NTNU

Norwegian University of
Science and Technology

Applicability of the Residue Number System in a Radio Receive Chain

Eivind Fylkesnes

Master of Science in Electronics
Submission date: June 2016
Supervisor: Kjetil Svarstad, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem Description

Explore the possibilities to use non standard numeral systems in a radio receive chain. (RNS, etc)

Abstract

This thesis explores the residue number system and how it can be used in a radio receiver. In this number system the numbers are divided up into several residues. Each of these residues is associated with one particular modulus. The main advantage of this number system is that, during arithmetic operations like addition and multiplication, the calculation for each residue can be done in parallel, independent of one another. This advantage can be utilised to improve the performance of several of the operations done in a radio receiver.

In this thesis the operations FIR filtering, both using real and complex coefficients, and scaling is implemented using the residue number system for scaling and real FIR filter and the quadratic residue number system for the complex filter. In addition are RNS to binary and binary to RNS converters implemented. Both the RNS implementations of all these operations and their corresponding binary implementation are run through simulations to get data on their power consumption and area usage. The simulations are done for ASIC using a 55 nm technology library.

The results illustrate that the chosen implementation of FIR filters, both real and complex, does not provide an advantage over the binary implementation unless the dynamic ranges required in the filter are very large. For filters covering dynamic ranges above 32-bit, the area per tap is less for the RNS filter than for the binary. The power consumption per tap is assumed to be lower for filters with dynamic ranges above 24-bit. The overhead by conversion increases as the dynamic ranges increases and thus counteracts the increased gain per tap in the same situation. The overhead by scaling is shown to be minimal, in therms of both area usage and power consumption, compared to the other components implemented in this thesis.

Sammendrag

Denne oppgaven undersøker residytallsystemet og hvordan det kan brukes i en radiomottaker. I dette tallsystemet er numrene delt opp i flere residyer. Hver av disse residyene er assosiert med en bestemt modulus. Den største fordelen ved dette tallsystemet er at, for aritmetiske operasjoner som addisjon og multiplikasjon, kan kalkulasjonen for hver residy bli gjort i parallel, uavhengig av hverandre. Denne fordelen kan bli benyttet til å forbedre ytelsen til flere av operasjonene brukt i en radiomottaker.

I denne oppgaven blir operasjonen FIR filtrering, både ved bruk av reelle og komplekse koeffisienter, og skalering implementert ved bruk av residytallsystemet for skalering og reelle FIR filter og det kvadratiske residytallsystemet for komplexe filtre. I tillegg er RNS til binær og binær til RNS konverterere implementer. Både RNS implementasjonene av alle disse operasjonene og deres tilsvarende binære implementasjoner er kjørt gjennom simuleringer for å få data på deres effektforbruk og arealbruk. Simuleringene er gjort for ASIC ved bruk av et 55 nm teknologibibliotek.

Resultatene viser at den valgte implementasjonen for FIR filtre, både reelle og komplekse, gir ingen fordel over den binære implementasjonen med mindre det påkrevde dynamiske område er veldig høy. For filtre, som dekker et dynamisk område over 32-bit, er arealet per filter tap er mindre for RNS filtre enn for de binære. Effekforbruken per tap er antatt å være mindre for filtre med et dynamisk område på over 24-bit. Ekstrakostnadden for konverteringen øker når det dynamiske området øker. Dette motvirker den økte gevinsten per tap i den samme situasjonen. Ekstrakostnadden til skaleringen har vist seg å være minimal, både når det gjelder effektforbruk og arealforbruk, sammenlignet med de andre komponentene presentert i denne oppgaven.

Preface

I want to thank my supervisors professor Kjetil Svarstad and Ingil Sundsbø for help and guidance. I also want to thank Nordic Semiconductor providing me with office space and the tools I used in this thesis.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Goals 2

1.3 Approach 2

1.4 Constraints 3

1.5 Contribution 4

1.6 Report Structure 4

2 Theory and Background **5**

2.1 RNS - Residue Number System 6

2.2 RNS arithmetic 8

2.2.1 Scaling 9

2.2.2 Base Extension 10

2.2.3 Signed Scaling 13

2.3 Quadratic RNS 15

2.4 Input and Output Conversion 18

2.4.1 Input Conversion 18

2.4.2 Output Conversion 19

2.5 DSP in RNS 22

2.6 Radios using RNS 24

3 Implementation **27**

3.1 FIR filter 27

3.2 Input Converter 30

3.3 Output converter 32

3.4 Scaling 34

3.5 Complex Filter 38
3.6 RNS to QRNS Converter 39
3.7 QRNS to RNS converter 41

4 Simulations and Results 43
4.1 Simulation Setup 43
4.2 Simulation of FIR Filters 45
4.2.1 16-bit FIR Filter 46
4.2.2 24-bit FIR Filter 55
4.2.3 32-bit FIR filter 58
4.2.4 12-bit FIR filter 61
4.3 Simulation of Quadratic FIR Filter 63
4.4 Simulation of Scaler 66

5 Discussion 69
5.1 Validity of Results 69
5.1.1 Transferability to Other Implementation Methods 70
5.1.2 Estimation Errors 71
5.2 Findings 72
5.2.1 FIR filters 72
5.2.2 Quadratic FIR Filters 76
5.2.3 Scaler 78
5.3 Cases where RNS is beneficial 78

6 Conclusion 81
6.1 Future Work 82

Bibliography 84

List of Figures

2.1 Block diagram for base extension algorithm 12
2.2 Converter structure. Taken from [Jenkins and Leon (1977)] 20
2.3 Hybrid FIR filter for a digit of the residue number. Taken from[Ibrahim (1994)] 23
2.4 One cell of the filter in figure 2.3. Taken from[Ibrahim (1994)] 24

3.1 Block diagram of a FIR filter designed in the RNS domain 29
3.2 Block diagram of general input converter 31
3.3 Single cell of a input converter 32
3.4 Block diagram of general output converter 33
3.5 A scaled signal occupies a smaller portion of the dynamical area 35
3.6 Block diagram of implemented scaler using a base extender to get the scaled value of the final residue 37
3.7 Block diagram for a complex filter using QRNS 39
3.8 Block diagram RNS to QRNS converter 40

4.1 Simulation setup 44
4.2 Setup of RNS FIR filter simulation 46
4.3 Setup of the reference FIR filter simulation 47
4.4 Area 16-bit FIR filter 47
4.5 Power 16-bit FIR filter 49
4.6 2nd order regression of power consumption 52
4.7 Graph fitted to the power consumption of the RNS filter using the presented model 52
4.8 Regression of power consumption of reference filter 54
4.9 Area 24-bit FIR filter 55

4.10 Power 24-bit FIR filter 56
4.11 Graph fitted to the power consumption of the RNS filter using the presented model 57
4.12 Area 32-bit FIR filter 59
4.13 Power 32-bit FIR filter 60
4.14 Graph fitted to the power consumption of the RNS filter using the model presented in this thesis 60
4.15 Area 12-bit FIR filter 61
4.16 Power 12-bit FIR filter 62
4.17 Graph fitted to the power consumption of the RNS filter using the model presented in this thesis 63
4.18 Simulation setup of the quadratic FIR filter 64
4.19 Area usage of the quadratic FIR filter 64
4.20 Power consumption of the quadratic FIR filter 66
4.21 Area usage of the FIR filter with and without a scaler 67
4.22 Power consumption of the FIR filter with and without a scaler 67

5.1 Linear component of the area usage model 73
5.2 Constant component of the area usage model 74
5.3 Linear component of the power consumption model 76
5.4 Constant component of the power consumption model 77

List of Tables

2.1 Demostration of two different residue codes. Numbers taken from [Taylor (1984)] 8

2.2 Comparison of a traditional and a RNS FIR filter. Taken form [Cardarilli et al. (2000)] 23

5.1 Coefficient for the model of the area usage for the RNS filter and for the reference filter 72

5.2 Coefficient for the model of the power consumption for the RNS filter and for the reference filter 75

Abbreviations

ADC	Analog to Digital Converter
CRT	Chinese Remainder Theorem
DSP	Digital Signal Processing
FIR	Finite Impulse Response
LSB	Least Significant Bit
LUT	Look Up Table
QRNS	Quadratic Residue Number System
RNS	Residue Number System
ROM	Read Only Memory
RTL	Register Transfer Level

Chapter 1

Introduction

1.1 Motivation

Radios and radio receivers are active areas of study both in academia and in the industry. As typical for the field of electronics there is an eternal pursuit for higher performance, lower cost and improved energy usage. One suggested way to achieve this is to do the calculations using number systems other than the standard binary. Among them are the Residue Number System. This system has received a lot of interest due to the fact that normal arithmetic can be done in a more efficient manner if it is implemented using RNS. The reason for the improved arithmetic is that a number in Residue Number System is made up of residues associated with different moduli and these residues can be calculated in parallel when doing addition and multiplication. This parallelism will lead to less carry propagation, which will lead to lower energy usage and lower latency. It can also lead to a simpler multiplication blocks that will provide an area gain in systems where a lot of multiplication is preformed. A big part of the operations, which are done in a radio receive chain, involves digital signal processing. Signal processing, in turn, involves a lot of addition and multiplication. Therefore one can assume that a lot of gain can be made by implementing a radio receive chain using RNS.

There is, however, some challenges in using this number system, which has prevented it from becoming the preferred number system used in the design of commercial radios. The challenges includes: a great overhead by the conversion to and from this number system and that operations, aside from arithmetic operations, are difficult to do in an efficient manner.

This thesis will explore the trade off between the challenges and gains by using this number system and will try to identify cases in which it will be beneficial to implement a system or a part of a system in the RNS domain instead of in the binary domain. It will use common building blocks in an radio receive chain as the basis of this exploration and will see how the area and energy are effected by being implemented in an other number system.

The potential gain in power consumption and area usage by using RNS in the receiver of a radio will be welcome in all applications where radios are used. Lower energy usage is one of the main goal for all radios used in mobile applications. This includes radios used in mobile phones and in the upcoming "Internet of Things" marked. Area usage is always sought after, as it will decrease the price of an integrated circuit. This thesis will, as a result of its goal to identify when and how RNS can be used to provide gains in these tow, act as a tool for those who are looking for ways improve the power consumption and area usage of their design.

1.2 Goals

The overall goal of this thesis is to explore how the residue number system can be used in a radio receive chain, which effect it will have on the area and power usage and which cases are most suited for using this number system.

1.3 Approach

This thesis will have a empirical approach in exploring the viability of the residue number system. The method can be described by the following steps.

- Implement common building blocks of a radio receive chain using the residue number system.
- Do a netlist simulation on each of those building blocks to get their area usage and power consumption.
- Do the same simulations on reference designs. These reference designs perform the same mathematical operation as the RNS building block only that

they are implemented using the 2's complement binary number system.

- Use the results from the netlist simulations of both the RNS designs and reference designs to discuss the suitability of the residue number system in a receiver and identify in which cases RNS is advantageous compared to conventional number systems.

1.4 Constraints

When researching alternative number systems, one have to focus on one or some of the many available number system, as focusing on everything would be to much work for one master thesis. In this thesis only the residue number system and the related quadratic residue number system is explored. This is because it is preferable to go into dept of a few number systems rather than exploring a small bit of many number systems. As the goal is to say something about how suited this number system is in digital receivers, the thesis will look at how the number system can be used in relation to this. This thesis will not study radio receivers in itself but rater study specific operations, which are used in radio receivers, manly in the DSP domain. The operations studied in this thesis are: FIR filtering, both with real and complex coefficients, scaling and conversion to and from RNS. These operations are not exclusive for radio receivers and the results from this thesis will be applicable for other applications as well.

For each of the operations presented in this thesis, there are many ways they can be implemented. To keep the thesis limited and to be able to do a thorough analysis of the chosen implementations, there will only be focused on a few specific methods of implementation. This puts a serious limitation on the scope of thesis, as the results is only directly applicable to these particular implementations. However it is argued in the thesis that the result has a transferability to other methods of implementation, which are not discussed in this thesis.

Lastly the simulation setup also puts a constraint on the scope of this thesis. The simulation is only done on a particular ASIC technology, using a a particular simulation tool and only some parameters from this tools are used in this thesis. This is due to the simulations being so time consuming that the limitation had to be set at this point.

1.5 Contribution

This thesis has to be seen in conjunction with all other work done in this subject. The residue number system has been an active field of research for a long time. This is also the case for the subject of using RNS in digital circuits[Mohan (2012)]. Despite this there is still some challenges, which need to be solved, for this number system to be used in commercial applications[Mohan (2012)]. This thesis aims to expand the knowledge in this field of research and offers the following contributions to the field:

- This thesis will provide new empirical data on the use of the residue number system for both FIR filtering and a scaling and will provide data on the overhead by input and output conversion.
- While the designs presented in this thesis are not novel, do the setup of the simulations and presentation of the simulation data lead to a more thorough understanding in what the power consumption and area usage is for these systems than what earlier work do.
- The presented complex RNS FIR filter is a novel design based on the hybrid RNS/binary methodology, which is adapted to the quadratic residue number system.
- The thesis presents novel mathematical models for the area usage and power consumption of RNS FIR filters and both are found to fit well with the empirical data.

1.6 Report Structure

The report will be structured as follows: Chapter 2 goes through the theoretic framework of this thesis and mentions previous works, which has ben done on the topic of RNS. Chapter 3 describes how the RNS operations, which are explored in this thesis, are implemented. Chapter 4 describes how the simulations of these implementations are done and presents the results from these simulations. Chapter 5 discusses the results and discusses the viability of RNS in a radio receiver. Chapter 6 concludes this thesis and suggests future research that can be done to supplement the findings of this thesis.

Chapter 2

Theory and Background

This chapter will go through the theoretic framework of this thesis and also discuss some of the previous works, which have been done on this topic. It will start by going through the mathematical foundations of the Residue Number System. Here a definition of the number system will presented and an efficient RNS code can be constructed is thereafter discussed. Next the chapter will go through how arithmetic is done using the Residue Number System. The chapter will also mention some other operations and go into depth of one method to do the two operations scaling and sign extension. These operations are mentioned in detail, because both are essential to the construction of the scaler, which is implemented as part of the thesis. Also the Quadratic Residue Number System is explained. QRNS is a way to represent a complex number in the residue number system. Using this system, complex multiplication can be done in a simpler manner, using only two multipliers instead of four. Thus a significant gain can be achieved in systems with lots of complex multiplication. The chapter will also explain two methods of input- and output conversion and how they can be implemented. The topic of DSP in the RNS domain is also discussed. Earlier work on this topic will be presented. The focus is on FIR filters and how they can be constructed using RNS and what the gain of doing this is. The next section will discuss earlier work on the topic of using RNS in a radio receiver, which methods have been used and what result they have gotten.

2.1 RNS - Residue Number System

The study of the residue number system has it roots in the Chinese remainder theorem presented by the third century mathematician Sun Tzu in the book Suan-ching[Taylor (1984)] and has been an active field of study since then. Due to the number systems applications in digital electronics, it has received increased attention the last 50 years. Garner (1959) describes how this number system can be used to accelerate arithmetic operations.

Garner (1959) explains this number system in therms of linear congruence, which is expressed as

$$A \equiv a \mod b \tag{2.1}$$

and is read, A is congruent to a modulo b[Garner (1959)]. This means that the equation

$$A = a + tb \tag{2.2}$$

is valid for some value t. Where A, a, b and t are integers. a is called the residue and b is called the base or the modulus of A. All the following examples are valid congruences.

$$10 \equiv 7 \mod (3) \tag{2.3a}$$

$$10 \equiv 4 \mod (3) \tag{2.3b}$$

$$10 \equiv 1 \mod (3) \tag{2.3c}$$

The validity of these congruences can easily be seen, as all of them satisfies the equation 2.2. To take the example 2.3c. A is 10, a is 1 and b is 3. If these are inserted in to equation 2.2, and t is chosen to be 3, this equation is satisfied as $1 + 3 \cdot 3 = 10$. The same can be done for the other examples.

It is evident that, for a given modulus b, a number A can have several residues b. In the A residue number for a particular natural number is formed from its least positive residues with the respect to a number of bases. The least positive residue is the residue a for which $0 < a < b$ in regards to equation 2.1. In the example used in equation 2.3 1 is the least residue for the number 10 when the base is 3.

Another and maybe simpler way to explain the residues, is as the remainder a of the integer division

$$t = \frac{A}{b} \tag{2.4}$$

This is equivalent with the modulo operation used in programming. Often written $a = A\%b$.

For an efficient residue code, the moduli must be relatively prime. If the moduli are not relatively prime, redundancy will be introduced in the code and the range of numbers, which can be represented by this code, is lower. The number of unique codes for a set of moduli is the lowest common multiplier of all moduli. If the moduli are pairwise primes, the number of unique codes is as following.

$$M = \sum_{i=0}^{N-1} m_i \tag{2.5}$$

Where M is the number of unique codes, m_i is the modulus and N is the number of moduli. Numbers being pairwise primes means that any two numbers, of the moduli set, has no common prime factor. Consider two different residue code implementations. One using the two moduli 2 and 6, and one using the moduli 4 and 3. 2 and 6 are not pair wise primes, but 3 and 4 are. In table 2.1 one can see that using the residues associated with the moduli 2 and 6 provides 6 unique codes, whereas the residues associated with the moduli 3 and 4 provides 12 unique codes. This corresponds to the mentioned fact that residues associated with moduli, which are not pairwise prime, introduces redundancy. One can see that the residue associated with the base 2 is 1 only when the residue associated with 6 is an odd number. No such property exists in the base pair 3 and 4 and therefore a larger number of unique numbers can be represented using this base pair. The number of unique codes represented by the moduli set 2 and 6 is equal to the product of the prime factors, of the two numbers, which are not common.

For most applications, especially those that exploit the arithmetic advantages of RNS, being able to represent a as large number of unique numbers as possible using as few bits as possible is preferable. However, redundancy can be exploited to make the code more fault tolerant. For example, using the moduli 2 and 6, as shown in table 2.1, a code in which the residue associated with the base 2 is 1 and the residue associated with the base 6 is an even number is an invalid code and indicates that an error has occurred.

Number	Least positive residue			
	mod 2	mod 6	mod 3	mod 4
0	0	0	0	0
1	1	1	1	1
2	0	2	2	2
3	1	3	0	3
4	0	4	1	0
5	1	5	2	1
6	0	0	0	2
7	1	1	1	3
8	0	2	2	0
9	1	3	0	1
10	0	4	1	2
11	1	5	2	3

Table 2.1: Demostration of two different residue codes. Numbers taken from [Taylor (1984)]

2.2 RNS arithmetic

The main disadvantage with 2's complement and other kinds of fixed radix, weighted number systems, is that the carry information that must be passed from digits of lesser significance to those with higher significance.[Taylor (1984)] The RNS is carry free in the sense that the carry does not have to be passed from one digit of the RNS number to another. This can be exploited to speed up arithmetic performance.

As a consequence of the structure of the residue number system and the properties of congruence, multiplication and addition are valid as long as the number of unique states of the residue number is large enough to represent the respective product or sum. [Garner (1959)] Both operations, multiplication and addition, are done parallel on all residues of the residue number. Take the three residue numbers $X = \{X_1, \ldots, X_L\}$, $Y = \{Y_1, \ldots, Y_L\}$ and $Z = \{Z_1, \ldots, Z_L\}$ where Z is either the sum or the product of X and Y. A multiplication or addition will be given by

$$Z = \{Z_1, \ldots, Z_L\} = XoY = \{X_1 o Y_1 \bmod (p_1), \ldots, X_L o Y_L \bmod (p_L)\} \tag{2.6}$$

where o represents either operation $+$ or $\cdot$ [Taylor (1984)]. To exploit the parallelism, circuits that can efficiently do the $XoY \bmod (p)$ has to be used[Taylor (1984)]. There is no conventional ways to do these operations for the general case. However Vergos and Efstathiou (2008) propose a way to do efficient modulo addition if the bases are chosen to be $2^n - 1$, 2^n or $2^n + 1$. If this problem is to be solved for the general case, the most common method is to replace the arithmetic operations with lookup-tables[Taylor (1984)]. Another way to implement it, is to implement it directly using adders or multipliers for the adding or multiplication operations and doing a modulo operation on the result.

While the carry free nature of the residue number system speeds up the performance of addition, multiplication and subtraction, does it come short of conventional 2's complement in other regards. The operations that are difficult to do in an efficient manner using RNS includes [Garner (1959)]

- Algebraic comparison
- Dynamic range extension
- Division
- Sign detection

However there has been done research on how to conduct these operations more efficiently. For instance does Jullien (1978) present two ways to do a scaling operation.

2.2.1 Scaling

He claims that, although scaling is difficult to do using RNS, it is preferable to include scaling in many applications where RNS is suitable. An example is the use of scaling in aa digital correlator. If scaling is not used in the correlator, the dynamic range of the used RNS code needs to be large enough to avoid overflows. This means the complexity of the RNS code has to be larger than what is necessary for most of the calculation. However, if scaling is used, the RNS code can be less complex and the signal, in the calculation chain, can be scaled in those cases that overflow would be a problem. The argument here is that the gain of using a less complex RNS code is greater than the overhead of the scaling operation. Jullien argues that applications which mainly consists of subtraction, addition and multiplication and a few scaling operations may be beneficial to do in RNS. In general, one can say that, applications that include a lot of easy operations and few difficult operations are beneficial to do in RNS.

Jullien (1978) proposed method for scaling is based on division. As mentioned earlier, division is a difficult operation in the RNS. Unlike multiplication and addition, division can not be done isolated on the separate digits of the RNS code. However Jullien (1978) uses the following equation to calculate the result of the division:

$$y_i = |Y|_{m_i} = \left| |X - |X|_K|_{m_i} \cdot \left|\frac{1}{K}\right|_{m_i} \right|_{m_i} \tag{2.7}$$

Here Y is the result of the division $\frac{X}{K}$, which means that Y is scaled by the factor K. $\left|\frac{1}{K}\right|_{m_i}$ is the multiplicative inverse of K, and can be defined by the equation

$$\left| K \cdot \left|\frac{1}{K}\right|_{m_i} \right|_{m_i} = 1 \tag{2.8}$$

by choosing K as one of the moduli m_0, the equation can be written as

$$y_i = |Y|_{m_i} = \left| |X - |X|_{m_0}|_{m_i} \cdot \left|\frac{1}{m_0}\right|_{m_i} \right|_{m_i} = \left| |X - x_0|_{m_i} \cdot \left|\frac{1}{m_0}\right|_{m_i} \right|_{m_i} \tag{2.9}$$

This function can be realized using look up tables. However, large tables has to be used if the whole function is to be realized. It is possible to only realize $\left| \square \cdot \left|\frac{1}{m_0}\right|_{m_i} \right|_{m_i}$ using a look up table and then realize $|X - |X|_{m_0}|_{m_i}$ by doing the modulo operation on the result of $X - x_0$. This will lead to a smaller table, but will increase the logic needed for the arithmetic operations.

The problem of using this method is, the residue for the modulo m_0 can not be calculated this way. However, since Y is divided by m_0, its dynamic range is is limited by

$$Y < \prod_{i \neq 0} m_i \tag{2.10}$$

Therefore the residue y_0 can be calculated by doing a base extension on Y.

2.2.2 Base Extension

Both Jullien (1978) and Shenoy and Kumaresan (1989) describe a method for base extension they call the *Szabo-Tanaka method* first presented in Szabo and Tanaka (1967). This method uses a recursive way to calculate the residue of the extended base. Take a RNS number X, coded with the moduli $\{m_1, m_2, \ldots, m_n\}$. This number is limited by $x < \prod_{i=0}^{n} m_i$. This number is to be base extended to also include the residue of the modulo m_{n+1}. This residue is given by the formula.

$$|x|_{m_{n+1}} = \left| x_0 + \sum_{k=1}^{n} a_i \prod_{i=0}^{k-1} m_i \right|_{m_{n+1}}, \tag{2.11}$$

where a_i can be calculated by

$$a_i = \left| \left| R_i^{(i-1)} - a_{i-1} \right|_{m_i} \cdot \left| \frac{1}{m_{i-1}} \right|_{m_i} \right|_{m_i}, \tag{2.12}$$

$R_i^{(k+1)}$ can be calculated using the equation

$$R_i^{(k+1)} = \left| \left| R_i^{(k)} - a_k \right|_{m_i} \cdot \left| \frac{1}{m_k} \right|_{m_i} \right|_{m_i} \tag{2.13}$$

The corner cases $R_i^{(0)}$ and a_0 is defined as following: $R_i^{(0)} = |X|_{m_i}$ and $a_0 = |X|_{m_0}$. $\left|\frac{1}{x}\right|_{m_i}$ is the multiplicative inverse to x associated with the modulo m_i. And is defined by equation 2.8. The structure of this algorithm is shown in figure 2.1. Where the residue X_5 is found based on the residues $\{X_0, ..., X_4\}$.

To explain this method, a simple example is presented. Take a Number $X = 127$, which is represented by a residue code using the following moduli $\{4, 5, 7\}$. The residue code for this number is thus $\{3, 2, 1\}$. This code is to be extended to include the residue associated with the modulo $m_3 = 3$. Note that X is lower than the product of the three moduli $\{4, 5, 7\}$ and can therefore be uniquely presented by this modulo set. Equation 2.11 gives

$$|X|_{m_3} = \left| x_0 + a_1 m_0 + a_2 m_1 m_0 \right|_{m_3} = \left| 3 + a_1 \cdot 4 + a_2 \cdot 5 \cdot 4 \right|_3 \tag{2.14}$$

a_1 and a_2 is found with equation 2.12.

$$a_1 = \left| \left| R_1^{(0)} - a_0 \right|_{m_1} \cdot \left| \frac{1}{m_0} \right|_{m_1} \right|_{m_1} = \left| \left| 2 - 3 \right|_5 \cdot 4 \right|_5 = 1 \tag{2.15}$$

$$a_2 = \left| \left| R_2^{(1)} - a_1 \right|_{m_2} \cdot \left| \frac{1}{m_1} \right|_{m_2} \right|_{m_2} = \left| \left| R_2^{(1)} - 1 \right|_7 \cdot 3 \right|_7 \tag{2.16}$$

To solve a_2, $R_2^{(1)}$ must be found using equation 2.13.

$$R_2^{(1)} = \left| \left| R_2^{(0)} - a_0 \right|_{m_2} \cdot \left| \frac{1}{m_0} \right|_{m_2} \right|_{m_2} = \left| \left| 1 - 3 \right|_7 \cdot 2 \right|_7 = 3 \tag{2.17}$$

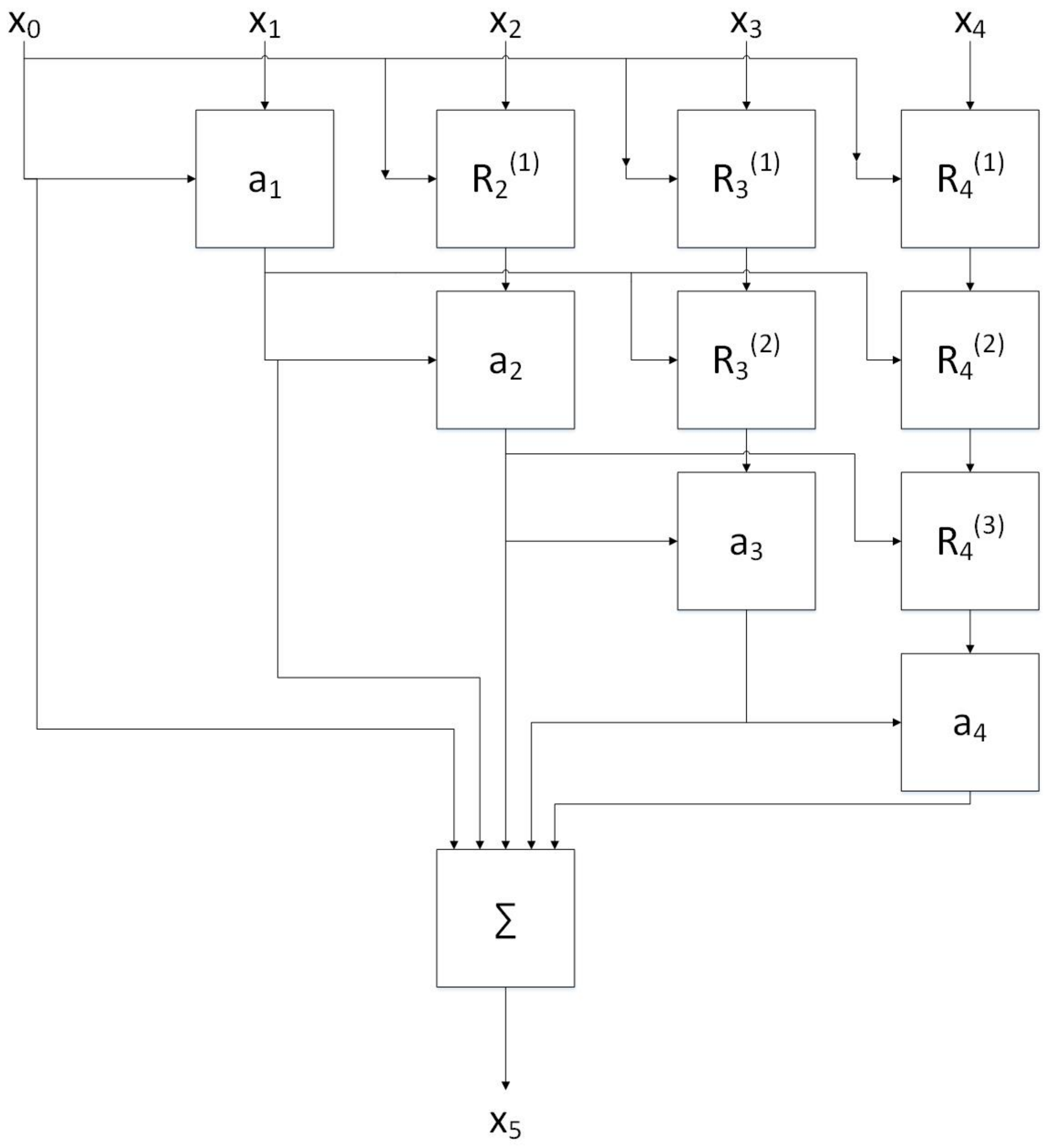

Figure 2.1: Block diagram for base extension algorithm

Which means $a_2 = 6$. a_1 and a_2 is inserted into equation 2.14

$$|X|_{m_3} = \left|3+1\cdot 4+6\cdot 5\cdot 4\right|_3 = 1 \tag{2.18}$$

Where 1 is the residue for the extended base $m_3 = 3$.

The advantage of using this method for base extension is that the partial calculations for all a_i and $R_i^{(k)}$ can be done using look up tables. The look up tables will, if the chosen moduli is small, be of limited size.

The look up tables for equation 2.12 can be addressed by the result of the modular subtraction $\left|R_i^{(i-1)} - a_{i-1}\right|_{m_i}$, for which it is m_i different result. Therefore a look up table of size m_i is needed for this function. The look up tables for equation 2.13 can be addressed by the result of the modular subtraction $\left|R_i^{(k)} - a_k\right|_{m_i}$. This subtraction does also have m_i different results and the size of the look up tables are m_i for this equation as well. Equation 2.11 can be implemented as the modular sum of several products, where each product is calculated from look up tables. As it is a modular product, each product can be limited by the modulus m_{n+1}. Therefore the size of these look up tables can be limited to m_{n+1}.

This means that, the sizes of all look up tables used, are determined by the size of the moduli used for the RNS code. Therefore this is an efficient method for base extension for RNS codes with small moduli.

2.2.3 Signed Scaling

The explained method of scaling does only work for unsigned numbers. The algorithm can be extended to work for signed numbers by subtracting the maximum value of the post division residue number form the result of the scaling, in the cases which the number before scaling is also a negative number. This property comes directly from how a signed number is represented in RNS, where the negative numbers occupies the upper half of the RNS number space and the positive numbers the lower half. When the number is scaled, the dynamic range of the RNS numbers are reduced with the scaling factor. For example take a number X which is represented using a RNS code allowing M different values. The number is a signed number and therefore the positive numbers are represented by numbers 0 to $\frac{M}{2}-1$ in the RNS, whereas the negative numbers are represented by the numbers $\frac{M}{2}$ to M. After the scaling, the positive numbers will occupy the range 0 to $\frac{\hat{M}}{2}-1$ and the negative from $\frac{\hat{M}}{2}$ to $\hat{M}$, where

$\hat{M} = \frac{M}{K}$ and K is the scaling factor. For the negative numbers to be represented properly in the post scaled number, the negative numbers must be moved to the numbers from $M - \frac{\hat{M}}{2}$ to M. This can be done by the operation $M + \hat{Y} - \frac{\hat{M}}{2}$ when the result is negative, where $\hat{Y}$ is the number after scaling.

To use this method it must be detected whether the resulting number is positive or not. This can be done by looking whether the residue number is higher or lower than $\frac{M}{2}$. The problem with this method is that a comparison in the RNS domain is difficult without converting to 2s complement. However, one can reduce the dynamic range of the RNS number by scaling it. The value $\frac{M}{2}$ is scaled by the same factor as the overall number. If the number is scaled sufficiently, so that the dynamic range of the resulting number is less than one of the moduli, finding out whether the number is larger or smaller than a given value is trivial. To be able to calculate this exactly, without introducing rounding errors, the dynamic range M of the RNS number as well as the scaled number has to be even. The easiest way to do this is to scale by division and divide repetitively using the moduli as divisors.

To illustrate this method, take a number X with the dynamic range M where 0 to $\frac{M}{2} - 1$ is positive numbers and $\frac{M}{2}$ to M represents negative numbers. The number is a residue number with the moduli $\{m_1, ..m_n\}$ where m_1 is an even number. If the number X is divided subsequently by each modulo m_i where $i \neq 1$, the resulting number $\hat{X}$ has a dynamic range of $\hat{M} = m_1$. Due to m_1 being even, it can be shown that if $\hat{X}$ is bigger than equal $\frac{\hat{M}}{2}$ then X is bigger than or equal $\frac{M}{2}$ and X is negative.

When implementing the sign detector, one can use the fact that the form of the base extension equations (2.12) and (2.13) are identical to the equation (2.7). The base extension algorithm can be seen as multiple division by the moduli m_i and the a_is are the resulting residues for the modulo m_i divided on the moduli $\{m_0, .., m_{i-1}\}$. The structure of the base extension in figure 2.1, can be used to explain this relation between division and the base extension. In each row r_i, a division by the modulus m_{i-1} is done for all residues X_n, for which $n \geq i$. For example is the result form the blocks in the first row, the result of the number represented by the residues $\{X_0, ..., X4\}$ divided by m_0 as a residue number based on the moduli $\{m_1, ..., m_4\}$. The same is the case for the other rows, and the number of residues used in the residue code representing the number is decreased by one for each row. In he last row, the result will be described with only one modulus m_N and a_N is thus the original number divided on all moduli except m_n. If m_N is chosen to be an even number, as explained in last paragraph, the original number is a negative number when $a_N \geq \frac{m_N}{2}$.

2.3 Quadratic RNS

The Residue Number System allows to represent complex numbers in a more efficient manner. One efficient way is using the Quadratic Residue Number System. This system has particular large advantage when it comes to complex multiplication. Using only two multiplications, as opposed to 4 or 3 when using normal 2's complement.

The property of the RNS, which allows the efficient complex representation, is that $j = \sqrt{-1}$ can be represented differently for residues associated with moduli on a particular form. The residues have to be on the form, such that the equation

$$x^2 + 1 = 0 \tag{2.19}$$

is valid for one integer x[Omondi and Premkumar (2007)]. As the modulo m_i is congruent with 0, is this valid for all moduli in which

$$0 = |k^2 + 1|_{m_i} \tag{2.20}$$

for an integer k. The quadratic residue number represented by the two values X and X* and which is calculated from the complex RNS number $(x + jy)$ in the following way

$$X = x + jy \tag{2.21a}$$

$$X^* = x - jy \tag{2.21b}$$

The value of both X and X^* can be found as a real number due to, the fact that j can be found by equation 2.19. An example is given. Take a residue code with one moduli $m_i = 5$. j is the result of equation 2.20 solved for k. For this particular modulo, j is 2, as $j^2 = 4 \equiv -1 \mod (5)$. Thus X and X^* are calculated as

$$X = x + 2y \tag{2.22a}$$

$$X^* = x - 2y \tag{2.22b}$$

As said previously the main advantage of using the QRNS is that a complex multiplication can be done using only two multipliers. Arithmetic in QRNS is done the following way

$$(X, X^*) \pm (Y, Y^*) = (X \pm Y, X^* \pm Y^*) \tag{2.23a}$$

$$(X, X^*) \cdot (Y, Y^*) = (X \cdot Y, X^* \cdot Y^*) \tag{2.23b}$$

Which means that addition is as complex using QRNS as using ordinary RNS. However, multiplication is easier. Note that QRNS works the same way as RNS in that the number is divided into several residues. The different way to represent the number, as described by equation 2.21, is isolated to a particular residue. This means that this equation must be applied to each of the residues to generate the QRNS number. The numerical values of $j = \sqrt{-1}$ for each residue is different, and must be considered when calculating the QRNS values.

The QRNS number can be calculated directly from a RNS number using the equation 2.21. This equation is divided into two parts calculating the complex and complex conjugate of the number. Both is calculated by using a multiplication by a constant and an addition or a subtraction. This leads to a fairly simple implementation of a QRNS to RNS converter, as both multiplication by a constant and addition and subtractions are trivial operations, especially for operands with low word widths, which is the case in the RNS domain. For each residue, two constant multipliers, one adder and one subtractor is needed. Thus for a QRNS number with N residues, a total of $2N$ constant multipliers, N adders and N subtractor are needed to convert a number form RNS to QRNS.

The calculation form QRNS to RNS can be done by equation 2.24[Omondi and

Premkumar (2007)]

$$x = \left| \frac{|X + X^*|_{m_i}}{2} \right|_{m_i} \tag{2.24a}$$

$$y = \left| \frac{|X - X^*|_{m_i}}{2j} \right|_{m_i} \tag{2.24b}$$

This equation can be written, using multiplicative inverses instead of division, as:

$$x = \left| |2^{-1}|_{m_i} |X + X^*|_{m_i} \right|_{m_i} \tag{2.25a}$$

$$y = \left| |2^{-1}|_{m_i} |j^{-1}|_{m_i} |X - X^*|_{m_i} \right|_{m_i} \tag{2.25b}$$

This equation is done independently on each of the residues. Therefore one has to do the calculation of the real and imaginary part for each of the residues.

The only part of these equations, which is not constant, is the results from the addition and subtraction $X \pm X^*$. Therefore the equations can be implemented using two look up tables with the results from this addition or subtraction as address. Alternatively one can use the result from a modulo addition and subtraction using the same operands. The look up tables using the non modular addition and subtraction will have $2m_i - 1$ entries, where m_i is the modulo this residue is associated to. Using the modular addition and subtraction, the look up tables will have m_i entries, but modular adders and subtractors have to be used, which are more complex than the non modular ones. However, the modular adder and subtractors can be constructed relatively simply, as the operations has $2m_i - 1$ possible results, due to the operands can max be $m_i - 1$. Therefore a correction of $\pm m_i$ can be used to get the modulo value of the result of the addition or subtraction.

To conclude are the benefits the QRNS the ease of doing multiplication. While the disadvantages are the overhead of converting from ordinary RNS to QRNS and the reduced flexibility regarding the choice of residues when using QRNS.

2.4 Input and Output Conversion

The limitations of the residue number system prevent its general usage. However, as the RNS can be used to speed up arithmetic calculation, it can be used to complement convectional number systems. Therefore there is a need to be able to convert to and form RNS in an efficient manner. As stated earlier, the main overhead by using RNS is the input and output conversion. A lot of literature has been published to solve this problem, as seen by the large number of references in chapter 3 and 7 in this book[Omondi and Premkumar (2007)], by making the overhead as low as possible.

2.4.1 Input Conversion

The task of an input converter is, given an integer input X to and a series of moduli or bases $\{m_1, m_2, \ldots, m_L\}$, to produce a residue code consisting of L numbers, where each number is the residue associated with each modulo on the form $x_n = X \mod m_n$. Likewise is the task of the output convert to, given L residues $\{x_1, x_2, \ldots, x_L\}$ associated with L moduli $\{m_1, m_2, \ldots, m_L\}$, to produce an integer X for that corresponds to this unique set of residues.

One method of input conversion is described by Jenkins and Leon (1977) in the following manner: Consider a binary number of the form

$$X = \sum_{j=0}^{t} B_j * 2^j \tag{2.26}$$

where B_j is the bit at position j of a $t+1$ length binary number. Then the residue associated with m_i is

$$|X|_{m_i} = \left| B_t(m_i - |2^t|_{m_i}) + \sum_{j=0}^{t-1} B_j |2^j|_{m_i} \right|_{m_i} \tag{2.27}$$

$|X|_{m_i}$ is another way to write "$X \mod m_i$". If the function $F(J)$ is defined as following:

$$F(j) = \begin{cases} |2^j|_{m_i}, & \text{if } j = 0, 1, \ldots, t-1 \\ m_i - |2^t|_{m_i}, & \text{if } j = t \end{cases}, \tag{2.28}$$

equation 2.27 can be written as

$$|x|_{m_i} = \left| \sum_{j=0}^{t} B_j F(j) \right|_{m_i} \tag{2.29}$$

$|x|_{m_i}$ can be calculated by using look-up tables to get the values from the function $F(j)$ and calculate the sum $\left|\sum_{j=0}^{t} B_j F(j)\right|_{m_i}$ using modulo adders. Jenkins and Leon (1977) presents the structure shown in figure 2.2 to do this operation. This structure does the operation in a serial manner and uses a latch to store the partial sum from the modulo adder. The residues associated with each modulo is calculated in parallel, where the calculation for each of them is identical apart from the implementation of the $F(j)$ function.

The operation can be altered by using several of the bits of the input word to address the results in the ROM. Using this method, less modulo adders are needed to calculate the resulting residue. Consider that the ROM of a converter for a 8-bit input word is divided in two functions.

$$f_u(B_7, B_6, B_5, B_4) = \left| \sum_{j=4}^{7} 2^j B_j \right|_{m_i} \tag{2.30}$$

and

$$f_u(B_3, B_2, B_1, B_0) = \left| \sum_{j=0}^{3} 2^j B_j \right|_{m_i} \tag{2.31}$$

Only two modulo adders are needed, but larger ROM is needed to calculate the residue for a particular modulo.

2.4.2 Output Conversion

The output conversion is even more of a limiting factor, for the wider use of the residue number system, than what the input conversion is[Omondi and Premkumar (2007)]. The main methods of output conversion are based on the Chinese Remainder Theorem and the Mixed-Radix Conversion technique. All other methods are variants of these two methods[Omondi and Premkumar (2007)]. Some methods of output conversion are based on specific moduli sets an other are method for the general case.

The Chinese remainder theorem can be written in the following manner[Burgess

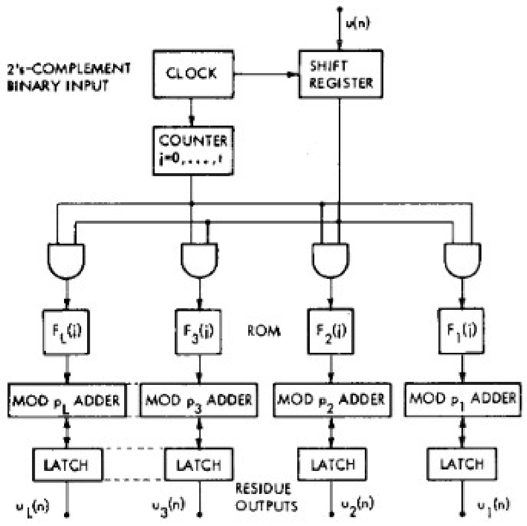

Figure 2.2: Converter structure. Taken from [Jenkins and Leon (1977)]

(2001)].

$$n = \left| \sum_{j=1}^{k} n_j \cdot B_j \right|_M \tag{2.32}$$

where n is the result of the conversion, n_j is the jth residue of n, and M is the unique number of codes that can be represented by a residue number system with k moduli and B_j is the function

$$B_j = \frac{M}{m_j} \left| \left(\frac{M}{m_j} \right)^{-1} \right|_{m_j} \tag{2.33}$$

where m_j is the jth modulo of the residue code.

Due to the big overhead by division and the fact that both M and m_j are constant for a given set of moduli, the B_j function is often realised using look up tables. Burgess (2001) realises the entire $n_j \cdot B_j$ function using a addressable look up table, where the number of addresses are equal the number of possible n_j values. For small modulo, the realization of this function in look up table is efficient both in speed and area, making the modulo summation the biggest reason it is difficult to do output conversion in an efficient manner using the Chinese Remainder Theorem. This is particularly the case if the unique number of residue codes M are large, as modulo addition is extremely inefficient for these cases.

One can argue that the look up tables for the function $n_j \cdot B_j$ is relatively small by looking at which values, that need to be stored in the ROM. As the B_j is constant, the look up tables needs to have the $n_j \cdot B_j$ for each possible n_j. If one assumes that n_j is positive, which it is due to the way residues are represented in RNS, the number of values in the look up table is $n_{j\max}$. Each value can be limited due to the fact that when doing modulo arithmetic, with respect to a modulo m, the value of each operand x can be limited to $|x|_m$, as the resulting value, after the final modulo, is the same both using x and $|x|_m$.

Several methods has been proposed as a solution to the mentioned problem of the modulo summation in equation 2.32. Burgess (2001) proposes a way using a SRT division-like architecture. This method is based on the fact that equation 2.32 can be written like

$$n = \sum_{j=1}^{k} \left| n_j \cdot B_j \right|_M - R(n) \cdot M \tag{2.34}$$

where $R(n)$ is known as the rank function[Burgess (2001)]. The rank function is used to determine how many times M has to be subtracted to the sum $\sum_{j=1}^{k} \left| n_j \cdot B_j \right|_M$ to get a result, which is the unique number, that has the residue $n_1 \dots n_J$ associated with the moduli $m_1 \dots m_J$ and is between 0 and M. $\sum_{j=1}^{k} \left| n_j \cdot B_j \right|_M$ will also give a number with these residues for these particular moduli, but this number is not necessarily between 0 and M. The reason this number can be transformed to a number with the exact same property and is between 0 and M by subtracting a multiple of M from the number, is because $M \mod m_i$ is 0 for all $m_1 \dots m_J$ and thus subtracting a multiple of M from the number will not change the residues associated with these particular moduli.

When using this method, the problem is finding a good way to calculate $R(n)$. For this 2.32 proposes the following method. First a partial sum and carry are calculated using a carry save adder. An estimate ñ of $R(n)$ is calculated using the most significant of the sum and carry form the carry save adder. The estimate must be good enough so that ñ $= R(n)$ or $R(n)+1$ optionally ñ $= R(n)$ or $R(n)+1$. One can chose between the two possibilities by looking at the LSB of the each of them and determine which $n \cdot M$ can be subtracted from the number so that the LSB of result of the subtraction matches the LSB of the real result. For this to work one has to know what the LSB of the real result is. Therefore one need to propagate the LSB from the input conversion alongside the RNS calculations all the way to the output conversion. It is argued that the gates needed for this propagation are a small fraction of a full length subtractor, and the gain of doing this outweighs the cost of the subtractor[2.32]. Lastly $R(n)$ is used to chose the correct multiple of M form a look up table, which is subtracted from the partial sum and carry using a carry save adder. And the carry and sum from this adder are added together using a carry propagate adder. It is to be noted that, if this method is used, the moduli has to be chosen so that no moduli is an even number.

2.5 DSP in RNS

Due to the residue number systems weaknesses, it is not suited for general purposes. However, it is very well suited for applications in which there are a lot of multiplication, addition and subtraction and few comparisons, divisions, etc. One such application is in digital signal processing. The use of RNS in FIR filters is very well explored in literature[Nannarelli et al. (2001) , Cardarilli et al. (2000),Freking and Parhi (1997),Ibrahim (1994)].

The main advantage of using the residue number system for FIR filters is the possibility to do the calculation in parallel for each residue of the residue number. The main draw back, however, is the overhead introduced by the input and output conversion[Nannarelli et al. (2001)].

Nannarelli et al. (2001) compare error free FIR filters with a word length of 20 bits implemented in both RNS and conventional 2's complement. They come to the conclusion that for transpose filters, if the number of taps is greater than 8 the filter implemented using RNS will be faster and, if the number of taps is greater than 40, it will use less area and energy. For direct FIR filters the RNS implementation is slightly faster and the area is smaller for filters having more than 16 taps. These results are valid for both filters with programmable coefficients and with constant coefficients. Nannarelli et al. (2001) also suggest that the supply voltage for calculation of residues, not in the critical path can be reduced as a way to further reduce the power consumption of RNS filters. This result agrees with the results form another article the same authors have published[Cardarilli et al. (2000)]. In this article the RNS filter and the traditional filter are implemented using 0.35µm technology. The results show that the RNS filter has a large overhead due to the conversion to and from RNS number, but the lower area and power increase at a slower rate as the number of taps

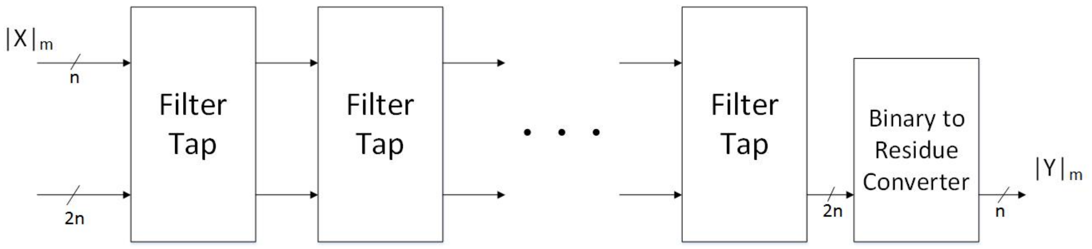

Figure 2.3: Hybrid FIR filter for a digit of the residue number. Taken from[Ibrahim (1994)]

increases, as shown in table 2.5.

Filter	cycle [ns]	Latency (cycles)	Area (gate equiv.)	Power [mW]
Trad.	5.0	N + 1	230 +1250N	14.9 + 13.5N
RNS	5.0	N + 3	2910 + 745N	51.0 +6.9N

Table 2.2: Comparison of a traditional and a RNS FIR filter. Taken form [Cardarilli et al. (2000)]

Ibrahim (1994) presents a way to construct FIR filters using a hybrid RNS-binary arithmetic method. Using this method the binary number is converted into RNS. Each digit of the residue number is calculated in parallel using convectional binary arithmetic instead of modulo arithmetic. In essence it will be n FIR filters calculating n results in parallel. After the calculation is done, the digit will be on a non RNS binary form and is converted back to RNS. The filters consist of cells consisting of a n bit multiplier, a $2n+1$ bit adder, a $2n$ and a $2n$ bit multiplexer. To avoid overflow in the calculation chains of the filters, $-m^2$ is added to the partial sum if the carry from the previous partial sum calculation is 1. Here m is the modulo used for this digit. Since $m^2 \mod m = 0$, this addition will not effect the final sum of the calculation for this particular modulo. Figure 2.3 shows the general structure of one of these parallel filters and figure 2.4 shows a single cell in this filter. Ibrahim (1994) claims that this architecture has a better performance and reduced cost, compared to conventional RNS arithmetic, for large moduli.

Freking and Parhi (1997) explore the use of RNS to reduce the power consumption of FIR filter, utilizing the method presented by Ibrahim (1994). Because RNS is used, the speed of the calculations is increased compared to a similar FIR filter designed using conventional binary numbers. Thus the supply voltage can be reduced while

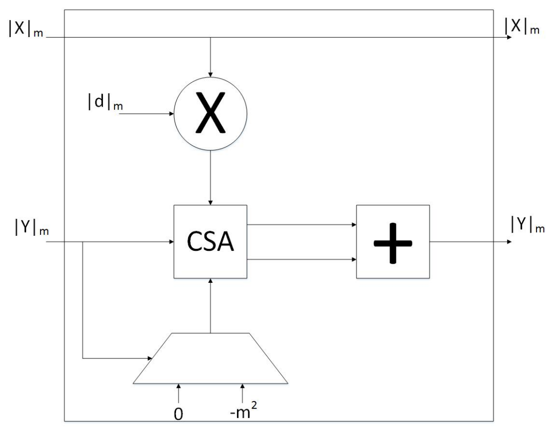

Figure 2.4: One cell of the filter in figure 2.3. Taken from[Ibrahim (1994)]

the speed of the operation is maintained. They show that, by using these techniques, the power per tap can be reduced for filters with word lengths as low as 8-bit and that, when the power overhead for the I/O conversion is taken into account, a FIR filter with 16-bits coefficients has a lower power consumption for filters with more than 20 taps.

2.6 Radios using RNS

The use of RNS in radio application is researched for example by Ramírez et al. (2002). In their paper the receiver consisting of a direct digital synthesizer (DDS) and a decimation filter is constructed. This receiver is implemented using field programmable logic.

The direct digital synthesizer is implemented using look up tables. These tables are addressed using the results from a phase accumulator and has a quarter wave symmetry to reduce the number of LUTs needed. The LUTs calculates the sine and cosine values for the given phases. The sine and cosine signals are then multiplied with the input signal of the receiver.

The decimation filter is a programmable RNS filter the products in the filters are calculated using a series of modulo adders, while the sum is calculated using a regular adder tree followed by a modulo stage.

In addition to the synthesizer and the filter, the receiver has a input and a output converter. This way the receiver will look like a normal binary receiver seen from the outside. Two output different types of converters are explored in this paper. One is based on the CRT algorithm and the other is based on the ε-CRT algorithm.

The receiver was implemented for a RNS codes with a number of dynamic ranges, ranging from 34- to 37-bit, and filters of varying lengths, form 8 to 64 taps. The result of the implementation shows that it is possible to increase the throughput and reduce the complexity of a receiver with certain dynamic ranges and filter lengths.

Chapter 3

Implementation

This chapter will present the implementation of different building block, which will be used in the continuation of this thesis. The implemented building blocks are: A FIR filter, an input converter, an output converter, a scaler, a complex FIR filter, a RNS to QRNS converter and a QRNS to RNS converter. The FIR filter and the scaler are implemented in the RNS domain and the complex FIR filter is implemeted in the QRNS domain. The different converters are of course an interface between two different domain and thus can not be said to be implemented in a particular domain. The input and output converters are interfaces between the binary and the RNS domain and the RNS to QRNS and QRNS to RNS converters are interfaces between the RNS and QRNS domain. The different design will be discussed in detail in each section and the energy, timing and area impact of these implementations relative to conventional implementations are discussed. Each section will also have a explanation of why this implementation is included, where the usefulness of doing the particular implementation in the RNS/QRNS domain is discussed based on the mathematical properties of the number system.

3.1 FIR filter

Chapter 2.5 discusses how RNS can be used in digital signal processing. In this chapter FIR filters are discussed in detail. The reason for the focus on FIR filters is that it is very well suited regarding the use of RNS. This is because FIR filters consists of registers, multipliers and adders. Registers are constructed the same way in the RNS domain as in the conventional binary domain. Although the RNS codes have some more redundancy in terms of the numbers of bits used for representing the same number. If the moduli used in the particular code is carefully selected, the number of extra bits needed is not that large as it will have a determinable effect on the RNSs suitability in the FIR filter. However, it may be significant enough to affect which set of FIR filters that are suitable for this number system. Multiplication and addition, as discussed in chapter 2.2, may be done more efficiently in the RNS domain compared to the conventional binary domain. This is especially the case when the word width of the operands are large enough. This is because the the parallelism of the RNS can be exploited more in these cases. It is believed that the gain of multiplication and addition is more significant than the disadvantage, due to the extra registers needed, for a big set of FIR filters. Especially those with large word lengths.

The advantages of using RNS in filter design must be weighed against the overhead introduced by the conversion to and from the RNS domain. The RNS filter can be implemented by itself or as a part of a bigger system implemented in the RNS domain. For the first case, when the signal is converted form binary to RNS run through a FIR filter and then converted back to binary, the gain/overhead of using RNS can be seen as the gain from the RNS FIR filter over the binary FIR filter minus the overhead from the conversion. The gain of the filter is believed to increase as the filter order increases, thus it is interesting to see at which order the gain outweighs the overhead. For the other case, when the filer is implemented as part of a larger system in the RNS domain, the gain or overhead of the other parts of the system must also be taken into account when deciding whether it is best to implement it in the RNS domain or the binary domain.

The implemented FIR filter is based on the method presented by Ibrahim (1994) and explained in chapter 2.5. The reason this method is chosen for the implementation is: Firstly, because the gains due to the less complex arithmetic units, as modulo arithmetic is not needed in for this construct. Secondly, because the abundance of modulo arithmetic units allows a more flexible selection of moduli. Modulo arithmetic units are most effective for moduli on the form $2^n - 1$, 2^n and $2^n + 1$. As the used method is effective for other moduli, it allows a broader selection of moduli.

The filter is constructed as several filters that run in parallel where each filter corresponds to one modulo. The possibility to construct the filter as several smaller filter in parallel is due to the properties of the RNS and is explained more in chapter

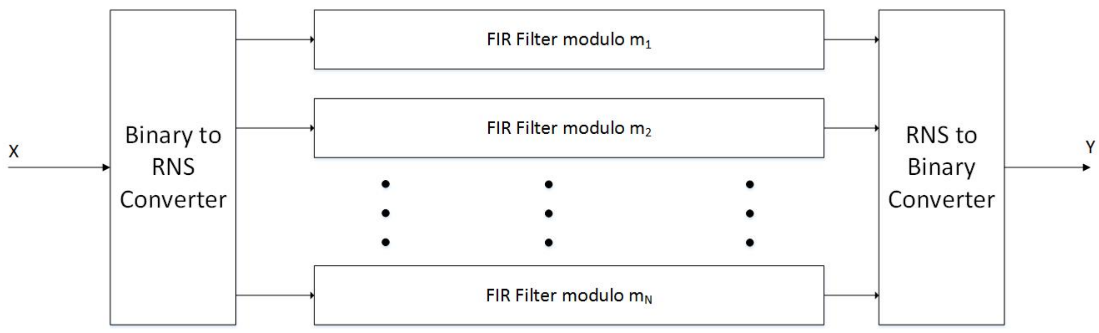

Figure 3.1: Block diagram of a FIR filter designed in the RNS domain

2. Each of the parallel filters are in turn made up of several filter cell that are joined together to construct the complete filter. The cell is one tap of the complete filter and is based on the cell in figure 2.4 on page 24. Each of the cells are connected together as seen in figure 2.3 on page 23 with a binary to residue converter to convert from the intermediate representation back to the residue representation.

To further optimize the power and area usage of the filter, constant coefficients are chosen instead of programmable coefficients. If we take the equation for a FIR filter:

$$Y = \sum x_i \cdot d_i \tag{3.1}$$

It is evident that the multiplication $x_i \cdot d_u$ accounts for a large portion of the complexity. Multiplication using constant coefficients reduces this complexity significantly and thus the complexity of the entire system is significantly reduced.

If the filter only consists of real coefficients. A fir filter for a complex signal can be realised by using two parallel filters. One for the real signal component and one for the imaginary. This is evident if we consider the complex multiplication as

$$Z = X \cdot Y = (X_r \cdot Y_r - X_i \cdot Y_i) + j(X_r \cdot Y_i + X_i \cdot Y_r) \tag{3.2}$$

If the coefficients Y are exclusively real, the multiplication will look like:

$$Z = X \cdot Y = (X_r \cdot Y_r) + j(X_i \cdot Y_r) \tag{3.3}$$

And we see that the real output is not dependent on the imaginary input signal, and similarly is the imaginary output not dependent on the real input signal. As filters consisting of only real coefficient are suitable for most application as well as the complexity of such a filter is significantly lower than a filter using complex coefficients, the coefficient for the filters presented is chosen to be exclusively real.

3.2 Input Converter

Due to the limitations of the residue number system, it is unlikely that a entire system is designed using it. However, as it is very well suited for some special operations, it is most likely to be used in systems utilizing both RNS and conventional binary number system. For such systems one need to be able to convert from binary to RNS to be able to transfer data from the binary domain to the RNS domain. There is a possibility to use binary and RNS in the same system, but the two systems has to be used in parallel and data can not transferred between the two domains. For a such system to work, an entire calculation has to be done in the RNS domain. The RNS calculation chain can for example have comparators or sign detector, to extract logic values, which are used in other parts of the circuit. However, these approaches are very inflexible, and the cases, which this will have a benefit over a combined RNS/binary system, are very limited. Another argument for the importance of input converter is, that a modern system will be made up of different IP blocks, which may be designed by different people or different companies. Data has to be transferred between these blocks. As 2's complement binary form is the standard way to represent a number, is the data, which is transferred between the IP blocks, most likely required to be on this form. And therefore, when the calculations in a block are done in the RNS domain, there will be a need for input and output converters in the system.

The task of the input converter is to take one input word, with a limited value range, and convert it into a set of residues, each associated with one of the predefined moduli. The operation can be explained by the following equation

$$X \Rightarrow \{x_1, x_2, ..., x_N\} \tag{3.4}$$

where x_i is given by

$$x_i = X \mod m_i \tag{3.5}$$

The block diagram of a general input converter is shown in figure 3.2.

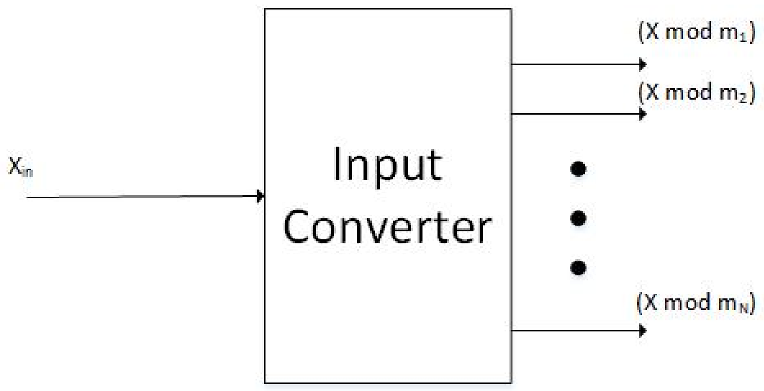

Figure 3.2: Block diagram of general input converter

An input converter is implemented based on the approach described by Jenkins and Leon (1977). However, their implementation is done in a sequential manner, but the implementation used is fully concurrent. This is done to avoid the extra delays that is introduced when by a sequential converter. The possibility to use a concurrent converter instead of the serial one is because of advances in technology, which means both that the whole calculation can be done fast enough to be executed in one clock cycle as well as that the converter occupies a smaller die area.

The converter is realized using a series of ROM, which stores the function $F(j)$, from equation 2.28, for each bit of the number that is to be converted. The values of each ROM is chosen depending on whether the bit on each position j is one or zero. The $F(j)$s, whose bit B_j is one, are added together using modulo adders to obtain the resulting residue for the used modulo. The modulo adders are made using a simple approach. The two numbers are added. If the result is higher or equal to the modulo m, the m is subtracted from the result. Otherwise the result is unchanged. The adder is explained by equation 3.6.

$$|a+b|_m = \begin{cases} a+b, & \text{if } a+b \leq m \\ a+b-m, & \text{if } a+b > m \end{cases} \tag{3.6}$$

Figure 3.3 shows what one of the cells in this converter looks like. As seen in equation 2.28 on page 18, the number of values, which need to be added together, are equal to the bits in the input word t. This means that the cell has to be connected t times in concession, where the partial sum out is the partial sum in of the next cell. The

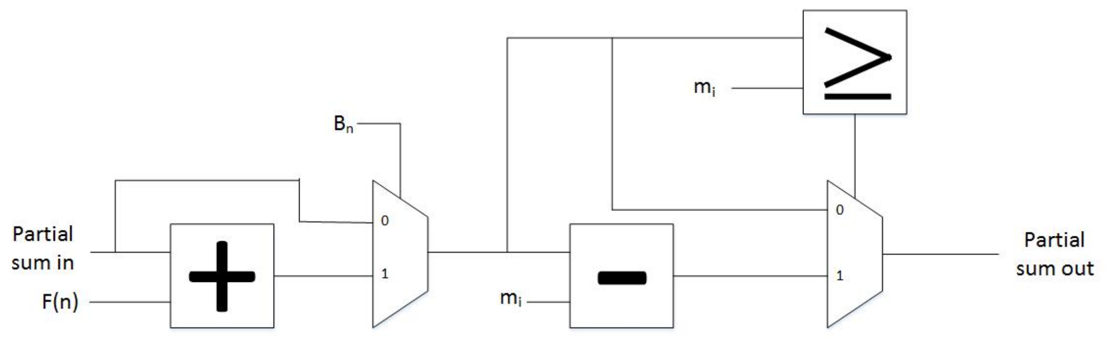

Figure 3.3: Single cell of a input converter

partial sum in of the first is set to zero and the part ial sum out of the last cell is the resulting residue for the modulo m_i. The value $F(n)$ comes from the ROM for the $F(j)$ values discussed earlier and B_n is the bit at position n of the input word.

3.3 Output converter

The output converter solves the same problem as the input converter, only the output converter converts from the RNS domain to the binary domain. The arguments for using an output converter are mainly the same as for a input converter. However, there is cases, in which one has to include an input converter and not an output converter or vice versa. One example is if one receives a signal, which already is on RNS form, but has to convert it back into binary form for further use in the system. This happens if the system includes an ADC, which converts the analogue signal directly into RNS or if one has a communication channel, in which the data is transmitted on RNS form. An example of a case, where an input converter is used and not an output converter, is when the signal is on binary form, is converted into RNS, the calculation is done in the RNS domain and the output of the calculation is a logic value provided by a comparison. Even though a comparison is difficult to do in the RNS domain, it may be preferable to convert the values to a binary form and do the comparison in this domain. Of these two, the case of having an input converter but no output

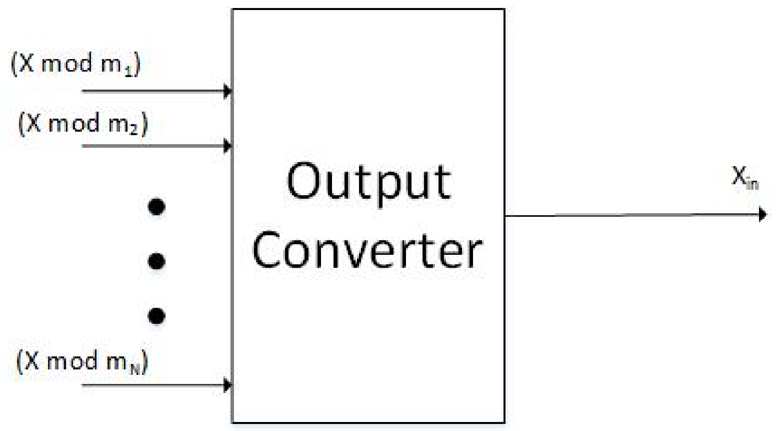

Figure 3.4: Block diagram of general output converter

converter is the most likely. This is because both RNS ADCs and transmission of data in RNS format are very unusual, but having the logic value of a comparison as the output of a calculation is not unusual.

The output converter, as the therm suggests, converts a signal on RNS form to a signal on binary form, as shown in the following equation

$$\{x_1, x_2, ..., x_N\} \Rightarrow X \tag{3.7}$$

The block diagram of a general output converter is shown in figure 3.3.

An output converter is implemented using the Chinese remainder theorem method. In this particular implementation, the $\left|n_j \cdot B_j\right|_M$ function from equation 2.34 on page 21 is implemented in ROM, as J different look up tables, where J is the number of moduli used for the RNS code. The ROM is addressed by the residue n_j associated with modulo m_j, choosing multiple of B_j, where B_j is as explained by equation 2.33 on page 20, that is to be used in the conversion.

Thus, from the ROM, there are J different $\left|n_j \cdot B_j\right|_M$ values, which have to be added together using a modulo adder. The modulo adder is on the form shown in equation 2.34. The limited number of moduli limits the number of possible $R(n)$s. If it is assumed the RNS code used in the converter has 5 different moduli, maximum result from the sum $\sum_{j=1}^{k} \left|n_j \cdot B_j\right|_M$ is $5M-5$, as the max number for each of the operands of the sum is limited to $M-1$ due to the modulo operation. The minimum value is 0, which is in the case all n_js are 0. This means that the possible values of

$R(n)$ ranges from 0 to 4 if the output from the alternative modulo addition is to be between 0 and $M-1$. To choose between these 5 possible values a form of a binary search tree is constructed.

Firstly, a partial sum and carry are calculated using a carry save adder tree. Then the correction of half the maximum value of the sum described in last paragraph is subtracted from this value using an additional carry save adder. The sum and carry of this adder is added together using a carry propagate adder. Now the overflow bit of this sum is checked to see if the resulting number is positive or negative. If depending on whether it is positive or negative a new correction is added to or subtracted from this value. This is done until a value is gotten, which is between 0 and $M-1$. The form of this conditional correction circuit resembles that of a binary search tree.

This value has to be converted into a signed form between $-\frac{M}{2}$ to $\frac{M}{2}-1$. As a signed residue code is made so that the values $0-\frac{M}{2}-1$ are the positive values 0 to $\frac{M}{2}-1$ and the values $\frac{M}{2}$ to $M-1$ are the negative values $-\frac{M}{2}$ to -1. To obtain the output value of the converter on this form, the values are left as they are if the result is between 0 and $\frac{M}{2}-1$ and M is subtracted from the result if the result is between $\frac{M}{2}$ to $M-1$.

3.4 Scaling

One of the disadvantages of using a residue number system, is that the scaling operation is difficult. The scaling operation described in this section is down scaling. Up scaling is trivial in the context of RNS, as it would be the same as multiplication by a scaling factor. Similarly is down scaling the same as division by a scaling factor and can be described by equation 3.8

$$Y = \frac{X}{K} \tag{3.8}$$

where Y is the value of X scaled by the factor K. Division is, as explained in chapter 2.2, very difficult to execute in an efficient manner in the RNS system. For this reason there must be as few scaling steps as possible in a system using RNS. However, there are many applications where the advantage of having a scaling operation outweights it overhead.

A common use case of scaling in a DPS system is to compress the amplitude of

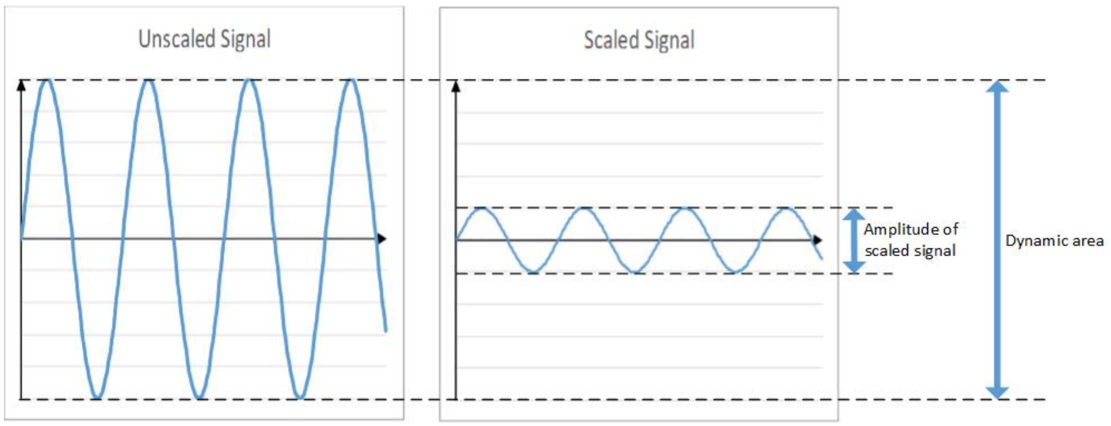

Figure 3.5: A scaled signal occupies a smaller portion of the dynamical area

the signal, so it occupies smaller portion of the dynamic area. For example in a system having several filter stages, it is wanted to scale the output signal form one filter before it arrives in the next filter. In a binary system it is possible to add new bits to the signal to increase the dynamic range of its representation and to shift the bits to reduce the amplitude of the signal. The bit shifting is similar to a power of 2 division, and therefore works like scaling by a power of 2. Both the scaling and the range extension are trivial to implement in the binary domain. Scaling can be implemented using the bit shifting method mentioned and by doing this the second filter can have a lower word width without experiencing overflow. Additional bits can be added to the word to increase the dynamic range of the signal as far as it is necessary.

A scaling will cause a loss of precision, as the least significant bits are removed. This loss of precision is not significant for many applications, because the most interesting aspect of the signal out of the filter is the relative amplitude of the samples. A trade off between how high precision is needed and the complexity of the circuit has to be made. The weighting of each of these aspects depends on the application and how the filter is used. Although the example discussed addresses a filter, the same elements are relevant for different applications, both in and outside the DPS domain.

When implementing the same system in the RNS domain, the difficulty of doing a division in the RNS domain has to be addressed. A naive solution to this problem is to convert the signal form a RNS form into a binary form, do the scaling in the binary domain and convert the signal back into RNS form. This, however, has a large overhead due to the input and output conversion. Chapter 2.2.1 discusses a method of scaling that reduces this overhead by not conducting the full conversion into a binary number.

The implemented scaler is realized using this method. This implementation is split into two parts. One for the calculation of the scaled residues $\{x_1,\ldots,x_{n-1}\}$ associated with the moduli $\{m_1,\ldots,m_{n-1}\}$ and another one for the calculation of the residue associated with the moduli m_n.

The calculation of the residues $\{x_1,\ldots,x_{n-1}\}$ is done using look up tables. It is, as described in chapter 2.2.1, based on the equation

$$\left| |x_i - x_n|_{m_i} \cdot \left| \frac{1}{m_n} \right|_{m_i} \right|_{m_i} \tag{3.9}$$

Note that the division factor is chosen to be m_n. This is because, by using this factor, the residue x_n can be used in the subtraction. If a factor, which was not a part of the moduli set is chosen, for example K, the subtraction value x_K has to be calculated as $x_K = X \mod K$, which increases the complexity of the scaling operation.

The realisation of the look up tables for equation 3.9 is quite simple, as $\left|\frac{1}{m_n}\right|_{m_i}$ is constant and the multiplication and final modulo operation can be done in one step. The input to each of the tables is the result of the subtraction $x_i - x_n$. The value of x_i is between 0 and $m_i - 1$ while the value of x_n is between 0 and $m_n - 1$. This means that the number of possible results of the subtraction is $m_i + m_n - 1$, which also is the number of entries in the look up tables. The result of the modulo subtraction $|x_i - x_n|_{m_i}$ could also be used to address the look up tables. This would reduce the number of entries in the table, as the result from this operation is between 0 and $m_i - 1$, which is m_i different values. However, modulo subtraction is far more complex than normal subtraction, and therefore the gain to the size of the tables using this value as address is nullified, which makes this methods inferior to the one used.

The output from these look up tables is the residue code of the scaled value for the moduli $\{m_1,\ldots,m_{n-1}\}$. The last residue must be found by doing a base extension of the number represented by these residues. The algorithm for the base extension is explained in chapter 2.2.1. The implementation of this algorithm is mostly done using look up tables, using the structure shown in figure 2.1 on page 12. The calculations of $R_i^{(K)}$ and a_i are on the same form as 3.9 and therefore are implemented the

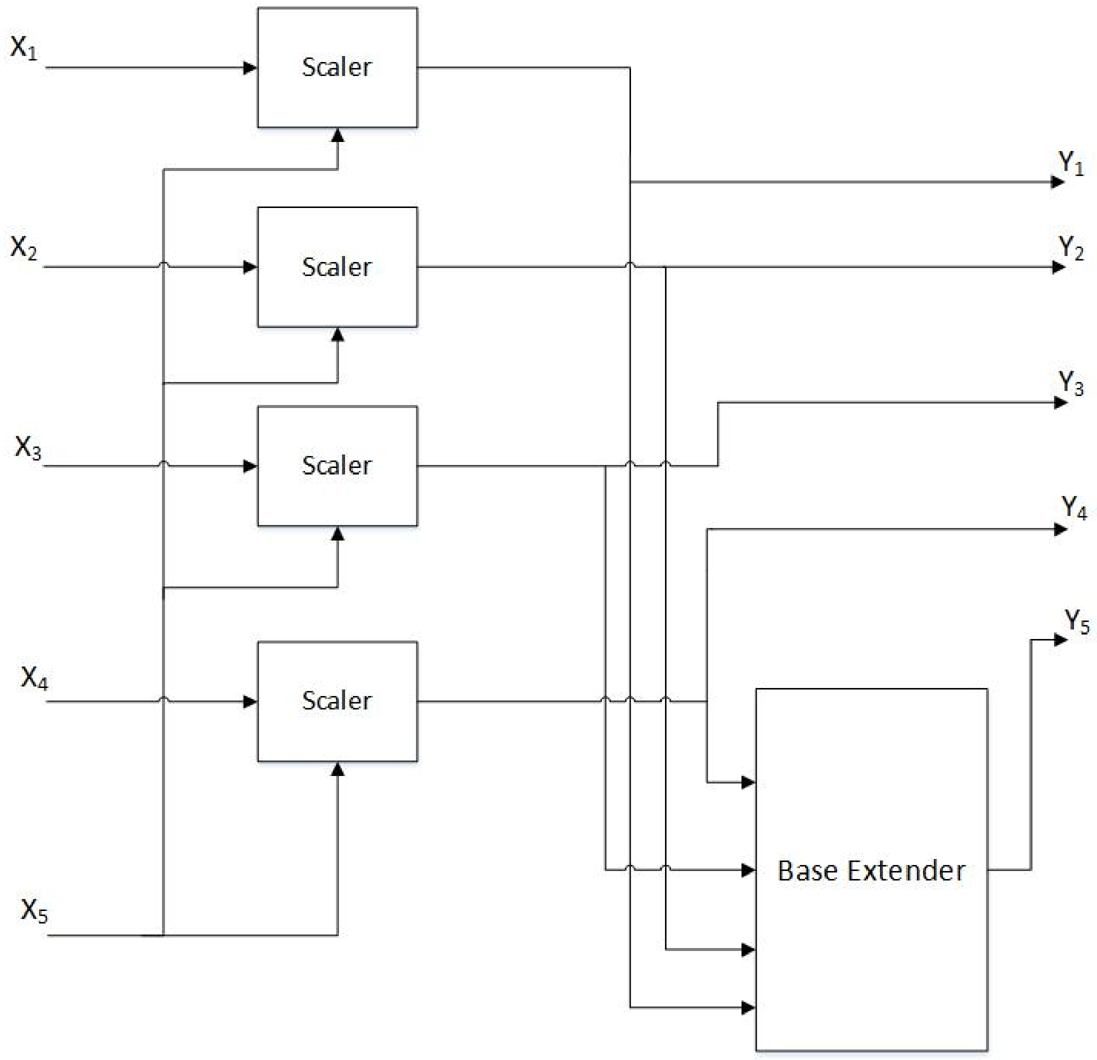

Figure 3.6: Block diagram of implemented scaler using a base extender to get the scaled value of the final residue

same way. The final sum of products is implemented as several look up tables for the products and a modulo adder tree for the sum. The modulo adder in the adder tree is on the same form as in figure 3.3.

The block diagram of the entire scaling operation is shown in figure 3.6. One can see the two steps this operation is divided into, the scaling of the first $n-1$ residues and the base extension to get the scaled value of the final residue.

The moduli used in the scaler is $\{3,4,5,7,11\}$. 11 is used the scaling factor in this scaler. This is because 11 is the largest of the moduli and therefore the scaling operation can be used less frequently. This reduces the overhead from this operation. This means that the residues associated with the moduli $\{3,4,5,7\}$ is calculated using equation 3.9. Similarly is the residue associated with the modulo 11 calculated using base extension. One thing to note, in this modulo set, is that modulo 4 arithmetic is easier to implement than modulo arithmetic using the other moduli. For this reason the residues associated with this modulo are used in the longest calculation chain for the base extension. That means, looking at figure 2.1, the rightmost column. The calculations associated with the other moduli is done in the shorter chains.

3.5 Complex Filter

A complex filter is implemented to explore how QRNS can be used to improve the performance of complex mathematics. The complex filter will be entirely implemented using the quadratic residue number system. The main idea is, that the complex multiplication can be implemented using only two multipliers, as opposed to 4 or 3, depending on implementation, using conventional complex numbers. As the multipliers represent a big part of both the complexity and power usage of the system, the reduced numbers of multipliers will cause considerable benefit in both areas. However, the disadvantages regarding the less flexible choice of moduli and the extra conversion stage will reduce this benefit.

One complex filter is implemented using the moduli set $\{13,17,37\}$. This set has relatively large moduli values compared to those used in the conventional FIR filter, due to the limitations in the choice of moduli. The j values for this set is $\{8,4,6\}$, which are used in the conversion to and from QRNS.

The filter itself is constructed the same way as the conventional FIR filter. The complex and the complex conjugate values, for each of the residues, are calculated in separate paths. This means there are 6 calculation paths, which are the complex and complex conjugate path for each residue. The hybrid RNS/binary filter implementation is also used for this filter.

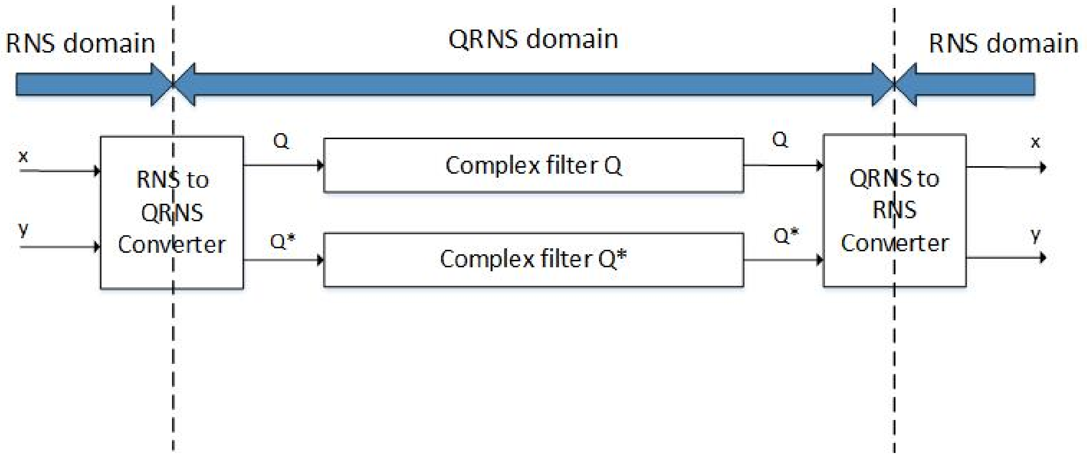

Figure 3.7: Block diagram for a complex filter using QRNS

Figure 3.7 shows how a subfilter for one residue of a complex FIR filter is implemented. For the whole filter, several of these subfliter are run in parallel, one for each residue of the RNS code. In this implementation the complex RNS number $x_i + jy_i$ is converted to the QRNS form (Q, Q^*), does the filtering in two different paths, one for Q and one for Q^*, and converts back to the complex RNS form. From this figure it is evident that the way the Q and Q^* part are run as two different filters differs from how it is done in a regular complex filter. In a regular complex filter the real and imaginary parts of the output is dependent on the both the real and imaginary parts of the inputs. For this reason, one can not run the complex filter as two separate paths using conventional complex RNS.

3.6 RNS to QRNS Converter

To use the quadratic residue number system, the signal has to be on QRNS form. For most cases this means that one has to convert it to QRNS form, either from a RNS form or from a binary form. As QRNS is just a special case of the residue number system, the conversion to QRNS from the binary form goes via RNS. Thus both when converting from RNS and binary form, one has to do the RNS to QRNS conversion.

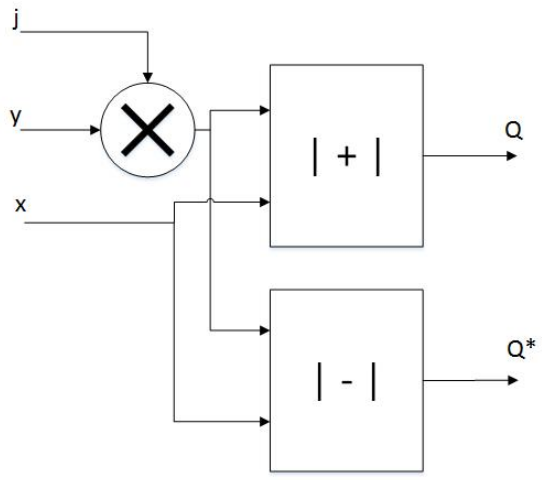

Figure 3.8: Block diagram RNS to QRNS converter

This is because the quadratic form only is valid in the RNS space, due to the imaginary value $j^2 = -1$ can not be found in using conventional binary numbers. As the binary to RNS conversion is described in another chapter, only the RNS to QRNS conversion implementation is discussed in this chapter.

The implementation of the conversion comes straight from how QRNS are defined. QRNS is defined as the set (Q, Q^*), where Q and Q^* are defined as

$$Q = x_r + j \cdot x_i \tag{3.10}$$

$$Q^* = x_r - j \cdot x_i \tag{3.11}$$

in which x_r and x_i are the real and imaginary part of the complex signal X. The Q and Q^* parts are, as usual in the RNS domain, are divided in n different residues, where the j value is unique for each of the residues in the residue number. Therefore one implementation of each of these two equations has to exist for each residue.

This equations are implemented directly using look up tables and adders. The inputs are x_r and x_i, $j \cdot x_i$ is calculated using look up tables with x_i as input and the result of $x_r + j \pm x_i$ is calculated using an adder or a subtractor. Both the complex and the complex conjugate result use the same $j \cdot x_i$ and therefore the same look-up table can be be used in both calculation. For the final adding or subtraction of the two operands, a separate adder subtractor is needed for each Q and Q^*. In total, to convert the complex RNS values (x_r, x_i) to the QRNS values (Q, Q^*), a $x_{i\,\max}$ sized look-up table, one adder and one subtractor are needed.

3.7 QRNS to RNS converter

To make use of the QRNS values outside the QRNS domain, one has to convert the values back from QRNS to RNS and therefore a QRNS to RNS converter is needed. The reasoning to why the the conversion is done from QRNS to RNS and not to binary is the same for the QRNS to RNS converter as for the RNS to QRNS converter. As with the input- and output converter, there are cases where one has to do the RNS to QRNS conversion but not the QNRS to RNS conversion and the other way around.

The formula for converting is explained in chapter 2.3. This formula is implemented using look-up tables, having the result from the modular addition or subtraction $\left|Q \pm Q^*\right|_{m_i}$ as address. This means that the size of the tables is equal to m_i. The result associated with one modulus is independent of the other modulus and the parallelism in the filters is therefore preserved in the QRNS to RNS converter. Using this implementation one need one modular adder, one modular subtractor and two m_i entry look up tables for each residue in the residue number. Due to this simplicity, the overhead by introducing a QRNS to RNS converter is extremely low and, due to only needing adders and look up tables, the reduced number of multipliers needed for complex multiplication in the QRNS domain is believed to outweight this overhead after only a few taps of a complex filter.

Chapter 4

Simulations and Results

This chapter presents the simulation setup used for all simulation. Afterwards it goes through the tree main classes of simulations done. These is the simulation of the regular FIR filters, simulation of the quadratic FIR filter and the simulation of the scaler. For each of these classes of simulation, the specific simulation setup is explained and the results from these simulations are presented. For the regular FIR filter, a model of both the are usage and the power consumption is presented and it is discussed how well this model fits the simulation results.

4.1 Simulation Setup

The goal of the simulation is to produce power and area data of the designs described in chapter 3. The data is used to discuss the pros and cons of RNS contra other number systems. To achieve data that is good enough to be used in this discussion, the simulations have to be set up in a way that makes comparison between different implementations and number systems possible. This requires the use of an as similar as possible setup for each of the simulations. This thesis focuses on how the RNS can be used to improve the implementation of the receiver of a radio, therefore the designs using RNS are compared to the types of designs they are intended to replace. As 2's complement binary is the most used number system in digital electronics, the reference design will be using this number system. The reference designs are to solve the same task as the RNS designs and from the results of the simulations one will therefore be able to directly tell whether this task is best implemented using this par-

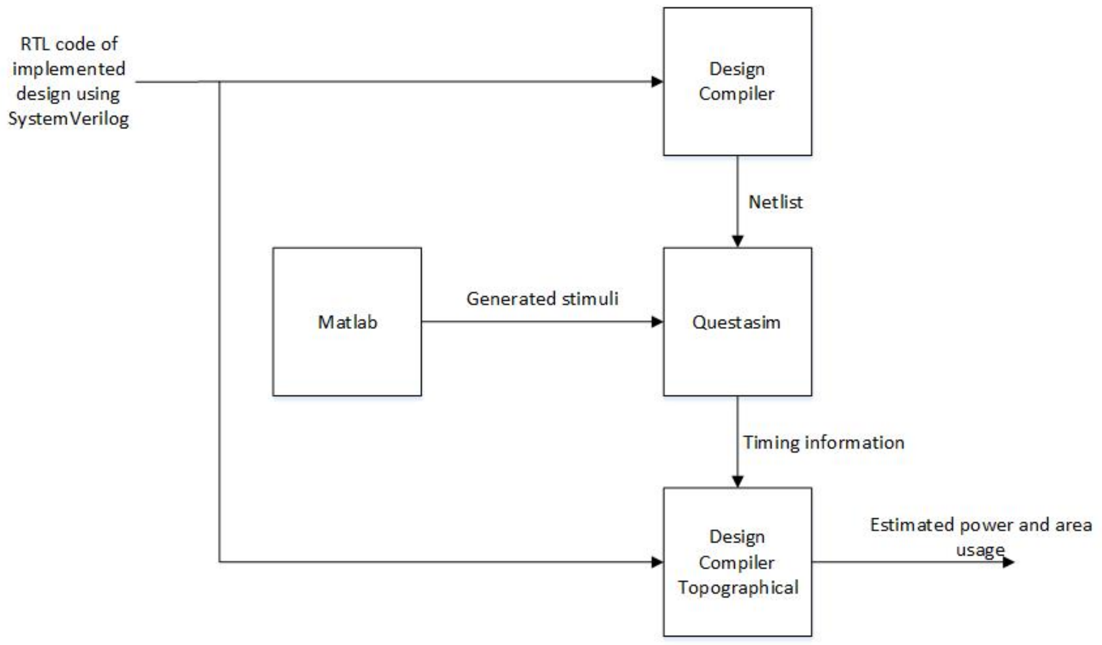
RTL code of implemented design using SystemVerilog
Design Compiler
Netlist
Matlab
Generated stimuli
Questasim
Timing information
Design Compiler Topographical
Estimated power and area usage

Figure 4.1: Simulation setup

ticular RNS implementation or binary implementation. However, the field of digital radios is a very mature field and thus there is a lots of different ways to design a radio receiver or the parts that makes up a radio receiver, excluding the choice of number system. In this thesis very straight forward implementations are used in the reference design used in the implementation. Despite this, it is believed that results will have some validity for implementations using techniques which are not discussed in this thesis.

The simulations are done on netlists, to get more accurate results. The netlist are generated using the tool Design Compiler[1] by Synopsis. The technology library is for 55nm in the slow corner at 125 °C. To get the correct switching activity in the estimations, a simulation is done on using the tool Questasim[2] by Mentor Graphics. From this simulation one will get the static switching information of this particular circuit. The stimuli used in the simulation is generated using Matlab[3]. This is done to achieve a better control over how the generated stimuli looks in therms of value range and shape, to get equivalent stimuli for the RNS design and the reference de-

[1]Version 1409-sp4
[2]Version 10.4c
[3]Version r2013a

sign and to calculate the expected output, which is used to verify correct behavior of the circuit. When the timing information of the design is generated, the Design Compiler tool is used again, this time in topographical mode. This uses the RTL code as input. The difference by applying this compared to the first use of Design Compiler is that the switching information is added this time. From Design Compiler we will get estimation of the power and the area usage of the design, which is the result that is presented in this chapter and will be used further in the discussion part of this thesis. The setup of the simulation can be seen in figure 4.1.

4.2 Simulation of FIR Filters

The implemented FIR filters are run through synthesis and simulation to get their power and area data. The filters are implemented as described in chapter 3.1. The implementations are simulated using different dynamic ranges and different filter orders. The different dynamic ranges requires different sets of moduli, which makes the changes to the implementation non trivial. This means that the results, from the filters with different dynamic ranges, are not as easily comparable with each other in the sense that the power usage and area usage increase in a predictable manner as the dynamic range increases. However, it is assumed that there will be a general trend with increased area and power usage for the larger dynamic ranges. The filter order, also called number of taps, is highly parametrizable. Each tap of filters, using the same moduli set, is designed the same way and therefore it is assumed that the area and power usage will increase in a very predictable manner as the filter orders increase. The results from the different filter orders, for filters having the same dynamic range, is assumed to have a very high predictive power in regards to filter orders outside the scope used in the simulations. The results from the different dynamic ranges will have a lower predictive power.

To make the results from the simulations of these designs comparable to the designs they are to replace, the simulations is set up so that the RNS implementation and the reference implementation do the same calculation only the way the calculation is done is different. For the FIR filters this means the input and the output will be in binary form. In the RNS filter the signal has to be converted to RNS before the filtering and from RNS to binary after the filtering. To take into account the different ways to implement an input and output converter and to make the changes in power

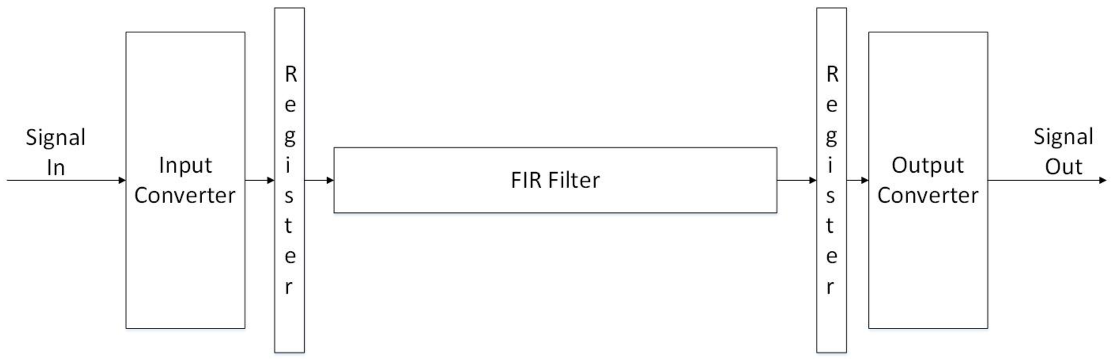

Figure 4.2: Setup of RNS FIR filter simulation

usage, due to increased filter order, independent from the types of converters used, a pipelineing stage is inserted between each of the input and output converters and the filter. This will prevent glitching caused by the filter to propagate through the output converter and glitching caused by the input converter to propagate through the filter. By doing this the results will be applicable for a more general case, as it is independent of the types of input and output converter used. Figure 4.2 shows how the simulation setup is for the RNS FIR filter.

The reference FIR filter is constructed in a straight forward manner. Each of the references is constructed to have a dynamic range, which is as close as possible to that of the RNS FIR filter it is a reference for. Since the reference design is implemented using 2's complement binary, it does not need any input and output conversion steps and therefore no pipeline steps as well. Figure 4.3 shows the setup of the simulation of the reference FIR filter. One can see that the lack of input and output converters makes the FIR filter the only component of the setup. The signal out and signal in in this figure are exactly the same as for the RNS FIR filter, thus both can directly replace each other without altering the logic behaviour of the circuit. In this setup all of the area and power usage will come from the filter compared to for the RNS filter, where the input and output conversion accounts for a large part of energy and area usage.

4.2.1 16-bit FIR Filter

The first dynamic range used in the simulation is the one that can be represented by 16 bits using the 2'complement binary representation. The moduli set used for RNS

Signal In
FIR Filter
Signal Out

Figure 4.3: Setup of the reference FIR filter simulation

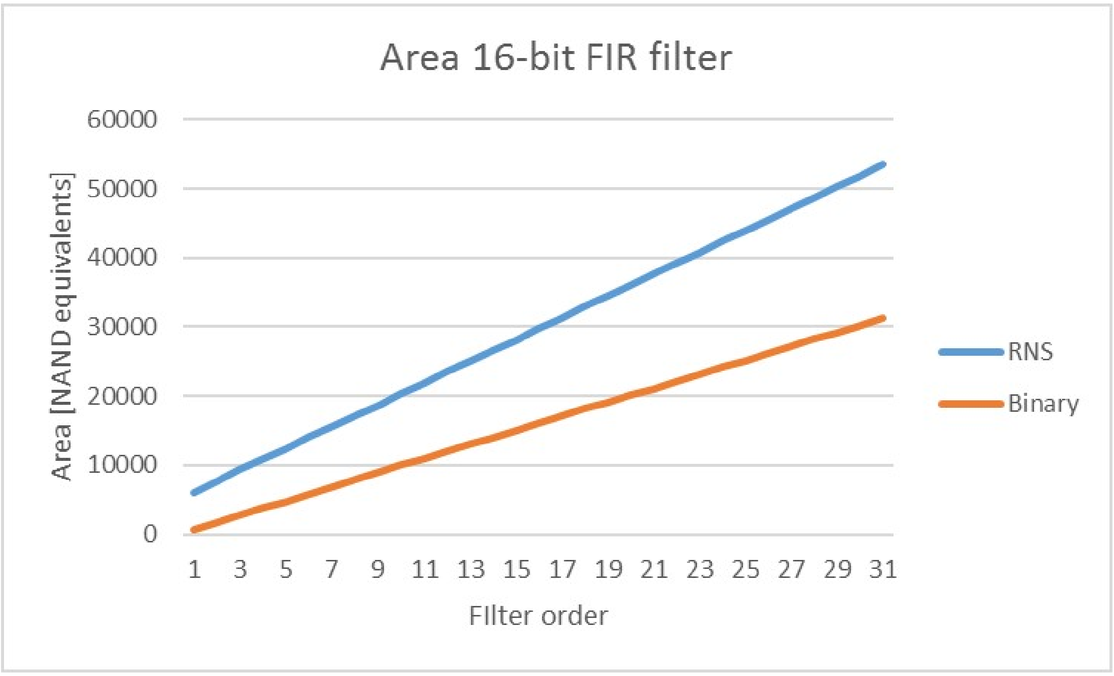

Figure 4.4: Area 16-bit FIR filter

implementation with the same range is 5,7,11,13,16, providing a dynamic range of 80 080 compared to the dynamic range of 65 536 of a 16-bits 2's complement representation. These are comparable filters, as the RNS has a large enough dynamic range to represent the same amount as the binary one. These filters will be referred to as 16-bit binary and 16-bit RNS filters from now on. The area usage of both these filter implementations are shown in figure 4.4.

It can be seen that the area increases in a linear fashion, as the filter order increases, for both the binary and the RNS filter. Using linear regression the growth of the Area of the RNS filter can be described using the function:

$$y = 1570x + 4580 \tag{4.1}$$

The same can be done for the binary filter using the function :

$$y = 1010x - 210 \tag{4.2}$$

The constant and non constant value in each of the equations is the area occupied by each tap of the filter while the constant value of the RNS filter is the area used by the input and the output converter. The negative offset value of the binary filter is due to the growth of the area not being perfectly linear and is so small that it can be discarded.

The area per tap is significantly lower for the binary filter than for the RNS filter. This is contrary to the assumption stated in chapter 3.1, where it is argued that the use of many smaller multipliers will have a smaller area usage than one big, due to the area of the multipliers is proportional with the square of the input word. However, the lower area per tap for the binary filter can be due to the fact that the word size of the reference filter is so low, so that the effect increased parallelism in the RNS domain is not large enough to offset the increased complexity of using hybrid RNS/binary arithmetic compared to regular binary arithmetic. The overhead by input and output conversion in the RNS filter is 4580 NAND equivalents. This is equal to about 3 taps of the filter, which means that the overhead by conversion is not too big for filters of high order if a sufficient large gain per tap can be achieved.

The power consumption of the filter is shown in figure 4.5. The RNS filter has a power consumption, which has super linear proportion with the filter order. Due to the power consumption in each tap of the filter, one would expect a linear increase in power consumption relative to the filter order. The super linearity is due to two factors. One is, due to the increased length of the filter, glitches caused in earlier parts of the filter will propagate longer and the dynamic power consumption increases. The other one is the glitches this tap causes and propagates through the remainder of the filter. These two are in essence the same thing, where the difference is which tap is used as reference. The internal power consumption for each tap is linear, while the glitching that propagates through the following taps is proportional with the number of taps multiplied with the number of following taps. Because of this one can divide the power consumption up in two parts: the internal part and the part caused by glitch propagation.

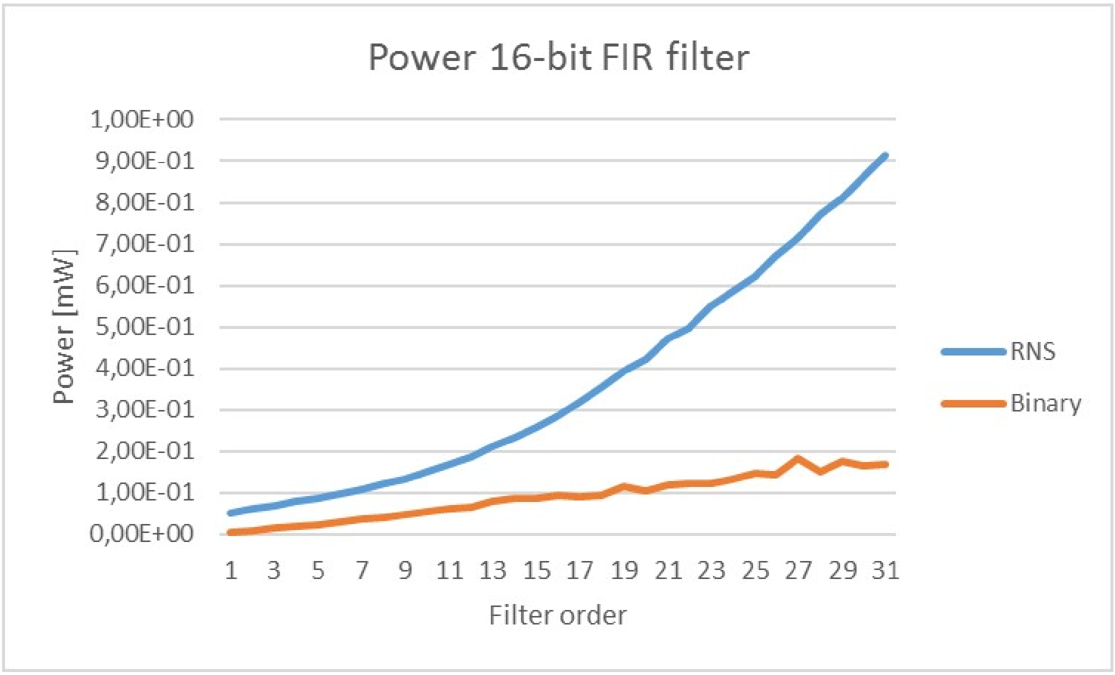

Figure 4.5: Power 16-bit FIR filter

$$P_{internal} = a_{internal} * n \tag{4.3}$$

$$P_{glitching} = \sum_{i=0}^{n} a_{glitching_i} * i \tag{4.4}$$

Here n is the filter order and $a_{internal}$ and $a_{glitching}$ are the internal power coefficient and glitching power coefficient. The internal power coefficients represents the power consumption a tap uses due to glitching caused internally in this tap, result propagation and static power usage. The glitching power coefficient represents the amount of glitching caused on the output of a tap multiplied with the average energy consumed per glitch for each tap these glitches propagates through. Equation 4.4 can be simplified by assuming that the glitching constant is the same for all the taps in the filter and for all filter orders. This assumption is not necessarily true because each glitch on the input of a tap does not always propagate to the output. Therefore a glitch will cause a lower power consumption if the distance between where the glitch is caused and the tap is larger than in the cases where distance is smaller. The simplified from is given as

$$P_{glitching} = a_{glitching}\frac{1}{2}n(n+1) \tag{4.5}$$

If it is taken into account that a glitch on the input of a tap may cause additional glitches to occur on the output of the tap, the glitches on the output of one tap can be described as a factor p times the glitches on the input. The glitch on the output can thus be described as:

$$g_{out} = g_{in} * p \tag{4.6}$$

If this equation is applied to several taps in succession, the total power consumption caused by a glitch on the output of one tap of a filter can be described by the following equation:

$$P_{glitch} = \sum_{i=0}^{n-1} g \cdot p^{i} \cdot k \tag{4.7}$$

where g is the amount of glitches on the output of the particular tap, n is the amount of taps after this tap and k is the power consumption a particular glitch causes. Considering that each tap of the filter causes glitching to propagate through the following taps, the total power consumption due to glitching is the sum of the power consumption due to glitching for each tap in the filter. This gives the following equation

$$P_{glitching} = \sum P_{glitch} = \sum_{i=0}^{n-2}\sum_{j=0}^{j=i} g \cdot p^{i} \cdot k \tag{4.8}$$

This component of the power consumption will lead to the overall power consumption having an exponential growth as this component becomes more significant than the linear component $P_{internal}$. However, in figure 4.5 one can notice that the equation becomes linear as the filter size approaches 31, which is the max filter size used in the simulations. This indicates that the glitches does not propagate through all the following i taps of the filter or that the propagation of glitches is reduced as it goes through several taps. This may be due to some glitches being suppressed by other glitches, which will have a large effect when the filter sizes are big. One way to model it is solely to have the glitching propagate through a max number of taps. Another is to have a factor that approaches zero when the tap is far from where the glitch is caused. Using the first method the model will look like equation 4.9. An equation using the second method is not derived.

$$P_{glitching} = \sum P_{glitch} = \sum_{i=0}^{n-2} \begin{cases} \sum_{j=0}^{j=i} g \cdot p^i \cdot k, & \text{if } i < L \\ \sum_{j=0}^{j=L} g \cdot p^i \cdot k, & \text{otherwise} \end{cases} \tag{4.9}$$

If we also consider a constant power consumption due to the input and output conversion, we have two equations to describe the power consumption of the RNS filter given a basic form of

$$P = P_{internal} + P_{glitching} + P_{conversion} \tag{4.10}$$

Namely the one using the simplified glitching constant

$$P = a_{internal} * n + a_{glitching} \frac{1}{2} n(n+1) + P_{conversion} \tag{4.11}$$

in which $a_{internal}$, $a_{glitching}$ and $P_{conversion}$ are constants. If one writes the $P_{glitching}$ part of the equation as as

$$\frac{1}{2} a_{glitching} n^2 + \frac{1}{2} a_{glitching} n \tag{4.12}$$

one can write equation 4.11 the following way

$$P = \frac{1}{2} a_{glitching} n^2 + (\frac{1}{2} a_{glitching} + a_{internal}) n + P_{conversion} \tag{4.13}$$

Here the equation is on a second order equation, for which the constants can be found using a regular second order regression.

Figure 4.6 shows the fitted second order equation to the estimated power of the RNS filter. The equation describing the fitted line is

$$P = 0,00087 n^2 + 0,00097 n + 0,055 \tag{4.14}$$

This equation fits well with the measured results for most of the samples. However, the slope does not match for the filter orders approaching zero, meaning the first order component of the equation is not correct and suggests that the power consumption can not be described by a second order equation.

Figure 4.7 shows a fitted graph using the equation 4.9 to describe the power con-

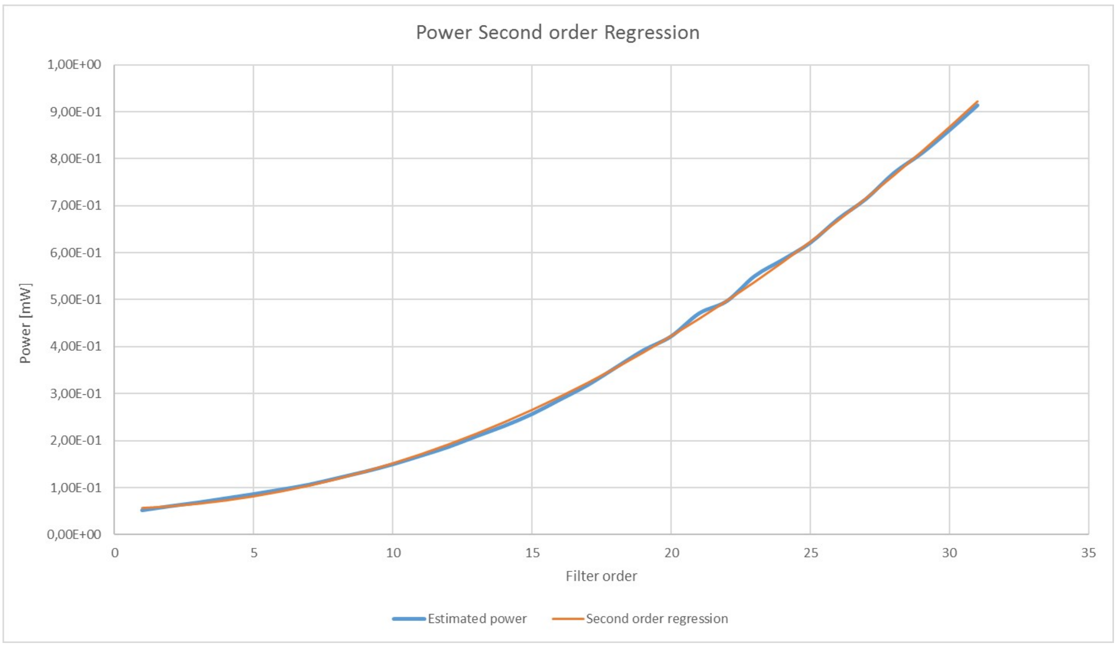

Figure 4.6: 2nd order regression of power consumption

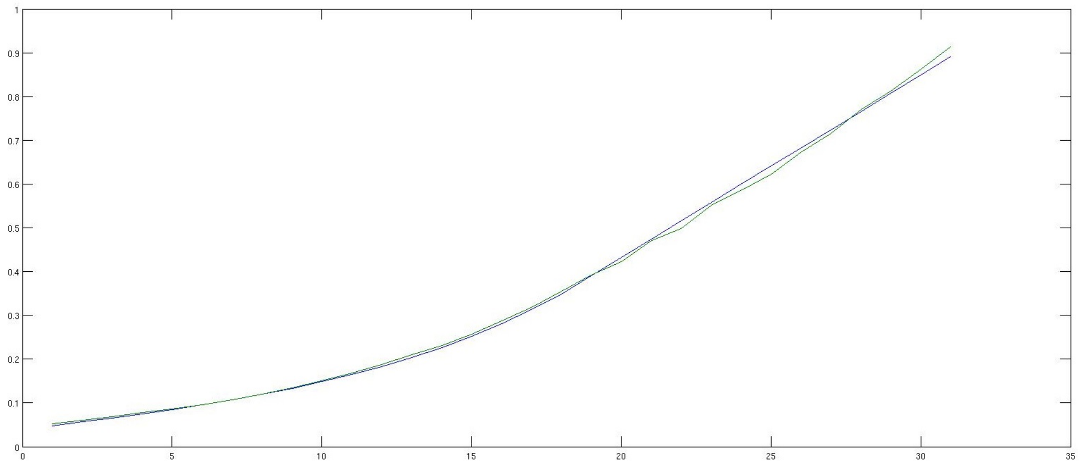

Figure 4.7: Graph fitted to the power consumption of the RNS filter using the presented model

sumption due to glitching. The equation used to describe it is:

$$P = 0.04 + 0.008n + 0.0004 \cdot \sum_{i=0}^{n-2} \begin{cases} \sum_{j=0}^{j=i} 1.16^j, & \text{if } j < 19 \\ \sum_{j=0}^{j=19} 1.16^j, & \text{otherwise} \end{cases} \tag{4.15}$$

It fits very well for lower filter orders but the slope is a bit off for filter orders of around 30. This is due to equation 4.15 is fully linear when n is larger than 16, while the measured results are slightly above linear. Due to this functions good fit for lower filter orders, it is more appropriate to describe the power consumption than the second order equation.

Of the suggested models of the power consumption of the RNS FIR filter, the model described by equation 4.15 fits best with the simulation result. Therefore the equation on the genral form shown in equation 4.16 will be used to model the power consumption of the RNS FIR filter in the continuation of this thesis.

$$P = K + a \cdot n + c \cdot \sum_{i=0}^{n-2} \begin{cases} \sum_{j=0}^{j=i} p^j, & \text{if } i < L \\ \sum_{j=0}^{j=L} p^j, & \text{otherwise} \end{cases} \tag{4.16}$$

Figure 4.8 shows a second order approximation to the power consumption of the reference filter. The equation used for this approximation is:

$$P = -4,6 \cdot 10^{-6} \cdot n^2 + 0.00566 \cdot n - 0.00456 \tag{4.17}$$

It is evident that this approximation is dominated by the first order component. This suggests that the glitching, on the output of each tap, does not account for a big part of the power consumption of the filter. This is a big difference from what is seen in the RNS filter, where the main part of the power consumption is perceived to be due to the glitching. This may be due to the difference in implementation between these two filters. The path, which the glitches propagate through, is the summation of the products. In the reference filter this path only consists of successive adders, while in the RNS filter this path consists of the correction elements, which purpose is to avoid overflow. The reduced complexity, in this path, for the reference filter compared to the RNS filter means that each time a glitch is generated, it causes less transistors to switch and thus less energy is consumed. Because of this a linear equation is used to model the power consumption of the reference FIR filter in the

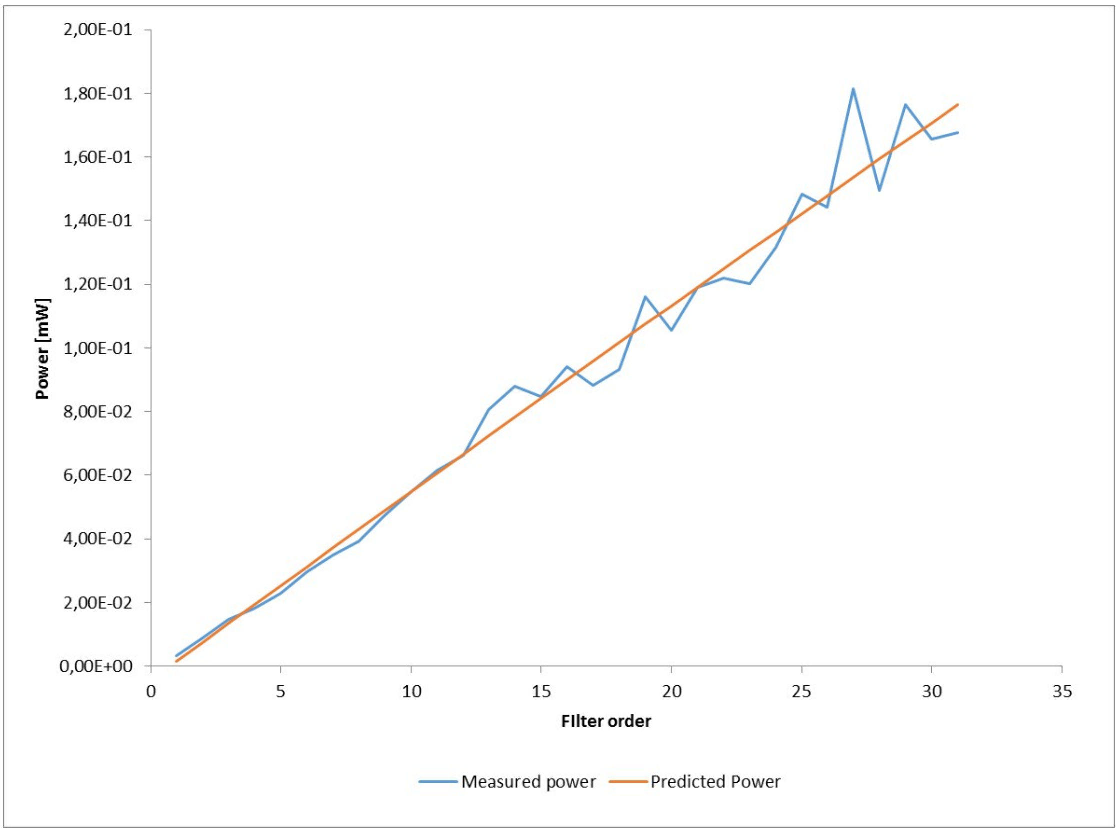

Figure 4.8: Regression of power consumption of reference filter

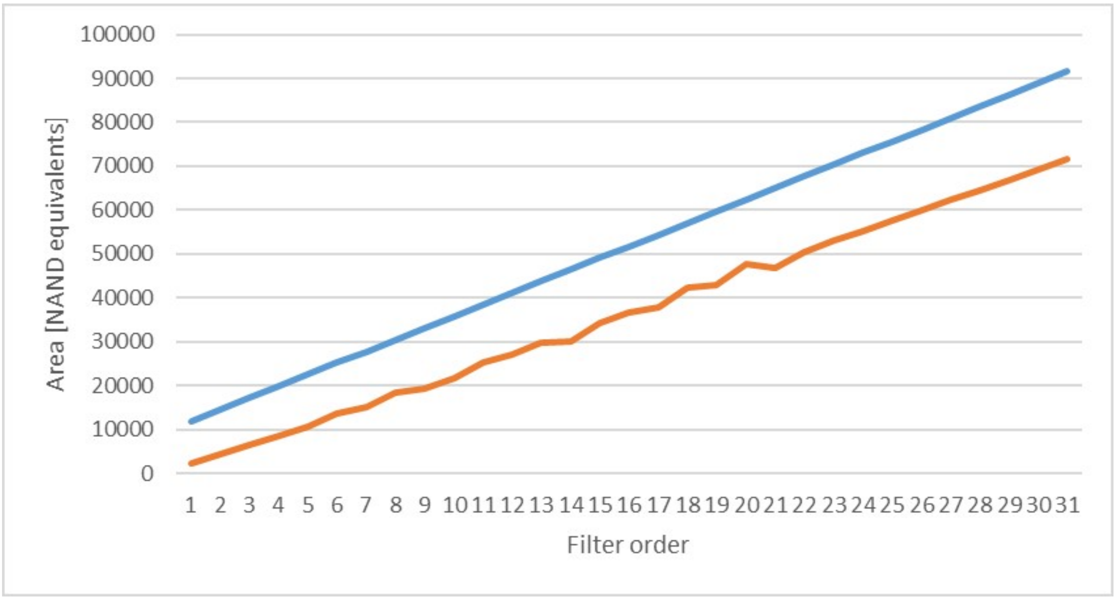

Figure 4.9: Area 24-bit FIR filter

continuation of this thesis.

4.2.2 24-bit FIR Filter

For the RNS filter with a 24-bit dynamic range, the moduli set $\{5,7,9,11,13,16,17\}$ is used. The dynamic range of a RNS number using this moduli set is 12252240 compared to a 24-bit binary number, which has 16777216. The binary 24-bit number has a slightly higher dynamic range, but the ranges are close enough to make this filter comparable to a 24-bit binary filter. Therefore a 24-bit binary filter is used as a reference for this particular RNS filter.

Figure 4.9 shows the area usage for both the RNS filter and the reference filter. Both the RNS and the reference filter has a linear increase in area usage as the filter order increases. The areas of the RNS filter can be described by the equation:

$$A = 2655n + 9253 \tag{4.18}$$

And the area of the reference filter can be described by the equation :

$$A = 2329n - 769 \tag{4.19}$$

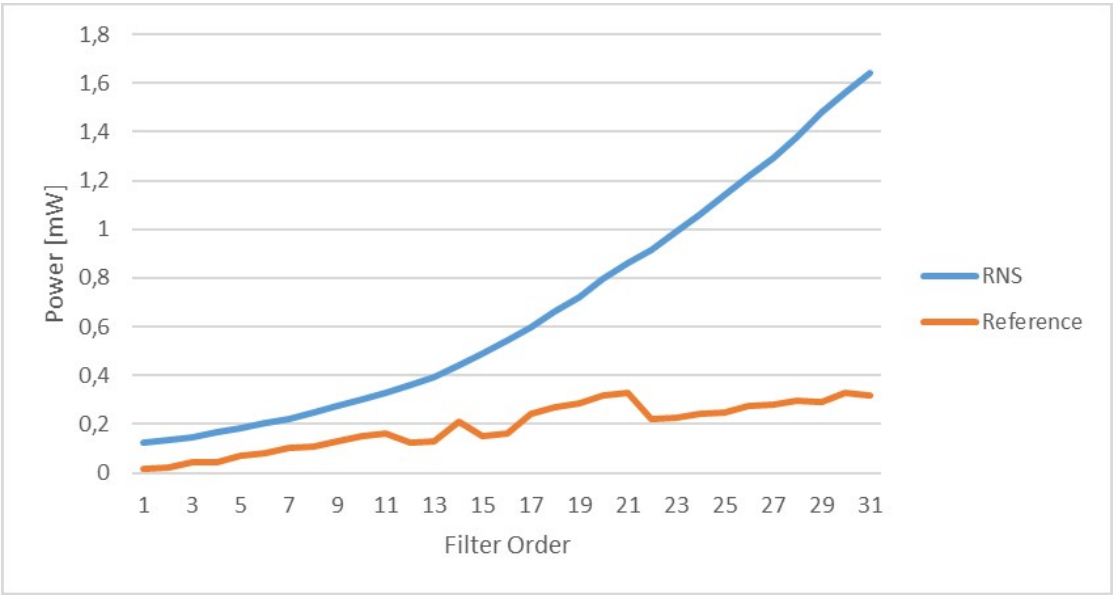

Figure 4.10: Power 24-bit FIR filter

In both equations A is the area used and n is the number of taps. For these filters the reference has a slightly lower area per tap than the RNS variant. Thus the increased parallelism is not high enough to offset the increased complexity by using hybrid RNS/binary arithmetic for the filter with 24-bit dynamic range. The total overhead by binary to RNS- and RNS to binary conversion is 9253 NAND equivalents, which is somewhere between 3 and 4 filter taps.

Figure 4.10 shows the power usage of the RNS and reference filters. The power usage of the RNS filter has a similar form as the one with 16-bits dynamic range. The same approximation is used to fit an equation to the power data form this filter. Figure 4.11 shows the resulting graph from this approximation superimposed on the measurement data. The equation describing this approximation is

$$P = 0.11 + 0.014n + 0.00067 \cdot \sum_{i=0}^{n-2} \begin{cases} \sum_{j=0}^{j=i} 1.18^j, & \text{if } j < 17 \\ \sum_{j=0}^{j=17} 1.18^j, & \text{otherwise} \end{cases} \tag{4.20}$$

The approximation fits very well in the lower bounds but diverts from the measured data for high filter orders. This is also the case for the approximation done to the 16-bit RNS filter and is due to simplifications done in the approximation equation in this case as well.

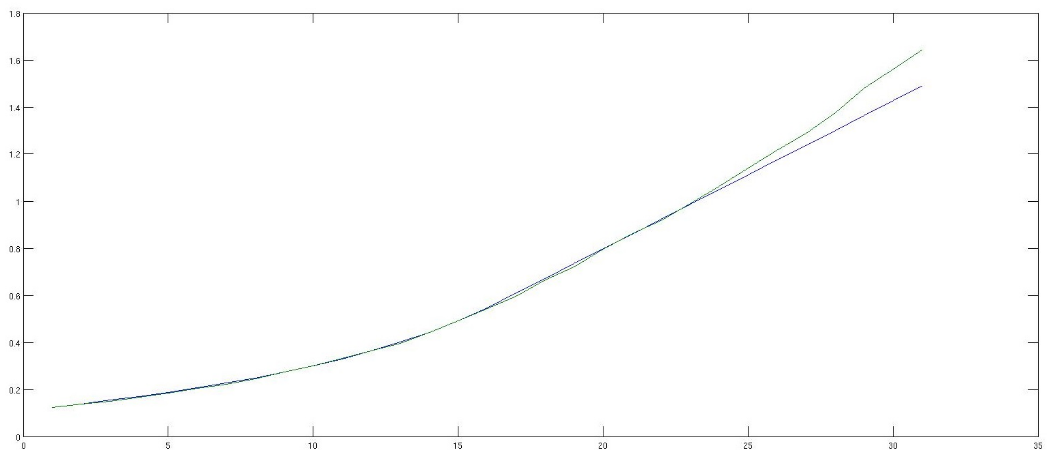

Figure 4.11: Graph fitted to the power consumption of the RNS filter using the presented model

The power usage of the reference filter is difficult to describe using an equation. One can recognize a linearity for parts of it. For example is the power consumption for the first 11 filter orders quite linear. The same is the case for the filter orders from 22 to 31. For the filter orders between 11 and 22 the power usage switches from following the same linear projection as the first 11 orders and the same as the orders from 22 to 31. The reason for this may be the inner workings of the synthesis tool used (Design Compiler) or due to the simulation setup. The first case, is unlikely, as there are no indication of differences in synthesis on the area usage graph. It is more likely that the simulation setup is faulty and thus gives wrong answers for some of the filter orders. The effect of a incorrect simulation setup, which can lead to wrong power consumption data, is that there may be a part of the signal which is not propagated through the entire circuit, which lower the total power consumption of the circuit, or that some additional toggling may be caused, which leads to a higher power consumption.

The equation describing this power consumption for the reference filter is

$$P = 0.015n - 0.006 \tag{4.21}$$

using a linear approximation of the power data from filter orders 1 to 11 and

$$P = 0.012n - 0.043 \tag{4.22}$$

using a linear approximation of the power data from filter orders 22 to 31. Looking at these two equations, the first one seems like a better approximation, as the constant component is closer to zero compared to the second one. This is the case if it is assumed that the power consumption will increase linearly with the filter order, as is the case with the 16-bit filter.

The power consumption of the RNS filter has a linear component of 0.014 mW per tap and the power consumption of the reference filter has a linear component of 0.015 mW per tap. This means, given the assumptions made are correct, and if the power consumption due to glitch propagation is excluded, the growth of power consumption is lower for the RNS filter than the reference filter as the filter orders increase.

4.2.3 32-bit FIR filter

For the RNS filter covering a 32-bit dynamic range, the moduli set $\{3,5,7,13,16,17,19,23,29\}$ is used. The dynamic range of a RNS number using this moduli set is 4705231440 compared to that of a 32-bit binary number, which is 4294967296. The RNS number has a slightly higher dynamic range, but the difference is small enough to make filters using these two number systems comparable.

Figure 4.12 shows the area usage of these filters as the filter orders increases. As the case is for the 24-bit and 16-bit filters, have the both the 32-bit binary and "32-bit" RNS filter a linear increase in area as the filter order increases.

The linear equation describing the relationship between area and filter orders for these two filters are as following:

$$A = 3824n + 19814 \tag{4.23}$$

for the RNS filter and

$$A = 3904n + 78 \tag{4.24}$$

for the reference filter. The RNS filter has a slightly lower area per tap than the reference. The difference in area per tap is only 80 NAND equivalents. To make up for the overhead by input and output conversion 248 taps is needed. Therefore, for most practical applications, the reference filter will have a lower area usage. However, the

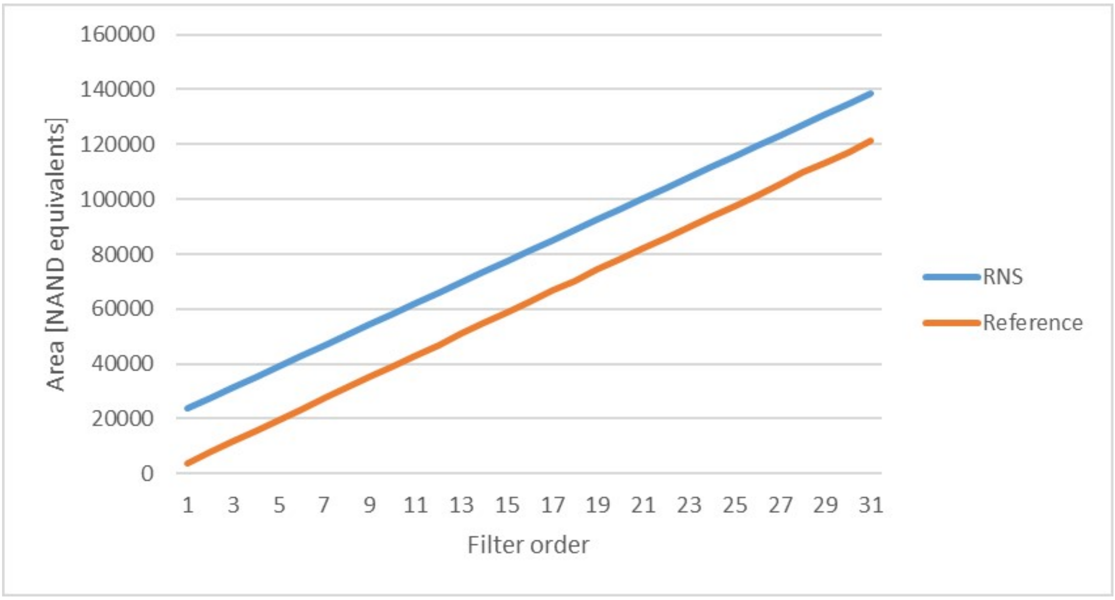

Figure 4.12: Area 32-bit FIR filter

difference in area between the two is low for most filter orders. For example for filter order 16 the RNS filter has an area usage, which is only 30% higher than the reference filter. This means that, in case the gains in other areas is high enough, the higher area of the RNS filter is not decisive for the choice of filter type.

Figure 4.13 shows the power consumption of these two filters. Both the graph for reference and RNS filter have a similar shape as for the other dynamic ranges. Using the same approximation for this RNS filter gives the graph seen in figure 4.14.

The equation describing this graph is :

$$P = 0.31 + 0.022n + 0.0014 \cdot \sum_{i=0}^{n-2} \begin{cases} \sum_{j=0}^{j=i} 1.14^j, & \text{if } j < 18 \\ \sum_{j=0}^{j=18} 1.14^j, & \text{otherwise} \end{cases} \tag{4.25}$$

The graph for the power consumption of the reference filter can be fitted linearly using the equation

$$P = 0.016n - 0.0062 \tag{4.26}$$

The linear part of the RNS filter is higher than that of the reference filter. This is contrary to what was the case for the 24-bit filters, for which the linear component of the

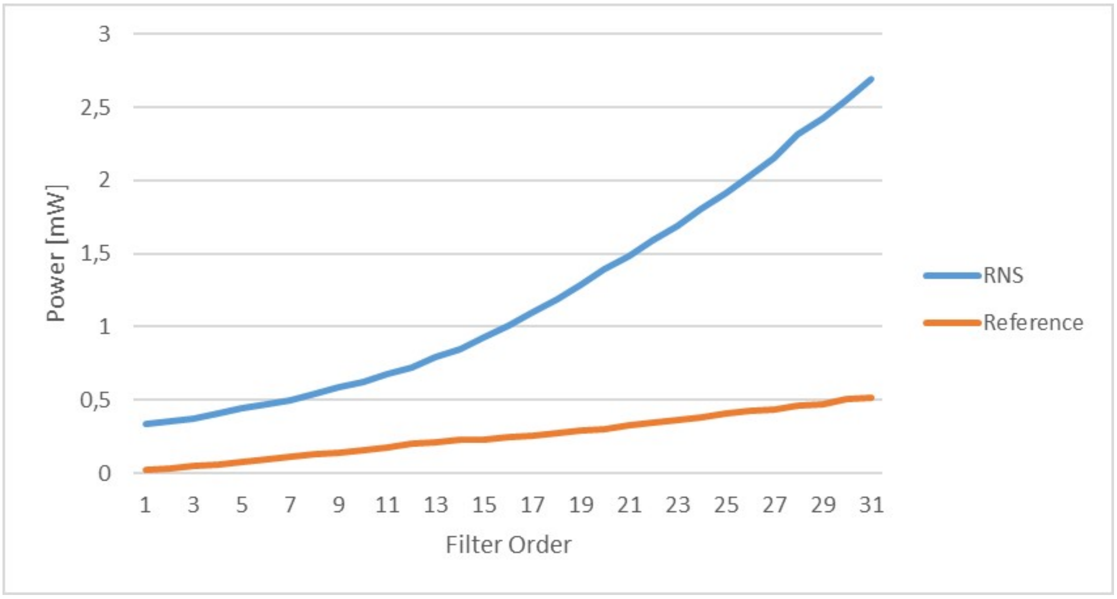

Figure 4.13: Power 32-bit FIR filter

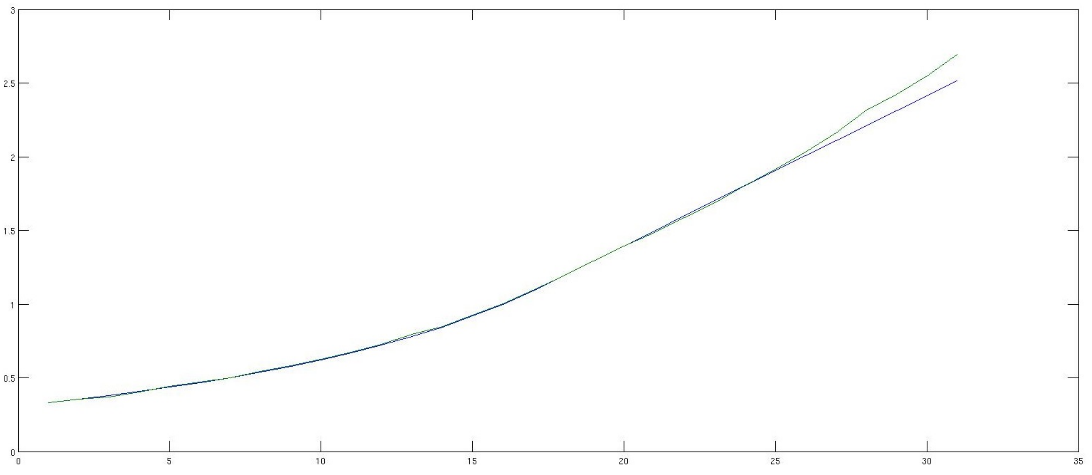

Figure 4.14: Graph fitted to the power consumption of the RNS filter using the model presented in this thesis

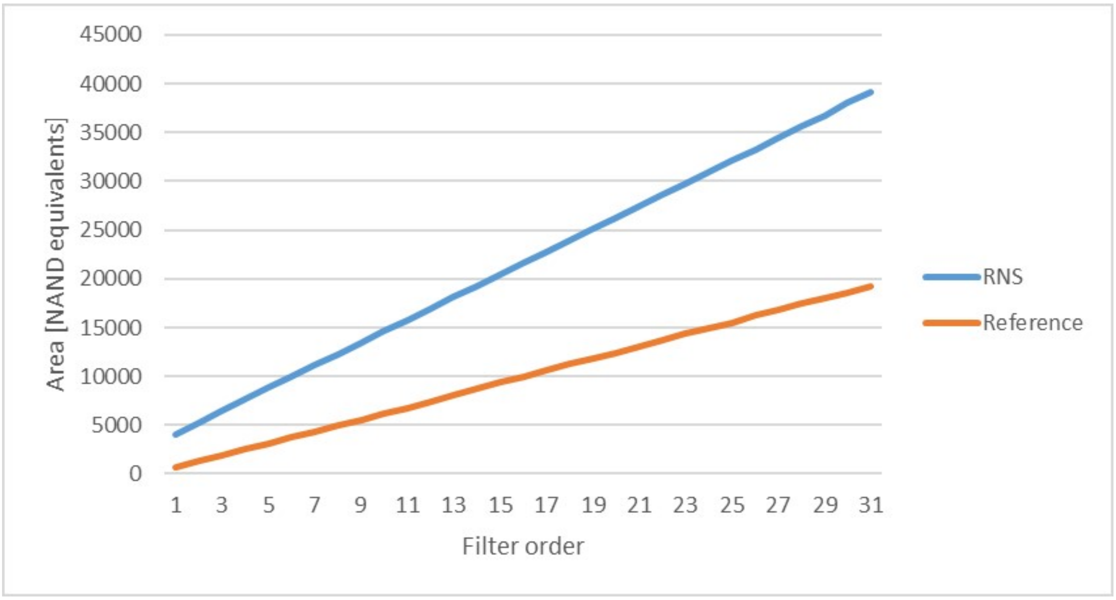

Figure 4.15: Area 12-bit FIR filter

RNS filter was lower than that of the reference. One would assume, given the higher gain due to parallelism for higher word widths, that the RNS filter in this case would also have lower linear power component than the reference and that the difference would be larger.

4.2.4 12-bit FIR filter

For the RNS filter covering a dynamic range corresponding a 12-bit binary number, the moduli set $\{3, 4, 5, 7, 11\}$ used. A RNS number using this moduli set has a dynamic range of 4620 compared to a 12-bit number, which has a dynamic range of 4096. The RNS number has a slightly higher dynamic range, but the ranges are so close that the number systems are comparable.

Figure 4.15 shows the area usage of the RNS and reference filters. The shape of the graphs is the same as for the other dynamic ranges and can be explained with the following linear equations.

$$A = 1167n + 2934 \tag{4.27}$$

for the RNS filter and

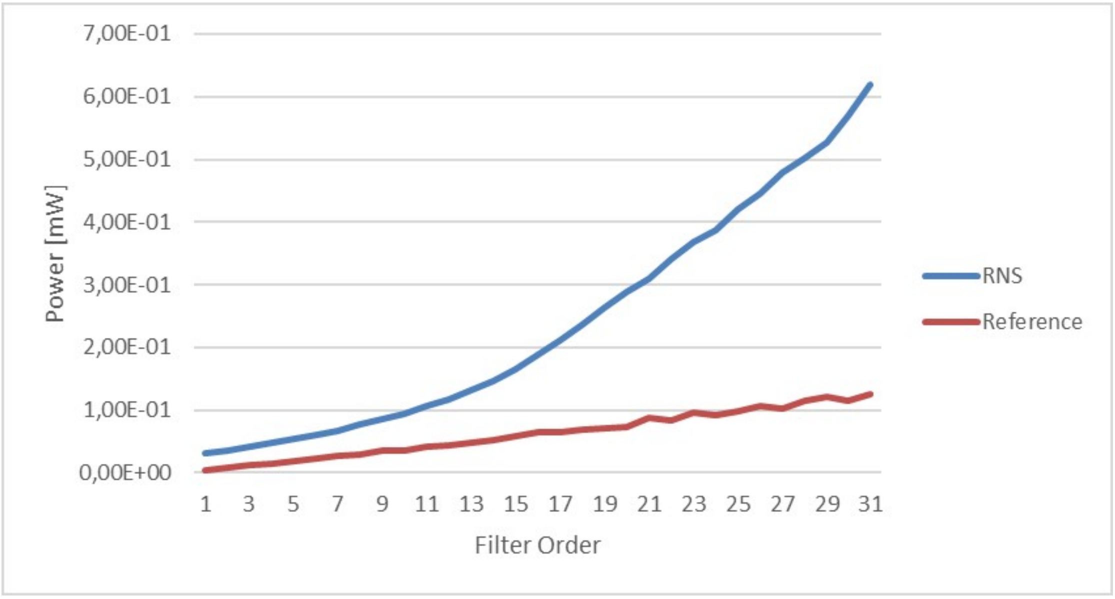

Figure 4.16: Power 12-bit FIR filter

$$A = 622n - 1 \tag{4.28}$$

for the reference filter. The reference filter has a clearly lower area usage than the RNS filter. This finding is as expected since this is the lowest dynamic range presented in this thesis and the gain in area will be most prominent where the increased parallelism of the RNS number system can be exploited most, which is for large dynamic ranges.

Figure 4.16 shows the power consumption of these filters. The shapes are also similar in this case as for the other dynamic ranges. Thus the graph fitted to the measurements of the RNS filter can be found the same way. Figure 4.17 shows this graph. The equation describing this graph is :

$$P = 0.026 + 0.005n + 0.00021 \cdot \sum_{i=0}^{n-2} \begin{cases} \sum_{j=0}^{j=i} 1.21^j, & \text{if } j < 17 \\ \sum_{j=0}^{j=17} 1.21^j, & \text{otherwise} \end{cases} \tag{4.29}$$

Likewise can the graph fitted to the measurements of the reference filter be found the same way as for the other dynamic ranges. Giving the equation

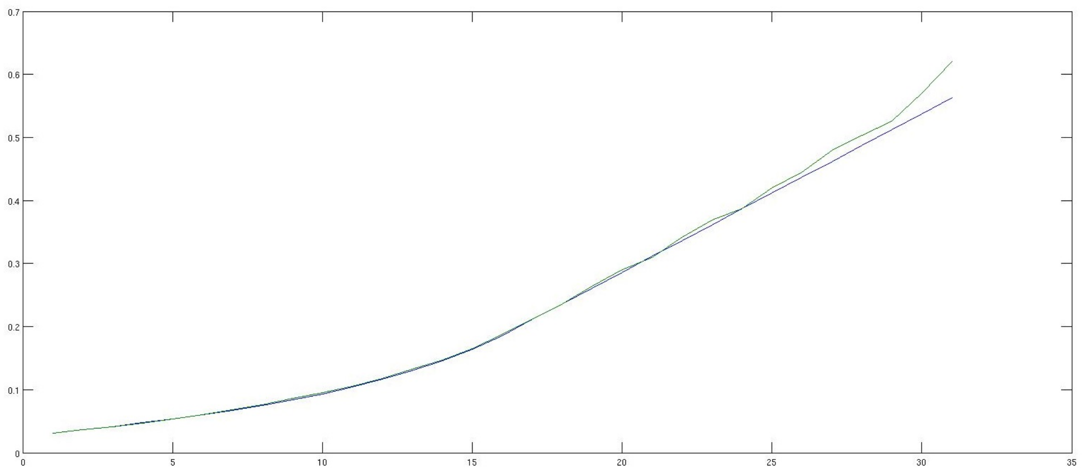

Figure 4.17: Graph fitted to the power consumption of the RNS filter using the model presented in this thesis

$$P = 0.004n - 0.0022 \tag{4.30}$$

The linear component of the power consumption is 0.005 mW per tap for the RNS filter and 0.004 mW per tap for the reference filter. Thus is the internal power usage a bit higher for the RNS filter than for the reference filter in this case.

4.3 Simulation of Quadratic FIR Filter

The quadratic FIR filter is implemented as described in chapter 3.5. The same simulation setup is used for the quadratic filter as for the regular FIR filters. Only one modulo set is used for the quadratic FIR filter simulation, namely the set $\{13, 17, 37\}$. This set gives the RNS number a dynamic range comparable to a 12-bit binary number. Due to similarities between the quadratic filters and regular FIR filters, the results from the simulations of the regular FIR filters can be compared to these results. A pipelining stage between the converters and the filter is used in this case as it is for the regular FIR filters. The motivation for doing this is the same here as in those cases. The structure of the quadratic FIR filters used in the simulation is shown in figure 4.18.

As reference a complex binary filter is used. The reference filter is constructed

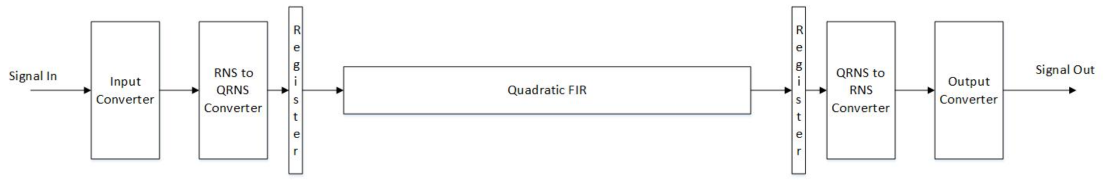

Figure 4.18: Simulation setup of the quadratic FIR filter

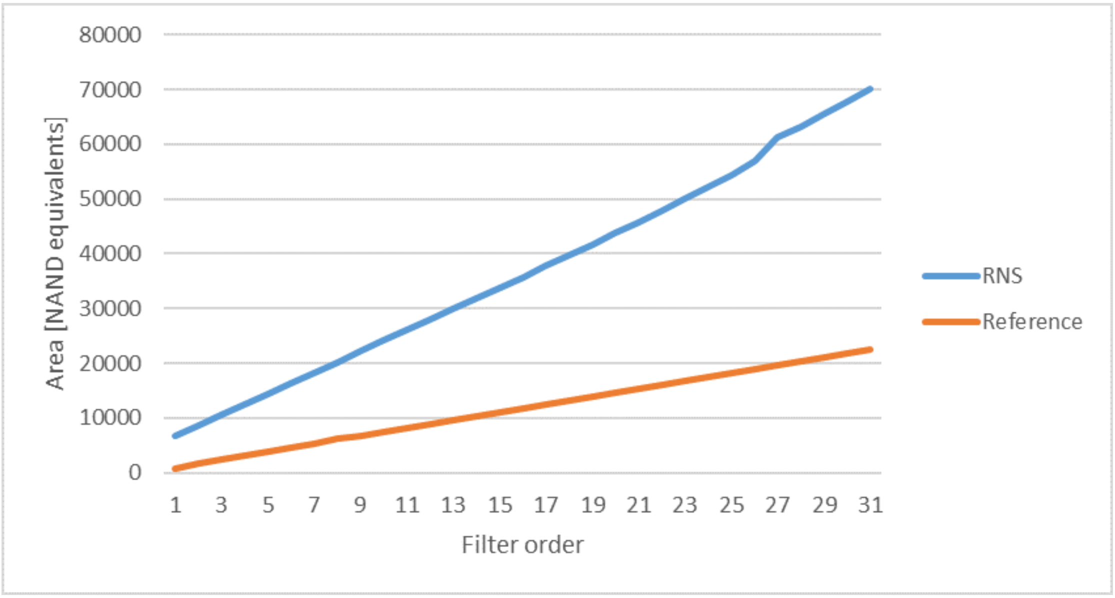

Figure 4.19: Area usage of the quadratic FIR filter

in a straight forward manner, where the general structure are the same as for the regular FIR filter. The only difference is that complex multiplication is done instead for regular multiplication.

Figure 4.19 shows the area usage of the QRNS filter and the reference complex FIR filter. As for the regular FIR filters these filters also have linear growth. However, for the QRNS filter, there is a jump in area between filter order 26 and filter order 27. Before and after this the growth is linear. The growth rate is larger after the 26th filter order than before. Making the liner function fitting the filter orders 1 to 26:

$$A = 1978n + 4425 \tag{4.31}$$

and for the filter orders 27 to 31:

$$A = 2230n + 891 \tag{4.32}$$

When the constant component of both equations is compared to the regular RNS FIR filter covering a 12-bit dynamic range, one can see that the first equation has a higher, 4425 NAND equivalences compared to 2934, and the second has a lower, 891 compared 2934. The constant components are the overhead due to conversion. Due to the additional QRNS to RNS conversion and larger moduli, the QRNS filter is predicted to have a larger conversion overhead than regular RNS FIR filters, making the 4.31 the linear approximation that describes the relationship of the area and filter orders for this filter. The reference filter can be described by the function.

$$A = 716n + 288 \tag{4.33}$$

The area usage of the reference filter is clearly lower than that of the QRNS filter. When compared to the regular FIR filter covering a 12-bit dynamic rage, the reference filter in this case has a slightly higher linear component area usage, 716 NAND equivalents compared to 622, while the QRNS filter has a significantly higher, 1978 compared to 1167. When it is taken into consideration how the linear components of the area growth rate changes as the dynamic ranges changes, one can expect that the growth rate of the area for the reference filter will increase at a higher rate than for the QRNS filter, but, due to the fact that the 12-bit quadratic filter having a worse area growth rate in relation to its reference than the regular RNS filter, the dynamic range will have to be even higher for the QRNS filter to have a lower area than the reference.

Figure 4.20 shows the power consumption of the QRNS filter. The power consumption graph has a different slope for this filter than for the regular RNS filters. This suggests that there are some errors in the simulation, but from what one should expect due to the much higher area usage of the QRNS filter and the power consumption data, the conclusion that the power consumption of the QRNS filter is higher than for the reference filter is believed to be true.

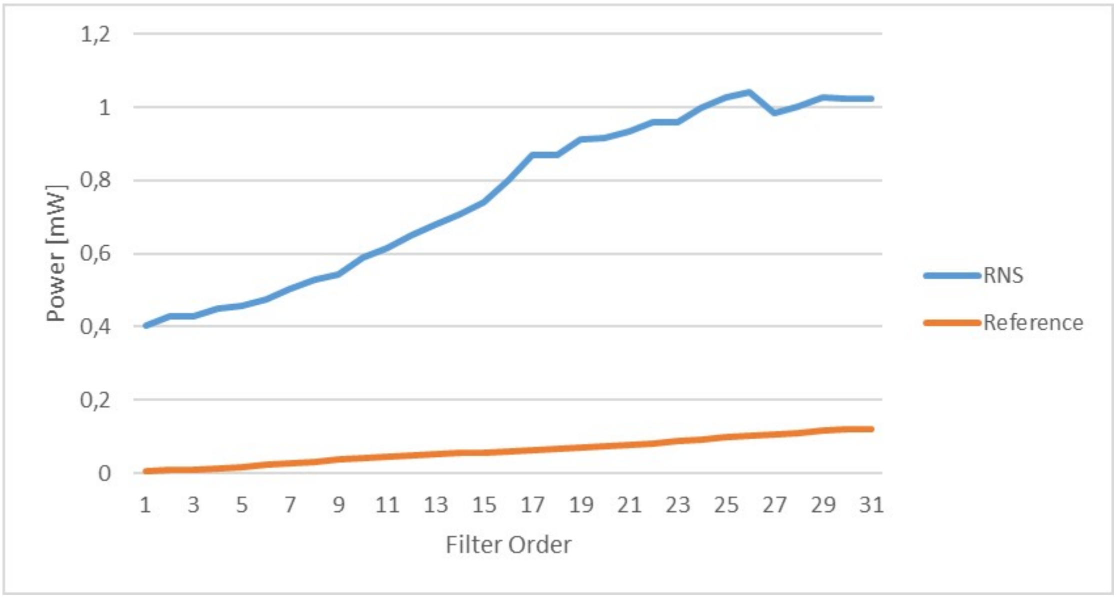

Figure 4.20: Power consumption of the quadratic FIR filter

4.4 Simulation of Scaler

The scaler is implemented using the same moduli as the 12-bit FIR filter. The setup is the same except that the scaler is added between the last filter tap and the converter. To isolate the power consumption of the scaler from the rest of the circuit, registers are put before and after the it. Figure 4.21 shows the area of the scaler compared to the same circuit without a scaler. It can be seen that the overhead due to scaling is minimal.

Figure shows the same graph for the power consumption. As one can see there is next to no difference in the power consumption of the circuit with scaler and that without. These results are in line with the result from the area measurements, and that there is next to no overhead in terms of power as well.

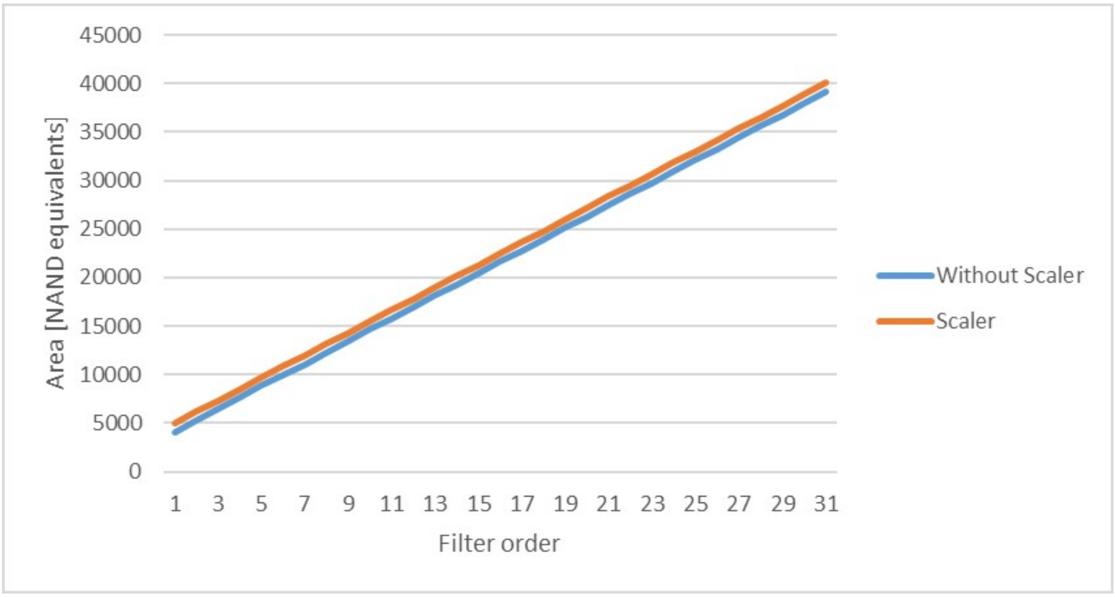

Figure 4.21: Area usage of the FIR filter with and without a scaler

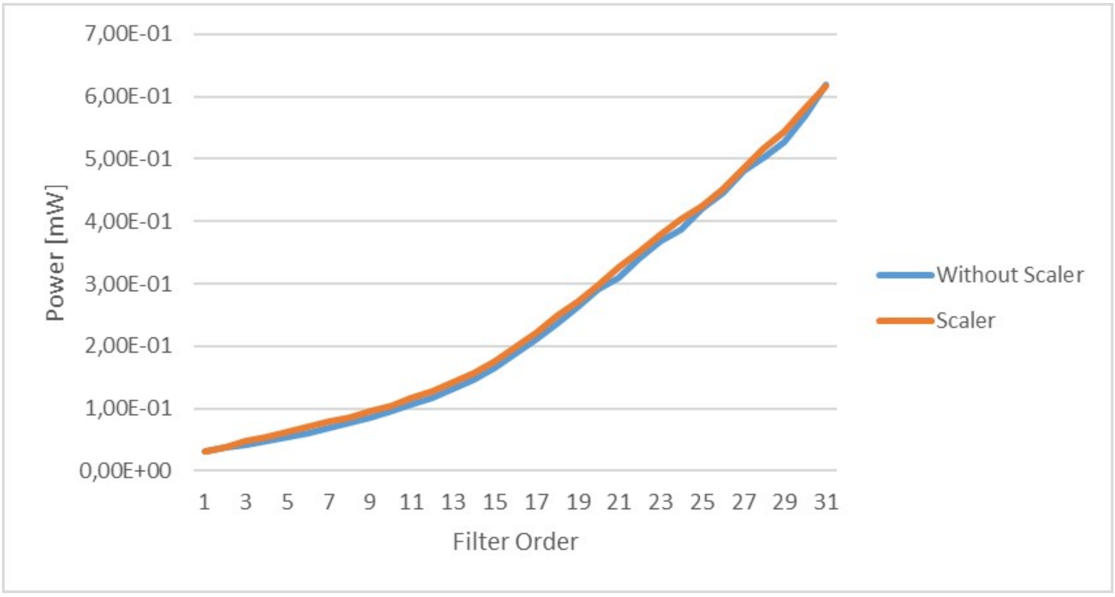

Figure 4.22: Power consumption of the FIR filter with and without a scaler

Chapter 5

Discussion

5.1 Validity of Results

Considering that the measurement data is very dependent on particularities of both measurement setup and the implementation of the designs used in the tests, it is hard to say whether the result provided in this thesis can be used in a general consideration of the validity of the residue number system. However it is believed that the data from this thesis can be used alongside other data to get a full overview on how and in which cases the residue number system can be used. This thesis has dealt with a specific type of implementation both when it comes to the FIR filters, where the hybrid RNS/binary method is used, and when it comes to the input and output converters. It is to be taken into consideration that the methods of implementation used in this thesis is not necessarily the best methods. Despite this, the conclusion taken based on the results from the simulations of these implementations can be used to make prediction on other designs if these differences are taken into account. Another aspect of the validity of the results is errors during simulation. Some of the data points presented in the result chapter departs from what is expected. This is in particular the case when it comes to the power estimations. The power estimations are dependent on the stimuli used and simulation setup and small errors during the simulations can have big impacts on the resulting power data. A third aspect is the specific simulation setup. Different setup can cause different result on the estimated power and it not given that two instances of the same implementation used in different simulation setups will lead to the same findings.

5.1.1 Transferability to Other Implementation Methods

As mentioned is the FIR and quadratic FIR implementations based on the hybrid RNS/binary approach. This approach has some particular aspects that have an effect on the resulting power and area usage of the circuit. Most notable is the use of the correction logic in the summation path. This logic makes the power usage due to glitching more prominent. In the result chapter it can be seen that glitching causes a big part of the power consumption for the RNS filter. For the regular FIR filter, the power consumption due to glitching is very low, which makes sense since it does not have the same correction logic. If the same simulation is done on RNS filters not using the hybrid approach, the power consumption due to glitching is expected to be lower.

The benefit of the hybrid approach is that simpler arithmetic units can be used. This leads to a lower area usage and lower power consumption internal to each tap. A RNS filter using a modulo arithmetic approach is expected to have a higher power consumption internal to each tap. This means, considering the shape of the power graphs in chapter 4.2, the linear component of the power consumption is higher and the nonlinear component is lower for nonhybrid RNS filters.

Also the different possible implementations of the input or output converters may lead to different power consumption and area usage. Only one version of each of the input and output converters are presented in this paper. There is a lot of literature describing different ways of constructing these converters, as seen in the references listed in chapter 3 and 7 in [Omondi and Premkumar (2007)]. From the results in chapter 4.2 it is easy to extract the area occupied by the converters and the power consumption caused by them. As such, if the power or area usage is to be estimated using other converters, one can subtract the constant power and area component and add a power consumption and area usage from the new converters. Due to this it is argued that the results of this thesis is very general in terms of converters used.

One aspect that is crucial to the implementation of filter is whether the filter are implemented in a serial or parallel manner. In this thesis the filters are implemented in a parallel manner. Despite the filters themselves being serial, pipelining stages is used between the filters and the converters to isolate the power consumption of the converters from that of the filter. The difference in a serial and a parallel has a lot to say in regards to whether the filter is best implemented using RNS or binary.

In a parallel filter there will be a longer path in which the glitching will propagate. In contrast, glitching in a serial filter will only propagate inside each tap, as each tap will be separated by registers. As stated earlier in this chapter, is the power consumption due to glitching a major part of the total power consumption of the RNS filter. In this chapter it is made an effort to separate what we have called the "internal" power consumption with the power consumption due through glitch propagation. Using this separation it is possible to use the results of the parallel implementation to make prediction of the validity of the use of RNS in serial filters.

5.1.2 Estimation Errors

As always are the results from the simulation prone to errors. As the results comes from simulations of net list using industry standard simulation tools, the errors are more likely to stem from errors in simulation setup or logical implementation than from errors in the simulation tool. Due to the synthesis being deterministic, simulations using the same RTL code and the same simulation setup will have the same results. Therefore there is no need to take into account random variation in the simulations.

The results are, for the simulations of the FIR filters and quadratic FIR filter, done using the same base structure, where the filter orders are taken in as a parameter. This provides one result of each of a series of similar implementations. From these series of implementations it is possible to identify values which do not fit with the trend derived from the other values. A form of errors that would not be detected as easily by comparing simulation results to each other is systematic errors. Systematic errors stems from faulty simulation setup and implementations. Because of a the chance of having systematic errors all measurement results have to be looked at in conjunction to what the expected values is for the same design. However, since only the relative performance between the proposed designs and the reference designs is relevant, the systematic errors is not necessarily determinant for the conclusion derived from the results.

	RNS		Reference	
Dynamic range	K	a	K	a
12-bit	2934	1167	-1	622
16-bit	4580	1570	-210	1010
24-bit	9253	2655	769	2329
32-bit	19814	3824	78	3904

Table 5.1: Coefficient for the model of the area usage for the RNS filter and for the reference filter

5.2 Findings

In chapter 4 the results from the simulations are presented and briefly discussed. In this section the main takeaways from the result will be discussed and it will be discussed what the main predictions one can make from these results are.

5.2.1 FIR filters

The area, for each of the dynamic ranges, has been described by a linear equation. Table 5.1 shows the coefficients used in the linear equation, which is on the form as shown in equation 5.1. How this equation is derived is explained in chapter 4.2.1.

$$A = a \cdot n + K \tag{5.1}$$

Figure 5.1 shows the linear component of the area usage for both the reference and RNS filters. It increases monotonically for both types of filters as the dynamic ranges increases, but the reference filter starts at a lower point and increases at a faster rate. This is as expected, because the main source of area usage for both filters are the multipliers and the size of multipliers are proportional with the square of the word width. The gain from having many smaller parallel multipliers, which is the case for RNS number, is larger for the larger dynamic ranges, as the parallelism of the RNS number is exploited more for them than for the lower dynamic ranges. For dynamical ranges corresponding to 32-bits, the linear part of the area usage is higher for the reference filter than for the RNS filter. It is assumed that this trend will continue to be present as the dynamic ranges increases beyond 32.

Figure 5.2 shows the constant component for the area usage for both the reference and RNS filters. It shows that the equation for the area usage reference filter has

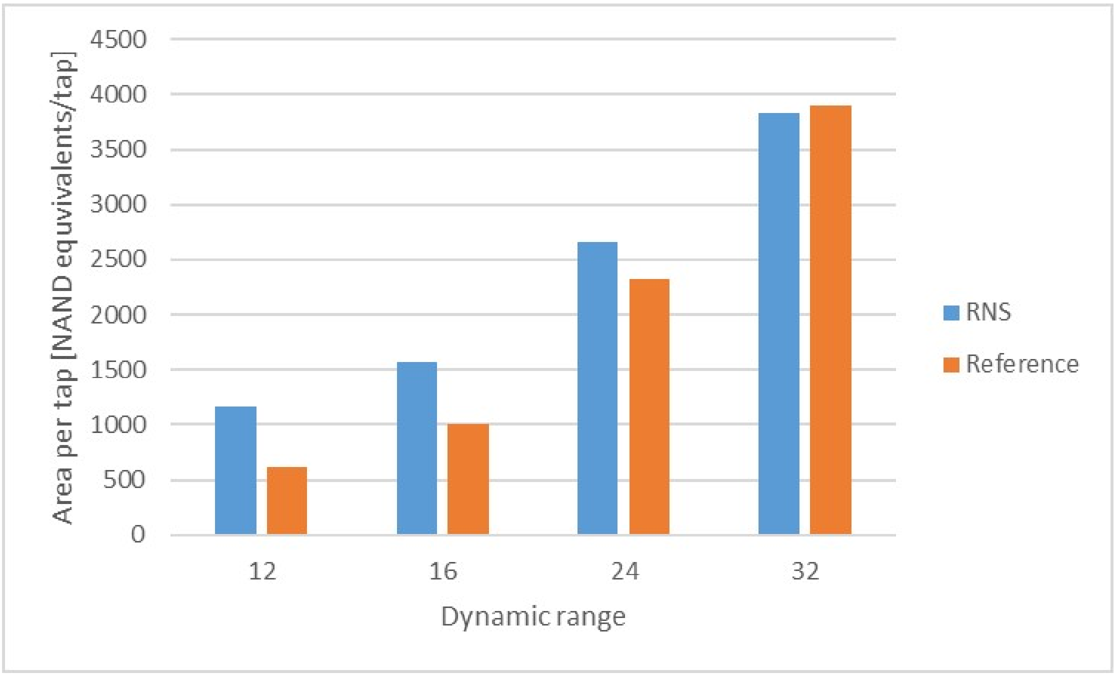

Figure 5.1: Linear component of the area usage model

almost no constant component. This is as expected since there is no conversion or other kinds of overhead for this type of filters. However, the equation for the RNS filter has a constant component which increases as the dynamic range increases. This is the overhead due to conversion. As expected the overhead is larger the larger the dynamic area is. This is because the conversion step is more complex as the RNS numbers covering a larger dynamic area uses more and/or larger moduli.

If one looks at the area usage of the converters compared to the area usage of the area usage of the taps, the area of the converter at 12 bits is about 2.5 times the area of a tap, while the area at 32 bits is about 5.2 times the area of a tap. This is opposite from the effect of the measurements for regarding area per tap. However, since the area per tap is only larger for the reference filter when the dynamic ranges are large, the only situations where the total area of the reference filter is larger than the RNS filter is when the dynamic ranges is large enough. The used converters are not necessarily the most optimal ones in terms of area and the overhead by conversion can therefore be lessened by using a more optimal solution.

The power consumption of the reference filters has been described using a linear equation, while the power consumption of the RNS filter has been described using

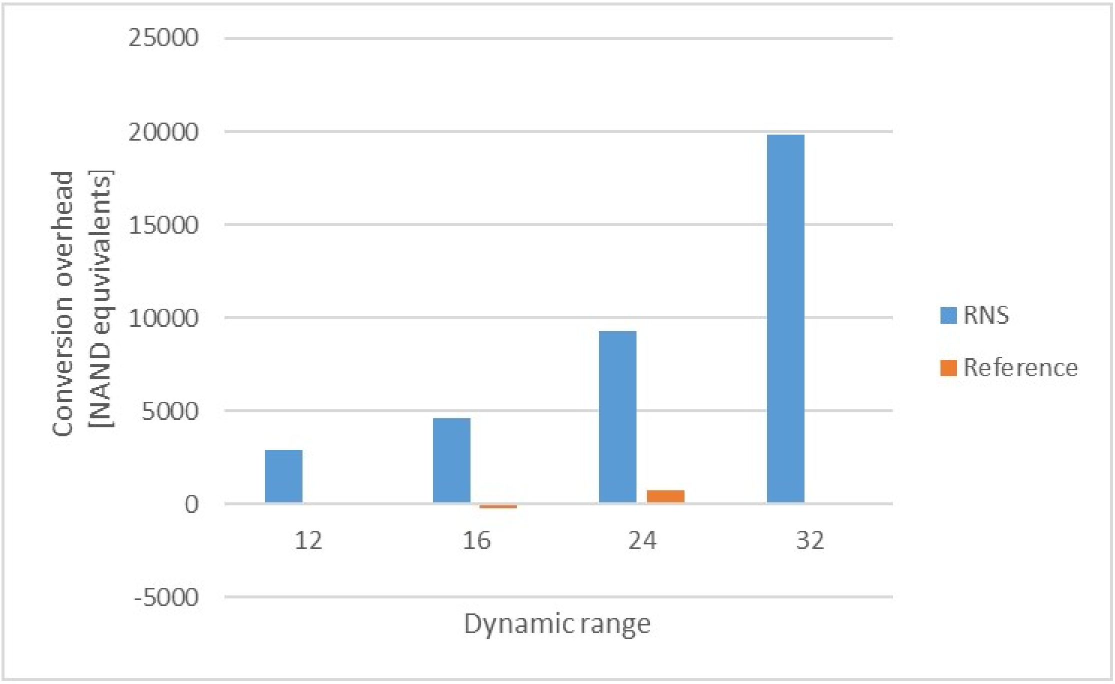

Figure 5.2: Constant component of the area usage model

a more complex function. The coefficients of both functions is shown in table 5.2.1. The equation for the power consumption of the reference filter is on the form shown in equation 5.1 and the equation for the power consumption of the RNS filter is on the form shown in equation 5.2. How this equation is derived is explained in chapter 4.2.1.

$$P = K + a \cdot n + c \cdot \sum_{i=0}^{n-2} \begin{cases} \sum_{j=0}^{j=i} p^j, & \text{if } i < L \\ \sum_{j=0}^{j=L} p^j, & \text{otherwise} \end{cases} \tag{5.2}$$

Figure 5.3 shows the linear component of the equation describing the power consumption of both the RNS and reference filters. This is what is considered the internal power consumption for each tap, which means that the power consumption due to the glitching on the output of each tap is not described by this component. The diagram shows that the linear component of the power consumption of the RNS filter increases at a steady rate as the dynamic range increases. The linear component of the power consumption of the reference filter, however, does not increase at a steady rate. The linear component increases very much between the dynamic

	RNS				Reference	
Dynamic range	K	a	c	p	K	a
12-bit	0,026	0,005	0,0022	1.21	0,004	0,0022
16-bit	0,04	0,008	0,0004	1.16	-0,00456	0,00566
24-bit	0,11	0,014	0,00067	1.18	-0,006	0,015
32-bit	0,31	0,022	0,0014	1.14	-0,0062	0,016

Table 5.2: Coefficient for the model of the power consumption for the RNS filter and for the reference filter

ranges 16 and 24 but not so much between 24 and 32. This indicates that some of the estimated values are incorrect. If one looks at figure 4.10 showing the measured power data of the reference filter, it shows that the linear approximation to the the measured power consumption is not unambiguous. This can indicate that the linear growth rate of the power consumption of the reference filter is to high for the dynamic range of 24-bits. However, if one looks at how the linear power components of the reference filters change in relation to the same components of the RNS filters, one can see that the component is a bit below half of the reference filter for the dynamic range of 12 bits, for 16 bits it is a bit below three fourths, for 24 bits its linear power component is higher for the reference than for the RNS filter and for 32 bits the it is a bit below three fourths of that of the RNS filter. Given the increase in the linear power component relative to the RNS filter from 12 bits to 16 bits, the estimated linear component for 24 bits fits the projection one can derive from these data points. Especially considering the distance from 16 to 24 is twice that from 12 to 16 and the expected development of power usage due to the RNS increased ability to exploit parallelism for higher dynamic ranges. Considering this the estimation for 32 bits seems to be too low. However, it is hard to make any final conclusion, as there is only 4 data points. The same estimations has to be done for other dynamic ranges to conclude how the linear component of the power consumption changes as the dynamical ranges changes.

Figure 5.4 shows the constant component of the power consumption for each of the dynamic ranges. The constant component of the power consumption of the reference filter is close to zero for all dynamic ranges. As for the area usage, this is as expected, because the reference filter have no conversion step. The constant part of the power consumption increases as the dynamic range increases. This can also be seen in conjunction to the area usage, where the area also increases as the dynamic

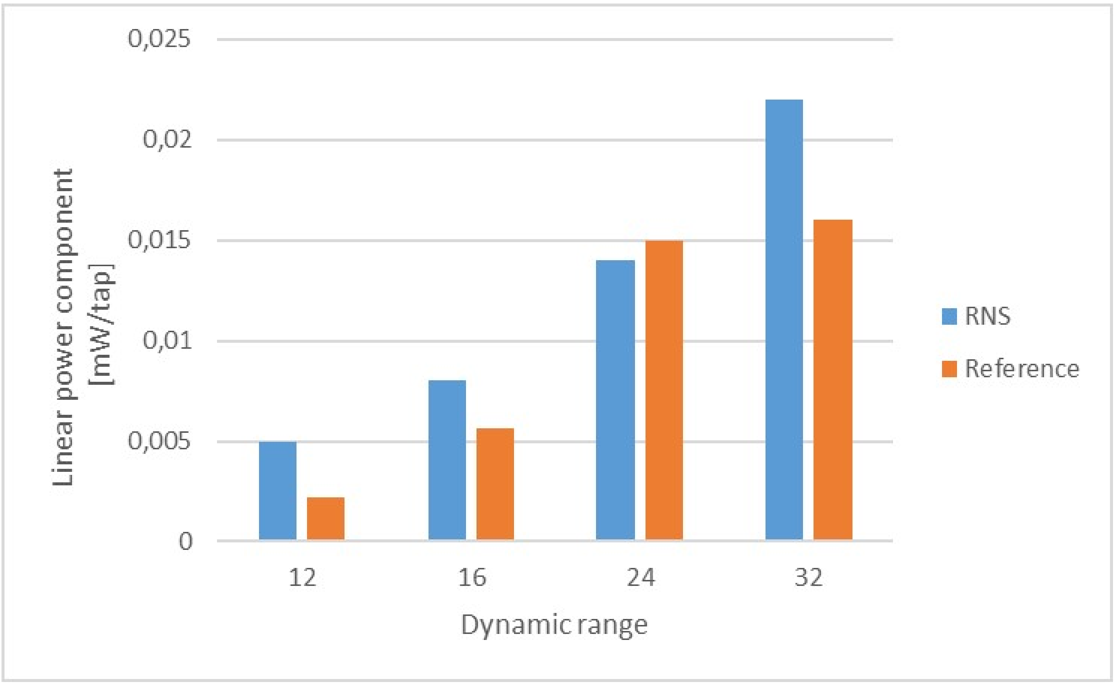

Figure 5.3: Linear component of the power consumption model

range increases. For the power consumption the more complex conversion steps for higher dynamic ranges is believed to be the reason for the increase as well. One can recognize that the relation between the overhead due to conversion and the linear component is worse for the power than for the area, with the conversion overhead being 5 times that of the linear power component at 12-bit and increasing to 15 times at 32-bit. This means that one must have a big number of filter taps for the gain in using RNS arithmetic to overtake the overhead by conversion.

5.2.2 Quadratic FIR Filters

The quadratic filters are only simulated for one dynamic range. Therefore there is not a as complete result for this filter as for the regular FIR filters. However, due to similarities in structure between the quadratic and the regular FIR filters, the result from the simulation of the regular FIR filters can supplement the results from the quadratic FIR filters.

The resulting area usage shows the same trend as for the regular FIR filters, having a linear growth for both the reference and the QRNS filter. The growth for the

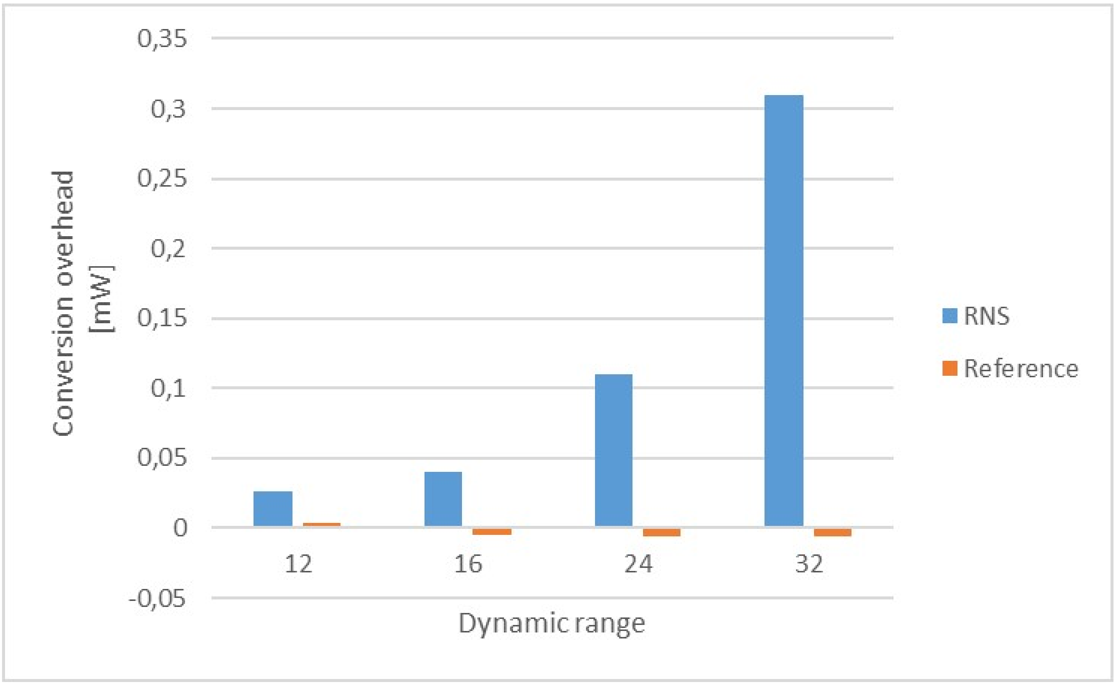

Figure 5.4: Constant component of the power consumption model

QRNS filter is steeper than that of the regular RNS filters covering the same dynamic ranges. This is due to the limited flexibility in the choice of moduli when using the QRNS number. The regular RNS FIR filter uses the moduli set $\{3,4,5,7,11\}$ while the QRNS filter uses the set $\{13,17,37\}$. The difference between these moduli is that the regular FIR filter has a higher number moduli with lower values, while the quadratic one has fewer moduli with higher values. The limited flexibility is due to the property the modulo numbers has to satisfy, which is described in chapter 2.3. The area usage of the quadratic filter is expected to follow the same trend as the nonquadratic one as the dynamic ranges increases. The higher area usage at 12-bit is assumed to be present at other dynamic ranges as well and thus lead to the dynamic ranges being even higher before the area per tap for the RNS filter is better than that of the reference filter. One thing that speaks in favor of the quadratic RNS filters, is that the complexity of the multipliers is what increases fastest as the dynamic ranges increases. Due to the QRNS filter can be designed using only 2 multipliers, compared to 4 of a non RNS complex filter, it can be argued that the effect of this will lead to area gains for higher dynamic ranges. Further tests has to be done, covering higher dynamic ranges, to achieve a final conclusion.

The power consumption of the QRNS filter compared to the reference is also worse in this case than for the regular FIR filters. It has to be said that the form of the graph describing the power consumption suggest that not all of the data points are correct. Especially the 15 highest filter orders seam to have a flatter slope than the regular FIR filter. If only the 16 lowest filter orders are taken into account, the result is in line with what one can expect from the results of the area usage, which is that the power consumption relative to the reference is clearly worse for QRNS filter than for regular FIR filters.

In general this specific implementation of a quadratic FIR filter is disadvantageous in terms of area usage and power consumption compared to the reference filter. This is mainly because of the moduli chosen and differences in the choice of moduli may have effects which are not reflected in the results in this thesis.

5.2.3 Scaler

The imagined scenario, in which a scaler is used, is when one has a system having a particular dynamic range. If an overflow is expected in this system, one can either use a higher dynamic range or insert a scaler in the calculation path. From chapter 5.2.1 and from the results in chapter 4.4 one can see that the increased power consumption and area usage of using a higher dynamic range is far greater than inserting a scaler.

If the overhead of the scaler is as favorable for other dynamic ranges as for the 12-bit dynamic range, the same type of scaler can be used in the same situations for those cases as well. It is not believed that the overhead of the scaler relative to for example the power consumption or area usage of a filter tap will vary significantly depending on the dynamic range. This is because the same structure is used and this structure does not have a large dependency on the number of moduli and sizes of each moduli.

5.3 Cases where RNS is beneficial

In general the results from the tests do find no benefit by using the residue number system instead of a conventional binary number system. However, one can use the results to point at which aspect of the RNS implementations that must be improved to make its usage viable and in which cases it is most likely to be viable.

The results show that the filter taps can be constructed with a lower area usage and a lower power consumption, if the filter requires a high enough dynamic range. This is especially the case if the filter is constructed in a serial manner, as the power consumption due to glitching will be reduced. However, the requirements for the dynamic ranges and filter order must be very high for this to make the RNS the preferred number system. In many situations where DPS systems are constructed, the requirements for dynamic ranges and filter orders are one of the first things one tries to reduce, as this will have a very large effect on the power consumption. This make the situations, whit high filter orders and high dynamic range requirements, very unlikely to occur.

The implementations presented in this chapter are just some of the many ways to implement the RNS systems. It is believed that some of the aspects, which leads to the RNS systems not having the expected gain over the conventional implementation, can be improved by using more favorable ways to implement these systems. Ibrahim (1994) states that the method presented in their paper, which is the one used for the FIR filters in this thesis, provides the gains superior to other methods to implement RNS arithmetic if the moduli sizes are big. The designs presented in this paper tries to use as small as possible moduli sizes. This may be one of the reasons why the result are not as positive, in terms of the viability of RNS, as expected.

The latency of the implementations is not estimated. The latency is in itself a interesting value, as it can describe how fast a system can be run and at what rate it can be clocked. However, a lower latency can also be exploited to reduce power consumption. Cardarilli et al. (2000) suggest, due to the reduced latency of the RNS system, to use a lower supply voltage for the RNS, which increases the latency but lowers the power consumption. This can be used in cases where the clock rate is given, but the latency is lower than the clock rate.

Chapter 6

Conclusion

In this thesis the viability of the residue number system in radio receivers was researched. The main motivation was to see if the advantages of doing arithmetic operations using this number system can be used to improve the performance of DSP operations in area usage and power consumption.

Several circuits were constructed using this number system, including FIR filters, both using linear and complex filter coefficients, input and output converters and a scaler. Both the linear and complex FIR filters were constructed using a hybrid RNS/binary approach. The complex FIR filters were designed using the quadratic residue number system, the input converter was designed using look up table based approach, returning the modular sum of the residue for the particular modulo for each of the digit in the binary input word, the output converter was constructed using a Chinese reminder theorem based approach and the scaler was constructed using a multiplicative inverse together with base extension.

These implementations were run through a simulation to find their area usage and power consumption. The simulations are done on netlists generated using Design Compiler, with switching information generated by Questasim using Matlab generated stimuli. All designs are simulated for 55nm technology at 125°C. For each of the simulations, the same simulation is done on a reference design, which is implemented using a conventional binary number system, so that the performance of RNS relative to this number system is found.

Mathematical models to describe the power consumption and area usage of the RNS and binary FIR filter were derived. These models were found to fit well with the results from the simulations. Using this models one can separate the different components of both the area usage and power consumption of the filters.

The results show that the implemented RNS FIR filters have a worse area usage and power consumption than the reference FIR filters for most application. However, it becomes better as the required dynamic ranges of the signals increases. It is argued that RNS filters with a dynamic rage requirement of more than 32-bits have a lower area per tap than the reference filter. And the RNS filters a dynamic rage requirement of more than 24-bits has a lower power consumption per tap, if the power consumption due to glitching is not included. The quadratic/complex RNS filters also perform worse than the references in terms of both area usage and power consumption. The overhead due to input and output is significant for all dynamic ranges and becomes worse as they increase, which counteracts the increased gains of RNS FIR filters. The overhead by scaling is shown to be minimal, in therms of both area usage and power consumption, compared to the other components implemented in this thesis.

The main conclusion of this thesis is that RNS DSP circuits, using the same methods of implementations used here, will perform worse than the equivalent binary circuits. However, the latency of the designs is not found. Therefore the conclusion is not final, as the latency decides whether the RNS designs can be run at a faster clock rate than the binary designs and whether it is possible to reduce the supply voltage resulting in reduced power consumption.

6.1 Future Work

This thesis touches only a small part of the field of research revolving the residue number system. Here the focus is put at how it is suited for certain DPS operations. For future work, the same simulations which are done for these operation can be done for other operations including IIR filters, sine and cosine synthesis and comparison.

In this thesis only some of many different methods of implementations are used when designing the RNS circuits. The same circuits can be designed using other methods, which can lead to different result than the ones designed in this thesis. One particular method of implementation, which is expected to lead to a lower power consumption and area usage, is using look up tables for the modulo arithmetic instead of the hybrid RNS/binary method. Other cases which would be interesting to investigate is: different ways to implement the input and output converters and to study how they can be improved, using different simulation setups, implementing the filters as a pipeline or in a serial manner and using different parameters for the same implementations used in this thesis, for example moduli sets and dynamic ranges.

One last ting that would be interesting to study is whether the latency of the circuit can be reduced using RNS and if this can be used to lower the power consumption by using a lower supply voltage.

Bibliography

Burgess, N. (2001). Efficient rns-to-binary conversion using high-radix srt division. *IEE Proceedings - Computers and Digital Techniques*, 148(1):49–52.

Cardarilli, G., Nannarelli, A., and Re, M. (2000). Reducing power dissipation in fir filters using the residue number system. In *Circuits and Systems, 2000. Proceedings of the 43rd IEEE Midwest Symposium on*, volume 1, pages 320–323.

Freking, W. and Parhi, K. (1997). Low-power fir digital filters using residue arithmetic. In *Signals, Systems amp; Computers, 1997. Conference Record of the Thirty-First Asilomar Conference on*, volume 1, pages 739–743.

Garner, H. L. (1959). The residue number system. *Electronic Computers, IRE Transactions on*, EC-8(2):140–147.

Ibrahim, M. K. (1994). Novel digital filter implementations using hybrid rns-binary arithmetic. In *Signal Processing*, volume 40, pages 287–294.

Jenkins, W. and Leon, B. (1977). The use of residue number systems in the design of finite impulse response digital filters. *Circuits and Systems, IEEE Transactions on*, 24(4):191–201.

Jullien, G. (1978). Residue number scaling and other operations using rom arrays. *Computers, IEEE Transactions on*, C-27(4):325–336.

Mohan, P. A. (2012). *Residue number systems: algorithms and architectures*, volume 677. Springer Science & Business Media.

Nannarelli, A., Re, M., and Cardarilli, G. (2001). Tradeoffs between residue number system and traditional fir filters. In *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*, volume 2, pages 305–308.

Omondi, A. and Premkumar, b. (2007). *Residue number systems : theory and implementation.* Imperial College Press, London.

Ramírez, J., García, A., Meyer-Baese, U., and Lloris, A. (2002). Fast rns fpl-based communications receiver design and implementation. In *Field-programmable logic and applications: Reconfigurable computing is going mainstream*, pages 472–481. Springer.

Shenoy, A. P. and Kumaresan, R. (1989). Fast base extension using a redundant modulus in rns. *IEEE Transactions on Computers*, 38(2):292–297.

Szabo, N. S. and Tanaka, R. I. (1967). *Residue arithmetic and its applications to computer technology.* McGraw-Hill.

Taylor, F. (1984). Residue arithmetic a tutorial with examples. *Computer*, 17(5):50–62.

Vergos, H. and Efstathiou, C. (2008). A unifying approach for weighted and diminished-1 modulo $2^n + 1$ addition. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 55(10):1041–1045.