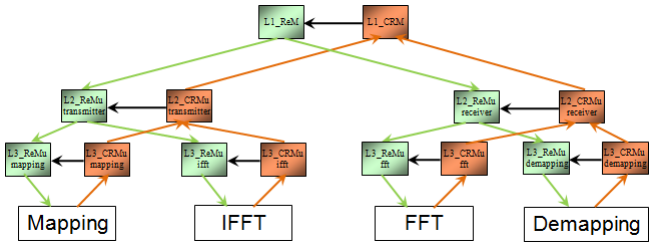


FIGURE 4.14 – Scenario 4.

guration of both the transmitter and receiver. The reconfiguration of the hardware FFT in the receiver is realized also by using dynamic partial reconfiguration technique.

L1_CRM

When the L1_CRM receives the network order and observes that the bandwidth has been changed, in order to adapt to this change, it sends reconfiguration command with the parameter FFT_size (=256) to its associated L1_ReM to reconfigure both the transmitter and the receiver to change the IFFT/FFT size from 128 to 256.

L1_ReM

If the L1_ReM receives the command from L1_CRM, it then sends the command to its target lower level managers, namely L2_ReMu transmitter and L2_ReMu receiver.

L2_ReMu transmitter

In this example, if the L2_ReMu transmitter receives the command from its upper level manager L1_ReM, it then takes action to execute the command and sends reconfiguration command to L3_ReMu ifft.

L3_ReMu ifft

L3_ReMu ifft manages the reconfiguration of its associated PE IFFT. If L3_ReMu ifft receives command from its upper level manager L2_ReMu transmitter, it then executes the command.

If IFFT_size received from L2_ReMu transmitter != current running IFFT_size, L3_ReMu ifft reconfigures the PE FFT to run at new IFFT_size.

L2_ReMu receiver

If the L2_ReMu receiver receives the command from its upper level manager L1_ReM, it then sends reconfiguration command to L3_ReMu fft.

L3_ReMu fft

L3_ReMu fft manages the reconfiguration of its associated PE FFT. If L3_ReMu fft receives command from its upper level manager L2_ReMu receiver, it then executes the command. If FFT_size received from L2_ReMu receiver != current running FFT_size, L3_ReMu fft performs a partial reconfiguration operation to reconfigure the PE FFT.

4.9 Scenario 5 : Merge them together

After the discussion of several scenarios, the involved metrics and their corresponding cognitive cycles, we would like to merge them together to show how the HDCRAM can easily manage all these scenarios.

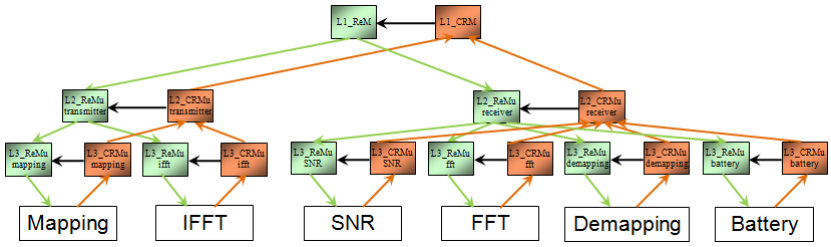


FIGURE 4.15 – Scenario 5.

L3_CRMus :

L3_CRMu mapping

The modulation scheme of PE Mapping is managed by the L3_CRMu mapping, and then is sent to the upper level manager L2_CRMu transmitter.

Metric : Modulation Scheme.

L3_CRMu ifft

The size of IFFT is managed by the L3_CRMu ifft, and then is sent to the upper level manager L2_CRMu transmitter. Metric : IFFT_size.

L3_CRMu SNR

The PE SNR is a channel condition sensor. The metric SNR is managed by the L3_CRMu SNR and then is sent to the upper level manager L2_CRMu receiver.

Metric : SNR.

L3_CRMu fft

When the receiver is implemented on Zynq platform, the PE FFT could be implemented either in software on PS or in hardware on PL. Therefore, in addition to FFT_size, it has some specific metrics.

Metrics :

- FFT_size
- FFT_type
- Power Consumption
- Resource. In this study, Resource and FFT_type are made one pair, and it is managed by L2_CRMu receiver as discussed in subsection 3.

The metrics are sent to the upper level manager L2_CRMu receiver.

L3_CRMu demapping

The demodulation scheme of PE Demapping is managed by the L3_CRMu demapping, and then is sent to the upper level manager L2_CRMu receiver.

Metric : Demodulation Scheme.

L3_CRMu battery

The PE Battery is a sensor monitoring the battery level. The metric Battery Level is managed by the L3_CRMu battery, and is then sent to the upper level manager L2_CRMu receiver.

Metric : Battery Level.

L2_CRMus :

L2_CRMu transmitter

L2_CRMu transmitter manages the metric Modulation Scheme received from L3_CRMu mapping and metric IFFT_size received from L3_CRMu ifft.

Metrics :

- Modulation Scheme (L3_CRMu mapping)
- IFFT_size (L3_CRMu ifft)

L2_CRMu receiver

L2_CRMu receiver manages the metrics received from its lower level L3_CRMus :

- Demodulation Scheme (L3_CRMu demapping)
- SNR (L3_CRMu SNR)
- FFT_size (L3_CRMu fft)
- FFT_type (L3_CRMu fft)
- Power Consumption (L3_CRMu fft)
- Resource
- Battery Level (L3_CRMu battery)

Based on these metrics, the L2_CRMu receiver makes a decision as discussed in previous subsections, and takes the following actions.

Actions :

- Request L1_CRM to change the Modulation Scheme
- Send a reconfiguration command to L2_ReMu receiver to change FFT_type
- Keep current status, no additional actions

The first action is based on the metric SNR received from L3_CRMu SNR to adapt the modulation scheme to the noise levels. But L2_CRMu receiver does not have the right

to change the modulation scheme of the transmitter, it has to hand over the right to L1_CRM.

Many metrics can result in the second action, which is a little bit more complex. The decisions are made as explained below.

- If FFT_type = software, and R_left > Resource (hardware pipelined), and Battery Level = high, the FFT should be implemented in hardware with pipelined architecture to achieve higher performance, L2_CRMu receiver sends reconfiguration command to its associated L2_ReMu receiver with the parameter FFT_type (=hardware pipelined).

- If FFT_type = software and Resource (hardware radix-2) < R_left < Resource (hardware pipelined), regardless of the Battery Level, L2_CRMu receiver sends reconfiguration command to its associated L2_ReMu receiver with the parameter FFT_type (=hardware radix-2). This is because there is not enough space to implement hardware pipelined architecture and the hardware radix-2 architecture consumes less power.

- If FFT_type = software and R_left < Resource (hardware radix-2), there is not enough space in hardware, therefore it is not possible to implement the FFT in hardware, L2_CRMu receiver decides to keep current status.

- If FFT_type = hardware pipelined and Battery Level = low, L2_CRMu receiver sends reconfiguration command to its associated L2_ReMu receiver with the parameter FFT_type (=hardware radix-2), because the hardware radix-2 architecture consumes less power.

- If FFT_type = hardware pipelined and Battery Level = high, L2_CRMu receiver decides to keep current status, because the hardware pipelined architecture provides higher performance.

- If FFT_type = hardware radix-2 and R_left > Resource and Battery Level = high, L2_CRMu receiver sends reconfiguration command to its associated L2_ReMu receiver with the parameter FFT_type (=hardware pipelined), because the hardware pipelined architecture provides higher performance.

- If FFT_type = hardware radix-2 and Battery Level = low, L2_CRMu receiver decides to keep current status, because the hardware radix-2 architecture consumes less power. For the sake of simplicity, this can be illustrated by a state machine in Figure 4.16.

Metrics submitted to L1_CRM :

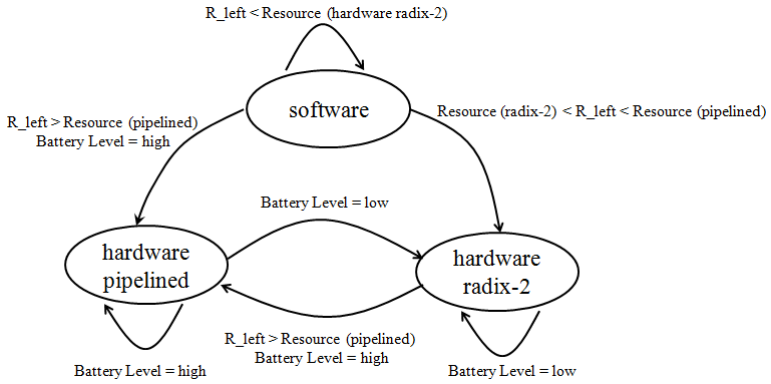


FIGURE 4.16 – The state machine representation of the management of FFT implementation.

- Demodulation Scheme
- FFT_size

L1_CRM

L1_CRM manages the metrics received from its lower level L2_CRMus and network/user order :

- Modulation Scheme (L2_CRMu transmitter)
- IFFT_size (L2_CRMu transmitter)
- Demodulation Scheme (L2_CRMu receiver)
- FFT_size (L2_CRMu receiver)
- network/user order

L1_CRM makes decisions based on the received metrics. For example, if L1_CRM receives the new demodulation scheme from L2_CRMu receiver, which means the channel condition has been changed, it sends reconfiguration command to its associated L1_ReM to reconfigure the modulation scheme of both the transmitter and the receiver. Or if L1_CRM receives the command from network to change the channel bandwidth, it sends reconfiguration command to its associated L1_ReM to reconfigure both the transmitter and the receiver to change the IFFT/FFT size.

L1_ReM

If the L1_ReM receives the command from L1_CRM, it then sends the command to its target lower level managers, namely L2_ReMu transmitter and L2_ReMu receiver.

L2_ReMus :

L2_ReMu transmitter

If the L2_ReMu transmitter receives the command from its upper level manager L1_ReM or from its associated L2_CRMu transmitter, it then takes action to execute the command and sends reconfiguration command to the target L3_ReMu.

L2_ReMu receiver If the L2_ReMu receiver receives the command from its upper level manager L1_ReM or from its associated L2_CRMu receiver, it then takes action to execute the command and sends reconfiguration command to the target L3_ReMu.

L3_ReMus :

L3_ReMu mapping

L3_ReMu mapping manages the reconfiguration of its associated PE Mapping. If L3_ReMu mapping receives command from its upper level manager L2_ReMu transmitter, it then executes the command. If the modulation scheme received from L2_ReMu transmitter != current running modulation scheme, L3_ReMu mapping reconfigures the PE Mapping to run at new modulation scheme.

L3_ReMu ifft

L3_ReMu ifft manages the reconfiguration of its associated PE IFFT. If L3_ReMu ifft receives command from its upper level manager L2_ReMu transmitter, it then executes the command. If IFFT_size received from L2_ReMu transmitter != current running IFFT_size, L3_ReMu ifft reconfigures the PE FFT to run at new IFFT_size.

L3_ReMu fft

L3_ReMu fft manages the reconfiguration of its associated PE FFT. If L3_ReMu fft receives command from its upper level manager L2_ReMu receiver, it then executes the command. If FFT_type received from L2_ReMu receiver != current running FFT_type,

L3_ReMu fft performs a partial reconfiguration operation to reconfigure the PE FFT. If FFT_size received from L2_ReMu receiver != current running FFT_size, L3_ReMu fft reconfigures the PE FFT to run at new FFT_size.

L3_ReMu demapping

L3_ReMu demapping manages the reconfiguration of its associated PE Demapping. If L3_ReMu mapping receives command from its upper level manager L2_ReMu receiver, it then executes the command. If the demodulation scheme received from L2_ReMu receiver != current running demodulation scheme, L3_ReMu demapping reconfigures the PE Demapping to run at new demodulation scheme.

From the above discussion, we can conclude that the HDCRAM is an open architecture and is extensible. We can easily add new metrics and update the decision engine in the CRMus. Also the decision making methods used in these examples are state machine like methods, more complex decision making algorithms can also be easily included in the CRMus.

We can also observe that the L2_CRMus can abstract the information from L3_CRMus, and only submit necessary information to the L1_CRM. In this example, the L1_CRM only manages the modulation scheme, FFT size, and network/user order, it does not care how the PE FFT is implemented. The L1_CRM does not need to know if the PE FFT is implemented in software, or in hardware. This feature is especially important when the HDCRAM is implemented on different heterogeneous platforms.

4.10 Conclusion

OFDM is a popular digital modulation method that is being used for many various standards in wireless communications. In this chapter, we employed a simplified OFDM system model, and introduced a management scenario of the OFDM transmitter and receiver. Especially, there are many choices to implement the PE FFT, which can be implemented on a PC, or on a Zynq platform either in software on PS or in hardware on PL. We can implement only the FFT on Zynq platform and all the rest on PC, in

this case, only part of level 3 management of the FFT needs to be implemented on Zynq platform to manage the reconfiguration of the FFT, and all other parts of the system are implemented on PC. But in order to show the efficiency of the HDCRAM, in this chapter, we chose to implement the receiver on Zynq platform, in this case, we have both a software PE Demapping and a PE FFT that can be either in software or hardware, therefore, a level 2 management is required to manage the 2 PEs. The transmitter and the level 1 management are put on PC.

Whether it is suitable or not, the OFDM scenario proposed in this chapter offers the possibility to glue almost all the aspects of the work introduced in this thesis.

In the OFDM scenario, we use some metrics described in chapter 3.

We take advantage of dynamic partial reconfiguration technique to reconfigure the hardware PE FFT. The DPR approach can not only reconfigure the FFT size but also the implementation architecture. As discussed in section 4.4, the DPR approach is advantageous over the traditional reconfigurable FFT in terms of resource utilization, performance (with the same architecture), power consumption and flexibility, except for the reconfiguration time. E.g., the DPR approach can win 0.014W (pipelined 2048 point FFT) to 0.039W (radix-2 128 point FFT) compared with the traditional reconfigurable FFT (pipelined 128 - 2048 point).

We employ the HDCRAM to manage all scenarios. The OFDM scenario is implemented on heterogeneous platforms that including a PC and a Zynq platform which is introduced in chapter 2. It shows that the HDCRAM can easily plan all scenarios presented in this chapter. HDCRAM can efficiently scale the management depending on the scenarios.

Chapter 5

Conclusions and Future Work

In this chapter, we conclude the work presented in this thesis and discuss some directions for future work.

5.1 Conclusions

The emergence of mobile Internet services and the rapid growth in the number of mobile subscribers result in the explosive growth of the mobile data traffic. As a result, the energy costs and the energy consumption are continuously rising, consequently leading to the increasing contribution to the carbon emission of the world. Therefore energy efficiency has drawn more and more attention.

Due to the ability to adapt its behavior to the changing environment, CR has been considered as an enabling technology for green radio communications. The cognitive cycle can be simplified into three essential parts : sensing, decision, and action. In order to efficiently manage these three parts, a management architecture, HDCRAM, has been proposed to glue the three parts together.

There are many different choices to implement HDCRAM. We can implement HDCRAM on a GPP all in software programmed in C++. But as its name implies, it is more interesting to implement HDCRAM on heterogeneous platforms. The communication between different platforms is through Ethernet using UDP protocol. This approach is flexible and efficient. Different devices do not have to be placed together very near to

each other. It makes the system scalable so that new devices can be added easily, and there is no need to change those devices that have already existed.

Originally, we have implemented HDCRAM on a PC and a Xilinx ML506 board. The level 1 management is unique and implemented on the PC side. Therefore, on the FPGA side, the highest level is level 2. We take advantage of the PR technique to manage the hardware PEs. Especially, we have developed a hardware UDP core, which works at 1Gbits/s, namely 125MBytes/s, to provide a high speed transmission of data and partial bitstreams. The downloading speed of partial bitstreams can reach 125Mbytes/s without considering the overhead, and nearly 120.6Mbytes/s taking into account the overhead of the headers of the Ethernet frame. The reconfiguration management can achieve the maximum theoretical throughput (400Mbytes/s) of the ICAP during partial reconfiguration.

But there are still some limitations. The software on Microblaze is standalone application without OS, and the codes are hardware dependent, thus hard to migrate. Besides, the power consumption of ML506 board is high. Therefore, when we have the Xilinx Zynq-7000 platform, which integrates a dual-core ARM Cortex-A9 as PS and a Xilinx's 7 series FPGA Artix-7 as PL in a single device, we decided to implement HDCRAM on the new platform. Because we can reuse most of the codes on PC. The codes are portable and run in Linux on ARM. It is easy to upgrade. And the power consumption of Zynq-7000 platform is much lower than ML506 board. Furthermore, Zynq-7000 platform supports both full and partial reconfiguration of the PL, which provides more flexibilities.

The dynamic partial reconfiguration technique is employed to reconfigure the hardware PEs, which makes the hardware PE some kind of software-like.

In order to efficiently manage the sensing information and the reconfiguration of a cognitive equipment, it is essential, first of all, to gather the necessary metrics of the PEs so as to provide enough information about the operating condition thus helping decision making. We have introduced some useful metrics on a FPGA platform. These metrics, obtained in the first place by L3_CRMus from the PEs, are then submitted to the upper level CRMus. Depending on the metrics, the CRMus learn about the environment and working state of the system, make decisions, and send reconfiguration orders to the associated ReMus, who execute the reconfiguration commands by means of a topdown

approach. And finally the L3_ReMus reconfigure the corresponding PEs so as to adapt to the environment.

Finally, we have shown a management scenario of a simplified OFDM system. Some metrics introduced in chapter 3 have been used in this example. According to the metrics obtained, how HDCRAM can efficiently manage the reconfiguration of the system to adapt to the changing environment, including the objective to save power consumption, has been explained. For example, in the scenario of HDCRAM management of automatic FFT implementation adaptation according to the battery level, about 20mW can be saved when the FFT implementation changes from pipelined 2048 to radix-2 2048. A demonstration of the OFDM scenario has been presented in an ETSI (European Telecommunications Standards Institute) workshop.

5.2 Future Work

There are still a lot of work to do. Some directions for further research are discussed below.

On Virtex 5 platform, we have developed a hardware UDP core. The data path from and to hardware PEs can be in hardware without passing through Microblaze, which provides a high speed transmission of data. But on Zynq-7000 platform, the default setup is that the Ethernet PHY is directly connected to the PS. The data are firstly transmitted to PS and then to PL by DMA approach. In order to improve the performance of data transmission when data are transmitted to or received from hardware PEs, we can develop a similar hardware UDP core on PL. Then depending on the UDP port, the soft UDP deals with the control signals and data to and from software PEs, and the hardware UDP handles the data to and from the hardware PEs.

In addition to the HDCRAM management architecture, we have also developed some PEs in the PE library, but in order to verify HDCRAM in more complex cognitive radio systems, it is not enough. Great efforts are needed to develop and add new PEs to the PE library. Therefore, it is better to find a way to reuse the existing open source libraries. But how to include and interface those open source PEs to HDCRAM architecture is also challenging.

In the OFDM example, the decision making methods in the CRMus are state machine like methods. As the decision part is consider to be the "heart" of a cognitive radio, more complex algorithms, such as machine learning [74], should be included in the CRMus.

High level modeling methodology to represent the HDCRAM for the management of complex systems and for those who do not want to involve in hardware details is also a promising research direction, which can use the Model Driven Architecture (MDA) [114, 115, 116]. The MDA approach can offer the ability to integrate tools to automatically generate codes, e.g., C++, VHDL. VHDL automatic generation tools (such as [117]) should be taken into account.

Appendix

Appendix A

Hardware UDP Core

A.1 Introduction

User Datagram Protocol (UDP) [118] is a connectionless protocol used for transport of data across an Internet Protocol (IP) [119] based network. UDP is an alternative to the Transmission Control Protocol (TCP) [120], but it does not perform handshaking as TCP does, or check for errors, or even to see if the transmitted data was received, so UDP is referred to as an unreliable, connectionless protocol. However, because UDP skips the handshaking and is focused on pure transmission, it has lower overhead and is faster than TCP.

Several related designs that implement hardware UDP protocol have been developed.

N. Alachiotis et al. [121] have designed an UDP/IP core for direct PC-FPGA communication. This design is only used for point-to-point communication. So, later, in their another paper [122], they have presented an improved version, which allows setting up a communication with any PC by sending three packets to the FPGA. But in these two designs, a communication must be started by a PC. Moreover, they do not include the ARP protocol. We expect both the FPGA and the PC can start a communication in our design. If we want to change the IP address of the FPGA, the ARP protocol is also necessary to set up a new communication.

A. Löfgren et al. point out that, when designing FPGA-based Ethernet connected embedded systems, the priority and necessity of requirements such as cost, area, flexibility etc. varies for each system [123]. Therefore, they present three different UDP/IP stack

cores for various demands. They classify them as “Minimum”, “Medium” and “Advanced” UDP/IP cores. The functionality of our design is at the level of “Advanced” as they described, but we do not implement the protocols such as RARP, ICMP and TCP.

A. Dollas et al. [124] have developed a TCP/IP core that also supports protocols such as ARP, ICMP and UDP. The TCP module works at 37.5 MHz, and the UDP module works at 77 MHz. The speed is not fast enough for our design.

So, we must develop our own UDP core. The design will be very complex if we want to control the Ethernet chip directly. If some modules are available and can be used as part of the design, the task would be greatly simplified.

Fortunately, Xilinx provides an Embedded Hard Tri-Mode Ethernet MAC (TEMAC) solution on the Virtex-5 device, which makes it easier to start developing Ethernet communication.

A.2 Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC Wrapper

We can generate the core by using the CORE Generator software. In this design, we use the SGMII interface. The generated wrapper includes an example design that has an address swap module as client logic. The example swaps the source and destination address of the incoming MAC frame and transmits it back to the source.

Physical Interface :

GMII / MII

The Media Independent Interface (MII), defined in IEEE 802.3, clause 22 is a parallel interface that connects a 10-Mb/s and/or 100-Mb/s capable MAC to the physical sublayers. The Gigabit Media Independent Interface (GMII), defined in IEEE 802.3, clause 35 is an extension of the MII used to connect a 1-Gb/s capable MAC to the physical sublayers. MII can be considered a subset of GMII, and as a result, GMII/MII together can carry Ethernet traffic at 10 Mb/s, 100 Mb/s, and 1 Gb/s.

SGMII

The Serial-GMII (SGMII) is an alternative interface to the GMII/MII that converts the parallel interface of the GMII into a serial format. It radically reduces the I/O count and

is, therefore, often favored by PCB designers. This configuration is achieved by connecting the Ethernet MAC to a RocketIO serial transceiver. SGMII can carry Ethernet traffic at 10 Mb/s, 100 Mb/s, and 1 Gb/s.

Figure A.1 illustrates the major functional blocks of the Ethernet MAC example design.

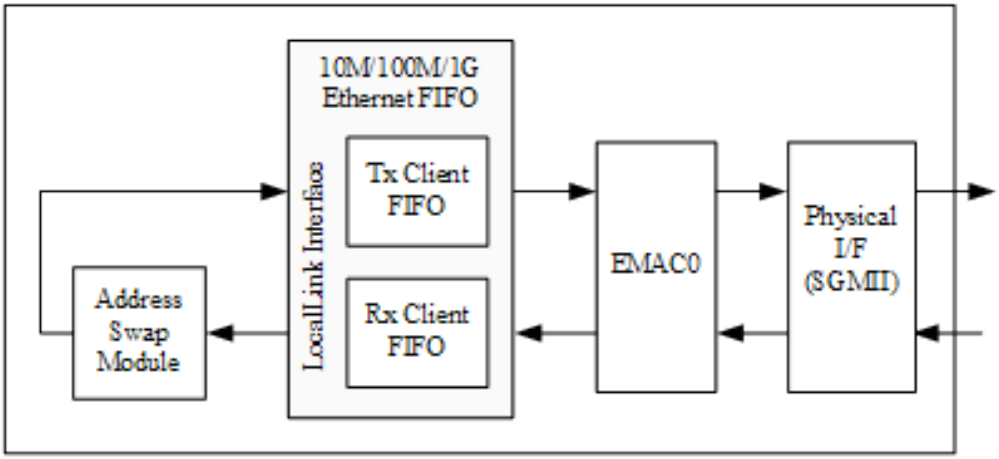


FIGURE A.1 – An example of management functionality.

Data is transferred on the LocalLink interface from source to destination, with the flow governed by the four active low control signals sof_n, eof_n, src_rdy_n, and dst_rdy_n. The flow of data is controlled by the src_rdy_n and dst_rdy_n signals. The individual packet boundaries are marked by the sof_n and eof_n signals. Only when these signals are asserted simultaneously is data transferred from source to destination.

Figure A.2 shows the transfer of an 8-byte frame.

The Address Swap module represents the back-end client logic user application. What we should do next is to replace this module with our own design. Since we are going to implement UDP protocol on the Virtex-5 device to communicate with a PC, we should replace this Address Swap module with an UDP core, which will be connected directly to the local link FIFOs of the Ethernet MAC wrapper. Figure A.3 shows how the UDP core is connected to the Ethernet MAC wrapper.

The Tri-Mode Ethernet MAC wrapper makes it easier for users to develop with the Ethernet communication by giving the user simplified inputs and outputs interfaces.

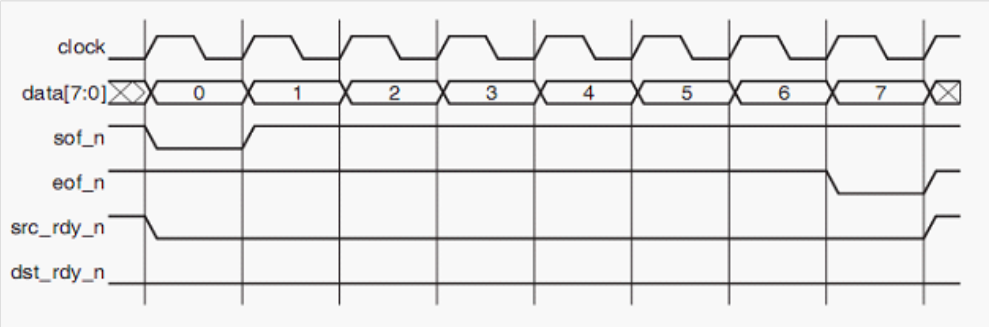


FIGURE A.2 – An example of management functionality.

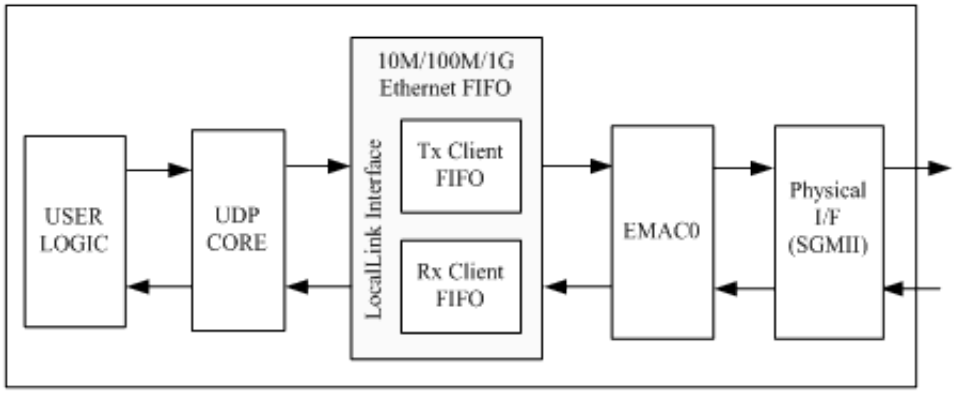


FIGURE A.3 – An example of management functionality.

A.3 UDP module

The format of an UDP packet is shown in Figure A.4. It consists of header and user data. The header is made up of three parts : Ethernet header, IP header and UDP header.

The UDP module is composed of two parts : UDP Receiver and UDP Transmitter. Each module mainly uses a finite state machine (FSM).

A.3.1 UDP Receiver

UDP Receiver is connected directly to the Local Link receive FIFO of the Ethernet MAC Wrapper.

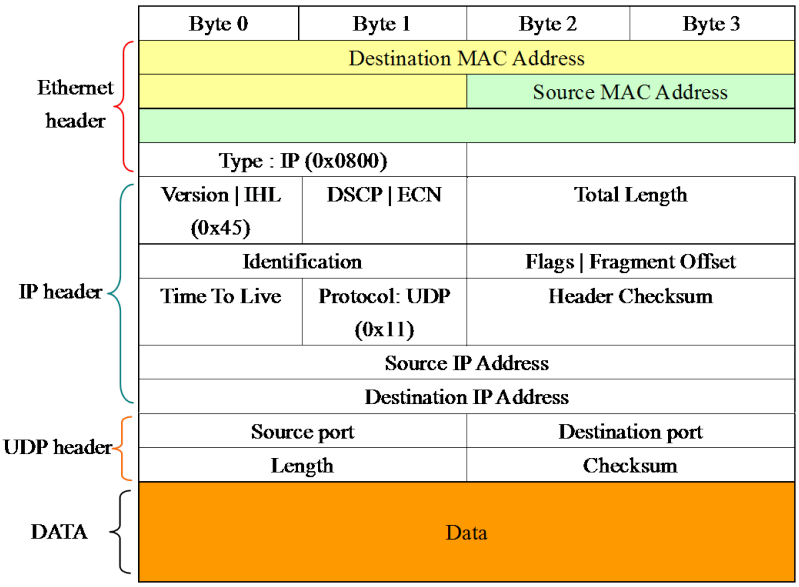


FIGURE A.4 – An example of management functionality.

When an active-low signal sof_n arrives, a new packet is coming, the counter begins to calculate the number of received bytes, and at the same time, the Receiver starts to read the data (including the header) from the Local Link receive FIFO, and a FSM is activated to decode each field of the packet's header. Some filters are used to check the header of the packet, in order to make sure that the user data will be received are what we indeed want.

When the FSM goes to Ethernet header state, if the type field in the Ethernet header is not 0x0800, which means the packet being received is not an IP packet, then it will be ignored, and the FSM jumps back to idle state waiting for next packet. Otherwise the source MAC address is stored in a register and transmitted to user logic, and the FSM goes to next state, IP header state.

In this state, three filters are applied, if the packet is not an IPv4 packet (the version field of the IP header is not the value of 4), or is not an UDP packet (the protocol field is not the value of 0x11), or the destination IP address does not equal the one set on the FPGA device, it will be discarded. If the IP header is what we are expecting, the source

IP address will be stored in a register and passed up to user logic, and then the FSM shifts to UDP header state.

In the UDP header state, the source port, the destination port and the data length will be stored in three different registers and delivered up to user logic. Since the length field of UDP header includes the length of the UDP header, we should subtract the UDP header length to obtain the correct data length.

The following state is read data state, a signal is asserted to inform the user logic that it's time to receive data, and the incoming user data are transferred to user logic for further processing. When the signal eof_n goes low, this indicates the end of the packet, then the FSM shifts to idle state waiting for next packet.

Now, a complete packet has been received ; some useful signals, source MAC address, source IP address, source port, destination port and data length, are passed up to user logic ; and the user data are transferred to user logic as well. Of course, in the receive process, we can add other filters we want depending on the actual needs, such as port number filter, etc.

A.3.2 UDP Transmitter

UDP Transmitter is connected directly to the Local Link transmit FIFO of the Ethernet MAC Wrapper.

Three active-low signals, sof_n, eof_n, and src_rdy_n must be created appropriately at the right time to control the transmission.

When the Transmitter detects an active-high pulse tx_start, the FSM goes to Ethernet header state, and the Ethernet header is being sent. At the very beginning of this state, the signal sof_n should be set to low for only one cycle, and the signal src_rdy_n should be set to low. The destination MAC address field is loaded from the Receiver (the source MAC address received), or can also be read from user logic. The source MAC address field is a constant signal set on the FPGA device. The type field is 0x0800, since this is an IP packet.

The next state is IP header state, the fist byte being sent is the version and internet header length (the number of 32-bit words in the header) field. The value of this field is 0x45, because the packet is IPv4 and the header length is 20 bytes. The total length field

has the value of user data length (reads from user logic) + 20 (IP header length) + 8 (UDP header length). The time to live field is set to 0x80, which is the default value. The protocol field is 0x11, because the packet follows UDP protocol. The source IP address field is a constant signal set on the FPGA device. The destination IP address field is loaded from the Receiver (the source IP address received), or can also be read from user logic. Since we know the values of all fields of the IP header, the value of header checksum field can be calculated before the time to send the header checksum. Other fields in the IP header are all set to 0.

The following state is UDP header state. The source port field and the destination port field read from the user logic. The length field has the value of user data length (reads from user logic) + 8 (UDP header length). The checksum field is set to 0. At the time of the last byte of the UDP header, a signal is asserted to inform the user logic that it's time to send user data and prepare the right data that are going to be sent.

Then the FSM shifts to send data state, reads data from user logic and sends these data to Ethernet MAC Wrapper byte by byte. At the time of the last byte of user data, the signal eof_n should be set to low for only one cycle.

The final state is finish transmit state, the signal src_rdy_n is set to high. Then the FSM goes to idle state waiting for another transmission.

A.3.3 UDP module test

In order to examine if this UDP module works properly, a test work has been done. On the PC side, two VLC processes were employed. One was used to send video streams that follow UDP protocol while the other was used to receive video from a certain port. On the FPGA side, an application was created and used to just receive packets from PC and send the incoming data back to PC at a certain port. Wireshark was used to capture packets sent and received on the PC side.

But what we captured in Wireshark were only ARP packets sent by PC. We typed "arp -a" in the command shell, and did not find any information about FPGA. The problem was that there was not a record of FPGA in the ARP table. So PC sent ARP packets asking "who has 192.168.10.5(IP address of FPGA)". We had to add a record manually

by typing "arp -s 192.168.10.5 00-0A-35-01-93-01". After doing this, we finally received the video, and it worked well.

Up to this point, it seemed that the work should be finished. If it is only used for point-to-point communication and without changing IP address, it is sufficient to build up an UDP communication. But if we want to change the IP address of FPGA, this will not be flexible. Actually, in some cases, the IP address needs to be configurable in our design. So an ARP module has to be added into the UDP core.

A.4 ARP module

Address Resolution Protocol (ARP) is responsible for resolution of IP addresses into physical addresses. Essentially, ARP is a table with a list of the IP addresses and their corresponding physical addresses. The Address Resolution Protocol uses a simple message format that contains one address resolution request or response. The format of an ARP packet is shown in Figure A.5. It consists of Ethernet header and ARP header.

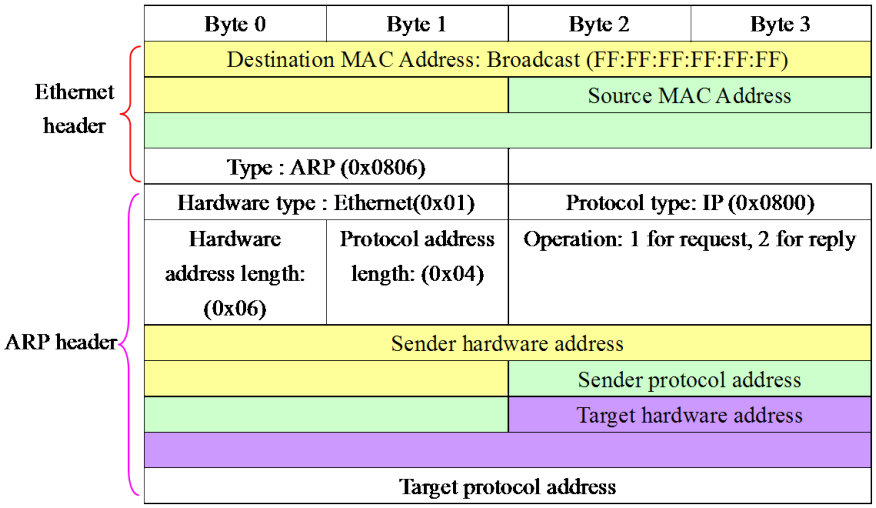


FIGURE A.5 – The format of an ARP packet.

Since we decide to add ARP module into the UDP core, some modification should be done accordingly in the UDP Transmitter module. When we send an UDP packet, we must read the lookup table in the ARP module first. If there exists the record of corresponding MAC address of the destination IP address, we need only read this information and add it

into the UDP Transmitter module, then start an UDP transmission. Otherwise, it needs to send an ARP request packet and wait for the response from the remote machine, and then the ARP module stores the information in the lookup table and tells the UDP Transmitter module to read this information. Two FSMs are used in the ARP module, one is receive FSM, and the other one is transmit FSM.

A.4.1 ARP Receiver

When the signal sof_n goes low, the receive FSM is activated to decode each field of the packet. When the receive FSM goes to Ethernet header state, if the type field in the Ethernet header is 0x0806, which means the packet being received is an ARP packet, and then the receive FSM shifts to next state, read type state.

The read type state, examines if it is a packet for Ethernet (the hardware type field is 0x0001) and IPv4 (the protocol type field is 0x0800).

The next state, read length state, checks hardware length (0x06) and protocol length (0x04).

The following state is operation type state, reads the operation field of the ARP packet to learn it is a request (0x0001) or a reply (0x0002).

Then the receive FSM goes to read sender state, the sender hardware address and the sender protocol address are stored.

The next state is read target state, if the target protocol address field does not equal the IP address set on the FPGA device, the receive FSM jumps to idle state, otherwise, the receive FSM shifts to process state.

In the process state, the sender hardware address and the sender protocol address are added to the ARP lookup table. If this ARP packet received is a request message, then the transmit FSM is activated to send a reply packet to the sender.

A.4.2 ARP Transmitter

There are two situations that can trigger the transmit FSM to start an ARP transmission.

* If FPGA received a request ARP packet, it needs to send a reply packet to the sender, which was mentioned above in subsection ARP Receiver.

* When the UDP Transmitter module reads the lookup table in the ARP module, and if it does not have the record of corresponding MAC address of the destination IP address, then it needs to send an ARP request packet.

When an ARP transmission starts, the transmit FSM goes to Ethernet header state. The destination MAC address field is 0xFFFFFFFFFFFF, which is a broadcast. The source MAC address field is a constant signal set on the FPGA device. The type field is 0x0806, since this is an ARP packet.

The next state is send type state, the hardware type field is 0x0001(Ethernet) and the protocol type field is 0x0800(IPv4).

Then the transmit FSM shifts to send length state, the hardware length field is 0x06 and protocol length field is 0x04.

The following state is send operation state. The operation field is 0x0001(request), if this transmission is triggered by UDP Transmitter module, or 0x0002(reply) if triggered by receive FSM.

Then the transmit FSM goes to send sender state, the sender hardware address field and the sender protocol address field both are constant signals set on the FPGA device, or can read from the user logic. The final state is send target state, if this is a request packet, the target hardware address field is all zeros, because this is just what we want to know, and the target protocol address reads from the UDP Transmitter module. If this is a reply packet, the target hardware address field and the target protocol address field read from the ARP lookup table.

A.5 Architecture

Because there is only one Ethernet MAC interface, it can not send an UDP packet and an ARP packet at the same time; only one of them at a time can be transmitted. So a multiplexer is employed to handle this issue, and it is set to send UDP packet by default, because most of the packets are UDP packets, an ARP packet is only needed at the beginning of a transmission to establish a communication between the FPGA and the PC. Finally, the architecture of our UDP core is shown in Figure A.6.

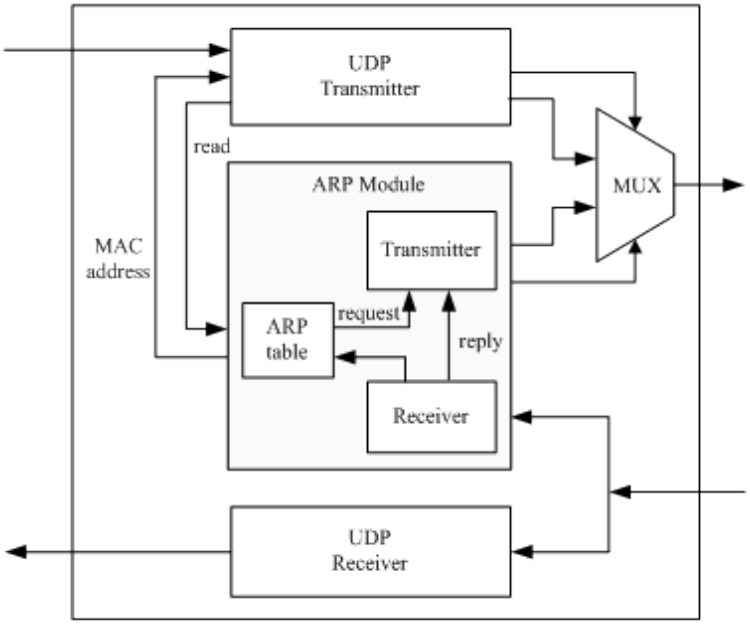


FIGURE A.6 – The architecture of the UDP core.

A.6 Test and validation

In order to examine if this UDP core works properly, we should set up a test platform first. An application, which is mainly a FIFO, is developed and used to just receive UDP packets from PC and then send the incoming data back to PC at a certain destination port. We call it Video stream. A soft core Microblaze is created as the management unit. In our design, we need to distinguish a control message from a process message depending on the destination port number of the received packet. If the port number is 1200, we consider the packet as a control message, and the received data must be sent to Microblaze. In the Microblaze, these control data are decoded, and corresponding processing must be done. If it is a change IP address command or a change port number command, the Microblaze sends configure signal and new IP address or new port number to the configure module. This module then sends new values received to the UDP core to replace the old ones.

The schematic diagram of the test platform is shown in Figure A.7. On the PC side, two VLC processes are executed. One is used to send video streams that follow UDP protocol while the other is used to receive video streams from a certain port. We use

Wireshark to capture all the packets sent and received on the PC side. We can compare the packets sent with the packets received to see if the data are the same. UDP Test Tool is used to send control packets at port 1200.

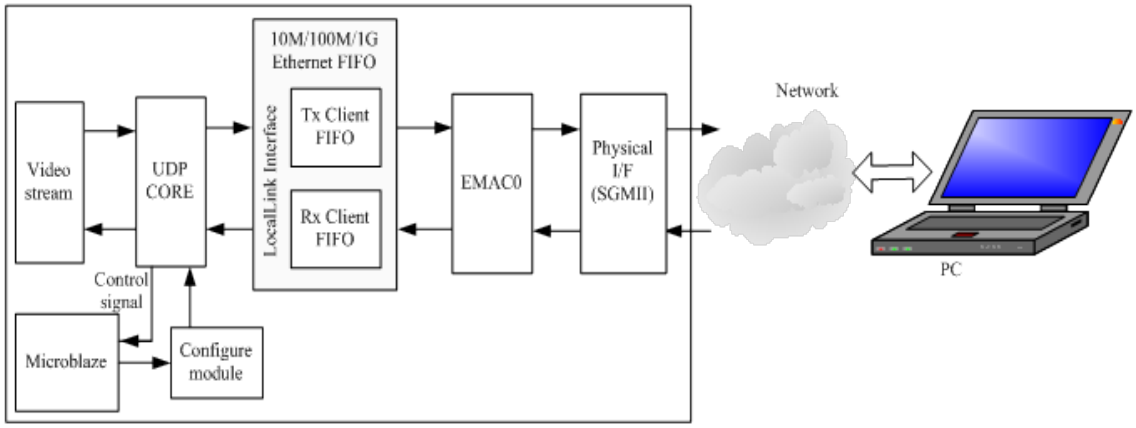


FIGURE A.7 – The schematic diagram of the test platform.

We can see the packets captured by Wireshark in Figure A.8, the fist two packets are ARP packets, they establish a communication between the PC and the FPGA. And then every packet sent by VLC is followed by a packet received from the FPGA, they are a pair. We can compare the data of every pair of packets, and we can find that they are the same. This verifies that the data transmission is correct and reliable.

Considering the Ethernet MAC wrapper works at 125 MHz and the UDP core has the same clock, it can send or receive one byte per cycle, and then we can estimate the maximum throughput : 125 Mbytes/s.

We can also see the video streams received from FPGA in figure A.9, the VLC sender is on the left side and the VLC receiver is on the right side. The results show that the UDP core works well. The data are transmitted reliably.

Next, we test if the UDP core is configurable. We define the control message format as shown in following tables.

TABLE A.1 – Set to default command.

00	00	00

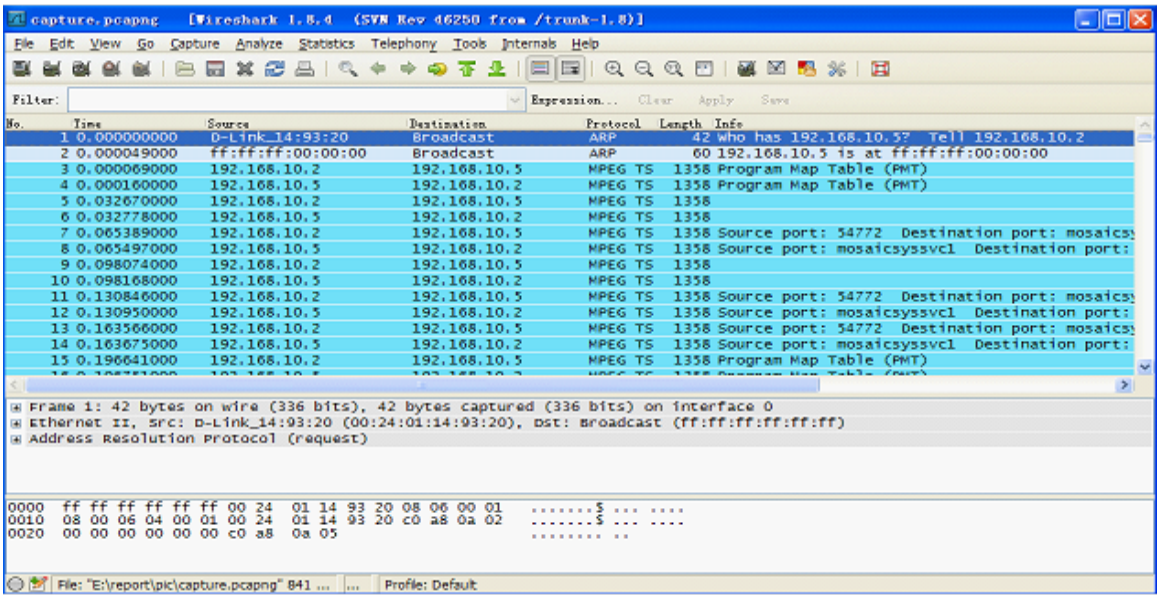


FIGURE A.8 – The packets captured in Wireshark.

FIGURE A.9 – The video streams sent and received.

TABLE A.2 – Change destination port number command.

01	Port number byte1	Port number byte0

TABLE A.3 – Change IP address command.

02	IP address byte3	IP address byte2	IP address byte1	IP address byte0

TABLE A.4 – Change data length command.

03	length byte1	length byte0

The default values of UDP core are shown as follows : Destination port number : 1234 ; IP address : 192.168.10.5 ; Data length : data length of received packet.

We send change destination port number command "01 04 d3" at port 1200 by using UDP Test Tool. This command changes the destination port number from 1234 to 1235(0x04d3). After this change, the VLC receiver cannot receive the packets, and the video frame stops. Then we change the receive port of the VLC receiver to 1235, and the video works again. This proves that we have changed the destination port number successfully.

Then, we send change IP address command "02 C0 A8 0A 08". The IP address changes from 192.168.10.5 to 192.168.10.8(0x C0 A8 0A 08). After doing this change, the FPGA cannot receive the packets from the VLC sender and the VLC receiver cannot receive the packets as well, because VLC sender sends video streams to 192.168.10.5. We change the destination IP address of the VLC sender to 192.168.10.8, and then the VLC receiver receives the video again. This proves that we have changed the IP address successfully.

Similarly, we test change data length command and set to default command respectively, both work properly. These tests show that the UDP core is configurable.

A.7 Conclusion

A hardware UDP core has been developed on Xilinx ML506 board in order to be able to communicate with PC following UDP protocol. ARP protocol is also implemented to make it flexible to build up a new communication. This design is connected directly to the local link FIFOs of the Ethernet MAC wrapper, and is configurable by using a Microblaze processor. In this design, control messages are distinguished from process messages depending on the destination port number of the received packet. Control messages must

be sent to Microblaze following the HDCRAM architecture. According to the test results, our design works well and meets the requirement.

In this design, the values of UDP core can be configured not only by Microblaze, but also by PC by sending command packets at port 1200. It can also work even without Microblaze by using default values. So, it is very flexible.

Appendix B

ML506 Evaluation Platform

The ML505, ML506, and ML507 platforms use the same printed-circuit board (PCB) [125]. The ML506 evaluation platform enables designers to investigate and experiment with features of Virtex-5 FPGAs. This appendix describes the features of the ML506 Evaluation Platform. For the detailed information and user guide for the Virtex-5 architecture, please refer to [126]. The picture of the ML506 evaluation board is shown in Figure B.1.

Features

- Xilinx Virtex-5 FPGA XC5VSX50T-1FFG1136

- Two Xilinx XCF32P Platform Flash PROMs (32 Mb each) for storing large device configurations

- Xilinx System ACE. CompactFlash configuration controller with Type I CompactFlash connector

- Xilinx XC95144XL CPLD for glue logic

- 64-bit wide, 256-MB DDR2 small outline DIMM (SODIMM), compatible with EDK supported IP and software drivers

- Clocking

* Programmable system clock generator chip

* One open 3.3V clock oscillator socket

* External clocking via SMAs (two differential pairs)

- General purpose DIP switches (8), LEDs (8), pushbuttons, and rotary encoder

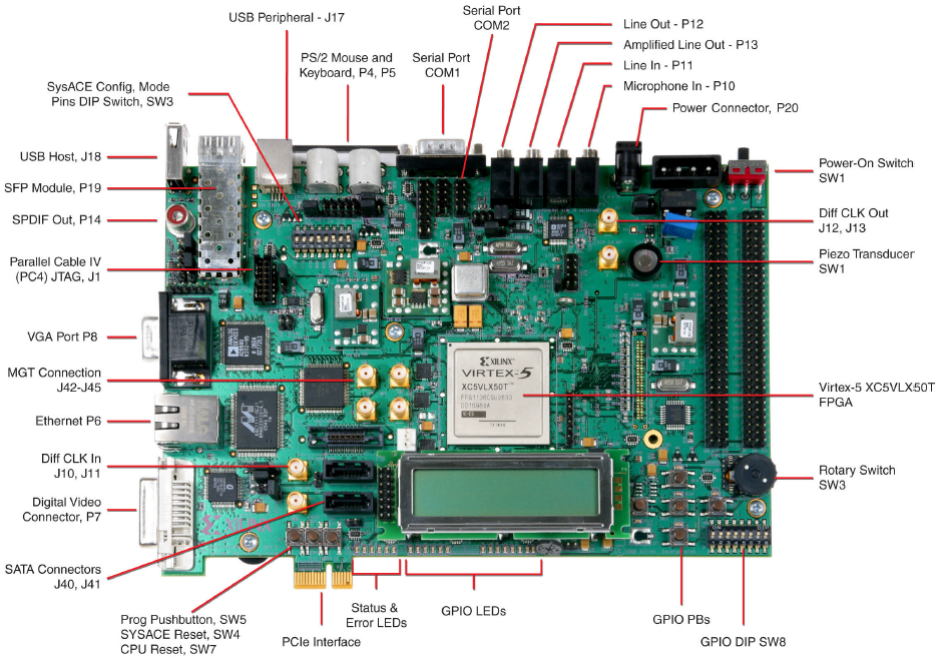


FIGURE B.1 – ML506 evaluation board. [127]

- Expansion header with 32 single-ended I/O, 16 LVDS-capable differential pairs, 14 spare I/Os shared with buttons and LEDs, power, JTAG chain expansion capability, and IIC bus expansion

- Stereo AC97 audio codec with line-in, line-out, 50-mW headphone, microphone-in jacks, SPDIF digital audio jacks, and piezo audio transducer

- RS-232 serial port, DB9 and header for second serial port

- 16-character x 2-line LCD display

- One 8-Kb IIC EEPROM and other IIC capable devices

- PS/2 mouse and keyboard connectors

- Video input/output

* Video input (VGA)

* Video output DVI connector (VGA supported with included adapter)

- ZBT synchronous SRAM, 9 Mb on 32-bit data bus with four parity bits

- Intel P30 StrataFlash linear flash chip (32 MB)

- Serial Peripheral Interface (SPI) flash (2 MB)

- 10/100/1000 tri-speed Ethernet PHY transceiver and RJ-45 with support for MII, GMII, RGMII, and SGMII Ethernet PHY interfaces

- USB interface chip with host and peripheral ports

- Rechargeable lithium battery to hold FPGA encryption keys

- JTAG configuration port for use with Parallel Cable III, Parallel Cable IV, or Platform USB download cable

- Onboard power supplies for all necessary voltages

- Temperature and voltage monitoring chip with fan controller

- 5V @ 6A AC adapter

- Power indicator LED

- MII, GMII, RGMII, and SGMII Ethernet PHY Interfaces

- GTP/GTX : SFP (1000Base-X)

- GTP/GTX : SMA (RX and TX Differential Pairs)

- GTP/GTX : SGMII

- GTP/GTX : PCI Express (PCIe) edge connector (x1 Endpoint)

- GTP/GTX : SATA (dual host connections) with loopback cable GTP/GTX : Clock synthesis ICs

- Mictor trace port

- BDM debug port

- Soft touch port

- System monitor

Appendix C

ZC702 Evaluation Board

The ZC702 evaluation board for the XC7Z020 All Programmable SoC (AP SoC) provides a hardware environment for developing and evaluating designs targeting the Zynq-7000 XC7Z020-1CLG484C AP SoC. The ZC702 board provides features common to many embedded processing systems, including DDR3 component memory, a tri-mode Ethernet PHY, general purpose I/O, and two UART interfaces. Other features can be supported using VITA-57 FPGA mezzanine cards (FMC) attached to either of two low pin count (LPC) FMC connectors.

The ZC702 board key features are listed in here and indicated in Figure C.1. For more detailed information about the ZC702 board, please refer to [89].

Key Features [129] :

Zynq-7000 XC7Z020 CLG484 -1

- ROHS compliant ZC702 kit including the XC7Z020-CLG484-1 AP SoC Power 12V wall adapter or ATX voltage and current measurement capability of supplies

Configuration

- Onboard configuration circuitry
- 16MB Quad SPI Flash
- SDIO Card Interface (boot)
- PC4 and 20 pin JTAG ports

Memory

- DDR3 Component Memory 1GB

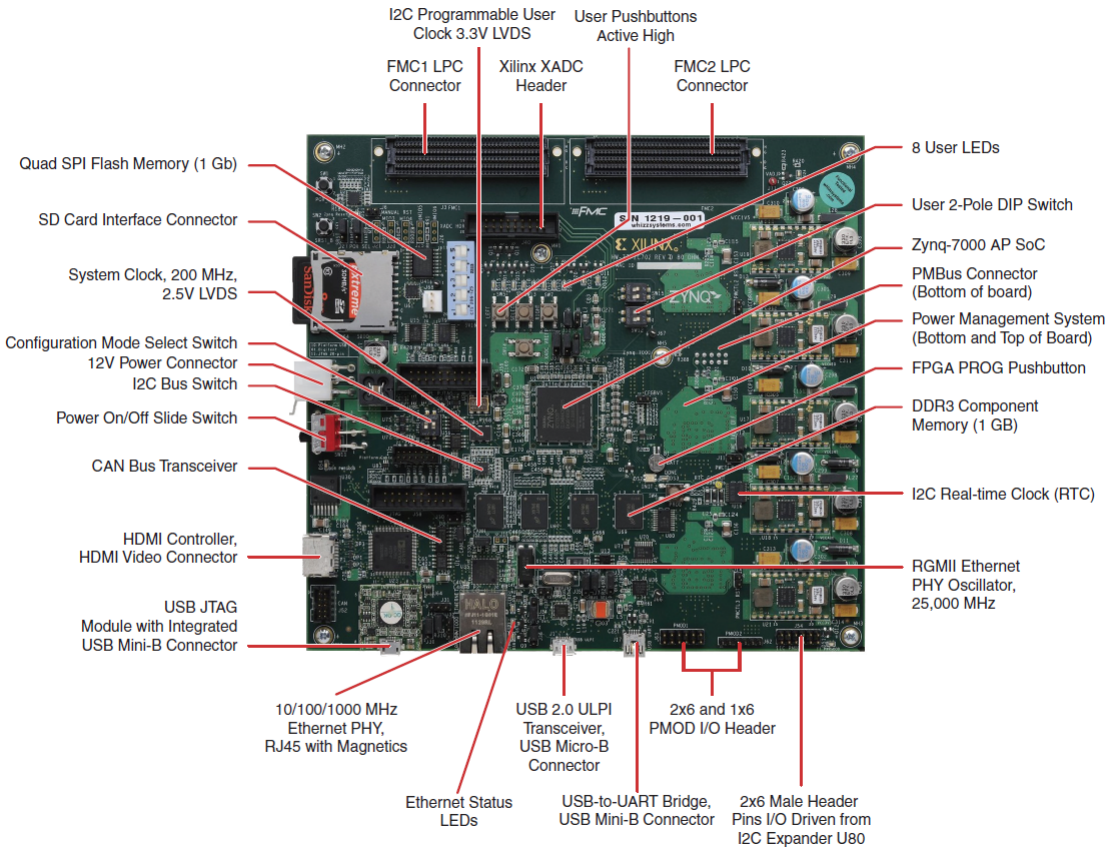


FIGURE C.1 – Features on the ZC702 board. [128]

- Support 32 data width
- 16MB Quad SPI Flash
- IIC - 1 KB EEPROM

Communication & Networking

- Gigabit Ethernet GMII, RGMII and SGMII
- USB OTG 1 (PS) - Host USB
- IIC Bus Headers/HUB (PS)
- 1 CAN with Wake on CAN (PS)
- USB UART (PS)

Video/ Display

- HDMI Video OUT
- 8X LEDs

Expansion Connectors

- FMC #1-LPC connector (0 GTX Transceiver, 68 single-ended or 34 differential user defined signals)

- FMC #2-LPC connector (0 GTX Transceiver, 68 single-ended or 34 differential user defined signals)

- IIC HUB/Expander

- Dual Pmod (8 I/O Shared with LED's)

- Single Pmod (4 I/O Shared with PJTAG)

Clocking

- 200MHz Fixed PL Oscillator (Differential LVDS)

- 156.25MHz (default) I2C Programmable Oscillator (Differential LVDS)

- 33.33MHz Fixed PS System Oscillator (Single-Ended CMOS)

Control & I/O

- 3 User Push Buttons

- 2 User Switches

- 8 User LEDs

Power

- 12V wall adapter or ATX

- Voltage and Current measurement capability of supplies

Analog

- XADC header

C.1 Zynq-7000 AP SoC architecture

The Xilinx Zynq-7000 All Programmable SoC (AP SoC) architecture integrates a feature-rich dual-core ARM Cortex-A9 based processing system (PS) and Xilinx 7-series FPGA fabric as the programmable logic (PL) in a single device. The PS and PL can be used independently or together. A system can be implemented by mapping custom software on PS and custom logic on PL respectively.

The block diagram of the Zynq-7000 AP SoC architecture is shown in Figure C.2.

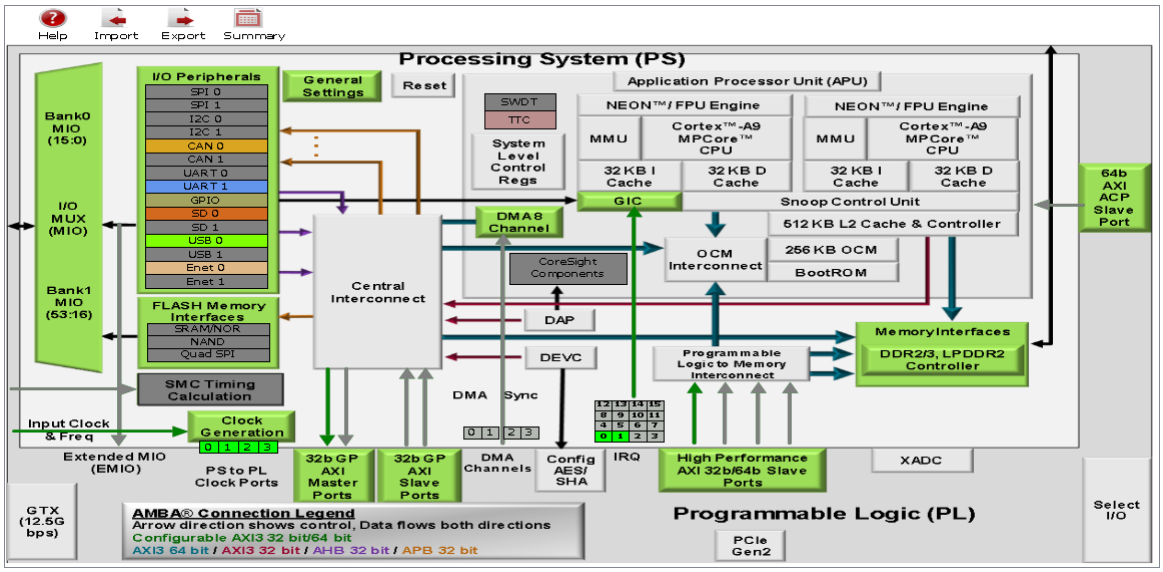


FIGURE C.2 – Overview of the Zynq-7000 AP SoC architecture.

The PS includes a dual-core ARM Cortex-A9 processor, on-chip memory, external memory interfaces, and a rich set of I/O peripherals. The PS and PL can be tightly or loosely coupled using multiple interfaces. This enables the designer to effectively integrate user-created hardware accelerators and other functions in the PL logic that are accessible to the processors and can also access memory resources in the PS. The custom applications designed on PL can take advantage of partial reconfiguration technique, which allows the reuse of the PL by reconfiguring a portion of the PL.

The use of an embedded Linux can reduce the development time and cost. Applications in Linux are easy to migrate to new processing platforms.

Xilinx Zynq-Linux is an open source OS freely available from Xilinx. It is based on the 3.0 Linux kernel from kernel.org and includes a number of additions from Xilinx, such as a BSP and specific device drivers. Figure C.3 illustrates the Linux System architecture.

C.2 Boot Stages

You can boot or configure Zynq-7000 All Programmable SoC devices in secure mode using static memories only (JTAG disabled) or in non-secure mode using either JTAG or static memories.

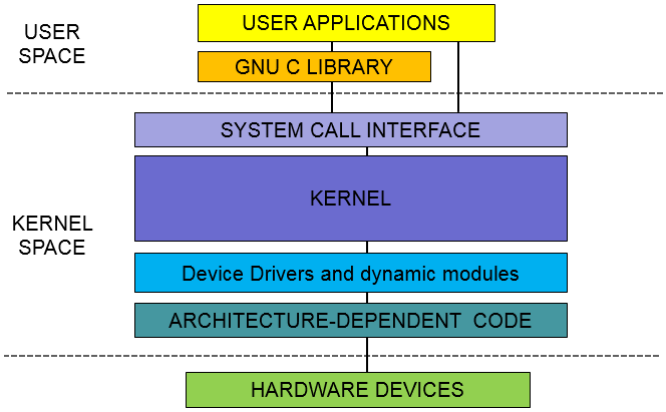


FIGURE C.3 – A High-level GNU/Linux system architecture.

Zynq Linux boot process is depicted in Figure C.4.

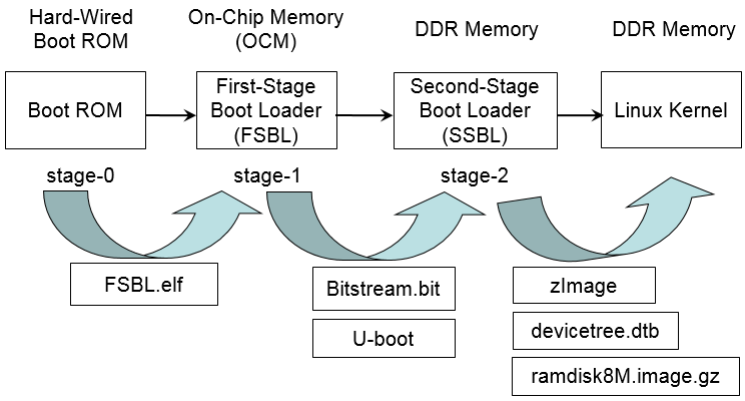


FIGURE C.4 – Zynq Linux boot process.

C.2.1 Stage-0 Boot (BootROM)

An internal BootROM stores the stage-0 boot code, which configures one of the ARMÂ® processors and the necessary peripherals to start fetching the First Stage Boot-loader (FSBL) boot code from one of the boot devices. The programmable logic (PL) is not configured by the BootROM. The BootROM is not writable.

The FSBL boot code is typically stored in one of the flash memories, or can be down-loaded through JTAG. BootROM code copies the FSBL boot code from the chosen flash

memory to On-Chip Memory (OCM). The size of the FSBL loaded into OCM is limited to 192 kilobyte. The full 256 kilobyte is available after the FSBL begins executing.

C.2.2 Stage-1 (First-Stage Bootloader)

The FSBL is loaded into the OCM by the boot ROM after the initial boot period. The FSBL is responsible for a number of initialisation functions which include initialising the CPU with the PS configuration data, programming the PL using a bitstream, loading thesecond-stage bootloader or initial user applications into memory, and beginning execution of the second-stage bootloader/initial user application code.

C.2.3 Stage-2 (Second-Stage Bootloader)

If using Linux, the SSBL is U-Boot and it is responsible for loading the compressed Linux kernel image, the system device tree and the ramdisk image into memory. Once these images are loaded into memory, U-Boot will initialise the execution of the Linux kernel.

Appendix D

FFT implementation architectures

In chapter 4, the hardware implementation of FFT uses Xilinx FFT IP core [112], which implements the Cooley-Tukey [130] FFT algorithm, a computationally efficient method for calculating the Discrete Fourier Transform (DFT).

The FFT core uses the Radix-4 and Radix-2 decompositions for computing the DFT. When using Radix-4 decomposition, the N-point FFT consists of $log_4(N)$ stages, with each stage containing N/4 Radix-4 butterflies. An N-point FFT using Radix-2 decomposition has $log_2(N)$ stages, with each stage containing N/2 Radix-2 butterflies.

The FFT core provide four architecture options : Pipelined Streaming I/O, Radix-4 Burst I/O, Radix-2 Burst I/O, and Radix-2 Lite, and support point sizes from 8 to 65536.

We would like to introduce here two of them used in chapter 4. For the detailed introduction and user guide of the Xilinx FFT IP core, please refer to [112].

D.1 Pipelined Streaming I/O

The block diagram of the pipelined architecture is shown in Figure D.1.

The Pipelined Streaming I/O architecture employs several Radix-2 butterfly processing engines to offer continuous data processing. Each processing engine has its own memory banks to store the input and intermediate data (Figure D.1). The core has the ability to simultaneously perform transform calculations on the current frame of data, load input data for the next frame of data, and unload the results of the previous frame of data. You can continuously stream in data and, after the calculation latency, can continuously

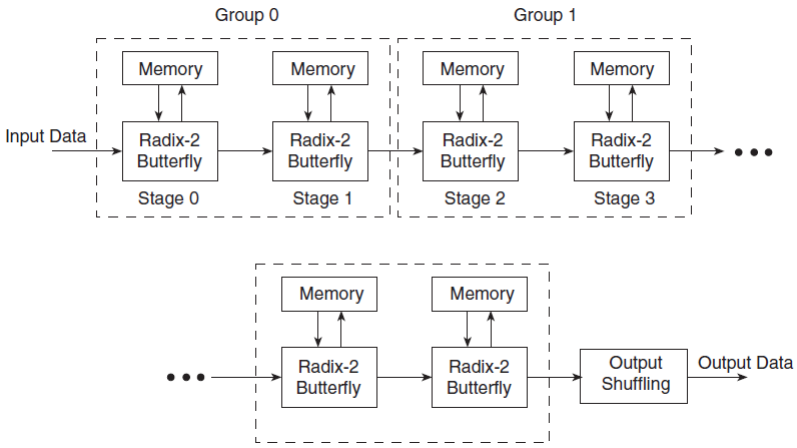


FIGURE D.1 – Pipelined architecture. [112]

unload the results. If preferred, this design can also calculate one frame by itself or frames with gaps in between.

This architecture covers point sizes from 8 to 65536. You have the flexibility to select the number of stages to use block RAM for data and phase factor storage. The remaining stages use distributed memory.

D.2 Radix-2 Burst I/O

The block diagram of the Radix-2 architecture is shown in Figure D.2.

The Radix-2 Burst I/O solution uses one Radix-2 butterfly processing engine. After a frame of data is loaded, the input data stream must halt until the transform calculation is completed.

The Burst I/O architectures do not allow frame overlapping to the same degree as the Pipelined Streaming I/O architecture. It loads and processes data separately, using an iterative approach. It is smaller in size than the pipelined solution, but has a longer transform time.

When the FFT is started, the data is loaded. After a full frame has been loaded, the core computes the transform. When the computation has finished, the data can be unloaded, but cannot be loaded or unloaded during the calculation process.

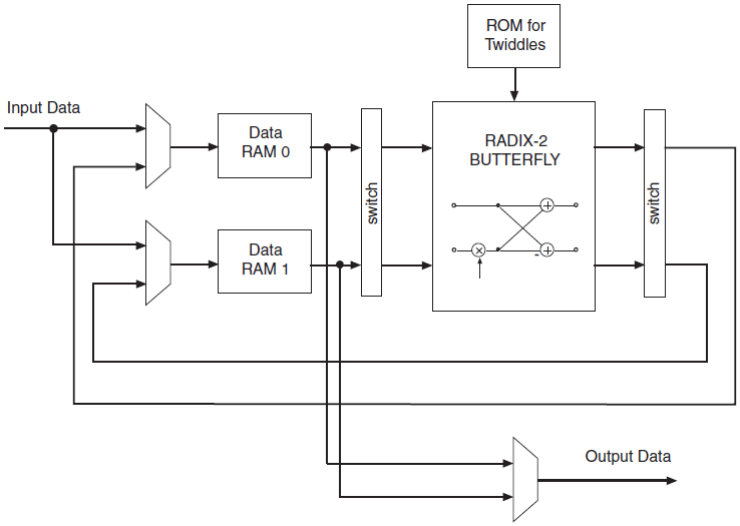


FIGURE D.2 – Radix-2 architecture. [112]

This solution supports point sizes from 8 to 65536. Both the data memories and phase factor memories can be in either block RAM or distributed RAM (the latter for point sizes less than or equal to 1024).

List of Abbreviations

ADC	Analog-to-Digital Converter
ARP	Address Resolution Protocol
ASIC	Application Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
AXI	Advanced eXtensible Interface
BPSK	Binary Phase Shift Keying
BRAM	Block Random Access Memory
BTS	Base Transceiver Station
CDMA	Code Division Multiple Access
CLB	Configurable Logic Block
CMT	Clock Management Tile
CR	Cognitive Radio
CRM	Cognitive Radio Management
CRMu	Cognitive Radio Management unit
DCM	Digital Clock Manager
DLMB	Data-side Local Memory Bus
DFT	Discrete Fourier Transform
DMA	Direct Memory Access
DPR	Dynamic Partial Reconfiguration
DAB	Digital Audio Broadcasting

DSA	Dynamic Spectrum Access
DSP	Digital Signal Processor
DVB	Digital Video Broadcasting
EDK	Embedded Development Kit
FCC	Federal Communications Commission
FDM	Frequency Division Multiplexing
FIR	Finite Impulse Response
FFT	Fast Fourier Transform
FP7	Seventh Framework Programme
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GMII	Gigabit Media Independent Interface
GPIO	General Purpose Input/Output
GPP	General Purpose Processor
GSM	Global System for Mobile communications
GUI	Graphical User Interface
HDCRAM	Hierarchical and Distributed Cognitive Radio Architecture Management
HDL	Hardware Description Language
ICAP	Internal Configuration Access Port
ICT	Information and Communication Technology
IDFT	Inverse Discrete Fourier Transform
IFFT	Inverse Fast Fourier Transform
IP	Internet Protocol
ITU	International Telecommunication Union
JTAG	Joint Test Action Group
LTE	Long-Term Evolution
MAC	Media Access Control

MII	Media Independent Interface
NoC	Network on Chip
OCM	On-Chip Memory
OFDM	Orthogonal Frequency Division Multiplexing
OPEX	Operating Expenditure
OS	Operating System
PCAP	Processor Configuration Access Port
PE	Processing Element
PL	Programmable Logic
PLB	Processor Local Bus
PS	Processing System
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
ReM	Reconfiguration Management
ReMu	Reconfiguration Management unit
SDK	Software Development Kit
SDR	Software Defined Radio
SNR	Signal to Noise Ratio
SR	Software Radio
TI	Texas Instruments
UDP	User Datagram Protocol
VHDL	Very-high-speed integrated circuit Hardware Description Language
WIMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
XPA	Xilinx Power Analyzer
XPE	Xilinx Power Estimator
XPS	Xilinx Platform Studio

List of Figures

List of Tables

Publications

Journals

- Wu X, Palicot J, Leray P. "Metrics on Energy Efficiency for Cognitive Green Equipment Based on FPGA Platform," *IEEE Systems Journal*, vol. PP, Issue 99, pp. 1-12, July 2015. DOI : 10.1109/JSYST.2015.2448596.

- Wu X, Naoues M, Palicot J, Leray P. "Hierarchical reconfiguration management for heterogeneous cognitive green radio equipments," to be submitted to *IEEE Transactions on Circuits and Systems II* .

Conferences

- Wu X, Palicot J, Leray P, "COGNITIVE RADIO MANAGEMENT BENEFITING FROM FLEXIBLE RECONFIGURATION," *4th International Conference on Telecommunications and Remote Sensing*, Rhodes, Greece, September 17-18, 2015.

- Wu X, Palicot J, Leray P, "Reducing Power Consumption by Switching between Serial Mode and Parallel Mode," *1st URSI Atlantic Radio Science Conference (URSI AT-RASC)*, Gran Canaria, Spain, May, 2015.

- Wu X, Darak S J, Leray P, Palicot J and Zhang H, "Reconfiguration Management on FPGA Platform for Cognitive Radio," *XXXI General Assembly and Scientific Symposium of the International Union of Radio Science (URSI)*, Beijing, China, Aug. 2014.

- Darak S J, Wu X, Palicot J and Zhang H, "Linear Phase Filter Bank Design with Unabridged Control over Bandwidth and Center Frequency of Subbands," *XXXI*

General Assembly and Scientific Symposium of the International Union of Radio Science (URSI), Beijing, China, Aug. 2014.

– Xiguang Wu, Pierre Leray, and Jacques Palicot,"A High Speed Approach of Downloading FPGA Partial Bitstreams through UDP for Reconfigurable SDR," *8th Karlsruhe Workshop on Software Radios*, Karlsruhe, Germany, March 12-13, 2014.

– Oussama LAZRAK, Xiguang WU, and Christophe MOY, "Design Approach for Cognitive Radio on Heterogeneous Platform," *8th Karlsruhe Workshop on Software Radios*, Karlsruhe, Germany, March 12-13, 2014.

Demonstration

– X. Wu, J. Palicot, P. Leray, C. Moy, "HW/SW Heterogeneous multi-standard OFDM transceiver," *international workshop on Radio Virtual Machine and Security for Multi-RAT Reconfigurable Systems*, Rennes / Cesson-Sévigné, France, March 10, 2016.

Bibliography

[1] United Nations, "Report of the world commission on environment and development," *General Assembly Resolution 42/187*, 1987.

[2] J. Palicot and C. Roland, "On the use of cognitive radio for decreasing the electromagnetic radiations," in *URSI*, vol. 5, pp. 23–29, 2005.

[3] J. Palicot, "Cognitive radio: an enabling technology for the green radio communications concept," in *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly*, pp. 489–494, ACM, 2009.

[4] "Greentouch." [EB/OL]. `http://www.greentouch.org`.

[5] "Energy aware radio and network technologies (earth)." [EB/OL]. `https://www.ict-earth.eu`.

[6] "Cognitive radio and cooperative strategies for power saving in multi-standard wireless devices (c2power)." [EB/OL]. `http://www.ict-c2power.eu/`.

[7] "Smart power grid for energy efficient small cell networks (sogreen)." [EB/OL]. `http://www.agence-nationale-recherche.fr/?Project=ANR-14-CE28-0025`.

[8] "Toward energy proportional networks (tepn)." [EB/OL]. `http://www.tepn.cominlabs.ueb.eu/`.

[9] J. Mitola, "The software radio architecture," *Communications Magazine, IEEE*, vol. 33, no. 5, pp. 26–38, 1995.

[10] J. Mitola III and G. Q. Maguire Jr, "Cognitive radio: making software radios more personal," *Personal Communications, IEEE*, vol. 6, no. 4, pp. 13–18, 1999.

[11] J. Mitola, *Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio*. PhD thesis, Royal Inst. of Tech, Sweden, May 2000.

[12] J. Palicot, ed., *Radio Enineering: From Software radio to Cognitive Radio.* Wiley, 2011.

[13] S. S. Compagny, "Spectrum occupancy measurement.," *http://www.sharedspectrum.com/measurements/*, 2005.

[14] S. Haykin, "Cognitive radio: brain-empowered wireless communications," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 2, pp. 201–220, 2005.

[15] X. Zhang, W. Jouini, P. Leray, and J. Palicot, "Temperature-power consumption relationship and hot-spot migration for fpga-based system," in *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, pp. 392–397, IEEE, 2010.

[16] M. A. Marsan, L. Chiaraviglio, D. Ciullo, and M. Meo, "Optimal energy savings in cellular access networks," in *Proceedings of IEEE ICC Workshops 2009*, (Dresden, Germany), pp. 1–5, June 2009.

[17] J. Wu, S. Rangan, and H. Zhang, eds., *Green Communications: Theoretical Fundamentals, Algorithms and Applications.* CRC Press, 2012.

[18] A. Fehske, G. Fettweis, J. Malmodin, and G. Biczok, "The global footprint of mobile communications: The ecological and economic perspective," *IEEE Communications Magazine*, vol. 49, no. 8, pp. 55–62, 2011.

[19] The Climate Group, "Smart 2020: Enabling the low carbon economy in the information age." Global e-Sustainability Initiative (GeSI), 2008.

[20] G. Gur and F. Alagoz, "Green wireless communications via cognitive dimension: an overview," *IEEE Network*, vol. 25, no. 2, pp. 50–56, 2011.

[21] Y. Zhao, S. Mao, J. O. Neel, and J. H. Reed, "Performance evaluation of cognitive radios: Metrics, utility functions, and methodology," *Proceedings of the IEEE*, vol. 97, no. 4, pp. 642–659, 2009.

[22] L. Godard, C. Moy, and J. Palicot, "From a configuration management to a cognitive radio management of SDR systems," in *Cognitive Radio Oriented Wireless Networks and Communications, 2006. 1st International Conference on*, pp. 1–5, IEEE, 2006.

[23] C. Moy, "High-level design approach for the specification of cognitive radio equipments management apis," *Journal of Network and Systems Management*, vol. 18, no. 1, pp. 64–96, 2010.

[24] Xilinx, Inc, *Partial Reconfiguration User Guide, UG702 (v12.3)*, 2010.

[25] C. Langton, "Orthogonal Frequency Division Multiplexing (OFDM) Tutorial." [EB/OL]. http://complextoreal.com/tutorials/tutorial-22-orthogonal-frequency-division-multiplex-ofdm-dmt/#.Vlwmf7_eLaQ.

[26] I. T. Union, "Ict facts & figures." [EB/OL], May 2015. http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2015.pdf.

[27] I. T. Union, "Measuring the information society report 2014." [EB/OL], 2014. http://www.itu.int/en/ITU-D/Statistics/Documents/publications/mis2014/MIS2014_without_Annex_4.pdf.

[28] "Green communication technologies." [EB/OL]. http://mobile.research.southwales.ac.uk/green-communication-technologies/.

[29] A. P. Bianzino, C. Chaudet, D. Rossi, and J. Rougier, "A survey of green networking research," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 1, pp. 3–20, 2012.

[30] D. Feng, C. Jiang, G. Lim, L. J. Cimini Jr, G. Feng, and G. Y. Li, "A survey of energy-efficient wireless communications," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 167–178, 2013.

[31] T. Chen, Y. Yang, H. Zhang, H. Kim, and K. Horneman, "Network energy saving technologies for green wireless access networks," *IEEE Wireless Communications*, vol. 18, no. 5, pp. 30–38, 2011.

[32] G. Y. Li, Z. Xu, C. Xiong, C. Yang, S. Zhang, Y. Chen, and S. Xu, "Energy-efficient wireless communications: tutorial, survey, and open issues," *IEEE Wireless Communications*, vol. 18, no. 6, pp. 28–35, 2011.

[33] L. M. Correia, D. Zeller, O. Blume, D. Ferling, Y. Jading, I. Gódor, G. Auer, and L. Van Der Perre, "Challenges and enabling technologies for energy aware mobile radio networks," *IEEE Communications Magazine*, vol. 48, no. 11, pp. 66–72, 2010.

[34] C. Han, T. Harrold, S. Armour, I. Krikidis, S. Videv, P. M. Grant, H. Haas, J. S. Thompson, I. Ku, C.-X. Wang, *et al.*, "Green radio: radio techniques to enable energy-efficient wireless networks," *IEEE Communications Magazine*, vol. 49, no. 6, pp. 46–54, 2011.

[35] Y. Chen, S. Zhang, S. Xu, and G. Y. Li, "Fundamental trade-offs on green wireless networks," *IEEE Communications Magazine*, vol. 49, no. 6, pp. 30–37, 2011.

[36] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy efficiency in the future internet: a survey of existing approaches and trends in energy-aware fixed network infrastructures," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 2, pp. 223–244, 2011.

[37] K. Hinton, J. Baliga, M. Z. Feng, R. Ayre, and R. Tucker, "Power consumption and energy efficiency in the internet," *IEEE Network*, vol. 25, no. 2, pp. 6–12, 2011.

[38] J. Baliga, R. W. Ayre, K. Hinton, and R. Tucker, "Green cloud computing: Balancing energy in processing, storage, and transport," *Proceedings of the IEEE*, vol. 99, no. 1, pp. 149–167, 2011.

[39] "Mobile vce green radio." [EB/OL]. `http://www.mobilevce.com/green-radio`.

[40] "Opera-net 2." [EB/OL]. `http://projects.celticplus.eu/opera-net2/docs/OPERA_Net2_leaflet_hq.pdf`.

[41] "Green grid." [EB/OL]. `http://www.thegreengrid.org/`.

[42] "Fit4green." [EB/OL]. `http://www.fit4green.eu/`.

[43] "low energy consumption networks (econet)." [EB/OL]. `http://www.econet-project.eu/`.

[44] "Cognitive heterogeneous reconfigurable optical network (chron)." [EB/OL]. `http://www.ict-chron.eu/`.

[45] GreenTouch, "Gwatt - visualizing the greentouch results." [EB/OL]. `http://www.greentouch.org/index.php?page=gwatt-visualizing-the-greentouch-results`.

[46] GreenTouch, "Power model for wireless base stations." [EB/OL]. `http://www2.imec.be/be_en/research/wireless-communication/power-model-html.html`.

[47] "Towards real energy-efficient network design (trend)." [EB/OL]. `http://www.fp7-trend.eu/`.

[48] "Trendmeter." [EB/OL]. `http://trend.polito.it/`.

[49] "powerlib." [EB/OL]. `http://powerlib.intec.ugent.be`.

[50] E. Buracchini, "The software radio concept," *Communications Magazine, IEEE*, vol. 38, no. 9, pp. 138–143, 2000.

[51] FCC Spectrum Policy Task Force (SPTF), "Report of the Spectrum Efficiency Working Group," Nov. 2002.

[52] FCC Spectrum Policy Task Force (SPTF), "Spectrum Policy Task Force Report." ET Docket No. 02-135, Nov. 2002.

[53] Shared Spectrum Company, "Spectrum Occupancy Measurements," 2004. `http://www.sharedspectrum.com/papers/spectrum-reports/`.

[54] V. Valenta, R. Maršálek, G. Baudoin, M. Villegas, M. Suarez, and F. Robert, "Survey on spectrum utilization in europe: Measurements, analyses and observations," in *Cognitive Radio Oriented Wireless Networks & Communications (CROWNCOM), 2010 Proceedings of the Fifth International Conference on*, pp. 1–5, IEEE, 2010.

[55] O. Lazrak, C. Moy, and P. Leray, "Modeling cognitive radio equipments for opportunistic spectrum access," in *Wireless Innovation Forum European Conference on Communications Technologies and Software Defined Radio 2013*.

[56] "Besting Reading Topics on Cognitive Radio." [EB/OL]. `http://www.comsoc.org/best-readings/topic/cognitive-radio`.

[57] "Cognitive Radio." [EB/OL]. `https://en.wikipedia.org/wiki/Cognitive_radio`.

[58] J. Palicot, J. Mitola, Z. Z. Lei, and F. K. Jondral, "Special issue on 10 years of cognitive radio: state-of-the-art and perspectives," *Eurasip Journal on Wireless Communications and Networking*, vol. 2012, no. 1, pp. 1–4, 2012.

[59] J. Wu, I. Bisio, H. Li, E. Hossain, C. Gniady, and M. Valla, "Context-aware networking and communications," 2014.

[60] J. Palicot, C. Moy, and R. Hachemani, "Multilayer sensors for the sensorial radio bubble," *Physical Communication*, vol. 2, no. 1, pp. 151–165, 2009.

[61] M. Ghozzi, F. Marx, M. Dohler, and J. Palicot, "Cyclostatilonarilty-based test for detection of vacant frequency bands," in *Cognitive Radio Oriented Wireless Networks and Communications, 2006. 1st International Conference on*, pp. 1–5, IEEE, 2006.

[62] R. Hachemani, J. Palicot, and C. Moy, "A new standard recognition sensor for cognitive radio terminals," *EURASIP, Kessariani, Greece*, 2007.

[63] Z. Khalaf, A. Nafkha, and J. Palicot, "Blind spectrum detector for cognitive radio using compressed sensing," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pp. 1–5, IEEE, 2011.

[64] A. Nafkha, R. Seguier, J. Palicot, C. Moy, and J.-P. Delahaye, "A reconfigurable baseband transmitter for adaptive image coding," in *Mobile and Wireless Communications Summit, 2007. 16th IST*, pp. 1–5, IEEE, 2007.

[65] W. Jouini, "Energy detection limits under log-normal approximated noise uncertainty," *Signal Processing Letters, IEEE*, vol. 18, no. 7, pp. 423–426, 2011.

[66] W. Jouini, C. Moy, and J. Palicot, "On decision making for dynamic configuration adaptation problem in cognitive radio equipments: a multi-armed bandit based approach," in *6th Karlsruhe Workshop on Software Radios, WSR*, vol. 10, pp. 21–30, 2010.

[67] C. J. Rieser, *Biologically inspired cognitive radio engine model utilizing distributed genetic algorithms for secure and robust wireless communications and networking.* PhD thesis, Virginia Polytechnic Institute and State University, 2004.

[68] T. W. Rondeau, B. Le, D. Maldonado, D. Scaperoth, and C. W. Bostian, "Cognitive radio formulation and implementation," in *Cognitive Radio Oriented Wireless Networks and Communications, 2006. 1st International Conference on*, pp. 1–10, IEEE, 2006.

[69] V. P. S. Kirar, "Artificial neural networks for cognitive radio network: A survey,"

[70] K. Tsagkaris, A. Katidiotis, and P. Demestichas, "Neural network-based learning schemes for cognitive radio systems," *Computer Communications*, vol. 31, no. 14, pp. 3394–3404, 2008.

[71] N. K. Kasabov, "Ecos: Evolving connectionist systems and the eco learning paradigm," in *Iconip*, vol. 98, pp. 1232–1235, 1998.

[72] N. Kasabov, *Evolving connectionist systems: the knowledge engineering approach.* Springer Science & Business Media, 2007.

[73] T. Weingart, D. C. Sicker, and D. Grunwald, "A statistical method for reconfiguration of cognitive radios," *Wireless Communications, IEEE*, vol. 14, no. 4, pp. 34–40, 2007.

[74] W. Jouini, C. Moy, and J. Palicot, "Decision making for cognitive radio equipment: analysis of the first 10 years of exploration.," *EURASIP J. Wireless Comm. and Networking*, vol. 2012, p. 26, 2012.

[75] W. Jouini, D. Ernst, C. Moy, and J. Palicot, "Multi-armed bandit based policies for cognitive radio's decision making issues," in *3rd international conference on Signals, Circuits and Systems (SCS)*, 2009.

[76] W. Jouini, D. Ernst, C. Moy, and J. Palicot, "Upper confidence bound based decision making strategies and dynamic spectrum access," in *Communications (ICC), 2010 IEEE International Conference on*, pp. 1–5, IEEE, 2010.

[77] W. Jouini, C. Moy, and J. Palicot, "Upper confidence bound algorithm for opportunistic spectrum access with sensing errors," in *6th International ICST Conference on Cognitive Radio Oriented Wireless Networks and Communications, Osaka, Japan*, vol. 17, 2011.

[78] J.-P. Delahaye, C. Moy, P. Leray, and J. Palicot, "Managing dynamic partial reconfiguration on heterogeneous SDR platforms," in *SDR Forum Technical Conference*, vol. 5, 2005.

[79] J. Palicot, C. Moy, B. Résimont, and R. Bonnefoi, "Application of hierarchical and distributed cognitive architecture management for the smart grid," *Ad Hoc Networks*, 2015.

[80] J. Delorme, A. Nafkha, P. Leray, and C. Moy, "New OPBHWICAP Interface for Realtime Partial Reconfiguration of FPGA," in *Reconfigurable Computing and FPGAs, 2009. ReConFig'09. International Conference on*, pp. 386–391, IEEE, 2009.

[81] P. Leray, A. Nafkha, and C. Moy, "Implementation scenario for teaching partial reconfiguration of fpga," in *Proc. 6th International Workshop on Reconfigurable Communication Centric Systems-on-Chip (ReCoSoC), Montpellier, France*, 2011.

[82] "GNU Radio." [EB/OL]. `http://gnuradio.org/redmine/projects/gnuradio/wiki`.

[83] M. Braun and J. Pendlum, "RFNoC: RF Network on Chip," in *GNU Radio Conference 2015*, 2015.

[84] P. Sutton, J. Lotze, H. Lahlou, S. A. Fahmy, K. Nolan, B. Ozgul, T. W. Rondeau, J. Noguera, and L. E. Doyle, "Iris: an architecture for cognitive radio networking testbeds," *Communications Magazine, IEEE*, vol. 48, no. 9, pp. 114–122, 2010.

[85] Xilinx, Inc, *Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC User Guide, UG194 (v1.10)*, 2011.

[86] X. Wu, P. Leray, and J. Palicot, "A high speed approach of downloading fpga partial bitstreams through udp for reconfigurable sdr," in *8th Workshop on Software Radios (WSR 2014)*, (Karlsruhe, Germany), Mar. 2014.

[87] S. Velusamy, "Lightweight ip (lwip) application examples," *Xilinx, XAPP1026 (v1.0)*, 2008.

[88] P. Bomel, J. Crenne, L. Ye, J.-P. Diguet, and G. Gogniat, "Ultra-fast downloading of partial bitstreams through ethernet," in *Architecture of Computing Systems–ARCS 2009*, pp. 72–83, Springer, 2009.

[89] Xilinx, Inc, *ZC702 Evaluation Board for the Zynq-7000 XC7Z020 All Programmable SoC User Guide, UG850 (v1.2)*, 2013.

[90] C. Kohn, "Partial reconfiguration of a hardware accelerator on zynq-7000 all programmable soc devices," *Xilinx, XAPP1159 (v1. 0)*, 2013.

[91] Xilinx, Inc, *Zynq-7000 All Programmable SoC Technical Reference Manual, UG585 (v1.6.1)*, 2013.

[92] "Fusion Digital Power Manufacturing Tool." [EB/OL]. `http://www.ti.com/tool/fusion_mfr_gui`.

[93] Xilinx, Inc, *Spartan-6 FPGA Power Management User Guide*, 2012.

[94] Xilinx, Inc, *Virtex-5 FPGA System Monitor User Guide, UG192 (v1.7.1)*, 2011.

[95] A. Nafkha, P. Leray, Y. Louet, and J. Palicot, "Moving a Processing Element from Hot to Cool Spots: Is This an Efficient Method to Decrease Leakage Power Consumption in FPGAs?," in *Green Communications: Theoretical Fundamentals, Algorithms and Applications* (J. Wu, S. Rangan, and H. Zhang, eds.), ch. 8, pp. 198–219, CRC Press, 2012.

[96] G. M. Quénot, N. Paris, and B. Zavidovique, "A temperature and voltage measurement cell for VLSI circuits," in *Proceedings of Euro ASIC'91*, pp. 334–338, IEEE, 1991.

[97] K. M. Zick and J. P. Hayes, "On-line sensing for healthier fpga systems," in *Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays*, pp. 239–248, ACM, 2010.

[98] E. Boemo and S. López-Buedo, "Thermal monitoring on fpgas using ring-oscillators," in *Field-Programmable Logic and Applications*, pp. 69–78, Springer, 1997.

[99] N. Michael, C. Moy, A. P. Vinod, and J. Palicot, "Area-power trade-offs for flexible filtering in green radios," *Communications and Networks, Journal of*, vol. 12, no. 2, pp. 158–167, 2010.

[100] N. S. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir, and V. Narayanan, "Leakage current: Moore's law meets static power," *computer*, vol. 36, no. 12, pp. 68–75, 2003.

[101] Xilinx, Inc, "Virtex-5 FPGA System Power Design Considerations." White Paper, Feb. 1997.

[102] I. Krikidis, J.-L. Danger, and L. Naviner, "An iterative reconfigurability approach for wcdma high-data-rate communications.," *IEEE Wireless Communications*, vol. 13, no. 3, p. 8, 2006.

[103] Xilinx, Inc, *Xilinx Power Tools Tutorial, Spartan-6 and Virtex-6, UG733 (v1.0)*, 2010.

[104] Xilinx, Inc, *Power Methodology Guide, UG786 (v13.1)*, 2011.

[105] "IEEE standard for local and metropolitan area networks Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications." IEEE Standards Association, 1999.

[106] "Radio broadcasting systems ; Digital Audio Broadcasting (DAB) to mobile, portable, and fixed receivers, ETSI 300 401." European Telecommunications Standards Institute, (ETSI), 2001.

[107] "Radio broadcasting systems ; Digital Video Broadcasting (DVB) ; framing structure, channel coding and modulation for terrestrial television, ETSI 300 401." European Telecommunications Standards Institute, (ETSI), 2004.

[108] 3GPP, "LTE ; Evolved Universal Terrestrial Radio Access (E-UTRA) ; Physical channels and modulation." 3rd Generation Partnership Project (3GPP), TS 36.211 v10.0.0, Jan. 2011.

[109] J. Palicot and C. Roland, "Fft: a basic function for a reconfigurable receiver," in *Telecommunications, 2003. ICT 2003. 10th International Conference on*, vol. 1, pp. 898–902, IEEE, 2003.

[110] A. Al Ghouwayel, Y. Louët, and J. Palicot, "A reconfigurable butterfly architecture for fourier and fermat transforms," in *WSR'06*, 2006.

[111] M. Naoues, D. Noguet, L. Alaus, and Y. Louët, "A common operator for fft and fec decoding," *Microprocessors and Microsystems*, vol. 35, no. 8, pp. 708–715, 2011.

[112] Xilinx, Inc, *LogiCORE IP Fast Fourier Transform v8.0, DS808*, 2012.

[113] J. Delorme, J. Martin, A. Nafkha, C. Moy, F. Clermidy, P. Leray, and J. Palicot, "A fpga partial reconfiguration design approach for cognitive radio based on noc architecture," in *Circuits and Systems and TAISA Conference, 2008. NEWCAS-TAISA 2008. 2008 Joint 6th International IEEE Northeast Workshop on*, pp. 355–358, IEEE, 2008.

[114] O. Lazrak, X. Wu, and C. Moy, "Design approach for cognitive radio on heterogeneous platform," in *WSR'14*, pp. 6–pages, 2014.

[115] A. Mattsson, B. Lundell, B. Lings, and B. Fitzgerald, "Linking model-driven development and software architecture: a case study," *Software Engineering, IEEE Transactions on*, vol. 35, no. 1, pp. 83–93, 2009.

[116] O. M. G. (OMG), "MDA Guide Version 1.0.1." [EB/OL]. `http://www.omg.org/news/meetings/workshops/UML_2003_Manual/00-2_MDA_Guide_v1.0.1.pdf`.

[117] S. Derrien, S. Rajopadhye, P. Quinton, and T. Risset, "High-level synthesis of loops using the polyhedral model," in *High-level synthesis*, pp. 215–230, Springer, 2008.

[118] J. Postel, "User Datagram Protocol." RFC 768, Aug. 1980.

[119] J. Postel, "Internet protocol." RFC 791, Sept. 1981.

[120] J. Postel, "Transmission control protocol." RFC 793, Sept. 1981.

[121] N. Alachiotis, S. A. Berger, and A. Stamatakis, "Efficient pc-fpga communication over gigabit ethernet," in *CIT*, pp. 1727–1734, IEEE Computer Society, 2010.

[122] N. Alachiotis, S. A. Berger, and A. Stamatakis, "A versatile udp/ip based pc-fpga communication platform," in *ReConFig*, pp. 1–6, IEEE, 2012.

[123] A. Lofgren, L. Lodesten, S. Sjöholm, and H. Hansson, "An analysis of fpga-based udp/ip stack parallelism for embedded ethernet connectivity," in *23rd NORCHIP Conference*, pp. 94–97, IEEE, 2005.

[124] A. Dollas, I. Ermis, I. Koidis, I. Zisis, and C. Kachris, "An open tcp/ip core for reconfigurable logic," in *13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'05)*, pp. 297–298, IEEE, 2005.

[125] Xilinx, Inc, *ML505/ML506/ML507 Evaluation Platform User Guide ,UG347 (v3.1.2)*, 2011.

[126] Xilinx, Inc, *Virtex-5 FPGA User Guide, UG190 (v5.4)*, 2012.

[127] Xilinx, Inc, *ML505/506/507 Overview and Setup*, 2010.

[128] Xilinx, Inc, *Zynq-7000 All Programmable SoC: ZC702 Evaluation Kit and Video and Imaging Kit (ISE Design Suite 14.5) Getting Started Guide, UG926 (v4.0)*, 2013.

[129] "Xilinx Zynq-7000 All Programmable SoC ZC702 Evaluation Kit." [EB/OL]. `http://www.xilinx.com/products/boards-and-kits/ek-z7-zc702-g.html#hardware`.

[130] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

As the digital communication systems evolve from GSM and now toward 5G, the supported standards are also growing. The desired communication equipments are required to support different standards in a single device at the same time. And more and more wireless Internet services have been being provided resulting in the explosive growth in data traffic, which increase the energy consumption of the communication devices thus leads to significant impact on global CO_2 emission. More and more researches have focused on the energy efficiency of wireless communication. Cognitive Radio (CR) has been considered as an enabling technology for green radio communications due to its ability to adapt its behavior to the changing environment. In order to efficiently manage the sensing information and the reconfiguration of a cognitive equipment, it is essential, first of all, to gather the necessary metrics so as to provide enough information about the operating condition thus helping decision making. Then, on the basis of the metrics obtained, an optimal decision can be made and is followed by a reconfiguration action, whose aim is to minimize the power dissipation while not compromising on performance. Therefore, a management architecture is necessary to be added into the cognitive equipment acting as a glue to realize the CR capabilities. We introduce a management architecture, namely Hierarchical and Distributed Cognitive Radio Architecture Management (HDCRAM), which has been proposed for CR management by our team. This work focuses on the implementation of HDCRAM on heterogeneous platforms. One of the objectives is to improve the energy efficiency by the management of HDCRAM. And an example of a simplified OFDM system is used to explain how HDCRAM works to efficiently manage the system to adapt to the changing environment.

Keywords: Software Radio, Cognitive Radio, Green Radio, HDCRAM, FPGA, Partial Reconfiguration.

Pour supporter l'évolution constante des standards de communication numérique, du GSM vers la 5G, les équipements de communication doivent continuellement s'adapter. Face à l'utilisation croissante de l'internet, on assiste à une explosion du trafic de données, ce qui augmente la consommation d'énergie des appareils de communication sans fil et conduit donc à un impact significatif sur les émissions mondiales de CO_2. De plus en plus de recherches se sont concentrées sur l'efficacité énergétique de la communication sans fil. La Radio Intelligente, ou Cognitive Radio (CR), est considérée comme une technologie pertinente pour les communications radio vertes en raison de sa capacité à adapter son comportement à son environnement. Sur la base de métriques fournissant suffisamment d'informations sur l'état de fonctionnement du système, une décision optimale peut être effectuée en vue d'une action de reconfiguration, dans le but de réduire au minimum la dissipation d'énergie tout en ne compromettant pas les performances. Par conséquent, tout équipement intelligent doit disposer d'une architecture de gestion de la reconfiguration. Nous avons retenu l'architecture HDCRAM (Hierarchical and Distributed Cognitive Radio Architecture Management), développée dans notre équipe, et nous l'avons déployée sur des plates-formes hétérogènes. L'un des objectifs est d'améliorer l'efficacité énergétique par la mise en œuvre de l'architecture HDCRAM. Nous l'avons appliquée à un système OFDM simplifié pour illustrer comment HDCRAM permet de gérer efficacement le système et son adaptation à un environnement évolutif.

Mots-clés: Radio Logicielle, Radio Intelligente, Eco-Radio, HDCRAM, FPGA, Reconfiguration Partielle.