Dynamic Reconfiguration of Link Adaptation algorithms for a Multi Carrier System

Software Defined Radio

Group - 07gr1139

Sri Hanuma Chitti
Gaurav Kulkarni

Spring 2007
AALBORG UNIVERSITY

AALBORG UNIVERSITY

The Faculty Of Engineering, Science and Medicine
Fredrik Bajers Vej 7, A1
DK-9220 Aalborg East
Phone: 9635 8695
Web: http://kom.aau.dk

TITLE: Dynamic Reconfiguration of Link Adaptation Algorithms for a Multi Carrier System

PROJECT PERIOD:
5th February, 2007 - 7th June, 2007

GROUP: 1139

SEMESTER:
Spring 2007

GROUP MEMBERS:
Sri Hanuma Chitti
Gaurav Kulkarni

SUPERVISORS:
Yannick Le Moullec
Andreas Popp

Number Of Duplicates:5

Number Of Pages In Report:79

Number Of Pages In Appendix:13

Total Number Of pages:95

ABSTRACT:

The objective of this project is to analyze whether dynamic partial reconfiguration (PR) of an FPGA is feasible for Link Adaptation (LA) algorithms for an OFDM system such as WiMAX. Partial reconfiguration of an FPGA saves hardware area by virtue of its reusability. In [1] two algorithms SRA and SAMPDA were implemented and a large difference in area usage was noted while providing the same performance at certain conditions. To exploit this property, partial reconfigurability of an FPGA is studied. A new methodology called ATtACk is proposed that helps to move systematically toward implementing a PR system. A step by step process is followed in three domains to finally assess implementation possibilities. A live setup is used to calculate the SNR which is used as the criteria for selecting an algorithm. Guidelines for PR are first studied. It is seen that the pin connections on the FPGA development board, architectural limitations of the FPGA and size of the hardware area requirement do not allow implementation of a complete PR system on the development board considered. Thus each portion of the design is separately implemented and these limitations are analyzed and further changes are proposed. Considerations regarding a practical reconfigurable LA are also discussed. Based on the limitations experienced and the algorithm to be implemented, it is concluded that a PR system for the two LA algorithms is possible provided that further architectural changes and algorithmic optimizations are performed.

Preface

This report serves as documentation for the master's thesis "Dynamic Reconfiguration of Link Adaptation Algorithms for a Multi Carrier System" and presents the work done by Sri Hanuma Chitti and Gaurav Kulkarni during the period February 2007 to June 2007 at the Software Defined Radio specialization at The Faculty of Engineering, Science and Medicine, Department of Electronic Systems, Aalborg University, Denmark.

Wireless communication has undergone a tremendous change since the time it was first operated more than 100 years ago. Today, wireless communication has affected our lives to such an extent that it seems impossible to live without it. As technology becomes more advanced, forthcoming challenges have to be addressed which is the motive of this project. Advances in FPGA technology has opened up new possibilities and this project considers the partial reconfigurability feature of the FPGA. The aim of this project is to analyze the feasibility of implementing a partially reconfigurable system for Link Adaptation algorithms on an FPGA and moves through three major stages:

1. Analyzing the project specifications.
2. Creating a System Model and implementing the design.
3. Analyzing the feasibility of the implemented design.

A new methodology is proposed to implement PR systems. Based on the architectural limitations experienced, future work is suggested.

We would like to take this opportunity to thank everyone because of whom this project has been successful. First and foremost we would like to thank our parents who have been by our side and been a continuous source of inspiration. Next we would like to thank our supervisors, Yannick Le Moullec and Andreas Popp for guiding and supporting us throughout this endeavour. We extend our gratitude toward Kim Olesen for providing us the experimental setup. We also thank other members of the research staff Patrick Eggers, Jesper Kristensen, Suvra Sekhar Das and Muhammad Imadur Rehman for helping us to define the project in the initial stages

....	....
(signature)	(signature)
Gaurav Kulkarni	Sri Hanuma Chitti

Aalborg, June 7, 2007

Report Structure

The first chapter introduces the project and the problem definition.

In Chapter 2, the project environment and the proposed methodology for implementing reconfigurable systems is described. Based on this methodology, the requirements and the overall project setup is presented.

Following the methodology, the parameters of the system model is presented in Chapter 3. Design considerations are also discussed.

Chapter 4 describes the implementation process and the challenges encountered in the different domains considered in this project.

Testing the implementation and its analysis along with the proposed solution is presented in Chapter 5. It also describes the considerations to be done for implementing a partially reconfigurable system in a practical case.

Chapter 6 concludes the report along with future areas of research required.

Figures are represented by the chapter number first. For example, "Figure 3.1" means the first figure in Chapter 3. Similar rule holds true for tables. List of Abbreviations, Figures and Tables are provided. Appendices are included at the end of the report. The CD-ROM enclosed holds the report in pdf format, Xilinx ISE project reports, Handel-C codes, MATLAB codes, System Generator design, testing results and the references used.

Contents

List of Figures

List of Tables

Abbreviations

3Gcdma	3^{rd}Generation Code Division Multiple Access
ADC	Analog to Digital Converter
APDA	Adaptive Power Distribution Algorithm
ASIC	Application Specific Integrated Circuit
BPF	Band Pass Filter
BPSK	Binary Phase Shift Keying
BWA	Broadband Wireless Access
CCK	Complementary Code Keying
C/I	Carrier to Interference Ratio
CLB	Configurable Logic Block
CR	Cognitive Radio
DAC	Digital to Analog Converter
DSSS	Direct Sequence Spread Spectrum
DSP	Digital Signal Processor
DMA	Direct Memory Access
DVB	Digital Video Broadcast
FDA	Filter Design and Analysis
FDM	Frequency Division Multiplexing
FFT	Fast Fourier Transform
FIR	Finite Impulse Responsev
FPGA	Field Programmable Gate Array
FUSE	Field Upgradable Systems Environment
HiperLAN	High performance Local Area Network
ICAP	Internal Configuration Access Port
IEEE	Institute of Electrical and Electronics Engineers
IF	Intermediate Frequency
ISI	Inter Symbol Interference
ISE	Integrated Software Environment
JTAG	Joint Test Action Group
LAN	Local Area Network

LA	Link Adaptation
LNA	Low Noise Amplifier
LOC	Location
LOS	Line of Sight
LPF	Low Pass Filter
MBPR	Module Based Partial Reconfiguration
NF	Noise Figure
NLOS	Non Line of Sight
OFDM	Orthogonal Frequency Division Multiplexing
PCI	Peripheral Component Interconnect
PDA	Personal Digital Assistant
PE	Processing Element
PR	Partially Reconfigurable
QPSK	Quadrature Phase Shift Keying
QAM	Quadrature Amplitude Modulation
RAM	Random Access Memory
RF	Radio Frequency
RSSI	Received Signal Strength Indication
SDR	Software Defined Radio
SRA	Simple Rate Adaptive Algorithm
SAMPDA	Sub carrier based Adaptive Modulation and Power Distribution Algorithm
SNR	Signal to Noise Ratio
TBUF	Tri State Buffer
UCF	User Constraint File
UMTS	Universal Mobile Telephone System
USB	Universal Serial Bus
VHDL	Very high speed integrated circuit Hardware Description Language
WiMAX	Worldwide interoperability for Microwave Access
WLAN	Wireless Local Area Network

Chapter 1

Introduction

1.1 Introduction

Wireless technology has seen a tremendous growth in the last two decades and is still continuously evolving. Wireless services have offered a lot of convenience to the user by allowing freedom of movement. The growth in wireless technology and ever increasing demands for higher data rates has led to the development of various wireless standards to provide various services and facilities. Cellular standards such as Universal Mobile Telephone System (UMTS) and 3^{rd} Generation Code Division Multiple Access (3Gcdma) have changed the face of wireless communications. Evolving from a pure voice service, we are now in the era of wireless high speed internet. Standards for wireless broadband internet access have also emerged and today, wireless communication has the potential to provide data rates in excess of 100 Mbps.

Due to the large number of standards, spectrum availability has become an important issue. Spectrum usage regulations prohibit unlicensed users to operate in a licensed spectrum. However, it has been observed that not all of the licensed spectrum are used at all places all the time. An unlicensed user can take advantage of such a situation to communicate thereby increasing spectrum efficiency. This is the basic idea behind Cognitive Radio (CR) [2].

1.1.1 The Cognitive Radio

In a practical situation, not all frequency bands are occupied all the time and there are some spectrum holes existing. A spectrum hole is defined as a band of frequencies assigned to a user, but at a specific time and place that band is not being utilized by the user. So if there is a smart radio terminal that can detect this vacancy and perform transmission/reception in that band, spectrum efficiency will increase. This is the idea behind Cognitive Radio (CR) as proposed by Joseph Mitola [3]. In other words, the radio device is aware of its surroundings and adapts itself accordingly [2].

The goal of cognitive radio is to establish reliable communication with efficient utilization of the available spectrum. In the particular bandwidth that the device wants to operate, it has to make the best use of it or in other words deliver the highest data rate possible. The spectral efficiency of such a system should be high so as to exploit the opportunity in the best way and one technique to do this is through link adaptation. Link adaptation algorithms improve the spectral efficiency by adapting transmission to the current channel conditions. One standard that could use the ideas of CR is the Orthogonal Frequency Division Multiplexing (OFDM) based IEEE 802.16 also known as WiMAX (Worldwide interoperability for Wireless Microwave Access).

1.1.2 The IEEE 802.16 standard (WiMAX)

The IEEE 802.16 standard also known as WiMAX is a Broadband Wireless Access (BWA) technology which provides data rates up to 75 Mbps in the uplink as well as downlink. It operates between 2 - 11 GHz and 10 - 66 GHz and has a maximum range of 48 km. It has a channel width of 1.25 MHz to 28 MHz. It's air interface is based on OFDM wherein multiple orthogonal subcarriers are used which transforms a wideband channel into multiple narrow band channels. [4] [5] [6]

This standard is attractive to be used as it avoids the need for laying cables for internet access since the last mile is wireless. WiMAX can be operated as a fixed as well as mobile service supporting speeds upto 70 mph [1][7]. In this project, the framework of WiMAX which uses OFDM is considered [1].

1.2 Multi Carrier Systems (OFDM)

Orthogonal Frequency Division Multiplexing is a multi carrier scheme which is an improvement over Frequency Division Multiplexing (FDM). The advantage of OFDM is that when the channel is in fade or an interferer in the propagation path causes severe destruction, it is less likely that all the individual carriers in the multi carrier system are also adversely affected by the channel. The difference between FDM and a typical OFDM spectrum is shown in Figure 1.1

OFDM differs from FDM in packing the subcarriers orthogonally. As seen in Figure 1.1, the peak of one subcarrier coincides with the null of the adjacent ones. By doing so, bandwidth is saved. Another advantage of multi carrier systems using parallel data transmission is the possibility of achieving high data rates which is one of the demands of modern wireless communications. Modern standards such as Worldwide Interoperability for Microwave Access (WiMAX), Digital Video Broadcast (DVB) and Wireless Local Area Network (WLAN) use OFDM as the air interface [4].

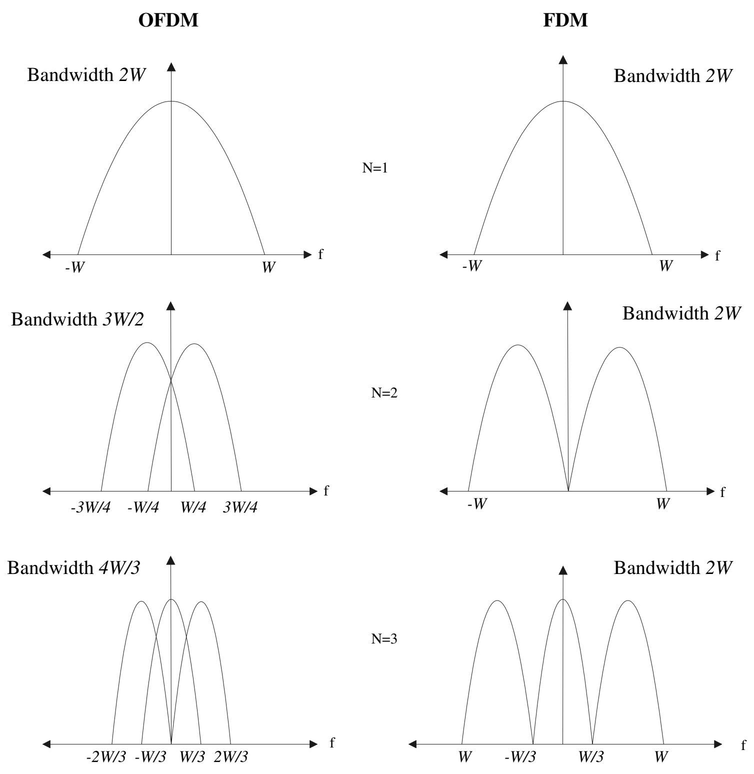


Figure 1.1: *Carrier packing in an OFDM system as compared to FDM. OFDM utilizes the spectrum more efficiently due to it's orthogonal subcarriers.*

The OFDM technique is effective in combating the critical problem of Inter Symbol Interference (ISI) which plays a major role in multi path scenarios in Non Line of Sight (NLOS) conditions. ISI occurs when the delay spread is greater than the symbol duration. In OFDM, a data stream of data rate R bps and bandwidth of B Hz is divided into n parallel streams each with a data rate of R/n bps and a bandwidth of B/n Hz. As long as the subcarrier bandwidth (B/n) is much less than the coherence bandwidth (Bc) each subcarrier experiences flat fading, hence nullifying the effect of ISI [8] [9]. An advantage of choosing a standard based on OFDM as a candidate for CR is that CR relies on sensing the radio environment in terms of spectrum usage and OFDM already has this functionality in the form of the FFT operation to demodulate the signal [10]. In an OFDM system, each subcarrier experiences independent fading. If all the subcarriers use the same modulation scheme, then the highest attenuated subcarrier will dominate the error probability and thus adapting each subcarrier individually is important. This can be achieved through Link Adaptation [1].

1.2.1 Link Adaptation

The SNR of a wireless link varies with space and time. Link adaptation (LA) is used in wireless communications where the transmitter adapts itself to best suit the requirements. A single modulation and coding scheme cannot be optimal under all scenarios, that is, for all Signal to Noise Ratios (SNR). By choosing the modulation scheme for the worst case scenario, resources are not fully utilized as some users may have better channel conditions as compared to others and could support higher order modulation schemes. Figure 1.2 shows the spectral efficiency curve for different modulation schemes versus SNR (γ). For each modulation scheme, there exists a saturation point beyond which the spectral efficiency is maximum and constant. Thus, a technique that determines the channel conditions and then adapts the transmission according to the link requirements is the idea behind link adaptation employing adaptive modulation [1]. By allocating modulation modes to the carriers according to the channel characteristics (for ex. SNR), spectral efficiency can be increased which in turn increases throughput. This process is termed as Link Adaptation.

Two LA algorithms for an OFDM system, Simple Rate Adaptive (SRA) and Sub-carrier based Adaptive Modulation and Power Distribution Algorithm (SAMPDA) were implemented in [1] on a Virtex-4 Field Programmable Gate Array (FPGA). These two algorithms adapt the power and modulation schemes of each subcarrier independently depending on their SNR.

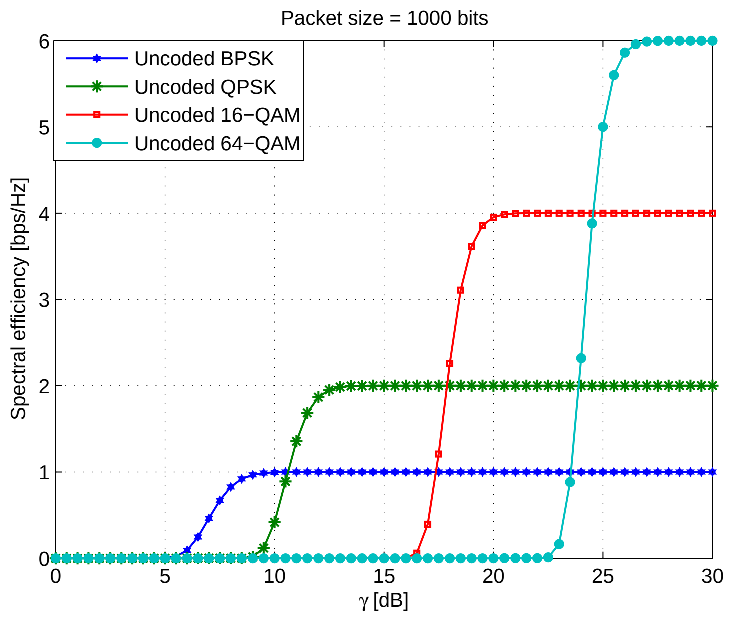


Figure 1.2: *Spectral efficiency for different modulation schemes as a function of SNR [1].Higher modulation schemes provide higher spectral efficiency but at a cost of higher SNR.*

1.2.1.1 Comparison of Two LA Algorithms for OFDM Systems

An implementation comparison was made in [1] among the two LA algorithms with respect to occupied hardware resources and latency i.e. execution time. Table 1.1 shows the comparison between the two algorithms for 16 subcarriers [1]. The execution time of SAMPDA is 0.29ms but it occupies 94% of the area on the FPGA and SRA executes in 0.69ms but utilzes just 35% of the area.

Table 1.1: *Comparison of SRA and SAMPDA implementation on a Xilinx Virtex-4 SX35 FPGA.*

Algorithm	**Number of slices utilized**	**Execution time**
SAMPDA	94%	0.29ms
SRA	35%	0.69ms

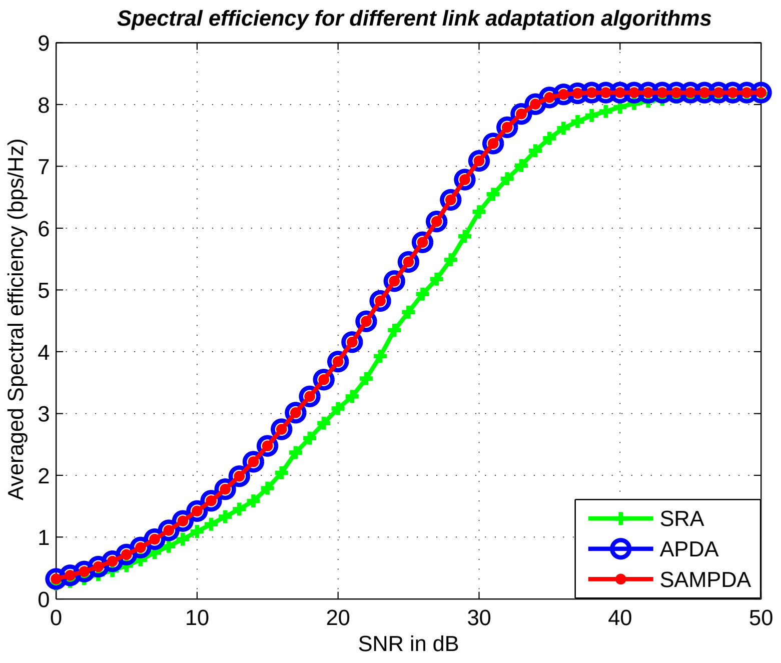


Figure 1.3: *Spectral efficiency comparison of SRA and SAMPDA under varying SNR [1]. Both the algorithms provide the same spectral efficiency below 10 dB and above 40 dB of SNR.*

The spectral efficiency curves for the three algorithms SRA, SAMPDA and Adaptive Power Distribution Algorithm (APDA) are shown in Figure 1.3. The SNR values used here is the average SNR over all the subcarriers. APDA and SAMPDA have the same spectral efficiency but APDA is more complex to implement and thus it is not considered in this project [11].

Based on Table 1.1 and Figure 1.3, two important conclusions are drawn.

1. The spectral efficiency for both the algorithms is almost same for SNR below 10dB and above 40dB. Therefore, in such a case, if time constraints are not too stringent, then SRA algorithm could be used since it occupies less hardware resources. However, between 10dB and 40dB of SNR, SAMPDA provides more spectral efficiency.

2. Further, it was also concluded in [1] that the maximum number of subcarriers that could be adapted by SAMPDA on a Virtex-4 FPGA were only 16 and any further increase would lead to hardware area requirements that were not satisfied by the FPGA. Thus, when the number of subcarriers goes beyond 16, it is necessary to use SRA in this setup.

Therefore a trade off exists between the spectral efficiency, latency of the LA algorithm and the hardware resource usage. To achieve a balance between the three, there is a need for a system that could use both the algorithms and execute the appropriate one depending on the SNR and number of subcarriers.

To have multiple algorithms running on a hardware increases resource utilization and a better and intelligent system is needed to manage hardware resources efficiently. One approach to do so is through a Software Defined Radio (SDR).

1.3 Software Defined Radio

CR exploits available spectrum holes and the system has to be able to adapt itself dynamically to the radio environment. Such a device should be able to support different modem configurations. Having all types of configurations in a hardware will prohibitively increase the size of the device and thus there has to be a different implementation technology. SDR is seen as an enabling technology for CR which exhibits features such as scalability, flexibility and reconfigurability. These qualities are essential for a CR radio terminal. Reconfigurability means that the processing hardware does not always execute the same set of instructions or algorithms. Rather, it is loaded with the appropriate software based on the situational demand. Thus reconfigurability allows the hardware to be customized for the specific task [2][12]. This project focuses on the issue of reconfigurability which is further explained in Chapter 3 and Chapter 4.

1.3.1 Problem Definition

As described in the previous sections, CR is the future of wireless communications where radio devices will be aware of their surroundings and have to efficiently use the available radio resources and thus the spectral efficiency of a CR has to be improved. One technique to improve spectral efficiency is by link adaptation where reliable and high data rates can be achieved by exploiting the channel conditions.

As seen in [1], two Link Adaptation algorithms SAMPDA and SRA can be used under different scenarios. It is interesting to have both the algorithms and using one of them depending on the SNR and number of subcarriers to maximize the spectral efficiency. But having two algorithms in turn increases cost in terms of hardware resource utilization which makes it important to achieve a balance between spectral efficiency and hardware constraints. To reduce hardware utilization, there is a need for dynamic configuration of the device that executes the appropriate LA algorithm which is decided by weighing some system parameters. Factors such as hardware constraints, configuration time and performance have to be taken into account and a systematic method has to be followed.

The objective of this project is to investigate whether real time dynamic reconfiguration of an FPGA by measuring SNR to adapt to the transmission conditions is feasible for efficient spectrum utilization.

1.4 Scope, Assumptions and Delimitations

The scope of this project and important assumptions made are enlisted. More limitations based on further analysis of the available setup are enumerated later.

1.4.1 Scope

- Evaluating reconfiguration of an FPGA for Link Adaptation considering SRA and SAMPDA algorithms.
- An OFDM transmitter and receiver are not considered.
- The transmitted signal is used solely for the purpose of calculating SNR.
- Optimization of Handel-C codes is not considered.

1.4.2 Assumptions and Delimitations

- The subcarriers are not considered to be orthogonal since that would require implementing an OFDM receiver which is not the focus of this project.
- Perfect frequency synchronization between the transmitter and receiver exists.
- The time varying signal is considered as interference free.
- Channel parameters such as delay spread and doppler effect are not considered.

1.5 Goals and Tasks

The goals of this project and tasks to be performed are enlisted.

1.5.1 Goals:

- A simple scheme for calculating the SNR of a live signal.
- Analyzing the feasibility of link adaptation using reconfigurable architecture.
- Proposing an architecture for partial reconfiguration in a practical system.

1.5.2 Tasks:

- Defining a methodology for implementing a reconfigurable architecture.
- Designing and implementing SNR calculation for a baseband signal.
- Analyzing the method of partitioning the FPGA.
- Analyzing FPGA reconfiguration requirements and it's compatibility with the development platform provided.

1.6 Conclusions

In this chapter a brief introduction to Cognitive Radio and the background of this project is presented. LA algorithms are used to improve the spectral efficiency and the need for partial reconfiguration for the two LA algorithms SRA and SAMPDA has been described thus leading to the problem definition. In the next chapter, the project environment and a methodology for implementing partial reconfiguration on a development platform is presented.

Chapter 2

Project Environment and Proposed Methodology

2.1 Introduction

In the previous chapter, the background of this project has been presented. As seen in the problem definition, this project aims at evaluation of a reconfigurable LA architecture for IEEE 802.16. In this chapter, the considerations for the project setup are elaborated. Then a methodology for implementing a partially reconfigurable system is proposed. Following the methodology, the requirements are analyzed and the system setup is presented.

2.2 Project Setup

As seen in Chapter 1, factors such as the SNR of the live signal and number of subcarriers can be used to determine which LA algorithm has to be executed. To have a varying number of subcarriers, a scalable multi carrier transmitter is required. However, to implement such a system would need considerable effort which is not the purpose of this project. Thus the number of subcarriers are fixed but having a varying SNR. So the intention is to measure the SNR of a live signal in order to decide which LA algorithm has to be executed.

An OFDM transmitter is not available at the University and so two single carriers separated by some frequency are transmitted. The maximum possible subcarriers is only two since the FPGA development platforms provided have only two Analog to Digital Converters (ADC). The bandwidth of IEEE 802.16 is 28 MHz which is occupied by 256 subcarriers. Thus each subcarrier occupies 109.375 kHz [6]. For simplicity, this signal is transmitted with a bandwidth of 100 kHz using a data generator and up converted to separate carrier frequencies before transmitting through a single antenna.

The signal is received by an antenna, split into two branches using a power splitter and then each carrier is down converted and sampled. After the signal is sampled, the SNR of each carrier is calculated. As mentioned before, at different SNRs, different LA algorithms can be used and so the average SNR of the two is then used to determine which of the two LA algorithms has to be loaded onto the FPGA. Based on this initial thought, the setup is shown in Figure 2.1. S is the transmitted signal and S_1 and S_2 represent the two carriers being received by the antenna. Their SNR is calculated and based on a decision algorithm, the appropriate LA algorithm is loaded onto the FPGA and executed. The Decision algorithm is explained in the next chapter.

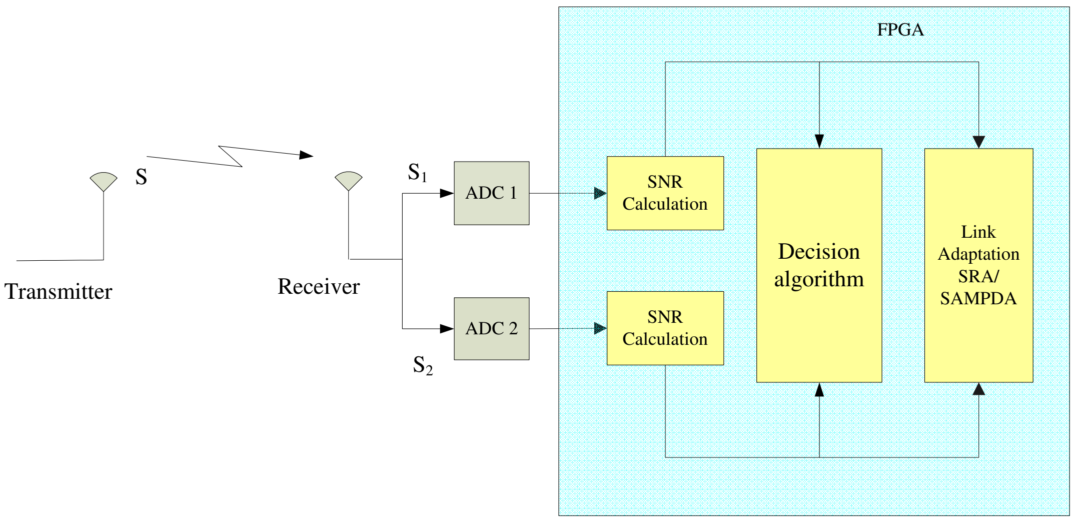


Figure 2.1: *Initial thought of the setup. The SNR is calculated after the signal is digitized to determine the appropriate LA algorithm.*

To decide upon the final setup with the RF components, a generic receiver is considered. This is shown in Figure 2.2

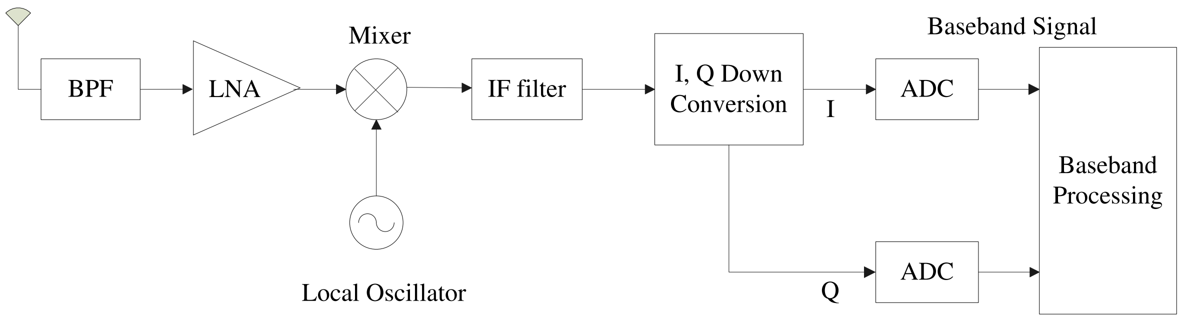


Figure 2.2: *Generic receiver with multiple down conversion stages in analog domain. This setup requires an ADC for I and Q components each.*

As seen in Figure 2.2, multiple down conversion stages are used to obtain the signal in baseband. The Radio Frequency (RF) signal is first converted to an Intermediate Frequency (IF) signal as an intermediate stage. This signal is then filtered and then further down converted to baseband. In case of quadrature modulation schemes, the I and Q components of the signal have to be separately down converted and independently sampled using two ADCs. The FPGA development platforms available at the University have only two ADCs and thus if I and Q signals are sampled, they would be of the same carrier and would have experienced the same channel effects. This defeats the purpose of this project since at least two sub-carriers are needed in order to implement the subcarrier based LA algorithms. Because of these problems, it was decided to sample the signals in IF instead of baseband and then do the further processing. Figure 2.3 shows the setup based on IF/Bandpass sampling. More on this topic is discussed in the next sub-section.

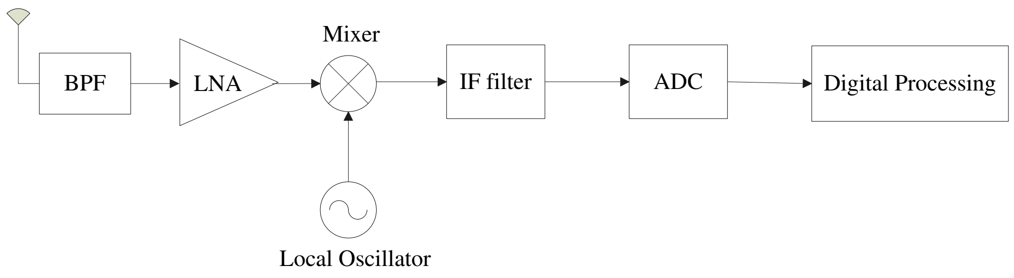


Figure 2.3: *Receiver considered in this project employing Bandpass sampling i.e. the ADC samples the signal at IF rather than at baseband frequency.*

2.2.1 Bandpass Sampling

The Nyquist criteria for sampling a baseband signal states that the sampling frequency should be at least twice the highest frequency in order to extract the information successfully. In a multi mode SDR transceiver, it is not always feasible to sample at twice the highest frequency since this would mean extreme sampling rate requirements [13]. Instead, the signal is sampled at twice the bandwidth and not the highest frequency. It is well known that sampling generates replicas of the frequency spectrum at integral multiples of the sampling frequency. As long as the spectral contents are not aliased, bandpass sampling will not introduce distortion. Figure 2.4 shows the frequency band existing between f_l and f_h. According to Nyquist criteria, this signal will have to be sampled at $2f_h$. But even if the signal is sampled at $2(f_h - f_l)$ the spectrum is replicated in the frequency domain without aliasing.

When the sampling frequency f_s is between two times the bandwidth and two times the highest frequency component, it must satisfy the condition [13]:

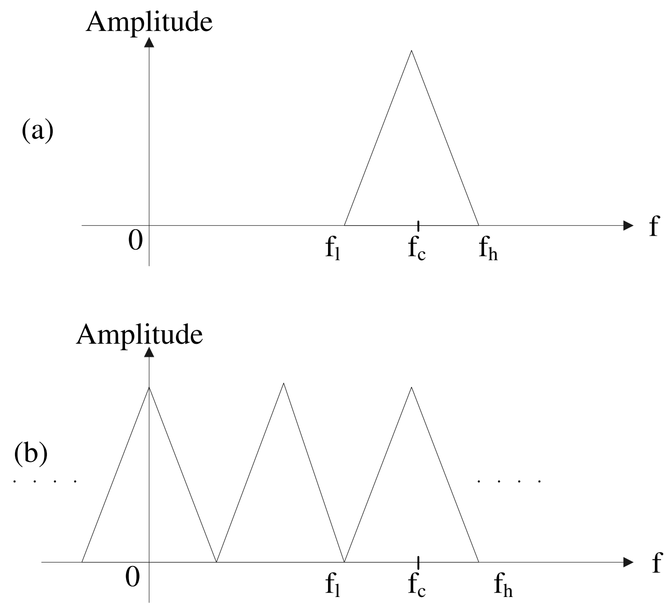


Figure 2.4: *(a) Bandpass signal (b) Spectrum replication after bandpass sampling i.e. sampling frequency is twice the bandwidth.*

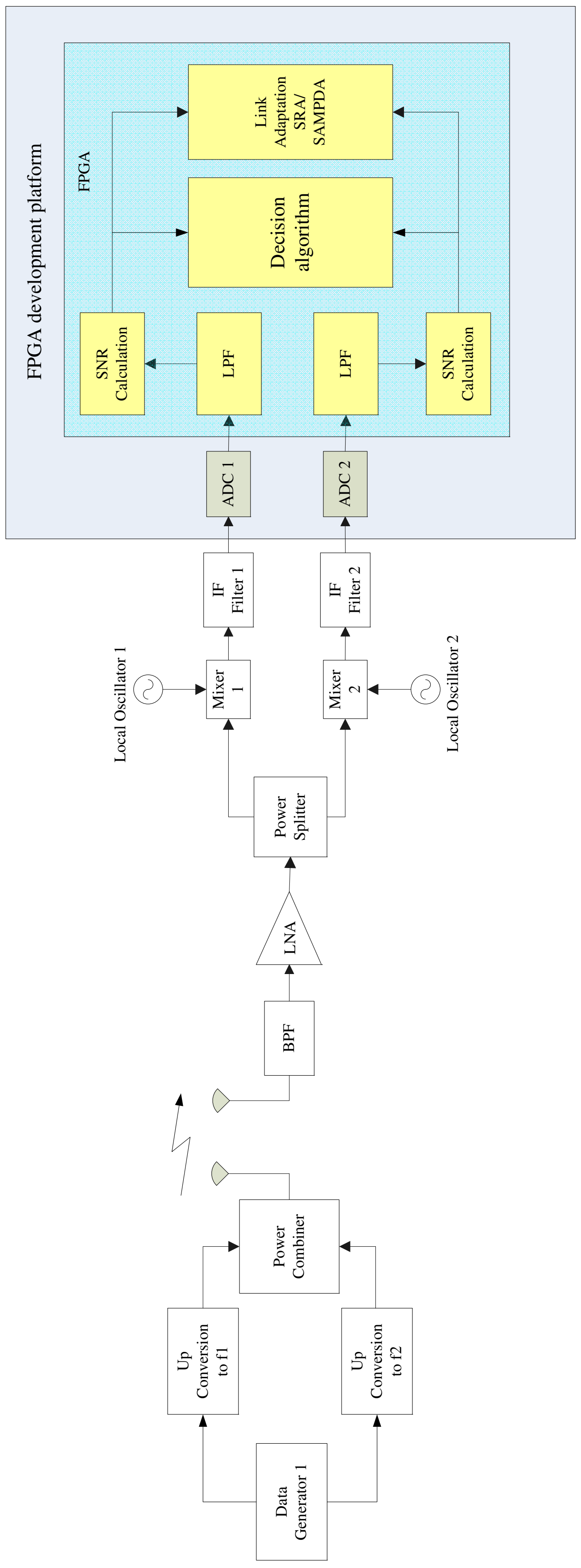
$$\frac{2f_h}{k} \leq f_s \leq \frac{2f_l}{(k-1)} \tag{2.1}$$

where k is the integer values between

$$2 \leq k \leq \frac{f_h}{f_h - f_l} \tag{2.2}$$

If there is a replica at the baseband level, this can be filtered out and used for the further processing stages. Thus, bandpass sampling automatically down converts the signal which is a major advantage. This relaxes the requirements on the mixer and down converter. In terms of the ADC, this means that an ADC with a lower sampling rate can be used thus saving power and cost. However, practical considerations regarding the location of the IF signal have to be made since most ADCs are designed to operate at half of the maximum sampling frequency. As the input frequency increases, the performance degrades. Also, band pass filters with steep roll offs are required in case of closely spaced carriers to prevent distortion [13].

Bandpass sampling has special importance in SDR based transceivers. The ideal SDR terminal digitizes the signal right after the antenna. Until this is achieved, sampling at IF has already



A 100 kHz signal is generated using a data generator and is upconverted to separate carrier frequencies. When received, they are individually sampled at an IF of 30MHz and then fed to the processing stages in the FPGA.

Figure 2.5: *Project setup with transmitter and RF front end using bandpass sampling.*

brought the ADC closer to the antenna as an intermediate step. In multi-mode receivers, this means that only one ADC can be used instead one for each mode thus saving cost [14]. Thus the setup for this project is shown in Figure 2.5.

A data generator transmits with a bandwidth of 100 kHz in the 2.4 GHz band. Using two separate upconverters, they are upconverted to separate carrier frequencies and are transmitted after passing through a power combiner. This signal is received by an antenna and is filtered using a Band Pass Filter (BPF) and passed through a Low Noise Amplifier (LNA). The signal is then split into two branches by using a power splitter. At each branch, the signal is then down converted to an Intermediate Frequency (IF) of 30 MHz. This is passed through an IF filter with a bandwidth of 600 kHz centered at 30 MHz. The bandwidth of this filter does not match our requirements but it was the one most suitable for this project. Because of this issue, the two carriers must be spaced at least 600 kHz apart when transmitting. After bandpass sampling, the signal is passed through a Low Pass Filter (LPF) which allows only the baseband spectrum to pass into the further stages. The SNR of the signal is then calculated to load the appropriate LA algorithm which is chosen by the Decision algorithm. The LPF and further stages are to be implemented on the FPGA.

To proceed toward implementing the system on an FPGA in a systematic manner, a well defined methodology is required. Various design methodologies and models exist such as the Y chart [15], Rugby Meta model [16], A^3 paradigm [17] etc. which define systematic ways to proceed toward implementation by defining domains and their abstraction levels. These methodologies could be used for DSP or FPGA based solutions. None of them are specifically targeted for FPGAs. In this project, the possibility of partial reconfiguration for implementing the LA algorithms is explored. Partial reconfiguration, which is of growing interest for SDR platforms has not been dealt with from a methodological point of view to the best of our knowledge. Therefore we propose a methodology to proceed in a systematic manner toward realizing partial reconfiguration for FPGA based systems.

2.3 The Proposed Methodology: ATtACk

We define four domains as **A**lgorithm, **T**ime, **A**rchitecture and **C**ommunication and thus the name ATtACk. The reason for choosing these domains is due to their importance mentioned in the A^3 Paradigm and Rugby-Meta model. This methodology is represented by Figure 2.6 showing the domains and the different phases. Each domain is divided into four phases starting from the ends moving toward the center.

1. **Specifications or Requirements:** In this phase, the initial specifications or requirements are described. These are broad level descriptions without going into finer details of the implementation. Based on the availability of components, some trade offs can be considered.

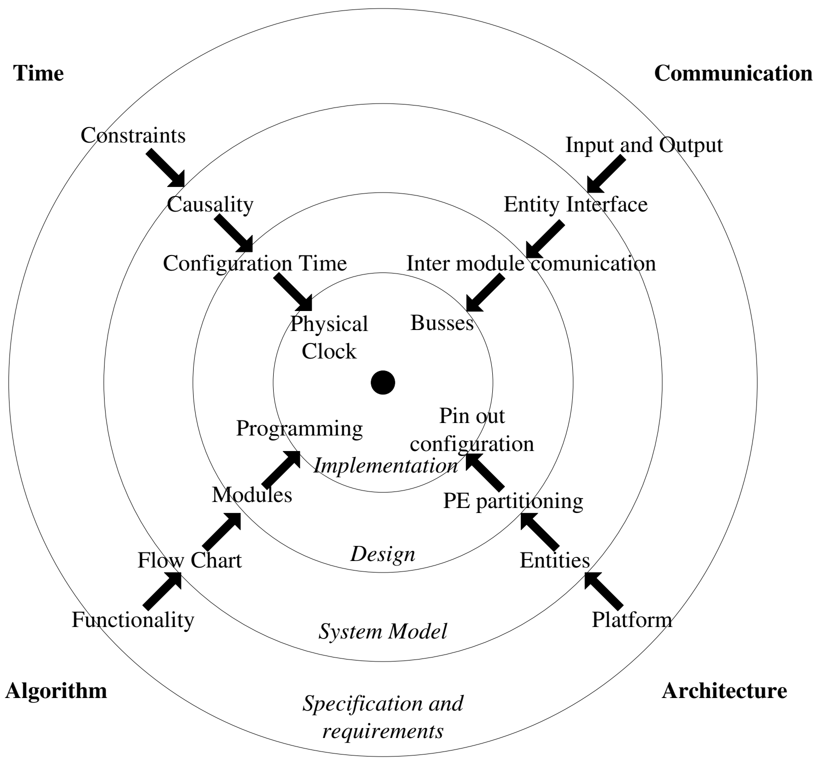


Figure 2.6: *The ATtACk methodology for implementing partially reconfigurable systems. The four domains are divided into four phases starting from the end toward the center to systematically proceed toward implementation. The names of the four phases are marked in italics.*

2. **System Model:** Once the specifications are defined, a system model can be created to have a closer look into the required setup. At this stage, practical considerations are taken into account such as chip interfaces or memory modules.

3. **Design**: After the System Model is ready, a design can be prepared from an implementation point of view. Here, actual implementation issues are considered so that the system can be successfully designed.

4. **Implementation:** The last phase is actual implementation based on the design. A high level language or specific vendor tools can be used for this purpose.

2.3.1 Algorithm

The aim is to achieve partial reconfiguration of an FPGA for a specific functionality and thus there will be a minimum of two or more algorithms to start with which have to be implemented. In the algorithm domain, the first phase is to describe the algorithms. In this phase, details such as the steps within the algorithms are not included. The basic understanding of it's functionality and its related background study is included. Then comes the second phase where the algorithms are studied and a flow chart is prepared by profiling the algorithms manually detailing every step of execution. This helps the third phase which is defining the modules. The flow chart can be divided into two modules: Static and Dynamic. The static module would be the common execution steps among the algorithms and is always running on the FPGA. The dynamic module is the one that has to be loaded onto the FPGA based on some computation and decision. If there are no common blocks, then this separation has to be done from a system level point of view, i.e. based on broad level functionalities which can be represented by block diagrams. Once the static and dynamic modules are separated, the next step is to proceed with implementation using high level languages like Handel-C, SystemC or SpecC. The designer may also use a hardware description language such as Very high speed integrated circuit Hardware Description Language (VHDL).

2.3.2 Architecture

This domain deals with the implementation target platform. The first phase is to define the requirements and decide the platform to be used. In case of a development board, factors such as input/output interfaces, memory requirements, FPGA specifications, debugging options using external peripheral connections can be specified. The next phase is to identify specific entities on the board that will be part of the actual system design, for example the FPGA and on-board memory that will be used for the reconfigurable architecture. These two phases are distinct only in case of implementing a design on a development platform. If a reconfigurable architecture is implemented in a real system, then the entities form a part of the requirement. In the next phase, the FPGA is divided into static and dynamic portions to facilitate partial reconfiguration. The static modules from the Algorithm domain will be mapped onto the static portions of the FPGA and the dynamic modules are implemented on the dynamic portion. This is important since the pin configurations will have a direct impact on how the modules have to be implemented. More on this is explained in Chapter 4. These limitations have to be kept in mind before proceeding to the final implementation. Specific place and route tools provided by the FPGA vendor can be used here to partition the FPGA during the actual implementation.

2.3.3 Communication

In this domain, the data and control communication that takes place in the entire reconfigurable system is considered. The first phase is to identify the input and output parameters of the platform from the functionality of the algorithm. The next phase is to understand the platform and identify the interfaces that are available among the different entities from the Architecture domain. This is essential since for a reconfigurable system, the programming files will be stored in an external memory and they have to be retrieved to configure the FPGA. The next phase is to define the inter module communication between the static and dynamic portions as identified in the Architecture domain. By doing so, data integrity will be maintained and the system can be reconfigured on the fly without affecting the data produced by the static modules. The next phase is to design the interfaces between the modules in the high level language or hardware description language for implementation.

2.3.4 Time

Time is an important domain to be considered since designs should be able to cope with the demands of the system in terms of execution speeds. The first phase is to define the constraints that are to be satisfied by the system. Once this is defined, the algorithm that has been described as a flow chart is used to extract the causality of events or execution steps. Here, if needed, steps that can be executed in parallel can be identified to reduce overall run time of the algorithm. Moving to the next phase, the configuration time has to be taken into account. If the dynamic modules as identified in the Architecture domain are to be loaded onto the FPGA, the time for this operation has to be estimated. The next phase is implementation and the actual clock cycles will be obtained here.

Validation forms an integral part of the methodology. At the end of each phase, the work done is validated so as to ensure that the next phase begins without any previous errors. In case errors are found, then corrections in that particular domain along with the domains it affects, have to be done. ATtACk can be used by any one who wishes to implement a reconfigurable architecture using an FPGA based on an SDR platform. In this project, ATtACk is used so as to suit the requirements of this project. With the proposed methodology, the first phase which is to analyze the requirements and specifications is presented next.

2.4 Initial Analysis, Specifications and Requirements

As proposed in the methodology, four domains and phases are considered along which we proceed step by step. In each domain, the requirements and specifications in our project are analyzed.

2.4.1 Algorithm: Functionality

In the Algorithm domain, the two algorithms, SRA and SAMPDA being considered are presented. These two algorithms are described in brief.

Simple Rate Adaptive algorithm (SRA)

SRA algorithm is proposed in [18] for a HiperLAN/2 (H/2) system and is a two step algorithm as shown in Figure 2.7. It maximizes the throughput while keeping the total transmission power equal to the H/2 system. In H/2, modulation schemes that are possible using OFDM are BPSK, QPSK, 16 QAM and 64 QAM. SRA improves the throughput by means of bit loading.

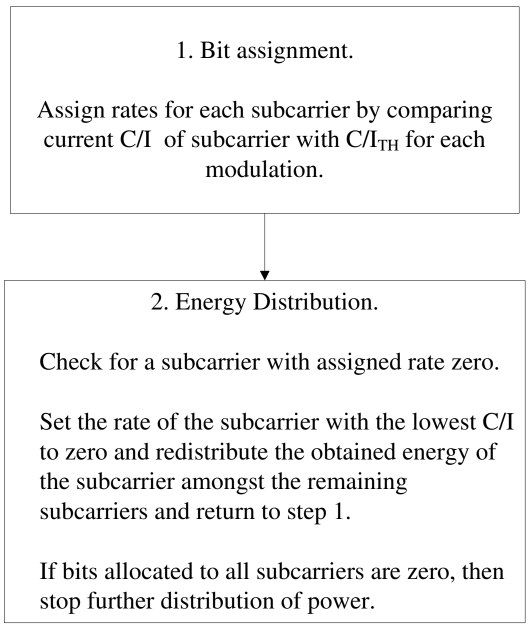

1. Bit assignment.
Assign rates for each subcarrier by comparing current C/I of subcarrier with C/I$_{TH}$ for each modulation.
2. Energy Distribution.
Check for a subcarrier with assigned rate zero.
Set the rate of the subcarrier with the lowest C/I to zero and redistribute the obtained energy of the subcarrier amongst the remaining subcarriers and return to step 1.
If bits allocated to all subcarriers are zero, then stop further distribution of power.

Figure 2.7: *SRA Algorithm showing only the broad level functionality [1].*

Bit loading is a technique in which different modulation schemes are assigned to individual subcarriers based on the instantaneous Carrier to Interference Ratio (C/I) conditions [1]. Since in this project interference is not considered, SNR is used in place of C/I. Power is equally distributed and a simple iterative bit loading scheme is used where predefined C/I thresholds are used for assigning a modulation scheme to each of the subcarriers [1].

The Subcarrier based Adaptive Modulation and Power Distribution Algorithm (SAMPDA)

Figure 2.8 shows this algorithm proposed in [11] based on SRA [18] and APDA [19]. This algorithm can significantly improve the spectral efficiency while keeping the complexity low. It starts with equal power distribution for all the subcarriers. The SNR for each subcarrier is compared with an SNR look up table to determine the number of bits and power that can be assigned for a subcarrier. Figure 2.8 shows the flow diagram of the SAMPDA algorithm.

SAMPDA is based on iterative mechanism and it improves the spectral efficiency while maintaining the target Bit Error Rate (BER) and does not exceed the transmit power threshold. The transmit power threshold is the maximum available transmit power for all subcarriers. "In each iteration, the power and the corresponding number of bits are loaded in the subcarrier at which there is a modulation mode demanding the minimum average incremental transmit power per incremental bit to guarantee the pre-designated BER performance" [19]. The iterations terminate when the transmit power threshold is reached or when all subcarriers transmit at the highest

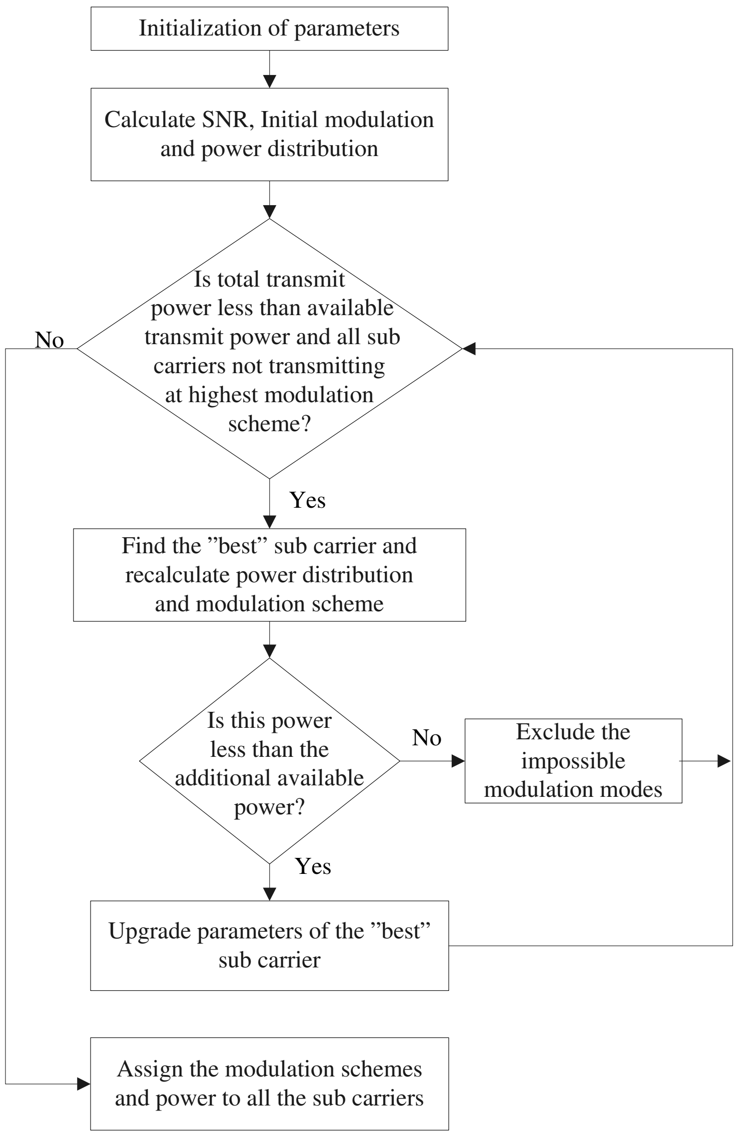

Initialization of parameters
Calculate SNR, Initial modulation and power distribution
Is total transmit power less than available transmit power and all sub carriers not transmitting at highest modulation scheme?
No
Yes
Find the "best" sub carrier and recalculate power distribution and modulation scheme
Is this power less than the additional available power?
No
Exclude the impossible modulation modes
Yes
Upgrade parameters of the "best" sub carrier
Assign the modulation schemes and power to all the sub carriers

Figure 2.8: *SAMPDA Algorithm showing only the broad level functionality [1].*

modulation level [1].

2.4.2 Architecture: Platform

The platform requirements are defined in the architecture domain. As discussed previously, at least two ADCs are required for the setup and a memory element to store the algorithms which will be loaded dynamically. For this project, we can either have the Nallatech board with a Xilinx Virtex-4 SX35 FPGA or the Altera board with a Stratix II FPGA. Both the platforms have two ADCs and two DACs. The main requirement of this project is to have dynamic reconfigurability on the fly. i.e. without disrupting the current operation of the FPGA. This is possible on the Xilinx FPGA only. On a Stratix II FPGA, dynamic partial reconfiguration can be done only for specific operations such as multiply, divide and bandwidth selection settings [20]. Thus the Nallatech platform is used for this project. However, this platform does not have on-board memory which can be used for configuring the FPGA and thus an external memory source such as that of a computer has to be used. The block diagram of the Nallatech board is shown in Figure 2.9.

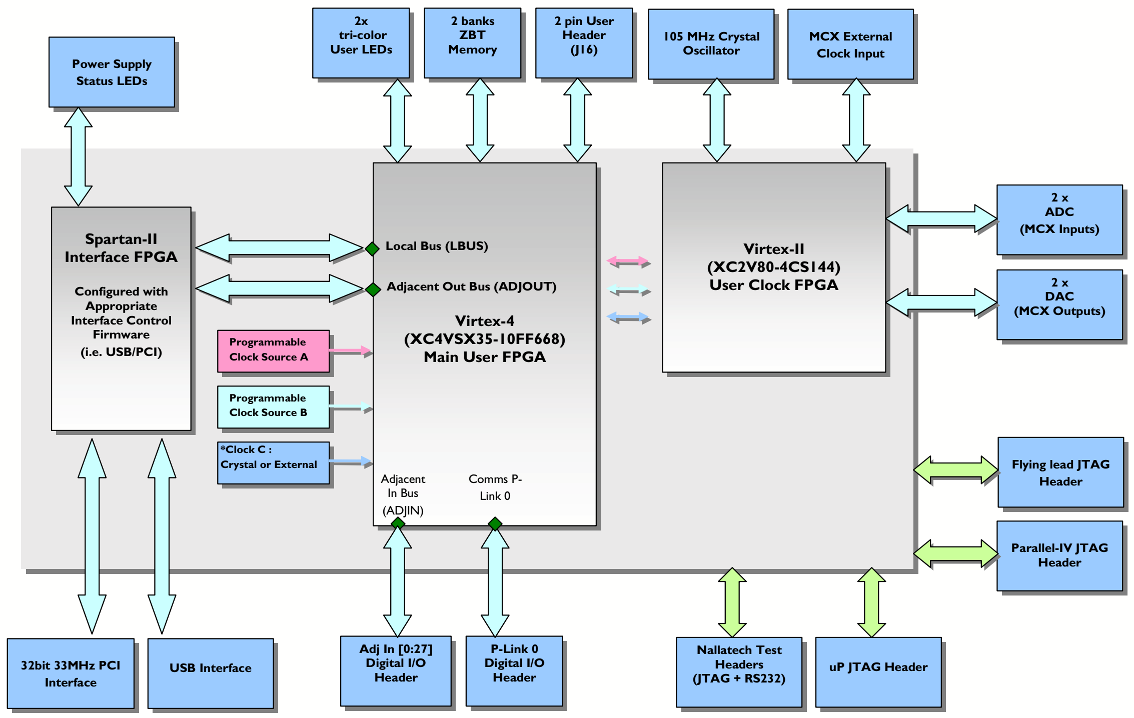

Power Supply Status LEDs
2x tri-color User LEDs
2 banks ZBT Memory
2 pin User Header (J16)
105 MHz Crystal Oscillator
MCX External Clock Input
Spartan-II Interface FPGA
Configured with Appropriate Interface Control Firmware (i.e. USB/PCI)
Local Bus (LBUS)
Adjacent Out Bus (ADJOUT)
Virtex-4 (XC4VSX35-10FF668) Main User FPGA
Virtex-II (XC2V80-4CS144) User Clock FPGA
2 x ADC (MCX Inputs)
2 x DAC (MCX Outputs)
Programmable Clock Source A
Programmable Clock Source B
*Clock C : Crystal or External
Adjacent In Bus (ADJIN)
Comms P-Link 0
Flying lead JTAG Header
Parallel-IV JTAG Header
32bit 33MHz PCI Interface
USB Interface
Adj In [0:27] Digital I/O Header
P-Link 0 Digital I/O Header
Nallatech Test Headers (JTAG + RS232)
uP JTAG Header

Figure 2.9: *Nallatech Development Kit being used for this project (Figure 2 in [21]).*

This board features three Xilinx FPGAs.

- A Virtex-4 FPGA (XC4VSX35-10FF668) for user designs also called as User FPGA.
- A pre-configured Spartan-II FPGA to handle Peripheral Component Interconnect (PCI) and Universal Serial Bus (USB) interfaces called as the Interface FPGA.
- A Virtex-II FPGA for clock management.

The interface FPGA communicates with the User FPGA via a dedicated communications bus. This platform has two ADCs and two DACs.

2.4.3 Communication: Input and Output

The input to the platform is a live signal received by an antenna as seen in Figure 2.10. A live signal that varies with time in terms of its signal power is received by an antenna. Since the algorithms adapt the transmission parameters based on the SNR of the received signal, it is calculated at each of these two branches and are fed as an input to the LA algorithm which is loaded dynamically. The output of the algorithm is bit and power loading for each of the two signals. Note that in a practical system, the SNR calculation is done by the receiver and fed back to the transmitter. Based on this feedback, the transmitter adapts itself. However, in this project the SNR calculation and link adaptation is implemented on the same platform.

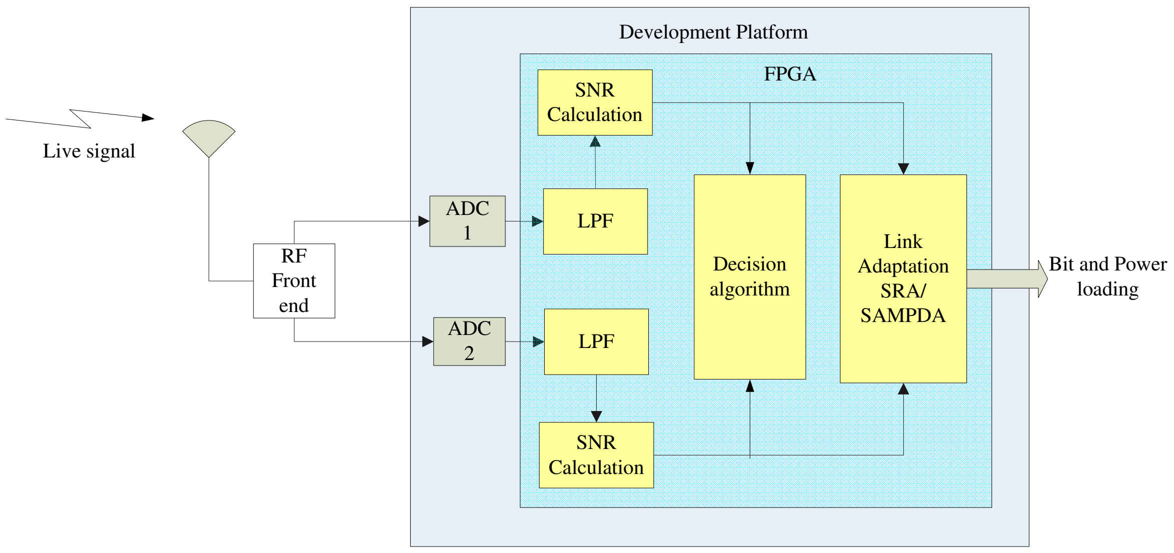


Figure 2.10: *Input and Output of the development platform. A live signal is fed to the system and the output of the LA algorithm is bit and power loading for each carrier.*

2.4.4 Time: Constraints

The execution time (latency) of the algorithms are known and shown in Table 1.1. The time constraint that is considered also includes the time taken to reconfigure the hardware. This has to be found out after the actual implementation is done and thus, in this project, timing constraint from a reconfiguration point of view is not considered.

With the requirements and specifications ready, the next phase which is the System Model is presented next.

2.5 System Model

In this phase the system model being considered in each of the four domains mentioned in the methodology is defined.

2.5.1 Algorithm: Flow chart

In the previous phase, the two algorithms SRA and SAMPDA being considered in this project have been described. Now moving a step further, every step of execution within the algorithms is detailed. Figure 2.11 shows the flow chart of SRA and Figure 2.12 shows the flow chart of SAMPDA [1]. These flow charts are created by manually studying the MATLAB codes from [11] where the algorithms were first simulated. These flow charts show the steps from a stand alone point of view. Meaning, as an independent block without any interfaces to the rest of the transceiver chain.

SRA

The SRA algorithm is shown in Figure 2.11 begins by initializing (1) the parameters such as number of subcarriers and threshold SNR for the modulation schemes. A condition is checked (2) to see if all subcarriers are not already at the highest modulation scheme and that if any subcarrier is not loaded. Then power is equally distributed to all the subcarriers and based on individual SNR of each subcarrier, a modulation scheme is assigned (3). For those subcarriers that do not support even the lowest level of modulation (BPSK), the transmission is switched off and its power is distributed among the remaining subcarriers (4,5). The algorithm terminates after all the subcarriers have been adapted (6).

SAMPDA

In SAMPDA shown in Figure 2.12, the initializations (1) begin with the number of subcarriers and threshold SNR. The first step in this algorithm is same as in SRA. Then in the iterative loop, a condition (2) is checked if the total transmit power allocated is available and that all subcarriers do not transmit at the highest modulation level. The algorithm searches for the best subcarrier (3,4). A best subcarrier is the one that requires the least amount of incremental power to go to the next modulation scheme. Due to this feature, this algorithm performs more iterations than SRA and also performs better than SRA in mid SNR regions (between 10-40 dB). The algorithm terminates (5) when the power requirement exceeds the availability or when all subcarriers are transmitting at the highest modulation level.

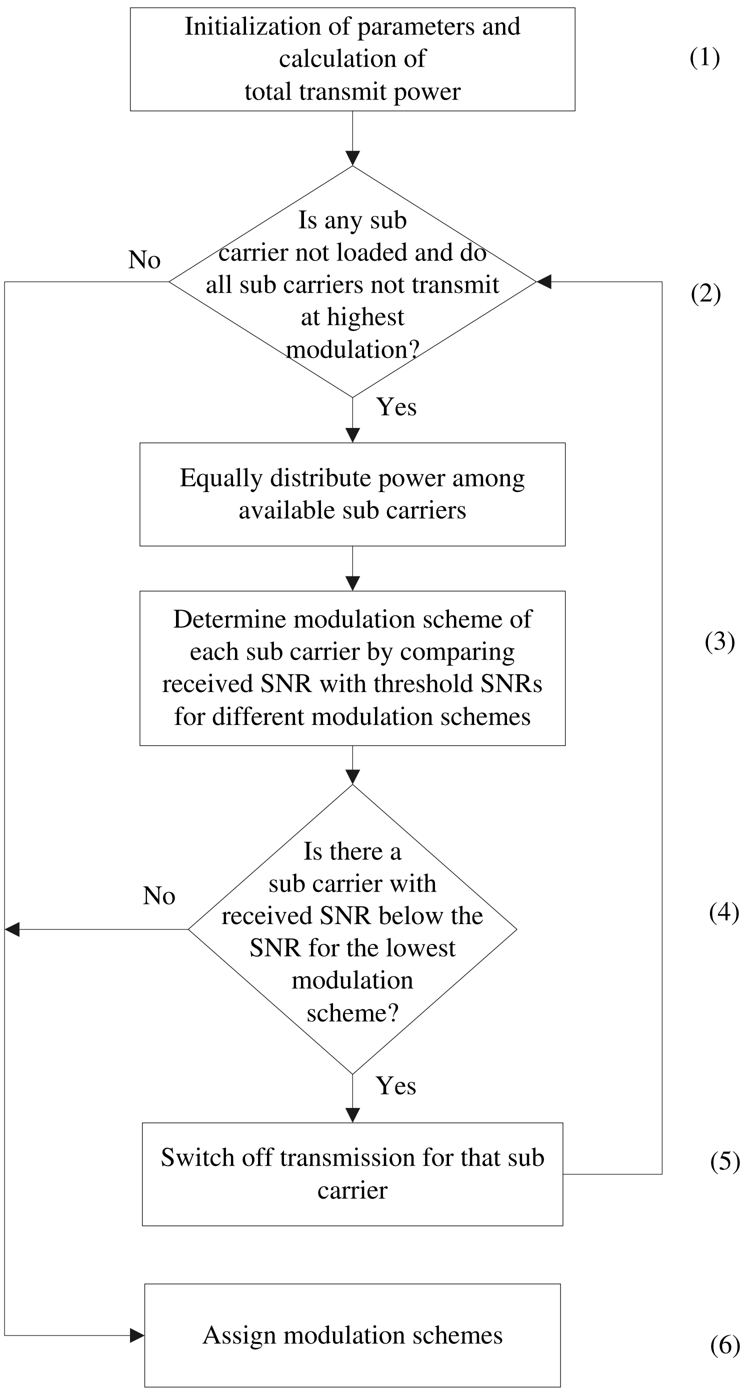

Initialization of parameters and calculation of total transmit power
(1)
Is any sub carrier not loaded and do all sub carriers not transmit at highest modulation?
No
(2)
Yes
Equally distribute power among available sub carriers
Determine modulation scheme of each sub carrier by comparing received SNR with threshold SNRs for different modulation schemes
(3)
Is there a sub carrier with received SNR below the SNR for the lowest modulation scheme?
No
(4)
Yes
Switch off transmission for that sub carrier
(5)
Assign modulation schemes
(6)

Figure 2.11: *Flow chart of SRA algorithm with different steps of execution [1]. This algorithm distributes power equally and assigns the modulation scheme by comparing the received SNR with threshold SNR for each modulation scheme. Power redistribution is done if any surplus power is obtained by switching off a subcarrier.*

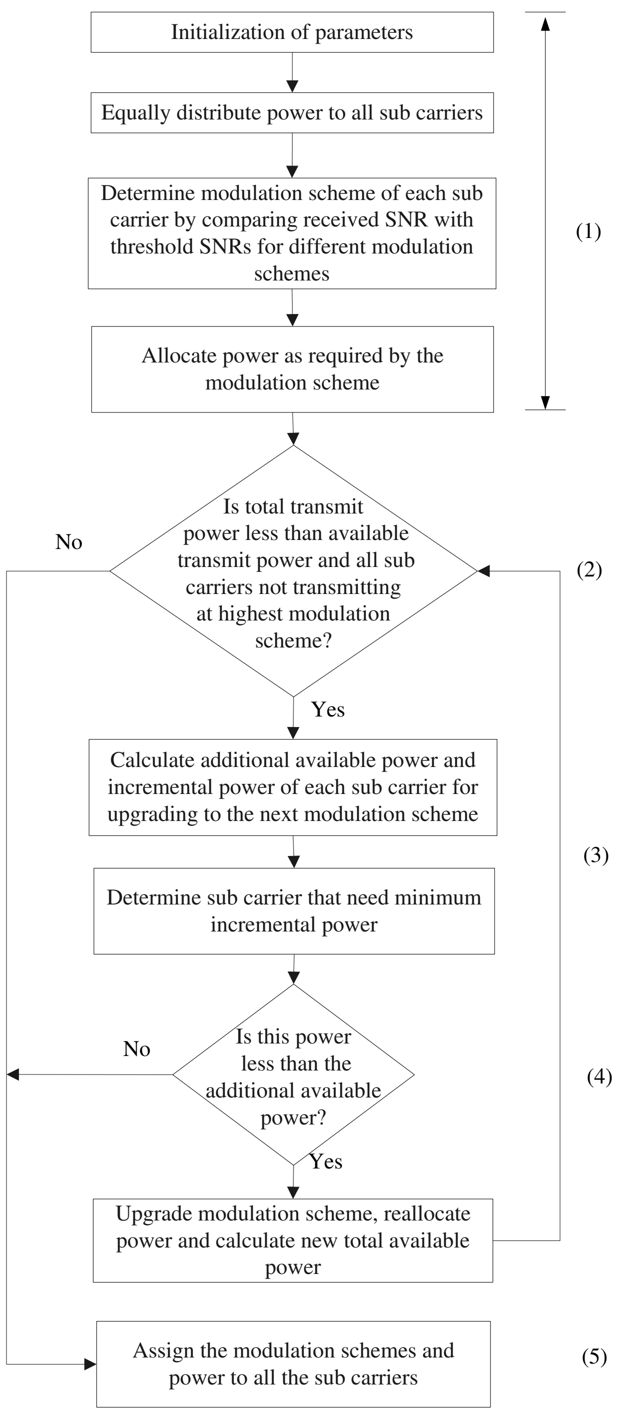

Initialization of parameters
Equally distribute power to all sub carriers
Determine modulation scheme of each sub carrier by comparing received SNR with threshold SNRs for different modulation schemes
Allocate power as required by the modulation scheme
(1)
Is total transmit power less than available transmit power and all sub carriers not transmitting at highest modulation scheme?
No
Yes
(2)
Calculate additional available power and incremental power of each sub carrier for upgrading to the next modulation scheme
Determine sub carrier that need minimum incremental power
(3)
Is this power less than the additional available power?
No
Yes
(4)
Upgrade modulation scheme, reallocate power and calculate new total available power
Assign the modulation schemes and power to all the sub carriers
(5)

Figure 2.12: *Flow chart of SAMPDA algorithm with different steps of execution [1]. The initial steps are same as in SRA. Then the algorithm finds the best subcarrier which is the one that needs the minimum additional power to go to the next modulation scheme. It assigns power as required for the particular modulation scheme.*

2.5.2 Architecture: Entities

Here the specific entities required in the design of the system that are available on the development platform are identified and a deeper study is done in terms of how they can be used for a specific purpose. On the Nallatech board, the ADCs and the Xilinx Virtex-4 SX35 FPGA are used. There is no on board memory that can be used to store the configuration files for the FPGA. Due to this issue, a computer memory is used which is interfaced with the board through the Interface FPGA. The *bit* files for partial reconfiguration are stored in the computer's memory and retrieved when the need arises. Thus the two ADCs, the User FPGA and the Interface FPGA are used in this project.

2.5.3 Communication: Entity Interfaces

To connect the development platform and the computer, either the Joint Test Action Group (JTAG) interface, Universal Serial Bus (USB) or Peripheral Component Interconnect (PCI) interface can be used. The User FPGA connects to these interface through the Interface FPGA. This is shown in Figure 2.13.

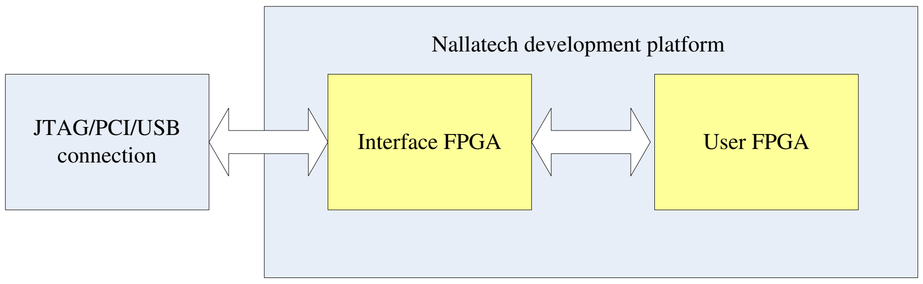


Figure 2.13: *A JTAG, PCI and USB connection are available to connect to the development board. The User FPGA connects to them through the Interface FPGA.*

The function of the interface FPGA is to facilitate the connection between the computer and the User FPGA. A PCI and USB communications core is provided by Nallatech to communicate with the FPGA which can be used in the application design. This core provides the interface in two ways [21]:

1. Memory Map Interface: This is a simple interface to perform single word read/writes in registers or RAM.
2. Direct Memory Access (DMA) interface: This interface is used when large amounts of data have to be transferred to and from the application.

The configuration files are stored at a memory location which can be accessed only by using the PCI and USB communications core and so the USB interface is chosen. Further, DMA interface is used since the configuration *bit* file used to configure the FPGA is large in size and is stored in a computer. To facilitate communication between the User FPGA and Interface FPGA, the Interface Communication Bus by Nallatech provides a pre-defined communications protocol which has to be included in the design as a drop-in IP core.

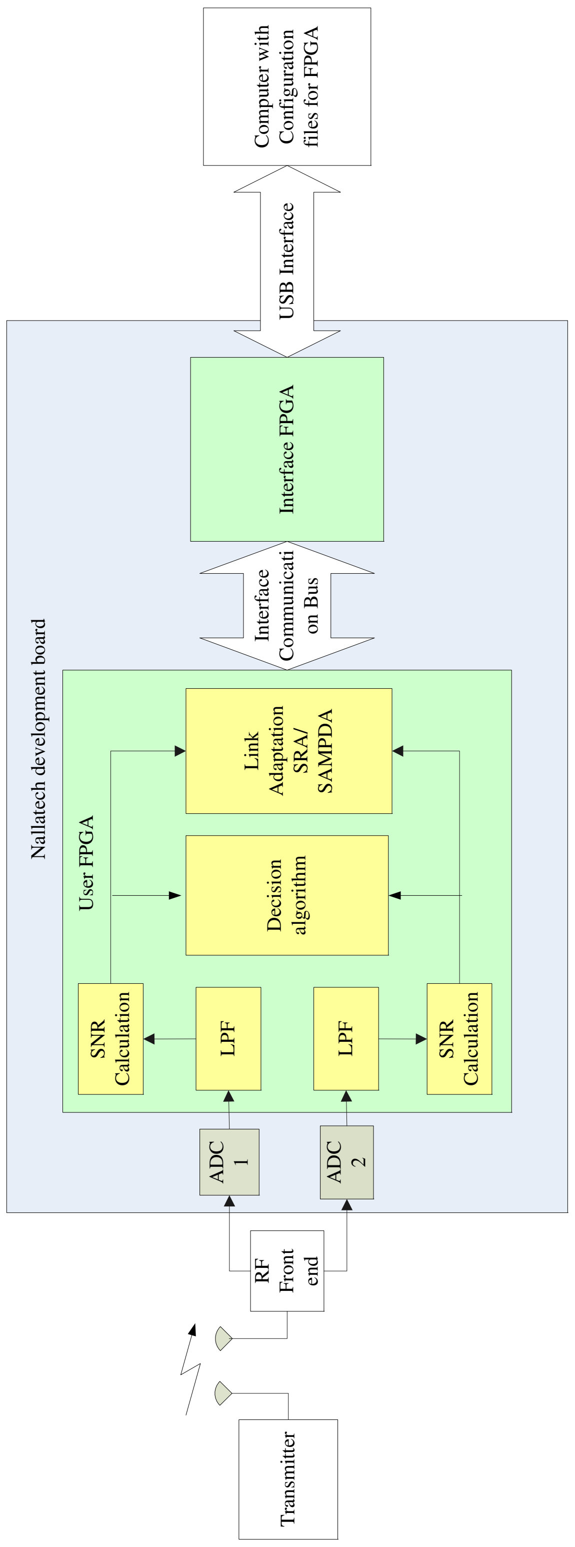
2.5.4 Time: Causality

Using the flow chart, the events that take place sequentially can be identified. If parallelism has to be included, blocks which do not have data dependencies, can be executed in parallel. This strategy is employed when having time as a constraint. However, in this project as mentioned earlier, timing constraints are not considered and so no parallelism is considered. Reduction of execution time was done in [1] by introducing parallelism and it was seen that as parallelism increases, area also increases. In this project, the FPGA has to handle operations such as filtering, SNR calculation and link adaptation. From Table 1.1 it can be seen that SAM-PDA algorithm already occupies 94% of the area on the FPGA and so increase in area is not feasible. The area requirements of the design without any parallelism are first considered and if any surplus exists, parallelism options are explored.

2.6 Conclusions

In this chapter, the setup of this project employing bandpass sampling has been introduced to calculate the SNR of two carriers. To implement reconfigurable designs, we proposed a methodology called ATtACk. Then, based on this methodology, the first two phases of Specifications and System Model are covered. In the Specifications phase, an overview of the requirements and specifications is taken. Then, moving a step closer toward implementation, the System Model is considered. In this phase the LA algorithms, the entities to be used on the development platform and their interfaces are presented. Based on this analysis, the system model is shown in Figure 2.14.

The data transmitter transmits a signal having a bandwidth of 100 kHz in the 2.4 GHz band at two separate carrier frequencies. This is transmitted from a single antenna using a power combiner. This signal is received by an antenna and are passed through the BPF and LNA. The two carriers are separately down converted to an IF frequency of 30 MHz. Then after IF filtering this signal is sampled and filtered with a LPF. The SNR is calculated and based on the decision algorithm, the LA algorithm is loaded through the Interface FPGA via the USB interface with the computer. The next phase is Design which is discussed in the next chapter.


Computer with Configuration files for FPGA
USB Interface
Nallatech development board
Interface FPGA
Interface Communicati on Bus
User FPGA
Link Adaptation SRA/ SAMPDA
Decision algorithm
SNR Calculation
LPF
LPF
SNR Calculation
ADC 1
ADC 2
RF Front end
Transmitter

The transmitted signal is received and downconverted to an IF after which it is sampled. After passing through a LPF, the SNR is calculated and fed to the Decision algorithm. To retrieve configuration bit files from the computer, the Interface FPGA is used which connects the User FPGA to a computer through the USB interface.

Figure 2.14: *System Model for this project showing all entities and interfaces.*

Chapter 3

System Parameters and Design

3.1 Introduction

In the previous chapter, the ATtACk methodology and the project setup is introduced. In this chapter, the concept of SNR measurement is first introduced and then the system design and it's parameters such as the sampling frequency of the ADC and the design of the LPF. Continuing as per ATtACk methodology to the Design phase the concept of partial reconfiguration and it's intended operation in this project is presented.

3.2 Calculation of SNR

The instantaneous average SNR of the two carriers transmitted is to be calculated which acts as the criteria for choosing the algorithm. In order to evaluate the SNR of the signal impaired by the channel, several types of estimations are described in [22] which are useful in practical receiver systems. However, in this project channel estimation is not employed to keep the complexity of the receiver low.

Hence in keeping the scope of the project and requirements in view, a practically feasible way to measure SNR is required. A procedure to calculate signal power and noise power individually is considered. The signal power is calculated by measuring the Received Signal Strength Indication (RSSI) [23]

3.2.1 Signal Power Calculation through RSSI measurement

To calculate the signal power, a measurement technique which gives the RSSI value at the antenna is used . This method calculates the signal strength at the antenna as given in Equation 3.1 [23].

$$\text{RSSI} = 10^{\frac{-G_{rf}}{10}} \cdot \frac{12567 \cdot V_c^2}{2^{2B} \cdot R} \left(\frac{1}{N} \sum_{n=0}^{N-1} \mid Y[k,n] \mid \right)^2 \text{mWatt} \tag{3.1}$$

where

G_{rf} is analog gain from antenna connector to ADC input
V_c is input clip level of ADC
B is precision of ADC
R is input resistance of ADC
N is the total number of samples
Y[k,n] is n^{th} sample at the ADC output within signal k

Considering the project setup, Equation 3.1 is simplified by substituting values of gain, input resistance of ADC etc. as per the specifications of ADC and other RF components. The converter AD6645 used in this project has an input resistance of 50Ω, $1.1Volt$ of input clip level and has a precision of 14 bits [24]. The analog gain of the receiver G_{rf} in dB is calculated by measuring the input and output power levels which gives $42dB$ approximately. This is explained in Appendix B.1. By using these values in Equation 3.1 the RSSI calculation can be computationally reduced as shown in Equation 3.2:

$$\text{RSSI} = 0.7148 \cdot 10^{-10} \left(\frac{1}{N} \sum_{n=0}^{N-1} \mid Y[k,n] \mid \right)^2 \text{mWatt} \tag{3.2}$$

Hence RSSI is calculated by taking the summation of the absolute value of each sample within the signal period. This is divided by the number of samples N and the value obtained is squared. From an implementation point of view this procedure requires an absolute value estimator, accumulator and a multiplier [23]. RSSI also reflects the effects of propagation loss. Implementation issues regarding this method are discussed in detail in the following chapter.

In a real time communication system noise has a very wide meaning and it can be added to the received signal in a number of ways but in this project, it is limited to the calculation of noise floor of the receiver.

3.2.2 Noise Power Measurement

The RF components in the receiver add thermal noise which is because of thermal agitation of electrons to the received signal [25]. The received analog signal is converted into digital signal by the ADC and noise added by the digital signal processing components can be neglected. So, calculation of the noise floor gives the estimate of noise power added by the receiver components which is given by Equation 3.3[25].

$$Noisefloor, dBm = 10 \cdot log\,(P)\,\text{dBm} + NF_{tot}\text{dB} \tag{3.3}$$

where

NF_{tot} is total Noise Figure of the receiver.

The thermal noise power (P) is given by Equation 3.4[25].

$$P = k \cdot T_e \cdot B_s \tag{3.4}$$

where

B_s is the Signal Bandwidth
k is the Boltzman's constant
T_e is the Effective Temperature

Therefore,

$$\textit{Noise floor, dBm} = 10 \cdot log\,(k \cdot T_e)\,\text{dBm} + 10 \cdot log\,(B_s)\,\text{dB} + NF_{tot}\text{dB} \tag{3.5}$$

The Noise Figure (NF) of a component is defined as the ratio of the SNR at the input to the output of that component. The amount of noise added to the signal by a component can be known from the NF of that component. Once the gain and NF of the devices in receiver chain is known, equivalent NF (or NF_{tot}) for a receiver containing n number of devices in cascade is calculated by Equation 3.6[25]. The situation is as shown in Figure 3.1

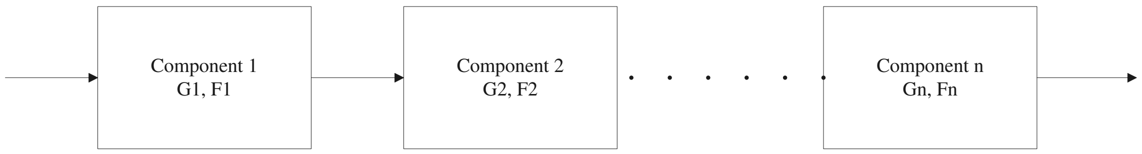


Figure 3.1: *Example of cascaded connection of components in a system with Gain and Noise figures.*

$$NF_{tot} = NF_1 + \frac{NF_2 - 1}{G_1} + \frac{NF_3 - 1}{G_1 \cdot G_2} + ... + \frac{NF_n - 1}{G_1 \cdot G_2 \cdot G_3 \cdots G_{n-1}} \tag{3.6}$$

Where NF_i refers to the NF of i^{th} component and G_i denotes gain of i^{th} component. The calculation of the noise floor of the receiver chain considered in this project is presented in the next chapter.

As explained before, the signal power is measured by the RSSI measurement as described in Section 3.2.1. Noise power is calculated by taking the noise floor of the receiver. Then, signal to noise ratio is calculated by taking the ratio of signal power to noise power.

3.3 System Parameters

Figure 2.5 shows all the components that are needed for the setup. The analog components are taken from the Antenna Laboratory at the University to best match the requirements. The parameters of the ADC and LPF is presented next.

3.3.1 ADC Sampling Frequency

As seen in Section 2.2.1 Bandpass Sampling is used in this project. The sampling frequency has to be chosen such that the replicated frequency components do not undergo aliasing. The sampling frequencies possible on the ADC are 20 MHz, 25 MHz, 30 MHz, 33.33 MHz, 40 MHz, 45 MHz, 50 MHz, 60 MHz, 66.66 MHz, 70 MHz, 75 MHz, 80 MHz, 90 MHz, 100 MHz and 120 MHz [21]. The IF filter used has a bandwidth of 600 kHz at 30 MHz of center frequency and the transmitted signal has a bandwidth of 100 kHz. The situation is shown in Figure 3.2.

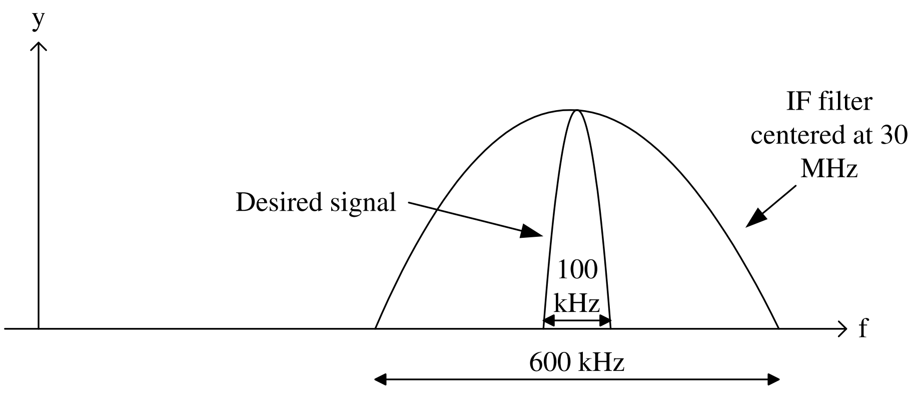


Figure 3.2: *100 kHz signal filtered by an IF filter centered at 30 MHz with a bandwidth of 600 kHz.*

Thus the 100 kHz signal ranges from 29.95 MHz to 30.05 MHz. From Equation 2.2, k is the integer values between

$$2 \leq k \leq \frac{30.05MHz}{0.1MHz} \tag{3.7}$$

$$2 \leq k \leq 300.5 \tag{3.8}$$

By using a lower value of k the spacing between the spectrum replicas increases thus allowing an anti alias filter with a smooth roll off. In this case, the IF filter has a bandwidth of 600 kHz but the signal is only 100 kHz. Taking k as 3, and from Equation 2.1.

$$\frac{2f_h}{k} \leq f_s \leq \frac{2f_l}{(k-1)} \tag{3.9}$$

$$\frac{2 * 30.05MHz}{3} \leq f_s \leq \frac{2 * 29.95}{2} \tag{3.10}$$

$$20.033MHz \leq f_s \leq 29.95MHz \tag{3.11}$$

Thus any sampling frequency between 20.033 MHz and 29.95 MHz can be chosen. Looking at the available sampling frequencies, the possible sampling frequency is 25 MHz. Taking a higher value of k, the sampling frequency required becomes less than 20 MHz which is not possible by the ADC used.

3.3.2 Low Pass Filter

After bandpass sampling a replica of the spectrum of the signal is obtained centered at 0 Hz. To extract this, a low pass filter is needed. The signal has a bandwidth of 100 kHz which appears centered across 0 Hz and so a filter is needed with a one sided bandwidth of 50 kHz. This LPF is designed using the Filter Design and Analysis Toolbox from MATLAB by using a windowing technique and entering the values needed. The FDA tool generates linear phase filter. The filter designed is a Direct Form FIR filter and the parameters entered are:

- Order: 32
- Window: Hamming
- Sampling frequency: 25 MHz
- Cut off frequency: 50 kHz (Since it is two sided at 100 kHz)

The coefficients are directly obtained from the tool and the filter response is tested in the next chapter.

3.4 Reconfigurability in the FPGA

In this project, the feasibility of a partially reconfigurable architecture for LA algorithms on an FPGA is to be analyzed. The need and importance of such a system has been elaborated in Chapter 1. In this project, partial reconfiguration is explored which means that only a portion of the FPGA is reconfigured while the rest of the FPGA is active and running. By using partial reconfiguration, a new design can be loaded onto a particular area without resetting or reconfiguring the whole device. The advantages of partial reconfiguration are [26]:

- Reconfiguration on the fly, that is, during runtime.
- Efficient use of hardware.
- Less power consumption due to less hardware usage.

Reconfiguration of the FPGA can be classified in three categories: endo-reconfigurable and exo-reconfigurable and a hybrid approach [27].

- Endo-reconfigurable: In this system, the FPGA platform has a facility to store reconfiguration files in a memory element on the platform itself which is retrieved whenever partial reconfiguration is required.
- Exo-reconfigurable: In this system, the reconfiguration files are stored in an external location such as a computer.
- Hybrid approach: This system is both endo and exo-reconfigurable.

As mentioned earlier, the platform being used in this project does not have an on-board memory to store configuration files and thus a computer is used for this purpose. Hence, the system in this project is exo-reconfigurable. The criteria to choose which algorithm has to be loaded onto the FPGA is explained next.

3.4.1 Decision Algorithm for Loading the LA Algorithm

As mentioned earlier, based on the criteria of the number of subcarriers and average received SNR, an algorithm has to be loaded on the FPGA. The basis of this algorithm is Table 1.1 and Figure 1.3. The SAMPDA algorithm occupies 94 % of the area on the FPGA when the number of subcarriers is 16 whereas for SRA, it is 35 %. Thus, if the number of subcarriers goes beyond 16, it is inevitable to use SRA for the available setup. If it is less than 16, then the average SNR is used to decide which algorithm has to be loaded. Between SNR of 10 dB and 40 dB, SAMPDA has to be loaded since it has better spectral efficiency. The decision algorithm is shown in Figure 3.3. In this project, only two subcarriers are considered and thus the first condition to check the number of subcarriers is always false. Only the average SNR is used to load one of the algorithm.

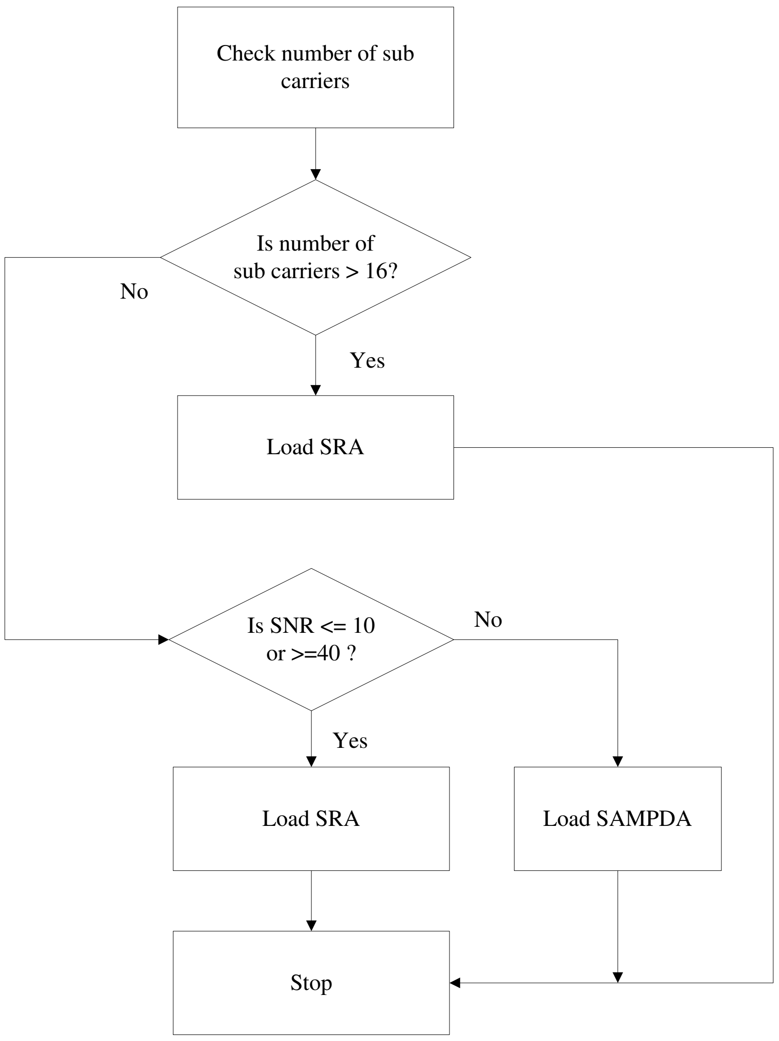


Figure 3.3: *Decision algorithm to load the appropriate algorithm. Based on the SNR values and number of subcarriers, a LA algorithm is loaded. However, the condition to check number of subcarriers is not used in this project since they are limited to two.*

3.5 Design

Continuing with the ATtACk methodology, the Design phase is now considered.

3.5.1 Algorithm: Modules

For partial reconfiguration, Static and Dynamic Modules exist. The Static Module contains the blocks that are common to the entire system and remain on the FPGA and the Dynamic Module is the reconfigurable part. Based on this consideration, it is see that the step to determine the modulation schemes based on the received SNR of each of the carriers is in both the algorithms as shown in Figure 2.11 and Figure 2.12. However, this step is not considered as part of the Static module due to implementation constraints described in the next chapter. So, as per the methodology, the modules are differentiated based only on the system functionality. As shown in Figure 2.5, the LPF and SNR calculation have to be implemented on the FPGA. Then, based on the Decision algorithm, the appropriate LA algorithm is executed on the FPGA. Thus the LPF, SNR calculation and Decision algorithm are always a part of the system and can be termed as the Static Module. The LA algorithms form the Dynamic module.

3.5.2 Architecture: Processing Element Partitioning

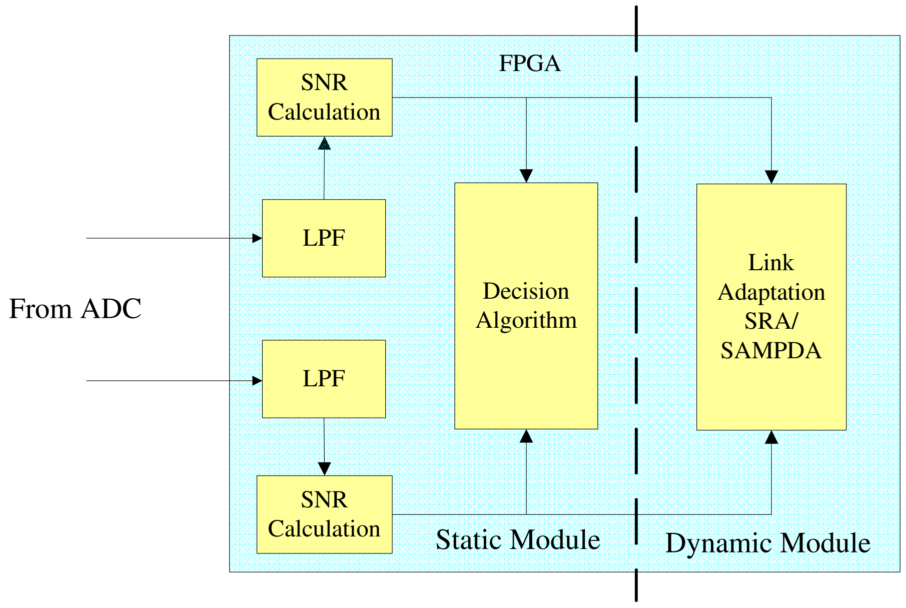


Figure 3.4: *The implementation is differentiated into Static and Dynamic modules. The Static module consists of the blocks that do not change. The LA algorithms form the Dynamic module.*

The processing element is the FPGA being used for implementing the design. In the Algorithm domain, the modules are separated as Static and Dynamic. A requirement of implementing a reconfigurable system on a Virtex-4 FPGA is that the partitions on the FPGA have to be rectangular and occupy 16 CLB rows in height or its multiple rather than the entire height of the device as in earlier FPGAs like Virtex-II [26] [28]. The system is partitioned into Static and Dynamic modules as shown in Figure 3.4 which is a schematic representation. The LPF, SNR calculation block and the decision algorithm always remain a part of the system (called as the Static Module). The LA algorithm which is dynamically loaded onto the FPGA is the only block that has to be reconfigured (called as the Dynamic Module) and hence this system is called a partially reconfigurable (PR) system. The modules identified now are from the system's point of view.

3.5.3 Communication: Inter Module Communication

The design is differentiated as Static and Dynamic modules which need to communicate. The development platform being used has a Xilinx Virtex-4 FPGA and Xilinx provides two methods to partially reconfigure anFPGA:

- Module-Based Partial Reconfiguration: As the name suggests, this method is used when a module on the hardware has to changed.
- Difference-Based Partial Reconfiguration: This method is used when there is a small change in the design such as changing the pin connections.

In this project, the Module-Based Partial Reconfiguration method is used since the whole LA algorithm has to be reconfigured.

3.5.3.1 Module Based Partial Reconfiguration (MBPR)

MBPR is further classified in two categories. First, where there is no communication between the modules and second, where some communication exists between the modules. The SNR values have to be fed as an input to the LA algorithm and hence some communication does exist between the Static and Dynamic modules. Communication between these modules is facilitated by bus macros which act as data paths. By using them, routing resources are fixed between the Static and reconfigurable (Dynamic) module. These resources do not change during reconfiguration.

The Static module provides the SNR value of both the received signal to the Dynamic module. To allow this communication, bus macros are used that assign routing resources for data flow between the two modules. The bus macro acts as a fixed routing bridge and specifies the exact channels that are used for communication [26].

3.5.4 Time: Configuration time

In the Time domain, no constraints are considered and so during the Design phase, this domain is not considered.

Based on the discussion in the Design phase, the implementation is shown in Figure 3.5. The left side is the Static module which comprises of the LPF, SNR calculation and the Decision algorithm. The algorithms form the Dynamic module. The Static module provides the SNR values to the LA algorithms through the Bus Macros.

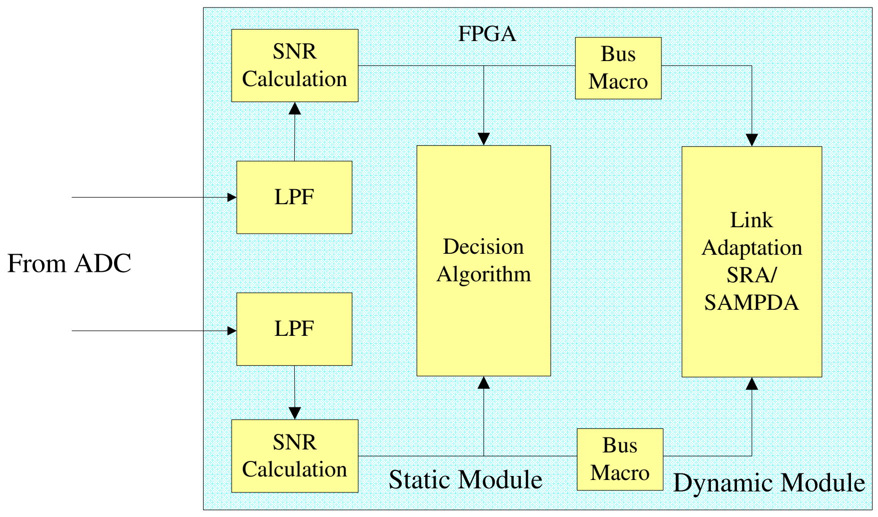


Figure 3.5: *Communication using Bus Macro interface between the Static and Dynamic modules to transfer the SNR values.*

3.6 Conclusions

In this chapter, a simple scheme for calculating the SNR calculation is presented along with the system parameters such as the sampling frequency of the ADC which is 25 MHz and the design of the LPF. The concept of partial reconfiguration is presented. Then, following the ATtACk methodology, the Design of the system is presented. The implementation is split into Static and Dynamic modules where the LA algorithms form the Dynamic module. Communication between the modules using bus macros is also presented. The next phase is Implementation which is discussed in the next chapter.

Chapter 4

Implementation Considerations and System Realization

4.1 Introduction

In the previous Chapter, the System Design is presented and the method to partition the FPGA for partial reconfiguration is presented. Now, as per the ATtACk methodology, the Implementation phase is presented where tools are used to aid the implementation process. The implementation of the Static and Dynamic module along with the issues to be considered and their effect in the system is presented.

Partially Reconfigurable (PR) systems have been under research and [29], [30], [31] explain in a brief manner as to how they can be implemented on Virtex FPGAs. However, they are targeted toward older FPGAs such as Virtex-II or Spartan. Major architectural changes have taken place in the design of the FPGA since then and the limitations possessed by the older FPGAs are no longer in the Virtex-4 FPGA. Thus new guidelines have to be followed for newer FPGAs.

4.2 Implementation

Following the ATtACk methodology, the Implementation phase is presented along with the guidelines to be followed.

4.2.1 Algorithm

Before proceeding toward implementation, certain guidelines that have to be followed in order to implement a partially reconfigurable (PR) system are considered. The first requirement is that a Modular Design flow has to be followed as described by Xilinx Inc. [32]. This method describes how small modules of a large system can be designed independently in Verilog or

VHDL and then merged together to form one complete design. In case of a PR system, the Static and Dynamic modules have to follow the Modular Design flow since later they will be merged as a single system. A detailed description of the Modular design flow and Module Based Partial Reconfiguration flow are beyond the scope of this project and the interested reader can refer to [26] and [32].

The Static module contains the blocks that do not change during runtime whereas the Dynamic module consists of blocks that are to be reconfigured dynamically when required. The Static module in this project consists of ADC, LPF, SNR calculation and the Decision algorithm. The Dynamic module consists of both LA algorithms. Implementation of the Static and Dynamic module is explained next.

4.2.1.1 Implementation of Static Module

In this project the Static module is designed and implemented using Xilinx System Generator which is a software tool for modelling and designing FPGA based Digital Signal Processing (DSP) systems in the Simulink environment using the Xilinx Blockset [33]. The Xilinx Blockset is a Simulink library provided in System Generator which consists of blocks like accumulator, adder, counter, memories along with tools like Filter Design and Analysis (FDA) tool, resource estimation block etc. System Generator is used for implementing Static module rather than using Handel-C language which is used to implement the Dynamic module. The Static module in this project requires blocks like counter, accumulator, multiplier etc. and are readily available in the Xilinx Blockset. The Xilinx Blockset also provides the M-code block where the functionality of the block is defined by a MATLAB code. The MATLAB code written for this block should be in the special Xilinx fixed point format which is provided by the System Generator [33].

All the blocks provided in the Xilinx Blockset should be used in between the *GatewayIn* and *GatewayOut* blocks which convert the input data type into Xilinx fixed point format and vice versa [33]. These blocks can also be connected to other Simulink blocks but System Generator can generate hardware realization only for the blocks provided in the Xilinx Blockset. These options provided by System Generator makes the design easier to build than using the Handel-C language which is used for implementing the Dynamic module.

Another major advantage using System Generator is that the data width can change based on the designer's choice. This is a flexibility provided by FPGA's over Application Specific Intergrated Circuits (ASIC) [34]. This facility is used in implementation of Static module which is explained further. The drawbacks in using System Generator are mainly the limited operations that it provides which are described in [33] and the complexity in designing large arithmetic calculations. For example, System Generator allows division operation only when the divisor is a power of 2. The division for any other value of the divisor can only be done by using

COordinate Rotation DIgital Computer (CORDIC) divider block provided by Xilinx Blockset at the expense of delay and can be performed only for positive inputs [33]. The Xilinx fixed point format used to write MATLAB code for M-code block does support some control statements like *if-else*, *for* but not *while* [33]. But these operations can be performed by designing the system in an intelligent way using other blocks provided in the Xilinx Blockset. Comparing the wide DSP functionalities it provides over it's drawbacks, System Generator is a designer friendly platform for implementing DSP designs on FPGA platforms. Implementation of the Static module is presented next beginning with the LPF.

Implementation of Low Pass Filter

As described in Chapter 3, an LPF is needed after the ADC in order to filter out the low frequency components. A 32-tap low pass Finite Impulse Response (FIR) digital filter with a sampling frequency of $25MHz$ is designed using FDA tool to get the filter coefficients as described in Section 3.3.2. The number of taps is chosen as 32 and not higher since filtering requirements are not too stringent due to absence of an interferer and also to limit the size of the Static module in terms of it's usage of multiply and accumulate logic. An FIR filter of length M which corresponds to number of taps with input $x(n)$ and output $y(n)$ is described by Equation 4.1 [35].

$$y(n) = \sum_{k=0}^{M-1} b_k \cdot x(n-k) \tag{4.1}$$

As shown in Equation 4.1 b_k represents the filter coefficients. A MATLAB code in Xilinx fixed point type is written in order to implement the linear phase FIR filter as described in Equation 4.1 which can be used by an M-code block. Initializations provided to the *LPF_32tap* block in Figure 4.1 are:

- Number of taps: 32
- Number of input bits: 14
- Binary point in input: 13
- Number of output bits: 14
- Binary point in output: 13
- Coefficients: xlfda_numerator('FDATool')
- Latency: 1

The filter coefficients generated by the FDA tool are given as input to the LPF block by using the instruction *xlfda_numerator('FDATool')*. Every model that is using at least one block provided by the Xilinx blockset should contain a Xilinx System Generator token as shown in Figure 4.1 [33]. This token is not connected to any other block, but it is used to generate HDL and NGC netlists for different FPGAs depending on the user's interest. It can also be used to simulate the design for hardware co-simulation. Similarly, the FDA tool block is also to be added in the model in order to transfer the filter coefficients generated by the tool to the intended blocks in the design. Figure 4.1 also shows the data width used for the signals at different stages. For example *Fix_14_13* represents the data associated with that signal is of type *Fixed* with a total width of 14 bits where 13 bits are for the fractional part. The reason for using those specific data widths is that the resolution of the ADC that is used in this project is 14 bits with 13 fractional bits.

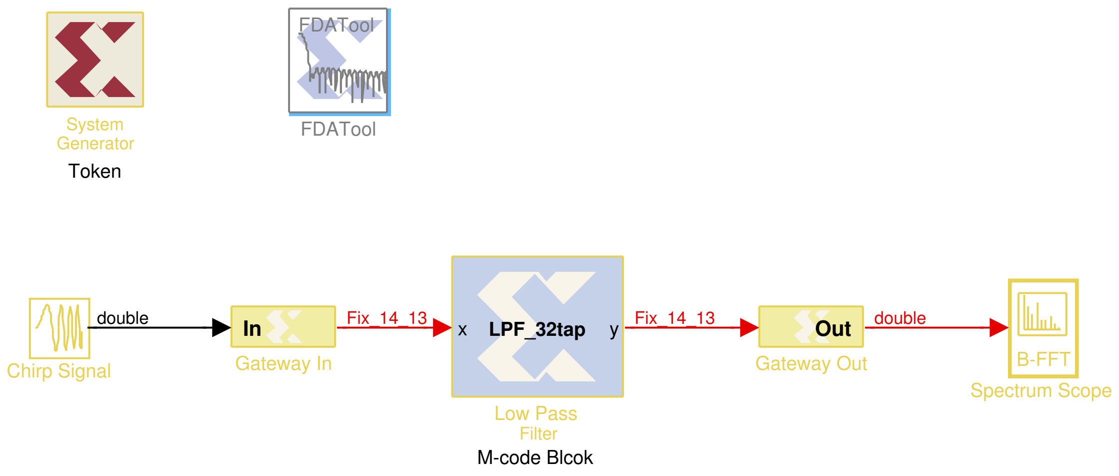


Figure 4.1: *Test setup for Low pass filter design in System Generator. A chirp signal sweeps from 0 to 25 MHz which is fed to the filter. The output of the filter is seen on the scope.*

Validation of Filter Response

The test setup for the filter is shown in Figure 4.1. In order to monitor the performance of the designed LPF, the magnitude response of the filter is plotted using the spectrum scope by giving a chirp signal input which sweeps from one frequency to another. Chirp signals are used to characterize systems such as filters [37]. The chirp signal input in this case has initial frequency of 0 Hz and a target frequency of 25 MHz. The magnitude response of the filter is observed as shown in Figure 4.2 with a cutoff frequency approximately at 50 kHz. Thus the filter behaves as expected.

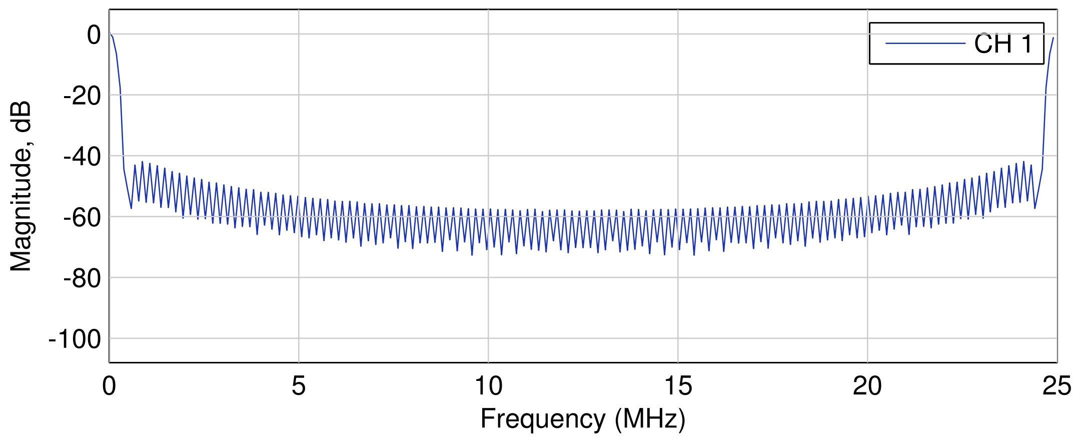


Figure 4.2: *Magnitude response of Low Pass Filter for Chirp signal input. The cutoff frequency is observed at 50 KHz approximately.*

Implementation of SNR Calculation

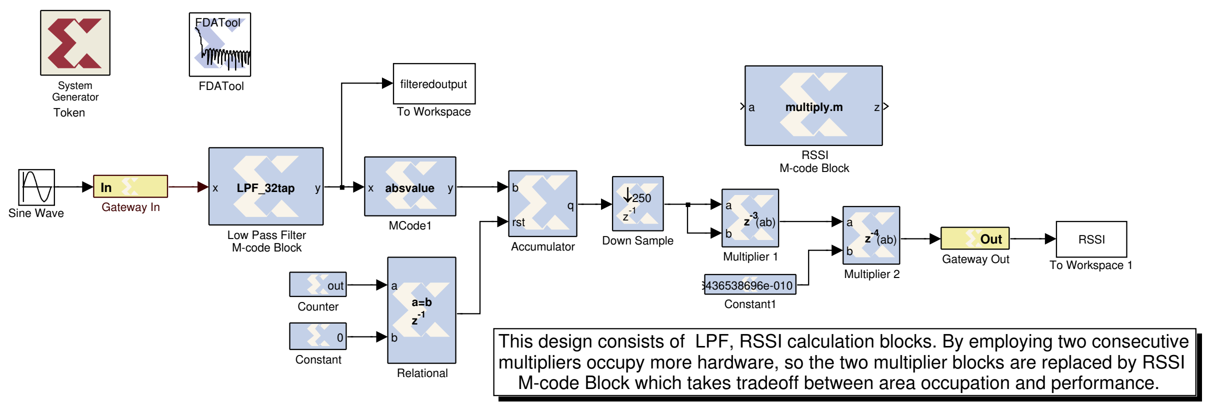
As described in Chapter 3, the SNR of the received signal is calculated by taking the ratio of RSSI and noise power. In this section, the noise power calculation and RSSI calculation using System Generator is presented prior to the calculation of SNR. In order to calculate RSSI using Equation 3.2, the design is divided into two steps, namely, extracting sum and arithmetic calculation which are described as Step 1 and Step 2 respectively. Step 1 includes taking the sum of absolute values of every N samples from the output of the LPF i.e.the operation of $\sum_{n=0}^{N-1} \mid Y[k,n] \mid$. Step 2 includes squaring, divide by N and multiplication with $0.7148 \cdot 10^{-10}$ as in Equation 3.2. The System Generator design for calculating the RSSI is as shown in Figure 4.3.

N in Equation 3.2 represents the number of samples in the signal for one signal period. The Bandwidth (B_s) of the signal is $100KHz$ and sampling frequency (f_s) is $25MHz$. Thus the Time Period T_p is:

$$\text{Time Period, } T_p = \frac{1}{B_s} = 10^{-5}\text{sec} \tag{4.2}$$

So, the number of samples in a time period of $10^{-5}sec$ for a sampling frequency of $25MHz$ is:

$$\text{Number of Samples, } N = T_p \cdot f_s = 10^{-5} \cdot 25 \cdot 10^{6} = 250 \tag{4.3}$$



This design consists of LPF, RSSI calculation blocks. By employing two consecutive multipliers occupy more hardware, so the two multiplier blocks are replaced by RSSI M-code Block which takes tradeoff between area occupation and performance.

Figure 4.3: *Design using System Generator for RSSI calculation.*

Thus value of N is 250.

Step 1: Extracting Sum

In order to extract the sum, output from the LPF is converted to its absolute value which is accumulated and reset for every N samples by using an Accumulator block provided in Xilinx Blockset. To reset the accumulator a unit pulse of boolean type is generated for every N samples and is given as an input to the accumulator as shown in Figure 4.3. The reset signal is generated by using a counter, a constant block with the value of 0 and a relational block. The counter is limited to 250 and whenever it goes back to 0 the relational block send a reset signal since both of the inputs are equal.

In order to collect the accumulated sum for every N samples a down sample block is used which outputs every N_{th} sample input for an input of N samples. It is assumed that the aliasing effect because of down sampling is negligible in order to make the design simpler to save hardware resources by avoiding anti-alias filter design.

Step 2: Arithmetic Calculation

The accumulated sum is to be divided by N and squared before multiplying with the constant in Equation 3.2 to get the RSSI value. In order to get the squared value the down sampled output is given to the inputs of the multiplier block provided in Xilinx Blockset which consists of only two input ports and one output port. Then, by using another multiplier block, multiplication with the constant is performed as shown in Figure 4.3. But the consecutive multiplier blocks occupy more hardware resources to establish pipeline processing. Hence, the multiplier blocks are replaced with a M-code block denoted by *RSSI M-code block* to handle all the multiplication operations as shown in Figure 4.3. The MATLAB code in Xilinx fixed point type

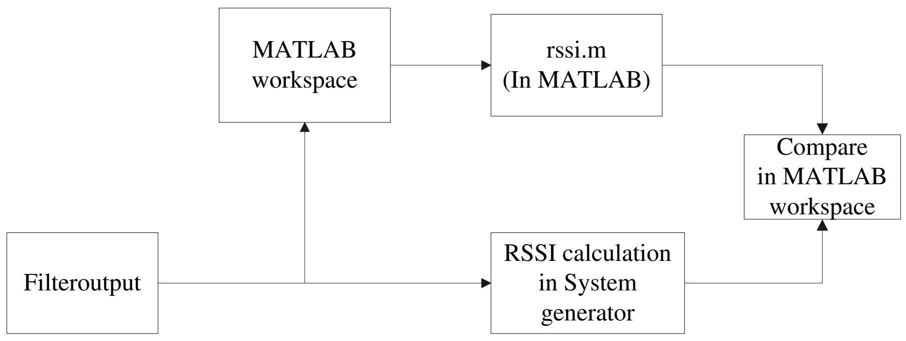

MATLAB workspace
rssi.m (In MATLAB)
Compare in MATLAB workspace
Filteroutput
RSSI calculation in System generator

Figure 4.4: *Validation of RSSI calculation. The filtered output is transferred to a MATLAB workspace which acts as an input to the rssi.m code and the output of System Generator is compared with that of rssi.m*

supports division only for divisors with a value having a power of 2 [33]. Hence Step 2 is simplified with the result of Equation 4.3 by substituting N as 250 in Equation 3.2 as shown by Equation 4.4 and Equation 4.5.

$$\text{RSSI} = 0.7148 \cdot 10^{-10} \left(\frac{1}{250} \sum_{n=0}^{249} \mid Y[k,n] \mid \right)^2 \text{mWatt} \tag{4.4}$$

$$\text{RSSI} = 1.144 \cdot 10^{-15} \left(\sum_{n=0}^{249} \mid Y[k,n] \mid \right)^2 \text{mWatt} \tag{4.5}$$

As shown in Figure 4.3 the RSSI M-code block contains a multiplication operation for multiplying the squared accumulated sum with the constant in Equation 4.5. The value 10^{-15} is not used in the implemented design since later it gets cancelled when divided by noise power which is shown in Equation 4.8.

Validation of System Generator Design for RSSI Calculation

The process to validate the RSSI calculation is shown in Figure 4.4. In order to test the design for calculating RSSI using System Generator, a normal MATLAB code is written for Equation 4.5 (represented as *rssi.m* hereon). The purpose of this test is to verify that the System Generator design provides the same output as that of the MATLAB code. A signal is applied to the *GatewayIn* block which is connected to the LPF. The output of LPF is saved in MATLAB workspace called as $filteredoutput$ as shown in Figure 4.3 and is given as input to the *rssi.m* which gives the RSSI values. Similarly, as shown in Figure 4.3 the RSSI M-code block output is given to the MATLAB workspace to be compared with the simulated result of *rssi.m*.

The results of both the simulations are presented in Appendix B.2 for comparison which are approximately equal. The To/From workspace block is provided in the Simulink blockset to store or pass the values from or to the designs in System Generator.

Noise Power Calculation

As described in the design chapter the noise power is calculated as the noise floor of the receiver considered in this project from Equation 3.5 by substituting Boltzman's constant k as 1.38x10^{-23}, effective temperature T_e as $290K$, signal bandwidth B_s as $100KHz$ and equivalent noise figure NF_{tot} as $4.46dB$ as shown in Equation 4.6.

$$\text{Noise floor, dBm} = -174\text{dBm} + 50\text{dB} + 5.5\text{dB} = -118.5dBm \tag{4.6}$$

Hence to calculate the total noise power (NP_{tot}) in linear values, the noise floor is converted to linear value as shown in Equation 4.7.

$$NP_{tot}, (W) = \text{Noise floor, W} \approx 10^{-15}Watts \tag{4.7}$$

Hence, from Equation 4.5 and Equation 4.7 the SNR is calculated as shown in Equation 4.8.

$$SNR = \frac{1.14 \cdot 10^{-15}\left(\sum_{n=0}^{249} \mid Y[k,n] \mid\right)^2 \text{mWatt}}{10^{-12}\text{mWatt}} \tag{4.8}$$

$$SNR = 0.00114\left(\sum_{n=0}^{249} \mid Y[k,n] \mid\right)^2 \tag{4.9}$$

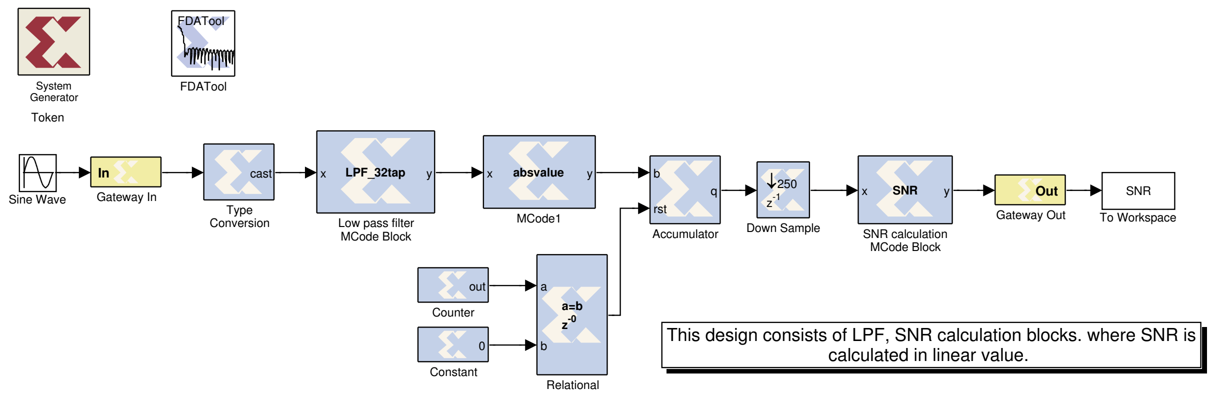


Figure 4.5: *Design using System Generator for SNR calculation.*

Hence, Equation 4.9 gives the SNR value of a single signal for a subcarrier. In Equation 4.9, k corresponds to a single signal i.e. for one time period of the signal. Hence, the System Generator design to calculate the SNR is as shown in Figure 4.5 where SNR M-code block contains the multiplication operation as shown in Equation 4.9. The same procedure to calculate the SNR is repeated for the second subcarrier also and the design is shown in Figure 4.6. The SNR is calculated as linear value instead of converting into dB and the average SNR is calculated from SNR's of both the carriers which is used to select appropriate LA algorithm using the decision algorithm. The reason for calculating SNR in linear value only is that the Xilinx Blockset does not provide any block for base-10 logarithm except natural logarithm at the expense of high latency and hardware resources. This also saves the computation required in LA algorithms while converting SNR in dB to linear value. The effect of this change on the Decision algorithm is explained next.

Implementation of the Decision algorithm

As explained in the design chapter the decision algorithm selects the appropriate algorithm based on the average SNR (avg. SNR) of the current link conditions. The conditions in decision algorithm for average SNR values in dB are:

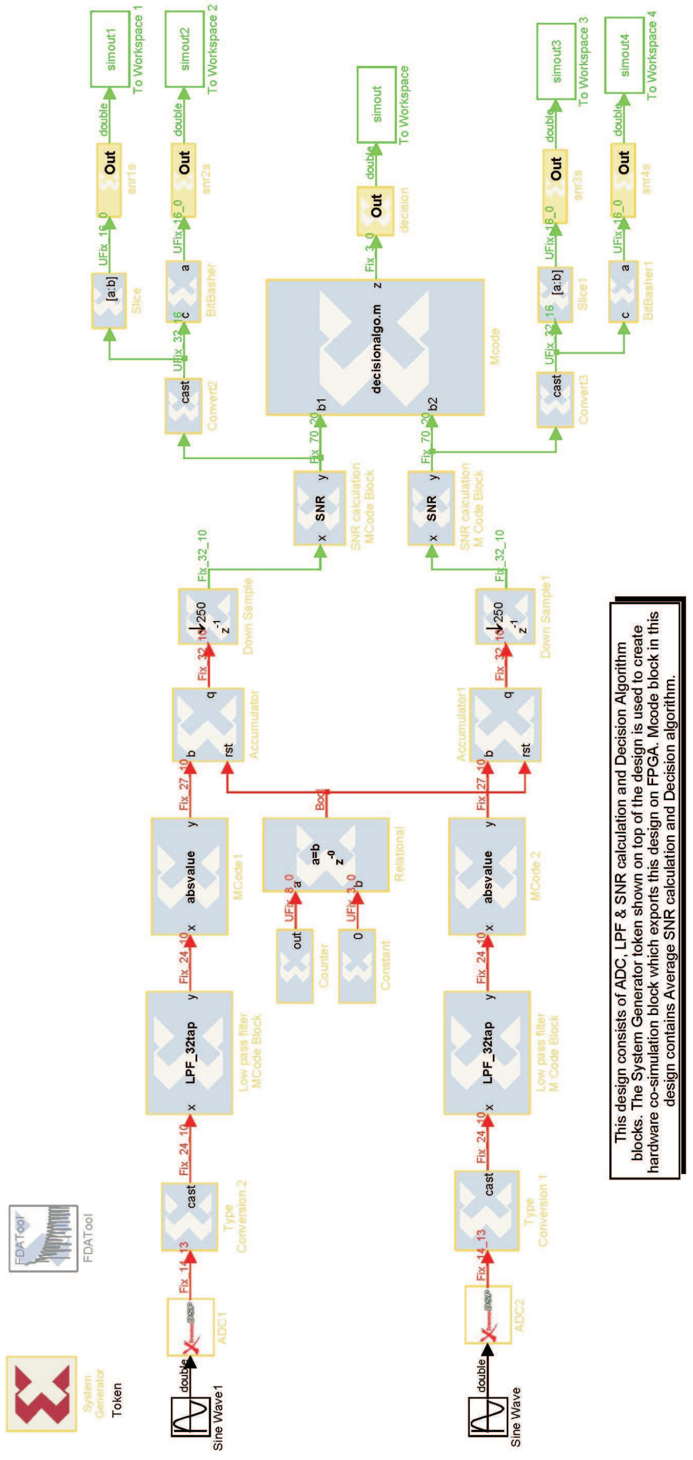
1. Select SRA when avg. SNR $\leq$ 10dB or avg. SNR $\geq$ 40dB
2. Select SAMPDA when 10dB $<$ avg. SNR $<$ 40dB

When converting SNR units from dB to linear scale, the conditions changes to:

1. Select SRA when avg. SNR $\leq$ 10 or avg. SNR $\geq$ 10000
2. Select SAMPDA when 10 $<$ avg. SNR $<$ 10000

As shown in Figure 4.5 the decision algorithm M-code block contains the average SNR calculation from SNR's of two subcarriers by giving the output as 1 when SRA is selected and 2 when SAMPDA is selected based on the conditions described above.

In order to provide the SNR values into the Dynamic module as input, the data width is converted to 16 integer bits and 16 fractional bits since that is the format used for the Dynamic module. This is because, a special library *fixed.hcl* is used in the Handel-C code which requires this type of number representation. For both the carriers the integer and fractional bits are given from separate out ports as shown in Figure 4.6 along with decision output by using a slice operation provided in System Generator. The data widths at each level is is also shown.



This design consists of ADC, LPF & SNR calculation and Decision Algorithm blocks. The System Generator token shown on top of the design is used to create hardware co-simulation block which exports this design on FPGA. Mcode block in this design contains Average SNR calculation and Decision algorithm.

Figure 4.6: *Design of Static module in System Generator for the two branches.*

Hardware Co-simulation

The System Generator design can run on a target hardware through a feature called hardware co-simulation. The System Generator token placed on top of the design generates the FPGA bitstream for the design between the *GatewayIn* and *GatewayOut* blocks. When the design is compiled for hardware co-simulation, a hardware co-simulation block as shown in Figure 4.7 is automatically created. The hardware co-simulation block accesses the connected FPGA platform using a USB or PCI interface during simulation to automate tasks like transferring the configuration bitstream and clocking. By using this feature the design can execute on the hardware for the inputs given in either System Generator or to the FPGA platform and it is possible to check the output back on a computer using the MATLAB workspace or an M-file.

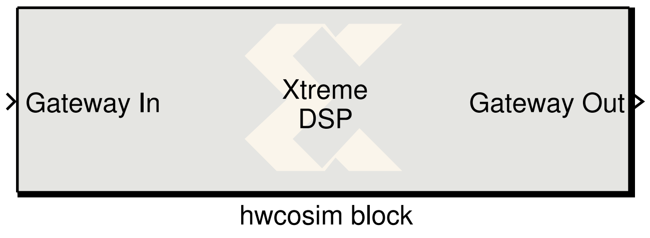
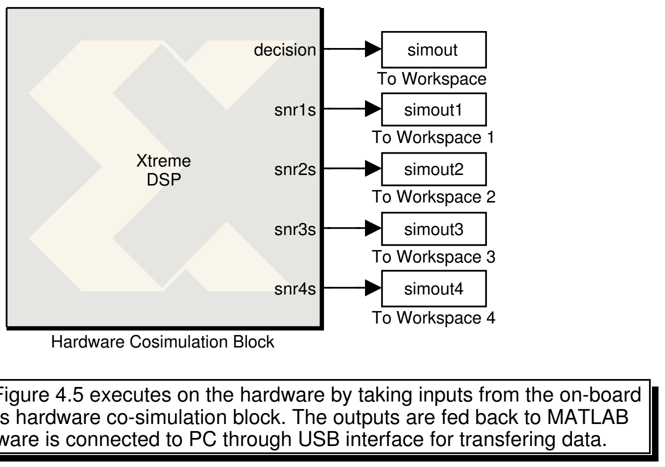


Figure 4.7: *Hardware Co-simulation block without board specific I/O ports. This block is run on the FPGA by providing real time input through the ADCs.*

Interfacing the Static Module with the ADC

The FPGA platform used in this project provides various board specific I/O ports to communicate with it which include shared memory, ADC, DAC etc. In order to communicate with the real time analog signals the Static module is interfaced with the ADC which converts them into Xilinx fixed point data for further processing. This block is provided in the Xilinx blockset. By including this block while compiling the design for hardware co-simulation, System Generator creates hardware co-simulation block including ADC interfacing logic. Hence, the design is simulated for the input given to the on-board ADC of the FPGA platform and it is not possible to give input in the System Generator model.

By co-simulating the design as shown in Figure 4.6 the library block created is as shown in Figure 4.8 where the input ports are masked while having only output ports. Hence by giving the input to the two on-board ADC's the design executes on FPGA and the output can be stored in MATLAB workspace by connecting the *To Workspace* blocks to each of the output ports as shown in Figure 4.8. Testing details of the Static module is included in the next chapter.



The design shown in Figure 4.5 executes on the hardware by taking inputs from the on-board ADC by simulating this hardware co-simulation block. The outputs are fed back to MATLAB workspace. The hardware is connected to PC through USB interface for transfering data.

Figure 4.8: *Hardware Co-simulation block for Static module using two ADCs.*

Optimization of the Static Module

The System Generator token creates a Xilinx ISE project to synthesize and implement the design while performing the hardware co-simulation. During the process of hardware co-simulation area and time estimates of the design are generated. It is observed that the slice area required for the implementation of the Static module is 27% and 4% of DSP48 slices are required. The DSP48 slices are dedicated resources on the FPGA for operations such as multiply and accumulate.

As seen in Table 1.1, the area requirement of SAMPDA is 94% for 16 subcarriers. Even after designing the algorithm for two subcarriers, the area reduction will not be so much that the design would be able to accommodate SAMPDA along with the Static module. Thus the Static module cannot be implemented along with SAMPDA due to lack of hardware resources. Because of this issue, the Static module has to be optimized so that it occupies less area. It is observed that the area occupation on the hardware is related to the data widths of the signals specified in System Generator. The on-board ADC output signal width is specified as 14 bits with 13 fractional bits. In order to carry the arithmetic operations in the RSSI calculation, the data width was increased to 24 bits by using the *Type conversion* block provided in Xilinx Blockset as shown in Figure 4.6.

To reduce the data widths, this is rearranged to 12 bits having 8 fractional bits as shown in Figure 4.9 by trading off number of bits for precision. Also, as shown in Figure 4.6 the SNR calculation block provides the SNR value which has a large data width to the decision algorithm M-code block. The large change in the data width of input and output of the SNR M-code block is because of the multiplication operation taking place in that block. So, it is observed that

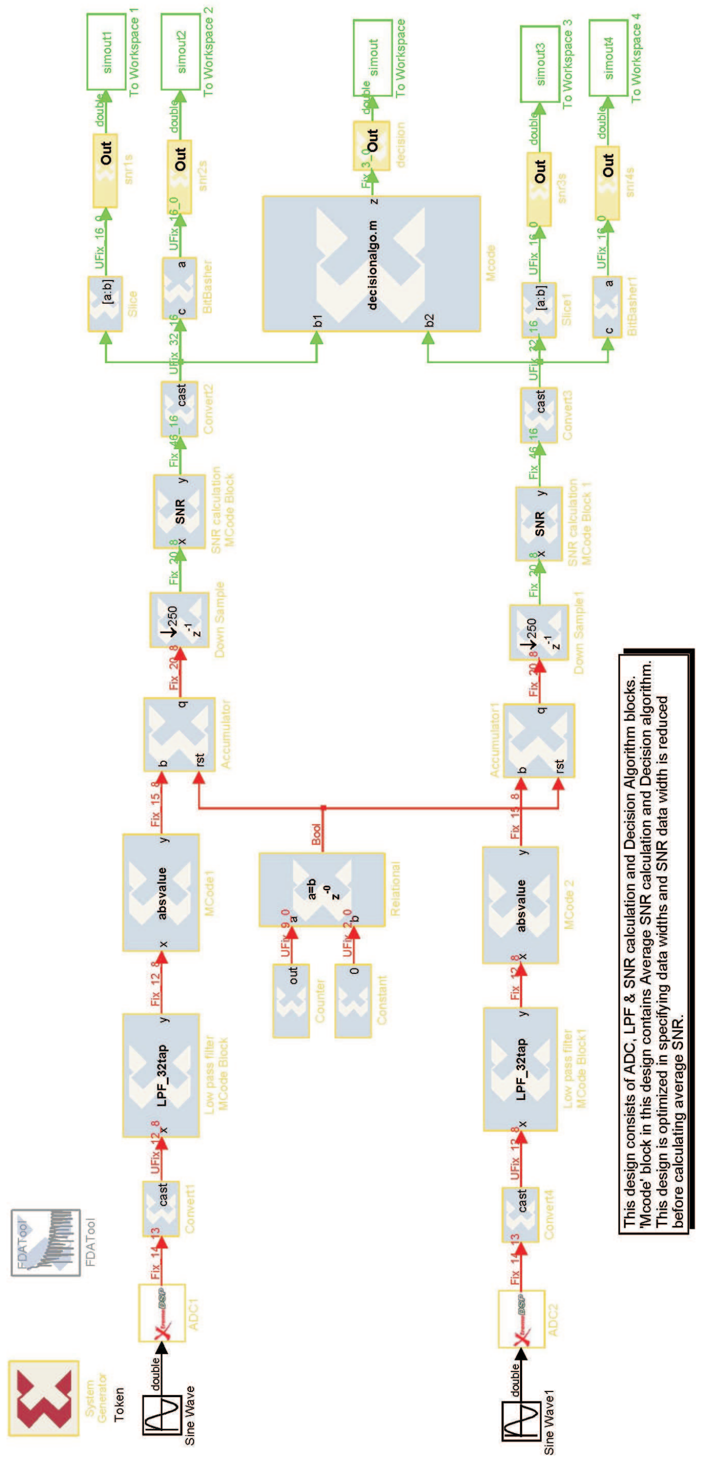

This design consists of ADC, LPF & SNR calculation and Decision Algorithm blocks.
'Mcode' block in this design contains Average SNR calculation and Decision algorithm.
This design is optimized in specifying data widths and SNR data width is reduced before calculating average SNR.
System Generator
Token
FDATool
Sine Wave
Sine Wave1
ADC1
ADC2
Convert1
Convert4
LPF_32tap
Low pass filter MCode Block
Low pass filter MCode Block1
absvalue
MCode1
MCode 2
Counter
Constant
Relational
Accumulator
Accumulator1
Down Sample
Down Sample1
SNR
SNR calculation MCode Block
SNR calculation MCode Block 1
Convert2
Convert3
decisionalgo.m
Mcode
Slice
Slice1
BitBasher
BitBasher1
snr1s
snr2s
snr3s
snr4s
decision
simout1
simout2
simout3
simout4
simout
To Workspace 1
To Workspace 2
To Workspace 3
To Workspace 4
To Workspace

Figure 4.9: *Design of static module in System Generator with optimization.*

more area is saved if the SNR value is transferred to the decision algorithm after changing its width by type conversion as shown in Figure 4.9. This reduction of the width minimizes the overhead in the arithmetic calculations taking place in the decision algorithm but causes rounding error which is not considered in order to implement the design. The Quantization and Overflow modes during the type conversion operation in the design are set as *Truncate* and *Wrap* respectively which do not require any hardware resources, while *Round* and *Saturate* modes require hardware resources [33]. By simulating the design for hardware co simulation with the optimizations discussed, the results of the hardware resource utilization is as shown in Table 4.1.

Table 4.1: *DSP resource utilization on the FPGA for Static module after optimization.*

Slices	**DSP48s**
4 %	2 %

4.2.1.2 Implementation of the Dynamic module

The Dynamic module comprises of two LA algorithms. These algorithms were previously implemented by [1] using Celoxica's DK4 design tool. The input to the Dynamic module is the SNR value of each subcarrier which are floating points. As mentioned earlier, floating point numbers are implemented by using the *fixed.hcl* library where the integer and fractional parts of the number are separately expressed using 16 bits each. In [1], the algorithms were implemented for 16 subcarriers. The change done for this project is that the number of subcarriers is two. A VHDL description consisting of a top level file and another for the actual design is automatically generated by Celoxica's DK4 design tool which is then used as the design entry language.

The top level VHDL file as required for PR is shown in Figure 4.10. It consists of instantiations of Static and Dynamic module, I/O interfaces and bus macro, all as black boxes. The Dynamic module is instantiated only once even though two algorithms exist. The VHDL file for each of the modules are separately synthesized in Xilinx ISE. For the Static module, a VHDL description is generated through System Generator which contains a top level file specifying the entities and another file describing the actual operation of the designed system. The interfaces to be specified is discussed later in the Communication section.

4.2.2 Architecture

In this section, first the floorplan of the Virtex-4 SX35 FPGA is described. Then the guidelines to be followed for implementing a PR system are enumerated. Based on those guidelines the architecture is partitioned and some issues are discussed.

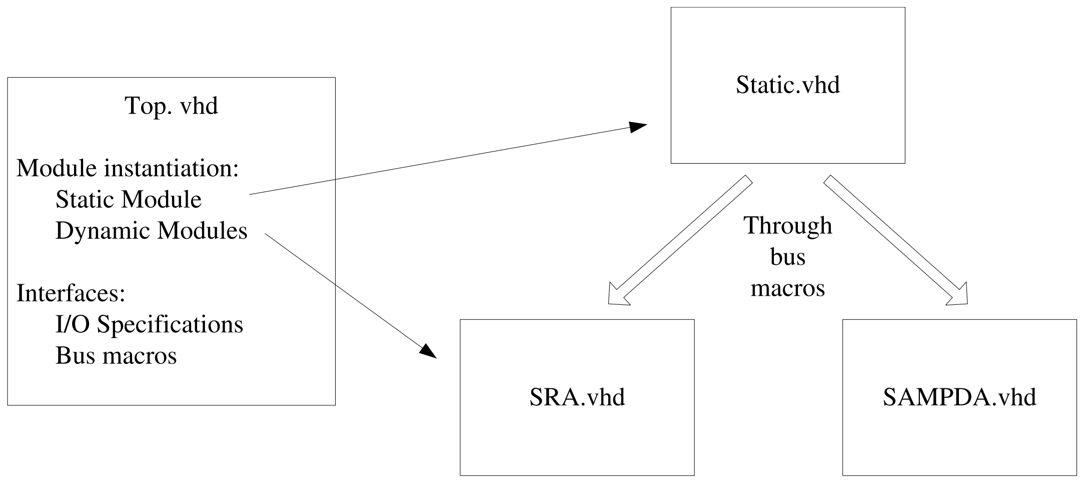


Figure 4.10: *Structure of top level VHDL file for Partial Reconfiguration. It consists of the module instantiations and the interface specifications using I/O ports and bus macros.*

Floorplan of the Virtex-4 SX35 FPGA

The floorplan of the Virtex-4 SX35 FPGA is shown in Figure 4.11. There is a significant difference in the design of the FPGA as compared to previous devices by Xilinx Inc. In Virtex-4, the pins are no longer at the outer edge of the FPGA fabric but are in the form of three columns, two on the sides and one at the center. Dedicated DSP columns are provided on the FPGA for multiply, accumulate and addition operation [36].

To implement the design, the Early Access Partial Reconfiguration flow [28] is followed. Access to this document has been restricted by Xilinx Inc. since PR is still under development for newer devices like Virtex-4. A special permission has to be obtained in order to access these documents that describe how PR systems can be realized on a Virtex-4 FPGA. Another tool that helps for PR is PlanAhead which has a feature that runs the Partial Reconfiguration flow. By default, this is deactivated and a special Tcl script is needed to activate it. Still, Xilinx Inc. does not provide any documentation on how to use this feature except an example which is designed using Verilog as the design entry language. In this project, the guidelines as mentioned in [28] are followed.

Guidelines for Partitioning the FPGA for a PR System

For a PR system, the FPGA is divided into Static and Dynamic modules which put some constraints on how the system is implemented. Certain guidelines for implementing MBPR are elaborated in [28] and [38]. A few of them worth mentioning are:

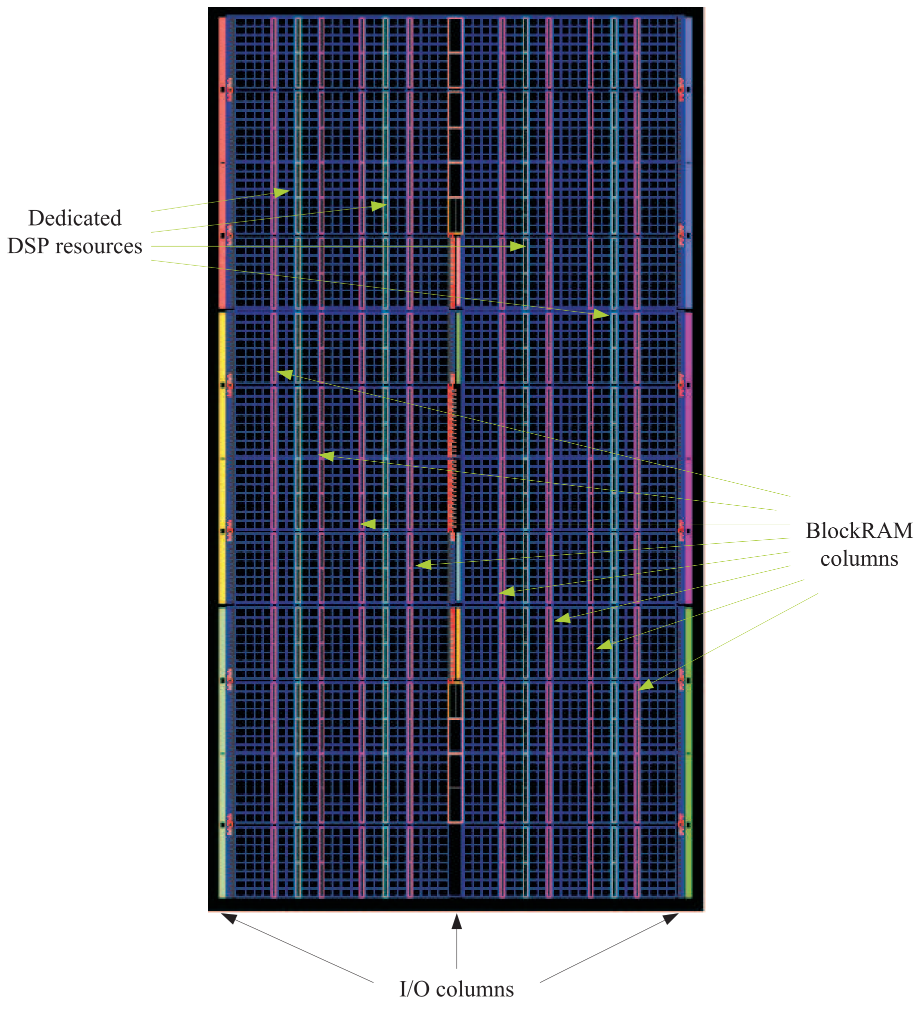


Figure 4.11: *Floorplan of the Xilinx Virtex-4 SX35 FPGA as seen in PlanAhead. This FPGA features the BlockRAM and dedicated DSP resources (XtremeDSP slices).*

- The modules of the FPGA (Static and Dynamic) have to occupy at least 16 Configurable Logic Block (CLB) rows in height of the device or its multiple. This was not the case with Virtex-II FPGAs where the entire height of the FPGA had to be reserved.
- The shape of the partitions allotted for the dynamic modules have to be rectangular.
- The boundary of the modules once set cannot be changed.
- BlockRAMs inside the module partition have to be a part of that particular module.
- The Static module should not be constrained in area.
- The PR region should not cross the center column which consists of pins. This means that the maximum size of the Dynamic module is limited to 50%.

FPGA Partitioning for this project

From [1], it is known that SAMPDA requires more area than SRA and so to estimate the area required by the Dynamic module, we look at how much of area is occupied by SAMPDA. After synthesizing the VHDL description of SAMPDA, the estimate of area requirement as given by Xilinx ISE is 87%. The remaining 13% of the area can be utilized by the Static module. Xilinx Floorplanner is then used to assign the area required by the Dynamic module. Two options for assigning the area exist as shown in Figure 4.12 and Figure 4.13.

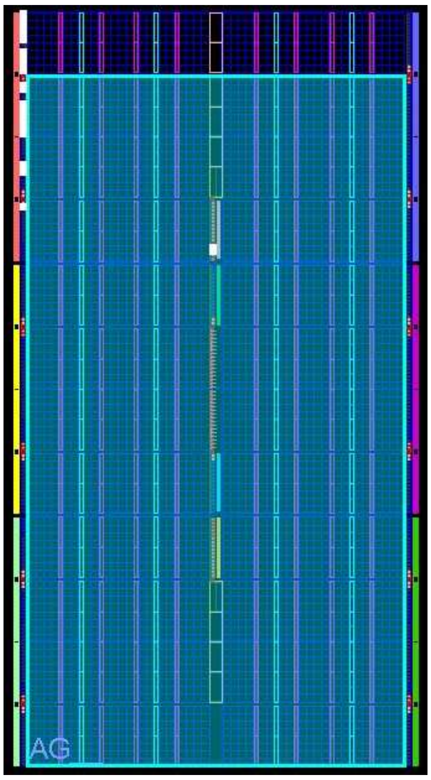


Figure 4.12: *Option one for area assignment of Dynamic module. Shaded area is assigned for the Dynamic module. The white spots indicate pin connections from the ADCs to the FPGA.*

The shaded region is the one assigned for the Dynamic module. The regions are assigned so as to comply with the rule that the height should be a multiple of 16 CLBs. The area remaining is to be utilized by the Static module. The X and Y co-ordinates of the partition are entered as an *AREA_GROUP_Range* constraint in the User Constraints File (UCF) within the Xilinx ISE tool. The Nallatech platform has two ADCs and the white spots are the pins that are the input to the FPGA from the ADC. Out of the total 15 pins for ADC1, two pins, namely A17 and B17 lie on the center column and the Dynamic module overlaps these pins. This is not

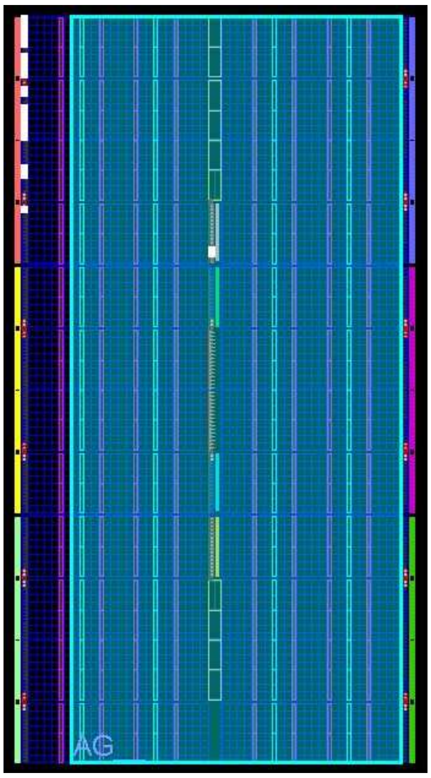


Figure 4.13: *Option two for area assignment of Dynamic module. Shaded area is assigned for the Dynamic module. The white spots indicate pin connections from the ADCs to the FPGA.*

allowed since during reconfiguration, the pins will be interfered. Thus due to the fact that the Dynamic module is too big resulting in the ADC output pins interfering with the PR region, a PR system cannot be implemented in this project. Hence it becomes necessary that the pin connections be studied carefully and based on design requirements, the development platform should be chosen.

Also, as mentioned earlier, the PR region should not cross the center column. But the area required by the PR region is large which necessitates that it crosses the center column. Current tools by Xilinx Inc. are not able to handle this restriction [28]. Also due to major architectural changes in Virtex-4 as compared to older FPGAs, newer tools and methods have to be followed. The tools needed are Xilinx ISE 8.2.01i along with a patch that supports PR for Virtex-4. The university has only ISE 8.1i and the free WEBPack version of ISE 8.2.01i which do not support the SX35 Virtex-4 FPGA used in this project. Thus due to the problems of the PR region being too large, unsuitable pin connections and unavailability of tools, a complete PR system cannot be implemented in this project. Thus a PR system is presented as a modular system along with the bus macros that would have to be used if the right tools were available.

Implementation Considerations

Assuming that the pin connections are not an issue and to assess the implementation if it was possible, each of the modules (Static and Dynamic) are independantly implemented by constraining the area on the FPGA. Two types of possible implementations are shown in Figure 4.14 and Figure 4.15. These figures show the Xilinx Floorplanner output after the design is implemeted by constraining the area on the FPGA using Xilinx ISE. In Figure 4.14, all the ADC pins are assumed to be on the left side and in Figure 4.15, the ADC pins are assumed to be distributed across the three I/O coumns within the area for the Static module.

In a real PR system, the Static module should not be constrained in area and the tool used to implement the PR design interprets the UCF file where area constraints are specified and places the Static module on the area remaining. However, since a PR system cannot be implemented in this project, area constraints are applied to verify if Static module could be implemented on the area remaining after reserving area required by the Dynamic module. It is seen from Figure 4.14 that the Static module does not follow the constraint and occupies the DSP columns which are part of the Dynamic module which is marked by a red outline. This is because the blocks used in System Generator for designing the Static module require the dedicated DSP resources. Even after restricting DSP usage to 0% in Xilinx ISE, the implemented design still occupied the DSP resources. However, this is not the case with Figure 4.15 where the Static module occupies the DSP resources which are part of the remaining area. But in this case, the pin connections from the ADC do not match the location of the Static module. The implementation of the Dynamic module is done in Xilinx ISE 9.1i and that of Static module is done in Xilinx ISE 8.1i. This is because, upon entering the area constraints for the Static module, ISE 9.1i failed to place it for the conditions in Figure 4.14. However, ISE 8.1i defied the constraints and placed the design on the DSP columns. DSP resource usage and Slice utilization for the modules is shown in Table 4.2. The detailed Xiliinx ISE reports are shown in Appendix C.

Table 4.2: *DSP resource usage on the FPGA for different modules.*

Module	**DSP resource usage**	**Slice utilization**
Static	2%	4 %
SRA	1%	27%
SAMPDA	2 %	90%

Thus, if the PR system was designed, it would not be possible to implement it because on the following reasons:

1. The pin connections from the ADC to the FPGA are not as required since they cannot be connected to the Dynamic module.

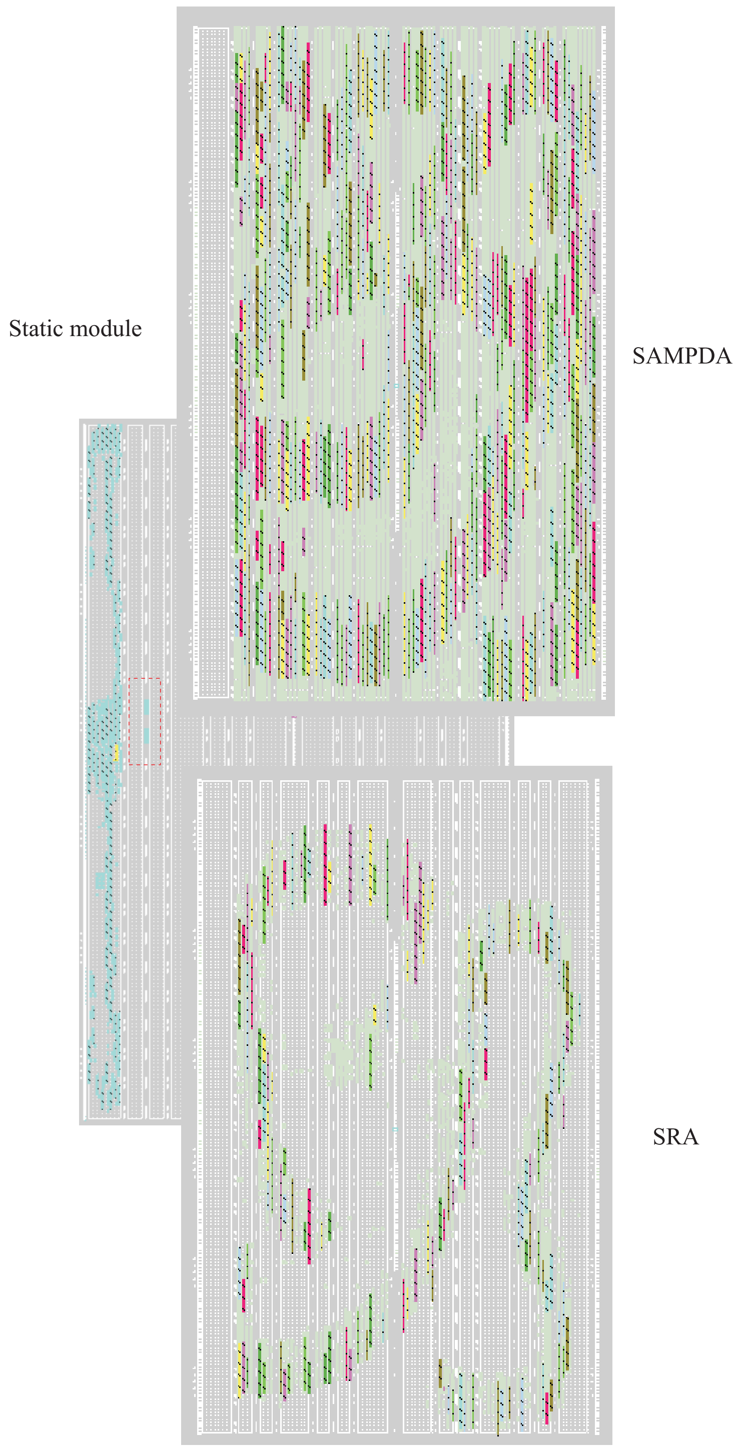


Figure 4.14: *Implementation option one for the Dynamic module. The colors are automatically assigned by ISE for different logic elements.*

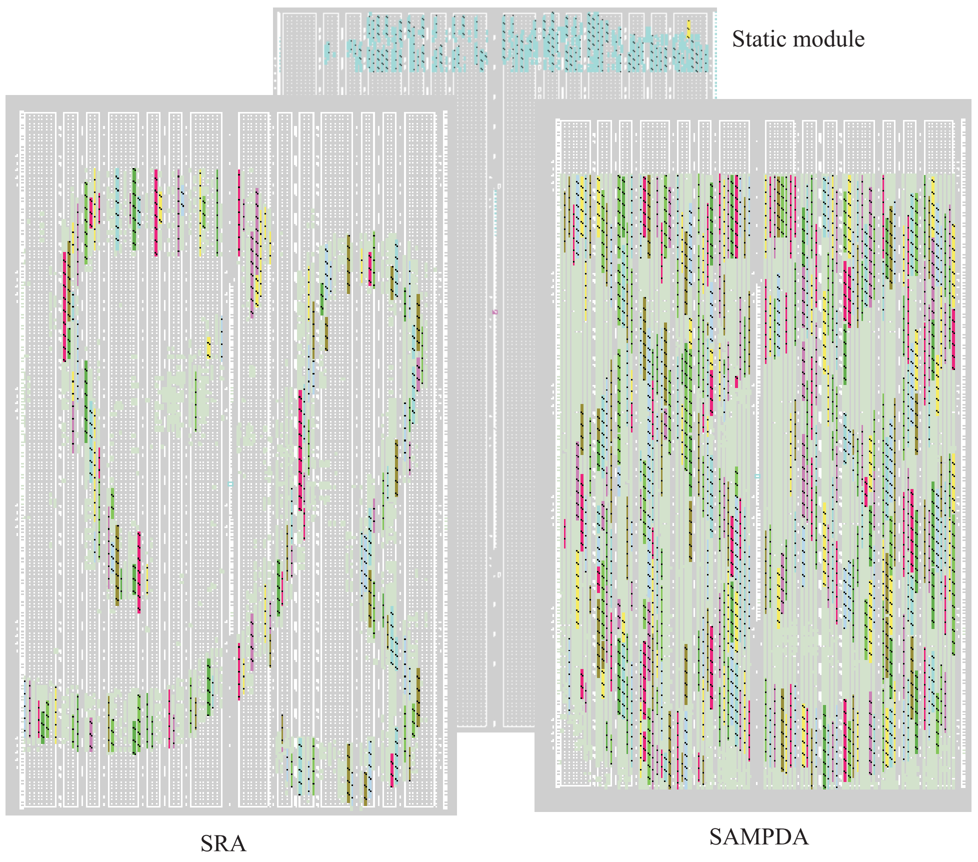


Figure 4.15: *Implementation option two for the Dynamic module. The colors are automatically assigned by ISE for different logic elements.*

2. The size of the Dynamic module is too big and crosses the center colum which is not supported by the design tools.
3. The tool required is the full version of ISE 8.2.01i which is not available at the University.

4.2.3 Communication

Several issues regarding communication have to be considered for PR systems and each of them are briefly explained.

Top Level File

For PR systems, care has to be taken that for both the LA algorithms, the interface description between the Static and Dynamic module is same with respect to pin and port naming. This ensures that connections between the two modules are maintained and so the same interfaces are used in the Handel-C code for both the algorithms. Thus the top level VHDL file generated by DK4 which consists of the input and output ports is the same for both the algorithms and so one of them is used. Also, the top level VHDL file from System Generator is merged manually with this to have one single top level file. The structure of the top level file is shown in Figure 4.10.

Configuring the FPGA

Initially when the FPGA is powered up, a full bit stream is loaded onto the FPGA to make it fully functional. It depends on the designer to choose which configuration has to be loaded. To partially reconfigure the FPGA, either the SelectMAP, Serial, JTAG or Internal Configuration Access Port (ICAP) interface could be used [28]. The reconfiguration type used in this project is exo-reconfigurable system which means that a computer is used to store the required *bit* files for reconfiguring the FPGA. In this project, the USB interface is used to communicate between the FPGA and computer which can be used for configuring the FPGA via the Field Upgradable Systems Environment (FUSE) software provided by Nallatech. To be able to retrieve the specific *.bit* file to reconfigure the FPGA, the specific memory location where the file is stored on the computer has to be accessed. This is also the reason for choosing the USB interface because communication from the development platform to the computer via USB is handled by an Interface FPGA as shown in Figure 2.13. This can handle Memory Map Interface and DMA interface and thus can retrieve the appropriate *bit* file from a specified memory location. However, the Interface FPGA is not connected to the SelectMAP or the Serial pins on the User FPGA on the development platform. Because of this, partial reconfiguration cannot be controlled by the Interface FPGA.

If the Interface FPGA was connected to either SelectMAP or Serial interface, then the output of the Decision algorithm would be read by the Interface FPGA through the Interface Communication Bus. Based on this input, the Interface FPGA would access the specific *bit* file located in the computer via the DMA interface. Upon retrieving, this *bit* file would then be used to partially reconfigure the FPGA as shown in Figure 4.16.

When the Decision algorithm provides an output (1) indicating that the algorithm has to change, the Interface FPGA requests the computer (2) for the appropriate *bit* file through the DMA interface. The computer responds by providing the *bit* file (3). This bit file is then used to reconfigure the Dynamic module (4). All this operation requires control information overheads that have to be exchanged between the Interface FPGA and the computer which is part of the Interface FPGA. The software library provided by Nallatech to control the DMA interface is not able to handle reconfiguration operations automatically and manual intervention is required. Thus further development in this area is required.

Bus Macros

To facilitate communication between Static and Dynamic modules, bus macros are used which act as fixed routing resources. A bus macro provides 8 bits of communication in one

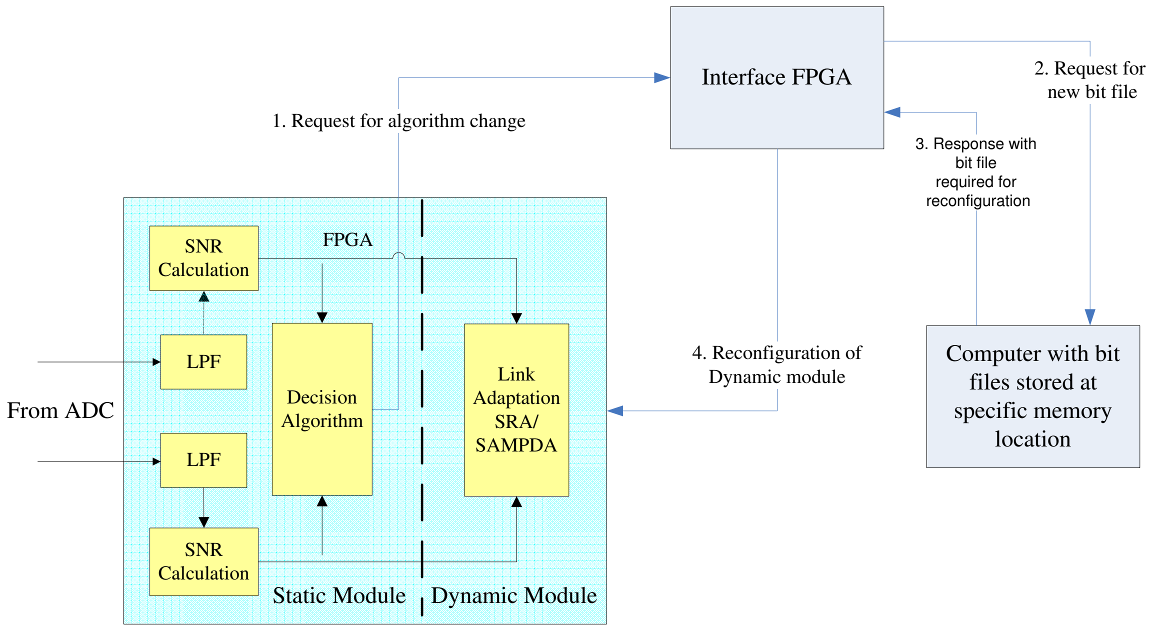

Interface FPGA
1. Request for algorithm change
2. Request for new bit file
3. Response with bit file required for reconfiguration
4. Reconfiguration of Dynamic module
Computer with bit files stored at specific memory location
FPGA
SNR Calculation
LPF
LPF
SNR Calculation
From ADC
Decision Algorithm
Link Adaptation SRA/ SAMPDA
Static Module
Dynamic Module

Figure 4.16: *Example of reconfiguration process that could be possible through the Interface FPGA as provided on the development platform.*

direction only i.e. either left-to-right, right-to-left, top-to-bottom and bottom-to-top. These bus macros are provided as *.nmc* files which are used during implementing the PR system. They have to be instantiated in the top level file too and since these are fixed routing resources, their location on the FPGA have to be fixed using Location (LOC) constraints in the UCF. Bus macros in earlier devices like Virtex-II and Spartan are based on Tri State Buffers (TBUFs) and provide only 4 bits of data communication in one direction. However, for Virtex-4, TBUFs do not exist and bus macros are based on CLBs providing 8 bits of one-way data communication. As mentioned in the Algorithm domain in Section 4.2.1, the inputs to the Dynamic module are four SNR values each of 16 bits. Thus a total of 64 bits of communication are needed. Since each bus macro provides 8 bits of communication, 8 bus macros are needed to establish proper communication.

Xilinx Inc. has provided bus macros that are classified based on [28]:

- Direction of data: left-to-right, right-to-left and additionally top-to-bottom, bottom-to-top for Virtex-4.
- Physical width: wide - 4 CLBs and narrow - 2 CLBs. Both of them provide 8 bits of interface.
- Whether signals passing through the bus macro are registered or not: synchronous and asynchronous

Synchronous bus macros provide better timing performance [28]. Wide bus macros can be placed in a nested fashion to provide more than 8 bits of data flow in any direction. However, additional rules regarding placement of bus macros have to be followed as outlined in [28]. As mentioned in the previous chapter, even though the step to determine the modulation scheme based on the received SNR is common between the two algorithms (SRA and SAMPDA),it is not implemented in the Static module. Figure 2.11 shows that this step takes place in a loop and to implement this, bus macros for both directions (Static-to-Dynamic and Dynamic-to-Static) will have to be used. The complications in implementing such a system outweighs the benefits of having one additional step in the Static module and so it is not feasible.

In this project, a narrow and synchronous bus macro is used that transfers data from left-to-right since the Static module is on the left of the FPGA. As mentioned earlier, bus macros are instantiated in the top level file and the interfaces are also provided. These bus macros are pre-routed and the instantiation template is provided by Xilinx Inc. where each input and output bits have been explicitly specified. In the Handel-C code, each variable that reads the SNR value is of 16 bits. Thus 2 bus macros are required to transfer each SNR value. An example of implementing the bus macro is shown in Figure 4.17.

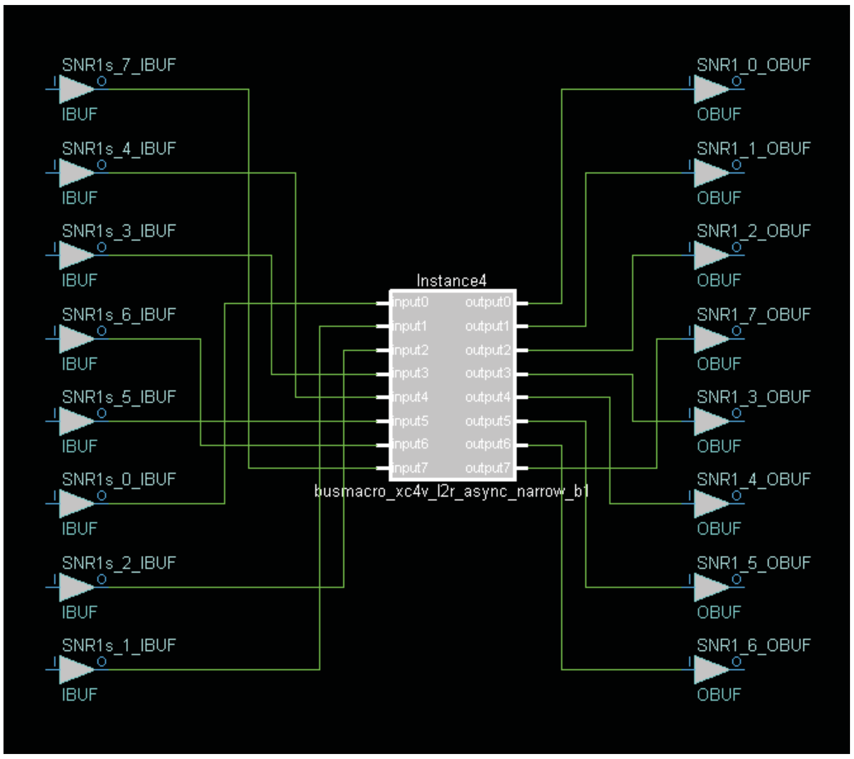


Figure 4.17: *Example of bus macro interface between Static and Dynamic module as seen in PlanAhead. Each output bit of the Static module is explicitly interfaced with the input bits of the Dynamic module*

The internal structure of the bus macro is not seen here since it is declared as a black box in the top level VHDL file where the interfaces are defined. Each output and input bit from the Static and Dynamic module is explicitly associated with a unique bit of the bus macro thereby establishing a connection between the two. The example shows how the first eight bits of the integer part of the SNR value from the Static module is connected to the corresponding variable in the Dynamic module. In order to fix the location of the bus macro, certain guidelines have to be followed. One specific guideline is that the bus macro should not straddle the DSP or BlockRAM columns on the FPGA. To avoid doing so, the X co-ordinate of the LOC constraint for the bus macro has to divisible by 2 in case of a narrow bus macro [28]. One limitation that these bus macros possess is that the instantiation templates provided by Xilinx Inc. has to be used which means that each bit has to be explicitly interfaced. This can become quite a cumbersome task in case of large designs.

4.2.4 Time

On implementing each of the module, the minimum clock period that is possible is obtained from the Place and Route report in Xilinx ISE which is shown in Table 4.3.

Table 4.3: *Minimum clock period for each module.*

Module	Minimum clock period
Static	116ns
SRA	197ns
SAMPDA	274ns

When the PR system is successfully implemented, a uniform clock frequency has to be maintained. Thus the clock period for the PR system if implemented is 116ns + 274ns = 390 ns which is 2.56 MHz.

The timing constraint i.e. the maximum time allowed to execute the LA algorithm is 38.9 ms. This calculation is given in Appendix A.2. Within 38.9ms, the FPGA should reconfigure itself as well as execute the LA algorithm. Reconfiguration time is directly proportional to the bitstream size. In an exo-reconfigurable system the type of interface (PCI/USB), type of configuration (SelectMAP/Serial) also have to be considered. Based on the size of these files and the type of interface used to reconfigure the FPGA, the reconfiguration time can be calculated. After performing Implementation on ISE, *.bit* files are generated and the *bit* file for SRA and SAMPDA which is 1683 KB in size for both. Thus it is important to note that the size of the *bit* file directly depends on the area reserved for the Dynamic module. The SelectMAP interface has 32 bits that are used for configuring the FPGA while the Serial interface has only one pin [42]. The FPGA is configured by loading 1 configuration bit per clock cycle [39]. Assuming that a SelectMAP interface is used which is faster than the Serial interface, the time taken to reconfigure the FPGA is given by:

$$T_r = \frac{\text{Size of bit stream}}{\text{Clock frequency} \cdot 32} \tag{4.10}$$

If a PR system is implemented, the bit files are automatically compressed [32]. SO the bit files obtained from implementing SRA and SAMPDA are compressed by using the *Compress bit file* option in Xilinx ISE Thus the size of the compressed bit file is 977 KB for SRA and 1119 KB for SAMPDA. As mentioned earlier, the system in this project can operate at 2.56MHz. Therefore,

$$T_r = \frac{1119KB}{2.56MHz \cdot 32} = 13.6ms \tag{4.11}$$

To calculate the latency in transferring the bit file from the computer to the FPGA, a USB interface is assumed which operates at 480 Mbps [40]. A 1119KB file would thus require 2.3ms. Additionally, control information generation and transmission i.e. request for bit file is assumed to be approximately 2ms. Thus total time to reconfigure in case of an exo-reconfigurable system is

$$13.6ms + 2.3ms + 2ms = 17.9ms \tag{4.12}$$

Thus the reconfiguration operation takes 17.9ms. Now, to calculate the execution times of the algorithms, the number of clock cycles needed for execution is multiplied with the clock period. The number of clock cycles is obtained from [1] and the same are used assuming that the number of clock cycles needed do not differ significantly. Table 4.4 shows the number of clock cycles for each algorithm and the corresponding execution time.

Table 4.4: *Execution time of SRA and SAMPDA.*

Algorithm	**Number of clock cycles**	**Execution time**
SRA	3660	1.42ms
SAMPDA	1214	0.47ms

Thus the maximum time required for reconfiguration and execution is 17.9ms + 1.42m = 19.32ms which is less than 38.9ms. Hence the timing constraints are satisfied by the system. However, these timings are obtained for the test setup. In a practical scenario, the algorithms have to adapt more than just two subcarriers which increases execution time. It has to be noted that the initial constraint of 38.9 ms has been derived for a user with a pedestrian speed who is close to the base station. This is not always true and practical considerations regarding this are explained in the next chapter.

If the Serial interface is used for reconfiguration, then the time will increase 32 times since only one input pin is available for this operation. Thus it is advisable to use the SelectMAP interface. However, since SelectMAP interface uses 32 pins of the FPGA, these have to be utilised for reconfiguration when it is demanded. After reconfiguration, these pins can be used for general I/O purposes [39]. But switching between I/O operation and reconfiguration operation can be very complicated. This would also further increase latency since to reconfigure, first the SelectMAP I/O have to be disabled, the FPGA has to be reconfigured, and then the pins for I/O operations have to be enabled.

As mentioned in the System Model phase, no parallelism has been included in the algorithms to speed up execution. From the area requirements it is seen that 90% of the area is needed for SAMPDA and only 27% for SRA and so 90% of the area on the FPGA has to be reserved for the Dynamic module. The remaining 63% of the area can be utilized to reduce the execution time of SRA by exploiting inherent parallelism. However, this is not suitable for SAMPDA. This is also necessary since SRA does not satisfy the timing constraints for WiMAX when adapting 256 subcarriers which is 153ms [1]. The design summary obtained from Xilinx ISE after implementing these modules are shown in Appendix C.

4.3 Conclusions

In this chapter, the implementation of the entire system on the FPGA is presented. The Static module is implemented after optimization. Based on the architectural challenges faced, it is seen that a practical PR system cannot not be implemented in this project. Also, the pin configurations on the development platform are not suitable to implement a PR system. Each module is then separately implemented as they would have been for a practical PR system. Communication between the Static and Dynamic modules through bus macros is also presented. The reconfiguration process and execution of the algorithm satisfied the timing constraint for WiMAX for the test setup. In the next chapter, the Static module is tested and the solutions to the challenges faced during implementation are discussed.

Chapter 5

Testing and Analysis

5.1 Introduction

In the previous chapter, the implementation challenges of the Static and Dynamic module are presented due to which a complete PR system is not implemented in this project. The issues in each of the four domains are analyzed and the possible solutions to them are proposed in this chapter. The testing of the Static module in a live setup is first presented and then the analysis.

5.2 Testing

There are two modules namely, Static and Dynamic in this project. The Dynamic module which consists of the LA algorithms, is the same as the one designed in [1] using Handel-C where it was also tested and validated. Since this module does not change from a design point of view, testing is not required. To validate the functionality of the Static module, it is tested in two steps. In the first step, the RSSI calculation is validated by comparing the System Generator design with a MATLAB code. This has been explained in the previous chapter. Once it is validated, the next step is the testing of the overall operation of the Static module with real time signals which is explained next.

5.2.1 Validation of Static module

The Static module is validated in a real setup which is presented first.

5.2.1.1 The Live Setup

Since the SNR in an interference free environment has to be calculated and so any interference existing in the lab is first determined. To do so, the receiver is first tuned to capture signals in the 2.4 GHz band and the output is seen on an oscilloscope. It is observed that there are random bursts of signal seen at the oscilloscope. This is because of the Wireless LAN network operating in the University. To avoid this, the transmitting frequency is changed to 2.6 GHz

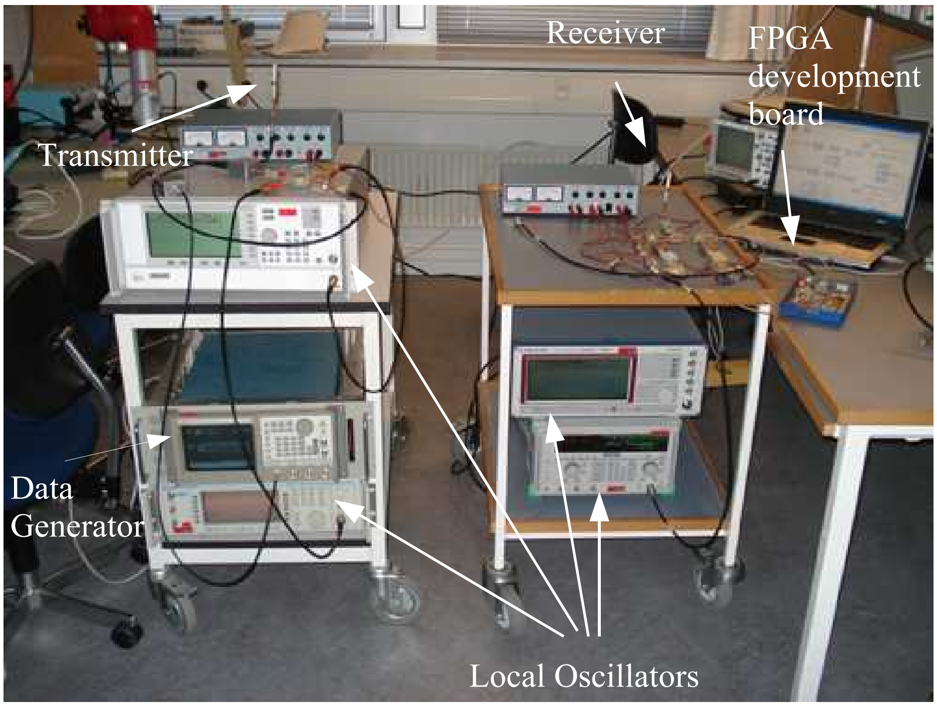


Figure 5.1: *Lab setup for testing the output of Decision algortihm. The hardware co-simulation is executed on the FPGA using the inputs given through the ADC.*

where no interference exists. A data generator produces a random sequence of 100 kHz which is fed to two local oscillators at frequencies of 2.6 GHz and 2.601 GHz. Thus the same sequence is transmitted from both the branches. The spacing between the two carriers is kept as 1 MHz which is acceptable considering the fact that the IF filter has a bandwidth of 600 kHz. The parameters at the transmitter are shown in Table 5.1.

Table 5.1: *Transmission parameters of the live setup for testing the Static module.*

Branch no.	**Frequency**	**Power level of the local oscillators**
1	2.600GHz	7dBm
2	2.601GHz	6dBm

At the receiver, bandpass sampling is employed which digitizes the signal at a frequency of 30 MHz. To down convert the signal at this IF, the local oscillators at the receiver are tuned at 2.630 GHz and 2.631 GHz. The first receiver branch down converts the carrier which is transmitted at 2.600 GHz and the second branch down converts the carrier transmitted at 2.601GHz. Thus at the receiver mixer the following operation takes place:

$$2.630GHz - 2.600GHz = 30MHz \tag{5.1}$$

$$2.631GHz - 2.601GHz = 30MHz \tag{5.2}$$

Now both the carriers are down converted to 30MHz which is passed through the IF filter and then sampled at 25 MHz. The maximum input voltage at the ADC is limited to 2.2 V peak-to-peak since it operates best in this range [21].

During hardware co-simulation, the *bit* file generated by System Generator is transferred to the FPGA. Validation of the RSSI calculation has been previously presented by comparing the output of the System Generator design with that of the MATLAB code. The next step is to validate the system in a live setup and observe the output of the Decision algorithm which is explained next.

5.2.1.2 Validation of Decision algorithm

Based on the instantaneous SNR value, the Decision algorithm provides an output which is used to load the appropriate LA algorithm. This output is validated manually since the ranges required for each algorithm are known. The Decision algorithm provides an output as 1 which indicates SRA and 2 for SAMPDA. The output of the Decision algorithm along with the average SNR values are shown in Appendix B.3 and it is seen that it functions correctly..

5.3 Analysis

In this section, the challenges encountered in each domain are analyzed and the possible solutions to them are proposed.

5.3.1 Algorithmic challenges

For the Static module, System Generator is used and for the Dynamic module, Handel-C is used and the compiler generates the VHDL file. So we do not have total control on how the hardware is utilized. As seen earlier, System Generator automatically uses the dedicated DSP resources on the FPGA. Due to this, the Static module is not placed on the area remaining on the FPGA and overlaps with the Dynamic module as shown in Figure 4.14. If VHDL is used for designing, then the designer can exploit the resources and this conflict can be avoided. However, for large designs, VHDL can be very cumbersome to use and a higher level language like Handel-C can be more convenient. Thus usage of System Generator and Handel-C is more suitable for prototyping systems. However, more designers can work together on smaller modules in VHDL and combine them to form one complete design as described in the Modular Design Flow in [32].

The size of the Dynamic module on the FPGA is decided by the largest reconfigurable module which is SAMPDA in our case. Due to its large area requirement, 90% of the area on the FPGA has to be reserved for the Dynamic module. This means that even though SRA requires only 27% of the hardware resources, the remaining area remains unused on the FPGA. This is a huge loss in terms of silicon area which defeats the whole purpose of having reconfigurable systems. Thus it is important that reconfigurable modules should consume the same area to have an efficient design. This can be achieved by optimizing the algorithms to reduce the need of computationally intensive blocks.

To do so, the SAMPDA algorithm should be optimized so that it occupies less area. During implementation, all default options of Xilinx ISE were used. If area optimization options are used, area requirement of SAMPDA reduces to 80% which is a huge gain in terms of FPGA area. Further, the Handel-C code for the algorithm should also be optimized. DSP resources in the Dynamic module also have been under utilized. If more of them are used, area requirement will also reduce. Optimizations can be performed at both the algorithmic design lvel and the implementation level. At the algorithmic design level, probability techniques can be used which allow the LA algorithm to anticipate the changes that occur in the channel. Then the corresponding changes to be done in the bit and power loadings will need less computations which in turn reduce hardware resources. These options were not explored at the beginning of implementation since the architectural limitations were not foreseen.

5.3.2 Architectural challenges

From the pin connections and size of the Dynamic module, it is seen that a PR system cannot be implemented in this project. To be able to successfully implement it, some changes are required in the connections between the ADC and the FPGA on the development platform used. The desired location of the pin connections are shown in Figure 5.2.

If all the ADC pins are connected to the left side of the FPGA, a partition on the right side for the Dynamic module is unaffected by them. This will allow seamless reconfiguration without interfering with the ADC connection pins. But in this scheme, no DSP resources are available and thus the Static module would not be successfully implemented. If the partitioning is done as in Figure 5.2 (b), DSP resources are available but not enough pins are available since they are only 26. In both the cases, the partition still crosses the center column which is not yet supported by the design tools provided by Xilinx Inc. To overcome this issue, multiple regions can be allotted and connected via a bus macro. This is shown in Figure 5.3.

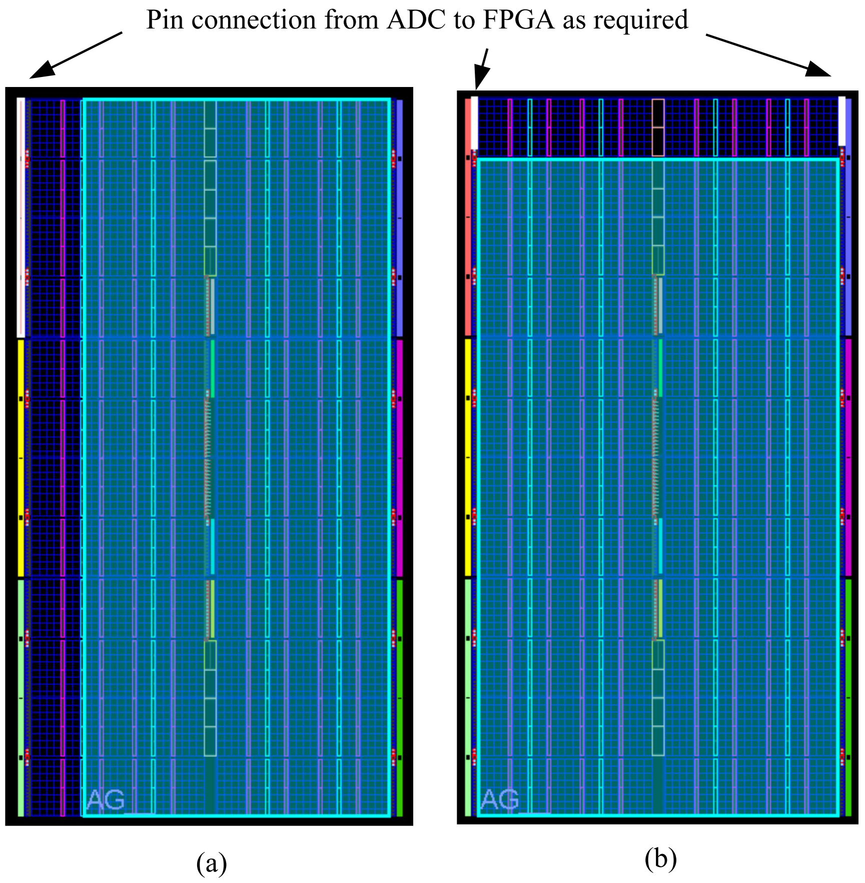


Figure 5.2: *Pin connection from ADC to FPGA as required marked by white spots. In (a), the pins are on the left side but this partitioning does not allow utilization of DSP resources. In (b), DSP resources are available but the number of pins that are connected to the remaining area are not sufficient.*

In Figure 5.3, the shaded regions are the partitions allotted for the Dynamic module which is split in two parts. This type of partitioning solves the problem of having a partition over the center column. But this also increases design complexity. The LA algorithm will have to be split in two separate VHDL descriptions for each part. Communication between the two parts will be through the I/O pins at the center column. This means that the pin connections have to be free and not used by any other I/O interfaces.

The intention to investigate partial reconfiguration was to see if the hardware area gained when using SRA instead of SAMPDA can be used. However, current limitations regarding reserving an area which is the size of the largest Dynamic module prohibit the usage of area gained in this process. If 90% of the area has to be reserved for LA all the time, then there is no gain in implementing a reconfigurable architecture since the free area on the Dynamic module cannot be used. However, if number of subcarriers go more than 16, then a reconfigu-

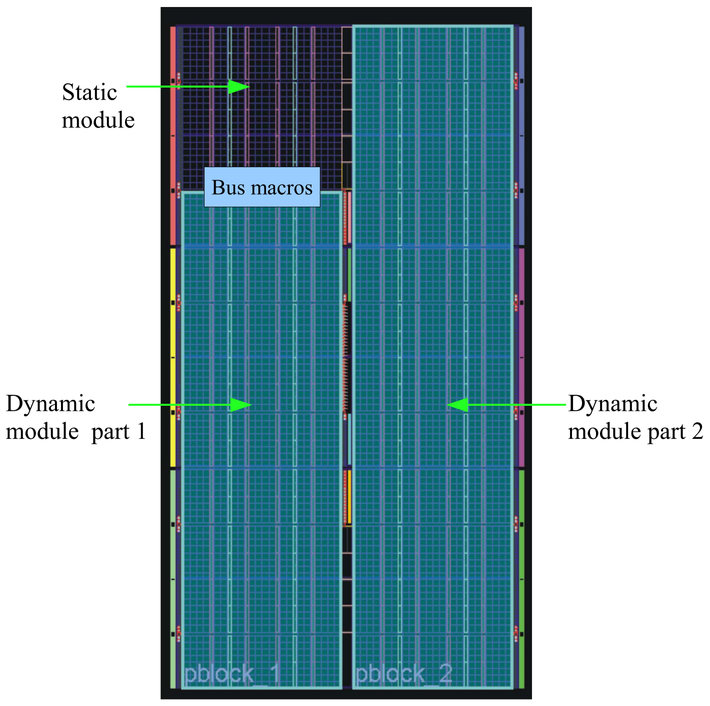


Figure 5.3: *Alternate method for partitioning the FPGA for large designs. The multiple Dynamic modules have a separate VHDL description and they communicate among them through the I/O pins at the center column.*

ration operation is necessary to load SRA. But execution time constraints for higher number of subcarriers is not satisfied by SRA [1].

This leads to an important area of research which is allowing usage of area on the Dynamic module when it is under utilized. This would require flexible bus macros which are able to transfer data from Static to Dynamic module even after change in placement of the Dynamic module. The area gain when using a Dynamic module of smaller size can be used for processing purposes that are not needed when the larger design is in place. This is because if the larger design also needs that particular processing then it will have to be implemented in some other way and so area gained in nullified. Thus usage of area gained has to be handled carefully.

5.3.3 Communication challenges

For managing the reconfiguration requirements of the FPGA the Interface FPGA on the development platform is used to configure the User FPGA. Nallatech has already provided a core to handle external memory interfaces. However, handling partial reconfiguration is not a part of the software provided. To facilitate this type of control, additional development is required. As shown in Figure 4.16, the Interface FPGA has to be able to interpret the output from the User FPGA and perform the necessary reconfiguration operation. Additionally, to autonomously manage this, an embedded processor such as MicroBlaze soft core processor or PowerPC processor can also be used inside the fabric of the user FPGA [28]. An external controller can also be a part of the system that is able to handle partial reconfiguration on demand basis.

The bus macros that are provided by Xilinx Inc. have a pre decided instantiation format. Each input and output has to be explicitly connected to the Dynamic module. If large amounts of data has to be exchanged, this can be very complicated. Nested bus macros can be used to transfer more than 8 bits of data but that still does not solve the problem of interfacing each connection. One solution to this can be that the format of instantiation of the bus macros should be the choice of the designer. This however means that more flexible bus macros should be provided by Xilinx Inc. which would reduce the complexity from the designer's point of view. An optimum solution between bus macro utilization and latency to transfer data into the Dynamic module is required. More bus macros will lead to faster data transfer but during reconfiguration, the bus macros have to be disabled and again enabled after the reconfiguration process is complete thus increasing design complexity.

5.3.4 Timing Challenges

The timing constraints are calculated for the test setup. However, in a practical scenario, such conditions do not exist. For mobile WiMAX (IEEE 802.16e), the speeds supported are upto 110 km/hr [4]. At this speed, the coherence time reduces to 1.6ms. The LA algorithms do satisfy this constraint in terms of their execution time. But if a reconfigurable system is employed, then it becomes unrealistic since reconfiguration time becomes too high as compared to the coherence time. By the time the decision algorithm provides the output, the channel changes significantly and choice of algorithm may become outdated. Hence it is not useful to have a PR system in such a case and only one of the two algorithms should be used. The choice would be the one providing the best performance in terms of its execution speed and spectral efficiency which is SAMPDA in this case.

5.4 Partially Reconfigurable LA in a Practical System

Based on our experience in this project, an architecture that can be used in a practical system employing reconfigurable LA algorithms is proposed and shown in Figure 5.4. To achieve a balance between SRA and SAMPDA, it is assumed that the area requirement will be half of the FPGA. The transmitter baseband processing is performed on a Virtex-4 FPGA and the reconfigurable LA is allotted half of the FPGA area based on the guidelines provided in [28]. The bit loadings for the subcarriers is sent back from the LA to the modulator which is a part of baseband processing. It is necessary in such a case that the implementation be as optimized as possible in order to fit in all baseband functionalities along with LA on one FPGA.

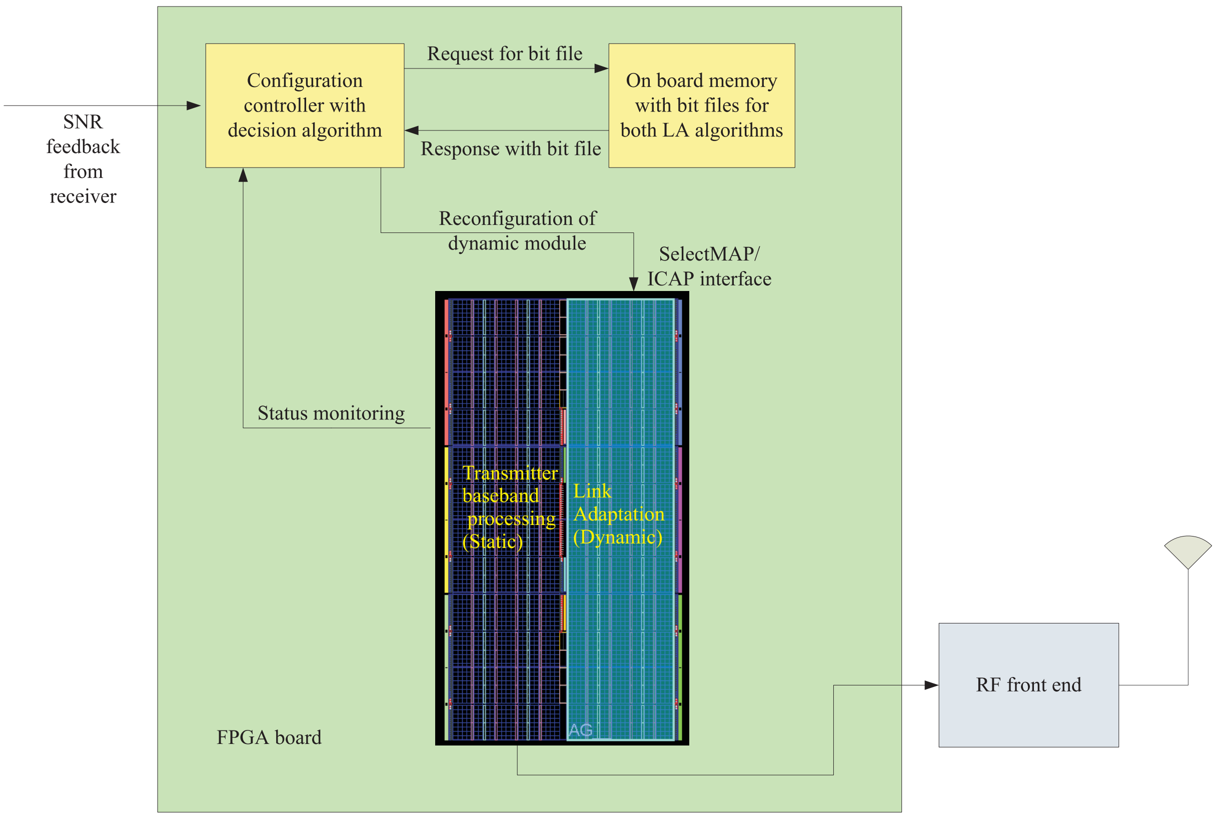

Configuration controller with decision algorithm
Request for bit file
On board memory with bit files for both LA algorithms
Response with bit file
SNR feedback from receiver
Reconfiguration of dynamic module
SelectMAP/ ICAP interface
Status monitoring
Transmitter baseband processing (Static)
Link Adaptation (Dynamic)
AG
RF front end
FPGA board

Figure 5.4: *Proposed architecture for reconfigurable transmitter with LA. A dedicated controller handles the reconfiguration operation thereby providing seamless reconfiguration.*

In a practical WiMAX scenario, the user equipment feeds back the channel state information to the base station. This channel state information provides the received SNR for that user based on each subcarrier or each subchannel. Two cases are investigated when 512 subcarriers are used. Even with 512 subcarriers, only 384 subcarriers carry data with 8 sub channels [5]. A sub channel considered here is a group of adjacent subcarriers. Thus each subchannel consists of 48 adjacent subcarriers.

1. **Case 1: Subchannelized link adaptation:** In this case, each subchannel is treated as a subcarrier and the SNR value is of the subchannel. When the transmitter gets a feedback from the users about the channel state or SNR, the configuration controller determines whether the right LA algorithm is running on the FPGA through the status monitoring interface. If that is not the case and a change in the LA algorithm is required, the controller requests the on-board memory for the other LA algorithm which is stored in the form of a bit stream. Upon receiving it, the controller reconfigures the FPGA and provides the SNR values of each sub channel which is then adapted accordingly. The advantage of having a dedicated controller is that it can independently handle the reconfiguration operation and the FPGA can be dedicated purely for signal processing operations. This also reduces reconfiguration latency. It may be possible that all 8 SNR values are transferred in parallel

to the LA algorithm. The timing constraint should be satisfied in this case for adapting 8 subchannels and not each subcarrier individually. Assuming that each subchannel is allocated to one user, the average SNR decides which algorithm has to be utilized.

2. **Case 2: subcarrier based link adaptation:** In this scenario, each of the 384 subcarriers are adapted individually. The user equipment will have to feedback the channel state information of each subcarrier. Based on this information, appropriate power and bit loading is performed. For more than 16 subcarriers, the decision algorithm chooses SRA and so this algorithm has to be optimized for this case. However, SRA does not satisfy the timing constraint for higher number of subcarriers [1] and so it has to be optimized. Providing the SNR values of 384 subcarriers will also be tricky. They can be transferred serially but then timing issues have to be considered. The latency of the memory element is critical since it should transfer the *bit* files as quickly as possible.

In Figure 5.4, the *bit* streams are available on the platform itself and so this is an endo-reconfigurable system. The advantage of having an on-board memory is that retrieval of bit streams is faster, thereby reducing reconfiguration time. At the same time, seamless upgradation of the bit streams is not possible since it would require manual reprogramming of the memory element. This limitation is not possessed in an exo-reconfigurable system since the *bit* file can be easily replaced with a new one. However, reconfiguration latency is higher in exo-reconfigurable systems. Thus a trade-off exists between upgradability and time [27]. In systems where time constraints are too stringent, an exo-reconfigurable system is not the right solution. However, in extremly time critical applications, access times for the memory element should also be accounted which depends on the type of hardware technology used.

To implement a PR system such that it improves the spectral efficiency, the limitations experienced have to be solved. Even if the architectural constraints are not present, the system has to be able to cope with the rapid changes in the radio environment. This however is severly limited due to the reconfiguration time of the FPGA.

5.5 Methodology Evaluation

We proposed the ATtACk methodology to proceed in a systematic manner toward implementing a partially reconfigurable system considering a development platform. In the implementation phase, the process as described by [28] has to be followed. Thus, using the ATtACk methodology, we were able to look at different aspects of the whole system which is necessary before beginning with the implementation phase. As systems become more complex in nature, a formal approach is needed which is described in ATtACk.

In the Architecture domain during the design phase, PE partitioning has been defined as the step to be performed. This is valid only for older FPGAs such as Virtex-II since the modules have to occupy the entire height of the FPGA and the pin configurations already decide how the device has to be divided into modules. Here the designer has to be aware of how the pins are utilized on the platform since this greatly influences partitioning of the FPGA. We thus recommend that this issue has to be looked into before deciding on the platform to be used. This is not required to be followed with Virtex-4 if reconfigurable modules are small because the pins are no longer on the outer edge of the fabric of the FPGA. After synthesizing, the design estimates obtained by Xilinx ISE can be used to partition the device. Thus PE partitioning becomes a part of the Implementation phase. However, if the PR regions are designated so that they occupy the entire height of the device, then pin restrictions come into picture. This is because the pins that are connected to the module can be used by that particular module only.

In the Time domain during the design phase, configuration time considerations has been mentioned. However, this can only be considered as a constraint to be followed since the actual timings are obtained after implementation is performed. In case timing constraints are not met, optimizations have to be done. In this situation, the designer may go back to the System Model phase to explore more parallelism if any were missed. Thus timing considerations lead to an iterative process. More exploration in the Architecture and Communication domain can also be performed thus leading to architecture space exploration. It is recommended to explore all the domains before attempting the implement a PR system.

The four phases in the ATtACk methodology: Specification or Requirements, System Model, Design and Implementation have guided us in the three domains namely Algorithm, Architecture and Communication that are of concern for PR in this project. Although Time has been described as a domain, it is to be considered in systems where it is a constraint. Based on that, the designer will have to optimally partition the FPGA. The way in which signals are routed through bus macros between different modules influences the execution time. In case timing requirements are not met, the partitioning process has to be repeated. Due to the fact that Virtex-4 provides more flexibility for creating partitions than its predecessors, the designer has many options. In this project, time has not been considered as a constraint and so we looked into this domain only after Implementation. We thus recommend that ATtACk should be used by designers who want to implement PR systems.

5.6 Conclusions

In this chapter, the issues regarding partial reconfiguration and it's impact on the LA algorithms are analyzed. Since PR is still under development, quite a few limitations exist in implementing it. Based on the architectural issues, we conclude that designs having large reconfigurable modules are not feasible to be implemented. The size of the module directly depends on the size of the design and so reconfigurable modules should be so designed that they occupy less area on the FPGA. If any of the modules being swapped do not require the area reserved, then the purpose of PR is defeated. For the LA algorithms, size is a major issue and further optimizations are necessary. Reducing area usage and at the same time reducing the execution time is challenging. This would require simplifying the design of the algorithms but this may lead to a compromise on the performance in terms of spectral efficiency which is of concern for cognitive radios. Also, the reconfiguration time is too high to be used in a practical situation. A reconfigurable architecture suffering from so many impairments, will not provide the desired benefits of improved spectral efficiency.

Based on the current technical possibilities, it is not advisable to use SRA and SAMPDA algorithms in a reconfigurable architecture unless optimizations are performed. These include reduction of hardware area and time requirement. The FPGA should also provide greater implementational flexibility. The limitations experienced in this project should be analyzed before attempting to implement PR systems.

Chapter 6

Conclusions

This chapter summarises and concludes the work done in this project. Based on the conclusions in the previous chapter, future areas of research are also suggested.

6.1 The Initial Problem

Future cognitive radios will operate in the spectrum holes that exist in the radio environment. In order to efficiently exploit the opportunity, spectral efficiency should be high. An OFDM based system is a good candidate for CR due to its inherent spectrum sensing abilities [10]. To improve spectral efficiency, Link Adaptation algorithms are used which adapt power and bit loading of each subcarrier based on their instantaneous SNR values. Two LA algorithms SRA and SAMPDA were implemented in [1] and it was seen that there is a trade off between spectral efficiency, execution time and hardware resource usage. To achieve a balance between the three, a reconfigurable system is investigated that is able to provide the best possible performance. Partial reconfiguration in FPGAs is a possibility due to technical advances that have taken place in the architecture of the FPGA. We investigated whether the PR functionality is feasible for the two LA algorithms SRA and SAMPDA for an OFDM system.

6.2 The Steps to Investigate the problem

The SNR value and number of subcarriers directly influence the choice of the algorithm and so these parameters are used to decide which LA algorithm has to be loaded on the FPGA. Variable subcarrier transmission requires a scalable transmitter and implementing it requires considerable effort. Because of this, the number of subcarriers are fixed as two considering the limitations of the development platform provided. Also, since the aim is to evaluate the feasibilty, this provision is enough and it can be scaled. Bandpass sampling is used to automatically downconvert the signal from IF to baseband. The baseband signal is then digitally filtered and the SNR is calculated.

To systematically move toward implementing a PR system, a well defined methodology does not exist to the best of our knowledge. We thus proposed ATtACk considering **A**lgorithm, **T**ime, **A**rchitecture and **C**ommunication as domains and divided them into four phases along which considerations and analysis are done. This methodology helped us in all the domains except Time since time was not considered as a constraint at the start of the project.

Moving along three domains, the requirements are first analyzed and the project setup is defined. The functionality of the algorithms are studied, the development platform is chosen and the input and output of the system is defined. Since in a PR system, modules can be differentiated as the one that does not require change (Static) and the one that has to be reconfigured (Dynamic), the blocks to be implemented are suitably differentiated. In the next phase, system model is defined and a flow chart of the two algorithms is created. The entities to be used on the development platform are also identified along with their interfaces. Then in the design phase, the SNR calculation scheme is presented. The implementation was divided into two modules, Static and Dynamic. The Static module provides the SNR output of each carrier and also the decision of which algorithm has to be loaded. The Dynamic module consists of the two LA algorithm that have to be loaded based on the output of the Decision algorithm. The FPGA is then studied to see how it can be partitioned. The required communication between them using bus macros is also analyzed.

In the next phase, the Static module using System Generator is implemented. Based on the area utilization, further optimizations are done to the design of the Static module. The issues and guidelines to be followed to implement a PR system are then studied. Based to the location of the pin connections from the ADC to the FPGA, it is concluded that a PR system cannot be implemented in this project. The size of the Dynamic module which had to be reserved overlapped with the pin locations. Because of this, the possibilities of implementing a PR system had the pin connections not been an issue are explored. It is seen that due to the usage of DSP resources on the FPGA, only one type of partitioning is possible. It is also observed that due to the large size of the Dynamic module, it's partition crossed the center column which is not yet allowed in Virtex-4 FPGAs. Further, specific tool requirements and their related documentation for PR systems are not yet commercially provided by Xilinx Inc. which are critical to successfully design PR systems on Virtex-4 FPGAs.

The communication requirements are then explored to interfacing the two modules using bus macros. Partial reconfiguration also requires a special controller with specific interfaces to the FPGA to handle these operations which is not available on the development platform. Based on a modular implementation, the timing requirements are analyzed and it is observed that the two algorithms do satisfy the timing constraints. This was analyzed by taking into account the reconfiguration time and the execution time of each algorithm.

Then the Static module is tested in a live setup since the Dynamic module is unchanged from where they were first implemented in [1]. The difficulties possesed in implementing a complete PR system is analyzed and possible solutions are proposed. The development platform considered is not suitable for partial reconfiguration which had to be identified earlier. With the right development platform and software tools it is possible to implement a PR system.

6.3 Perspectives

6.3.1 Short Term Perspectives

Considering the size of the algorithms, it was necessary that the requirement of SAMPDA be reserved for the Dynamic module. This led to under utilization of resources when SRA has to operate since it led to a surplus of 63% of area. In order to reduce this gap, SAMPDA algorithm has to be further optimized in terms of it's implementation. The complexity of this algorithm can also be reduced so that the requirement of computationally intensive resources can reduce. This algorithm does satisfy the timing constraint and this property should not be hampered in the optimization process. Algorithmic level improvements will thus be critical in this case. The dedicated DSP resources are not exploited in this project and doing so will reduce execution time as well as area requirements. This option has to be exploited in order to make full use of the resources available on the FPGA.

Pin connections largely influence placing of the Dynamic module which restrict FPGA partitioning when they are large in size which has been the case in this project. To overcome this, the large module should be split into multiple submodules which are interconnected using bus macros. Each submodule will be independant performing a limited set of operations. By doing so, more flexibility in partitioning the FPGA is possible.

Using System Generator and Handel-C as a design language restricts control on hardware. Converting the design into VHDL manually will provide more choices for the designer and this option can be explored. Development boards that provide PR functionality are available commercially and they can be used to analyze the feasibility for a fully functional PR system.

6.3.2 Long Term Perspectives

Current architectural limitations possessed by Virtex-4 do not allow flexible resource allocation for the Dynamic modules and so hardware design improvements concerning this limitation have to be considered. Utilization of unused area in the Dynamic module would allow further flexible implementation possibilities and will benefit the development of software defined radios. Communication within the FPGA is an area of concern since implementational restrictions do not allow the designer to have large flexibility. This topic has been addressed in [43] where using a new type of reconfiguration technique, flexible bus macros have been implemented. However, the problem of interfacing each bit individually is not solved.

Restricting height of the Dynamic module to 16 CLBs and it's multiple may lead to non-optimum implementation. Full flexibility for partitioning the FPGA as per the requirement will lead to more efficient resource utilization. The Dynamic modules can then occupy an area of any arbitrary shape on the board. A method to relocate the modules at run time is also presented in [43] but it leads to additional overheads in the bit file resulting in additional time requirement for reconfiguration.

Reducing the reconfiguration times will also lead to improvement in timing performance. This can be achieved by reducing the size of the bit file. One way to do so is by reducing the area requirement of the modules but a tradeoff exists between execution time and area occupied. Further, FPGA reconfiguration times can be improved by better configuration techniques where bit files can be loaded on the FPGA faster than the existing speeds.

Thus to summarize the perspectives,

- Short term goal: Optimize the algorithms to reduce area usage without increasing execution time.
- Long term goal: Provide more flexibility in the implementation of PR systems.

Appendix A

Basics of OFDM and Calculation of Timing Constraint

A.1 Orthogonal Frequency Division Multiplexing (OFDM)

This section is reproduced from [1].

"In a high data rate system using wide band channels, the symbol duration is larger than the coherence time of the channel which leads to frequency selective fading. Due to multi path effect of the wireless channel, some delay spread is introduced and if the delay spread is equal or greater than the symbol duration, Inter Symbol Interference (ISI) occurs where one symbol overlaps the adjacent symbol thereby introducing errors. In OFDM, a wide band channel is split up into multiple narrow band channels called as subcarriers. A narrow band channel has a larger symbol period as compared to a wide band channel and thus, Inter Symbol Interference (ISI) is minimized. This is the principle behind OFDM. By using multiple narrow band channels, a wide band is occupied thereby achieving large data rates and low ISI. OFDM is being used in existing standards like Wireless LAN, Digital Audio Broadcast (DAB), WiMAX and is a contender for 4G systems. Spectrum of an OFDM channel is as shown in Figure A.1.

Figure A.1 shows that the spectral peak of one subcarrier coincides with the null of all other subcarriers. This makes the subcarriers orthogonal. The basic transceiver system of OFDM is as shown in Figure A.2.

The binary data is first passed through the Error correction coding and interleaving block to make the signal robust and immune to deep fades. Further, modulation or symbol mapping is performed and pilot symbols are inserted to allow the receiver to do channel estimation.

A serial to parallel converter is applied and an Inverse Fast Fourier Transform (IFFT) operation is performed. Cyclic prefix is then appended to the data in order to make the signal

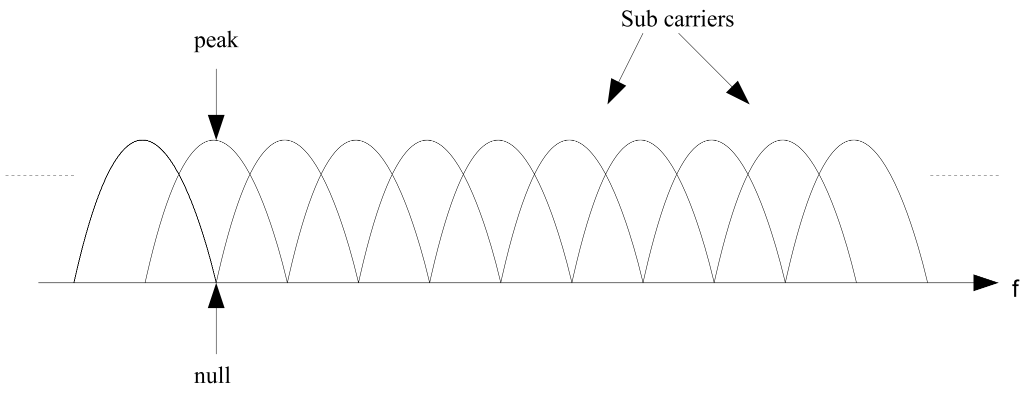


Figure A.1: *OFDM spectrum with orthogonal subcarriers.*

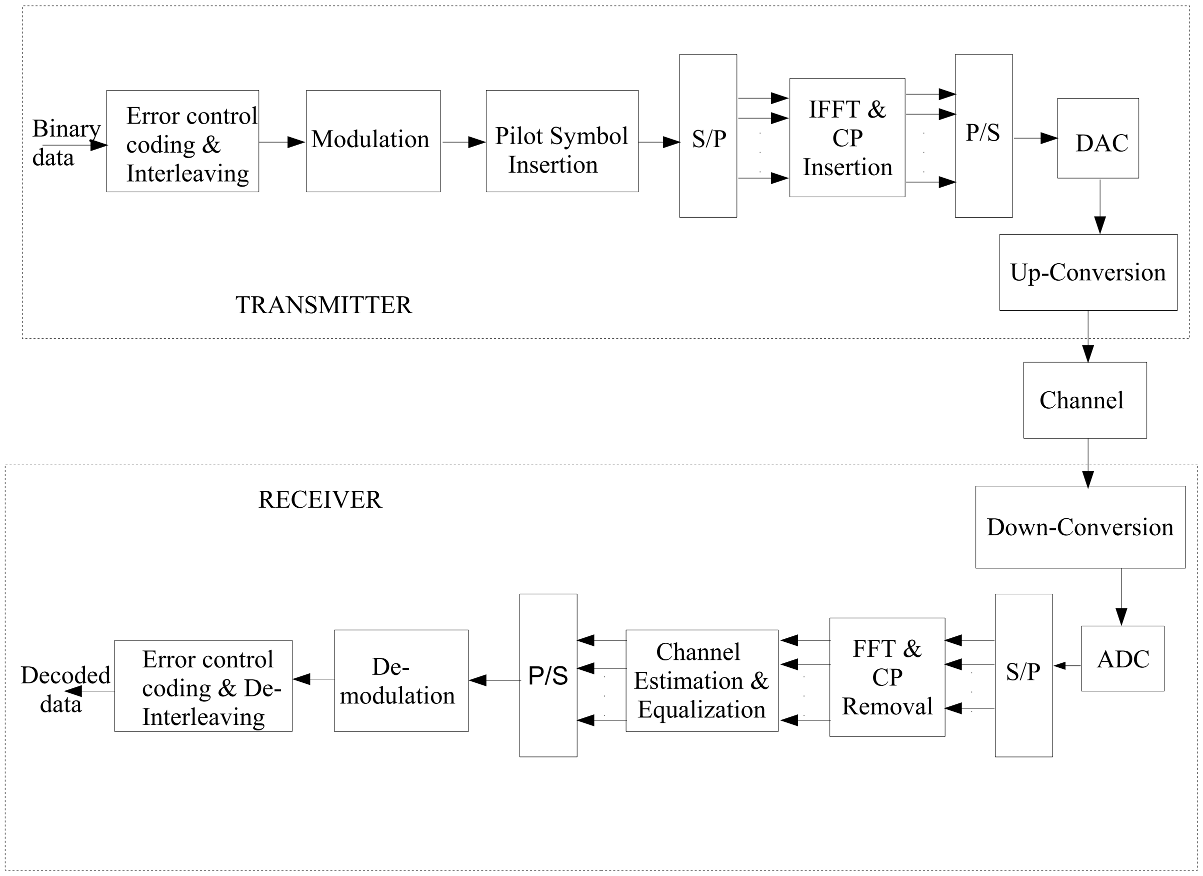


Figure A.2: *Block diagram of OFDM transceiver [1].*

orthogonal and to reduce ISI. Further, the data is converted to analog via a Digital to Analog Converter (DAC) and then up converted and transmitted.

At the receiver, signal is first down converted, and then passed through the Analog to Digital Converter (ADC). The cyclic prefix is removed and the signal in frequency domain is obtained from the FFT block. Based on the pilot symbols, channel estimation is done and since the data is in parallel form, a parallel to serial conversion is performed. Further, the signal is demodulated and passed through error correction and de-interleaving blocks.

Advantages of OFDM is that it provides high data rates even though the symbol period is greater as compared to a normal wideband signal thereby combating ISI. Since the channels now occupy a narrow band, a single tap equalizer can be used thereby reducing equalizer complexity. In order to provide high data rates, spectral efficiency has to be good which is largely influenced by channel dynamics. Link Adaptation is a technique to improve the spectral efficiency where power and/or modulation scheme is changed based on the current channel conditions" [1].

A.2 Timing constraint

The channel characteristics remain constant during the coherence time of the channel. During this coherence time, the transmitter has to transmit one frame and acquire the channel information for the transmission of next frame and accordingly adapt it's transmission parameters. The channel information should not be outdated for transmitting the next frame. Thus the three operations should take place within the coherence time of the channel [1]. Therefore,

Coherence Time $\geq$ Processing time of the transmitter + 3 $\times$ propogation time between transmitter and receiver. [1]

The system has to satisfy the timing constraint provided by the WiMAX standard. We calculate timing constraint for a speed of 0.834 m/s (3 km/hr). Considering the operation at 2.6 GHz which is used in the test setup, the Doppler shift is given as:

$$f_d = \frac{\upsilon}{\lambda}$$

$$f_d = \frac{0.834 m/s}{0.115 m}$$

$$f_d = 7.25 Hz$$

where, f_d is the Doppler frequency, υ is the speed of vehicle and λ is the wavelength at the frequency 2.6 GHz. The coherence time (T_c) of the channel, corresponding to the Doppler shift of 7.25 Hz is calculated as [41]:

$$T_c = \sqrt{\frac{9}{16 \cdot \pi \cdot f_d^2}}$$

$$T_c = \sqrt{\frac{9}{16 \cdot \pi \cdot 7.25^2}}$$

$$T_c = 58.36ms$$

After calculating coherence time, it is assumed that:

Time constraint for LA = T_c - (Processing time of the transmitter + 3 $\times$ propagation time)

Assuming that the receiver and the transmitter are located 10 meters apart [**?**]. Thus, propagation time comes as $\frac{10m}{3\times10^8 m/s} = 0.034\mu s$. Hence, time constraint is calculated as:

Time constraint for LA = 58.36 - 3 $\times$ 0.000034 ms - Processing time $\approx$ 58.36 - Processing time

If we assume that the processing of signal at the transmitter takes roughly one-third of the total coherence time, that is,

Processing time of signal at the transmitter $\approx$ 19.45 ms

then, we can say that for a speed of 3 km/hr and transmitter-receiver separation of 10 meters:

Time constraint for LA $\approx$ 38.9 ms

Appendix B

RF Chain Considerations and Test Results

B.1 Gain of the RF chain used

The RF gain of the receiver chain used in this project is calculated as follows. A signal with a known power without modulation from a signal generator is given to the receiver chain. By using a Power Meter measurements are taken at the end of the RF chain. The RF gain is measured as the difference in the power levels at the signal generator and power meter measurement. This gives approximately 42 dB.

In order to calculate the equivalent noise figure (NF_{tot}) of the receiver as given in Equation 3.6 only two components in RF set up are considered. The bandpass filter with a gain of -1 dB, noise figure of 1 dB and Low Noise Amplifier (LNA) with a gain of 13 dB, noise figure of 4.5 dB. The gain of the LNA is high which makes the ratio of NF to gain very low for further stages. The equivalent noise figure is as shown in Equation B.1.

$$NF_{tot} = 10^{0.1} + \frac{10^{0.45} - 1}{10^{-0.1}} \tag{B.1}$$

$$NF_{tot}(dB) = 1.2589 + \frac{2.8184 - 1}{0.7943} = 3.5481 = 5.5dB \tag{B.2}$$

The gain and noise figure values of the RF components used in the above calculations are taken from the discussion with Kim Olesen[1]

B.2 Testing Results of RSSI Calculation

RSSI is validated by applying a sine wave to the Figure shown in 4.3 and the filtered outputs are stored in MATLAB workspace. Those filtered outputs are given as input to rssi.m to compare its output with system generator design output.

[1]Laboratory Engineer, Antenna and Propagation Division, Aalborg University.

Table B.1: *Outputs of MATLAB code for RSSI calculation and that of the System Generator design.*

RSSI from System Generator	**RSSI from MATLAB code**
45.2454	46.0761
279.7072	281.7246
19.1528	19.9636
44.5053	46.8423
263.7503	273.3766
114.6024	117.512
34.8491	36.5558
279.072	289.5792
178.8954	183.9744
20.7303	21.5998

The actual RSSI is the number obtained, multiplied by 10^{-15} in $mWatt$. This operation is not performed since it gets cancelled by noise power when calculating the SNR. The values obtained differ slightly which may be due to the limited number of data bits provided to the System Generator design.

B.3 Testing Results in the Live Setup for Validating Decision Algorithm

A live setup is used to validate the operation of the decision algorithm. The SNR values of the two carriers are calculated on the FPGA and observed on a MATLAB workspace along with the output of the Decision algorithm. The SNR values cannot be compared theoretically since the signal is live. Table B.2 shows the SNR values of the two carriers and the corresponding Decision algorithm output where 1 represents SRA and 2 represents SAMPDA. The values are in linear form and comparison is done on these values.

Test runs were conducted in different conditions and each time 100 values were obtained but only a few are listed to maintain brevity.

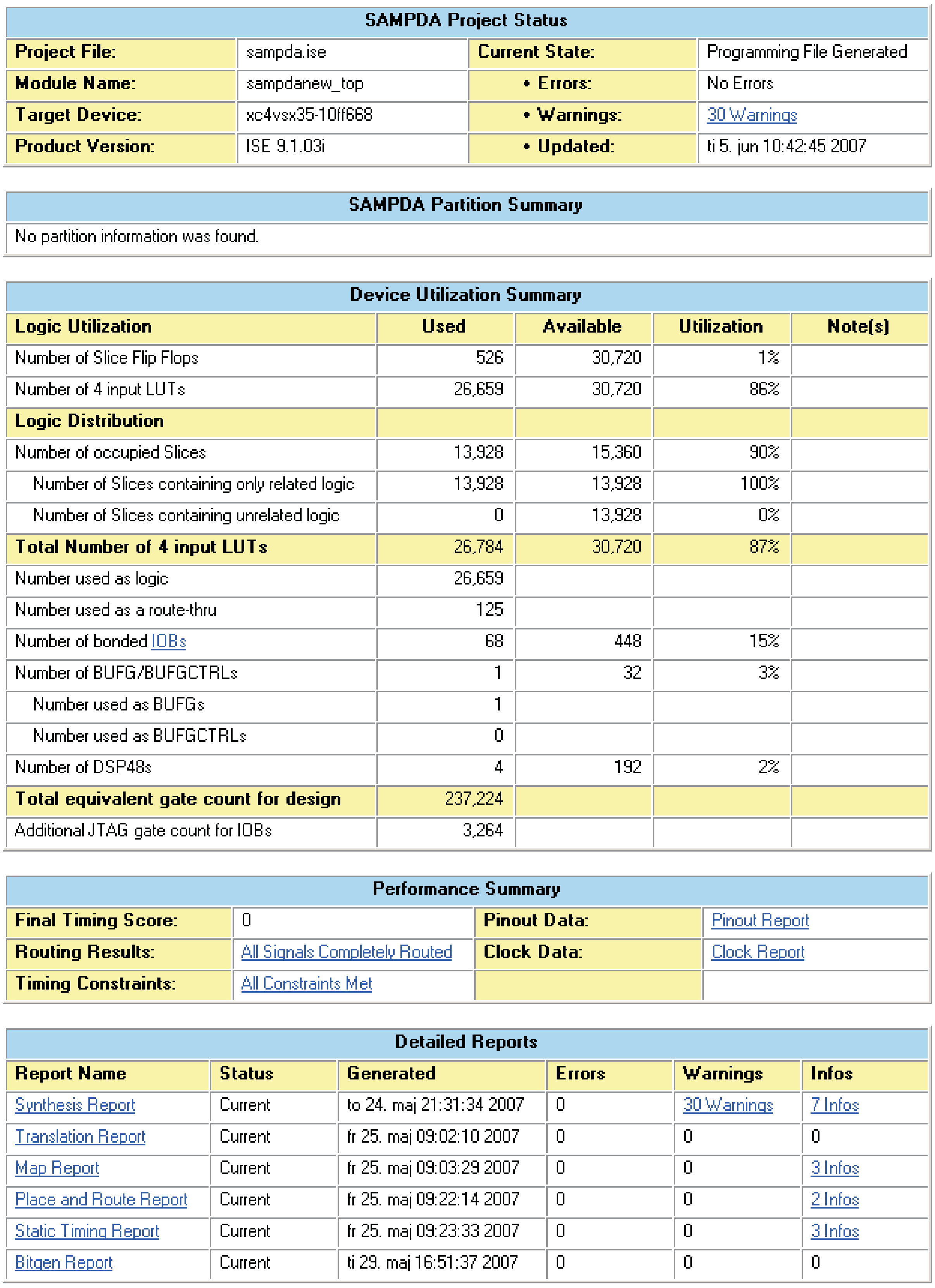
Table B.2: *SNR values of the two carriers and the corresponding output of the Decision algorithm.*

Carrier 1	**Carrier 2**	**Decision algorithm output**
0.780212	9.789276	1
272.7618	117.2879	2
3148.5143	168.9556	2
233.7032	204.8197	2
2.066406	2.640625	1
103.81	83.36813	2
1031.184	1710.194	2
341.2796	135.2684	2
649.9263	500.4658	2
1039.59	939.3866	2

Appendix C

Design Summary from Xilinx ISE Implementations

The design summary reports obtained from Xilinx ISE are shown here.



SAMPDA Project Status			
Project File:	sampda.ise	**Current State:**	Programming File Generated
Module Name:	sampdanew_top	**• Errors:**	No Errors
Target Device:	xc4vsx35-10ff668	**• Warnings:**	30 Warnings
Product Version:	ISE 9.1.03i	**• Updated:**	ti 5. jun 10:42:45 2007

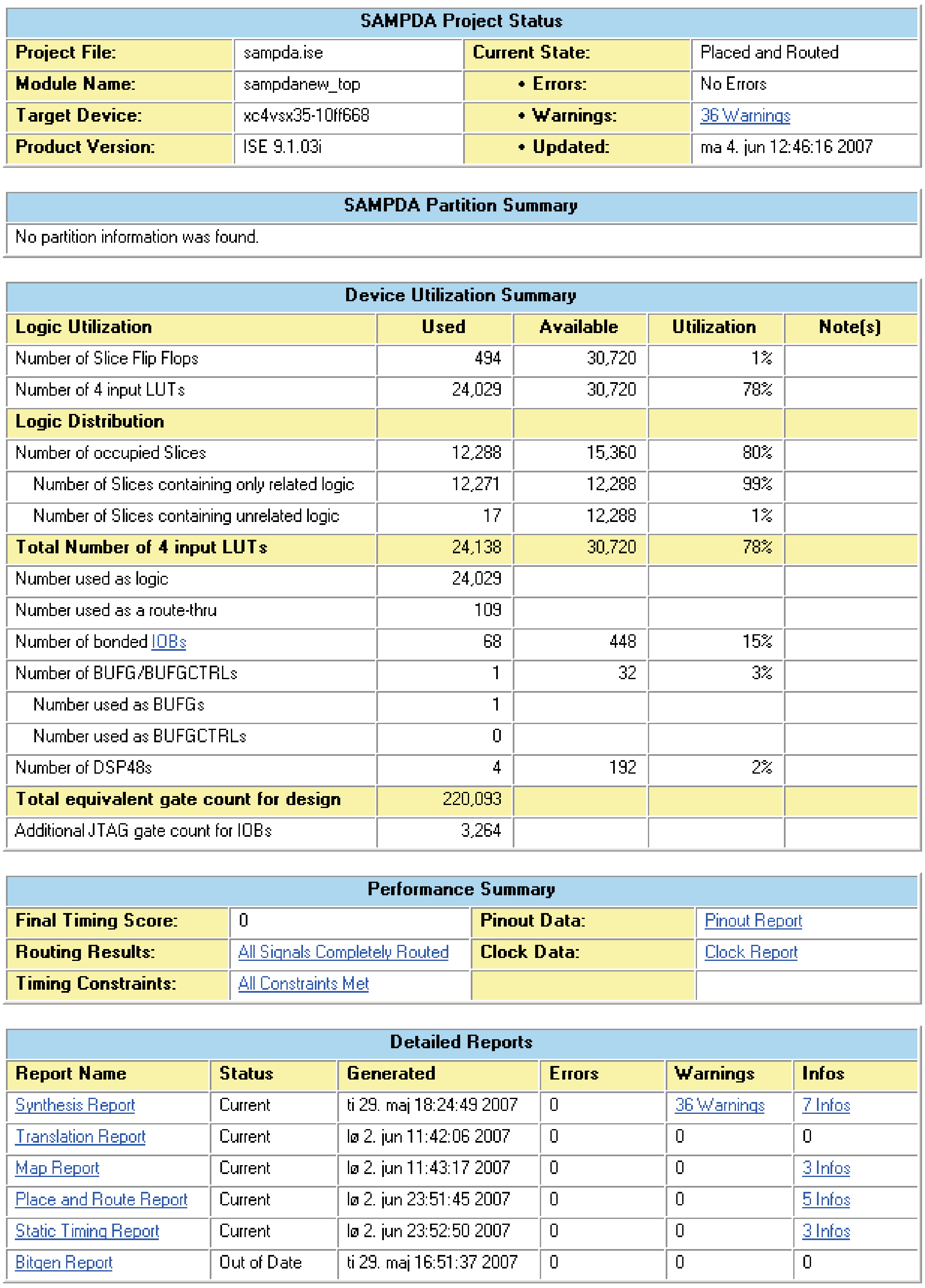
SAMPDA Partition Summary
No partition information was found.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	526	30,720	1%	
Number of 4 input LUTs	26,659	30,720	86%	
Logic Distribution				
Number of occupied Slices	13,928	15,360	90%	
Number of Slices containing only related logic	13,928	13,928	100%	
Number of Slices containing unrelated logic	0	13,928	0%	
Total Number of 4 input LUTs	26,784	30,720	87%	
Number used as logic	26,659			
Number used as a route-thru	125			
Number of bonded IOBs	68	448	15%	
Number of BUFG/BUFGCTRLs	1	32	3%	
Number used as BUFGs	1			
Number used as BUFGCTRLs	0			
Number of DSP48s	4	192	2%	
Total equivalent gate count for design	237,224			
Additional JTAG gate count for IOBs	3,264			

Performance Summary			
Final Timing Score:	0	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

Detailed Reports					
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	to 24. maj 21:31:34 2007	0	30 Warnings	7 Infos
Translation Report	Current	fr 25. maj 09:02:10 2007	0	0	0
Map Report	Current	fr 25. maj 09:03:29 2007	0	0	3 Infos
Place and Route Report	Current	fr 25. maj 09:22:14 2007	0	0	2 Infos
Static Timing Report	Current	fr 25. maj 09:23:33 2007	0	0	3 Infos
Bitgen Report	Current	ti 29. maj 16:51:37 2007	0	0	0

Figure C.1: *Design summary as obtained in Xilinx ISE for SAMPDA. 90% of the slice area is occupied.*



SAMPDA Project Status			
Project File:	sampda.ise	**Current State:**	Placed and Routed
Module Name:	sampdanew_top	**• Errors:**	No Errors
Target Device:	xc4vsx35-10ff668	**• Warnings:**	36 Warnings
Product Version:	ISE 9.1.03i	**• Updated:**	ma 4. jun 12:46:16 2007

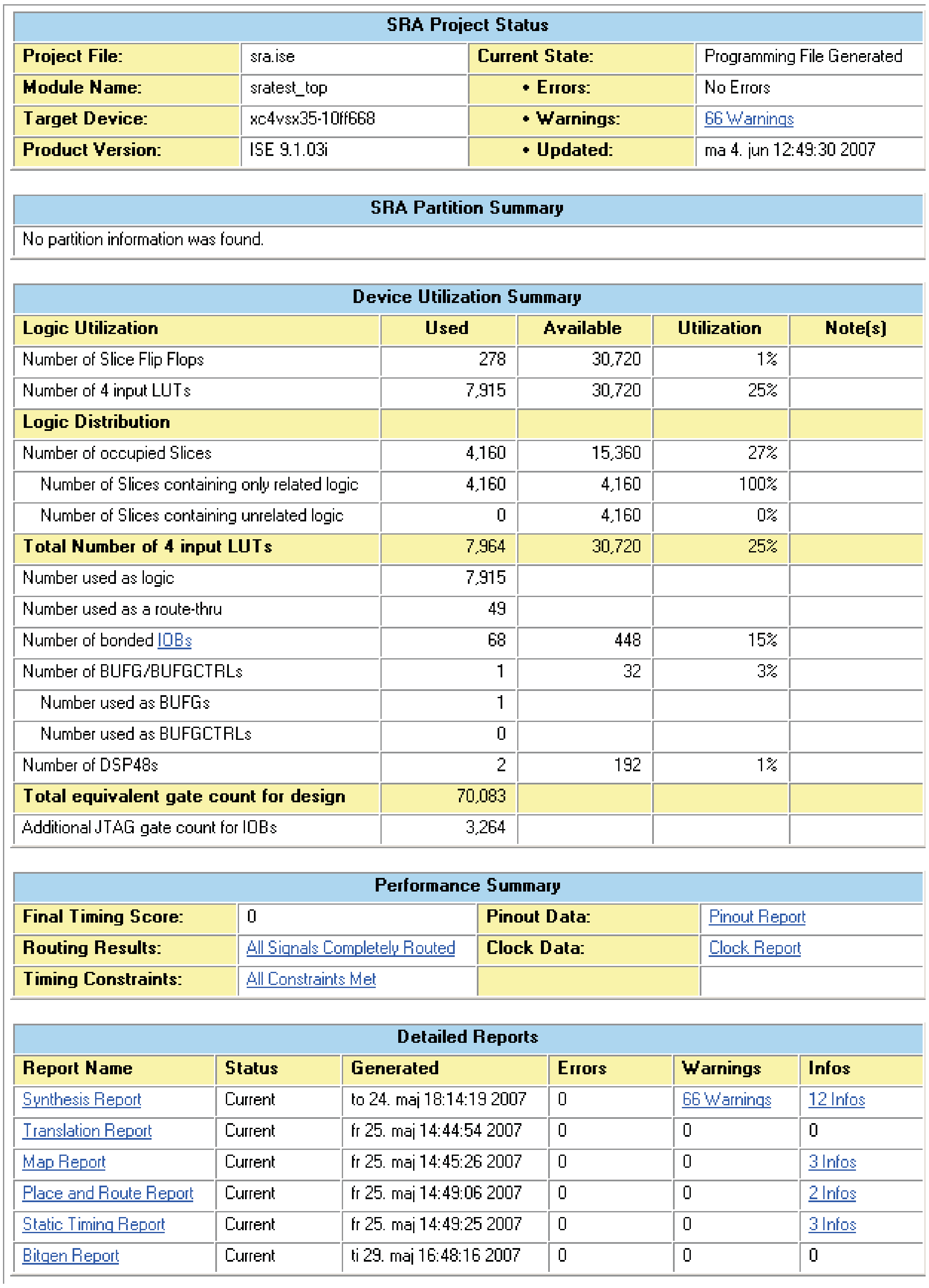
SAMPDA Partition Summary
No partition information was found.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	494	30,720	1%	
Number of 4 input LUTs	24,029	30,720	78%	
Logic Distribution				
Number of occupied Slices	12,288	15,360	80%	
Number of Slices containing only related logic	12,271	12,288	99%	
Number of Slices containing unrelated logic	17	12,288	1%	
Total Number of 4 input LUTs	24,138	30,720	78%	
Number used as logic	24,029			
Number used as a route-thru	109			
Number of bonded IOBs	68	448	15%	
Number of BUFG/BUFGCTRLs	1	32	3%	
Number used as BUFGs	1			
Number used as BUFGCTRLs	0			
Number of DSP48s	4	192	2%	
Total equivalent gate count for design	220,093			
Additional JTAG gate count for IOBs	3,264			

Performance Summary			
Final Timing Score:	0	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

Detailed Reports					
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	ti 29. maj 18:24:49 2007	0	36 Warnings	7 Infos
Translation Report	Current	lø 2. jun 11:42:06 2007	0	0	0
Map Report	Current	lø 2. jun 11:43:17 2007	0	0	3 Infos
Place and Route Report	Current	lø 2. jun 23:51:45 2007	0	0	5 Infos
Static Timing Report	Current	lø 2. jun 23:52:50 2007	0	0	3 Infos
Bitgen Report	Out of Date	ti 29. maj 16:51:37 2007	0	0	0

Figure C.2: *Design summary as obtained in Xilinx ISE for SAMPDA when area optimization options are used. Slice area utilization goes down to 80%*



SRA Project Status			
Project File:	sra.ise	**Current State:**	Programming File Generated
Module Name:	sratest_top	**• Errors:**	No Errors
Target Device:	xc4vsx35-10ff668	**• Warnings:**	66 Warnings
Product Version:	ISE 9.1.03i	**• Updated:**	ma 4. jun 12:49:30 2007

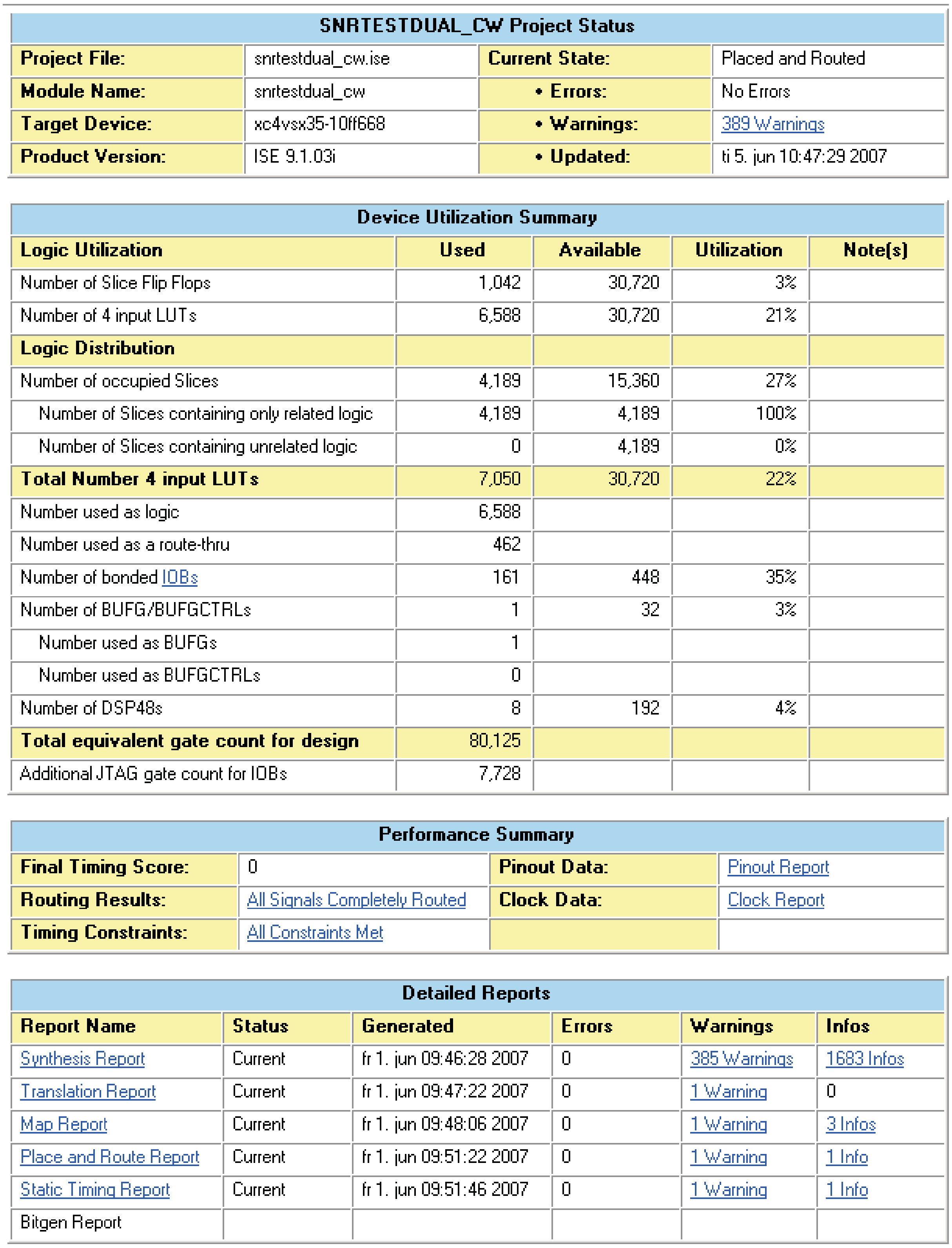
SRA Partition Summary
No partition information was found.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	278	30,720	1%	
Number of 4 input LUTs	7,915	30,720	25%	
Logic Distribution				
Number of occupied Slices	4,160	15,360	27%	
Number of Slices containing only related logic	4,160	4,160	100%	
Number of Slices containing unrelated logic	0	4,160	0%	
Total Number of 4 input LUTs	7,964	30,720	25%	
Number used as logic	7,915			
Number used as a route-thru	49			
Number of bonded IOBs	68	448	15%	
Number of BUFG/BUFGCTRLs	1	32	3%	
Number used as BUFGs	1			
Number used as BUFGCTRLs	0			
Number of DSP48s	2	192	1%	
Total equivalent gate count for design	70,083			
Additional JTAG gate count for IOBs	3,264			

Performance Summary			
Final Timing Score:	0	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

Detailed Reports					
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	to 24. maj 18:14:19 2007	0	66 Warnings	12 Infos
Translation Report	Current	fr 25. maj 14:44:54 2007	0	0	0
Map Report	Current	fr 25. maj 14:45:26 2007	0	0	3 Infos
Place and Route Report	Current	fr 25. maj 14:49:06 2007	0	0	2 Infos
Static Timing Report	Current	fr 25. maj 14:49:25 2007	0	0	3 Infos
Bitgen Report	Current	ti 29. maj 16:48:16 2007	0	0	0

Figure C.3: *Design summary as obtained in Xilinx ISE for SRA. 27 % of the slice area is utilized.*



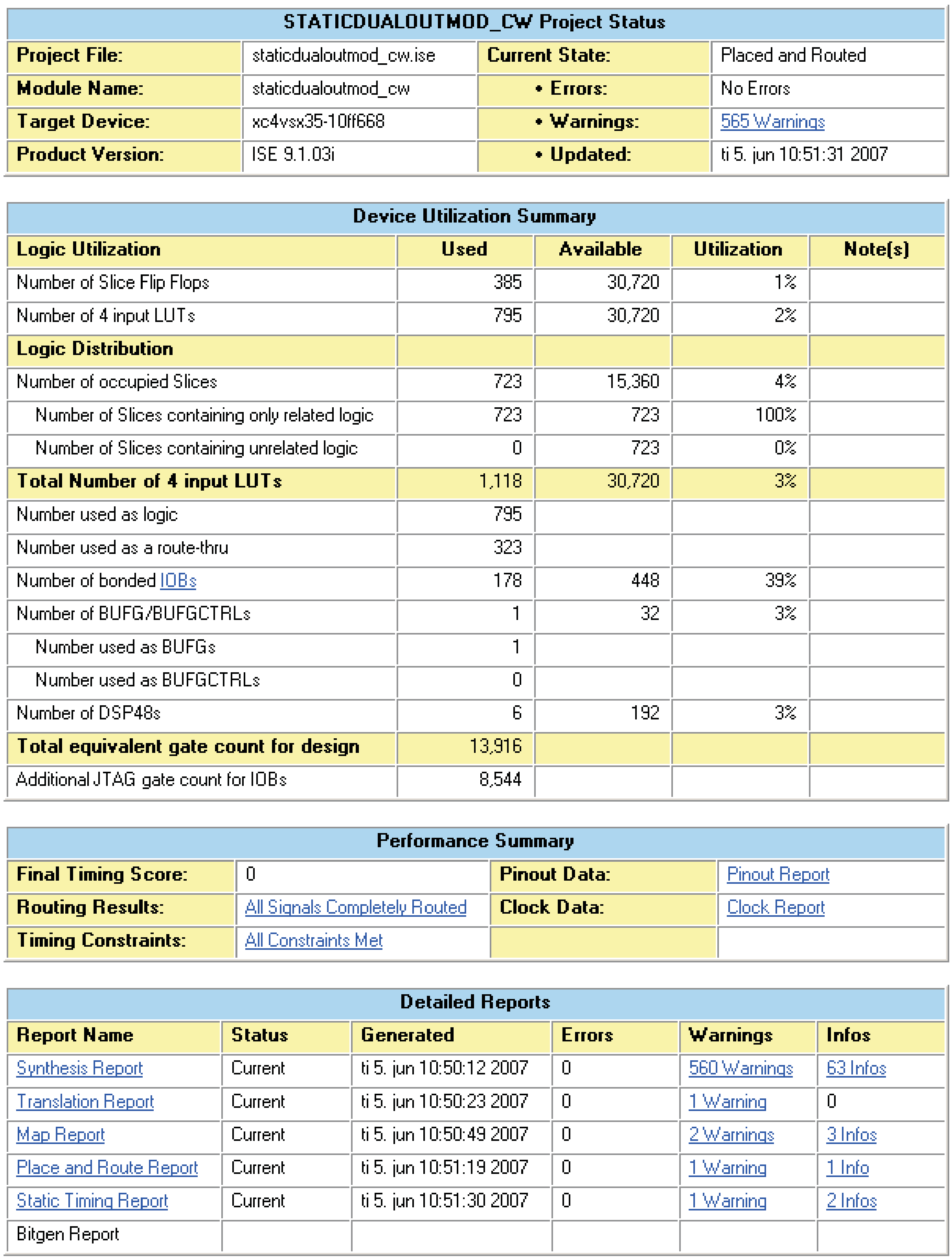
SNRTESTDUAL_CW Project Status			
Project File:	snrtestdual_cw.ise	**Current State:**	Placed and Routed
Module Name:	snrtestdual_cw	**• Errors:**	No Errors
Target Device:	xc4vsx35-10ff668	**• Warnings:**	389 Warnings
Product Version:	ISE 9.1.03i	**• Updated:**	ti 5. jun 10:47:29 2007

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	1,042	30,720	3%	
Number of 4 input LUTs	6,588	30,720	21%	
Logic Distribution				
Number of occupied Slices	4,189	15,360	27%	
Number of Slices containing only related logic	4,189	4,189	100%	
Number of Slices containing unrelated logic	0	4,189	0%	
Total Number 4 input LUTs	7,050	30,720	22%	
Number used as logic	6,588			
Number used as a route-thru	462			
Number of bonded IOBs	161	448	35%	
Number of BUFG/BUFGCTRLs	1	32	3%	
Number used as BUFGs	1			
Number used as BUFGCTRLs	0			
Number of DSP48s	8	192	4%	
Total equivalent gate count for design	80,125			
Additional JTAG gate count for IOBs	7,728			

Performance Summary			
Final Timing Score:	0	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

Detailed Reports					
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	fr 1. jun 09:46:28 2007	0	385 Warnings	1683 Infos
Translation Report	Current	fr 1. jun 09:47:22 2007	0	1 Warning	0
Map Report	Current	fr 1. jun 09:48:06 2007	0	1 Warning	3 Infos
Place and Route Report	Current	fr 1. jun 09:51:22 2007	0	1 Warning	1 Info
Static Timing Report	Current	fr 1. jun 09:51:46 2007	0	1 Warning	1 Info
Bitgen Report					

Figure C.4: *Design summary as obtained in Xilinx ISE for the Static module before optimization. 27% of the slice area is utilized.*



STATICDUALOUTMOD_CW Project Status			
Project File:	staticdualoutmod_cw.ise	**Current State:**	Placed and Routed
Module Name:	staticdualoutmod_cw	• **Errors:**	No Errors
Target Device:	xc4vsx35-10ff668	• **Warnings:**	565 Warnings
Product Version:	ISE 9.1.03i	• **Updated:**	ti 5. jun 10:51:31 2007

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	385	30,720	1%	
Number of 4 input LUTs	795	30,720	2%	
Logic Distribution				
Number of occupied Slices	723	15,360	4%	
Number of Slices containing only related logic	723	723	100%	
Number of Slices containing unrelated logic	0	723	0%	
Total Number of 4 input LUTs	1,118	30,720	3%	
Number used as logic	795			
Number used as a route-thru	323			
Number of bonded IOBs	178	448	39%	
Number of BUFG/BUFGCTRLs	1	32	3%	
Number used as BUFGs	1			
Number used as BUFGCTRLs	0			
Number of DSP48s	6	192	3%	
Total equivalent gate count for design	13,916			
Additional JTAG gate count for IOBs	8,544			

Performance Summary			
Final Timing Score:	0	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

Detailed Reports					
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	ti 5. jun 10:50:12 2007	0	560 Warnings	63 Infos
Translation Report	Current	ti 5. jun 10:50:23 2007	0	1 Warning	0
Map Report	Current	ti 5. jun 10:50:49 2007	0	2 Warnings	3 Infos
Place and Route Report	Current	ti 5. jun 10:51:19 2007	0	1 Warning	1 Info
Static Timing Report	Current	ti 5. jun 10:51:30 2007	0	1 Warning	2 Infos
Bitgen Report					

Figure C.5: *Design summary as obtained in Xilinx ISE after optimization. Slice utilization goes down to 4%.*

Bibliography

[1] G. Kulkarni et. Al. "SDR Implementation of Link Adaptation Algorithm for OFDM based Mobile Broadband Wireless Access", 9th semester project report, *Aalborg University*, 2006

[2] Simon Haykin, "‘Cognitive Radio: Brain Empowered Wireless Communications."’ *IEEE Journal on Selected areas in Communications*, Vol 23, No.2, February 2005.

[3] J. Mitola III, "‘Cognitive Radio, An Integrated Agent Architecture for Software Defined Radio."’ Doctor of Technology Dissertation, *Royal Institute of Technology*, Sweden, 2000.

[4] http://www.radio-electronics.com/info/wireless/wimax/wimax.php, 15/02/07.

[5] Intel Technology Journal, Volume 8, Issue 3, August 20,2004.

[6] WiMAX General information about the standard 802.16, Application Note, Rohde & Schwarz GmbH, 06/2006.

[7] http://www.wimax.com/education/wimax/wimax_overview, 08/12/06.

[8] P. Javier, C. Sunghyun, "‘Link Adaptation strategy for IEEE 802.11 WLAN via Received Signal Strength Measurement."’ Technical Report, IEEE, 2003.

[9] P. Chevillat, J. Jelitto,A. Noll Barreto, H.L. Truong, "‘A Dynamic Link Adaptation Algorithm for IEEE 802.11a Wireless LANs."’ Technical Report, *IBM Research Division*, Switzerland, 2003.

[10] I. Budiarjo, H. Nikookar, L. P. Ligthart, Invited paper 2nd Annual CTIF Workshop, Aalborg, "‘Techniques for Cognitive Radio"’, International Research Centre for Telecommunications and Radar, *Delft University of Technology*, The Netherlands, 2007.

[11] B.A. Jati, C.F. Mayorga, Y. Wang, N.H. Mahmood,"Exploiting Channel Dynamics for Bit and Power Allocation", *Aalborg University*, Spring 2006.

[12] S. Srikathyayani, J.H. Reed, A. Peter, B. Robert, "‘A Software Radio Architecture for Reconfigurable Platforms"’. *IEEE Communications Magazine*, VOl. 38, No. 2, February 2000.

[13] Jeffery A. Wepman "‘Analog-to Digital Converters and Their Applications in radio Receivers. *IEEE Communications Magazine*, Vol. 33, May 1995.

[14] P. Burns, "Software Defined Radio for 3G", *Artech House*, London, 2002. ISBN: 1-58053-347-7

[15] Daniel D. Gajski, Robert H. Kuhn, "Guest Editors Introduction: New VLSI Tools", *IEEE Computer*, 1983.

[16] A. Jantsch, S. Kumar, A. Hemani,"A Metamodel for Studying Concepts in Electronic System Design", *IEEE Journal*, Vol 17, 2000.

[17] Anders Brodlos Olsen,"From Functionality to Silicon",8th Semester DSP Algoritms and Architectures Lecture Slides, MM1, 07/02/2006.

[18] J. Ayan, A.T. Toyserkani,"Sub-carrier based Adaptive Modulation in HIPERLAN/2 System", *IEEE Communications Society*, Vol 6, 2004.

[19] M. Lei, H. Harada, H. Wakana, P. Zhang,"An Adaptive Power Distribution Algorithm for Improving Spectral Efficiency in OFDM", *IEEE TRANSACTIONS ON BROADCASTING*, Vol 50, 2004.

[20] Altera Corporation, Stratix II device handbook, Volume 1, April 2006.

[21] XtremeDSP Development Kit-IV User Guide, NT107-0272, Issue 2, 22/07/2005.

[22] A. Huseyin, R. Sharath, “Noise Power and SNR Estimation for OFDM Based Wireless Communication Systems”, Department of Electrical Engineering, *University of South Florida*.

[23] Choong Il Yeh, Hyoung Soo Lim, Dong Seung Kwon (ETRI), "RSSI Measurements" *IEEE802.16 Broadband Wireless Access Working Group*, 2004-01-04.

[24] AD6645 Data Sheet, "Analog to Digital Converter", Analog Devices, 2003.

[25] Jeffrey H. Reed, "Software Radio: A Modern Approach to Radio Engineering", *Prentice Hall PTR*, ISBN:0-13-081158-0.

[26] Two Flows for Partial Reconfiguration: Module Based or Difference Based. Applicaion Note, Xilinx XAPP290 (v1.2), September 2004.

[27] A. Lagger, “Self-Reconfigurable Platform for cryptographic application”Master Thesis, School of Computer and Communication Sciences, *Swiss Federal Institute of Technology*, Lausanne, February 17, 2006.

[28] Xilinx Inc. Early Access Partial Reconfiguration User Guide For ISE 8.1.01i, v1.1, 06/03/2006.

[29] M. Hubner, J. Becker, "‘Exploiting Dynamic and Partial Reconfiguration for FPGAs-Toolflow, Architecture and System Integration"’, Tutorial,ITIV, *University of Karlsruhe*, Germany.

[30] G. Braeckman, G. V.d. Branden, A. Touhaffi, G. v. Dessel "‘Module Based Partial Reconfiguration: a quick tutorial"’, *Erasmus Highschool*, Brussel, July 2004.

[31] G. Mermoud, "‘A Module-Based Dynamic Partial Reconfiguration tutorial"’, Logic Systems Laboratory, *Ecole Polytechnique Federale de Lausanne*, November 2004.

[32] Xilinx Inc. Development System Reference Guide.

[33] Xilinx System Generator for DSP: User guide, Version 9.1.01, 19/03/2007.

[34] www.timelogic.com/technology_fpga.html, 07/06/07.

[35] John G. Proakis, Dimitris G.Manolakis: Digital Signal Processing Principles, Algorithms and Applications, Second Edition, ISBN:0.02-396815-X.

[36] Xilinx Inc. Virtex-4 Family Overview, v2.0, January 23, 2007.

[37] http://zone.ni.com/devzone/cda/tut/p/id/3007, 23/05/07.

[38] P. Sedcole, B. Blodget, Y. Becker, J. Anderson and P. Lysaght "‘Modular Dynamic reconfiguration in Virtex FPGAs"’, *IEE proc.-Comput. Digit. Tech.*, Vol 153, No.3, May 2006.

[39] Xilinx Inc. Virtex-4 Configuration Guide, January 12, 2007.

[40] http://www.everythingusb.com/usb2/faq.htm, 02/06/07.

[41] Theodore S. Rappaport, “Wireless Communications Principles and Practice”, Second Edition, 2005.

[42] Xilinx Inc. Virtex-4 Packaging and Pinout Specification, v3.0, January 24, 2007.

[43] N. P. Sedcole, "Reconfigurable Platform-Based Design in FPGAs for Video Image Processing", Master thesis, Department of Electric and Electronic Engineering, Imperial College of Science, Technology and Medicine, University of London, January 2006.