CHALMERS

Implementation of the DSP chain of MPD radar using FPGAs and Residue Number System (RNS) Arithmetic

Master of Science Thesis in Integrated Electronic System Design

ASHKAN GHEYSVANDI

MOHAMMAD ATTARI

Implementation of the DSP chain of MPD radar using FPGAs and Residue Number System (RNS) Arithmetic

ASHKAN GHEYSVANDI
MOHAMMAD ATTARI

Examiner: LARS BENGTSSON

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2011

Abstract

With the ever increasing demands for computational power in the digital processing systems of the present and the future, the emphasis on the power consumption and the area requirements of designs is tangible. To meet these stringent requirements this paper investigates the effects of using an alternative numbering system known as the Residue Number System or RNS.

The Medium pulse repetition frequency Pulsed Doppler (MPD) radar principles are studied and the various components in the MPD DSP chain are introduced. These include filters, FFT processors, and target detectors.

The MPD signal processing chain is modeled in both binary and RNS number systems and their respective performance is juxtaposed. To verify the operation of the developed models, a MATLAB model of the DSP system is also provided.

To describe the DSP system in hardware VHDL is used as the hardware description language of choice. Verification of the correct operation of the design using simulation is carried out using Mentor Graphics ModelSim software. FPGA platform is selected for implementation of the DSP system in hardware and the Xilinx ISE software is used for synthesis and placement and routing. In addition to verification through simulation the system is verified in-circuit.

The system described in VHDL is modular and divided into a number of functional, memory (RAM and ROM), and control modules. These modules are written to be synthesis friendly and the targeted systems' guidelines were followed.

Keywords:

Digital signal processing, Pulsed Doppler effect, MPD radar, Unconventional Number systems, VHDL, Simulation, Verification, FPGA synthesis, Low power design, MATLAB

Table of Contents

Acknowledgements

The authors would like to thank their supervisor Dr. Lars Bengtsson from Chalmers University of technology and Dr. Anders Åhlander from SAAB Microwave Systems.

1. Introduction

As a new generation of airborne radar has been developed, most of their capabilities are achieved in the presence of digital processors with high throughput capability. Nowadays finding an appropriate architecture for this key feature of radar processing is a big challenge. There are many factors to be considered in the design of the architectures such as pipelining, fault tolerance, high throughput, efficient modular design, area utilization, and power consumption.
Considering the low volume of the products, specific design and cost of the product make Field Programmable Gate Arrays (FPGA) a suitable choice for the processor. To find a solution for the architecture of the processor, two different systems have been designed for FPGA implementation in this thesis: Residue Number System (RNS) and binary numeral system. This made it possible to compare these two systems together. The authors' main goal was to investigate if the RNS is a competitive solution for the architecture. The two designs have been compared in terms of power dissipation, utilized area, and speed.

The following sections give us a walk through all the main principles of the Doppler effect and pulsed Doppler radar. This is done by introducing their main components and providing examples and illustrations to assist understanding of the concepts. A brief introduction to the RNS number system is also presented.
The theory introduced is put to practice by implementing the system in VHDL for hardware implementation. All the major components of the implemented system are explained. RNS design changes will as well be discussed. To conclude, simulation and synthesis offer comparisons between the binary and RNS systems.

2. The Doppler Effect

The Doppler effect could be understood by the classic example of the change in pitch of a locomotive's whistle or the roar of a racing car, as they pass. The waves from stationary sources move out in a circular fashion from the source. But when the source is in motion, we have the interesting phenomenon of changes in the frequency. The waves in front of a non-stationary source are bunched together and the waves behind it are spread out. In the racing car example we hear the pitch rising when the car is moving toward us but dropping when it gets passed us.

3. Pulsed Doppler Radar

Pulsed Doppler radar employs the Doppler effect with pulsed transmissions to do the radar ranging. A generic pulsed radar consists of some major components. These include the transmitter, receiver, time-

shared antenna, and the display. Transmitter typically changes DC pulses into a pulse of radio frequency energy. Received signals are translated by the receiver to low-pass-filtered signals. Antenna in its basic form is composed of a radiator and a reflector in parabolic shape [1].

In generic pulsed Doppler radar three basic innovations have been utilized: coherence, digital processing and digital control. All these features make radar more flexible in comparison to pulsed radar. Moreover they enable the detection of Doppler frequencies and make the radar more accurate. Figure 1 illustrates a system diagram of pulsed Doppler radar.

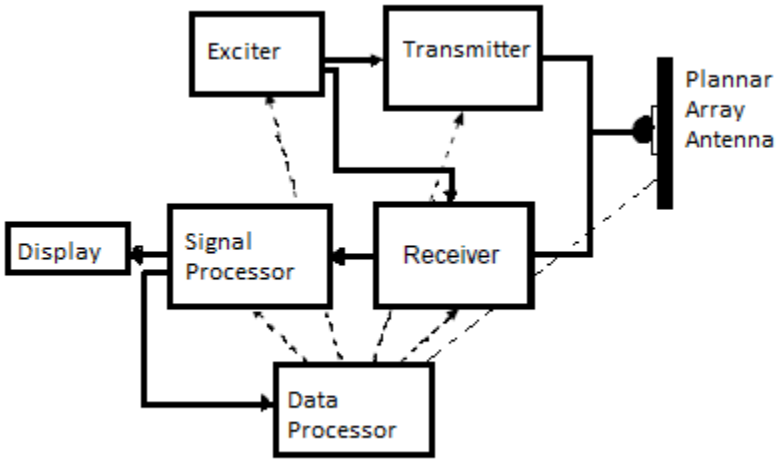

FIGURE 1 - PULSED DOPPLER RADAR SYSTEM COMPONENTS

Data Processor carries out routing and control for different units of the radar. Additionally it monitors the radar performance and performs fault detection. Continuous low-power signals of a specified frequency and phase are generated by the Exciter to be transmitted. The Transmitter amplifies Exciter's pulses for transmission. In this part the pulse could be modified, modulated, or coded for pulse compression. As can be seen in the figure above, the Transmitter uses the same antenna as the Receiver. The task of the Duplexer is to switch between these two. The frequency of the transmitted pulses is in the band of 1 to 40 GHZ.

Ideally, the transmitter sends pulses with high peaks and short widths in the direction called bearing. In practice, however, energy is transmitted in different directions. Consequently, most of the transmitted energy will be in the bearing direction, which is also known as the *main lobe*, and the rest of the it gets transmitted in directions called *side lobes*. Figure 2 shows the transmitted pulse and the main and side lobes [2].

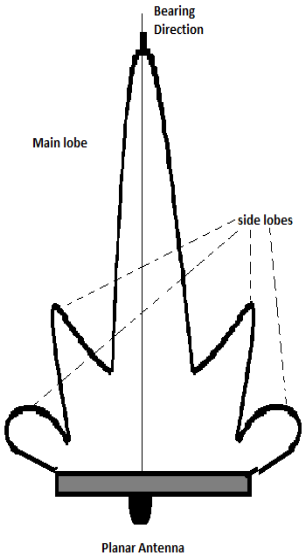

FIGURE 2 - TRANSMITTED PULSE

The antenna is a planar array antenna. It consists of many individual radiators on a flat surface. These radiators decrease the side-lobes.

Received signals consist of the target's echo and some unwanted echoes know as *clutters* and may also be disturbed by the so called *jammer* signals. The receiver increases the power of incoming echoes by a low-noise preamplifier. By comparing the output of the received signals with a reference signal the video frequency is obtained. Figure 3 shows the different parts of the receiver.

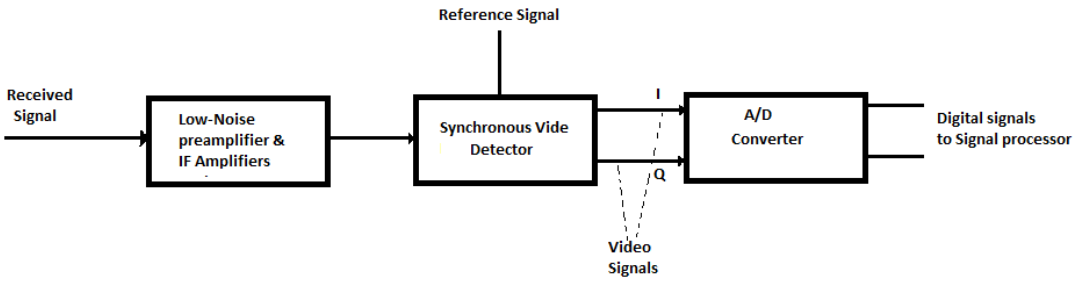

FIGURE 3 - RECEIVER'S MAIN COMPONENTS

The received signal is passed through a synchronous detector depicted in Figure 4. A bipolar video signal is generated as the output. The first output is the in-phase (I) signal and the second one is the quadrature (Q) signal.

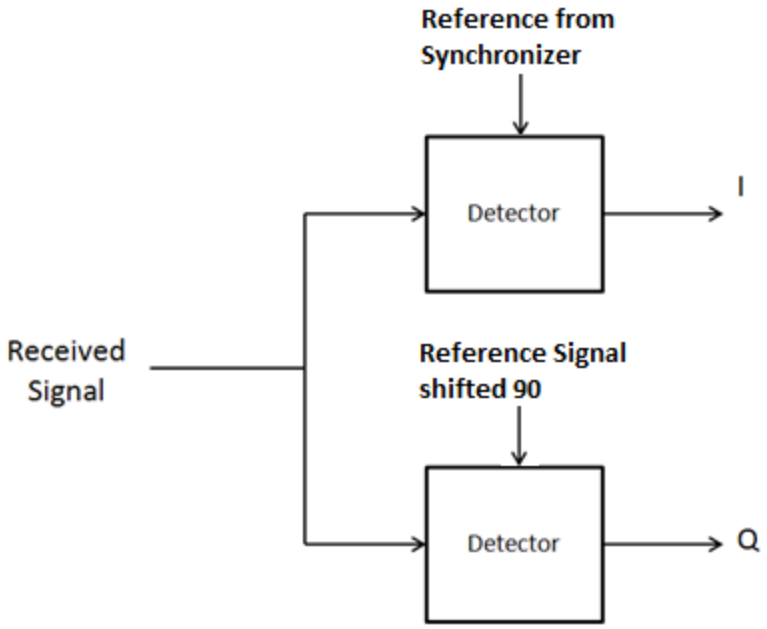

FIGURE 4 - SYNCHRONOUS DETECTOR

Sampling the video signals should be done at a precise rate. Figure below demonstrates the sampling procedure.

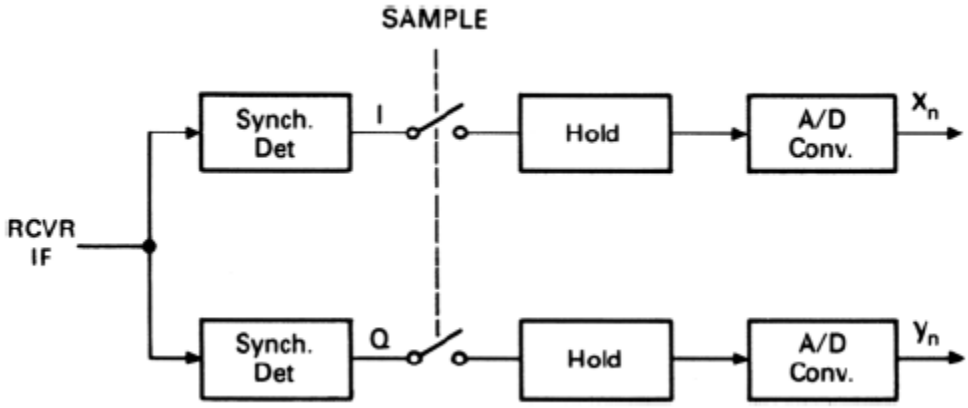

FIGURE 5 - SAMPLE AND HOLD

4. MPD Radar Signal Processing Chain

The following sections describe different modules of the MPD radar.
A block-diagram of the various components of the radar signal processing chain is provided in Figure 6.

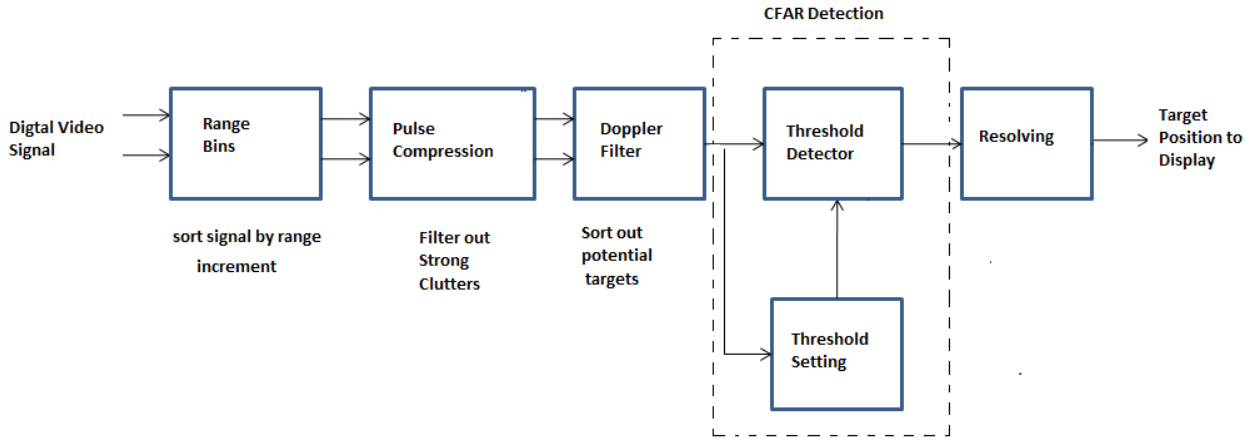

FIGURE 6 - DSP CHAIN OF MPD RADAR

4.1. Data Collection

The first phase of the chain is responsible for the collection of data from the A/D convertors. Data are stored into the so called *range bins.* Each range represents a distance of the object to the radar. This is calculated by considering the fact that the distance is proportional to the time elapsed from signal transmission to signal reception. As described earlier, the range bins are collected in the sample frequency or f_{sample}. The radar pulses are sent in a frequency called pulse repetition frequency (prf). The interval in which all the transmitted pulses are sent and the range data are collected is called Coherent Processing Interval (CPI). For each received pulse, a row of samples is gathered. Figure 7 demonstrates the data collection procedure of the CPI window [2].

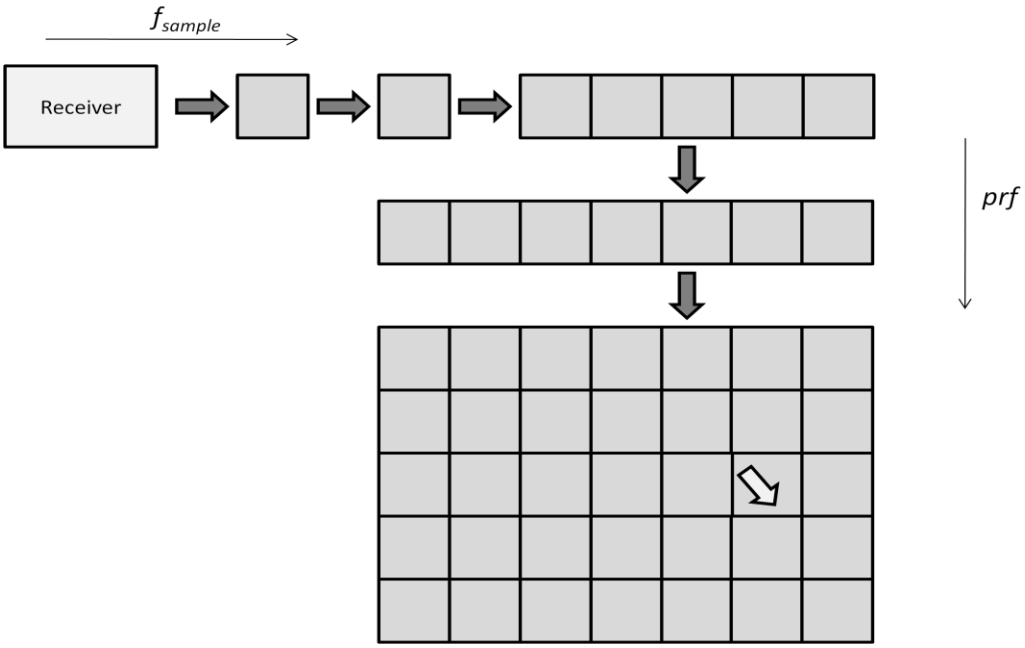

FIGURE 7 - DATA INPUT COLLECTION

There are three categories of pulsed Doppler radar categorized according to their prf. These are divided in low (0.25-4 kHz), medium (10-20 kHz) and high (100-300 kHz) prf classes. This thesis is concerned with the medium class of the pulsed Doppler radar.

4.2. Pulse Compression

For high resolution and long range detection, pulses should be narrow and high. In practice however the power of the pulses are limited. To solve this issue a technique called pulse compression is utilized. In this method pulses are internally modulated and have sufficient width to provide enough energy at the peak. In the receiver the received echoes are compressed by decoding. There are two commonly used methods for pulse coding. These are linear frequency modulation and binary phase coding. The latter is chosen for implementation in this thesis and is further explained below [1].

4.2.1 Binary Phase Modulation

In this method the phase of radio frequency is modulated on finite increments; using either a 0 or 180 phase. Each pulse is divided into a number of sub-pulses. The phases of certain sub-pulses are shifted by 180 in a pattern which is determined by the coding system. Each positive shift is equivalent to a subtraction (-) operation and unshifted sub-pulses act as addition (+) operations. The received echoes are time-delayed with the same time duration associated with the uncompressed pulses (τ). The delay line is composed of a filter with a specified number of taps. When the line is full, the output is amplified based on the number of available taps.
In practice the outputs of the taps do not have an exact 0 or 180 phase. Because of this limitation shift operation the outputs of the taps could not cancel each other. So, despite the main lobe some side lobes are generated. Using a set of Barker codes generates side-lobes with lower amplitude in comparison to mail lobe. A five-digit barker code set is demonstrated in Figure 8 [2].

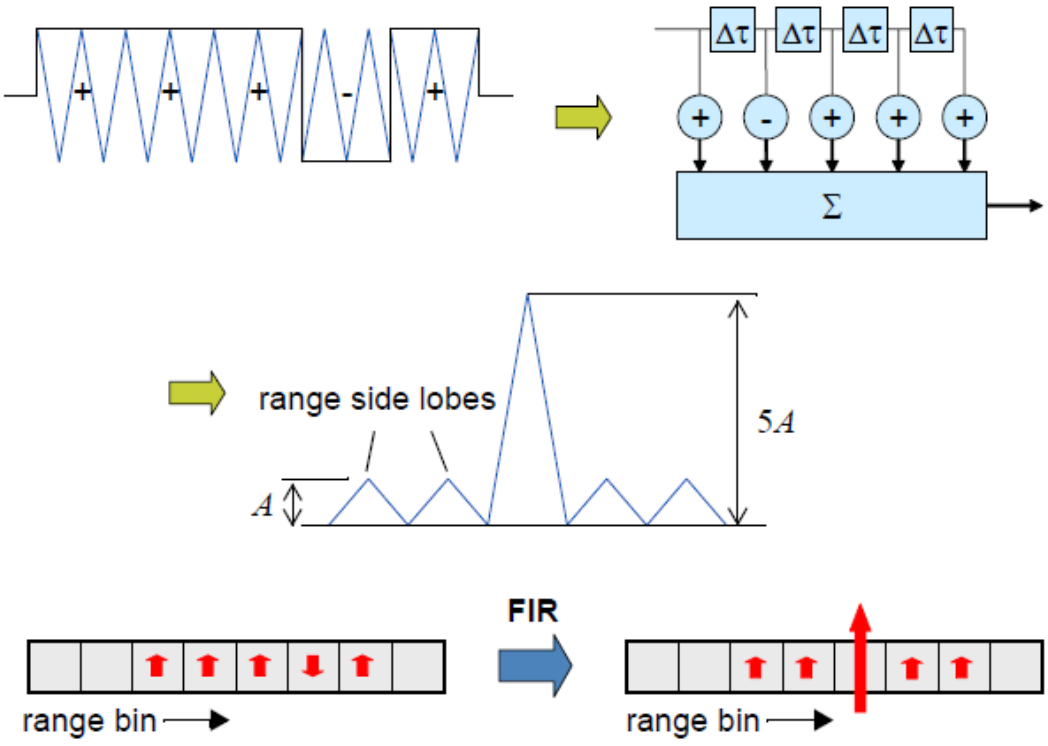

FIGURE 8 - BINARY PHASE MODULATION

4.3. Doppler Filter

The received signal passes through the Doppler filter for detecting echoes from a variety of sources and sorting them in the Doppler frequency. The task of this filter is to integrate the input within the Doppler frequency and eliminate the rest.

After gathering the data window in a CPI, it is passed through a filter and a separate bank of Doppler filter is obtained. Owing to the fact that filtering increases the signal to noise ratio (SNR), the filter gives an estimation of the target speed.

The filter performs as a Discrete Fourier Transform (DFT) on the samples contained in each column. A fast way of implementing the DFT is the Fast Fourier Transform (DFT). FFT is much faster and reduces the computational cost dramatically. Performing the FFT computation on the N samples in each column of the matrix yields the Doppler channel or velocity bin. Figure 9 depicts the FFT processing carried out on the range bins. Ground clutter has nearly zero Doppler frequency and they could be safely removed. In the final stage of filtering the absolute value of data is calculated and the phase information is discarded.

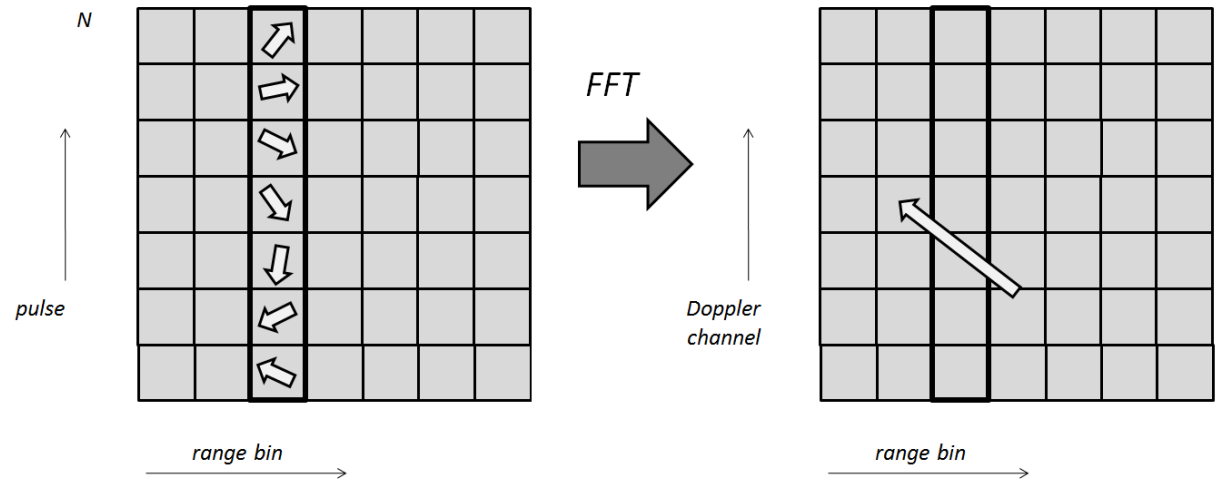

FIGURE 9 - DOPPLER FILTER PROCESSING

4.4. Target Detection

The next stage involves detecting the potential targets. The process for this task is Constant False Alarm Ratio (CFAR). The matrix sets threshold to remove false samples. The threshold value is calculated based on the average value of the cells surrounding the cell under test (CUT). If the CUT has a value higher than the amount of the average it is preserved as a potential target, otherwise it will be zeroed out. Figure 10 shows an example CUT in the CFAR process.

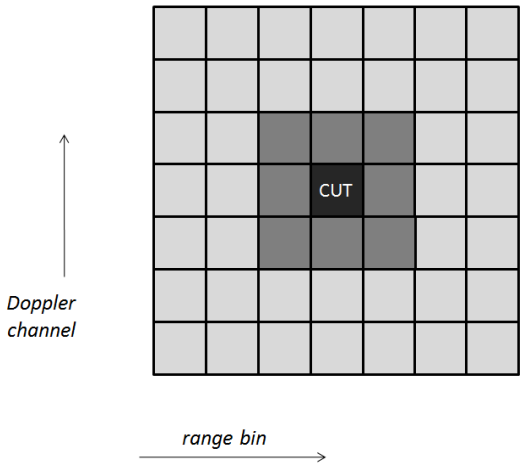

FIGURE 10 - CAFR PROCESSING

4.5. Resolving

When a target is far away, the time of flight could be longer than the time between pulses. This causes aliases in some echoes. For instance while processing pulse number two, the receiver may get echoes from pulse number one.
This problem could be solved by a process named resolving. By changing prf in the subsequent CPIs, non-aliased targets appear in the same range of CPI. Figure 11 shows a row of different CPIs [2].

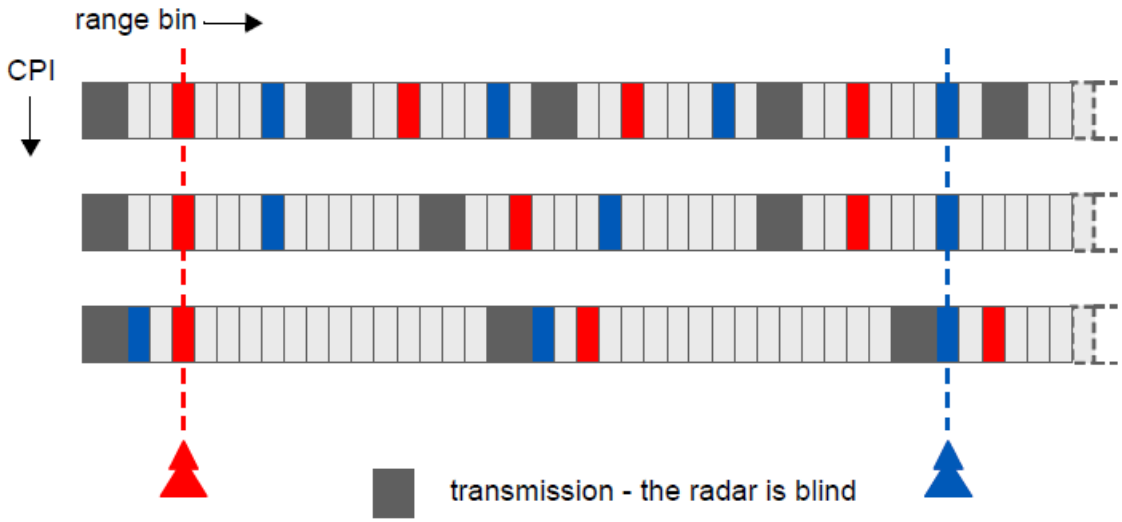

FIGURE 11 - RESOLVING PROCESS

5. Residue Number System (RNS)

Residue number system represents number with the reminders. Due to carry-free and parallel operations, the RNS numbering system is popular in arithmetic calculations and digital signal processing applications. The key operations of this numbering system are addition, subtraction and multiplication [3].
The RNS is represented in terms of a set of relatively prime moduli in which P denotes the moduli set:

$$P = \{ m_1, m_2, ..., m_N \}$$

Each RNS digit is obtained by calculating the *i*th residue of the integer number in the moduli set. This gives a unique representation of integer number X in RNS:

$$RNS(X) = \{ x_1, x_2, ..., x_N \} = \{ X \bmod m_1, X \bmod m_2, ..., X \bmod m_N \}$$

The dynamic range of the RNS numbers with N moduli is:

$$M = (m_1 * m_2 * ... * m_N)$$

Addition, subtraction, and multiplication are performed efficiently but operations such as division and comparisons are non-trivial.
If RNS representation of X, Y are given by $\{ x_1, x_2, ..., x_N \}$ and $\{ y_1, y_2, ..., y_N \}$ and respectively, then the result of the arithmetic operations producing output Z is calculated by the following, If Z belongs to the dynamic range:

$$Z = XoY$$

$$Z(RNS) = \{ (x_1 \text{ o } y_1) \bmod m_1, (x_2 \text{ o } y_2) \bmod m_2, ..., (x_N \text{ o } y_N) \bmod m_N \}$$

In which o is one of (+, -, *).

6. Binary System Implementation

6.1. Top-level System Organization

The DSP system is composed of a number of functional, control, and storage components. The following block diagram illustrates how the system is structured form a high-level point of view.

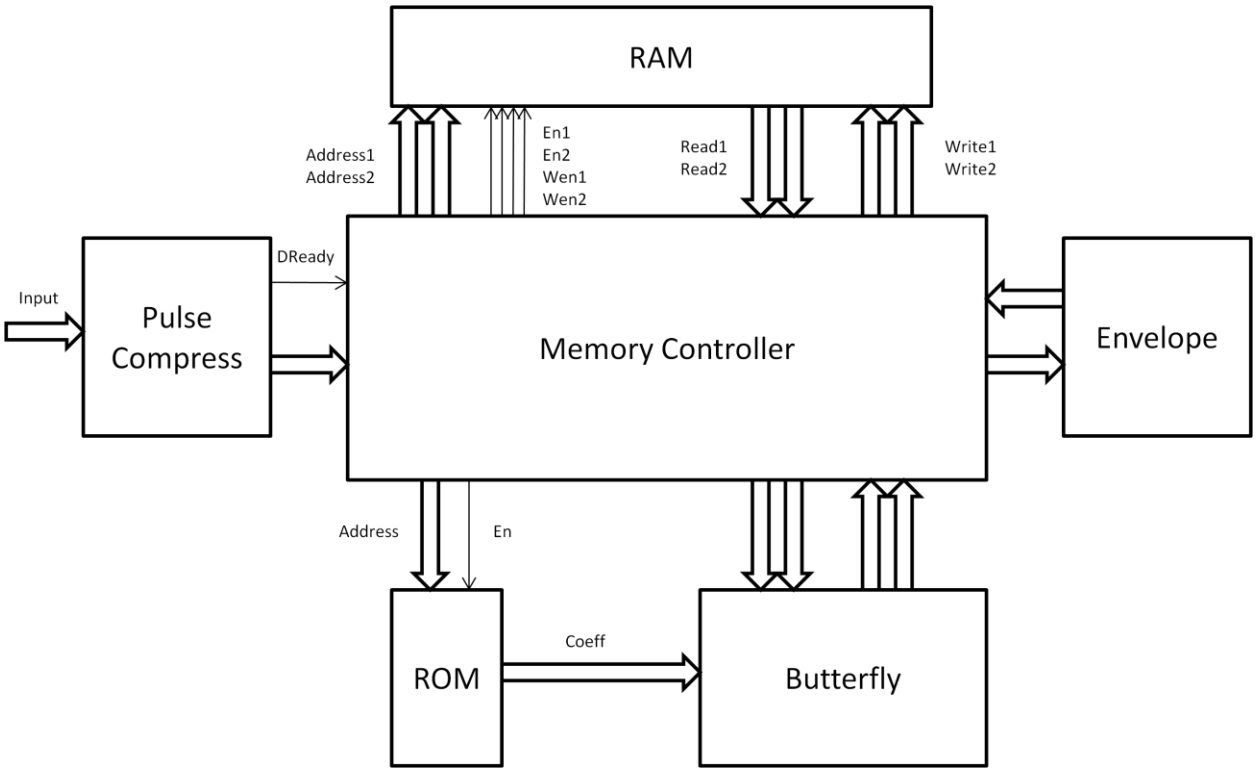

FIGURE 12 - SYSTEM REGISTER TRANSFER LEVEL BLOCK DIAGRAM

The real-imaginary input pair is entered into the system by feeding it directly to the Pulse Compression unit. The pulse compressed samples are handed over to the Memory Controller which is in charge of controlling the dataflow between the various components in the system. This means generating proper addresses to access the RAM and ROM storage units and coordinating the redirection of data to/from the RAM and functional components. The results of pulse compression, FFT processing, and envelope detection are all stored in the dual-port RAM. The ROM unit stores the twiddle factor coefficients (sine and cosine values), which are necessary for the FFT algorithm, in fixed-point binary format. These values are only used for the FFT computation and hence are supplied directly from the ROM to the Butterfly unit. The Butterfly component implements a multiply-and-accumulate operation which computes a 2-point DFT, leading to a break-down of the entire DFT operation into a radix-2 FFT algorithm. Finally, the phase information of the complex filtered samples is removed in the Envelope unit.
The operation of each component is further explained in more detail in the sections that follow.

6.2. RAM

As mentioned previously, the RAM unit functions as a storage memory for the results of the pulse compression, FFT processing (including intermidiate results), and envelope detection. Each cell of the RAM can hold one complex sample. As we will see in a later section, the butterfly module operates on two complex samples and also produces two complex results. This makes a RAM with two read and write ports a suitable choice for implementation of the radix-2 FFT algorithm.

The RAM used in the design of the DPS chain is implemented as a dual-port block RAM with two write ports. The RAM is realized on the block SelectRAM+ resources incorporated into the Virtex-II Pro FPGA devices, and thus consumes almost no logic on the chip. The behavior of the RAM is similar to a regular register and it has a synchronous operation. The Write operation has been given precedence over the Read operation and as a result no new read occures when a cell has been just written to. This will prevent unneccessary oscillations on the input to the functional units.

The size of the RAM is determined by the size of the input data matrix collected during a CPI. For an input matrix of size 64x64 this trasnlates into a memory cpacity requirement capable of holding 4096 elements. A complex number of size 32 bits (two 16-bit numbers) will result in a storage demand of approximately 131 kb. This will not be an issue as the chosen FPGA chip has a about 2500 kb of block RAM available [4].

The implemention in the VHDL language is based on the Xilinx Sythesis Technolgoy (XST) guidelines [5] and is perfectly sythesizable. Figure 13 presents a block view of this RAM.

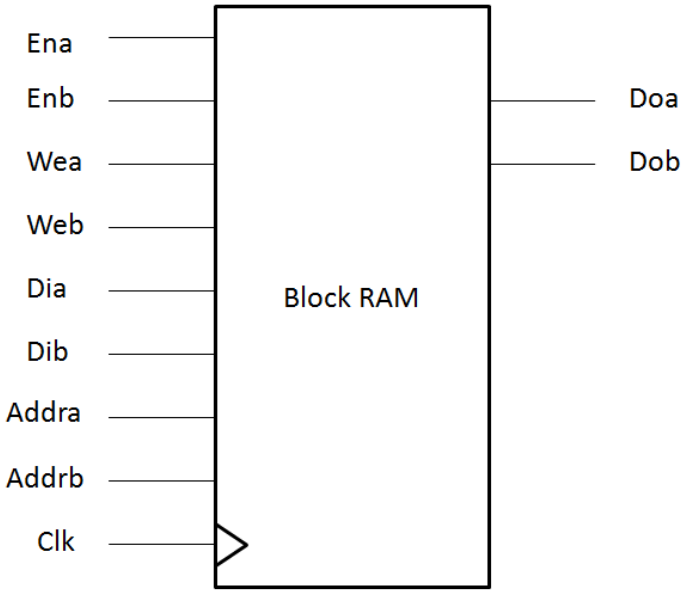

FIGURE 13 - DUAL PORT BLOCK RAM WITH TWO WRITE PORTS

6.3. Pulse Compression

Pulse compression is the convolution of the received signal with its pulse code [6]. The coded pulses at the transmission node are filtered with a match filter in the receiving end. The technique utilized to compress the pulses here is binary phase coding. The figure below depict the RTL structure of the Pulse Compression module. As can be seen it operates as a simple filter on the input samples.

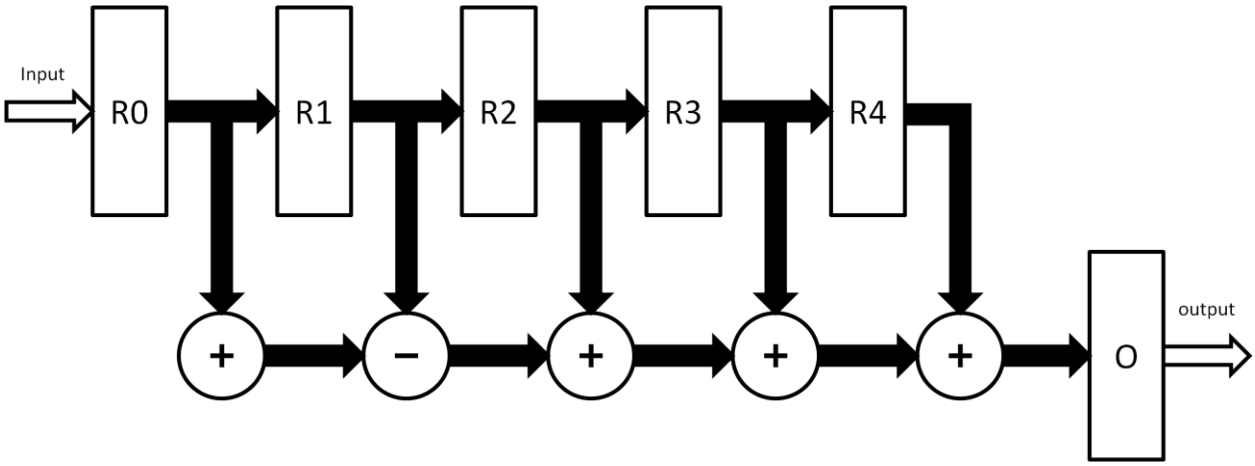

FIGURE 14 - PULSE COMPRESSION

The Pulse Compression unit compresses each row of input samples (i.e. all the samples in a particular pulse) individually, and one output sample (real-imaginary pair) is produced each clock cycle. As soon as the first output sample is ready, the DataReady signal informs the Memory Controller of its availability.

6.4. ROM

The ROM contains the twiddle factors required for the calculation of the FFT algorithm. Since the size of the ROM is rather small compared to the RAM size, it has been designed to be synthesized utilizing the FPGA logic, although it could be easily altered to use the block RAM available on the device.
The contents of the ROM which are sine and cosine values and the ROM size depend on the number of points of the FFT algorithm and the number of pulses in the CPI. The twiddle factors take the following form:

$$W_N^k = \cos\frac{2\pi k}{N} - \mathrm{j}\sin\frac{2\pi k}{N}$$

In which N signifies how many points are in the FFT, and k is in the range [0, $N/2$)
The values of sines and cosines are evaluated in MATLAB in advance and converted to fixed-point representation with a formation of (2, 7), which means 7 bits are set aside for fraction and 2 bits for sign and magnitude (two's complement representation). This will yield a range of values covering the span of

[-1.0, 1.0]. To make the adaptation of the design to different FFT sizes easier, the converted fixed-point values are stored in a text file and later loaded into the ROM in VHDL.
The twiddle factors are accessed by their index k which translates into the corresponding row number in the ROM. Therefore the ROM has $N/2$ cells (rows) and each cell holds both cosine and sine values.

6.5. Butterfly

The butterfly operation forms the basic fundamental component of the radix-2 FFT algorithm. The structure of the radix-2 decimation-in-time butterfly unit is provided in Figure 15.

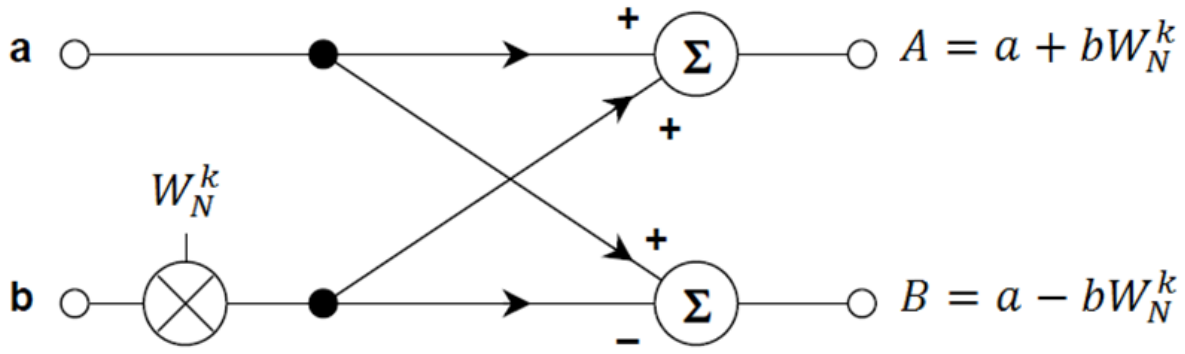

FIGURE 15 - RAIDX-2 DECIMATION-IN-TIME BUTTERFLY

This butterfly unit consists of one complex multiplication and two complex additions/subtractions, as shown above. The complex multiplication involves:

$$W_N^k = \cos\frac{2\pi k}{N} - \mathrm{j}\sin\frac{2\pi k}{N}$$
$$b = Rb + j\,Ib$$
$$b \times W_N^k = (Rb + j\,Ib) \times (\cos\frac{2\pi k}{N} - \mathrm{j}\sin\frac{2\pi k}{N})$$
$$b \times W_N^k = \left(Rb \times \cos\frac{2\pi k}{N} + Ib \times \sin\frac{2\pi k}{N}\right) + j(Ib \times \cos\frac{2\pi k}{N} - Rb \times \sin\frac{2\pi k}{N})$$

As can be seen there are four simple multipications, one addition, and one subtration in the complex multiplication. The simple multiplications are with twiddle factors, which are fractions in fixed-point format, and hence after the multipication operation the least significant bits are truncated to remove the fraction in the result, and only the whole (integer) part is taken. Finally, the result of the complex multiplication is both added to and subtrated from the other complex input to the butterfly and the results are supplied as the outputs of this module.
To demonstrate how the FFT algorithm is done in practice, Figure 16 illustrates the detailed process of computing the simple case of an 8-point decimation-in-time radix-2 FFT algorithm showing all the multiplications and additions involved [7].

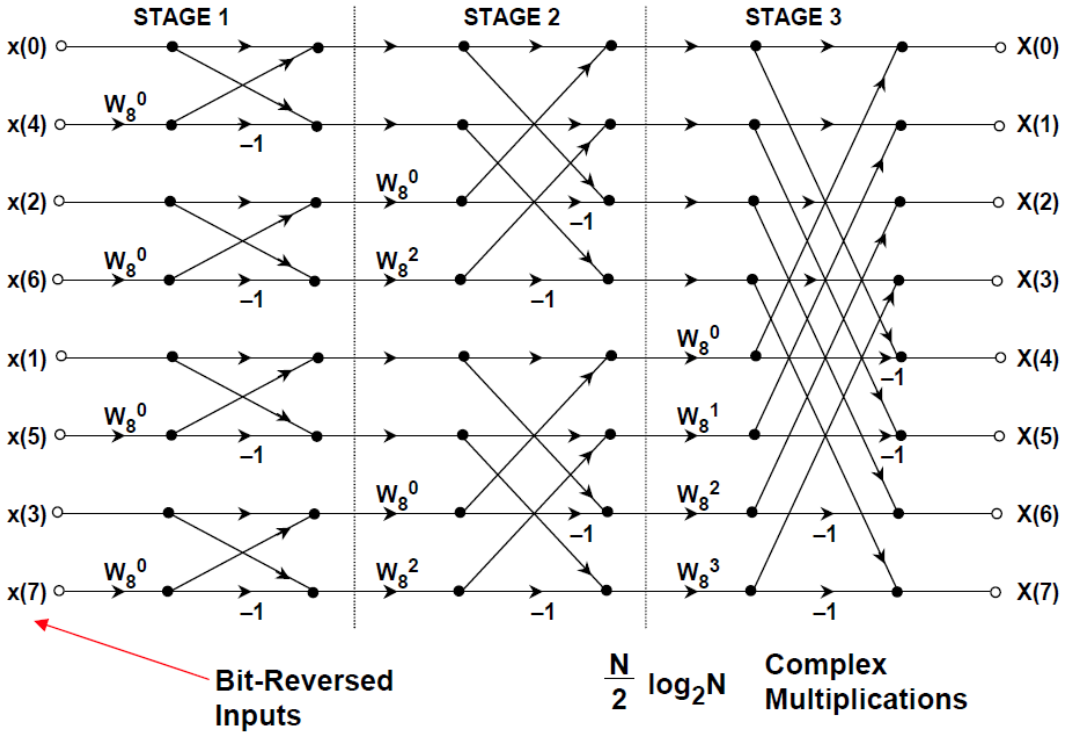

FIGURE 16 - 8-POINT DECIMATION-IN-TIME RADIX-2 FFT

In the case of an 8-point FFT, there are $(\frac{8}{2} log_2 8)$ or 12 butterfly operations needed in total, and for an FFT with 64 points the required number of operations will increase to 192. But as mentioned earlier there's only one butterfly unit available in this design and as a result the FFT processing is performed in multiple clock cycles, with each cycle supplying the required inputs to the butterfly. Given that generating the proper addresses to get the correct values from the RAM is rather complex, a technique called bit-reversal is utilized to make this process a little simpler. Taking care of all these issues, i.e., address generation, bit-reversal, twiddle selection, and data handling falls on the shoulders of the Memory Controller unit, which is explained in more detail in a later section.

6.6. Envelope

The Envelope Detection unit removes the phase information of the FFT processed samples. This is done using an approximate calculation called absolute-value calcultion formulated as in the following [6]:

$$A = \max(|I|, |Q|)$$
$$B = \min(|I|, |Q|)$$
$$y = \max(A, 0.875 \times A + 0.5 \times B)$$

The The block diagram implementing the formulas above in hardware is illustrated in the figure shown below:

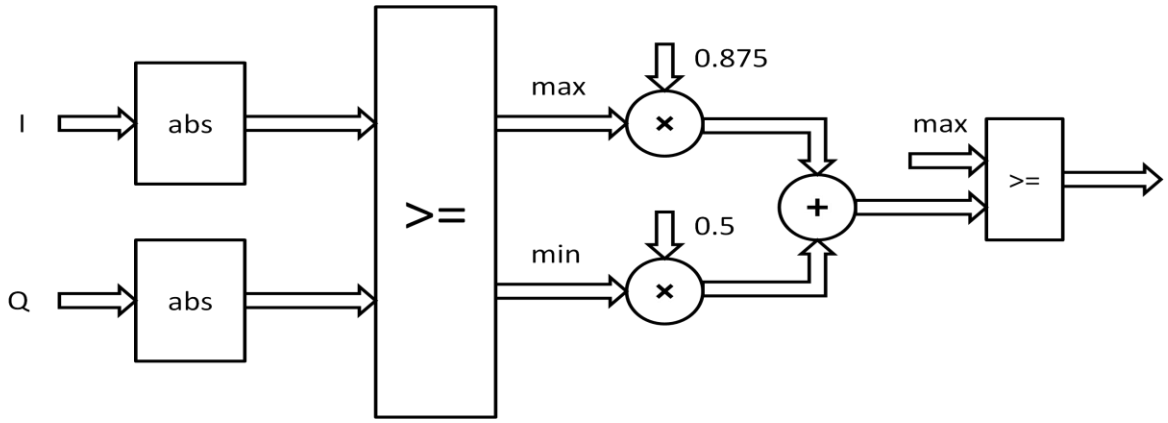

FIGURE 17 - ENVELOPE DETECTION

6.7. Memory Controller

Memory Controller, as its name suggests, cotrols the flow of data between the various components in the system. It acts as a state machine with states signifying the stages of the DSP signal processing chain. Initially and after the system resets, it resides in the State_Standby state, ready for the start signal from the Pulse Compression unit. When the PulseCompressDataReady is asserted, a state transition occurs into the State_FillRAM stage and the Memory Controller starts receiving the compressed samples from the Pulse Compressin unit and stores them one by one in the RAM, filling each row corresponding to a pulse in the CPI until all rows are full. Only one write port of the dual port RAM is used at this stage.
The data connections for the State_FillRAM state are shwon below:

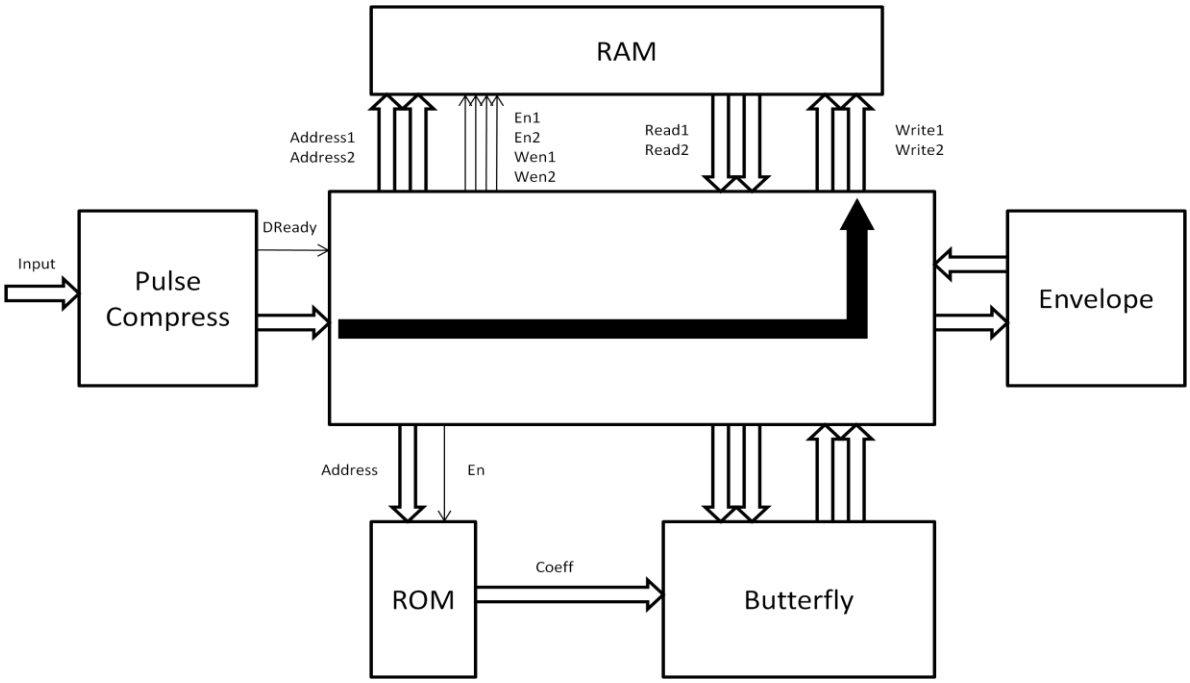

FIGURE 18 - PULSE COMPRESSION DATA CONNECTION

Now that all the samples are available in memory, they can be processed using the FFT algorithm. The processing commences by entering into the State_FFT state. In this state RAM read ports are connected to the inputs to the butterfly unit and its write ports are attached to the outputs of the butterfly illustrated by the block diagram in Figure 19.

The FFT processing is performed on each column of the RAM. Processing each column involves a number of stages based on the FFT points. Two samples are read each cycle and their results are stored in the next cycle in the same RAM cells. As partly discussed in the Butterfly section ealrlier, the last write operations in processing each column do not bit-reverse the address, which means the cells that are read from are different from the cells that will be written to. This is ignored in the actual implementation by storing the final results in bit-reversed addresses and applying the bit-reversal to the address again in the State_Envelope (or any later states), practically simulating the correct procedure.

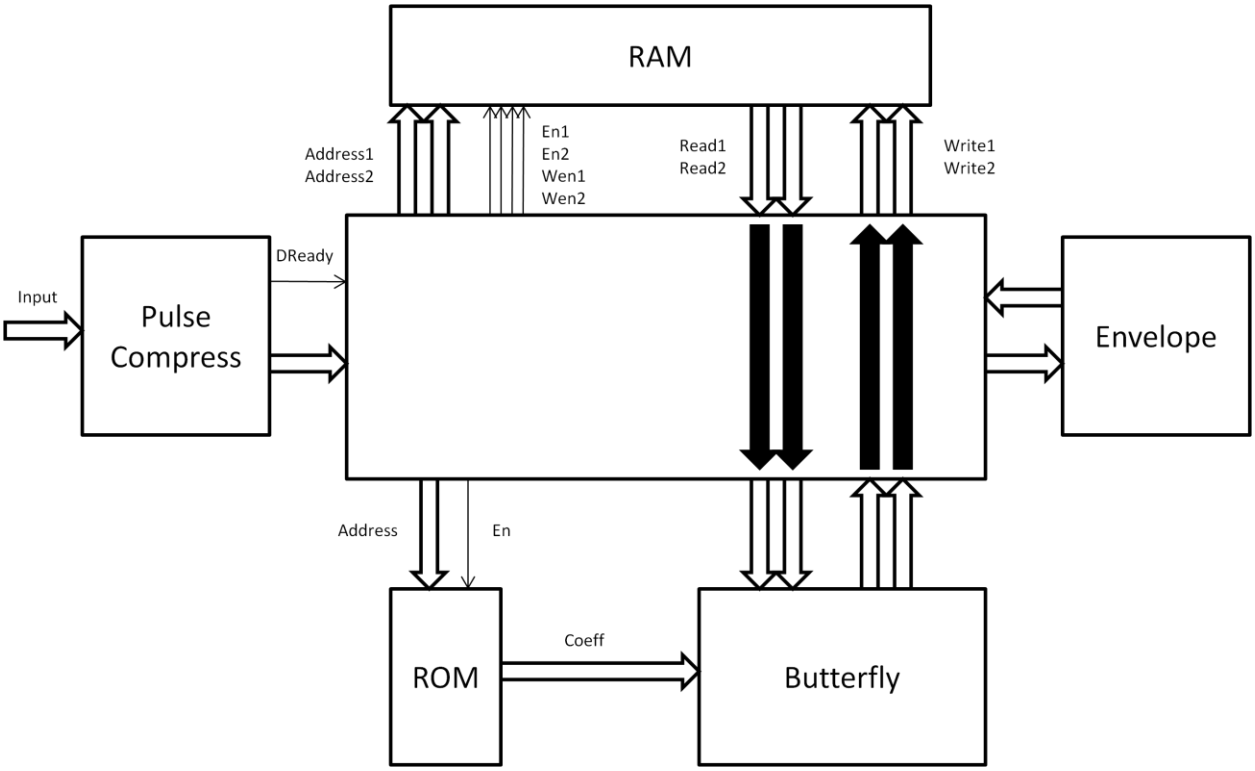

FIGURE 19 - BUTTERFLY DATA CONNECTION

After FFT processing is complete, envelope detection starts by connecting one RAM read port to the input of the Envelope unit and one RAM write port to its output as demostrated by Figure 20. Again samples are read one at a time and the results are written at the next cycle. To read the samples from memory the address bits will be reveresed as explained previously.

The output of the Envelope Detection unit has its phase removed and is no longer a complex number and hence occupies half the size of the RAM cell which used to hold a complex value. To resolve this, instead of using another RAM, we append the ramaining bits with a duplicate version of the value to be stored and deposit it in the same RAM used in earlier stages.

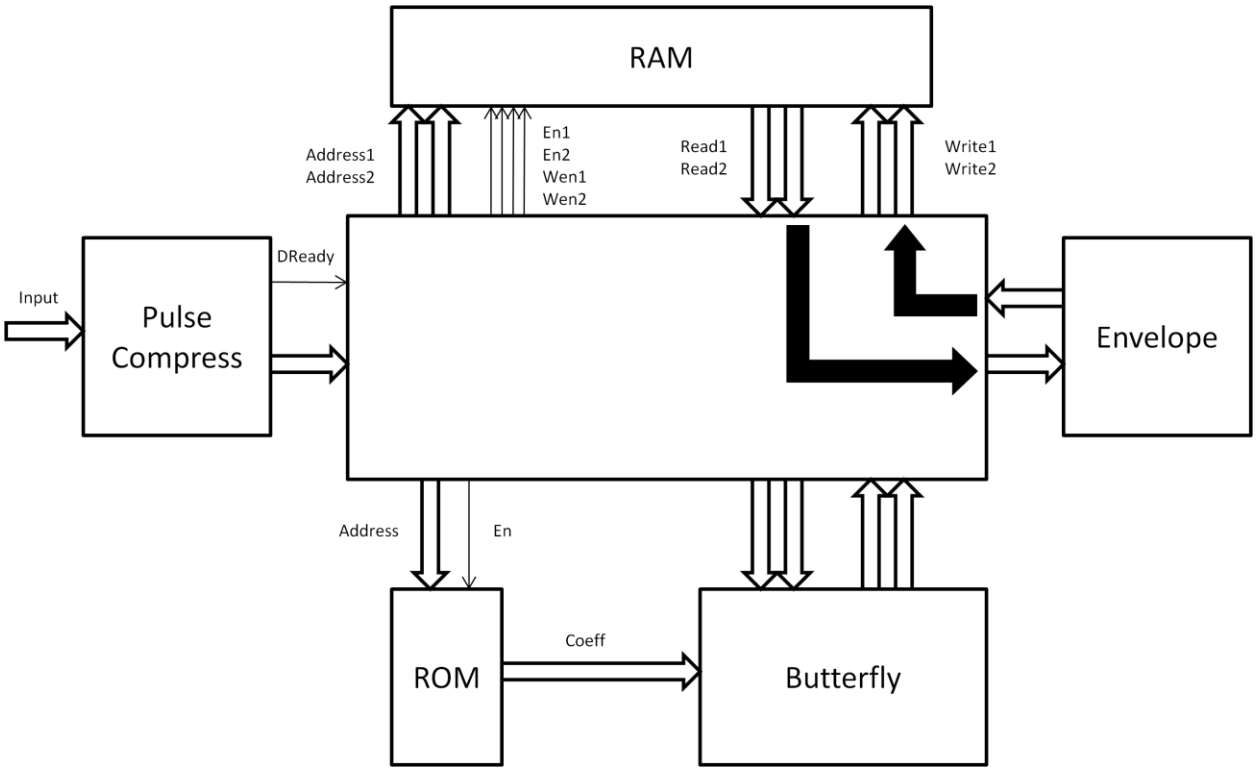

FIGURE 20 - ENVELOPE DATA CONNECTION

7. RNS Design Modifications

7.1. Pulse Compression

The input binary values need to be converted to RNS formation. This is performed in the Pulse Compression unit before the registers capture the inputs. By using the 3-moduli set {P1, P2, P3} = $\{2^n - 1, 2^n, 2^n + 1\}$ each integer number in this residue class will have a 3-tuple representation given by:

$$X \rightarrow (X1, X2, X3)$$

with a dynamic range of:

$$M = P1 \times P2 \times P3 = 2^{3n} - 2^n$$

The binary values have an original binary bit-length of 16. To convert these values into RNS numbers with three segments their bit-length has to be increased to become a multiple of three. This is taken care of by padding the numbers with their sign bit (sign extension) to get a 24-bit digit and then converting it to RNS. This means the utilization of the moduli set of {255, 256, 257} and a dynamic range of 16776960.
The arithmetic units used in the Pulse Compression unit are replaced with their RNS counterparts.

7.2. RAM

The only change in the RAM is to modify its data ports to accommodate for the RNS numbers.

7.3. ROM

For consistency the same ROM used in the binary system is used in the RNS system as well, that is, the contents of the ROM will be still in binary fixed-point representation and their value stays intact. This is possible by thinking of the fractional point moved all the way to the right, effectively multiplying the cell contents by 128 (there are 7 fractional bits in each number). Of course this entails having to deal with it later in the Butterfly unit, since the results of the multiplications will be all scaled by 128.

7.4. Butterfly

The most important changes happen in the Butterfly unit. As mentioned in the section above, the twiddle factors are stored in fixed-point binary format which means they have to be converted to RNS numbers here in the Butterfly unit. Furthermore it was pointed out that these coefficients are in effect scaled by 128. This causes the result of the RNS multipliers to be a scaled version of what the correct

result would be. Left unaccounted for, this leads to incorrect results spreading through the FFT calculation. Hence we need to scale the results down right after the multiplication has taken place, i.e., we have to divide the result by 128 and then give the scaled-down result to the RNS adders/subtractors. The division operation is not trivial in the RNS number system and we instead convert the multiplication results to binary, perform the scaling operation in that format (i.e., discard the seven least significant bits) and then convert the scaled values back into RNS. This is illustrated in the figure below.

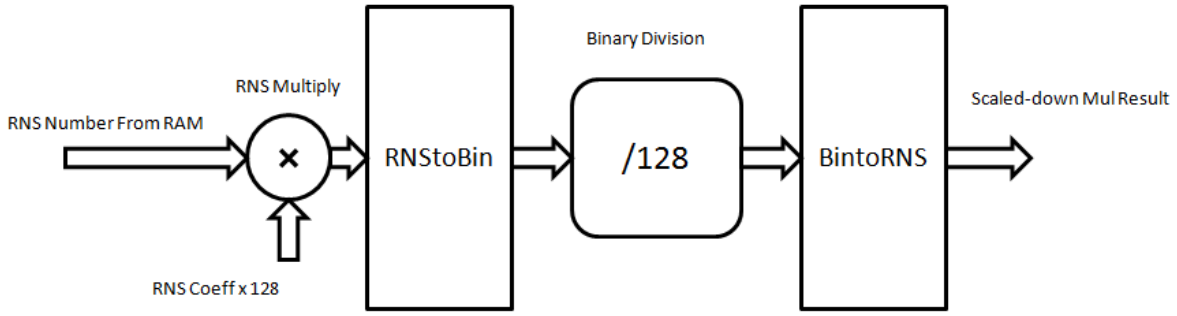

FIGURE 21 - RNS SCALING

7.5. Envelope

Envelope detection involves taking the absolute value of the inputs and a number of comparisons. These are not intuitive operations in RNS and as a result just as the RNS numbers enter the Envelope Detection module, they will be converted to binary format. There will be no reverse conversion and the results are stored back in the RAM in keeping their binary format.

7.6. Memory Controller

The control unit requires no changes as the operation of the system remains exactly the same. Only the width of the data exchange ports is modified to accommodate for the RNS numbers.

8. Simulation

To verify the operation of the system a MATLAB model of the DSP chain is provided. This model takes the input through a matrix and performs the processing on it and finally generates the output matrix. In order to put the DSP system described in the VHDL language under test, the input and output matrices in the MATLAB model are converted to binary format and stored in text files similar to the way the twiddle ROM contents were produced. Each row in these files stores one sample. The contents of these test vector files are loaded into the VHDL testbench and applied to the DSP system.

Verification involves comparing the outputs of the FFT processing stage with the output vectors generated by MATLAB. Of course MATLAB uses floating-point numbers for calculation and our system processes values in binary fixed-point and thus dissimilarities are to be expected. Most of the results, that is about 70 percent of them, are less than 10% different from the MATLAB model results and the discrepancies in the rest are due to the output value being too close to zero which makes the difference bigger than 10% in some cases. Additionally, the outputs were successfully tested to check that the difference with the MATLAB model results would stay within a limited window.

The simulation was carried out using the ModelSim software. Figure 22 provides a segment of the simulation window showing real and imaginary system outputs and their expected values, the corresponding differences, the amount of error for each output sample produced, and the percentage of the output results falling below the 10% difference mark.

Both the binary and the RNS systems were tested and they produce very similar results.

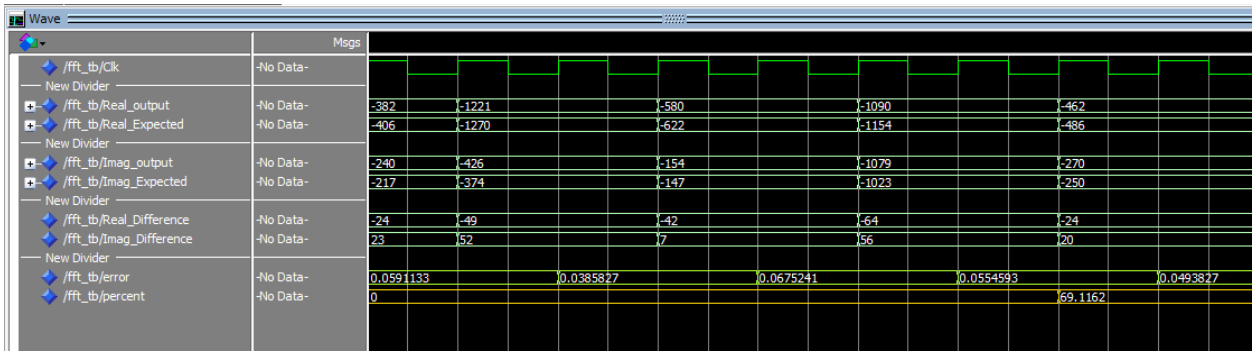

FIGURE 22 - MODELSIM SIMULATION WINDOW

9. Synthesis and in-circuit Verification

9.1. Verification

The Xilinx Virtex-II Pro FPGA device was selected as the target for synthesis. The XC2VP30 device model houses more than 30,000 logic cells, 136 18-bit multipliers, and about 2500 kb of block RAM [4]. Unfortunately this device is no longer supported by the newer versions of the Xilinx ISE program and consequently Xilinx ISE 10.1 was used to perform the synthesis and hardware mapping.
To verify the system in-circuit, two more ROMs were introduced to the system, one holding the test vector samples and the other the test vector results. A TestGenerator unit was designed to apply the samples one by one to the DSP system and a TestValidator unit verified the results generated by comparing them to the vectors stored in the test vector result ROM. If the test was passed successfully a signal would indicate this by lighting an LED on the board. Because of the big size of the vector ROMs, they are synthesized to be realized using the block RAM on the chip.

9.2. Synthesis Analysis

In order to provide a fair comparison between the binary and the RNS systems, the bit-length of the input samples to the binary system is increases from 16 to 24 bits. Furthermore, both systems are configured to run at the same clock speed.
The binary system has a 4% utilization of the 4-input LUTs. In comparison, the RNS system uses 13% of the total number of LUTs available on the chip. Flip-flop utilization of both systems is almost identical. This is equivalent to an RNS area usage of almost three times that of the binary system.
Xilinx XPower provides power estimates for the FPGA design after Place & Route has taken place. To conduct an acceptable power estimation, a VCD (Value Change Dump) file is required. XPower uses this file to set frequencies and activity rates of internal signals. The *vcd* command included in ModelSim is used to generate the VCD file for the unit under test. This file is supplied as the input to the XPower analyzer. To perform the power analysis a command is issued in the TCL Shell of the Xilinx ISE after Place & Route in the following format [8]:
xpwr DSP.ncd DSP.pcf -s DSP.vcd
The power estimates produced using this command for the binary and RNS systems are provided in Figures 23 and 24 respectively. The RNS system consumes about 60% more power compared to the binary system.
Based on the results obtained, the RNS system has higher area and power consumption requirements. By analyzing the design of the RNS system it can be noticed that the lack of intuitive support for division, sign detection, and comparison in the RNS are the main issues in achieving better results. The operations mentioned are an essential part of the MPD radar DSP processing chain. The RNS weakness in dealing with them causes the extra forward and backward converters in the system, which ultimately reduces its potential.

Power summary		I(mA)	P(mW)
Total estimated power consumption			134

Total Vccint	1.50V	57	85
Total Vccaux	2.50V	10	25
Total Vcco33	3.30V	1	4
Total Vcco25	2.50V	8	19

Clocks		3	4
Inputs		0	0
Logic		1	2
Outputs			
Vcco25		6	16
Signals		3	4

Quiescent Vccint	1.50V	50	75
Quiescent Vccaux	2.50V	10	25
Quiescent Vcco33	3.30V	1	4
Quiescent Vcco25	2.50V	1	3

FIGURE 23 - BINARY SYSTEM POWER ESTIMATION

Power summary		I(mA)	P(mW)
Total estimated power consumption			212

Total Vccint	1.50V	102	153
Total Vccaux	2.50V	10	25
Total Vcco33	3.30V	2	6
Total Vcco25	2.50V	11	27

Clocks		4	6
Inputs		0	0
Logic		17	26
Outputs			
Vcco25		10	24
Vcco33		0	2
Signals		31	46

Quiescent Vccint	1.50V	50	75
Quiescent Vccaux	2.50V	10	25
Quiescent Vcco33	3.30V	1	4
Quiescent Vcco25	2.50V	1	3

FIGURE 24 - RNS SYSTEM POWER ESTIMATION

10. Conclusion

The implementation of the MPD DSP processing chain targeted toward an FPGA platform for binary and RNS systems was followed by comparing their respective outputs, area utilization, and power consumption. The RNS system performed very similarly to the binary system in calculation of the results but was inferior to its binary counterpart in both area utilization and power consumption perspectives. RNS is highly efficient in handling arithmetic addition, subtraction and multiplication but it unfortunately is incapable of doing the same for other operations such as division and comparison. This proves to be fatal for the RNS system to compete with the binary system as it needs forward and backward convertors to be able to handle the operation correctly.

References

[1] G. W. Stimson, Introduction to Airborne Radar, Second Edition, SciTech publishing, 1998

[2] A. Åhlander, Efficient Parallel Architecture for future Radar Signal Processing, Göteborg : Department of Computer Science and Engineering, Chalmers University of Technology ; Halmstad : School of Information Science, Computer and Electrical Engineering, Halmstad University, 2007

[3] Taylor, F.J. , Residue Arithmetic A tutorial with Examples, Computer, IEEE Computer Society, vol 17 issue 5, pp 50 – 61, May 1984

[4] Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet v4.7

[5] XST User Guide 10.1

[6] M.Taveniku, A.Åhlander, M.Jonsson, and B.Svensson, ”A multiple SIMD mesh architecture for multi-channel radar processing”, Proc.International Conference on Signal Processing Applications & Technology, 1996, (p.5-8)

[7] Walt Kester, Mixed-Signal and DSP Design Techniques, Fast Fourier Transforms

[8] Xilinx Development System Reference Guide 10.1