Progress In Electromagnetics Research B, Vol. 92, 47–70, 2021

Design and Implementation of Field-Programmable Gate Array Based Fast Fourier Transform Co-Processor Using Verilog Hardware Description Language

Yung-Chong Lee[1], Yee-Kit Chan[2, *], and Voon-Chet Koo[2]

Abstract—In this research project, the hardware implementation of a Field-Programmable Gate Array (FPGA) based Fast Fourier Transform (FFT) will be carried out by using Verilog Hardware Description Language (HDL). Since FFT serves as the core for the Range Doppler Algorithm (RDA) in Synthetic Aperture Radar (SAR) processing, it is of paramount importance to evaluate the algorithm and its computational complexity for the design of an efficient FFT hardware architecture. The design process and Verilog hardware description language which is used to describe and model a digital FPGA-based SAR processor will be introduced. Detailed explanation of the hardware implementation for FFT and Inverse Fast Fourier Transform (IFFT) in SAR processing are thus presented. The performance evaluations of the proposed processors including the comparison of the proposed processor with MATLAB-based processor, timing considerations of the processor, and lastly the hardware resources usage considerations are delivered at the end of this paper.

1. INTRODUCTION

SAR enhances radar's information acquisition capability and information awareness ability by obtaining high resolution imageries. Now, SAR has been widely used and become an important tool in airborne as well as space borne radar systems for civilian and military users. The potential of SAR in a diverse range of applications such as sea and ice monitoring [1], mining [2], oil pollution monitoring [3], oceanography [4], snow monitoring [5], classification of earth terrain [6], automatic change detection for multitemporal images [7], and multipass Coherent Change Detection (CCD) in polarimetric radar image [8] have led to the development of a number of airborne and spaceborne SAR systems [9–14].

An extensive literature exists on various processing techniques that generate images from the radar returns of a SAR [15–20]. The SAR signal processing can be broken into two phases: range processing and azimuth processing. Most coherent radars use some forms of modulation or coding of the transmitted waveforms to improve resolution. The main purpose of SAR signal processing is to carry out the reverse transformation from signal space to target space via two dimensional inverse convolution. The bandwidth of the pulse determines the nominal resolution of the system in the cross track direction, and the azimuth or along track beamwidth of the antenna determines the nominal along track resolution.

SAR can provide high-resolution image of range by emitting large time-bandwidth multiplication signal. Resolution much finer than the length of ground intercepted by the physical antenna's beam in the along track direction is achievable by properly processing the signal returned during the ground illuminated by the physical beam. Matched filter is the touchstone allowing synthetic aperture

Received 28 December 2020, Accepted 17 March 2021, Scheduled 29 March 2021
* Corresponding author: Yee-Kit Chan (ykchan@mmu.edu.my).
[1] iRadar Sdn. Bhd., Melaka, Malaysia. [2] Faculty of Engineering & Technology, Multimedia University, Melaka 75450, Malaysia.

processing to achieve azimuth resolution finer than that provided by the physical antenna's azimuth resolution. As a result, a two-dimensional high-resolution image can be obtained.

In the application of digital SAR processing, signal processing poses a significant challenge because of its stringent computation power and data storage requirements. A high resolution SAR system captures the return echoes and yields huge amount of raw data. The size of these data ais dependent on several parameters such as dynamic range, spatial resolution, and also sampling frequency of the SAR system. The majority of prior works in the design and development of digital SAR systems are based on DSP based or computer-based SAR processor.

Therefore, this research work is motivated by the need to overcome the challenges that arise in the hardware implementation of SAR processor due to inflexibility of customised hardware solutions as well as stringent space and power requirements in the SAR platform. In this paper, a FPGA-based system is proposed for the hardware implementation of an efficient digital SAR processor due to its flexibility, reconfigurability and exploit pipelining. In addition, FPGA-based system is able to fulfill the demands of massively parallel processing, lower power consumption, as well as smaller form factor.

This paper elaborates the design and development of the FPGA-based FFT and IFFT co-processor by using Verilog HDL. Section 1 introduces the SAR image formation and Verilog HDL. The introduction of Verilog HDL that specifies a digital system in a wide range of levels of abstraction is also presented. Section 2 presents and explores the hardware architecture and specification for the design and development of FFT and IFFT coprocessors using Verilog HDL. The target hardware platform architecture for the implementation of the SAR processor is introduced as well. Besides, the performance metric of the proposed processor is evaluated in terms of timing consideration, hardware usage resources consideration, power consumption, and accuracy in Section 3. At the end of Section 3, comprehensive performance comparisons among MATLAB, IP core, and the proposed coprocessors are presented.

1.1. SAR Image Formation

The generation of a SAR image is a computational intensive task [21]. Both range compression and azimuth compressions can be summarised as time domain convolution with respective reference signal. In order to reduce the computational load, the time domain correlation processes are often completed in frequency domain. The transformation between time and frequency domains is performed by FT

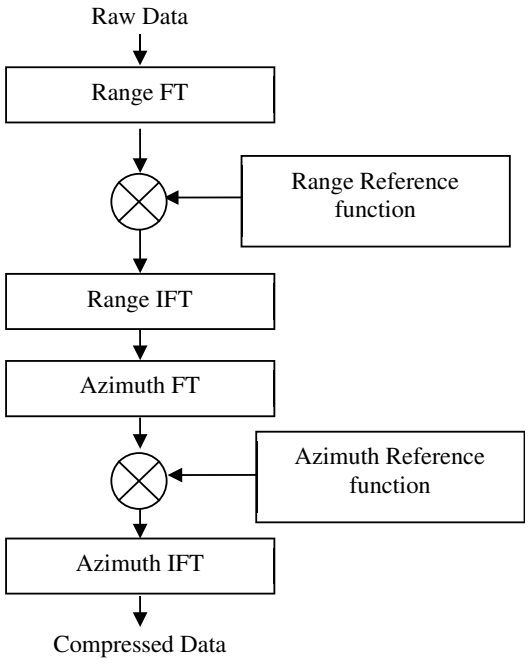

Figure 1. SAR processing block diagram.

algorithms and IFT algorithms. They require most of the processing power of SAR signal processing. The block diagram of SAR signal processing based on Range Doppler Algorithm (RDA) is shown in Figure 1 [22–24].

At present time, the real time SAR imaging has a variety of applications such as identifying man-made objects on the ground, executing search and rescue operation, and estimating earth surface activities. Therefore, the hardware implementation of FPGA-based FFT/IFFT core for real time SAR processing is essential.

Real time SAR images can be used for national monitoring and management of earth resources as well as disaster monitoring. The importance of unlimited and timely supply of the required SAR images cannot be understated. Besides, the generation of a two-dimensional image is a computational intensive task. Conventional range compression and azimuth compression utilise FFT and IFFT in order to perform convolution with respective reference signal. In fact, FFT and IFFT are estimated to occupy about 70% of the total computation operation in SAR image formation. Thus implementation of multiple processors and the application of parallel digital signal processing platform are needed in order to realize the real time SAR processing.

1.2. FPGA-Based On-Board Processor

FPGA has unique characteristics of reconfigurability, application power efficiency, and high throughput rate, which allows user to perform various direct controls for the implementation of a digital system, including a digital SAR processor. NASA and USAF have identified that on-board processing is an indispensable technology required in order to improve the performance of a SAR system [25]. In 2007, Kuon and Rose showed that the main advantage of FPGA is the reprogrammable ability, and therefore, it is the perfect and best matched candidate for the implementation of real-time processing systems. In particular, researchers from JPL proposed an on-board FPGA-based SAR processing system which is possible to be implemented for the application of SAR [26]. The proposed architecture consists of major objectives and several challenges for an FPGA-based SAR processor. From the proposal, it is shown that the implementation of an FPGA-based on-board processor in SAR mission is feasible.

By implementing SAR on-board processing with the use of an FPGA, the FPGA-based on-board processor can be adapted into different SAR systems with different specifications as the processor is scalable and reconfigurable according to the user requirements. As long as the hardware resources are sufficient, the processor can be reconfigured anytime in order to match the desired requirement. The term of scalability is defined as the processing power, and architecture of the processor can be easily reconfigured while maintaining the same hardware platform without any physical hardware modification. Besides, an FPGA-based SAR processor has the ability to make trade-off between hardware and software in order to maximise the efficiency and performance of the processor. For instance if a bottleneck is identified to be the SAR software algorithm, a customised FPGA-based co-processor can be designed particularly for the specific algorithm. The co-processor is attached to the main processor through low latency channels for real-time on-board processing due to high throughput rate.

1.3. Verilog Hardware Description Language Based Design

Verilog was originally designed in 1993 at Gateway Automation as a hardware modeling language which was associated with their simulator products [27]. Verilog language is a hardware description language (HDL) that specifies a digital system in a wide range of levels of abstraction. The unique feature of this language is that it supports the early conceptual phases of a design with its behavioral level of abstraction followed by its structural abstractions in the later implementation phases. The language supports a design ranging from the algorithm-level to the gate-level, and even to switch-level. The design of digital systems using HDLs has become an essential way as it includes hierarchical constructs and descriptions by allowing the user to define the complexity of the designs [28].

Verilog HDL-based design is gaining popularity due to several bottlenecks that appear in the traditional schematic-based design. In traditional schematic-based design, one of the major bottlenecks faced is that it is not able to provide the capability of design reuse as it is not easy to handle the designs with large number of gate counts in a schematic. A design flow of the FPGA HDL-based SAR Processor paradigm is portrayed in Figure 2. Generally, the design flow can be divided into two sections

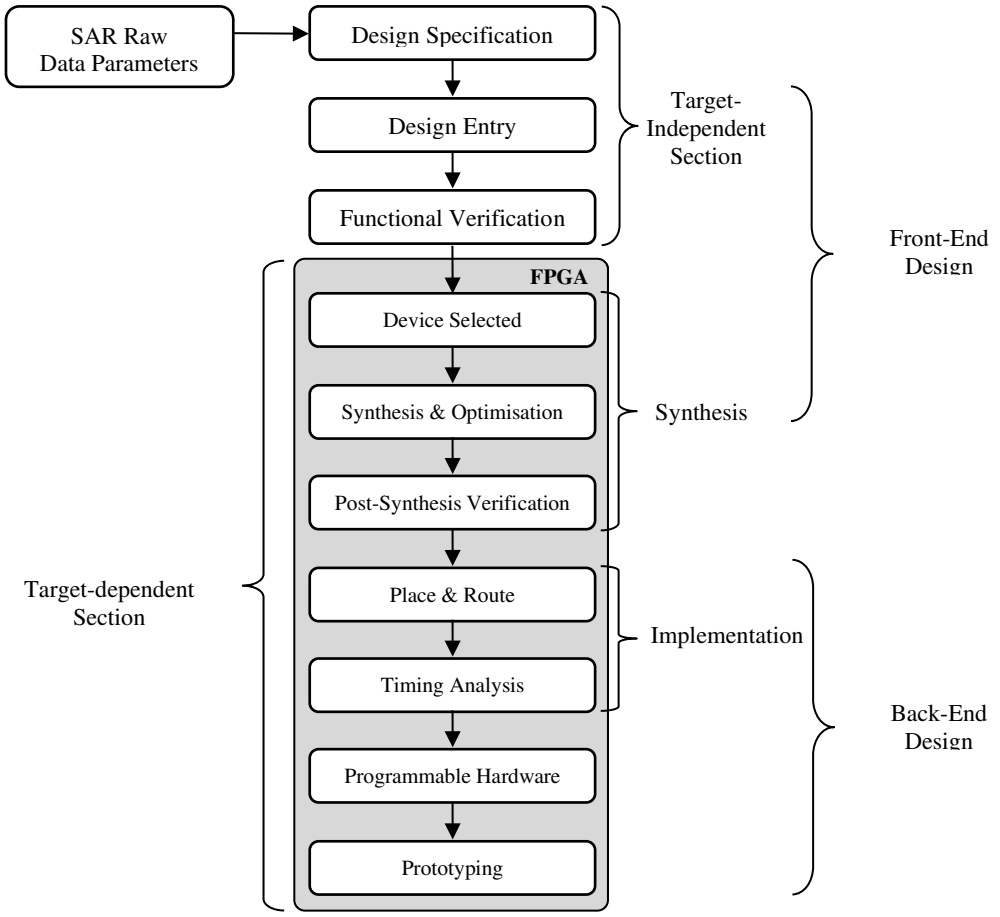

Figure 2. A design flow of the FPGA HDL-based SAR processor paradigm.

which are target-independent section and dependent section. In the target-independent section, the design begins with the desired specification according to the SAR raw data parameters, followed by design entry (Verilog HDL), and ends with functional verification. The functional verification is an important step as it assures that the functionality of design entry is correct and corresponds to the required specification as the completion of target-independent section.

In target-dependent section, it is further branched into several subsections due to the complicated structure of the FPGA devices. These subsections can be roughly classified into three major parts which are synthesis, implementation, and followed by programming. The synthesis process is composed of the FPGA device selection, design synthesis, and optimisation, together with the post-synthesis verification. In the synthesis process, the switching function designed by the user is optimised based on the selected FPGA device. The mapping of the abstract logic elements to the actual logic blocks as in the device is carried out after optimisation. Upon completion of the synthesis and optimisation process, the synthesised result is verified through post-synthesis simulation or formal verification.

As for the implementation process, it consists of place and route operations and timing analysis. The process starts with the placement and routing of components where the logic blocks are placed into the actual logic elements, and the related interconnects are then set up and routed in order to connect the placed components. After the place and route operation, timing analysis is required in order to verify whether the desired timing constraints meet the required specification. This step is necessary due to the natural propagation delays associated with the logic elements and interconnect. Typically, there are two types of timing analysis, namely dynamic timing analysis (DTA) and static timing analysis (STA) [29]. The former is done by simulation, and the latter is done by analysing the timing paths of the design without performing any actual simulation. Once the implementation process is completed,

the programming file for the design will be generated by the CAD tool in order to program the target device. The programmed device is often used in a real world experiment in order to see if it indeed works as expected. In the design flow of a digital system with HDL-based approach, front-end design is usually referred to as the target-independent part and synthesis part with the remaining part as back-end design.

2. RESEARCH METHOD

2.1. Design and Implementation of FFT and IFFT Coprocessors

SAR signal processing is a computational intensive task which involves several complex operations and algorithms. It is important to introduce Fourier Transform (FT) in the context of SAR as it plays a major role in convolution operations such as matched filtering which heavily utilise FT and inverse Fourier Transform (IFT) algorithms. SAR processing becomes formidable and arduous without the use of FT and IFT as pulse compression technique in SAR processing can be summarised as time domain convolution with respective reference signal. Therefore, the operations are often performed in frequency domain in order to reduce the complexity and computational load. The transformation from time domain to frequency domain is achieved by FT whereas the transformation from frequency domain back into time domain is achieved by IFT. FT and IFT play an important role in SAR processing, and hence, it is motivated to extensively study the FT algorithm for the hardware implementation.

Discrete Fourier Transform and its inverse (IDFT) are considered at the first place as the SAR digital processing algorithms are implemented in the finite length signal manner. DFT and IDFT are defined for periodic samples signals or finite length signal. For a discrete-time signal $x[n]$ with the finite length of N, the DFT and IDFT pair are given as

$$\text{DFT} \rightarrow X\left[k\right] = \sum_{n=0}^{N-1} x\left[n\right] W_N^{kn} \quad k = 0, 1, ..., N-1 \tag{1}$$

$$\text{IDFT} \rightarrow x\left[n\right] = \frac{1}{N} \sum_{n=0}^{N-1} X\left[k\right] W_N^{-kn} \quad n = 0, 1, ..., N-1 \tag{2}$$

where N in $X[k]$ is termed as spectral coefficients, and $W_N = e^{-j\frac{2\pi}{N}}$ is the trigonometric constant called twiddle factor with symmetry and periodicity properties due to the complex exponential function. In order to recover $x[n]$ into time domain with correct amplitude, the scaling factor of $\frac{1}{N}$ as in Equation (2) is required. In the time domain signal of $x[n]$, the first point is denoted as $x[0]$ which is the time zero of the signal. The signal samples are evenly spaced by an interval of $\frac{1}{f_s}$ where f_s is the sampling frequency of the ADC. In the frequency domain spectrum of $X[k]$, similar to the time domain signal, the first point is denoted as $X[0]$ which is the zero frequency of the spectrum. The spectrum samples are evenly spaced by an interval of $\frac{f_s}{N}$. Hence, the spectrum sample for $X[k]$ is corresponding to the frequency of $k\frac{f_s}{N}$.

One of the deficiencies arises from DFT and its inverse is the high number of operations required in order to complete the transformation. A forward DFT requires an order of N^2 complex arithmetic operations (computational complexity $O(N^2)$), and the same number of operations happen in its inverse. In SAR processing algorithm, it requires a continuous computation of the FT for the range lines and stores in the memory, thus, an efficient DFT is required in order to achieve this objective. If the signal length N is factored into many small sections with the power of d, the number of operations and complexity can be enormously reduced especially for long data sets in SAR data that can go up to thousands or even millions. This method is called FFT, and its inverse is called IFFT. FFT and IFFT algorithms are efficient methods for the implementation of DFT and IDFT, respectively. FFT algorithm exploits a structure that is employing a divide-and-conquer paradigm in order to reduce the number of operations in the order of $N \log_d N$ instead of N^2 where N is a power of d.

Generally, there are two basic classes of FFT algorithms which are the Cooley-Tukey FFT, Decimation-in-Time (DIT) algorithm and Sande-Tukey FFT, Decimation-in-Frequency (DIF) algorithm [30]. The two algorithms have the same complexity and require the same number of operations

in order to transform a signal into frequency domain. For a radix-2 FFT or its inverse, N is a power of 2, and the total number of operations required is in the order of $N \log_2 N$. More specifically, the number of complex multiplications in a radix-2 FFT is $\frac{N}{2} \log_2 N$, and the number of complex additions is $N \log_2 N$. A single complex multiplication consists of four real multiplications and two real addition whereas a single complex addition consists of two real additions. By considering the real addition or a real multiplication as one operation, a radix-2 FFT or IFFT only requires total $5N \log_2 N$ real operations.

The radix-2 DIF FFT is derived based on Equation (1) by decomposing the output frequency series into successively smaller subsequences, an even-indexed frequency samples set $X[2r]$, and an odd-indexed frequency samples set $X[2r+1]$.

$$X[k] = X[2r] + X[2r+1] \tag{3}$$

An N-point DFT is interlaced and decomposed into two $\frac{N}{2}$ point recursively. This decomposition is recursively applied in order to achieve the shorter-length of DFTs and results in the full radix-2 FFT. The decomposition is stopped when a 2-point DFT is obtained. The 2-point DFT is called a butterfly network. The even-indexed frequency samples set is written as

$$\begin{aligned} X[2r] &= \sum_{n=0}^{N-1} x[n] W_N^{n(2r)} = \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{2nr} + \sum_{n=0}^{\frac{N}{2}-1} x\left(n+\frac{N}{2}\right) W_N^{2r\left(n+\frac{N}{2}\right)} \\ &= \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) + x\left(n+\frac{N}{2}\right)\right] W_{\frac{N}{2}}^{nr} \end{aligned} \tag{4}$$

where $W_N^{2r(n+\frac{N}{2})} = W_N^{2nr} W_N^{rN} = W_N^{2nr} = W_{\frac{N}{2}}^{nr}$. The simplification of the twiddle factor is mainly due to the 2π periodicity and symmetry properties of the complex exponential function. Similarly for the odd-indexed frequency samples set, it is denoted as

$$\begin{aligned} X[2r+1] &= \sum_{n=0}^{N-1} x[n] W_N^{n(2r+1)} = \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{(2r+1)n} + \sum_{n=0}^{\frac{N}{2}-1} x\left(n+\frac{N}{2}\right) W_N^{(2r+1)\left(n+\frac{N}{2}\right)} \\ &= \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) - x\left(n+\frac{N}{2}\right)\right] W_N^n W_{\frac{N}{2}}^{nr} \end{aligned} \tag{5}$$

Hence, the major operations in the implementation of the FFT are done during the subdivision steps and the complex operations of subdivided samples with the twiddle factor. A radix-2 N-point FFT requires M stages for the completion of the transformation where $N = 2^M$. The butterfly network is illustrated by the annotated butterfly symbol as shown in Figure 3.

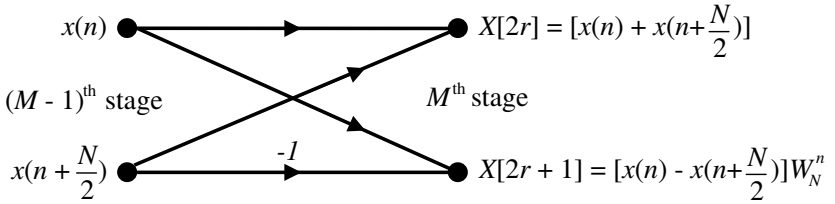

Figure 3. Butterfly network of DIF FFT.

A simple radix-2 8-point FFT is shown in Figure 4. Addition operations are represented by the arriving arrows while the constant coefficient multiplications are represented by a factor at an arrow. The same type of twiddle factor is used in the multiplications for each group.

An IFFT is required in order to transform the frequency domain spectrum, $X[k]$, back to time domain signal, $x[n]$. The IFFT algorithm can be easily implemented by modifying from FFT algorithm

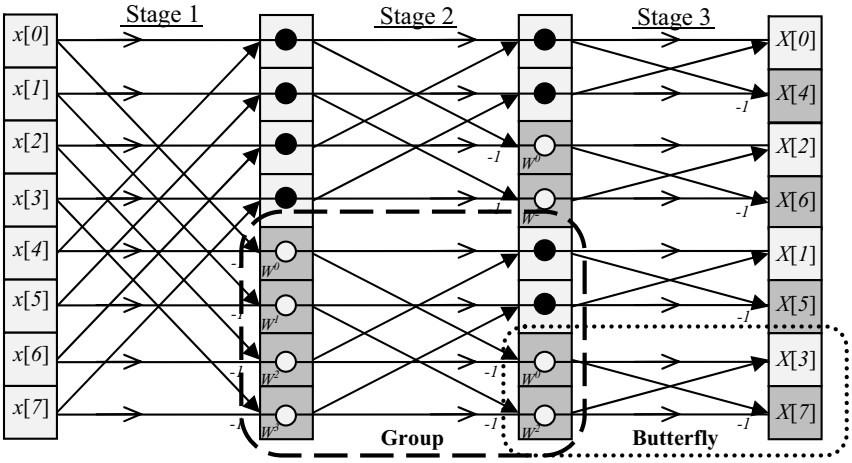

Figure 4. Signal flow graph of radix-2 8-points FFT.

as in Equation (1) with the use of complex conjugation method. The complex conjugate of both sides as in Equation (4.3.2) is calculated.

$$x*[n] = \frac{1}{N}\left[\sum_{n=0}^{N-1} X[k]\, W_N^{-kn}\right]^* \tag{6}$$

Since the complex conjugate of a product is equivalent to the product of the conjugates, Equation (6) can be further expanded as

$$x*[n] = \frac{1}{N}\sum_{n=0}^{N-1} X[k]^* \left(W_N^{-kn}\right)^* = \frac{1}{N}\sum_{n=0}^{N-1} X[k]^*\, W_N^{kn} \tag{7}$$

The similarity of Equation (7) to the forward FFT as in Equation (1) is surfaced. By taking the conjugate of both sides in Equation (7), the time domain signal $x(n)$ can be found. In short, IFFT is derived according to forward FFT with the following steps. The complex conjugate of $X[k]$ is firstly obtained as $X*[k]$. A forward FFT is performed on $X*[k]$ with a scaling factor of $1/N$ in order to obtain $x*[n]$. Finally, $x[n]$ which is the time sample signal is found by taking the complex conjugate of $x*[n]$.

The main impact that maked FFT and IFFT relatively important in SAR processing is the unique property of convolution theorem. The convolution in one domain is equivalent to the multiplication in the other domain.

$$x_1(t) \otimes x_2(t) \leftrightarrow X_1(f)\, X_1(f) \tag{8}$$

$$x_1(t)\, x_2(t) \leftrightarrow X_1(f) \otimes X_1(f) \tag{9}$$

In time domain, a convolution of two periodic sequences is equivalent to multiplication operation as in frequency domain and vice versa. This property significantly benefits SAR processing, and it is the most important property being utilised as match filtering technique used in SAR processing.

Based on the study on the FFT algorithm and its inverse, the proposed architecture for hardware implementation of a radix-2 N-points FFT processor will be presented and discussed. SAR processing operations require huge number of matched filtering operations in order to process useful and meaningful SAR data. The operations can be performed much faster and efficient in the frequency domain. Hence, an efficient FFT hardware architecture will always guarantee the performance of SAR processor.

2.2. FFT Coprocessor Architecture

The working principle and operation process of the radix-2 FFT algorithm are in discussed in Section 2.1. Based on the discussed FFT algorithm, the proposed hardware architecture of a configurable FPGA-based N-point FFT is illustrated in the block diagram of Figure 5. The generic module is designed

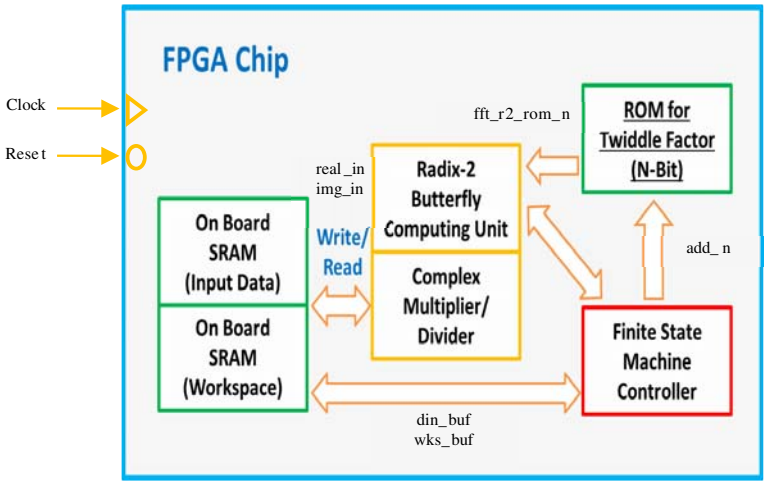

Figure 5. Architecture of the proposed configurable N-point FFT processor.

based on the radix-2 decimation in frequency algorithm of FFT with the in place computation for optimisation of memory resources. The number of points for the FFT and data width of all modules are configurable, and it can be configured into any length of N in the power of 2 according to the requirement of application. The data width of the processor in this research work is set to fixed-point format of 16 bits as the SAR raw data width is 16-bits.

The design and development of the generic FFT processor is carried out by using Verilog HDL approach and synthesised into FPGA hardware with Altera Quartus II CAD tool. Overall, the architecture is partitioned into several main modules, which are look-up-table (LUT) ROMs, FFT finite state machine (FSM) controller, radix-2 butterfly computing unit that consist of complex multiplier, on-board SRAMs, and lastly the memory handling unit. The twiddle factors for the FFT are pre-computed and stored into the internal ROMs of the FPGA. The on-board SRAMs are served as input buffer and workplace buffer memories. They are used to temporary store the SAR raw data and fed into the FFT core to perform the transform. The intermediate and final results of the transform will be stored into the workplace buffer. Based on the interrupt control signal to the FFT module, the FSM controller will trigger the butterfly computing unit to start the transform operation; meanwhile, the internal memory handling unit will generate the required memory addresses so that the transformed data will be placed into the workplace SRAM with an appropriate address.

The target FPGA device that is used to synthesise and implement the FFT coprocessor is Altera Stratix IV GX FPGA EP4SGX230KF40C2 device. This target FPGA is selected as it is a part of the Altera DE4 Development Board. A 16-bits N-FFT based on hardware resources available in the device Stratix IV GX FPGA EP4SGX230KF40C2 is synthesised by using Quartus II CAD tool. The architecture schematics of the N-point FFT processor with transform length from 64 to 4096 are available in Appendix E. Figure 6 shows the whole schematic representation of the internal structure of the proposed FFT coprocessor architecture with transform length of 1024. Label 1 in the figure indicates the FSM controller that controls the overall operation of the FFT coprocessor while label 2 is the radix-2 butterfly computing unit that internally contains a complex multiplier with several real adders. The section labelled with label 3 is a 32 bits ROM that store the twiddle factor coefficients. Label 4 and label 5 are the parameterised true single-port synchronous RAMs that act as an input buffer and a workplace buffer, respectively.

2.3. Finite State Machine (FSM) Controller

The zoomed in version of the FSM controller in Figure 6 is illustrated in Figure 7. In the FSM controller, total 5 inputs are required in order to produce the three unique state outputs for the FFT transformation. The three FSM states are idle state (FSM_ST_IDLE), read state (FSM_ST_RD_XA),

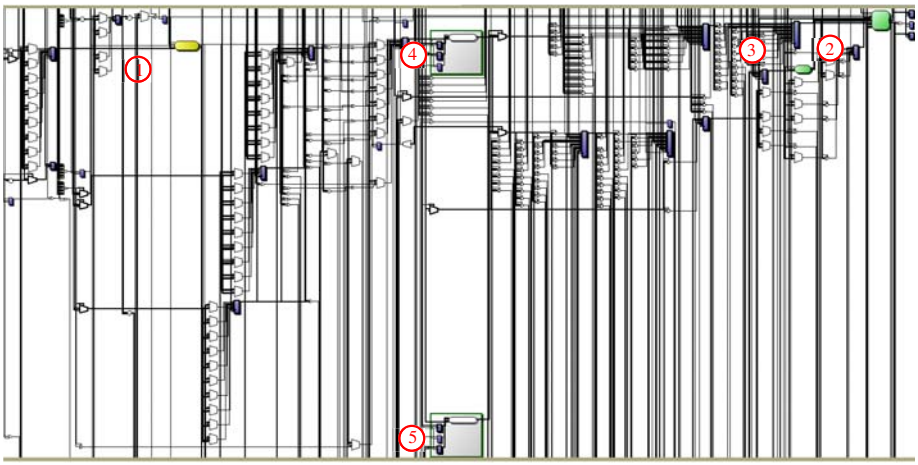

Figure 6. Synthesised of RTL 16-bits 1024-FFT in Stratix IV GX.

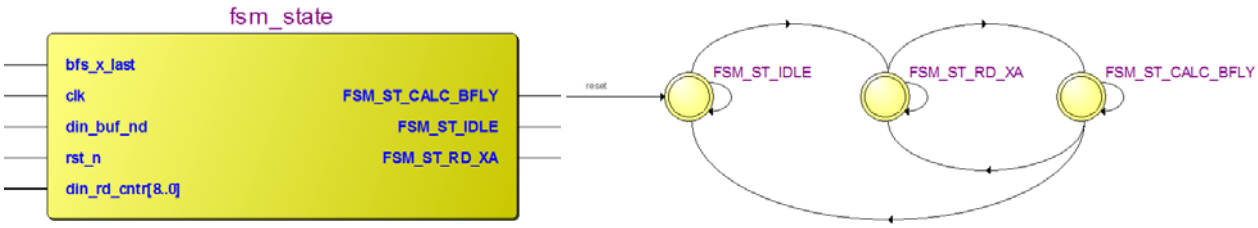

Figure 7. Synthesised of RTL FSM controller for FFT coprocessor and the finite states.

and butterfly calculation state (FSM_ST_CALC_BFLY). The FSM controller will generate necessary trigger signal to push the data from buffer into the butterfly unit. The clk pin is the input clock signal to drive the FSM, and the rst_n is the active low reset pin. The din_buf_nd pin is the trigger valid signal to the FSM to start a new FFT operation when the last sample of a transform period has been completed. Lastly, bfs_x_last and din_rd_cntr[8..0] are the control signal and counter output of internal memory handling unit that assign appropriate addresses to the buffer in order to read the data for the transformation.

2.4. Butterfly Computing Unit

The zoomed in version of the butterfly computing unit as in Figure 6 is shown in Figure 8. This butterfly computing unit module computes a single 2-points DFT operation including the twiddle-factor complex multiplication. The clk and rst_n pins are the input clock signal and active low reset pin, respectively. Pin x_nd is used to trigger the butterfly computing unit and indicates that there is valid new data on the input data ports. There are total six input data ports which are w_re, w_im, xa_re, xa_im, xb_re, and xb_im. The ports w_re and w_im are the real part and imaginary part of the twiddle factor coefficient respectively; meanwhile, xa_re, xb_re and xa_im, xb_im are the real part and the imaginary part of the input data, respectively.

The operation of the butterfly computing unit is divided into several stages in order to produce the output z_re and z_im which are the real and imaginary parts of the output data from the butterfly computing unit. The operation is started with the loading of input data and waiting for the multiplier to perform the complex multiplication. The complex multiplication is divided into real part and imaginary part, and hence the total two complex multipliers are required in the module. Lastly, the final output of the operations is scaled down to the data width as in the input data width, and the outputs are updated. The complete internal architecture of the butterfly computing unit module is portrayed in Figure 9. In the figure, the complex multipliers are labelled as CM symbols.

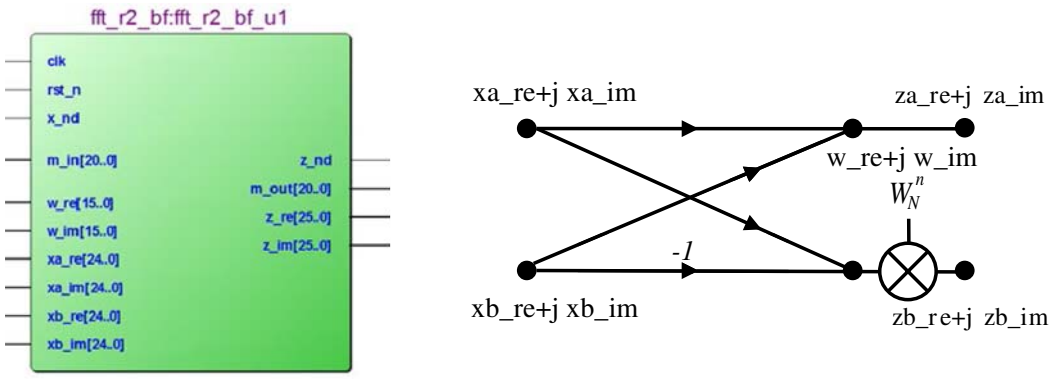

Figure 8. Synthesised of RTL butterfly computing unit for FFT coprocessor.

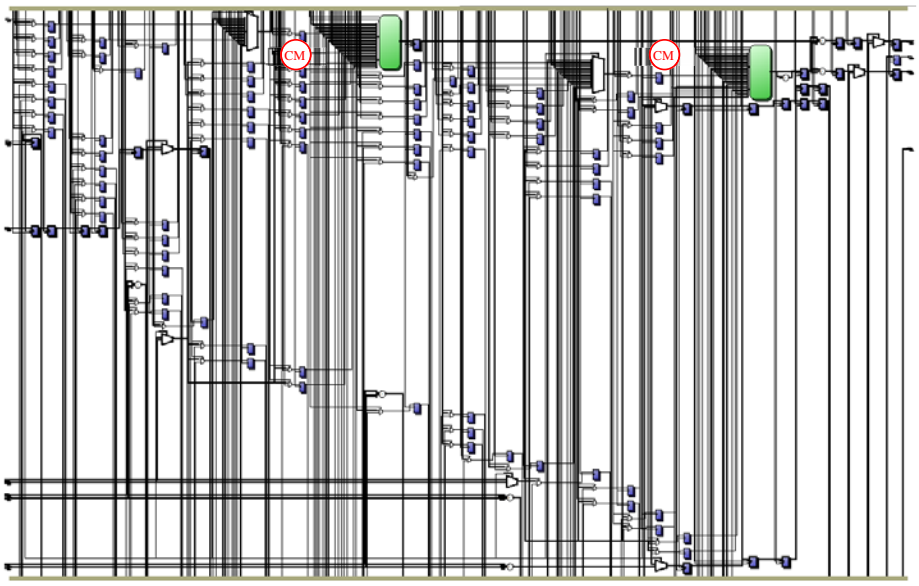

Figure 9. Synthesised of RTL butterfly computing unit internal architecture.

2.5. Complex Multiplier

The highlighted portions with label CM in Figure 9 are the two complex multipliers used in the butterfly computing unit module. It is of paramount importance to explore the internal structure of the butterfly computing unit as it is the core of an FFT processor. By referring to Figure 10, the architecture of butterfly computing unit is made up of a complex adder, complex subtraction, and a complex multiplier for the twiddle factor multiplications.

Equation (10) shows that a complex adder is implemented by two real adders that sum the real part and imaginary part of the signals accordingly. Equation (11) shows that a twiddle factor is multiplied towards the operation of x-y in order to produce the output Y. It is noticeable that a complex multiplication is much more complicated than real multiplication because it involves several real multiplications and additions in order to complete the operation. Equation (12) is the simplified version of twiddle factor which will be used to explain the operation of complex multiplier inside the butterfly computing unit.

$$X = (x_{\text{real}} + y_{\text{real}}) + i\,(x_{\text{imag}} + y_{\text{imag}}) \tag{10}$$

$$Y = [(x_{\text{real}} - y_{\text{real}}) + i\,(x_{\text{imag}} - y_{\text{imag}})]\,W_N \tag{11}$$

$$W_N = e^{-j\frac{2\pi}{N}} = C + jS \tag{12}$$

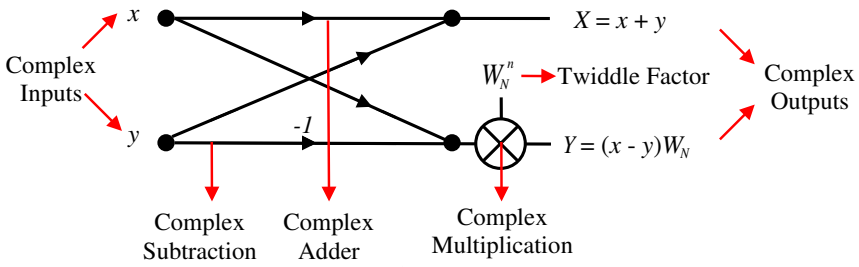

Figure 10. DIF FFT butterfly computing unit (2 points DFT operation).

Assuming an arbitrary complex input variable with A as the real part and B as the imaginary part multiplied with the twiddle factor as in Equation (12), the complex multiplication is shown as

$$(A + jB) \times (C + jS) = (AC - BS) + j\,(AS + BC) \tag{13}$$

Equation (13) shows the complex multiplier for the twiddle factor complex multiplication which is implemented by using 4 real multiplications (AC, BS, AS, BC) and 2 add/subtract operations ($AC-BS$ and $AS+BC$). Figure 11 shows the RTL of the synthesised complex multiplier. Two real multiplications and one adder are required on each of the real part and imaginary part in the operation. Therefore, two of the complex multipliers are instantiated in order to complete one complex multiplication, one for the real part and the other for the imaginary part. The clk and rst_n pins are the input clock signal and active low reset pin, respectively. Pin xy_nd is used to trigger the complex multiplier and indicates that there is valid new data available on the input data ports. Port x is the input to the multiplier while port y is the multiplicand. The result of the multiplication is valid after 2 clock cycles and output as port z. Pin z_nd indicates that the multiplier has a valid output data from the operation.

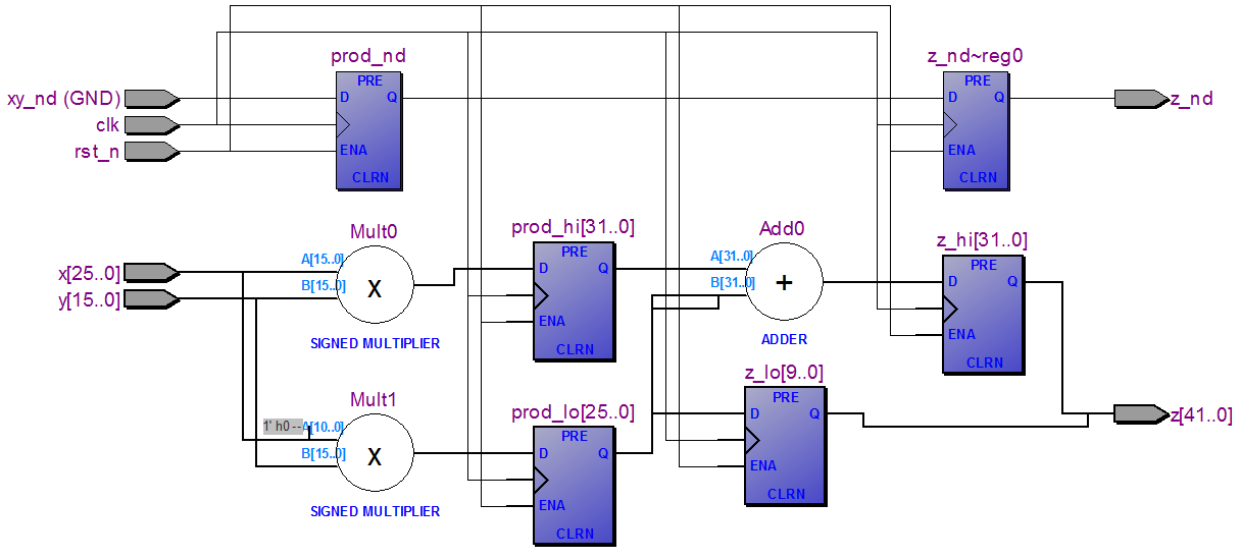

Figure 11. Synthesised RTL complex multiplier for FFT coprocessor.

In the complex multiplier, the multiplication of multiplicand and multiplier is carried out though 2-stages pipelined operations. The multiplicand is segmented into two parts which is the 10-bits of LSB and 15-bits of LSB before the multiplication. These two parts of multiplicand are then multiplied with the same multiplier concurrently, and the products of these two operations are latched by the flip-flop prod_lo and prod_hi. The number of bits for the intermediate product registers is subjected to the data width of the multiplication which is the summation of data width for multiplicand and multiplier.

After the intermediate products of the multiplication are obtained, the data width of prod_lo is then extended into 32 bits so that these products can be added or subtracted in the same data length in order to produce the z_hi as stage-1 product of the multiplication. The subtraction is carried out in the same manner as addition by using 2's complement numbering format. As for the stage-0 product of the multiplication, it is abstracted from the 10-bits of LSB from prod_lo and latched as z_lo. Lastly, the z_lo and z_high are combined through concatenate syntax in Verilog HDL in order to produce the final product of the multiplication. This entire operation will be carried out twice simultaneously by two dedicated multipliers, one for real data and the other for the imaginary data.

2.6. Twiddle Factor Coefficient ROM

The twiddle factor of the FFT operation as in Figure 10 is implemented by using a synchronous RAM in the FPGA device as the default memory resource available in an FPGA is RAM. Thus, RAM is treated as ROM and loaded with the precalculated twiddle factor that is stored in memory initialization file (.mif) to create a LUT ROM. Throughout the FFT operation, the twiddle factor coefficients are being read only, and hence the synchronous RAM is behaving as a ROM without any write operation. The data width of the twiddle factor ROM depends on the input data of the application. Since the input data width of the FFT coprocessor is set to 16 bits, the twiddle factor ROM with 32 bits data width is instantiated and synthesised as shown in Figure 12 and Figure 13. The twiddle factor is precomputed and generated by using MATLAB with the first 16 bits MSB (Bit 31 to Bit 16) as the real part of the twiddle factor coefficients and the first 16 bits LSB (Bit 15 to Bit 0) as the imaginary part of the twiddle factor coefficients. The address line width is set to 9 bits as there are 512 unique twiddle factor coefficients for 1024-points of FFT. Different numbers of points for the FFT processor will have their own unique twiddle factor ROM respectively, but generally the architectures are the same.

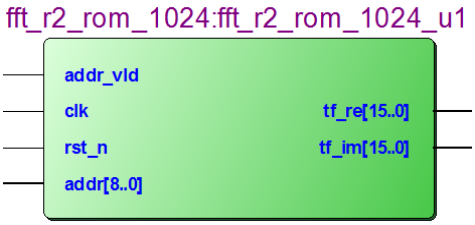

Figure 12. Synthesised RTL 32 bits twiddle factor ROM for FFT coprocessor.

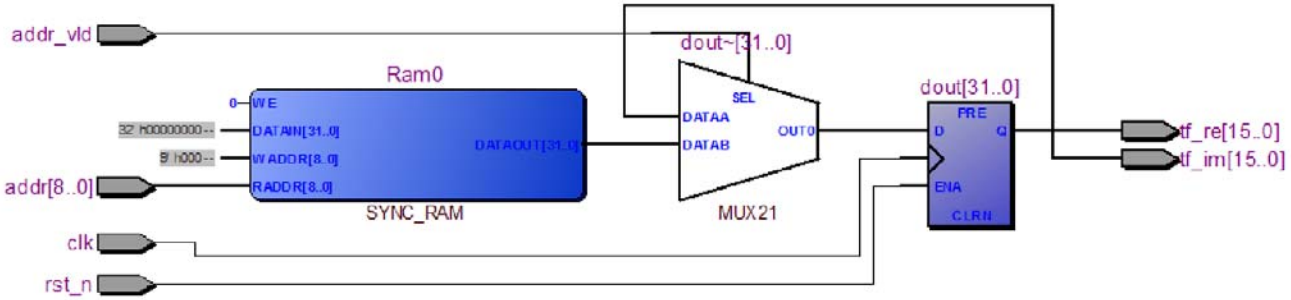

Figure 13. Internal architecture of synthesised RTL 32 bits twiddle factor ROM for FFT coprocessor.

As observed in Figure 13, the internal of the twiddle factor ROM consists of a synchronous RAM, a data multiplexer, and a D flip-flop. The multiplexer functions as the imaginary twiddle factor selection during the read operation according to the address validity while the D-flip flop will latch the real part of the twiddle factor from the ROM accordingly with the synchronisation of global clock signal.

2.7. Input and Workplace Buffers Synchronous RAMs

Label 4 and label 5 as shown in Figure 6 are the parameterised single-port synchronous RAMs that act as input buffer and workplace buffer during the FFT operation. The two RAMs have the same architectures and are clocked by a single clocking frequency. The RAMs support non-simultaneous write and read operations from a single address. It allows read or write operation to be performed at a time only. The only difference between these two RAMs is the data width which is defined by the input data width. Figure 14 and Figure 15 show the architectures of the mentioned RAMs.

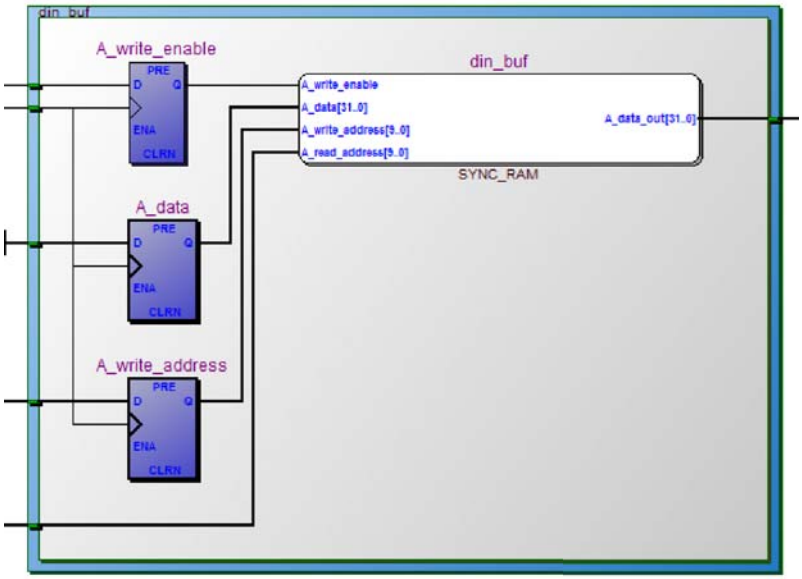

Figure 14. Architecture of synthesised RTL synchronous RAM input buffer for FFT coprocessor.

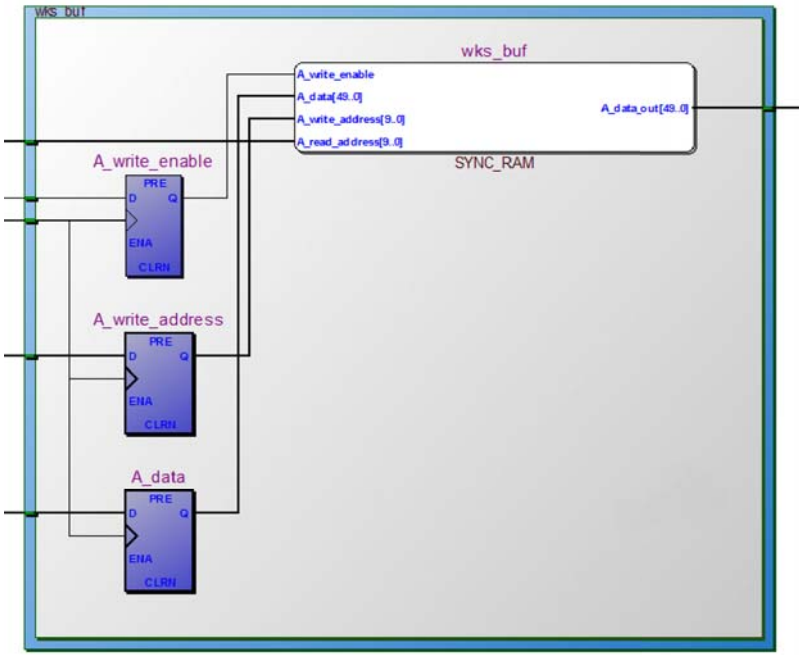

Figure 15. Architecture of synthesised RTL synchronous RAM workplace buffer for FFT coprocessor.

Since the FFT coprocessor is designed based on 16 bits data width, the input buffer synchronous RAM is 32 bits wide. The data is indexed into real part and imaginary part and write into the RAM before the FFT operation is carried out. The first 16 bits MSB data in the RAM (Bit 31 to Bit 16) is the real part of the input data while the first 16 bits LSB data (Bit 15 to Bit 0) are the imaginary part of the input data. The address line width is set to 10 bits as the sample size for the FFT processor is 1024 (2 to the power of 10). In other words, the address line width depends on the number of points of the FFT processor that is in the factor of power of 2.

As for the workplace buffer RAM, the data width is set to 50 bits based on 16 bits input data width. Similar to the input buffer, for workplace buffer, the first 25 bits MSB data in the RAM (Bit 49 to Bit 25) is the real part of the data while the first 25 bits LSB data (Bit 24 to Bit 0) is the imaginary part of the data. The reason for workplace buffer having larger data width than the input buffer is mainly that operations are carried out during the FFT operation. In the butterfly computing unit, it involves several operations such as additions and multiplications that will increase the data width. Therefore, workplace buffer requires larger data width in order to store the intermediate results of the sub operations in order to prevent overflow or data saturation. The output data from the FFT operation will be scaled, updated, and latched by two D flip-flops for both real and imaginary parts in order to keep the data width same as the input data width. The address line width for the workplace buffer RAM is similar to the input buffer RAM as discussed earlier. The performance evaluation including the hardware resources usage and timing considering of the entire FFT coprocessor will be discussed in the later section.

2.8. Design and Implementation of IFFT Coprocessor

The working principle and operation process of the radix-2 IFFT algorithm are discussed in Section 2.1. The IFFT coprocessor has a very similar architecture as discussed in Section 2.2. This is because a minor modification is done by manipulating the data in the buffer RAM and computing the IFFT by using an FFT coprocessor.

As discussed in Section 2.1, the IFFT is performed by using complex conjugate operation and FFT operation with scaling factor of N where N is the transform length. Since the data are stored in the buffer synchronous RAM with real and imaginary parts indexed with the respectively unique addresses, implementing the complex conjugate operation can be easily done by inversing the signed bit of the imaginary data in the RAM for the indexed addresses. The IFFT coprocessor implementation scheme is illustrated in Figure 16. In short, an IFFT coprocessor is composed of an FFT coprocessor as discussed earlier, a data manipulator to inverse the sign bit of the imaginary data which imitate the complex conjugate operation. Lastly, two shift registers will be used in order to scale the data with the factor of N, where N is required transform length, so that frequency domain data can be transformed into time domain data.

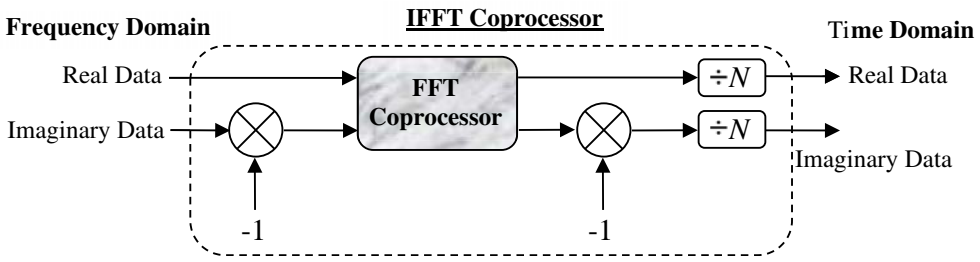

Figure 16. Architecture of IFFT coprocessor using modified FFT coprocessor.

The performance analysis and evaluation of the proposed FFT coprocessor will be carried out in the upcoming section. The proposed processors will be benchmarked according to the timing performance consideration and also overall hardware usage resources occupied in Altera Stratix IV GX FPGA EP4SGX230KF40C2 chip.

3. RESULTS AND DISCUSSIONS

In order to analyse and evaluate the performance of the proposed N-point radix-2 FFT coprocessor, the proposed coprocessor is benchmarked and compared with commercialised Altera Intellectual Property (IP) FFT MegaCore. Both proposed coprocessor and IP core are configurable in terms of transform length and data width. The benchmarks are carried out by implementing both proposed processor and IP core in the same device which is Altera Stratix IV GX FPGA EP4SGX230KF40C2 chip with 16 bit data width in fixed-point format from the transform length of 64 points to 4096 points. The performance evaluation metrics of both processors are conducted in terms of timing considerations, hardware resources utilisations, and total power consumptions.

3.1. Hardware Resources Utilisations and Power Consumptions

The proposed FFT coprocessor and Altera FFT IP MegaCore are instantiated in Altera Quartus II CAD tools in order to analyse and synthese the physical hardware resources utilisations and total power consumptions for both FFT coprocessors in the mentioned Stratix IV device. The total power consumptions of the coprocessors are measured by using the CAD tool built-in function called PowerPlay Power Analyser. The FFT coprocessors based on the transform lengths of 64, 128, 256, 512, 1024, 2048, and 4096 are instantiated and synthesised for the performance evaluation. The main reason that these

Table 1. Description of hardware resources utilisations metrics.

Hardware Resources	Description
Combinational ALUT	The ALUT is a construction of logical that is representing the design which is being implemented by the combinational logic hardware of an Adaptive Logic Module (ALM) in the supported device families (Stratix IV). The Adaptive Logic Module (ALM) is the fundamental building block of the device Stratix IV. A single ALM can support up to 8 inputs and 8 outputs that contains two or four register logic cells and two combinational logic cells, two dedicated full adders, a carry chain, a register chain, and a 64-bit LUT mask.
Total Dedicated Logic Registers	Represents the total number of logic registers in the design that will be implemented with core logic of ALMs. They are are also used as the implementation registers and routing optimisation registers.
Total Pin	The total number of pins being occupied in the design.
Total Block Memory Bits	Total number of memory bits that are used in the design.
M9K Block Memory	Asynchronous true dual-port memory block with registered inputs and registered outputs that is only available in selected device families (Arria II GX, Cyclone IV, and Stratix IV). The M9K block is useful for storing processor code and implementing lookup schemes such as the twiddle factor LUT in FFT processor. Each of the M9K block memory is a 256×36 RAM block and contains 9,216 programmable bits.
DSP Block 18-bit Elements	The DSP block 18-bit elements represents the total number of DSP block 18-bit elements utilised in the design. These DSP blocks contain input shift registers and can be easily used to implement multiply, multiply-add and multiply-accumulate features.

transform lengths are selected is mainly that the data sample size of SAR application typically falls under this range. Thus, in total 7 proposed FFT coprocessors and 7 Altera IP FFT cores are synthesised for the performance comparison purpose.

There are six metrics to be considered in the area of hardware resources utilisations. These metrics are Combinational Adaptive Look-Up Table (ALUT), total dedicated logic registers, total pins, total block memory bits, M9K block memory, and DSP block 18-bit elements. The descriptions of these performance metrics are summarised and tabulated in Table 1.

The performance evaluations are conducted by comparing the hardware resources utilisations in six metrics as discussed earlier and the total power consumptions for both Altera IP FFT coprocessor and the proposed FFT coprocessor with the transform length from 64 points until 4096 points. The evaluations are tabulated into Table 2 and Table 3 for the ease of comparison.

Table 2. Evaluation of hardware resources utilisations and power consumption of proposed system and Altera IP for different point of FFT (64-, 128-, 256-, 526-, 1024-, 2048-, and 4096-point).

Area of Hardware Resources	Combinational ALUTs	Total Dedicated Logic Registers	Total Pins	Total Block Memory bits	M9K Block Memory	DSP block 18-bit elements	Total Power Consumption
Altera IP 64-FFT	1,992	3,461	85	9,024	18	12	886.48 mW
Proposed 64-FFT	439	884	68	4,736	3	8	901.06 mW
Altera IP 128-FFT	2009	3,585	85	17,792	18	12	886.90 mW
Proposed 128-FFT	464	934	68	9,728	3	8	901.30 mW
Altera IP 256-FFT	2,078	3,698	85	39,168	20	12	887.99 mW
Proposed 256-FFT	543	984	68	19,968	3	8	901.52 mW
Altera IP 512-FFT	2,421	4,101	85	78,080	20	12	888.49 mW
Proposed 512-FFT	634	1,035	68	40,960	5	8	901.73 mW
Altera IP 1024-FFT	2,491	4,418	85	155,904	20	12	888.90 mW
Proposed 1024-FFT	803	1,085	68	83,968	10	8	901.99 mW
Altera IP 2048-FFT	3,417	5,631	85	311,552	39	12	889.27 mW
Proposed 2048-FFT	1,088	1,135	68	172,032	24	8	903.55 mW
Altera IP 4096-FFT	3,597	5,949	85	622,848	76	24	890.74 mW
Proposed 4096-FFT	1,602	1,185	68	352,256	41	8	905.82 mW

From the tables, it is noticeable that for all transform lengths of the FFT coprocessors, the proposed FFT coprocessors occupy fewer hardware resources in all the six metrics than the Altera IP FFT coprocessor. For the total number of pins metrics, the total number of pins occupied for all transform

Table 3. Resources reduction of proposed system (Different point of FFT, 64-, 128-, 256-, 526-, 1024-, 2048-, and 4096-point) reference to Altera IP.

Area of Hardware Resources	Resources Reduction						
	Altera IP 64-FFT vs Proposed 64-FFT	Altera IP 128-FFT vs Proposed 128-FFT	Altera IP 256-FFT vs Proposed 256-FFT	Altera IP 512-FFT vs Proposed 512-FFT	Altera IP 1024-FFT vs Proposed 1024-FFT	Altera IP 2048-FFT vs Proposed 2048-FFT	Altera IP 4096-FFT vs Proposed 4096-FFT
Combinational ALUTs	77.96%	76.90%	73.867%	73.81%	67.76%	68.16%	55.46%
Total Dedicated Logic Registers	74.46%	73.95%	73.39%	74.76%	75.44%	79.84%	80.08%
Total Pins	20.00%	20.00%	20.00%	20.00%	20.00%	20.00%	20.00%
Total Block Memory bits	47.52%	45.32%	49.02%	47.54%	46.14%	44.78%	43.44%
M9K Block Memory	83.33%	83.33%	85%	75.00%	50.00%	38.46%	36.84%
DSP block 18-bit elements	33.33%	33.33%	33.33%	33.33%	33.33%	33.33%	66.67%

lengths remain the same which is 85 pins for Altera IP FFT and 68 pins for the proposed FFT. The main reason for the Altera IP FFT occupying more pins is that the Avalon-ST protocol interface used for data transfers internally. This protocol is not used in the proposed FFT because of the hardware usage reduction purpose. Hence, this shows that the proposed FFT occupies fewer pins with 20% reduction as compared to the Altera IP FFT.

Besides, from the tables it is also apparent that when the transform length increases, the utilised hardware resources are drastically increased as well. However, the proposed FFT coprocessor shows the promising hardware resources reduction regardless the length of the transform as compared to the Altera IP FFT coprocessor. Another reason for the Altera IP FFT occupying more hardware resources than the proposed FFT is mainly the existence of internal error handler used to monitor the illegal usage of the Avalon-ST protocol. Since the proposed FFT coprocessors do not use this protocol, the error handler is not needed which leads to the reduction of hardware resources as shown in tables and figures. The reduction of hardware usage significantly helps in the development of the integrated SAR processor as more hardware resources quota can be allocated for the other sub-module designs in the SAR processor.

In terms of the total power consumption, the proposed FFT coprocessor consumes slightly higher power than the Altera IP FFT coprocessor in all the transform lengths. The differences of the power consumptions of all transform lengths are about 14 mW. However, the small difference of the power consumptions between the proposed FFT coprocessors and the Altera IP FFT coprocessor is negligible. The proposed FFT coprocessors have slightly higher power consumptions mainly due to the employment of global routing architectures done by the CAD tool. This is because for the global signal networks such as global clock that spans larger areas of the device and have high capacitance will significantly increase the power consumption of the design. Besides, the operating mode of each circuit element will also affect the power consumption. For instance, a DSP block element performing $n \times n$ multiplications and a DSP block element performing multiply-accumulate operations will consume different amounts of power due to the different amounts of charging internal capacitance during each transition. All of these factors affecting the power consumptions of the design are subjected to the design architecture in the Verilog HDL modeling and the CAD tool compiler settings.

3.2. Timing Performance Analysis

A series of test benches are carried out by using Mentor Graphic ModelSimin order to exercise the Altera FFT coprocessor and the proposed FFT coprocessor. The test benches allow the performance evaluation of the processors in terms of timing and functionality verification to be conducted. The coprocessors are instantiated and treated as the device under test (DUT) followed by inserting the required stimulus data into the DUT. The overall block diagram of the test bench is illustrated in Figure 17. The inserted stimulus will trigger and operate the DUT in order to produce output responses that consist of the transformed data, and the timing of the transform can also be observed.

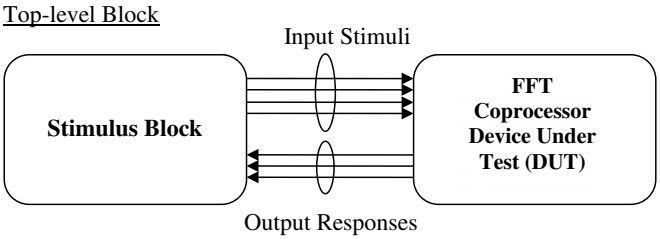

Figure 17. Test bench for performance evaluation of the FFT coprocessors.

The proposed FFT coprocessor and Altera FFT IP MegaCore undergone the test benches process in order to analyse and evaluate the timing performance in the mentioned Stratix IV device. The FFT coprocessors based on the transform lengths of 64, 128, 256, 512, 1024, 2048, and 4096 are instantiated as DUTs for the evaluation purpose. Thus, there are in total 7 proposed FFT coprocessors and 7 Altera IP FFT cores undergoing the test benches for the timing performance comparison purpose by using the same set of stimuli. There are 2 metrics to be considered in the timing performance metrics which are maximum restricted clocking frequency of the processors and the total transform time required to complete a single transform with the specified length. The maximum restricted clocking frequency depends on two factors which are the speed grade of the device family (Stratix IV) to be implemented and also depends on the design architecture done by user.

Table 4 tabulates the test benches results for both Altera IP FFT and the proposed FFT coprocessors with the transform length from 64 points to 4096 points. The test benches are conducted by clocking the coprocessors with a clock frequency of 340 MHz. The main reason for selecting this frequency is mainly the maximum restricted frequencies for both processors falling in the same frequency

Table 4. Timing performances comparison.

FFT Transform Length	Maximum Restricted Frequency, F_{max} (MHz)		FFT Transform Time (μs)	
	Altera IP FFT	Proposed FFT	Altera IP FFT	Proposed FFT
64	375.09	341.76	1.69	1.81
128	375.09	341.76	3.08	3.39
256	375.09	341.76	5.35	7.53
512	375.09	341.06	10.82	16.56
1024	375.09	341.06	21.08	36.13
2048	375.09	341.06	34.50	78.27
4096	375.09	340.58	67.80	168.59

range. The maximum clocking frequency is subjected to several factors in FPGA-based design. This clocking frequency depends on the speed grade of the device as well as the design of the coprocessor done by the user. Different design architectures will lead to different maximum clocking frequencies although the same FPGA device is used. This is mainly becaues the clocking frequency is subjected to the timing path, and it is defined by the slowest path in the designed circuit. From Table 4, it can be observed that the maximum restricted frequency for the Altera IP FFT is slightly higher than the proposed FFT about 10% for all transform lengths. This means that the Altera IP FFT coprocessor can operate slightly faster than the proposed FFT coprocessor. This condition explains the reason that Altera IP FFT coprocessor utilises significantly more hardware resources than the proposed FFT coprocessor in Section 3.1 because certain levels of timing optimisations are implemented by scarifying the hardware resources.

In terms of transform time, the Altera IP FFT coprocessor has faster transform time than the proposed FFT coprocessor. The FFT coprocessors transform times are highly competitive for the transform lengths of 64, 128, and 256. The two FFT coprocessors have very similar transform times in terms of microseconds. However, when the transform length is increased starting from 512 points to 4096 points, the Altera IP FFT coprocessors show faster transform time in terms of microseconds than the proposed FFT coprocessor. This is mainly because the optimisation of the Altera IP FFT coprocessors is dominant for speed over the area resources while the proposed FFT coprocessors are dominant for area resources over speed.

In spite of the transform time for Altera IP FFT coprocessors are faster than the proposed FFT coprocessors in the multiple of tens microseconds (the smallest difference is 5.74 µs, and the largest difference is 100.79 µs), and the proposed FFT coprocessor is regarded as satisfactory with respect to the implementation of SAR processor. This is because both FFT coprocessors are capable to perform a single transform within a single PRI, but the proposed FFT coprocessor achieves this transformation with lower hardware resources and lesser complexity. The tradeoff between timing and hardware utilisation is acceptable in this case. Therefore, the proposed FFT coprocessor is still preferable for the implementation of SAR processor due to lower hardware resources utilisation while maintaining reasonable timing performance. In such a way, more hardware resources quota can be allocated for the others sub-module designs in the SAR processor.

3.3. Precision Comparison of the Proposed FFT/IFFT coprocessors with MATLAB FFT/IFFT

After evaluating the proposed FFT coprocessor in terms of hardware resources utilisations and also timing performances, the proposed FFT coprocessor is also compared with MATLAB built-in FFT function in order to evaluate the performance in terms of precision and also verify the functionality. A complex chirp signal with bandwidth of 50 MHz is used as the input signal and loaded into the proposed FFT coprocessor and also MATLAB FFT in order to evaluate the transformed signal. The difference between these two transformed outputs is evaluated accordingly. Since the range and azimuth sample points of the SAR raw data are 1024 samples and 2048 samples, respectively, the evaluation in this section only involves the coprocessor with the transform lengths of 1024 points and 2048 points. This is because the coprocessors with these transform lengths will be used in the RDA for range compression and azimuth compression, respectively. Similarly, the evaluation of the inverse FFT is conducted in the same manner.

The test benches as conducted in Section 3.2 are modified with minor changes so that the FFT coprocessor can read the chirp signal from a binary file, transform it into frequency domain, and write the processed sample into a new binary file. The main purpose of doing this is to allow MATLAB to be able to read the processed data from the proposed FFT coprocessor and visualise the transformed signal so that comparison can be done with the MATLAB built-in FFT. Figure 18 shows the input 16 bits fixed point format complex chirp signal with its real part and imaginary part, respectively. This input signal will be transformed into frequency domain by using the proposed FFT coprocessor and also MATLAB built-in FFT function in order to evaluate the precision of the proposed FFT.

It can be observed that the number of sample points for the chirp signal is 432. The proposed FFT coprocessor and the MATLAB built-in FFT will pad the input signal with trailing zeros to the transform length. The proposed FFT coprocessor achieves this by pre-initialising the input buffer RAM

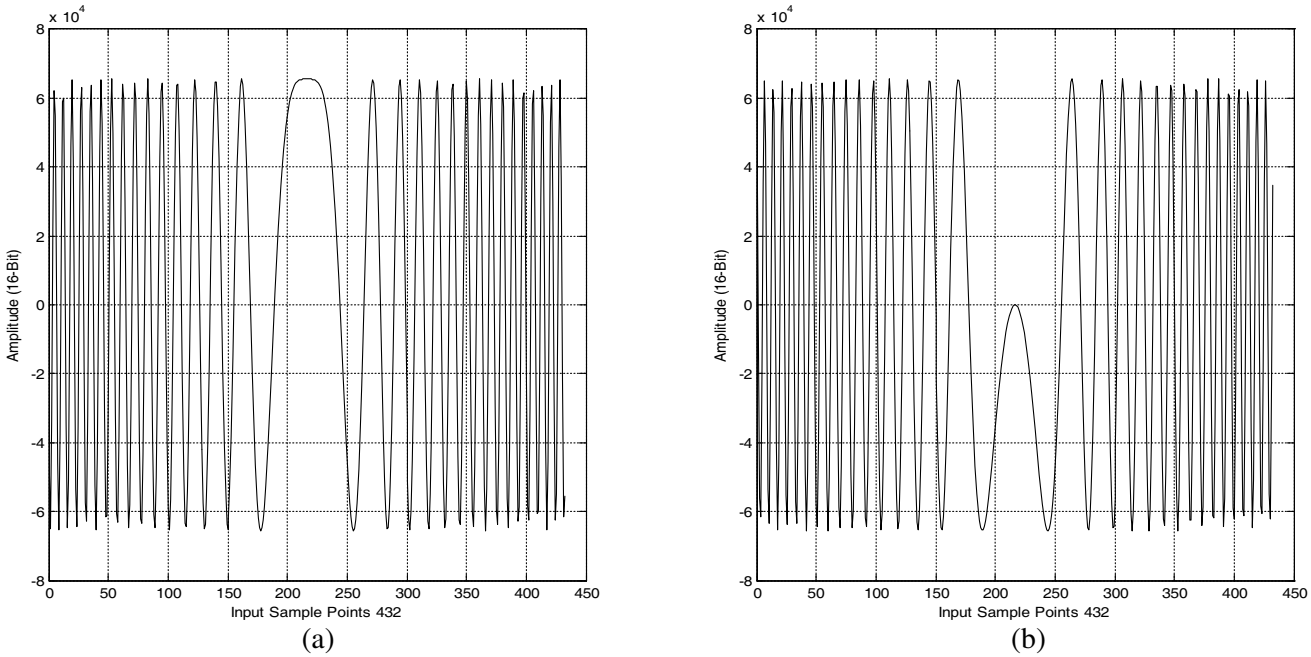

Figure 18. 16 bits input complex chirp signal: (a) real part, (b) imaginary part.

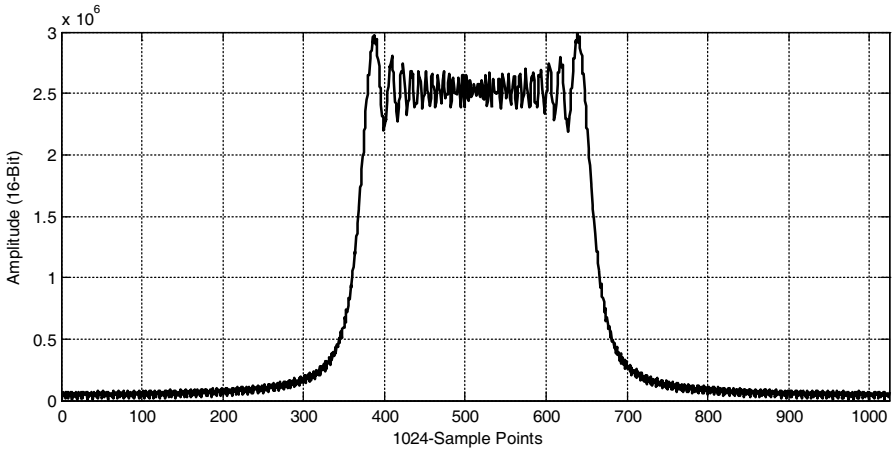

Figure 19. Frequency domain spectrum plot transformed by the proposed 1024-FFT coprocessor.

with zero padded data in the addresses that represent the transform length. Figure 19 and Figure 20 illustrate the frequency domain spectrum plots of the complex chirp signal transformed by the proposed 1024-FFT coprocessor and MATLAB.

From the figures, visually the two transformed signals are almost identical in terms of shape and also amplitude level. In order to have a quantitative comparison, the differences between the two FFT outputs of MATLAB and FPGA in terms of the magnitude is calculated and compared. The figure of merit in terms of percentage is calculated by using the following equation where s is the input chirp signal to be transformed, and the MATLAB FFT function is treated as the reference.

$$\Delta(\%) = \frac{Magnitude\ of\ FFT_{FPGA}(s) - Magnitude\ of\ FFT_{MATLAB}(s)}{Magnitude\ of\ FFT_{MATLAB}} \times 100\% \tag{14}$$

The difference is illustrated in Figure 21. The performance of the proposed FFT coprocessor in terms of precision is somehow satisfied as the amplitude differences between the two FFTs which are

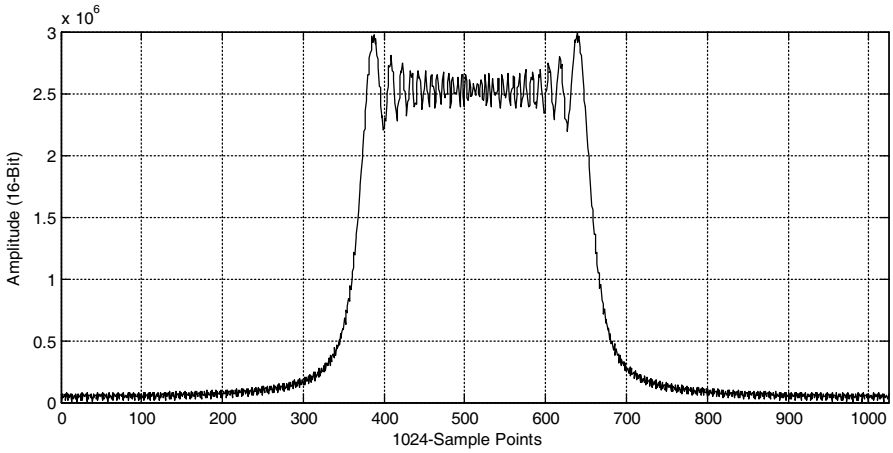

Figure 20. Frequency domain spectrum plot transformed by MATLAB 1024-FFT.

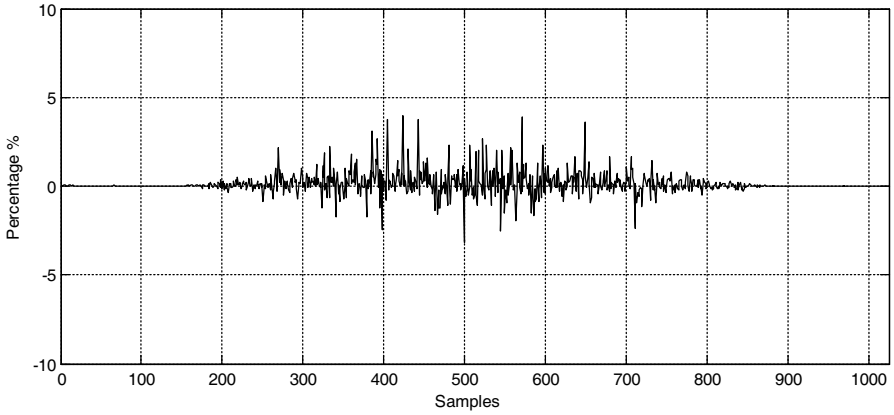

Figure 21. Difference in between the proposed 1024-FFT and MATLAB 1024-FFT.

in the range of $\pm 4\%$ only. The main reason of this error is the quantisation noise in the input signal as the data are truncated into 16 bits fixed point format.

The frequency domain spectrums as shown in Figure 19 and Figure 20 will be transformed back into time domain by using the proposed inverse FFT coprocessor and MATLAB built-in inverse FFT function, respectively. Figure 22 and Figure 23 show the time domain signals which are transformed by the proposed 1024-IFFT coprocessor and MATLAB, respectively.

Visually the two transformed signals are almost identical in the figures in terms of the shape and also amplitude level. The differences between the two IFFT outputs in terms of percentage are calculated in the same manner as in Equation (14). The differences for real and imaginary data are illustrated in Figure 24. Similar to the evaluation of the proposed FFT coprocessor, the performance of the proposed inverse FFT coprocessor in terms of precision is satisfactory as the differences for both FFTs are only in the range of $\pm 4\%$. The main reason for this error is the quantisation noise as discussed earlier. The differences in terms of magnitude between the IFFT real and imaginary signals will cause the phase to be changed, and the SAR image generated will be affected in terms of the signal phase. Therefore, the differences have to be made as small as possible ($< 5\%$) in order not to affect the quality of SAR image generated.

Throughout the evaluations of the proposed FFT and IFFT coprocessors, it can be seen that the proposed coprocessors show the promising results in the time-frequency transformation. Other than that, the performances of the coprocessors in the aspects of hardware utilisations, timing performance, and precision wise are satisfactory.

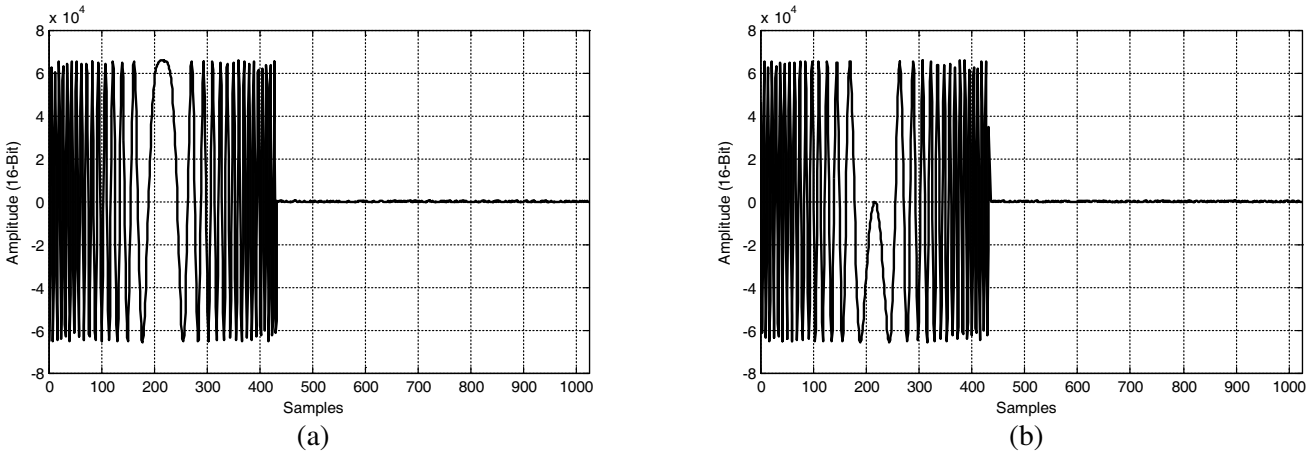

Figure 22. Time domain signal plot transformed by the proposed 1024-IFFT coprocessor, (a) real data, (b) imaginary data.

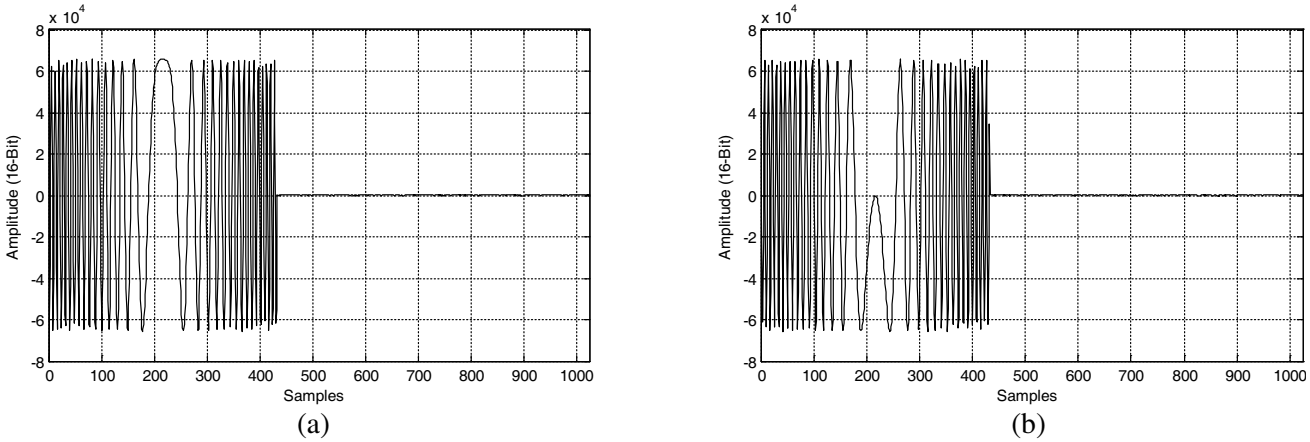

Figure 23. Time domain signal plot transformed by proposed 1024-IFFT coprocessor, (a) real data, (b) imaginary data.

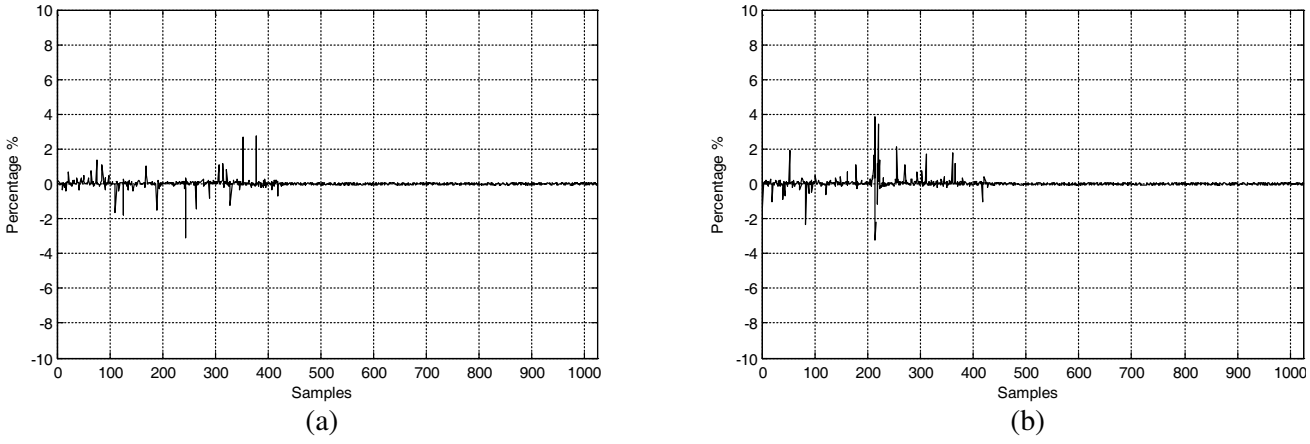

Figure 24. Difference in between proposed 1024-IFFT and MATLAB 1024-IFFT, (a) real data, (b) imaginary data.

4. CONCLUSION

In this paper, a Verilog HDL based design that is used in the implementation of FFT/IFFT coprocessors and SAR processor is introduced. In depth exploration and discussion of DIF radix-2 Fast Fourier Transform and its inverse are carried out. Based on the studied FFT/IFFT algorithm, the hardware implementation architectures of the FFT/IFFT coprocessors are proposed and discussed. The proposed coprocessors undergo several performance analysis and evaluations in the aspects of hardware resources utilisations, timing performance analysis, and precision. These evaluated coprocessors and additional sub modules can be further implemented and integrated as a SAR Processor in order to process the SAR raw data into SAR image. The hardware implementation of FPGA-based SAR processor can be carried out with the FFT/IFFT coprocessors developed and discussed in this paper. The FFT/IFFT coprocessors will be part of the hardware sub-modules integration for the SAR processor.

ACKNOWLEDGMENT

This research is supported by MOSTI E-Science Fund, 04-02-01-SF0200.3.

REFERENCES

1. Drinkwater, M. K., et al., "Synthetic aperture radar polarimetry of sea ice," *Proceeding of the 1990 International Geoscience and Remote Sensing Symposium*, Vol. 2, 1525–1528, 1990.
2. Lynne, G. L. and G. R. Taylor, "Geological assessment of SIR-B imagery of the Amadeus Basin," *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 24, No. 4, 575–581, 1986.
3. Hovland, H. A., et al., "Slick detection in SAR images," *Proceeding of the 1994 International Geoscience and Remmote Sensing Symposium*, 2038–2040, 1994.
4. Walker, B., et al., "A high-resolution, four-band SAR Testbed with real-time image formation," *Proceeding of the 1986 International Geoscience and Remmote Sensing Symposium*, 1881–1885, 1996.
5. Suzuki, S., M. Tsuchiya, O. Ochiai, T. Endo, H. Tanimoto, and H. Okubo, "Initial check-out result of the ALOS ground data system," *IEEE International Conference on Geoscience and Remote Sensing Symposium 2006*, 329–331, 2006.
6. Kong, J. A., et al., "Classification of earth terrain using polarimetric synthetic aperture radar images," *Progress In Electromagnetics Research*, Vol. 3, 327–370, 1990.
7. Bazi, Y., L. Bruzzone, and F. Melgani, "An unsupervised approach based on the generalized Gaussian model to automatic change detection in multitemporal SAR images," *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 43, No. 4, 874–887, 2005, doi: 10.1109/TGRS.2004.842441.
8. Ciuonzo, D., V. Carotenuto, and A. De Maio, "On multiple covariance equality testing with application to SAR change detection," *IEEE Transactions on Signal Processing*, Vol. 65, No. 19, 5078–5091, 2017, doi: 10.1109/TSP.2017.2712124.
9. Chan, Y. K. and V. Koo, "An introduction to synthetic aperture radar (SAR)," *Progress In Electromagnetics Research B*, Vol. 2, 27–60, 2008.
10. Vachon, P. W., et al., "Airborne and spaceborne synthetic aperture radar observations of ocean waves," *Atmosphere-Ocean*, Vol. 32, No. 1, 83–112, 1994.
11. Esposito, C., et al., "On the capabilities of the Italian airborne FMCW AXIS InSARSyste," *Remote Sensing*, Vol. 12, No. 3, 539, 2020.
12. Reigber, A., et al., "The high-resolution digital-beamforming airborne SAR system DBFSAR," *Remote Sensing*, Vol. 12, No. 11, 1710, 2020.
13. Yoon, S. S., et al., "A modified SweepSAR mode with dual channels for high resolution and wide swath," *Journal of Electromagnetic Engineering and Science*, Vol. 18, No. 3, 199–205, 2018.
14. Kraus, T., et al., "TerraSAR-X staring spotlight mode optimization and global performance predictions," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, Vol. 9, No. 3, 1015–1027, 2016.

15. Chang, C. Y., et al., "Squint mode processing algorithm," *Proc. IGARSS'89*, 1702–1706, 1989.
16. Cumming, I. G., et al., "Interpretations of the Omega-K algorithm and comparisons with other algorithms," *Proc. IGARSS'2003*, 1455–1458, 2003.
17. Cumming, I. G. and F. H. Wong, *Digital Processing of Synthetic Aperture Radar Data*, Artech House, London, 2005.
18. Franceschetti, G. and R. Lanari, *Synthetic Aperture Radar Processing*, CRC Press LLC, New York, 1999.
19. Raney, R. K., et al., "Precision SAR processing using chirp scaling," *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 32, No. 4, 786–799, 1994.
20. Soumekh, M., *Synthetic Aperture Radar Signal Processing*, John Wiley & Sons, 1999.
21. Xie, X., et al., "Embedded synthetic aperture radar imaging system on compact DSP platform," *2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, 1–4, 2017.
22. Jin, M. Y. and C. Wu, "A SAR correlation algorithm which accommodates large range migration," *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 22, No. 6, 592–597, 1984.
23. Wu, C., et al., "Modelling and a correlation algorithm for space borne SAR signals," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 18, No. 5, 563–574, 1982.
24. Curlander, J. C. and R. N. McDounough, *Synthetic Aperture Radar, Systems and Signal Processing*, John Wiley & Sons, New York, 1991.
25. Le, C., S. Chan, F. Cheng, W. Fang, M. Fischman, S. Hensley, R. Johnson, M. Jourdan, M. Marina, B. Parham, F. Rogez, P. Rosen, B. Shah, and S. Taft, "Onboard FPGA-based SAR processing for future spaceborne systems," *Proceedings of the IEEE in Radar Conference*, 15–20, 2004.
26. Kuon, I. and J. Rose, Measuring the gap between FPGAs and ASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 26, No. 2, 203–215, 2007.
27. "Verilog's inventor nabs EDA's Kaufman award," *EE Times*, Cambridge, 2005.
28. Thomas, D. E. and P. R. Moorby, *The Verilog®Hardware Description Language*, Springer, New York, 2013.
29. Lin, M., et al., "Simulation acceleration for dynamic timing analysis with static timing analysis," *TENCON 2006 — 2006 IEEE Region 10 Conference*, 1–4, 2006.
30. Cooley, J. W. and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics Computation*, Vol. 19, 297–301, 1965.