DEGREE PROJECT IN INFORMATION AND COMMUNICATION TECHNOLOGY,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, *SWEDEN 2017*

Crest Factor Reduction using High Level Synthesis

HASSAN MAHMOOD

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Crest Factor Reduction using High Level Synthesis

Author:
Hassan MAHMOOD

Supervisor:
Yuan YAO
Petter NILSSON
Examiner:
Zhonghai LU

This thesis work has been performed in fulfillment of the requirements for the Masters Programme in Embedded Systems

at the

School of Information and Communication Technology
KTH Royal Institute of Technology

October 2, 2017

Stockholm

Abstract

Crest Factor Reduction using High Level Synthesis

by Hassan MAHMOOD

Modern wireless mobile communication technology has made noticeable improvements from the technologies in the past but is still plagued by poor power efficiency of power amplifiers found in today's base stations. One of the factors that affect the power efficiency adversely comes from modern modulation techniques like orthogonal frequency division multiplexing which result in signals with high peak to average power ratio, also known as the crest factor. Crest factor reduction algorithms are used to solve this problem. However, the dominant method of hardware description for synthesis has been to start with writing register transfer level code which gives a very fixed implementation that may not be the optimal solution. This thesis project is focused on developing a peak cancellation crest factor reduction system, using a high-level language as the system design language, and synthesizing it using high-level synthesis. The aim is to find out if high-level synthesis design methodology can yield increased productivity and improved quality of results for such designs as compared to the design methodology that requires the system to be implemented at the register transfer level. Design space exploration is performed to find an optimal design with respect to area. Finally, a few parameters are presented to measure the performance of the system, which helps in tuning it. The results of design space exploration helped in choosing the best possible implementation out of four different configurations. The final implementation that resulted from high-level synthesis had an area comparable to the previous register transfer level implementation. It was also concluded that, for this design, the high-level synthesis design methodology increased productivity and decreased design time.

Keywords: Peak to average power ratio, Crest factor, Peak cancellation crest factor reduction, High level synthesis, Design space exploration, Object oriented programming, SystemC

Sammanfattning

Crest Factor Reduction using High Level Synthesis

by Hassan MAHMOOD

Användning av högnivåsyntes för reduktion av toppfaktor Det har gjorts noterbara framsteg inom modern trådlös kommunikationsteknik för mobiltelefoni, men tekniken plågas fortfarande av dålig energieffektivitet hos förstärkarna i dagens basstationer. En faktor som påverkar energieffektiviteten negativt är om signaler har en stor skillnad mellan maximal effekt och medeleffekt. Kvoten mellan maximal effekt och medeleffekt kallas för toppfaktor, och en egenskap hos moderna moduleringstekniker, såsom ortogonal frekvensdelningsmodulering, är att de har en hög toppfaktor. Algoritmer för reducering av toppfaktor kan lösa det problemet. Den dominerande metoden för design av hårdvara är att skriva kod i ett hårdvarubeskrivande språk med abstraktionsnivån Register Transfer Level och sedan använda verktyg för att syntetisera hårdvara från koden. Resultatet är en specifik implementation som inte nödvändigtvis är den optimala lösningen. Det här examensarbetet är inriktat på att utveckla ett system för reducering av toppfaktor, baserat på algoritmen Peak Cancellation, genom att skriva kod i ett högnivåspråk och använda verktyg för högnivåsyntes för att syntetisera designen. Syftet är att ta reda på om högnivåsyntes som designmetod kan ge ökad produktivitet och ökad kvalitet, för den här typen av design, jämfört med den klassiska designmetoden med abstraktionsnivån Register Transfer Level. Verktyget för högnivåsyntes användes för att på ett effektivt sätt undersöka olika designalternativ för att optimera kretsytan. I rapporten presenteras ett antal parametrar för att mäta prestandan hos systemet, vilket ger information som kan användas för finjustering. Resultatet av undersökningen av designalternativ gjorde det möjligt att välja den bästa implementationen bland fyra olika konfigurationer. Den slutgiltiga implementationen hade en kretsyta som är jämförbar med en tidigare design som implementerats med hårdvarubeskrivande språk med abstraktionsnivån Register Transfer Level. En annan slutsats är att, för den här designen, så gav designmetoden med högnivåsyntes ökad produktivitet och minskad designtid.

Keywords: Skillnad mellan maximal effekt och medeleffekt, Toppfaktor, Algoritmen peak cancellation, Högnivåsyntes, SystemC, Undersökningen av designalternativ, Objektorienterad programmering

Acknowledgements

I would like to thank my supervisor at Ericsson, Petter Nilson, for his guidance throughout the project and Pierre Rohdin for making sure that the work environment was adequate and comfortable. Special thanks to the support team at Cadence, including Nils Luetke-Steinhorst, Sarmad Dahir, and Dave Apte, for their indispensable knowledge and insight regarding the HLS tool used for the project. I am grateful to my examiner, Zhonghai Lu, and my supervisor at KTH, Yuan Yao, for their suggestions regarding the project report.

I would also like to express my gratitude towards Abhineet Tomar for his feedback and Abu Darda, Aditya Gahlaut, Prashant Sharma, Imran Habib, and Zahin Azher for being there throughout the project and making it worthwhile.

Finally, I would like to thank my family for their love and support throughout the Master's program.

Contents

List of Figures

List of Tables

List of Acronyms

ACK	Acknowledgement
ACLR	Adjacent Channel Leakage Ratio
ASIC	Application Specific Integrated Circuits
BER	Bit Error Rate
CMR	Code Motion Rule
CCDF	Complementary Cumulative Distribution Fucntion
CORDIC	COordinate Rotation DIgital Computer
CF	Crest Factor
CFR	Crest Factor Reduction
DSE	Design Space Exploration
DUT	Design Under Test
ECO	Engineering Change Order
EVM	Error Vector Magnitude
GUI	Graphical User Interface
HLS	High Level Synthesis
IC	Integrated Circuit
LCD	Loop Carried Dependency
MSR	Multi Standard Radio
NOP	No Operation
OOP	Object Oriented Programming
OFDM	Orthogonal Frequency Division Multiplexing
PC-CFR	Peak Cancellation Crest Factor Reduction
PC	Peak Cancellation Crest Factor Reduction
PF	Peak Filtering
PAPR	Peak to Average Power Ratio
PW	Peak Windowing
PCU	Peak Control Unit
RAM	Random Access Memory
RTL	Register Transfer Level
SOC	System On Chip
VHSIC	Very High Speed Integrated Circuits
VHDL	VHSIC Hardware Discription Language

Chapter 1

Introduction

Modern wireless mobile communication technology has made noticeable improvements from the technologies in the past. We are already in the fourth generation and the fifth generation is expected to be released to the market in the near future. However, the power efficiency in today's base stations is a factor that needs to be addressed. A major portion of the Capital Expenditure and the Operating Expenditure is incurred in the radio shelf which contains the power amplifiers [4]. One of the factors that affect the power efficiency adversely comes from modern modulation techniques like Orthogonal Frequency Division Multiplexing (OFDM) which result in signals with high peak to average power ratio(PAPR), also known as the Crest Factor (CF). There have been various scientific researches on this problem and a multitude of solutions have been presented to solve the issue. However, the dominant method of hardware description has been Register Transfer Level (RTL) code which gives a very hardware specific implementation. High-level synthesis (HLS) can be explored to develop these algorithms. HLS allows for quick design turnaround time and thus allows for design space exploration (DSE). If this is not done, we cannot possibly know if we have the best possible configuration of the hardware according to the requirement of the application. Performing design space exploration by writing the code in RTL can be quite time-consuming. It would be interesting to see if HLS can yield a comparable design to existing RTL implementations with less design time. This study will briefly go over the different Crest Factor Reduction (CFR) techniques, compare their pros and cons, and then will go on to develop a peak cancellation Crest Factor Reduction (PC-CFR) algorithm and implement it in a high-level language to compare it to some existing RTL implementations. This will provide some insight into the use of HLS for the development of similar algorithms. If the results show this to be a viable option, time to market can be reduced considerably. However, the results can not be applied as a general rule for all Application Specific Integrated Circuits (ASIC) designs as a single data point is not enough to justify a trend.

1.1 Research Problem

The study is important to explore HLS to decrease the design time and compare the resulting implementation with previous ones. The comparisons will be based on the hardware area, the percentage of area occupied by memory, and productivity. The Complementary Cumulative Distribution Function (CCDF) will be used to visualize how much time the signal power stays at or above a given power level [5]. Adjacent Channel Leakage ratio (ACLR) and Error Vector magnitude (EVM) are also helpful in characterizing the CFR implementation. The research question is whether the study will yield a CFR implementation that ranks higher on the comparison parameters discussed above.

1.2 Goals and Objectives

The goal of this thesis project is to investigate if the HLS tools today can be used for the design of an algorithm like CFR. The thesis project aims to design and implement a CFR algorithm using a high level synthesis, perform design space exploration, and compare the area and design time with a previous solution implented using the RTL design methodology.

1.3 Purpose

The purpose of this thesis project is to decrease the design time of ASIC and FPGA in the industry so that more focus can be put on performing design space exploration and getting the best possible hardware configuration according to the application specification.

1.4 Limitations and delimitations

The resulting implementation from this thesis is expected to be compared with a previously designed implementation. However, the design in this thesis is supposed to be completely independent of the previous design and will have different architectural and algorithmic properties. The limitation on the comparison is that the two designs will be compared on some general functionalities and features. Moreover, the area comparisons will be done keeping in mind the above-mentioned differences and will only be used to see if the new methodology can yield a system that is roughly of the same order in size. Another limitation on this project is that the results can not be generalized for all possible designs since different systems have different characteristics and architectural requirements, e.g. having multiple clock domains. This result will be taken as a single data point in the study of HLS design methodology.

A delimitation is set for this project is on the logic synthesis section. The focus of this project is solely on the HLS part. The RTL synthesis flow and results could certainly be optimized but the flow is used, without optimization, as provided and only for the purpose of comparing the results of the thesis.

1.5 Thesis Organization

Chapter 1 describes the problem and its context. Chapter 2 provides the background necessary to understand the problem and the specific knowledge that the reader will need to understand the rest of this thesis. Following this Chapter 3 describes the goals, metrics, and solution proposed in this thesis project. The solution is analyzed and evaluated in Chapter 4. Finally, Chapter 5 offers some conclusions and suggests future work.

Chapter 2

Background

Chapter 2 sheds some light on the background of CFR and HLS. It gives an insight into the cause of the problem and the types of techniques that can be used to mitigate it. It goes on to deliberate the reason for choosing PC-CFR in this thesis project by comparing some of the CFR techniques and then briefly discusses the main components of a PC-CFR implementation. The problems of this technique are also presented. The chapter also has a brief note on the need for HLS today in the design process of System On Chip (SOC). It discusses the common stages of HLS tools, advantages, and disadvantages of HLS. Finally, the chapter lists some related work done on this topic.

2.1 Cause of High Crest Factor and role of CFR

With the evolution of wireless mobile communication, OFDM has been the choice for many wired and wireless system standards like WiFi, WiMAX, LTE, PLC and so on. Single carrier modulation schemes were susceptible to multi-path fading. The solution was provided in the multi carrier modulation scheme, OFDM, by sending the data over multiple narrow-band carriers at a slower bit rate. Slowing down the bit rate over individual narrow bands solved the multi-path fading problem [6] and since multiple carriers were sent simultaneously, the overall bit rate did not deteriorate. However, this gave rise to a signal with non-constant envelope. Single carrier modulation schemes produced constant envelope signals which resulted in a constant and low peak power to average power ratio, also known as the crest factor. With the new multi carrier schemes like OFDM, this is no longer the case and the resulting CF is quite high [7]. A visual representation of CF of various signals is shown in figure 2.1.

This causes a problem while setting the operating point for the power amplifier before the transmitter. Since the power amplifier needs to be operated in the linear region to make sure there is no in-band and out-of-band emission which result in increased Bit Error Rate (BER) and Adjacent Channel Leakage [7], the operating point needs to be set with a safety margin known as a back-off so that the peak signal value does not cross over to the saturation region. This is what introduces the power inefficiency. There are two main solutions that can be employed to counter this problem. The first one is to increase the linear region of the power amplifier. Digital pre-distortion can be used to achieve linearity [4]. The second case is to reduce the maximum peak value before passing the signal through the power amplifier. This paper discusses the second solution. These methods are collectively known as CFR schemes. These achieve the PAPR or CFR by compensating with lower bit rate, higher transmit power, higher BER and so on [8]. A comparative view of the operating point for the power amplifier before and after applying the CFR technique is shown in Figure 2.2.

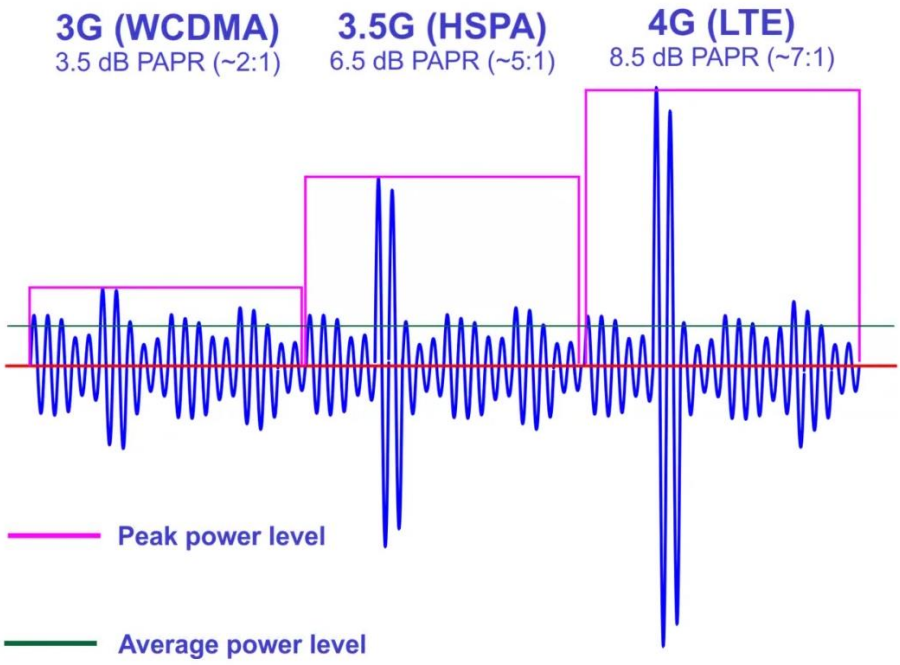


FIGURE 2.1: A comparison between PAPR for different protocols (source: [1])

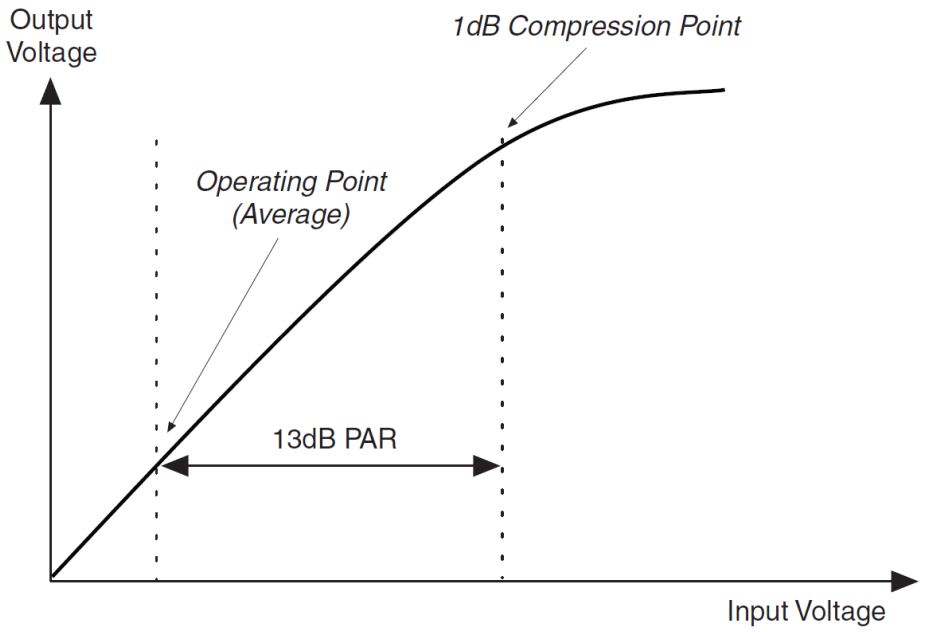
2.2 Types of CFR and their comparison

The CFR techniques can be categorized in three different ways [9].

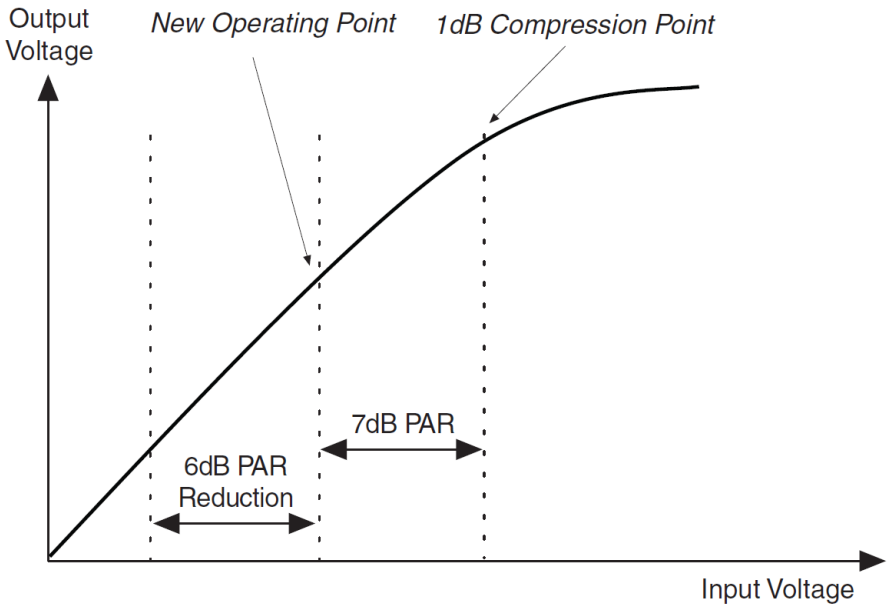
1. Deterministic and Probabilistic techniques
2. Distortion based and scrambling techniques
3. Clipping, Coding, Probabilistic and Pre-distortion techniques

2.2.1 Deterministic and Probabilistic techniques

The first method of categorization divides the techniques between deterministic and probabilistic approaches [9]. Deterministic algorithms consist of methods like time domain clipping and frequency domain coding. Clipping algorithms are the most simplistic, however, they introduce distortion in the signal. Frequency domain coding techniques deal with the search for code mappings of symbols in signals like OFDM, that yield the lowest PAPR or Crest Factor. However, this method becomes inefficient for a large number of carriers as the search algorithm becomes more complex and requires more hardware [8]. Probabilistic methods aim to reduce the PAPR while trying to reduce the distortion added during the process of reducing the CF. These methods can be further divided into blind and non-blind techniques in time and frequency domain. Blind techniques do not require the receiver to have any knowledge of the transmitted signal and thus are not very efficient in terms of BER since they do not have information regarding any change at the transmitter. Non-blind techniques, on the other hand, require some meta-data to be sent along with the original signal, thereby reducing the effective information rate. An example of a probabilistic blind technique in the time domain is peak clipping



(A) Amplifier Input/Output voltage characteristics before applying CFR



(B) Amplifier Input/Output voltage characteristics after applying CFR

FIGURE 2.2: Power amplifier operating point before and after applying CFR (source: [2])

and filtering. Filtering aims to reduce adjacent channel leakage due to peak clipping. However, the filtering itself can add to peak regrowth as well as add to the hardware complexity by increasing the number of multipliers. Companding techniques that compress the large peaks fall under the non-blind techniques. The receiver requires the decoding information to decompress the original signal.

Frequency domain techniques are mainly based on the principle of detecting and minimizing the correlation between the input signals as high correlation leads to high PAPR. This can be achieved by altering the phase or power of the input signal. Selective mapping and partial transmit sequencing [8] are examples of blind frequency based probabilistic algorithms.

2.2.2 Distortion based and scrambling techniques

Another way to categorize CFR algorithms is to divide them between Distortion and Scrambling techniques [9]. Scrambling techniques are based on searching for the best code that minimizes the PAPR. These techniques may also require some meta information to be sent along with the original signal. Furthermore, the complexity of the search for the best code increases with the increase in the number of channels of the signal. The search algorithm and the metadata cause the system to be designed with decreased effective throughput. Distortion techniques are much simpler to implement and do not decrease the data rate. However, as the name suggests, these methods introduce distortion in the signal by directly manipulating the signal peaks to keep the signal under a specified threshold. Examples of distortion based techniques are peak windowing, peak cancellation, companding etc. Examples of scrambling techniques are block coding and selective mapping etc.

The discussion about the third way of categorization is redundant for this paper and can be easily inferred from the previous two categories. Figure 2.3 shows a tree of CFR classes.

2.3 Delibration for the choice of CFR

The base stations of this generation are capable of transmitting different radio access technologies from a single unit and are known as Multi Standard Radio (MSR) base stations. There is a limited number of CFR techniques that are useful in an MSR application and they all fall under the category of distortion based techniques. According to the comparison in [8] receiver, they also do not decrease the data rate. In figure 2.3, these fall under the branch of deterministic time domain CFR algorithms. Potential candidates in this class are peak filtering, peak windowing, and peak cancellation.

2.3.1 Peak Filtering

Peak filtering (PF), also called clipping and Noise shaping, implements a generic and long filter for all the sub-carriers. An error signal is extracted and passed through the filter. The output is subtracted from the original signal. PF is known to be computationally complex because of the convolution operation between the error signal and the filter. This has the disadvantage of the need for a large number of resources for the convolution operation [4].

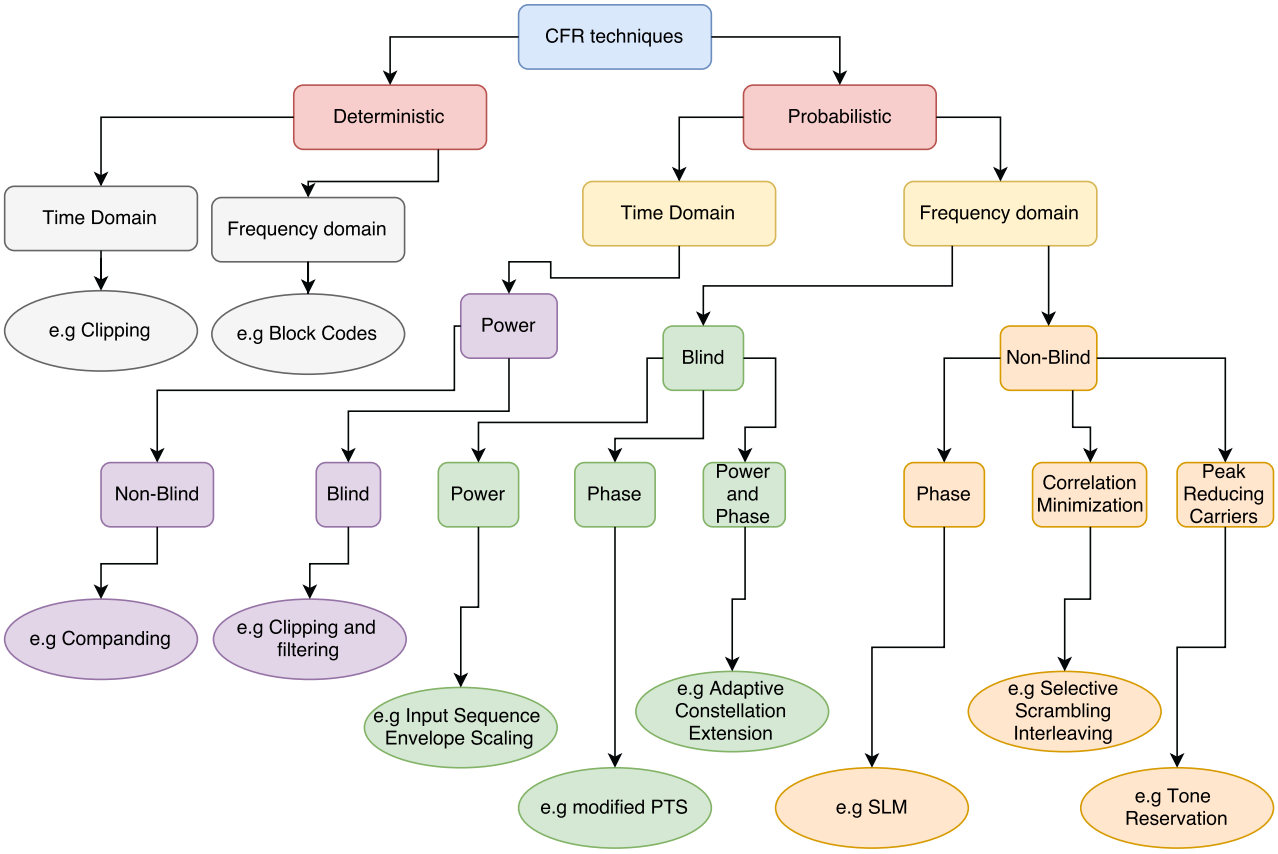


FIGURE 2.3: Categorization of Crest Factor Reduction Schemes

2.3.2 Peak Windowing

Peak windowing (PW) applies a window function to the original signal to limit it to a defined threshold. However, this technique has been shown to over-compensate in case of close peaks and introduces high EVM values [10] [11].

2.3.3 Peak Cancellation

Peak Cancellation (PC) is the last technique discussed in this paper. PC is very similar to PF with a few changes. PC removes the need for complex convolution operations by defining a peak differently from PF. PC defines a window and searches for the maximum value above the threshold in that window. That search window starts only if a value above the threshold is detected. An error impulse is generated from this operation and then sent through a filter with predefined coefficients. Since the convolution is with a single impulse, it greatly simplifies the filter design which can be implemented with a single multiplier [4] [12] [11] [13].

In comparison, PC has been shown to perform the same or, in some cases, better than PF and PW with respect to lowering PAPR and reducing EVM. On the other hand, PW has the lowest implementation complexity followed by PC. But PW does not perform well with respect to low EVM values and thus, PC is still preferred over it [4].

There are a number of features that can be added to the PC CFR technique to deal with some inherent shortcomings that are discussed in section 2.4. A block diagram of a simple peak cancellation algorithm can be seen in Figure 2.4.

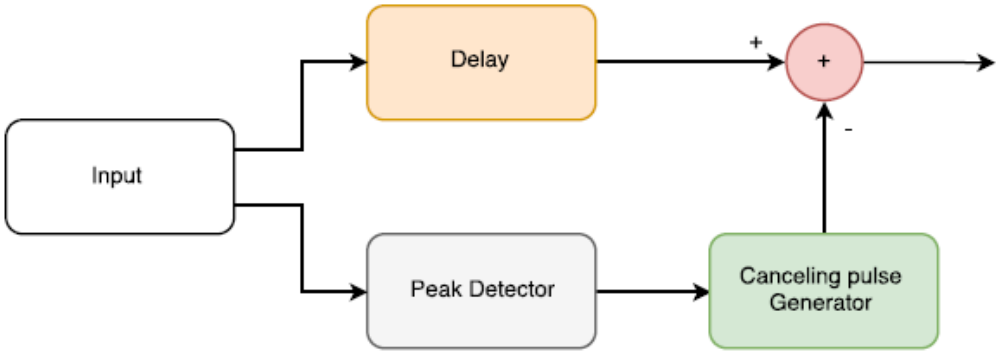


FIGURE 2.4: Basic Peak Cancellation Flow

A threshold is initially defined according to the desired output PAPR. The complex input signal is first divided into magnitude and phase components. A peak detector module detects the values above the specified threshold in a region. The highest value above the threshold in a range of samples is the recognized peak. An impulse with the magnitude equal to the difference between the threshold and the peak magnitude is passed through a filter with predefined coefficients. The output is a cancellation pulse. This cancellation pulse is converted back to the cartesian form and subtracted from the original signal. Delays are added in the path of the original signal to compensate for the delay in the cancellation pulse computation path. This is important to synchronize the two signals so that the correction is applied to the correct signal samples. A more behavioral implementation specific diagram of the system can be viewed in Figure 2.5.

2.4 Problems with Peak Cancellation CFR Technique

2.4.1 EVM and ACLR

Peak cancellation crest factor reduction reduces the PAPR by directly changing the peaks in the signal detected by setting a threshold. This leads to both in-band and out-of-band distortion. These can be measured by two parameter; EVM and ACLR respectively. EVM uses the information from the standard constellation diagram for the digital modulation technique which shows the expected place for the received symbols on the I/Q plane. The displacement of the actual received symbol from the expected position on the I/Q plane gives the EVM for that symbol as shown in 2.6. The EVM is calculated according to equation 2.1:

$$EVM = 10\,log_{10}(P_{error}/P_{ref}) \tag{2.1}$$

where P_{error} is the sum of the error vector powers for the received symbols and P_{ref} is the sum of expected symbol powers. ACLR is the measure of the out of band distortion i.e. it provides information about the power leaked to adjacent channels. This leakage causes the transmission to be inefficient as well as introduces noise in the adjacent channels. The ACLR is calculated using equation 2.2:

$$ACLR = Power\,in\,Adjacent\,Channel/\,Power\,in\,Main\,Channel \tag{2.2}$$

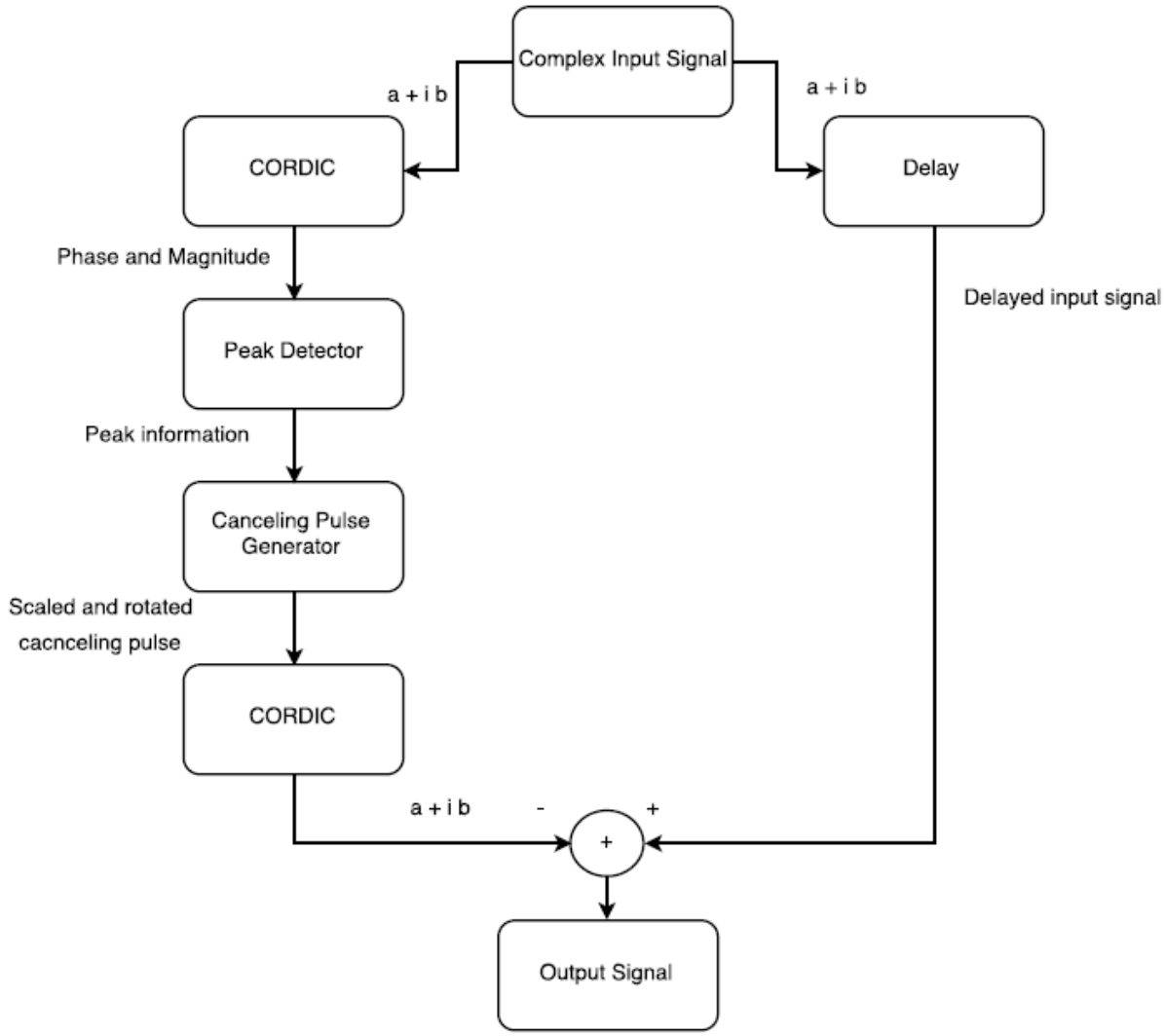


FIGURE 2.5: CFR Behavioral block diagram

The in-band and out of band distortions are dependent on the cancellation pulse. The smaller the cancellation pulse is, the better in terms of in band noise. However, out of band distortion increases if the cancellation pulse does not decay smoothly, i.e. discontinuities in the cancellation pulse where the pulse goes from a high value to zero instantly will introduce adverse effects on the out of band performance. This requires a longer cancellation pulse which is in contrast to the observation for reducing in band distortion. Thus, a compromise has to be made to get permissible in-band and out of band performance in compliance with the set standards. Another observation about the cancellation pulse is to end it at a zero crossing [13]. This basically means that the filtering signal should be terminated at a point where its amplitude is zero. This is a direct consequence of the fact that discontinuities increase out of band emissions and truncating the filter anywhere in between zero crossings would introduce discontinuities. Figure 2.7 elaborates on this concept. The red colored samples are zero crossings and are better options for the choice of truncation points. If the signal is truncated at or close to the blue colored samples, out of band emissions are expected to increase. It should be noted that the x-y axis

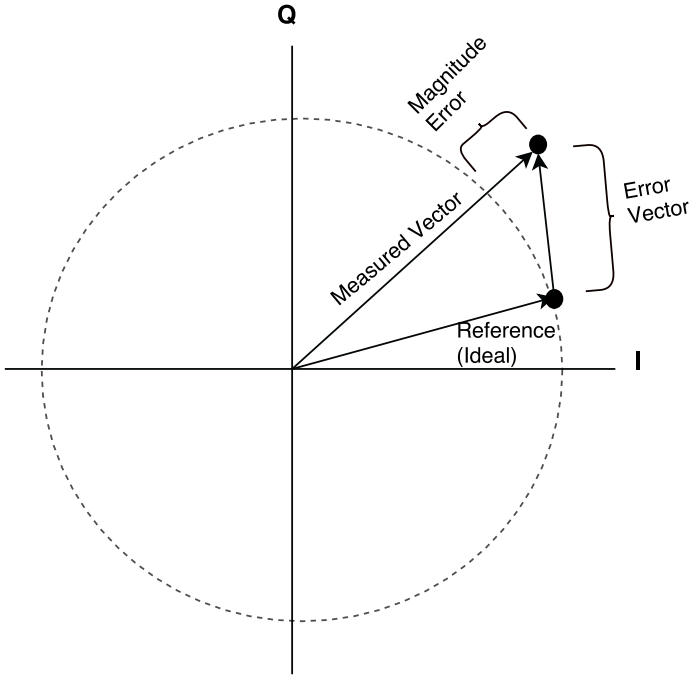


FIGURE 2.6: Graphical representation of error vector magnitude (source: [3])

numbers are not the actual numbers corresponding the filter coefficients.

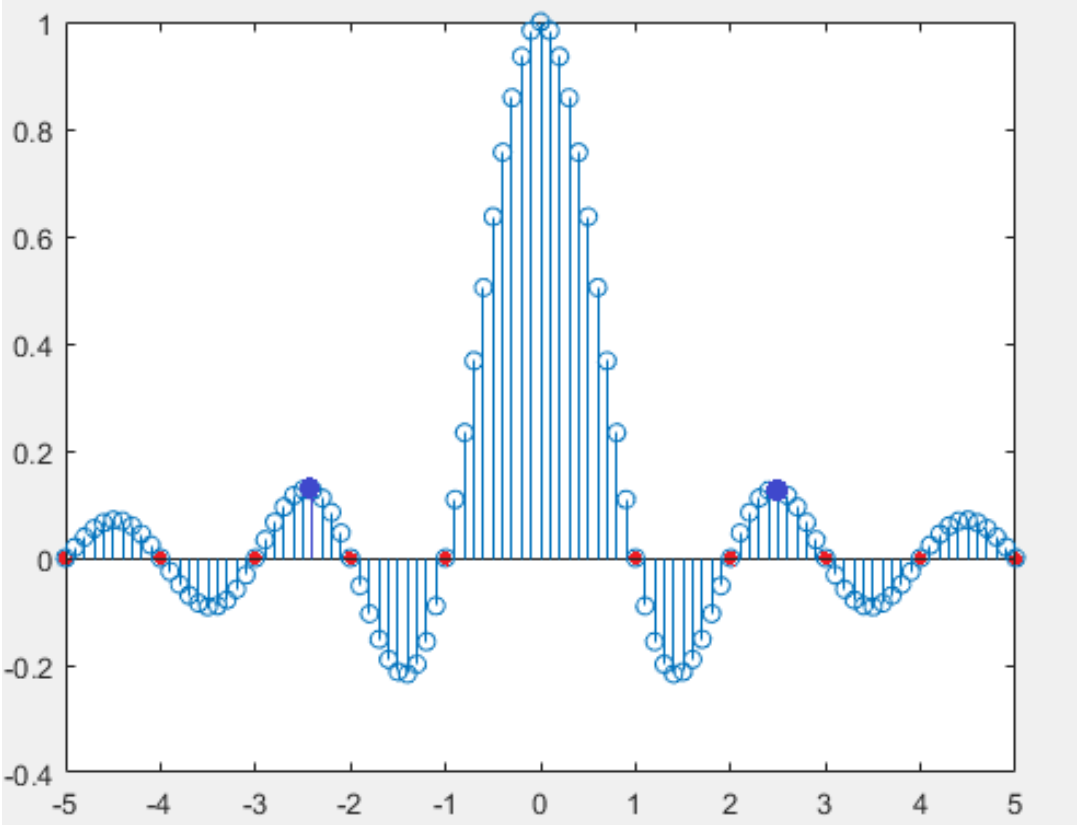


FIGURE 2.7: Truncation points for cancellation signal

2.4.2 Peak Leaks

Another problem that plagues this method is that the number of peaks canceled is dependent on the number of canceling pulse generators available. For the sake of simplicity, a canceling pulse generator is going to be called a resource from hereon. A resource can only work on generating a pulse for one peak at a time. If all the resources in a system are busy, newly detected peaks will be ignored or leaked. A number of smart designs can be incorporated to deal with this problem which includes time division multiplexing, priority based scheduling, preemption etc. A more detailed discussion of the solution to this problem is given in chapter 3, Solutions and Methods 3.

2.4.3 Peak Regrowth

Peak regrowth is a phenomenon where new peaks are introduced into the original signal after peak cancellation has been performed. This occurs due to the constructive interference of multiple cancellation pulses generated for different peaks. This becomes pronounced when densely populated peaks are detected as that would overlap the major lobes of the cancellation pulses. This is why the maximum amplitude sample in a window is selected as a peak. By changing the width of the window, peak regrowth can be reduced.

2.5 Use of CCDF Graphs in PC-CFR Performance Evaluation

When measuring the PAPR of a signal, knowing the percentage of time the signal remains at or above a certain power level can be very useful as it gives an insight into the ratio of the maximum power level and the distribution of the power content in the signal. The cumulative distributive function (CDF) of the power of a signal, where the x-axis is power level and the y-axis is the probability when evaluated at any point X, gives the probability of having power equal to or less than X. By subtracting the CDF function from one, the required data can be acquired, which is simply a better representation of the same information. This new function is known as the complementary cumulative distribution function (CCDF). Figure 2.8 shows a CDF and the corresponding CCDF graph. In the figure, the CCDF graph shows that when the function is evaluated at x = -5, it is at or above -5 between 80 to 90 percent of the time. Similarly, the function takes up the value of 0 or above 30 to 40 percent of the time.

$$CCDF = 1 - CDF \tag{2.3}$$

When measuring PAPR, the x-axis of a CCDF graph represents PAPR values whereas the y-axis shows the relative frequency in powers of ten(logarithmic scale). A value of 0.001 represents negligible value.

2.6 A Brief Introduction on HLS

HLS is an Integrated Circuit(IC) design technology that is receiving popularity in the recent years due to its potential in providing large productivity gains. This method has its benefits and limitations which are necessary to understand before adopting it in the design flow. This section briefly goes over the general methodology that HLS tools follow, the advantages and the limitations.

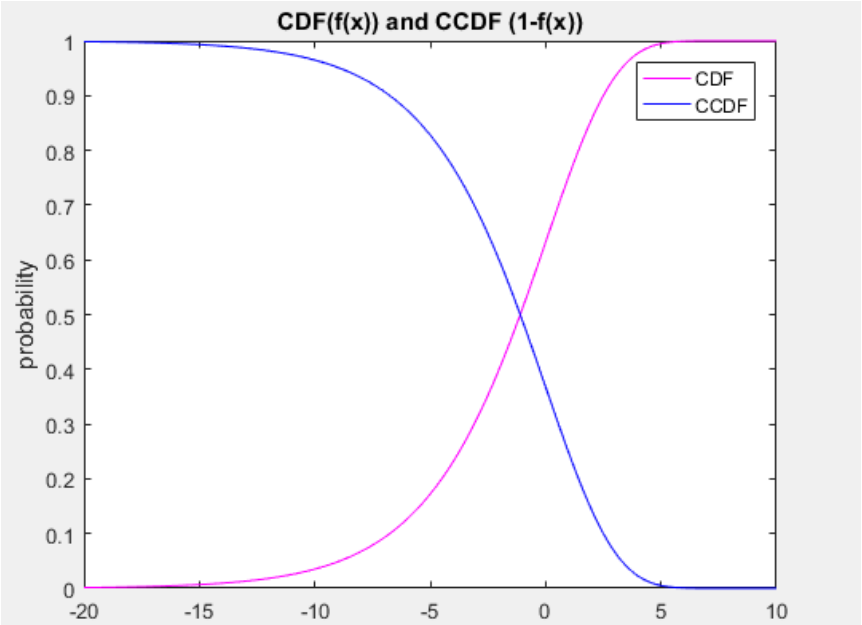


FIGURE 2.8: Example of a CDF and a CCDF graph

The complexity of SOC today has risen to a point where there is a big productivity gap between it and the production per designer and per year and leads to an exponential increase in the need for manpower which can not possibly be matched [14]. This trend is expected to continue according to Moore's law [15]. The solution to the problem is Electronic System Level Design which includes hardware/software co-design as well as HLS. Hardware/Software HW/SW co-design allows the software team to start software development early without having to wait for the hardware platform. This is done using virtual platforms that are bit and cycle accurate system level models of the hardware. By introducing HLS, behavioral models can be translated to RTL. HLS requires the developers to write behavioral code in a high-level language like C++/SystemC or Matlab etc. as opposed to RTL code in a hardware description language like Very High Speed Integrated Circuits (VHSIC) Hardware Description Language (VHDL) or Verilog etc. In the latter approach, the behavior of the algorithm has to be verified first in a high-level language and then RTL code has to be manually translated from the behavioral implementation. This introduces possibilities of human errors. Secondly, performing exhaustive searches for the best possible architecture, i.e. pipelining after analyzing the dataflow can be very time consuming and would delay the development process. High-level synthesis removes the need for manual translation and uses computer aided design methodology to perform DSE. Following are the basic steps for an HLS routine:

- Algorithm Optimization
- Control and dataflow analysis
- Resource allocation
- Scheduling
- Binding

- RTL generation

The HLS tool performs optimization of the algorithm. Then the control and dataflow is analyzed to find out the different operations that need to be performed. Hardware resources are allocated according to these and then scheduled in different clock cycles based on a scheduling algorithm. The operations are then bound to the hardware resources in specific clock cycles, the variables that carry values across boundaries are bound to storage units, and the transfers between components are bound to connectivity units. Finally, the RTL code is generated [16]. The advantages of this methodology are listed as follows:

- Reduced production time by concentrating on the design of system level abstraction without concerning with architectural level details. The HLS tool is responsible to investigate the architectural level implementation depending on the constraints provided by the user and generating an RTL translation.
- Reduced cost of design as compared to the trend of Moore's law [14]. This is a collective effect of the continuous advances in IC/system implementation tools, of which HLS is among the later contributions.
- Dataflow analysis which leads to possibilities of a pipelined architecture and increases utilization of the hardware.
- Loop unrolling for parallel operation after checking data dependencies across iterations.
- Possibility of a common testbench for behavioral code, RTL code and gate level netlist.
- Generation of control path inferred through the Control and DataFlow Graph.
- Integrated development environment developed by HLS vendors today that provides good debugging environment.

However, HLS does have some limitations as well. According to Samsung's view point, the technology is not mature enough yet to be introduced into the industry [14]. Following are a few of the problems that currently plague the technology:

- The high-level language used as an input in High-level synthesis tools characterize the tool itself. Mostly these are C/C++ variants. SystemC has hardware specific data structures and the ability to express timing information. On the other hand, C/C++ are better at describing the behavior of the algorithm at a higher abstraction level. So, trade-offs have to be made when selecting the language.
- Writing high-level code for HLS requires the coder to be familiar with both high-level languages and hardware architecture or RTL coding since the result of the synthesis is still dependent on the coding style. This poses a question about whether HDL engineers or software engineers should be given the task.
- The synthesizable subset does not include all the high-level constructs. However, HLS tools have come a long way in this aspect.

2.7 Related Work

This thesis project is of interest to Ericsson which is investigating the usage of HLS for development of algorithms like CFR. This project is built on a previous project done at Ericsson by Matteo Bernini [12], which proposes a cascaded implementation of a Peak Cancellation Crest Factor Reduction scheme. The paper also introduces a time division multiplexed version of the resource that runs at a faster clock as compared to the clip stages. This emulates multiple resources without having to increase the hardware. The paper also gives an estimation of the number of clip stages and resources to be used to obtain an optimal PAPR reduction based on a reference golden Matlab model. The same golden model is being used in the current project for bit level behavioral verification. Finally, the paper has suggested some interesting points as future work which can be introduced in the current project. The major difference between the two projects is the usage of HLS instead of RTL language to investigate productivity gains.

In the Xilinx application note 1033 (XAPP1033) [4], a Peak Cancellation Crest Factor Reduction is introduced with one peak detector and four cancellation pulse generators. It also compares the peak cancellation and the peak windowing function. This note introduces a clever way of defining a peak. Instead of marking every sample above the threshold, peaks are defined as the largest value above the threshold in a window. This mitigates the problem of peak regrowth due to the constructive interference of the canceling pulses in the other method.

In [13], the authors present a FPGA implementation of a simple peak cancellation scheme. This paper provides very good intuition in the generation of the canceling pulse and its effect on the in-band and out-of-band emissions. The paper also compares two conventional hardware implementation schemes and then proposes a novel approach. The first scheme is dependent on the number of available resources and can leak peaks. The second implementation passes the peak errors as impulse trains through a Finite Impulse Response filter which increases the hardware area. The novel approach takes the first approach and solves the issue of peak leaks by stopping short the generation of cancellation pulses and attending to the newly detected peak. This approach has the issue of introducing discontinuities. However, these can be taken care of by introducing averaging filters.

In [9], the author has discussed the issue of high PAPR in OFDM signals and formulated a number of new CFR algorithms. The paper has a section on CFR categorization that discusses the different classes very thoroughly.

[11] is a user guide for an IP core provided by Lattice Semiconductor. This document provides good deliberation on the choice between peak cancellation and peak windowing CFR algorithms. It also provides a guide on filter selection for generating the canceling pulse.

[2] is an application note that describes the solution used by Altera for solving the crest factor problem. The document discusses adaptive baseband, intermediate frequency clipping and filtering, and peak windowing.

In [8], the author describes some of the important PAPR reduction techniques. The paper also provides a criterion for selecting a CFR algorithm for a certain application.

In [6], the author analyzes the feasibility of OFDM to mitigate the multipath fading by comparing a single carrier technique to multi-carrier OFDM technique.

Chapter 3

Implementation of a PC-CFR System

Chapter 3 discusses the design methodology and the choice of the high-level language and HLS tool, as well as the implementation of the PC-CFR algorithm. A bottom up approach is used in the description of the modules, i.e. the constituent parts are listed first before piecing together entire modules. The purpose of each component is elaborated and control flow diagrams are used where necessary. The chapter starts with a vanilla version of the system and then goes on to describe a more advanced implementation. Further on, some architectural decisions are explained which add to the picture of the advanced implementation. At this point, the reader is expected to have a fairly good view of the finally implemented system. This hierarchical breakdown of the system was deemed necessary to simplify the explanation of the system. The chapter also discusses the HLS directives used in the project, testbench restrictions, the logic synthesis process and the future work that can be done to improve the system.

3.1 Design Methodology

This section goes over the deliberation for the choice of high-level language and HLS tool. It also goes over the design flow of the tool and the contents of an important file in each project that controls the outcome of the different synthesis levels. This file has some components that are vital in explaining some major features provided by the HLS tool.

3.1.1 Choice of tool and Language

Stratus High-Level Synthesis [17] by Cadence was chosen for this thesis project to develop and synthesize the PC-CFR algorithm. This tool takes a high-level language like SystemC, C++ or ANSI C as the input. A comparison between these languages and explanation of why SystemC is best suited as the system design language is presented as follows:

- All three languages allow informal high-level description of algorithms and thus are algorithm languages. This is needed for writing code at the behavioral abstraction level which is of one the major purposes of HLS.
- SystemC and C++ are Object Oriented Programming(OOP) languages and allow for a much more modular design that reduces complexity and allows reusability. This is necessary for hardware design to define modules.

- SystemC allows for having exactly the number of bits required for a data type whereas the C and C++ do not. The latter languages have data types with fixed bit widths. Having the ability to specify the exact bit width is important in hardware design as it is undesirable to have extra bits when they are not needed in the design.
- SystemC is the only language among the three candidate languages that allows for concurrent programming. The introduction of concurrent threads and methods in the language is vital to emulate hardware behavior.
- SystemC provides a two-tier timing structure to emulate delay between events as well as hardware delay (delta delay [18]) and allows for cycle accurate implementations of the design. C and C++ do not have timing structures.

Table 3.1 summarizes the above mentioned points.

Features	SystemC	C++	ANSI C
Algorithm language	Yes	Yes	Yes
Object oriented Language	Yes	Yes	No
Bit-exact data types	Yes	No	No
Concurrent simulation	Yes	No	No
Timing structure	Yes	No	No

TABLE 3.1: Comparison between High Level Languages

3.1.2 Design Flow

The design flow describes the steps taken throughout the process, starting with preparing input code for the HLS tool and ending with the logic synthesis process that returns a design implementation in the form of logic gates. Following are the design steps:

1. Write SystemC behavioral code and testbench for design verification.
2. Run the behavioral simulation and verify the functionality of the design.
3. If functionally correct, go to step four, else go to step one.
4. Run HLS and reuse testbench for an RTL simulation to verify protocol accuracy i.e. is the system cycle accurate as well.
5. If the RTL netlist is protocol accurate and meets constraints, go to step six, else go to step one for changing the algorithm or go to step 4 four changing the HLS directives.

6. Run Logic Synthesis and analyze the output. At this point, the logic synthesis result is already expected to meet timing.

In Stratus HLS, each project uses a Tcl [19] file that holds important information for the tool including:

- **Path to technology library:** This library holds the basic information regarding basic cell libraries to be used in the design.
- **Path to memory library:** This library includes the memories defined in the design.
- **HLS directives:** These are used by the tool to get an implementation that meets different constraints and characteristics.
- **System modules:** These modules are not synthesized by the HLS tool. For example, the testbench might not have to be synthesized.
- **HLS modules:** These modules are synthesized by the HLS tool. For example, the design under test (DUT).
- **Simulation configurations:** These configurations allow the user to simulate the code using different settings. For example, running a behavioral simulation with Transaction Level Modeling (TLM) [20] interfaces when only the functionality has to be checked. This would require the result to be bit accurate and not cycle accurate. This method allows for fast simulations. These can also allow SystemC I/O style communication for a cycle accurate simulation. These configurations can also be used to run HLS for different architectures, for example, a pipelined or an unpipelined design etc and then simulating them. These even allow simulations with mixed RTL and behavioral modules. When running an RTL simulation, the tool automatically runs HLS first, if the RTL implementation does not exist.
- **Logic Synthesis configurations:** These configurations are used to allow different settings for logic synthesis.

The simulation configurations are very important for early debugging of the code and allow DSE to become a lot simpler by having multiple implementations. Simulation and logic synthesis configurations have a big part in making the HLS process productive and flexible.

3.2 System

The basic PC-CFR block consists of two modules, the Design Under Test(DUT) and the testbench for verification. Following is a breakdown of these sub-blocks.

3.2.1 Testbench

The testbench consists of two clocked threads, a source, and a sink. There is no communication between the source and the sink. The source starts with sending the reset signal to the DUT and then toggles it after a couple of clock cycles. Next, the source reads coefficients from a file and sends them to the DUT which are needed for the coefficient memory in the resource. These coefficients are the taps of a filter

that were previously designed at Ericsson. Once all the coefficients have been sent, the source starts reading a file that has a sample of an input signal with two carriers which needs to undergo peak cancellation. These samples are sent to the DUT in a throughput defined by a clock to sample ratio. The value of the clock to sample ratio is based on an architectural decision which will be discussed in 3.4.

The sink reads the input port for the corrected signal each cycle and checks whether it is a valid sample. If the sample is valid, it is written to a response file which is compared with a golden file. The golden file is generated from a bit accurate golden model provided by Ericsson. This golden model was designed by a previous master thesis student. A few changes had to be made to make it consistent with the new SystemC model.

3.2.2 DUT

In the vanilla version of the system, the DUT consists of one clip stage and one resource that is capable of generating a canceling pulse for one peak at a time. The input to the DUT is sent to the clip stage. The clip stage processes the input and sends output to the resource. The resource generates a canceling pulse and sends it to the clip stage. The clip stage cancels the peak and sends the corrected signal to the testbench.

3.2.2.1 Clip Stage

The clip stage consists of two threads; the peak detection thread and the peak cancellation thread.

Peak Detection Thread

The peak detection thread is responsible for converting the Cartesian input to the polar form, detecting peaks in the original signal and sending the peak information to the resource. It is also responsible for delaying the original signal. The reason for this delay is that there is some latency between the detection of the peak and arrival of the generated canceling pulse samples at the clip stage. The peak needs to be centered with the canceling pulse.

Peak Definition

It is an observation that dealing with each sample above the threshold as separate peaks can cause the system to deteriorate performance and cause peak regrowth. Normally, each major sample above the threshold is surrounded by more samples that go over the limit. Thus, it is better to treat that group as a single peak centered at the maximum value. This also allows for decreasing the number of resources that are required to generate cancellation pulses simultaneously and, thereby, decreasing the area and power of the hardware. The peak is defined as:

> *"The maximum number with magnitude above the threshold in a window, where the window starts when the first number with magnitude above the threshold is detected and ends after a specified number of samples"*

The length of the window is a parameter provided to the system. This parameter helps in calibrating the system. If the input signal has a high frequency of detected peaks and the window length is too small, the system will over-compensate while

performing peak cancellation. In that case, the window length will have to be increased to decrease the frequency of detected peak. On the other hand, if the window length is too large, too many peaks might get "leaked" and the system will be under-compensating.

The threshold is another parameter that helps in changing the frequency of the detected peaks. A low threshold will generate a high frequency of detected peaks. However, the threshold has a direct impact on the output PAPR reduction and is normally dictated by the requirement of the application.

The Coordinate Rotation Digital Computer

The Coordinate Rotation Digital Computer(CORDIC) [21] in the clip stage is responsible for converting the incoming cartesian data from the testbench to polar form. The magnitude is needed to compare to the threshold and detect peaks. The phase is not needed in the peak detection process itself but is required in the later stage of the generation of canceling pulses. The CORDIC is an iterative process which translates to a 'for' loop with loop carried dependencies in SystemC code. This resulted in some architectural challenges which will be discussed in section 3.4.

Peak Detection Algorithm

The peak detector starts with checking whether the peak search window has started or not. If the window has not started yet, the peak detector checks if the magnitude exceeds the threshold. If yes, it is set as the maximum magnitude and a flag is used to signal the start of the window. Note, however, that a peak is not signaled to the resource yet. The next valid sample is read by the peak detector and it goes through the same procedure of checking the start of the window. Now that the start of the window has been signaled, the maximum peak is updated if the new value is greater than the current maximum value. In each iteration, a counter is updated until it equals the set length of the peak search window. When that happens, a valid peak flag is set high, the counter is reset to zero and the window flag is set low, indicating the end of the latest peak search window. Note that a new peak is signaled only at the end of the peak search window, regardless of where the peak occurs in it. This process is repeated indefinitely. A control flow diagram of the algorithm can be seen in Figure 3.1

Peak Information Packet

The peak information is sent to the resource in the form of a packet. The packet fields include

- peak magnitude
- peak phase
- peak index in peak search window, i.e. where exactly the peak was detected inside the window
- valid peak flag

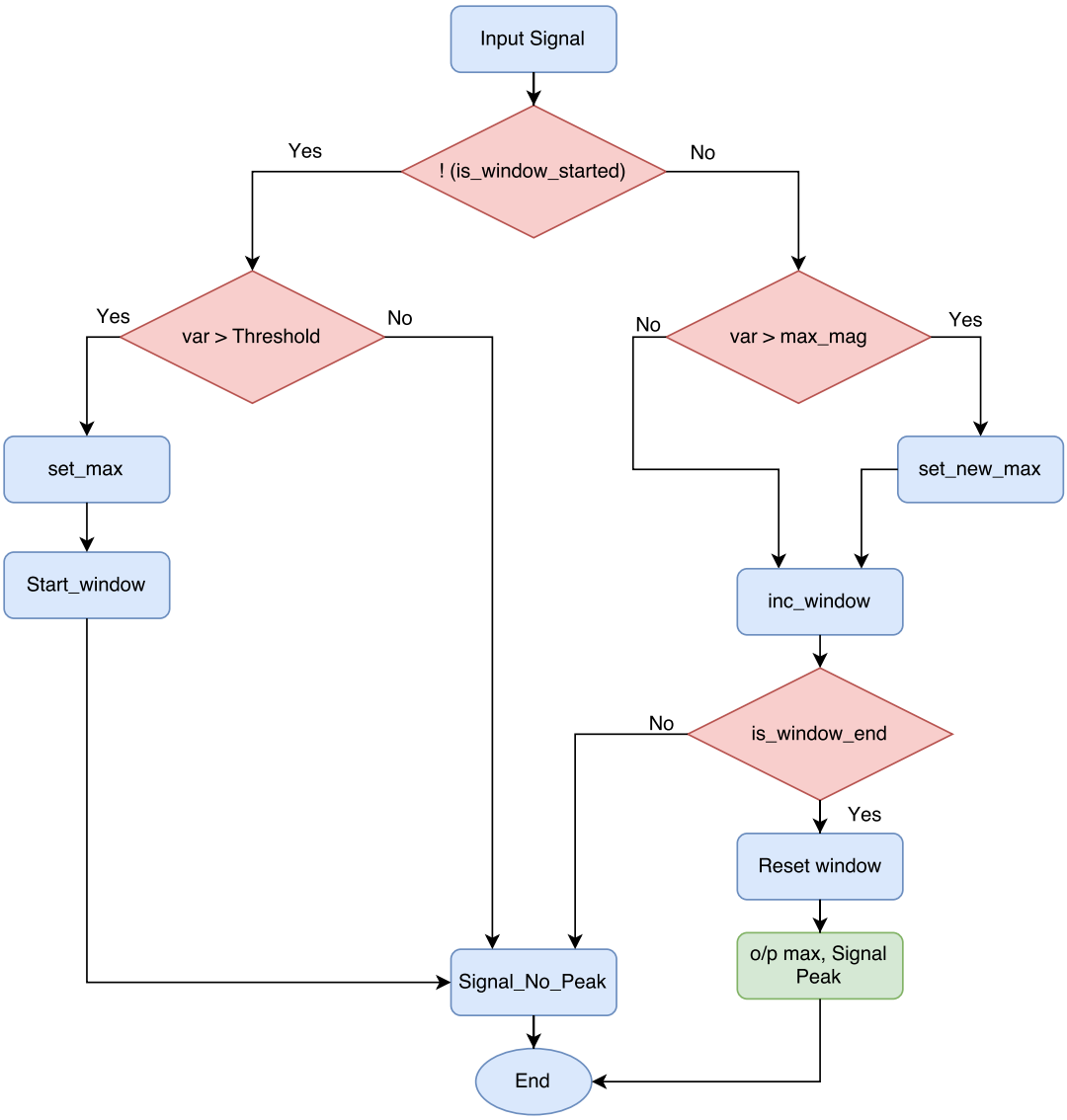


FIGURE 3.1: Peak Detection flowchart

Delay Memory

The delay memory is implemented as a circular buffer. The purpose of the delay memory is to delay the original signal long enough to compensate for the time spent in generating the first element of a canceling pulse for a detected peak. The factors that govern the length of this time are:

- The canceling pulse length as the detected peak has to be aligned with the center of the canceling pulse.
- The position of the peak inside the peak search window.
- The latency of the system in generating the canceling pulse.

So the delay memory size is calculated as follows:

$$Delay\ memory\ size\ =\ [(N/2)-1]+M+latency \tag{3.1}$$

where (N-1) is the maximum length of the canceling pulse and N is even. M is the length of the peak search window.

The memory is initialized by setting all elements to zero. In each iteration of the peak detection thread, the clip stage reads out an element from delay memory and increments the read pointer. After that, the clip stage increments the write pointer and writes the new incoming value to that index. The initial values of the read and write pointers define the delay required for a specific length of the canceling pulse, i.e. if the length of the canceling pulse is set to be (N-1) where N is an even number, a delay of length equal to the size of the array is needed. This would require the read pointer to be set to the first element of the circular buffer and the write pointer to the last. By varying the positions of these pointers, the length of the delay can be changed to accommodate for different canceling pulse lengths which is a desired feature as smaller canceling pulses are required for smaller peaks. This is due to the simple observation that larger peaks are neighbored by more values that exceed the set threshold and vice versa. However, this change cannot be done dynamically during normal operation of the system as it will disrupt the data but can be done while configuring the system. This thesis project has only been verified with a fixed canceling pulse length. Figure 3.2 shows the input, output, and the current state of the delay memory in three different cases.

Peak Correction Thread

The peak correction thread is responsible to read the delayed data from the delay memory and the canceling pulse from the resource, subtract the canceling pulse from the delayed original signal and send the corrected signal to the testbench for verification. Figure 3.3 shows the overall view of the clip stage. Sub-figure 3.3a shows both the threads of the module separately. Sub-figure 3.3b shows the module as a single unit where both the threads are merged.

3.2.2.2 Resource

The resource consists of a single thread, the canceling pulse generator.

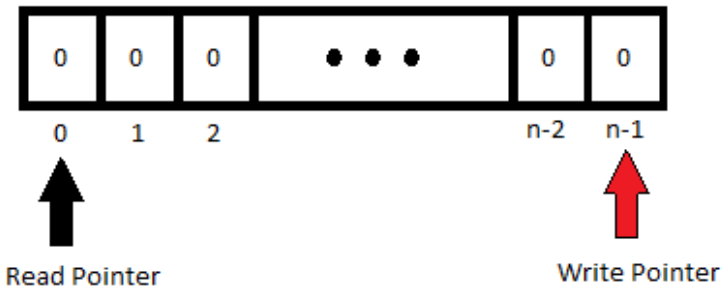
Canceling Pulse Generator

The canceling pulse generator is responsible for producing a signal corresponding to the detected peak information sent by the clip stage. The resource is a finite state machine and the output is dependent on the input and the current state. The resource outputs a sample of the cancellation pulse signal every cycle for cycles equal to the length of the cancellation pulse if it receives a high peak flag. Following is a description of the components and the actions taken in the resource.

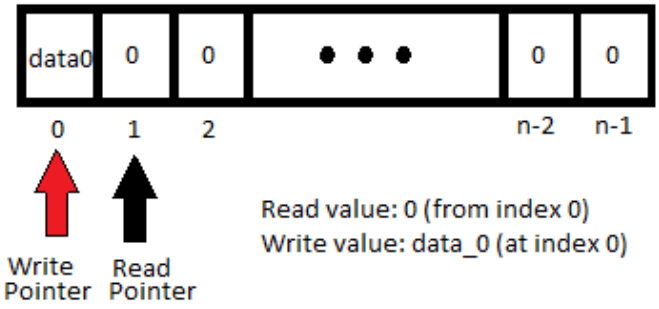
State Variables

The resource keeps track of the following variables:

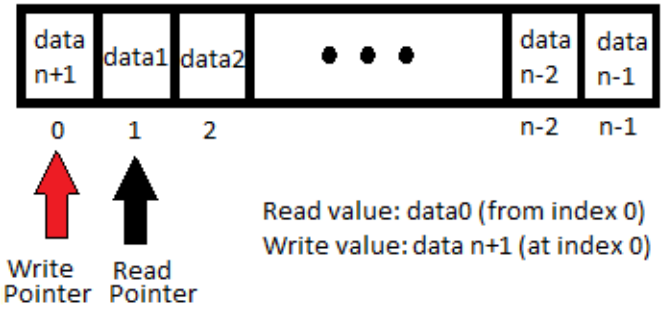
- Magnitude of the detected peak
- Phase of the detected peak
- Delay counter
- Resource free flag
- Resource counter



(A) Initial state of Delay Memory



(B) State of Delay Memory after first iteration of peak detector thread



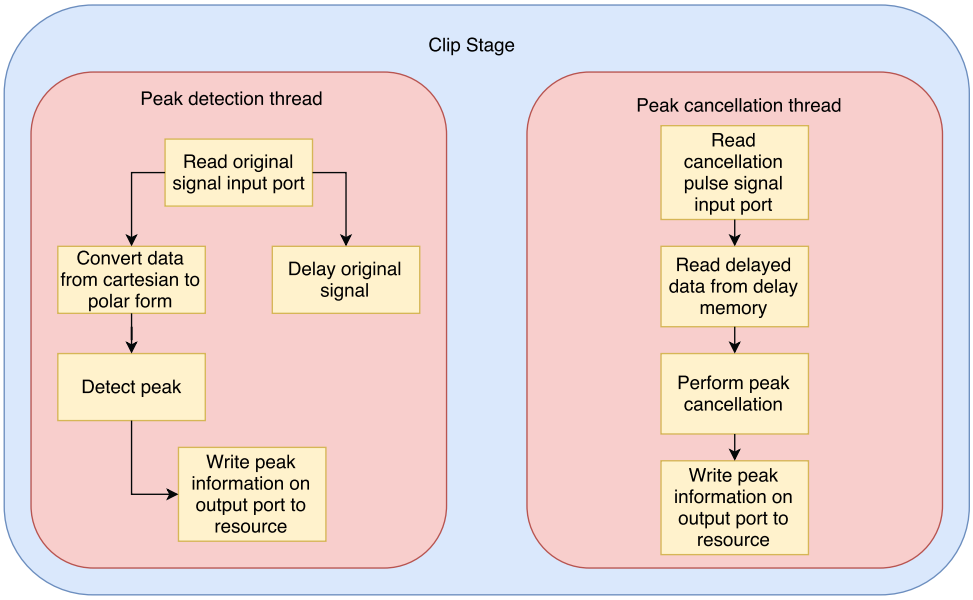
(C) State of Delay Memory after (N+1) iteration of peak detector thread

FIGURE 3.2: State of Delay Memory at different intervals

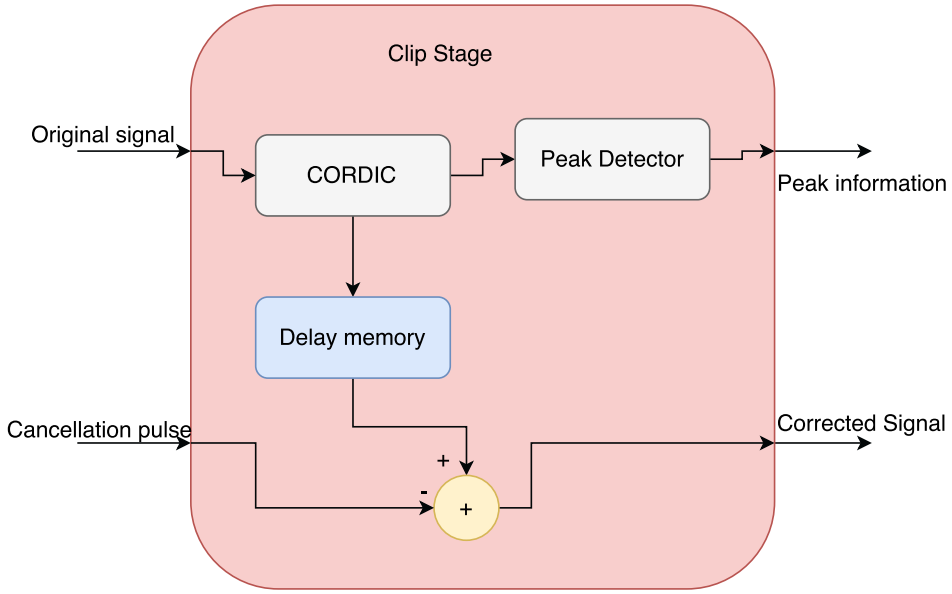
The magnitude and phase of the detected peaks are needed for the signal generation. The index of the peak is needed to instantiate a delay counter to delay the cancellation pulse. The reason for this is explained in the subsection Delay in Cancellation pulse of section 3.2.2.2. The Resource free flag keeps track of whether the resource is currently busy with generating a cancellation signal for a peak and, lastly, the resource counter is needed to stop the signal generation once the counter has run the allowed length of the cancellation pulse.

Control path and data path separtaion

The control path and data path are separated into two separate functions. This was an architectural decision and a general practice which is recommended for HLS to successfully generate RTL from the SystemC code. This is further discussed in section 3.6.



(A) Clip stage threads



(B) Clip stage as a single unit

FIGURE 3.3: Overall view of the clip stage

Cancellation pulse generation thread

The cancellation pulse generator thread starts with checking if a new peak has been detected or not. If yes, it checks if the resource free flag indicates availability. If the resource is available, the state variables are updated with the new values; the peak information state variables are updated with the incoming peak information packet, the resource free flag is set to false and the resource counter is reset to 0. Next, the data path function and then the control path functions are called to generate the output data and update the state variables respectively. If instead, the resource free flag indicates that the resource is not available to work on a new peak, the state variables are not updated before calling the data and control path functions. This means that the resource is currently busy generating a cancellation pulse for

a previously signaled peak. This case corresponds to a peak leak, i.e. the newly detected peak is discarded.

If no new peak is signaled, the resource free flag is checked. If the resource is available, this means that the resource does not have to generate any signal. In this case, the state variables are not updated and zero is sent to the output. If the resource free flag indicates that the resource is busy, the resource calls the data and control path functions one after the other.

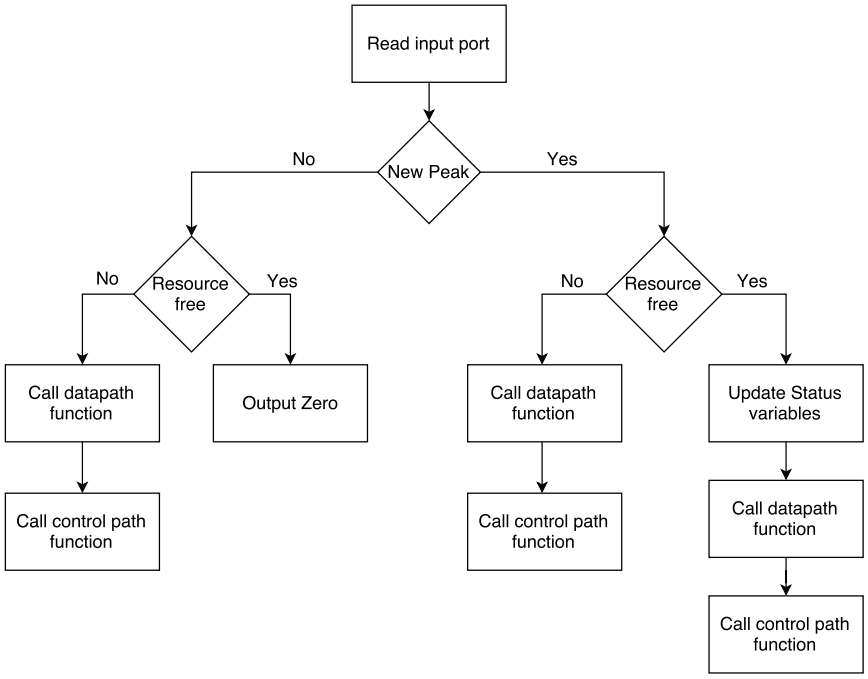


FIGURE 3.4: Resource flowchart

Figure 3.4 is a representation of all the states of the resource. By inspecting this flowchart, it can be seen that it can be simplified into 3.5. However, the first one depicts a better picture of the different cases in the design.

Delay in Cancellation Pulse Generator

A delay is needed in the cancellation pulse generation to synchronize it so that its center is aligned with the detected peak in the delayed original signal. In subsection delay memory of section 3.2.2.1, a formula was used to decide the delay in the original signal. One factor in that equation was the length of the peak search window. This factor delays the original signal with the assumption that the peak occurs on the first index of the peak search window. This delay accounts for the fact that the peak is signaled at the end of the window regardless of its position. However, the peak could have occurred at any of the indices. To compensate for this extra delay, the cancellation pulse also has to be delayed. The length of the delay is equal to the index position of the peak in the peak search window. A delay counter is assigned the value of the index of the peak. This delay counter is counted down until it reaches zero, at which point the generation of the cancellation pulse is started.

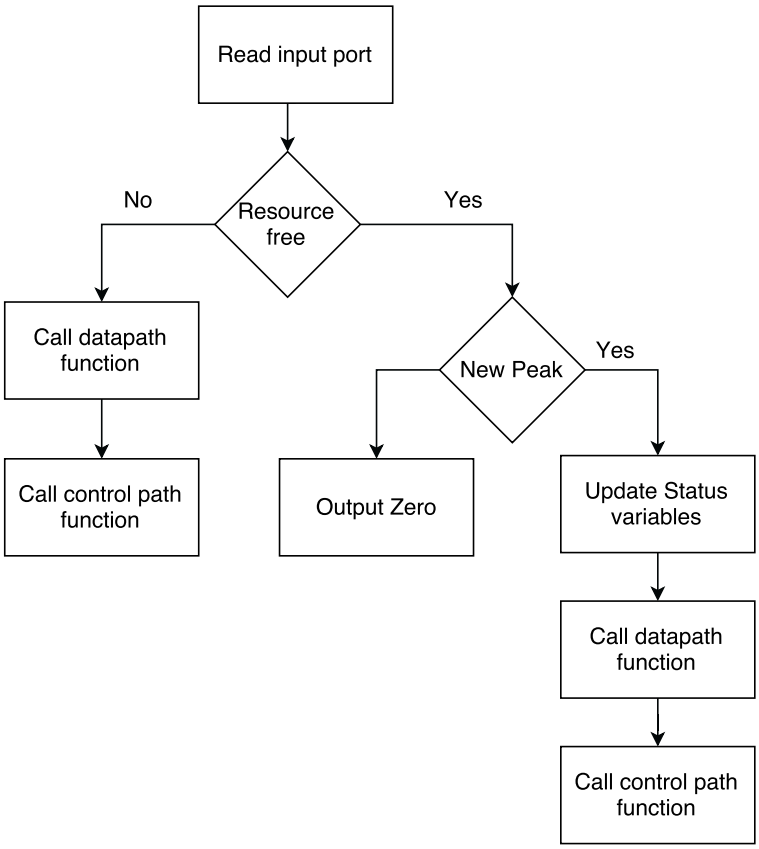


FIGURE 3.5: Simplified Resource flowchart

Data path function

The data path function is responsible for deciding what the output should be depending on the delay variable. If the delay counter is greater than zero, zero is output by the resource. Otherwise, a computation function is called to calculate the samples of the cancellation pulse. Figure 3.6 shows a simple flowchart of the resource datapath function.

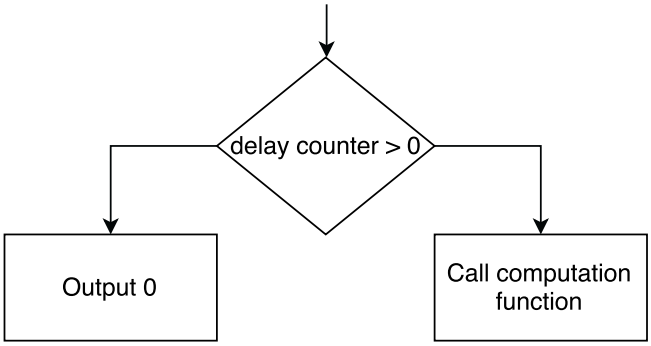


FIGURE 3.6: Resource Datapath flowchart

Coefficient Memory

A coefficient memory holds the complex taps, in the polar form, of an FIR filter that was pre-designed. The peak is passed through this filter to generate the cancellation pulse. The memory size needed to store the fir filter coefficients has been reduced by observing that the filter is symmetric, i.e. the magnitude of the signal is even symmetric and the phase is odd symmetric. The relevant properties are expressed as mathematical formulae in equations 3.2 and 3.3:

$$|X(k)| = |X(-k)| \tag{3.2}$$

$$\angle X(k) = -\angle X(-k) \tag{3.3}$$

The magnitude and phase components of the filter are shown in 3.7. Thus an array of size N/2, where the length of the filter is N-1 and N is even, is needed to replicate the entire filter.

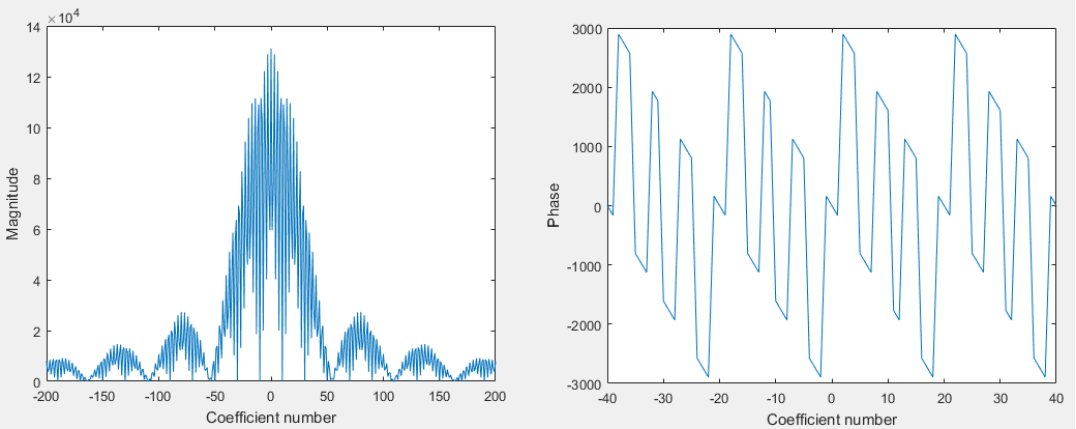


FIGURE 3.7: Coefficient magnitude and phase signals

Computation function

The computation function uses the resource counter variable to index a coefficient array. If the resource counter value is less than or equal to N/2, the array is accessed in the ascending order. Otherwise, it is accessed in the descending order. A negative sign is appended to the phase component when accessing in the descending order as per the equation 3.3.

Once the magnitude and phase are retrieved from the coefficient memory, the difference of the peak magnitude of the threshold is multiplied to the coefficient magnitude. This results in a scaled magnitude value for one sample of the cancellation pulse. The phase component from the coefficient memory is added to the peak phase component. This results in the corresponding rotated phase component of the cancellation pulse sample. Finally, the polar sample is passed through a CORDIC which is responsible for converting it to cartesian form. The resulting complex number in cartesian form is one sample of the cancellation pulse signal and is sent to the output of the resource. The flowchart is represented in figure 3.8.

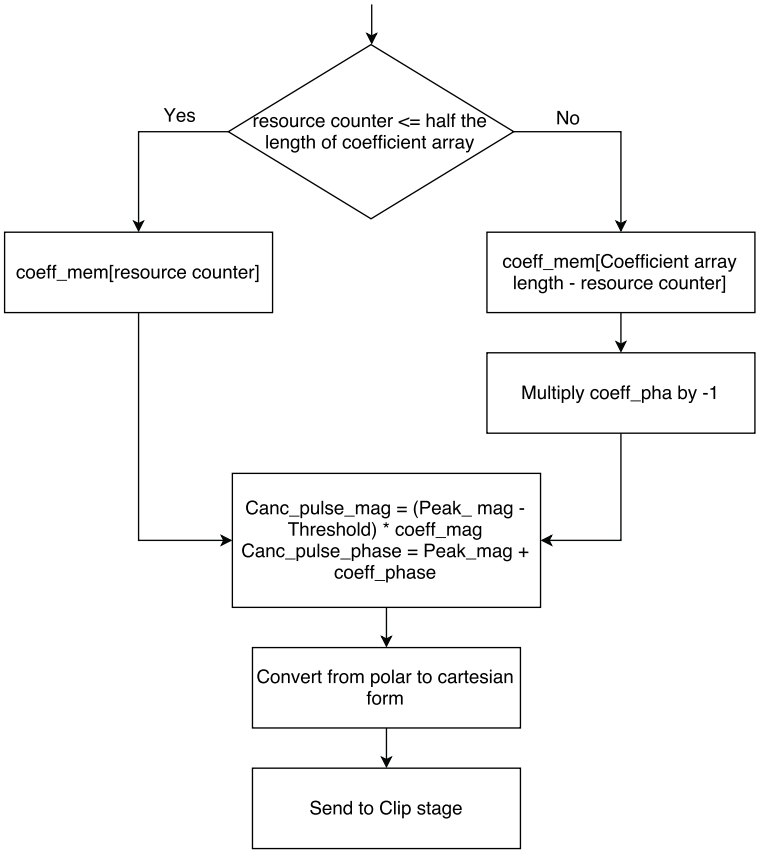


FIGURE 3.8: Computation function flowchart

Control path function

The control path function works on the state variables of the resource. If the delay counter is greater than zero, the delay counter is decremented. This case means that the cancellation pulse still needs to be delayed. If it is equal to zero and if the resource counter is equal to the length of the cancellation pulse length, the resource free flag is set to true and the resource counter is reset to zero. This case signifies the end of the cancellation pulse generation for a particular peak. Finally, if the delay counter is equal to zero and the resource counter is not equal to cancellation pulse length, the resource counter is incremented. Figure 3.9 shows the flowchart. Figure 3.10 shows the complete system diagram of the vanilla version of PC-CFR.

3.3 Advanced Implementation of PC-CFR System

The vanilla version of the PC-CFR system was inadequate in terms of PAPR reduction. The reason for this is that the number of peaks canceled is dependent on the number of resources available. More resources mean that more cancellation pulses can be generated with their durations overlapping. Apart from peak leaks, peak regrowth can also occur which would require further processing of the corrected signal to get the desired PAPR. So to summarize, there are two problems that need to be solved:

- The capacity to generate cancellation pulses in parallel needs to be increased.

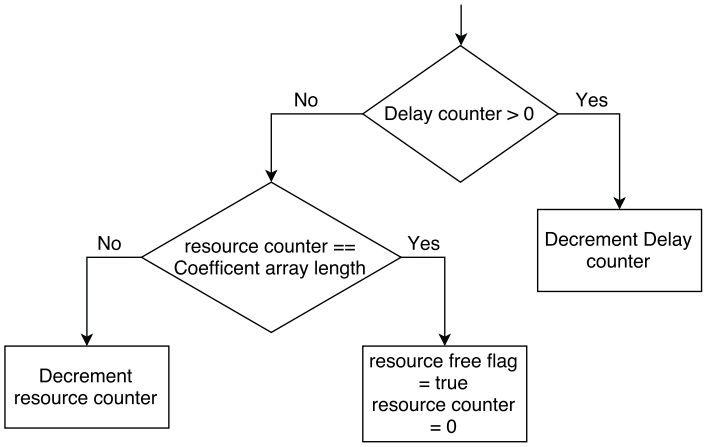


FIGURE 3.9: Control path flowchart

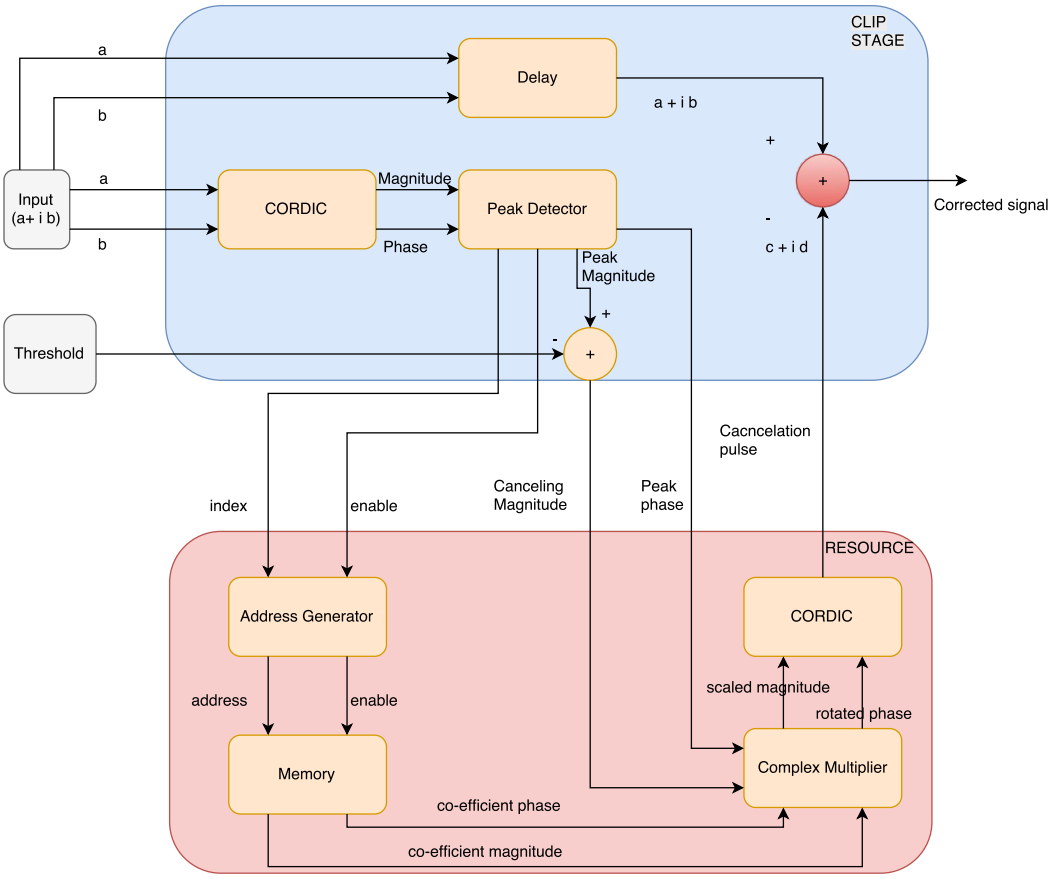


FIGURE 3.10: Vanilla version system diagram

- The signal needs to go through multiple passes of peak cancellation.

The trivial solution to the first problem would simply be to increase the number of resources. Increasing hardware means more parallel processing power. However, each resource contains a memory and it is not desirable to have too many memories in the design as they take up a lot of area. In an attempt to isolate different design decisions separate so as to keep the explanation simple, the trivial solution will be

discussed first. Later, a solution to decrease the memory area will be discussed in section 3.4.

The second problem is tackled by using a cascaded design where each clip stage feeds into the next clip stage except the last one, which sends the output to the testbench. The high-level abstraction of the cascaded design can be seen in 3.11. In this new design, there is not a one-to-one correspondence between the N clip stages and M resources. Thus, there needs to be an intermediate block between the clip stages and the resources to allocate the peaks to the resources and return the cancellation pulses back to the clip stages. Following is a breakdown of the advanced PC-CFR DUT.

- Clip stage block
  - Cascaded clip stages
- Peak manager
  - Allocator
  - Resources

3.3.1 Clip Stage Block

The clip stage block houses all the clip stages. Each clip stage has two inputs and two outputs. One of the inputs is labeled as the original signal which is the signal coming from either the testbench or the corrected signal coming from the previous clip stage. The second input is the cancellation pulse coming from the resource and through the allocator. Out of the two outputs from the clip stage, the first is the peak information that is sent to the resources through the allocator and the other one is the corrected signal that is sent either to the next clip stage or the testbench. This can be seen in Figure 3.11.

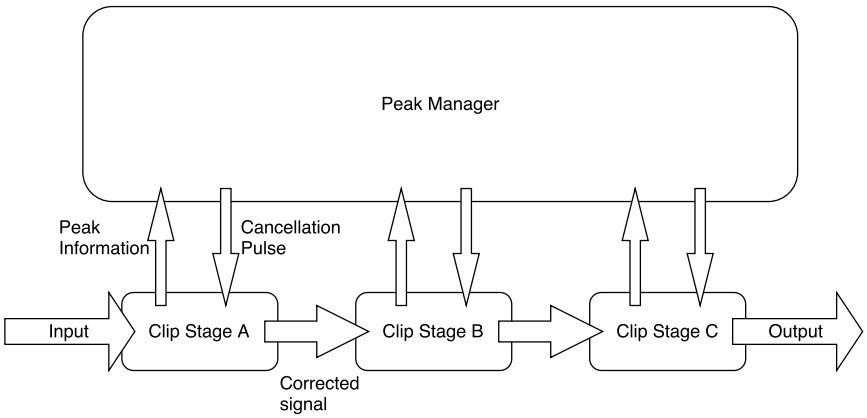


FIGURE 3.11: Cascaded PC-CFR design

3.3.2 Peak Manager

The peak manager is responsible for handling the peak information coming from the clip stages. It houses the allocator and the resources. Following is a description and breakdown of the allocator:

3.3.2.1 Allocator

The allocator consists of two threads. One of them is responsible for allocating the peaks to the resources based on a mapping function. The other thread is responsible for reading the outcoming cancellation pulses from the resources, identifying which of these are bound to the same clip stage, adding them up and sending them to that clip stage. Remember, there are more than one resources now, and so multiple peaks from the same clip stage can possibly be handled simultaneously which raises the need to add the overlapping cancellation pulses of the peaks from the same clip stage.

The Allocating Thread

As stated before, the allocating thread maps the incoming peaks information to the resources. The allocating thread has several major components that work together to perform these tasks. First, the allocator needs to find if there are any available resources to which the incoming peaks can be mapped. If there are available resources, then the peak will be mapped to one of them and the information of available resources will be updated. Following are the major components:

- Resource table
- Valid peak counter
- Resource available vector
- Search function
- Mapping function
- Resources per clip stage counter

Resource Table

The resource table holds the peak information coming from the clip stages. Each row in the resource table is dedicated to a resource, i.e. the peak information in the first row is sent to the resource with first index and the information in the Nth row goes to the Nth resource. Thus, the number of rows in the resource table is equal to the number of resources. Each row holds the following information

- Magnitude
- Phase
- Index in peak search window
- Valid peak
- Clip stage id

As can be seen, clip stage ID is the only new variable that is being sent to the resource as compared to the vanilla version. This is needed later in the addition of the cancellation pulses belonging to the same clip stage.

Valid Peak Counter

Valid peak counters are needed to keep track of the status of the resources. Each resource has its own dedicated counter. Once a counter reaches the length of the cancellation pulse, it means that the resource is now done with the generation of the signal and is now available for a new job. The resource available vector is updated to reflect this.

Resource Available Vector

The resource available vector is a vector with as many bits as there are resources in the system. Each bit represents the busy status of a resource. If a bit is high, it means that the corresponding resource is available and vice versa. The resource available vector is updated twice in a single run of the thread. First in the start of the thread after checking if any new resource has just become available and then after the mapping function has been called. Resources per clip stage counter Each clip stage has its own such counter. The counter keeps track of the number of resources that are currently working on a peak detected by a certain clip stage. This is needed as this number can be limited because as an observation, clip stages that come later in the chain tend to get a signal with a smaller number of peaks. Thus, the majority of the resources are needed for the clip stages in the start of the chain.

Search Function

The search function looks for available resources. It takes the resource available vector as an input and returns the indices of the first N available resources, where N is equal to the number of clip stages. The reason for this is that the maximum number of peaks that can arrive in a single iteration is equal to the number of clip stages. The search function is implemented quite similar to a priority encoder [22] [23] where the lower order indexes are checked first. Thus, the lower indexed resources have a bias or priority when peaks are being allocated. 3.12 shows the flowchart of the search algorithm.

As the diagram shows, the function checks each bit of the resource available vector starting from the least significant bit and goes on until either it has found enough available resources that can cater to all the clip stages, i.e. the number of available resources is equal to the number of clip stages, or it has checked all the bits in the resource available vector and has not found enough resources to work for all the clip stages. The resource identifier vector stores the available resource numbers. This vector is used by the mapping function.

Mapping function

There were a couple of options while designing the mapping function. These were:

- Static mapping
- Dynamic mapping

In static mapping, each resource would be connected to a fixed clip stage. So for example, the first clip stage would have access only to the first eight resources, the second would be connected to only the next four and so on. This method of assigning has its advantages and disadvantages. The advantage is that it greatly

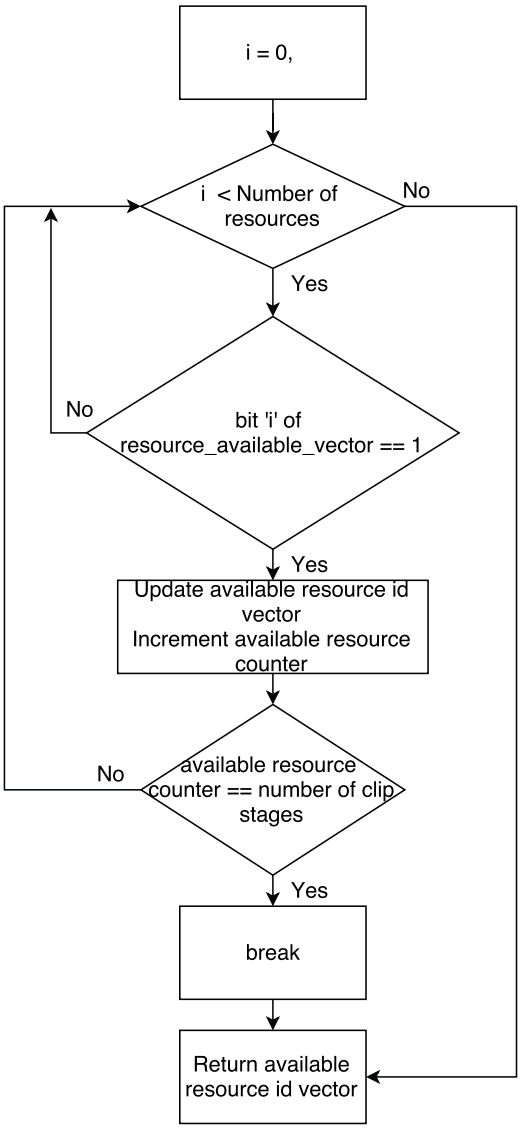


FIGURE 3.12: Flowchart of the function to search for available resources

simplifies the allocator design. The allocator does not have to know which resource is currently working for which clip stage to add up the cancellation signals. So this information does not have to be sent by the resource to the allocator. The disadvantage of this technique is that it causes the system to be under-utilized. If a clip stage detects new peaks but all of its connected resources are busy, it will not be able to use one of the available resources that are connected to other clip stages.

In dynamic mapping, resources are not dedicated to clip stages. Rather, the allocator selects from a pool of resources and assigns them to different clip stages. This removes the problem of under-utilization. However, now the allocator either has to keep track of which resource is assigned to which clip stage or resources have to send this information to the allocator. This design incorporates dynamic mapping.

The mapping function takes several inputs to decide whether a peak can be mapped to a certain resource. These inputs are:

- Valid peak

- Resource available
- Resource per clip stage counter

The function checks if the peak is valid and whether the search function returned any available resources and, lastly, whether the number of resources assigned to a clip stage has not reached a set limit. If all the conditions are met, then the peak information is written in the resource table row corresponding to the smallest available resource index and the valid peak counter is reset to indicate the start of the canceling pulse generation. The resource available vector is updated to show that the new busy resource. This procedure is repeated equal to the number of clip stages to check for valid peaks from all of them.3.13 shows the flowchart of the mapping algorithm.

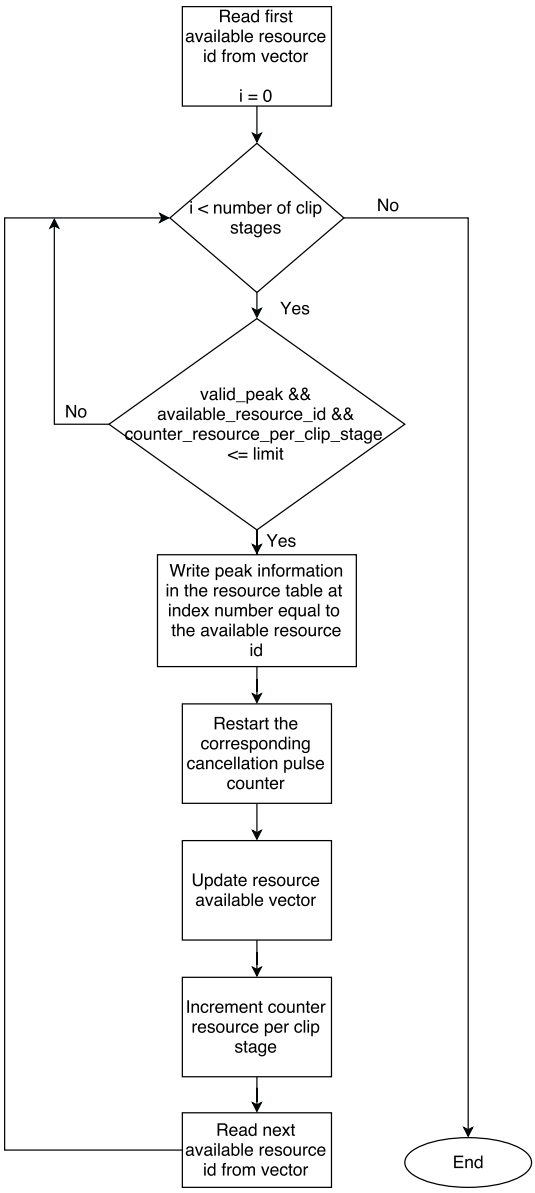


FIGURE 3.13: Flowchart of the mapping function

Dataflow of allocating thread

The allocating thread starts by reading the peak information from all the clip stages simultaneously. After that, the valid peak counters and resource available vector are updated. The search function is called to find available resources and then the mapping function is called to allocate these resources to peaks. The resource available vector is updated again and the peak information is written to the resources. The flowchart of the allocating thread is shown in 3.14

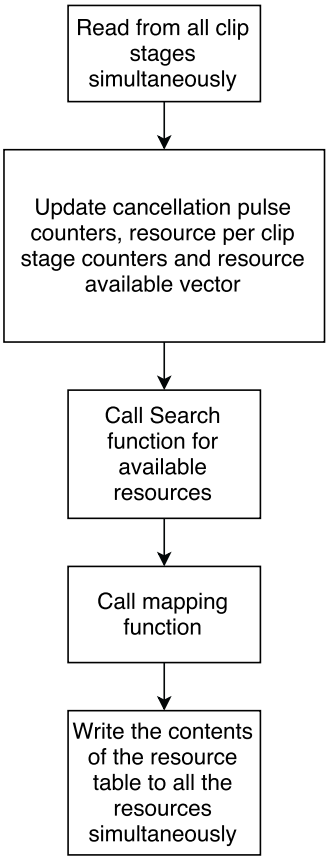


FIGURE 3.14: Flowchart of the allocating thread

Cancellation Signal Adder Thread

The cancellation pulse adder thread reads from all resources in parallel. The resources return a complex number which is a sample of the cancellation pulse along with a tag that identifies which clip stage it belongs to. In every iteration of the thread, all the samples that belong to a single clip stage are added up and sent to that clip stage for peak cancellation of the original delayed signal. Figure 3.15 shows the flowchart of this thread.

Figure 3.16 shows the system diagram of the advanced implementation of PC-CFR.

3.4 Important Architectural Features

This section describes some of the important architectural features implemented in the advanced PC-CFR system which has a major impact on the design.

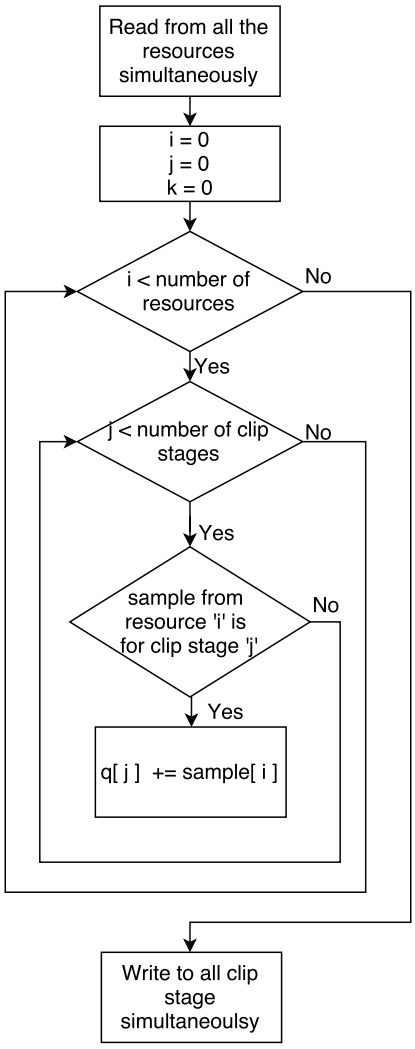


FIGURE 3.15: Flowchart of the cancellation pulse adder thread

3.4.1 Pipelining

The PC-CFR requires a predefined throughput as the system is expected to get a stream of input at a certain rate. Since the entire purpose of using SystemC and HLS was to design at a higher abstraction level without defining cycle accurate operations with a fixed latency and thus having a flexible implementation, pipelining was incorporated with an initiation interval that would allow the system to read the input at the desired rate. The initiation interval is the interval between the start of two consecutive operations of a given type. This concept is elaborated in figure 3.17.

After the elapse of the initiation interval, the pipelined module goes to the start of a new thread execution which starts with a read on the input ports. Similarly, after a time equal to the latency of a module, a new output is generated and written to the output port after every initiation interval. This way, the desired throughput is achieved.

If the clock frequency at which the system operates perfectly matches the input throughput, then the initiation interval will be set equal to one since the system expects a new input sample every cycle. This value can be greater than one for a

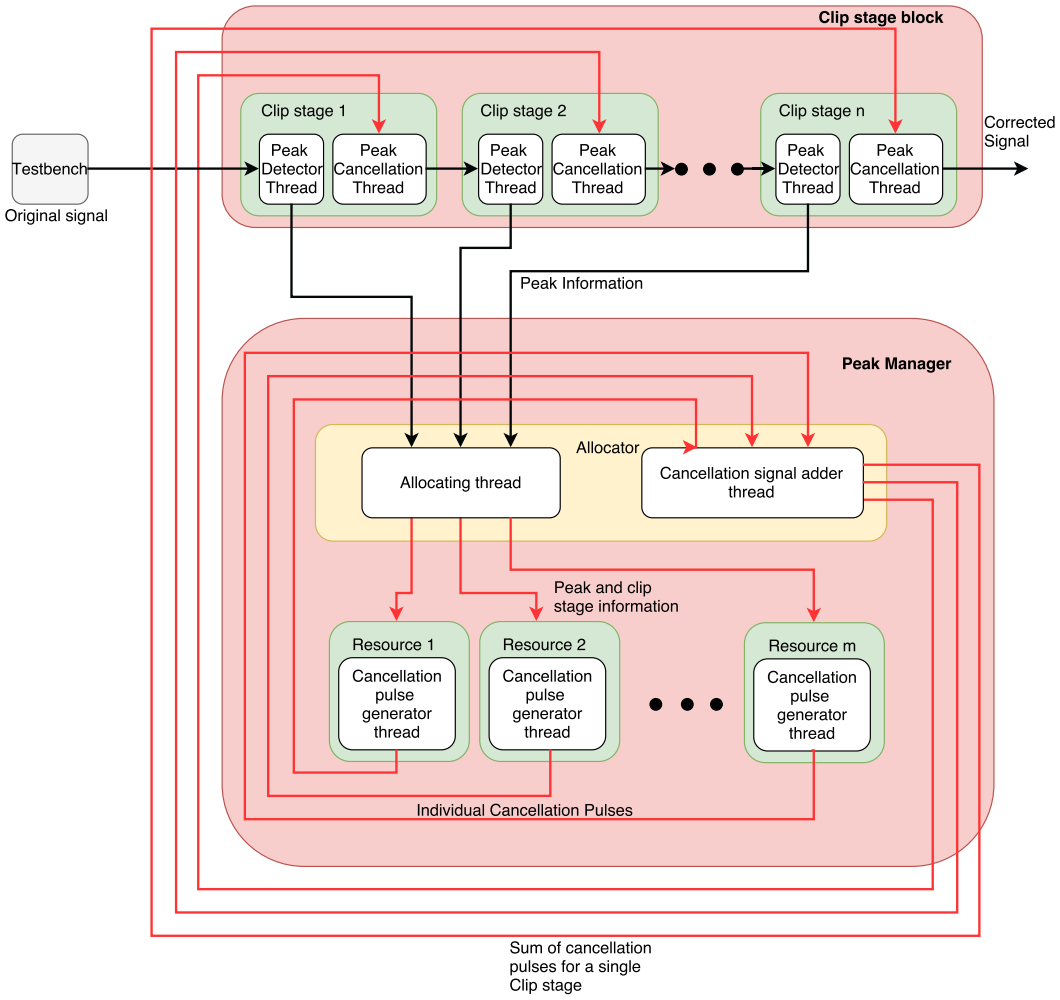


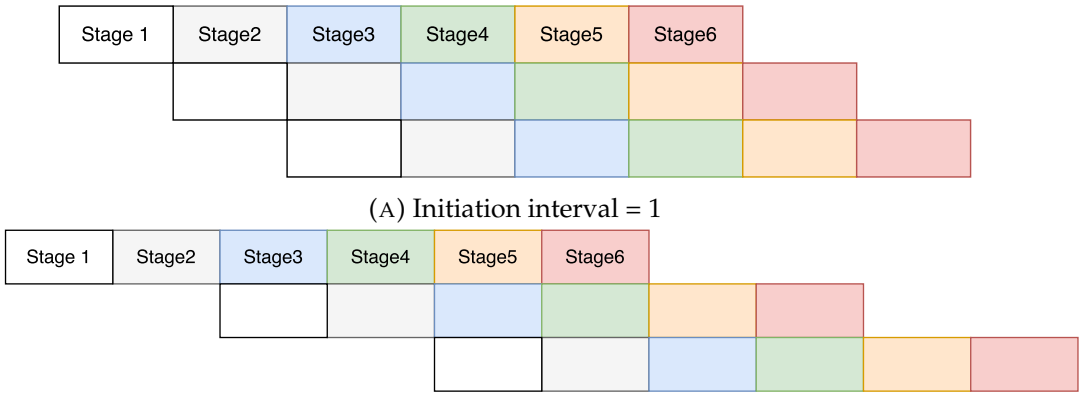
FIGURE 3.16: System Diagram of advanced implementation

faster clock. This scenario is discussed in the 3.4.2.

3.4.2 Time Division Multiplexing

As discussed before, an increase in the number of resources is needed to decrease peak leaks and achieve sufficient PAPR reduction. However, increase in the memory area is a concern as each resource contains a memory component. Time division multiplexing is one approach to get around the area inflation problem. The solution is to have the system run at a faster clock rate but keep the input rate the same. The resource would be time division multiplexed and each slot would work on a different peak. The number of slots is defined by the ratio of the clock frequency and the input rate. For example, a clock frequency of 1 GHz and an input rate of 250 MHz would allow the resource to be time division multiplexed with four slots.

This solution reduces the memory area in the sense that the system now has four times more virtual resources with the same number of memory. This can be understood by the fact that each resource can now tend to the generation of four cancellation pulses simultaneously. However, there is a slight increase in the resource area with TDM. This is because each slot needs a copy of the status variables as well as an increase in the multiplexer (MUX) area but this is a minor increase



(A) Initiation interval = 1

(B) Initiation interval = 2

FIGURE 3.17: Examples of initiation interval of a pipeline

as compared to having four resources with four separate memories. TDM does, nevertheless, increase the switching activity in the resource.

The TDM causes the initiation interval of the resource pipeline to be different than the other modules. Since the resource is expected to work on a new sample every cycle, the initiation interval is set to be one instead of the ratio between the clock frequency and the input signal rate of the system. Thus, the system reads in a new input every cycle and, after the latency of the module, writes a new output every cycle.

This design decision affects the allocator design as well as it needs to source the resource and sink the result of cancellation signal generation. Figure 3.18 shows the method of performing TDM and how it affects the allocator. In the figure, the allocating thread sends peak information to the resource with varying delays depending on the timing slots of the resource. The cancellation pulse adder thread reads from the resource with corresponding delay elements at its input to compensate for the delay added by the allocating thread. In this particular example, the resource has four time-slots and, essentially, behaves as four virtual resources.

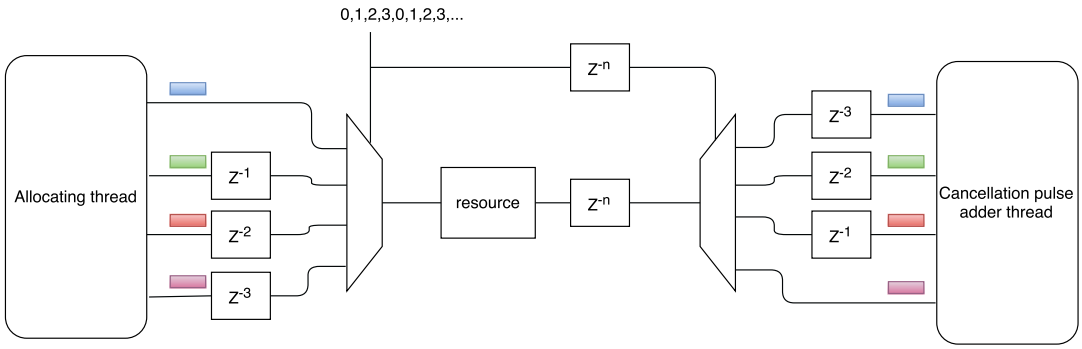


FIGURE 3.18: Method for implementing time division multiplexing

As opposed to the clip stage that only operates on a new input data every N cycles where 1: N is the ratio between the input rate and the clock frequency, the allocating thread in the allocator needs to write to the resource every cycle because of the time division multiplexing. However, the allocator still reads at the same rate as the rate of the output coming from the clip stage, which is every N cycles. Thus,

Thread name	Initiation Interval value
Peak detector thread	Clock to sample ratio
Peak cancellation thread	Clock to sample ratio
Allocating thread	Clock to sample ratio
Cancellation signal adder thread	Clock to sample ratio
Cancellation pulse generator thread	1

TABLE 3.2: Pipeline initiation intervals in threads

the allocating thread writes to all the resource every cycle but reads from all the clip stages every N cycles. On the contrary, the cancellation signal adder thread in the allocator reads from the resource every cycle but writes to the clip stages every N cycles. So, the pipeline initiation interval for both the threads is set to N but the allocating thread is forced to write every cycle instead of every N cycles as dictated by the pipeline initiation interval. Similarly, the cancellation signal adder thread is forced to read every cycle.

The pipeline initiation intervals of the threads in the PC-CFR after the implementation of time division multiplexing is are written in table 3.2

3.4.3 Separation of communication and computation regions

The HLS tool recommends separating the communication from the computation regions. This segregation of the communication from the computational part of the design is very helpful as, in HLS, it is desirable to perform design space exploration in the computational part while implementing the communication region as it is written. For example, if a module is expected to send five packets of data in five consecutive cycles, the HLS tool should not change that. This can be achieved by allowing the HLS tool to explore different implementations of the algorithm by varying the latencies of the computational part. This is done by specifying the protocol and non-protocol blocks. A protocol block is implemented as is, i.e. if the read or write has to occur in one cycle, the tool will not alter it. Outside of the protocol blocks, the tool ignores systemC wait statements and can add or remove cycles to get an optimal schedule. These regions are non-protocol regions. However, non-protocol blocks can have latency constraints.

```
{
//Protocol region for reading from input port
   read();
}
{
//Non-protocol region for
performing computations
   #HLS_LATENCY_CONSTRAINT(arguments)
   computation_function();
}
{
//Protocol region for writing to output port
   write();
}
```

It is to be noted that the communication protocol used in the system also governs the use of the protocol and non-protocol blocks. Since the non-protocol blocks do not have a fixed latency, the overall latency of the module would also be variable. If the communication protocol between modules does not include a handshaking mechanism, the system would not be synchronized. The PC-CFR system does not use a handshaking protocol. Thus, latency constraints on the non-protocol regions must be used so that the synchronization problem does not occur. The communication protocol is discussed more thoroughly in 3.4.4. The conditions on selecting the latency constraint values are discussed more in detail in 3.5.1.

3.4.4 Communication protocol

In SystemC, the interface is defined quite similar to the definition of other modules; as a class object. This allows for designing simple communication channels such as a wire or even complex hierarchical structures. The communication protocol used on the interfaces in PC-CFR includes two signals; a data signal and a valid signal as shown in figure 3.19. As stated earlier, a new input to the PC-CFR system is not sent every cycle, rather it is sent on the data line at a throughput defined by the clock to sample ratio. So for a few cycles, the input data remains stable. The valid signal is used to identify the cycles in which a new data value is sent. The signal goes high when the data is valid and remains low otherwise. If a module reads the data signal when the valid signal is low, it is still used for the next computation. However, the valid signal is propagated on to the next module. In this case, since the valid signal is low, the output data is also tagged as invalid.

The protocol does not use any form of acknowledgment. The receiving module does not send back an ACK(acknowledgment) signal to the sending module upon receiving data. The protocol does not use blocking reads or writes either. The sending module sends the data when the write function is called without checking if the receiving module is ready to read. Similarly, the receiving module does not wait for the sending module to send the data before reading the input port when the read function is called.

The protocol used in this design is needed because there is no handshaking in communication and is essential for synchronization of the system. If the modules are not properly synchronized, the output data from the PC-CFR will be invalid, as indicated by the valid signal, and debugging will become easier.

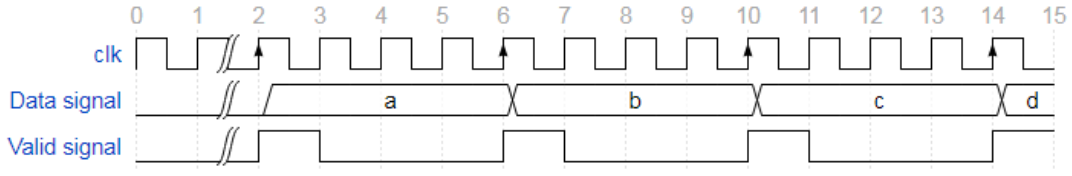


FIGURE 3.19: Communication channel signals

3.4.5 Vendor Memories

The tool allows for using vendor memories by specifying the size, timing information, ports and path to the Verilog files of the memory design in a memory compiler. These vendor memories can be bound to arrays in the design. The array that holds the coefficients of the filter used in the generation of the cancellation

pulse in the resource is bound to a RAM memory with a single port for reading and writing since it does not have to be written to after the initialization phase. The delay memory in the clip stage is also bound to a RAM memory. This one, however, has separate read and write address ports for allowing simultaneous read and write access. It should be noted that the delay memory is a multiport memory that is being used in a pipelined design. A max distance of one cycle between the read and write operations is set to guarantee a safe and functionally correct schedule. Otherwise, the read and write operations might not be scheduled in the correct order.

3.4.6 Parameter Signals and Design flexibility

The PC-CFR system can have several inputs that affect the performance and characteristics of the design. The design has been made to incorporate templates and macros as much as possible to have a flexible design. Thus, the parameters can be modified simply by changing the macros in a header file. The parameters are as follows:

- Number of clip stages
- Number of resources
- Latency of modules
- Limit on the number of resources per clip stages at a time
- Peak search window length
- Threshold
- Cancellation Pulse length

The number of clip stages and resources can be changed by just altering the values of the corresponding macros. These are hardware units, so synthesis has to be re-run each time these are changed. The same applies to the latency of the modules. The latency of different modules can also be changed by modifying macros for these modules and, similarly, synthesis has to be rerun afterward

The limit on the number of resources per clip stage at a time, peak search window length and threshold are implemented as input signals to the design. This is one major distinction between the previous design [12] and this one. The previous design had static limits hard-coded in RTL and thus did not allow these limits to be changed during run-time. This design allows this by introducing these parameters as an input signal to the design. These do not use the same interface as the rest of the design where there is a valid signal apart from a data signal. These signals only have a data wire and can be altered without having to re-run synthesis.

The cancellation pulse length is fixed for this design but can be modified to allow different ones as well. Normally, a smaller cancellation pulse is desirable for small peaks because such peaks would be neighbored by a lesser number of samples above the threshold as compared to a large peak. This would require some extra logic to calculate the size of the cancellation pulse apart from other changes in the delay functions.

3.5 Use of HLS Directives

The HLS tool provides some very useful directives to help users to gain control over different architectural features of the design. A subset of these directives was used in this project and their purpose is written down below:

3.5.1 Latency Constraint Directives

Design space exploration was one of the objectives of the project. Latency constraint directives were used to allow the tool to perform DSE. As explained in the 3.4.3, the non-protocol blocks are subject to latency change. However, the design cannot have just any range of latencies. As can be seen in the figure 3.16, the allocator reads from all the clip stages simultaneously. This would mean that the system should be set up so that all the clip stages get a new input in the same cycle. Since the clip stages are pipelined with an initiation interval of N, N being the ratio of the clock frequency and the input throughput, they read a new value every N cycles. So the latency of the path from a clip stage to the next clip stage (through the allocator and the resource) needs to be a multiple of N. This was achieved using latency constraint directives.

This means that the path latency can be incremented or decremented only in steps that are multiples of N. This is one of the limitations introduced by the nature of the design.

3.5.2 Loop Unrolling and Register Flattening Directives

Loop unrolling directives are used extensively in the design for simultaneous port reads and writes as well as updating arrays. If the loops are not unrolled, they become multi-cycle operations which can cause timing violations in the design. Loop unrolling goes hand in hand with another concept called register flattening. Register flattening allows for multiple elements of an array to be accessed in parallel. If the loop is unrolled, and an array is being accessed in that loop, then that array must be flattened. This is done using a register flattening directive. The downside of this is that the MUX area increases. On the other hand, if an array is not flattened, it is implemented as a single port ram and each access takes a cycle.

Loop unrolling, however, has a prerequisite and that is that there should not be any loop carried dependencies. A loop carried dependency occurs when an iteration needs data from a previous iteration of the loop to complete its task. The code had to be written with this in mind.

3.5.3 Datapath Optimization Directives

Datapath optimization directives were used in the design to avoid timing and pipeline violations. This option allows the tool to generate a custom part for the region of the algorithm where this directive is used. The tool binds the logic as a datapath operator and generates an optimized gate level result. If a single cycle part is not possible, then a multi-cycle part is created. This basically means that the part has a latency greater than one or, in other words, is pipelined. One example of the use of this directive is in the clip stage. The clip stage holds a CORDIC which is an iterative algorithm. The logic synthesis was returning a negative slack before the use of the datapath optimization directive and was solved after incorporating it.

3.5.4 Pipelining Directives

The pipelining of the modules is achieved by using a Pipeline HLS directive that allows selecting the region, initiation interval, and stalling mechanism. The stalling mechanism controls how the pipeline behaves when there is no new data at the input of the pipeline. There are two options to select from:

- **Hard Stall:** The pipeline stalls without letting itself get empty
- **Soft Stall:** The pipeline continues working until it is empty and then stalls

The soft stall option allows for the system to function normally until all data has been processed. This was more convenient of the design. However, the tool had a limitation to the use of the soft stall option. The soft stall option could only be used with an initiation interval of one cycle. The design required this interval to be greater than one in most cases. So, the hard stall option had to be used. A small trick was used to get around the problem of the system stopping without generating the entire output and stalling with data inside the pipeline. The testbench was designed to feed redundant zeros after the test sample had been completed. This solution sent new input to the pipeline even after the end of the actual test sample and kept the pipeline running until all the useful data was sent to the output.

3.5.5 Protocol Regions

Protocol regions were also defined using an HLS directive. The HLS tool synthesizes these regions as is, without any change. These are used to perform communication.

3.5.6 Array Mapping

HLS directives were also used to map arrays to memories designed in the memory editor, for example, the coefficient memory to a RAM with a single port for reading and writing, and the delay memory to a RAM with two separate ports for simultaneous reading and writing.

3.6 Major Design Decisions in different Modules

This section mentions some of the major problems that plagued the design and what was needed to remove them.

3.6.1 Resource

The control and datapath separation was a major design decision in the resource as it ensured successful behavior of the module. Without this, the module was running into pipeline violations, i.e. the number of cycles defined in the initiation interval of the thread was not enough to allow pipelining. This also helps to make the code more readable as well as increase the modularity of the program.

The control path was further optimized as well using high-level synthesis directives.

Transition from	Transition to	Legal Transition
Constrained	Constrained	Yes
Constrained	Unconstrained	No
Unconstrained	Constrained	Yes
Unonstrained	Unonstrained	Yes

TABLE 3.3: Code Motion Rules

3.6.2 Clip Stage

The CORDIC in the clip stage had to be optimized to meet timing constraints. The logical synthesis process was resulting in a small negative slack when CORDIC was not optimized.

3.6.3 Allocator

The allocator ran into a lot of pipeline violation problems as this module needed to interact with all the clip stages and the resources and perform different operations accordingly. The design complexity increases with the increase in the number of resources and clip stages. For example, the allocator code consists of many functions that are enclosed in 'for' loops which run as many times as the number of clip stages or resources. Many of these loops are not unrolled, i.e. they need to be completed serially. An increase in the number of modules increases the number of operations that have to be performed in a fixed amount of time. These led to the revision of the coding style. The iterations of the loop had to be made independent of each other and the code had to be written while keeping the hardware in mind.

The reason for the above-mentioned problems can be attributed to loop carried dependencies(LCD). LCD affect pipelined designs as well, as multiple iterations of the loop run simultaneously in a pipeline. LCD in a pipeline can occur due to forward data flow as well as code motion rules(CMR).

LCD due to Forward data flow occurs when the use of a variable is somehow dependent on the previous definition of the variable which goes through some series of operations. This series of operations need to be scheduled within the initiation interval of the pipeline and the interval may not be enough for any valid schedule.

CMR are a set of guidelines for the tool used when finding an optimized valid schedule for the design when performing HLS. As explained before, the code can have both constrained (protocol regions or regions with latency constraints) and unconstrained regions. When searching for an optimized valid schedule, the tool might move certain parts of the code in between these regions. This is known as code motion. Table 3.3 shows which transitions are legal:

LCD due to CMR does not occur because of a dependency between the use and the definition of the variable, rather it happens because the tool is not able to bring the definition and use to within the initiation interval of the pipeline due to CMR and, thus, is not able to find a valid schedule. This happens because the use or the definition of a variable is within a control branch or inside a protocol region.

3.7 Testbench restrictions

One of the restrictions on the verification options of the tool is the communication protocol used in the design. The tool allows using the same SystemC testbench for

verification of the system(behavioral) code as well as the RTL output result of the high-level synthesis process provided the design uses a handshaking protocol. This makes the design and debugging process faster as behavioral simulations have a lot fewer details and thus take a lesser amount of time. Since the PC-CFR does not use a handshaking protocol, this design was not able to take advantage of this feature of the tool.

3.8 Logic Synthesis

Disclaimer: *The focus of this project was solely on the HLS part, the RTL synthesis flow and results could certainly be optimized and the RTL synthesis flow was used without optimization as provided and only for the purpose of comparing the results of the thesis.*

The tool also allows for performing logic synthesis in-house. The logic synthesis process has a high chance of success as the HLS process calls an RTL synthesis tool, Genus [24] by Cadence, to characterize the parts in the design, such as the area and timing. Thus the RTL generated by Stratus is expected to meet timing. This feature helps in the design process as violations are normally detected at a higher level of abstraction and the programmer does not have to go through the logical synthesis process to find out about them.

Chapter 4

Results and Analysis

This chapter presents the result of the different experiments performed for the comparison between the old and the new solution on the parameters defined previously. It also presents the analysis of these results.

4.1 Results

This section presents the results collected based on the different comparison parameters.

4.1.1 Productivity

The productivity for the implementation of PC-CFR in this thesis is compared with that of a previous implementation [12] designed in RTL by a master thesis student at KTH. This is done to get a rough measure of the effort needed in HLS versus RTL design methodology. In an attempt to achieve this, productivity is measured in terms of the number of features implemented in both of these implementations with respect to time. Since both the new and the old solutions were designed as Master Thesis projects, the time available was roughly equal. A difference in the implemented features indicates a difference in design productivity. Table 4.1 lists features and whether they are implemented in a design or not.

4.1.2 Design Space Exploration Results

DSE was performed by varying the latencies of different modules and collecting the area from logic synthesis. DSE was not performed by changing the architecture or disallowing loop unrolling and register flattening. Table 4.2 shows the different latency configurations for a few different implementations along with the gate count from logic synthesis. It should be noted that the gate count is calculated from the area of the system after excluding memory area.

4.1.3 Area Comparison

The design area of the best choice from the DSE section is compared against the previous implementation [12]. It should be noted, however, that the previous implementation was synthesized with four clip stages and eight resources (called Pulse Control Units or PCUs by the designer). In comparison, the implementation of this project has been verified and synthesized with three clip stages and sixteen resources. So, the design in this thesis has roughly twice as many resource modules as the previous implementation. However, the purpose of comparing the areas of these different implementations is not to decide which one has the least design area. Rather, it is to find out if the HLS design methodology can deliver a system that has a

Features	New Solution	Previous Master Thesis Solution
Cascaded design	Yes	Yes
Pipelined design	Yes	Yes
Time division multiplexed resources	Yes	Yes
Variable length canceling pulses	No	Yes
Parametrized Resource limit	Yes	No
Allocation from resource pool	Yes	No
Variable latency	Yes	No
Parametrized number of modules	Yes	Yes

TABLE 4.1: Features comparison between previous and new solutions

gate count comparable to the RTL implementation. Table 4.3 lists the areas, memory area percentage, and the gate count of two designs: Matteo Bernini's design and the HLS implementation. The areas do not include the memory area.

4.1.4 System Performance

System performance is measured by passing a test signal through it and measuring the PAPR of the corrected signal and the percentage EVM. The input PAPR is 6.481443. The output PAPR is compared with the expected reduction in CF which can be calculated by setting the peak power to the specified threshold. Readings are taken after changing the values of different input parameters. These parameters are peak search window length, threshold and limits on the number of resources allowed simultaneously per clip stage. The different configurations and their results are listed in table 4.4. The extreme values have been highlighted.

The results are illustrated using two scatter plots; output PAPR against EVM and output PAPR against expected output PAPR. The plot in figure 4.1 is useful to gauge which configuration performs the best on the test signal in terms of output PAPR with respect to percentage signal distortion. The black line in the plot represents the PAPR value of the original signal. The system performance is highly dependent on the input signal. This plot can be used to calibrate the system for optimal results. Plot in figure 4.2 shows the output PAPR of each configuration against the expected output PAPR. This plot shows a trend regarding setting the threshold and the success rate of meeting the expected PAPR reduction. Figure 4.3 and 4.4 show the CCDF curves for the best and the worst configurations (configuration number 4 and 10 respectively) listed and highlighted in table 4.4. Figure 4.5 and 4.6 show the corrected signals for the best and the worst configurations. Finally, figures 4.7 and 4.8 show the utilization of resources by the three clip stages throughout the duration of the signal. The y-axis represents the number of simultaneously used resources or

	Imp A	Imp B	Imp C	Imp D
Peak detector thread latency	6	6	6	6
Peak canceling thread latency	6	6	6	6
Allocating thread latency	10	22	10	10
Cancellation signal adder thread latency	7	7	7	7
Resource thread latency	26	26	18	38
Total	55	67	47	67
Gate count	302.9k	316.3k	305.7k	314.2k

TABLE 4.2: Design Space Exploration Results

Pulse Control Units (PCUs) and the x-axis represents the input signal element index per clip stage.

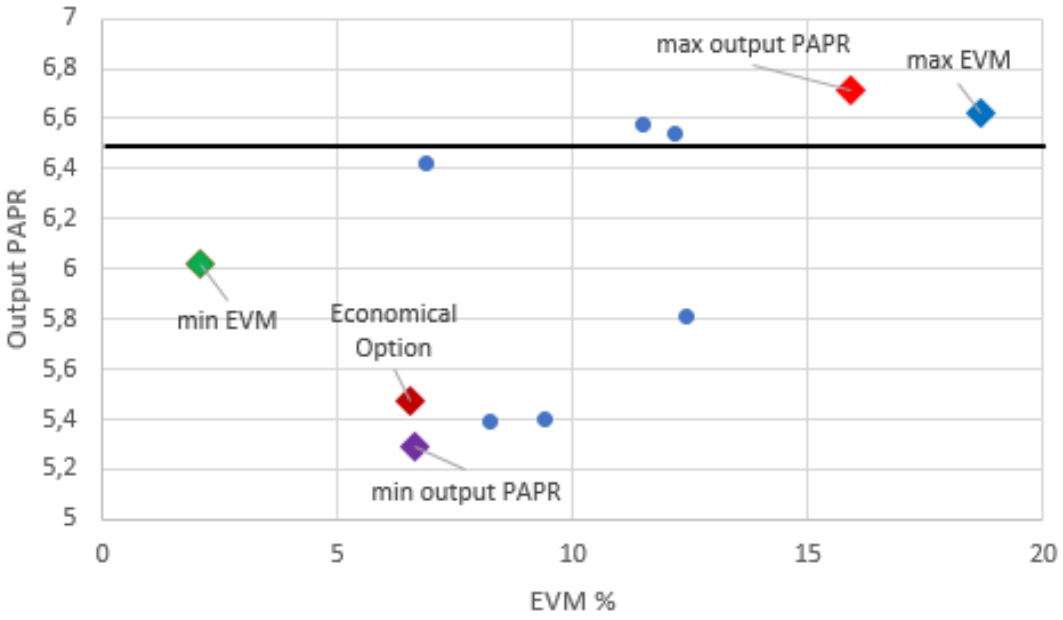


FIGURE 4.1: PAPR vs EVM for different configurations

	Previous Master Thesis Solution (4 CS, 8 PCU)	New Solution (3 CS, 16 PCU)
Logic area (square micrometer)	17774	43973
Memory area (percentage)	50	60
Gate count	122k	302.9k

TABLE 4.3: Area comparison between previous and new solutions

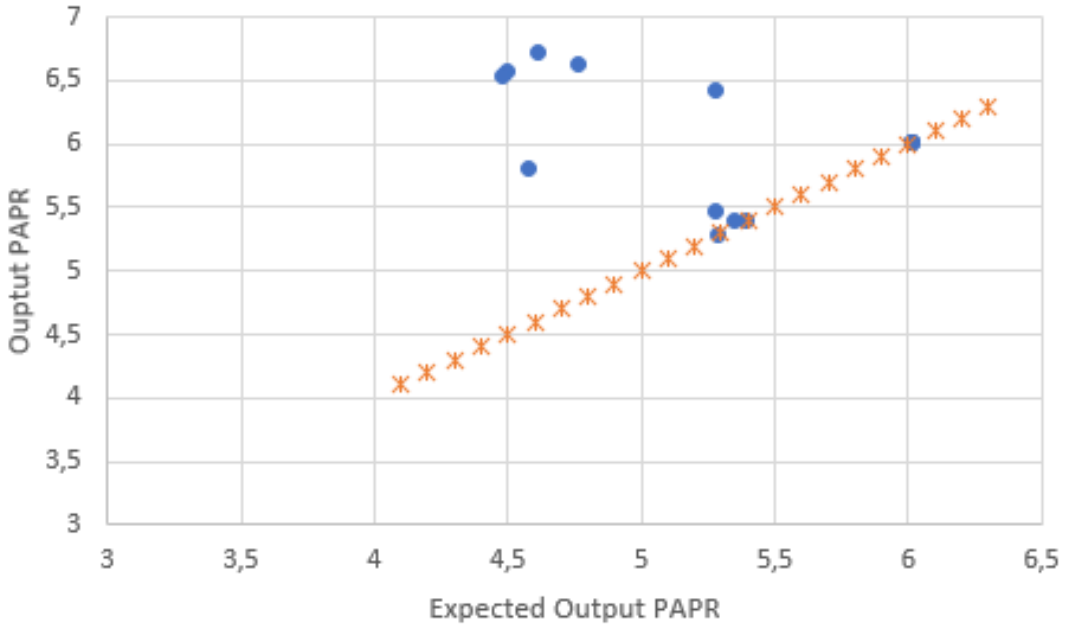


FIGURE 4.2: Actual PAPR vs expected PAPR

4.2 Analysis

This section presents some analysis on the results collected and written in section 4.1. The analysis is divided into separate segments in accordance with the result categories. These are the design time, DSE, area comparison and system performance.

4.2.1 Design Time

The design time was measured by comparing the number of features implemented in the two solutions. As table 4.1 shows, both the previous and the new solution have a cascaded, pipelined design and incorporate time division multiplexing of the resources to get more performance per unit area. The previous solution does, however, have the option of changing the cancellation pulse length depending on the size of the detected peak, whereas the new solution does not. On the other hand, the resources in the new solution are available as a pool and any clip stage can be allocated any one of the resource as long as that clip stage has not reached its allowed

No.	Threshold	PSW length	Resource limits per CS	Expected PAPR	EVM percentage	outputPAPR
1	41185	32	6, 4, 2	6.016474	2.096677	6.02075
2	41185	32	4, 2, 1	6.016462	2.096373	6.020254
3	41185	64	4, 2, 1	6.016376	2.094246	6.020167
4	41185	128	4, 2, 1	6.015155	2.059396	6.018946
5	37066	32	6, 4, 2	5.391947	9.418327	5.396189
6	37066	32	4, 2, 1	5.3483	8.256044	5.387556
7	37066	64	6, 4, 2	5.286837	6.649069	5.290881
8	37066	64	4, 2, 1	5.282105	6.55129	5.476721
9	37066	128	4, 2, 1	5.274904	6.859882	6.425217
10	32984	32	6, 4, 2	4.761797	18.651816	6.626374
11	32984	32	4, 2, 1	4.613527	15.889209	6.714934
12	32984	64	6, 4, 2	4.581006	12.41755	5.811003
13	32984	64	4, 2, 1	4.499202	11.49382	6.572774
14	32984	128	4, 2, 1	4.475888	12.176626	6.537019

TABLE 4.4: PC-CFR configurations and results

limit of simultaneous resources,i.e. the resources are not dedicated to any single clip stage. In the previous solution, each clip stage could only access a sub-pool of the resources and could select any resource within that pool only. The new method was a requirement of the design.

Apart from this, the new solution incorporates the limit on the number of resources allowed to be allocated per clip stage as an input to the system in the form of a signal. This allows user to change this configuration parameter of the system after resetting the system. The previous implementation has hard coded values instead. The new solution also allows for changing the latency of different modules in the system, and thereby, changing the latency of the entire system by just modifying the values of some macros defined in a header file. HLS and logic synthesis has to be run again after changing the latency of the system. Finally, both the solutions have parametrized number of clip stages and resources in the design. However, the new design has been verified up till 3 clip stages and 16 resources, whereas the previous design has been verified to a maximum of 4 clip stages and 8 resources.

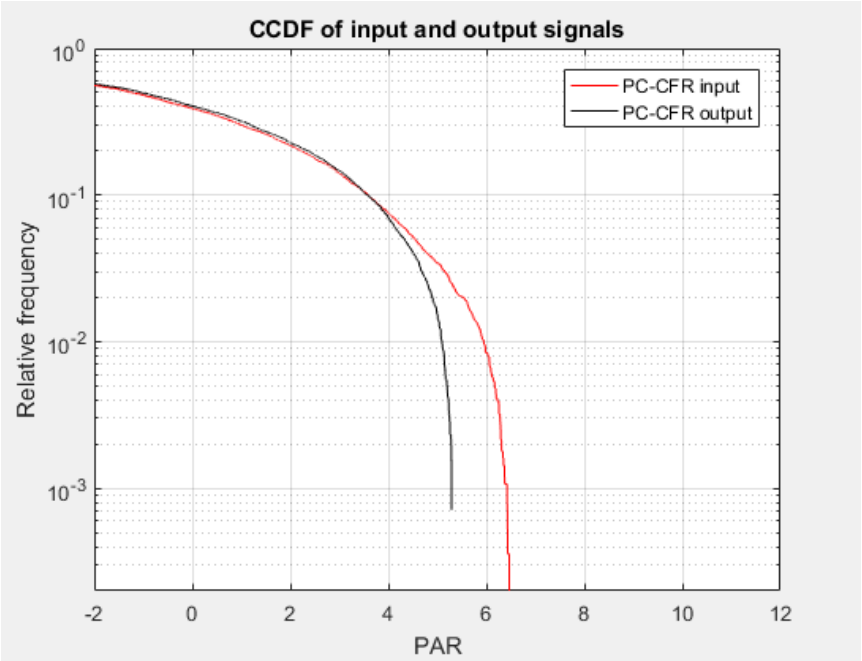


FIGURE 4.3: CCDF graph of a well performing configuration

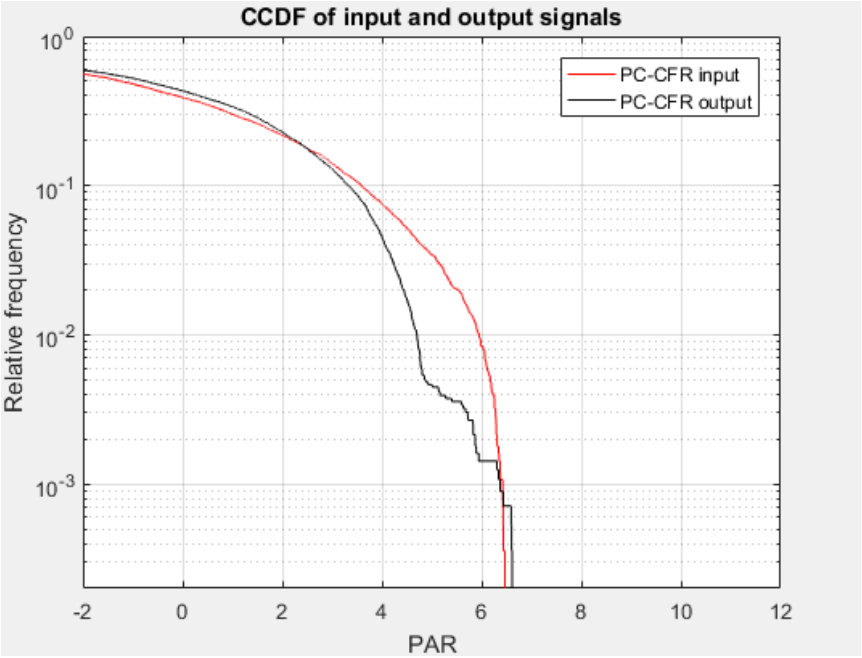


FIGURE 4.4: CCDF graph of a poorly performing configuration

After the comparison of the features in both solutions, it can be seen that the new solution includes almost all of the features of the previous solution, except for the variable cancellation pulse length. This feature would require the code for the clip stage, the allocator and the resource to be modified slightly which would add to the design time. Conversely, implementing pipelining and variable latency options would take up a whole lot of time, whereas doing this in HLS can be very time efficient. So, it can be analyzed that coding in a high level language and then letting

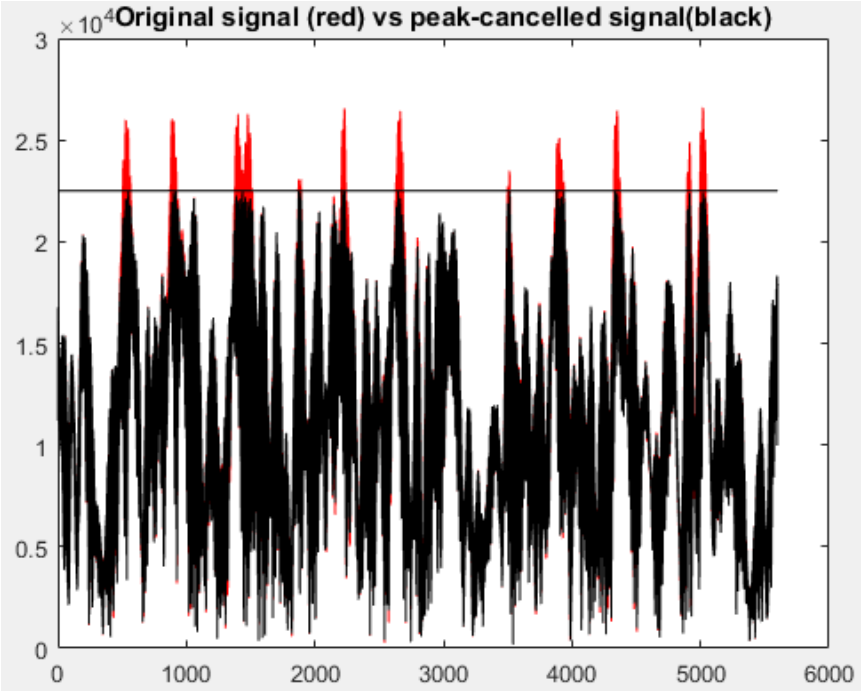


FIGURE 4.5: Original and corrected signal of a well performing configuration

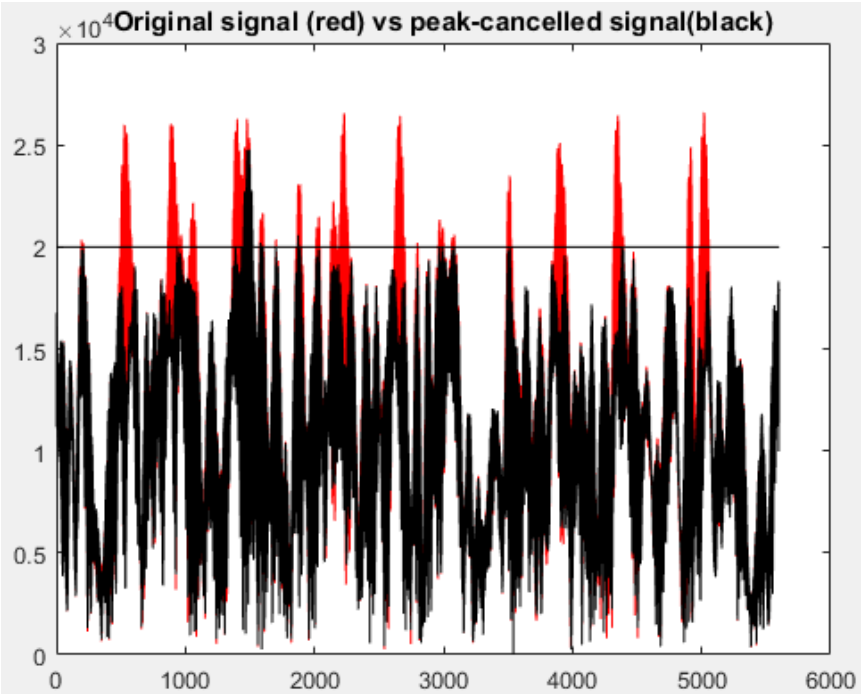


FIGURE 4.6: Original and corrected signal of a poorly performing configuration

an HLS tool perform the synthesis can be quite rewarding in terms of time.

It should be added here that the design time was also affected by the learning curve of the HLS tool and HLS design methodology itself. The HLS tool requires

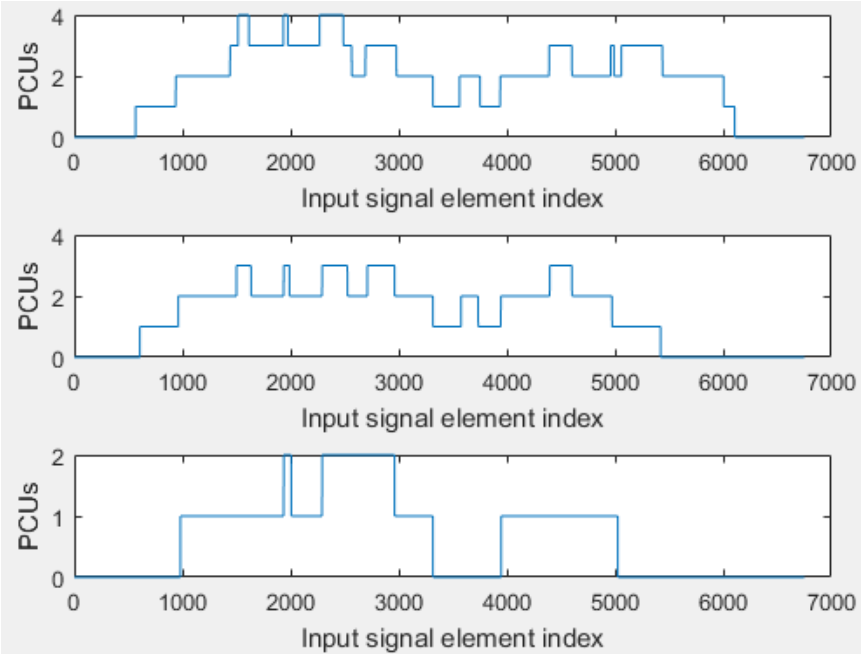


FIGURE 4.7: Resource utilization of a well performing configuration

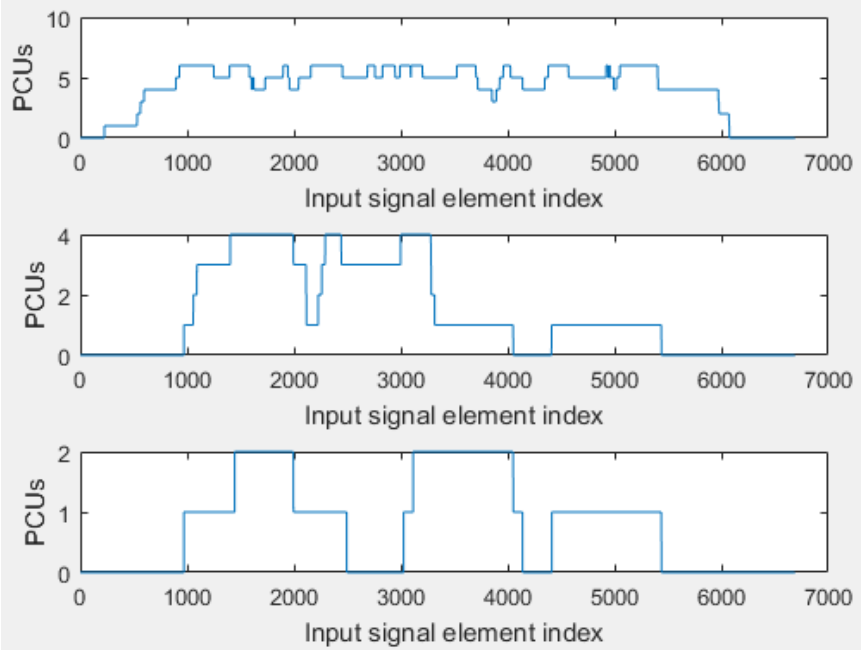


FIGURE 4.8: Resource utilization of a poorly performing configuration

the user to observe a certain coding style which is necessary for the tool to perform optimally and deliver the best possible results. Some of the rules to the coding style can be learnt from having knowledge of parallelism in software which is necessary to take advantage of the underlying hardware, like LCD, or high level synthesis rules in general, like separating communication from computation. Other rules, however, needs the user to study the documentation in detail and experiment with the tool,

for example, CMR and forward scheduling, how and when to use HLS directives etc. But the cost in terms of time was worth it as it yielded satisfactory results in the end.

4.2.2 Design Space Exploration

DSE was performed by varying the latency of the modules and running HLS. As the results in table 4.2 show: implementation 'A' has the second highest latency and the smallest area footprint. The change in area was not very significant in this design with a difference of approximately fourteen thousand gates between the lowest and highest area footprints. DSE is highly dependent on the coding style of the design. This design could not benefit very much from the DSE features because of the rather fixed implementation in terms of latencies to cater to the communication protocol defined in 3.4.4. As said before, the communication protocol requires the data to be perfectly synchronized as their is no blocking reads or writes, nor is there any handshaking mechanism. The HLS tool can vary latency and area by deciding between unrolling of loops and rolling of loops, flattening of registers or implementing them as memories, as well as varying schedules to meet different latency constraints. The current design has fixed constraints to unroll loops and flatten register where needed which cannot be modified by the tool and, thus, it cannot exploit these features. So when the latency constraint is set to a certain number, the HLS tool tries to provide a schedule where all the other constraints are also met which restricts the number of possible schedules. If the latency number is too big and the actual schedule is completed in lesser number of cycles, the tool adds 'no-operations' (NOP) in the end of the schedule to meet the latency constraint. NOP are instructions that do nothing and are added simply to meet the latency constraint. When looking at the results, we can see that implementations B and D have the highest latencies as well as the highest area footprint. This seems slightly counter intuitive. One would expect the area footprint to decrease as the latency increases since this allow the tool to reuse some hardware that was not possible before due to, possibly, a tight schedule. This is suspected to be because of the no-operations added in the end.

Increasing or decreasing the latency from fifty-five (implementation A) resulted in an increase in the area footprint. So, implementation 'A' was selected as the best implementation and used further in the area comparison with other solutions as well as system performance.

4.2.3 Area Comparison

The area of three solutions; the previous solution by Matteo Bernini designed in his master thesis, a configuration of the Ericsson Turbo Clipping solution, and the solution implemented in this thesis, is compared and written in table 4.3. As can be seen, Both the previous master thesis solution and the new solution have much smaller area footprint than the Turbo Clipping solution. The difference in their order of magnitude is 10.

When compared with each other, the ratio between the area footprint of the previous and the new solution is approximately 1:(5/2). However, the previous solution has four clip stages and eight resources, whereas the new solution has three clip stages and sixteen resources which means that the new solution has twice as many resources. When mapped to an equivalent implementation with equal number of modules, the difference in gate count between the two solution narrows

down even further as the area would decrease linearly with decreasing number of resources.

The percentage of the memory area is lesser in the previous solution as compared to the new one. However, the new solution is currently using coefficient memories in the resources that are almost twice as large as required. Due to the unavailability of smaller memory sizes, this was the best fit possible. If the current design can be changed so that the size of the data structure holding complex numbers can be decreased by one bit, a much smaller and better fitting memory is available and can be used to decrease the memory size. This is estimated to have a big impact on the percentage of the memory area in the design.

The comparison shows that the HLS tool was able to yield a hardware with area footprint comparable to the design in the previous master thesis.

4.2.4 System Performance

In this section, the system performance of the new solution is tested on a test input signal. The performance of the system is dependent on the input signal, the peak search window length, and the threshold. The results presented in 4.1.4, show what values of the input configurations output the best results. The output results are characterized using PAPR reduction versus output EVM and actual PAPR reduction versus expected PAPR reduction. These results help in tuning the system according to the input signal.

In the PAPR vs EVM plot 4.1, a subset of the data points from the table 4.4 are highlighted to show extreme and economical options. The configuration labeled as 'minimum EVM' (Green, config number 4) results in the smallest EVM value but does not perform very well with respect to PAPR reduction. The configurations labeled as 'maximum output PAPR' (Red, config number 11) and 'maximum EVM' (blue, config number 10) not only give the highest EVM values but also yield higher PAPR values than the PAPR of the original signal. The configuration labeled as 'minimum PAPR output' (Purple, config number 7) returns the lowest value of output PAPR and has on of the lowest EVM values. Another good option would be the configuration labeled as 'economical option' (Brown, config number 8). This configuration does not reduce the output PAPR as much as the minimum PAPR configuration but still provides a reasonable amount of reduction with lesser EVM levels.

In the output PAPR vs expected PAPR plot 4.2, the straight line (orange) is the ideal PAPR reduction line. This line depicts that the output PAPR is equal to the expected PAPR. The blue dots show the actual results of the different configurations listed in table 4.4. We can see that there are five configurations which yield results close to the expected ones. One of them (top right) can be discarded since it does not give sufficient PAPR reduction.

After analyzing the above plots along with the data in the configuration table 4.4, which holds the input parameters for all the configurations, it can be seen that the worst performing configurations have the lowest set threshold. This means that if the threshold is set too low, the output EVM can get rather high. This also leads to a bigger difference in real and expected output PAPR. These can be attributed to high peak density, overcompensation, and peak leaks. So, the best and worst configurations are chosen to be number 7 and 10 respectively in the table. The best configuration uses a threshold of 37066, a PSW length of 64 samples and resource limits per clip stage as 6 for the first clip stage, 4 for the second and 2 for the third.

The worst config uses a threshold of 32948, PSW length of 32 and limits of 6, 4, and 2.

Figures 4.3 and 4.4 show the CCDF graphs of the original and corrected signal for the best and the worst configurations. The graph for the best configuration saturates at 5.29 instead of 6.48 of the original signal whereas it saturates at 6.7, higher than that for the original signal, in the case of the worst configuration. When comparing the corrected signals of the both configurations in figures 4.5 and 4.6, it can be seen that the best configuration was able to cancel the peaks without overcompensating. It also did not leak any peaks. On the other hand, the worst configuration did a poor job of canceling the peaks and overcompensated beyond the threshold for a fair portion of the signal as well as leaked a major peak. When looking at the utilization of resources in both configurations in figures 4.7 and 4.8, the distribution of peaks among the different clip stages for the better configuration looks a lot smoother. The first and second clip stages actually never had to use all six resources of its allowed quota. On the contrary, the first clip stage for the worse configuration seems to be using its entire allocated quota of resources for a substantial amount of time. This shows that the peak density is, in fact, quite high and would, therefore, lead to peak leaks. It also means that the cancellation pulses of neighboring peaks would interfere with each other.

Chapter 5

Conclusion and Discussion

This chapter concludes the project and discusses the outcome based on the results and the experience gathered throughout the thesis. It also presents some features that could be added or improved in the future. Finally, it remarks on the sustainability, ethics, and social aspects of the thesis work.

5.1 Conclusion

From the analysis of the design time via comparing features of new and previous designs, we can say that the HLS tool, definitely improves productivity. Even more important of all, the tool introduces an element of reusability with the flexible nature of the coding style as well as the simulation configurations written in the project Tcl file. The user can have multiple implementations of the system with minimal effort by just changing a few options in each configuration. Even if the design has to be changed for an Engineering Change Order (ECO), which might require some partial change in the algorithm, Stratus allows the user to turn on a mode that can generate a new RTL code with minimal differences from the previous RTL code and generate a netlist or a polygon patch. These features make HLS dominate over the method where the designer has to write RTL code where flexibility and reusability are very limited.

The area comparison between different solutions and the DSE showed that the HLS design methodology performed almost the same as the RTL design methodology in this project. It is expected that the HLS tool can give even better implementation in terms of area if the design is optimized or altered to take advantage of other design space exploration parameters.

Another big improvement in the design flow introduced by Stratus HLS is the reuse of the same testbench for functional as well as RTL verification of the design. Verification at the SystemC/TLM level is much easier and since it is converted to RTL by the HLS tool, the verification applies to the RTL as well. Using the same testbench allows the developer to work on the design flow instead of writing extensive testbenches for each level. It also makes the transition from one layer of abstraction to the next a lot more seamless. However, the current design was not able to take advantage of this feature to a great extent as the reuse of the testbench requires that the system has some sort of handshaking protocol. This is a limitation imposed by the nature of the communication protocol which was necessary to cater to the constant stream of input to the system.

It should be added that extensive verification was not performed, for example, finding functional coverage, using assertions to check for failing conditions and the power of the testbenches were not explored. Even though UVM has been making its way into SystemC, there are still limitations that need to be removed to come closer to the verification power of languages like System Verilog, for example, support for

constrained random generator is still immature. Thus, a comment about this flow replacing the verification flow using languages like System Verilog cannot be made.

To conclude and summarize some positive points experienced through this project are:

- Usage of SystemC as the system design language allows for an object oriented and behavioral approach while still having enough capability to emulate hardware due to in-built support for bit-exact data types, hardware concurrency, and timing. TLM interfaces allow for fast behavioral simulations.
- An efficient debugging framework that allows to isolate functional or bit accuracy verification from protocol or cycle accuracy verification.
- Getting timing information after the HLS run which takes minutes in contrast with logic synthesis which takes hours to complete. Making changes to meet timing is an iterative process and this reduces the amount of time expended in this immensely.
- Easy and simple way to pipeline the design by the use of a single HLS directive. This improves the utilization of the hardware and also allows for keeping a constant throughput even if the design latency is kept flexible for different architectures needed for DSE. The designer just needs to work on the behavioral aspects of the design and does not need to concern with the implementation details of a pipeline.
- Designing and generating communication interfaces by characterizing the properties in a GUI.
- Integrated memory compiler that allows for adding vendor memories in the design.
- HLS directives to control architecture and meet constraints and their ability to explore design space.
- High productivity and reusability benefits as well as improved quality of results.

The project also led to some neutral observations that are listed as follows:

- There is a debate on whether software engineers without having a knowledge of hardware can also use HLS tools or not. After working with the tool, I concluded that the developer still needs a good knowledge of hardware architecture and the knowledge of what coding style translates to what type of hardware and how to take advantage of parallelism. This knowledge is invaluable to optimize the synthesized hardware. So, this still is closer to a hardware engineer's paradigm.
- There is a learning curve associated with the tool. The tool has a lot of documentation to inform developers about good practices and it takes time to go over them and, more importantly, be well acquainted with their use. Having good knowledge about the underlying scheduler of the tool takes a lot of time and experience. But the time investment can be worth it, keeping in view the benefits it provides.

- The tool also requires a certain coding style to get good results. Since the coding style is very modular, intuitive, and object oriented, developing becomes a lot simpler once the user is accustomed to it.
- The generalized testbench works only if the communication interface has some handshaking mechanism which makes the system immune to latency changes.
- The tool currently does not have the ability to generate a single memory from an array of a customized object. For example, an array of complex data type, that has a real and an imaginary component, will be generated as two separate memories; one for the real component and one for the imaginary. If a single memory is desired, the concatenation has to be performed manually.

5.2 Future work

This section discusses some improvements that can be added to this design in the future.

5.2.1 Improving Algorithm to get better DSE results

The design algorithm could be revisited to introduce a bit more flexibility to take advantage of loop unrolling and register flattening HLS directives. This would allow the tool to choose between implementations that vary between low latency/large area and high latency/small area values. This would also require a change in the communication interface between modules and make the system immune to latency changes.

One region that can be improved is that both the allocator and the resource keep track of the generation of cancellation pulses. The allocator needs this to update the 'available resource vector' whereas the resource needs this to know when to stop the generation of the cancellation pulse. This could be changed so that the resource would inform the allocator when it has finished the generation process.

5.2.2 Decreasing memory size

The current design has a delay memory per clip stage and a memory to hold filter coefficients per resource. As said before, vendor memories are being used in the design. The size of the vendor memories does not match the array sizes in the design as the selection of vendor memory sizes availabile for this project are limited. The current size of the coefficient vendor memory is very large as compared to the actual size of the array and thus adds to the area quite a bit. Since ordering a new memory with a more appropriate size is not a possibility, a different solution needs to be incorporated. One solution would be to decrease the size of the polar data type which is stored in the coefficient memory. In fact, decreasing the size by one bit would allow the use of another available memory that would decrease the coefficient memory area to less than half of the current area per resource. Since the design has four resources, the reduction factor will be multiplied with four. This would have a big impact on the area of the total design and will decrease the memory area percentage.

5.2.3 Increasing Logic Synthesis Effort

The main focus of this thesis was to perform HLS. However, logic synthesis was required to compare the final results with other solutions. The logic synthesis was

performed by using the in-built option of Stratus HLS but it was not optimized to get the best possible results and was used as provided. The next step could be to work on setting better configurations to improve logic synthesis results.

5.2.4 Variable cancellation pulse lengths

The system currently generates cancellation pulses of a constant length. The design can be changed to incorporate varying pulse lengths depending on the size of the peak. The smaller the peak, the smaller the cancellation pulse length. This would require some changes in the delay mechanisms of the system. Having varying pulse lengths would allow minimizing the effects of peak regrowth due to constructive interference of different cancellation pulses. However, when limiting the cancellation pulse length, the ending points will have to be chosen carefully to avoid distortion as discussed in section 2.4.1.

5.2.5 Priority based Preemptive scheduling

Peak leaks can occur if the threshold is set so that the peak density is high even when having a reasonable number of resources. However, it could be made a rule to categorize peaks according to priorities based on their magnitude and clip stage. These priorities could not only be used to decide which peak gets a resource, but also to preempt the cancellation pulse generation of a previous peak, i.e. to cut short the generation of the canceling signal for the peak with a lower priority, and start the generation for the peak with higher priority. This could improve the PAPR reduction of the system. An approach involving preemption of cancellation signals is also discussed in [13].

5.2.6 Power consumption investigation

The current project does not perform any kind of power consumption comparison between the old and the new implementation. In the future, power saving features could be added to the design to make the system power efficient, e.g. clock gating could be investigated to reduce switching activity which is a major contributor to the power cost. The resulting system would give valuable information about the quality of results achieved from HLS as compared to the system produced using RTL design methodology.

5.3 Sustainability, Ethics, and Social Aspects

This project is aimed towards increasing the power efficiency that is affected in the base stations due to the high crest factor of signals as well as investigating a different ASIC design methodology that claims higher productivity and reusability. Achieving higher power efficiency would decrease the power cost which will impact the economic value positively. This will contribute to making more efficient networks going into the fifth generation, which boasts faster data rates that might make new use cases possible, like remote surgeries etc, and this will impact the society in a positive way. This work does not present any unethical aspects. However, where faster networks with lower latency pave the way to new opportunities in the health sector like remote surgery, or transportation sector like autonomous cars, there is also a debate on the added risks that follow any high stake application that concerns the human life. A concern about the possibility of hacking has been shown.

Comparison of design methodologies to investigate design productivity is vital as a large number of resources are currently used to achieve the end result and it still does not offer a high degree of reusability. Investigating HLS to tackle these problems is necessary to improve the resource utilization. This would impact the economic cost in the industry positively. This would also allow for more complex designs to be made which would, in turn, increase the speed of technological advancements.

The outcome of this thesis work is important, primarily, for the ASIC developers and verification engineers as it reduces the design effort, increases productivity, and allows for the creation of more complex systems. The consumers are the secondary stakeholders as this would open up new possibilities with more powerful and faster hardware for various applications.

Bibliography

[1] I. Poole, "Radio-Electronics.com, Resources and analysis for electronics engineers." 2016. [Online]. Available: http://www.radio-electronics.com/

[2] Altera, "Crest Factor Reduction Application Note," *Application Note 396*, no. June, pp. 1–28, 2007.

[3] Electronic Design, "Error vector magnitude, EVM, BER, modulation quality." [Online]. Available: http://www.electronicdesign.com/engineering-essentials/understanding-error-vector-magnitude

[4] A. E. Hemphill, S. Summerfield, G. Wang, and D. Hawke, "Peak Cancellation Crest Factor Reduction Reference Design," *Xilinx Application Note*, vol. 1033, pp. 1–32, 2007.

[5] P. Kaur and R. Singh, "Complementary Cumulative Distribution Function for Performance Analysis of OFDM Signals," *IOSR Journal of Electronics and Communication Engineering (IOSRJECE)*, vol. 2, no. 5, pp. 5–7, 2012.

[6] Adewuyi O.S1, "Performance Analysis of OFDM in Combating Multipath Fading," *IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)*, vol. 6, no. 2, pp. 99–107, 2013. [Online]. Available: http://www.iosrjournals.org/iosr-jece/pages/v6i2.html

[7] M. C. P. Paredes and M. J. F. García, "The problem of peak-to-average power ratio in OFDM systems," *CoRR*, vol. abs/1503.08271, 2015. [Online]. Available: http://arxiv.org/abs/1503.08271

[8] S. H. Han and J. H. Lee, "An overview of peak-to-average power ratio reduction techniques for multicarrier transmission," *IEEE Wireless Communications*, vol. 12, no. 2, pp. 56–65, April 2005. doi: 10.1109/MWC.2005.1421929

[9] Z. S. H. Al-Hashmi, *An Overview : Peak to Average Power Ratio (PAPR) in OFDM system using some new PAPR techniques (with matlab code)*, University of Baghdad, College of Engineering Electronic and Communications Engineering Department. ISBN 9781329219076. [Online]. Available: http://www.lulu.com/shop/zainab-saad-and-zainab-saad-hadi-al-hashemi/peak-to-average-power-ratio/ebook/product-22217907.html

[10] O. Väänänen, "Digital Modulators With Crest Factor Reduction Techniques," Ph.D. dissertation, Helsinki University of Technology. ISBN 9512280817 2006. [Online]. Available: https://pdfs.semanticscholar.org/5eb9/edd7135512413cc8c5760066e2894d8a559d.pdf

[11] P.-C. I. C. U. guide Lattice semiconductor, "IPUG109 - Peak Cancellation Crest Factor Reduction IP Core User's Guide," Lattice Semiconductor, Tech. Rep. May, 2014.

[12] M. Bernini, "An efficient Hardware implementation of the Peak Cancellation Crest Factor Reduction Algorithm," Master's thesis, KTH Royal Institute of Technology, 2016. [Online]. Available: urn:nbn:se:kth:diva-206079

[13] J. Song and H. Ochiai, "A low-complexity peak cancellation scheme and its FPGA implementation for peak-to-average power ratio reduction," *EURASIP Journal on Wireless Communications and Networking*, 2015. doi: 10.1186/s13638-015-0319-0

[14] P. Coussy and A. Morawiec, Eds., *High-Level Synthesis, from algorithm to digital Circuit, pp 8-9, 13-14*. Springer. ISBN 9781402085871

[15] G. E. Moore, "Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff." *IEEE Solid-State Circuits Society Newsletter*, vol. 11, no. 5, pp. 33–35, Sept 2006. doi: 10.1109/N-SSC.2006.4785860

[16] P. Coussy, D. D. Gajski, M. Meredith, and A. Takach, "An introduction to high-level synthesis," *IEEE Design Test of Computers*, vol. 26, no. 4, pp. 8–17, July 2009. doi: 10.1109/MDT.2009.69

[17] Cadence, "Stratus High-Level Synthesis." [Online]. Available: https://www.cadence.com/content/cadence-www/global/en_US/home/tools/digital-design-and-signoff/synthesis/stratus-high-level-synthesis.html

[18] J. Bhasker, *A VHDL Primer*, 3rd ed. Prentice Hall PTR, 1999. ISBN 9780130965752. [Online]. Available: https://www.amazon.com/VHDL-Primer-Jayaram-Bhasker/dp/0130965758

[19] "Tcl Developer Site." [Online]. Available: https://www.tcl.tk/

[20] "SystemC TLM-2.0." [Online]. Available: https://www.doulos.com/knowhow/systemc/tlm2/

[21] J. E. Volder, "The cordic trigonometric computing technique," *IRE Transactions on Electronic Computers*, vol. EC-8, no. 3, pp. 330–334, Sept 1959. doi: 10.1109/TEC.1959.5222693

[22] F. Wahid, *Digital Design with RTL Design, VHDL, and Verilog*, 2nd ed. John wiley and sons, 2010. ISBN 978-0-470-53108-2

[23] M. D. C. M. Morris lora, *Digital Design*, 4th ed. Prentice Hall, 2006. ISBN 978-0-13-198924-5

[24] Cadence, "Genus Synthesis Solution." [Online]. Available: https://www.cadence.com/content/cadence-www/global/en_US/home/tools/digital-design-and-signoff/synthesis/genus-synthesis-solution.html

TRITA TRITA-ICT-EX-2017:171