Chapter 4

High Resolution Single-Chip Radix II FFT Processor for High-Tech Application

Rozita Teymourzadeh

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/66745

Abstract

Electrical motors are vital components of many industrial processes and their operation failure leads losing in production line. Motor functionality and its behavior should be monitored to avoid production failure catastrophe. Hence, a high-tech DSP processor is a significant method for electrical harmonic analysis that can be realized as embedded systems. This chapter introduces principal embedded design of novel high-tech 1024-point FFT processor architecture for high performance harmonic measurement techniques. In FFT processor algorithm pipelining and parallel implementation are incorporated in order to enhance the performance. The proposed FFT makes use of floating point to realize higher precision FFT. Since floating-point architecture limits the maximum clock frequency and increases the power consumption, the chapter focuses on improving the speed, area, resolution and power consumption, as well as latency for the FFT. It illustrates very large-scale integration (VLSI) implementation of the floating-point parallel pipelined (FPP) 1024-point Radix II FFT processor with applying novel architecture that makes use of only single butterfly incorporation of intelligent controller. The functionality of the conventional Radix II FFT was verified as embedded in FPGA prototyping. For area and power consumption, the proposed Radix II FPP-FFT was optimized in ASIC under Silterra 0.18 μm and Mimos 0.35 μm technology libraries.

Keywords: FFT, butterfly, Radix, floating point, high speed, FPGA, Embedded, VLSI

1. Introduction

The prevalent subject of Fourier analysis encompasses a vast spectrum of mathematics where parts may appear quite different at first glance. In Fourier analysis, the term Fourier transform often refers to the process that decomposes a given function into the harmonics domain. This process results in another function that describes what frequencies are in the original function.

Meanwhile, the transformation is often given a more specific name depending upon the domain and other properties of the function being transformed.

2. Fourier transform fundamental

Fourier transform was introduced with the main concepts of discrete Fourier transform (DFT) [1] in the heart of most DSP processor. The DFT is a Fourier representation of a finite-length sequence which is the most important fundamental operation in digital signal processing and communication system [2, 3]. However, the computation complexity of the direct evaluation of an N-point DFT involves a long phase computational time and large power consumption [4]. As a result of these problems, it is important to develop a fast algorithm. There are numerous viewpoints that can be taken toward the derivation and interpretation of the DFT representation of a finite-duration sequence. The sequence of $\tilde{x}(n)$ that is periodic with period N so that $\tilde{x}(n) = \tilde{x}(n + kN)$ functions for any integer value of k. It is possible to represent $\tilde{x}(n)$ in terms of Fourier series that is represented by the sum of sine and cosine values or equivalently complex exponential sequences with frequencies that are integer multiplies of the fundamental frequencies $2\pi/N$ associated with the periodic sequence. The same representation can be applied to finite-duration sequence. The resulting Fourier representation for finite duration sequences will be referred to as the DFT. Sequence of length N by a periodic sequence can be represented by a periodic sequence with period N, one period of which is identical to the finite-duration sequence. The sampled sequence signal in frequency is defined as

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \tag{1}$$

The DFT $X(\omega)$ is a function of continuous-frequency variable ω, and the summation in Eq. (1) extends toward positive and negative infinitively. Therefore, the DFT is a theoretical Fourier transform of a digital signal. However, it cannot be implemented for real applications. It is the sample of the signal in time domain at a particular time and can be expressed as:

$$x(n) = \int_{0}^{\infty} x(t)\delta\left(t - t_n\right) \tag{2}$$

The frequency analysis of a finite-length sequence is equal to the sample of continuous frequency variable ω at N equally spaced frequencies $\omega k = 2\pi k/N$ for $k = 0, 1, 2, \ldots, N - 1$ on the unit circle. These frequency samples are expressed as:

$$\begin{gathered} X(k) = X\left(\omega_k\right), \quad \omega_k = \frac{2\pi k}{N} \\ X(k) = \sum_{n=0}^{N-1} x(n)e^{\frac{-j2\pi kn}{N}} = \sum_{n=0}^{N-1} x(n)W_N^{kn}, \quad k = 0, 1, \ldots, N-1 \end{gathered} \tag{3}$$

where the twiddle factors are defined as:

$$W_N^{kn} = e^{-j\left(\frac{2\pi}{N}\right)kn} = \cos\left(\frac{2\pi kn}{N}\right) - j\sin\left(\frac{2\pi kn}{N}\right) \tag{4}$$

The DFT is based on the assumption that the signal $\mathbf{x(n)}$ is periodic. Therefore, $X(k)$ for $k = 0, 1, \ldots, N - 1$ can uniquely represent a periodic sequence $x(n)$ of period N. The inverse DFT is the

http://dx.doi.org/10.5772/66745

reversed process of the DFT. It converts the frequency spectrum $X(k)$ back to the time domain signal $x(n)$ [5]:

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{\frac{j2\pi kn}{N}} = \frac{1}{N}\sum_{k=0}^{N-1} X(k)W_N^{-kn},\ n = 0, 1, \ldots, N-1 \quad (5)$$

Direct computation of an $\boldsymbol{N}$-point DFT according to Equation Eq. (5) requires $\boldsymbol{N(N-1)}$ complex additions and $\boldsymbol{N(N-1)}$ complex multiplications. The complexity for computing an N-point DFT is therefore $\boldsymbol{O(N2)}$. High computation complexity in DFT algorithm and need for having efficient Fourier processor leads for introduction of a fast Fourier transform (FFT) processor.

2.1. Fast Fourier transform (FFT) algorithm

In 1965 Cooley and Tukey [6] developed the use of FFT in order to save time and avoid unnecessary complex calculations. FFT algorithm computes an N-point forward DFT or inverse DFT (IDFT) where N is 2 power of M. FFT algorithm divides N-point data into two $N/2$-point series and performs the DFT on series individually results in the order of $O(N/2)^2$ complexity as compared with the original N^2 operations in an N-point DFT. The process of dividing can be continued until a 2-point DFT is reached. FFT algorithm computes an N-point forward DFT or inverse DFT (IDFT) where N is 2 power of m. The FFT is a family of algorithms that efficiently implements the DFT. **Table 1** shows the comparison between the calculation of direct DFT and FFT when a different number of N is applied.

	DFT		Radix II FFT	
Number of points	**Complex addition**	**Complex multiplication**	**Complex addition**	**Complex multiplication**
N	$N(N-1)$	$N2$	$N\log 2N$	$(N/2)\log 2N$
4	12	16	8	4
8	56	64	24	12
16	240	256	64	32
32	992	1024	160	80
64	4032	4096	384	192
127	16,256	16,384	896	448

Table 1. Computation complexity of DFT and FFT algorithm.

To calculate FFT algorithm, there are two well-known methods identified as DIT-FFT and DIF-FFT calculations [7–9]. In general, FFT processor has many types in terms of Fourier calculation. Taking into account different types of FFT algorithms are:

- Different Radixes, such as Radix II, Radix IV, etc., and mixed-radix algorithms.
- DIT and DIF.
- Real and complex algorithm.

Here, further detail is provided for DIT and DIF processor.

2.1.1. DIT Radix II butterfly FFT processor

The FFT structure divides input series into odd and even sequences. The number of stream in FFT is $N = 2^m$ when m is a positive integer:

$$\begin{aligned} xe(n) &= x(2m),\ m = 0,\ 1,\ \ldots.,\ \left(\frac{N}{2}\right) - 1 \\ xo(n) &= x(2m+1),\ m = 0,\ 1,\ \ldots.,\ \left(\frac{N}{2}\right) - 1 \end{aligned} \tag{6}$$

Based on the DFT definition and combination of the FFT concept, $X(k)$ can be written as:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} = \sum_{m=0}^{\left(\frac{N}{2}\right)-1} x(2m) W_N^{2mk} + \sum_{m=0}^{\left(\frac{N}{2}\right)-1} x(2m+1)\, W_N^{(2m+1)k} \tag{7}$$

Since $W_N^{2mk} = W_{N/2}^{mk}$, the equation will be simplified as:

$$X(k) = \sum_{m=0}^{\left(\frac{N}{2}\right)-1} x_e(m) W_{\frac{N}{2}}^{mk} + W_N^k \sum_{m=0}^{\left(\frac{N}{2}\right)-1} x_o(m) W_{\frac{N}{2}}^{mk} \quad k = 0, 1, \ldots, N-1 \tag{8}$$

where W_N^k is complex twiddle factor with unit amplitude and different phase angles. The 8-point FFT utilizes the twiddle factors from W_N^0 to W_N^7. The first twiddle factor $W_N^0 = 1$. All twiddle factors are distributed around the unit circle. **Figure 1** shows the twiddle factor for 8-point Fourier transform. The twiddle factor W_N^k repeats itself after every multiple of N. The twiddle factors are periodic and for 8-point FFT twiddle factor 0 and 8 are equal.

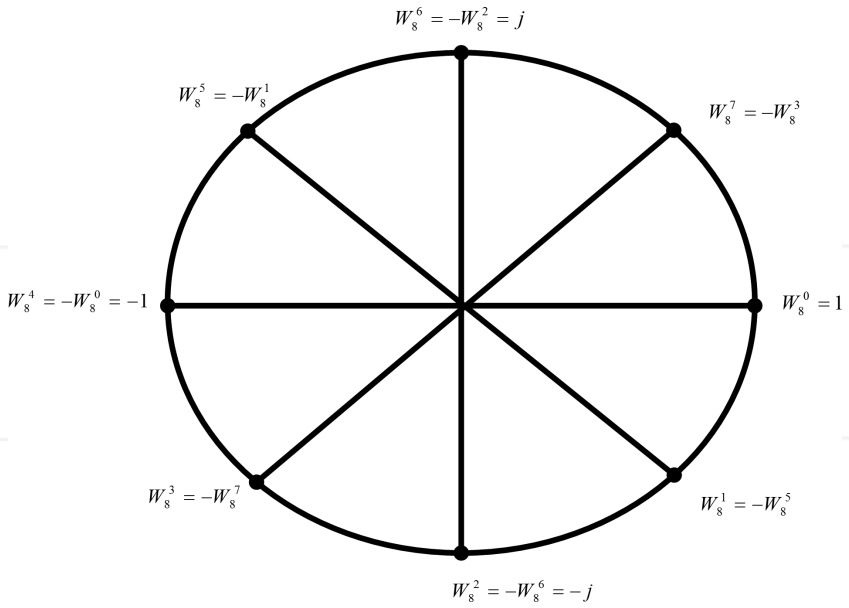

Figure 1. 8-point FFT twiddle factor.

http://dx.doi.org/10.5772/66745

By assuming $X_e(k) = \sum_{m=0}^{(\frac{N}{2})-1} x_e(m)\, W_{N/2}^{mk}$ and $X_o(k) = \sum_{m=0}^{(\frac{N}{2})-1} x_o(m) W_{\frac{N}{2}}^{mk}$, this symmetric property provides a reduction in calculations as:

$$X(k) = \begin{cases} X_e(k) + W_N^k X_o(k), \; k = 0, 1, \ldots, \left(\frac{N}{2}\right) - 1 \\ X_e(k) - W_N^k X_o(k), \; k = \left(\frac{N}{2}\right), \ldots, N-1 \end{cases} \tag{9}$$

Butterfly calculation is the fundamental concept of the FFT algorithm and 8-point butterfly structure is shown in **Figure 2**.

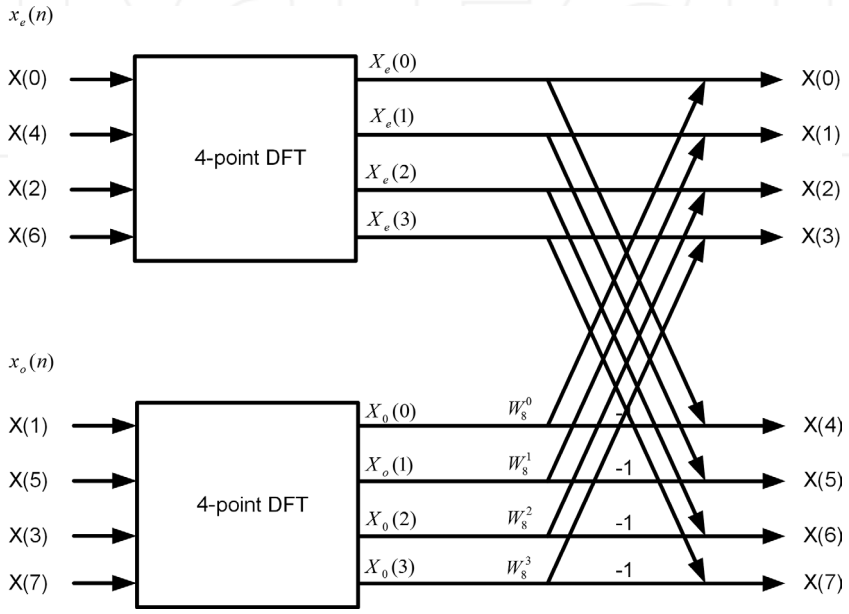

Figure 2. Decomposition of 8-point DIT FFT structure.

Radix II butterfly FFT is decomposed into M stages, where $M = \log_2^N$. In each stage, $N/2$ complexes are multiplied by the twiddle factors where N complex additions are required. Therefore, the total computational requirement is $(N \log_2^N)/2$ complex multiplications and $N \log_2^N$ complex additions. Consequently, expanding Radix II butterfly calculation into 8 data is shown in **Figure 3**.

2.1.2. DIF Radix II butterfly FFT processor

DIF-FFT calculation is similar to the DIT-FFT algorithm. As far as FFT calculation is involved, the time domain sequence is divided into two subsequences with *N/2 samples: The DFT concept of x(n)* expressed as:

$$X(k) = \sum_{n=0}^{(\frac{N}{2})-1} x(n) W_N^{nk} + \sum_{n=\frac{N}{2}}^{N-1} x(n) W_N^{nk} = \sum_{n=0}^{(\frac{N}{2})-1} x(n) W_N^{nk} + \sum_{n=0}^{(N/2)-1} x(n + N/2) W_N^{nk} \; W_N^{(\frac{N}{2})k} \tag{10}$$

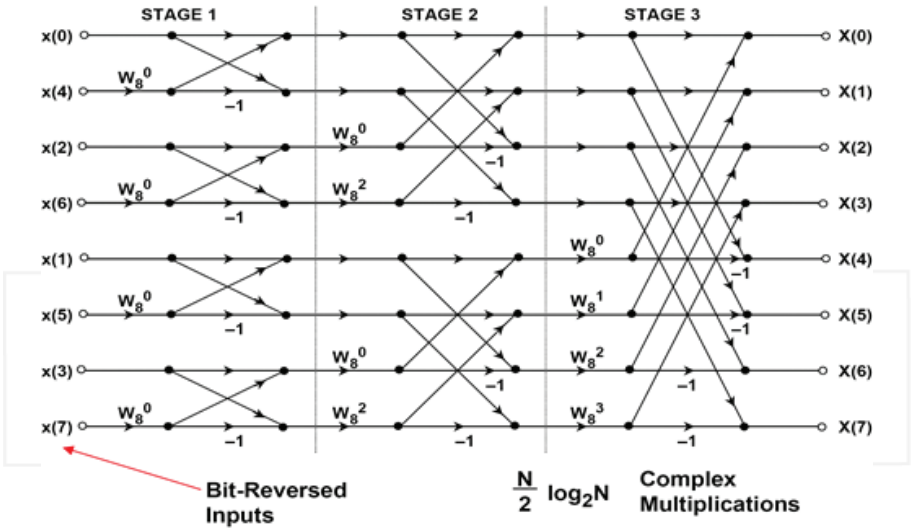

Figure 3. Decomposition of 8-point DIT-FFT.

Given that $W_N^{(\frac{N}{2})k} = (-1)^k$, Eq. (10) can be simplified to:

$$X(k) = \sum_{n=0}^{(\frac{N}{2})-1} \left[x(n) + (-1)^k x(n + N/2)\right] W_N^{nk} \tag{11}$$

Later, Eq. (11) is expanded into two parts including even $X(2k)$ and odd $X(2k+1)$ sam\twid-dle factor characteristic, Eq. (11) is simplified to:

$$\begin{aligned} &W_n^{2kn} = W_{N/2}^{kn}, W_n^{(2k+1)n} = W_N^n W_{N/2}^{kn} \\ &X(2k) = \sum_{n=0}^{(\frac{N}{2})-1} \left[x(n) + x\left(n + \frac{N}{2}\right)\right] W_{N/2}^{nk} = \sum_{n=0}^{(\frac{N}{2})-1} x_1(n) W_{N/2}^{nk} \\ &X(2k+1) = \sum_{n=0}^{(\frac{N}{2})-1} \left[x(n) - x\left(n + \frac{N}{2}\right)\right] W_N^{nk} W_{N/2}^{nk} = \sum_{n=0}^{(\frac{N}{2})-1} x_2(n) W_N^n W_{N/2}^{nk} \; k = 0,1,\ldots,\left(\frac{N}{2}\right)-1 \end{aligned} \tag{12}$$

Similarly, 8-point DIF FFT structure is shown in **Figure 4** with detail complex calculation in three stages. The output sequence $X(k)$ of the DIF-FFT is bit-reversed, while the input sequence $x(n)$ of the DIT-FFT is bit-reversed. In addition, there is a slight difference in the calculation of butterfly architecture. As shown in **Figure 4**, the complex multiplication is performed before the complex addition or subtraction in the DIT-FFT processor. In contrast, the complex subtraction is performed before the complex multiplication in the DIF-FFT. The process of decomposition is continued until the last stage is reduced to the 2-point DFT. Since the frequency samples in the DIF-FFT are bit-reversed, it is required to apply bit-reversal algorithm to the frequency samples. Likewise, the DIF-FFT algorithm also uses in-place computation.

http://dx.doi.org/10.5772/66745

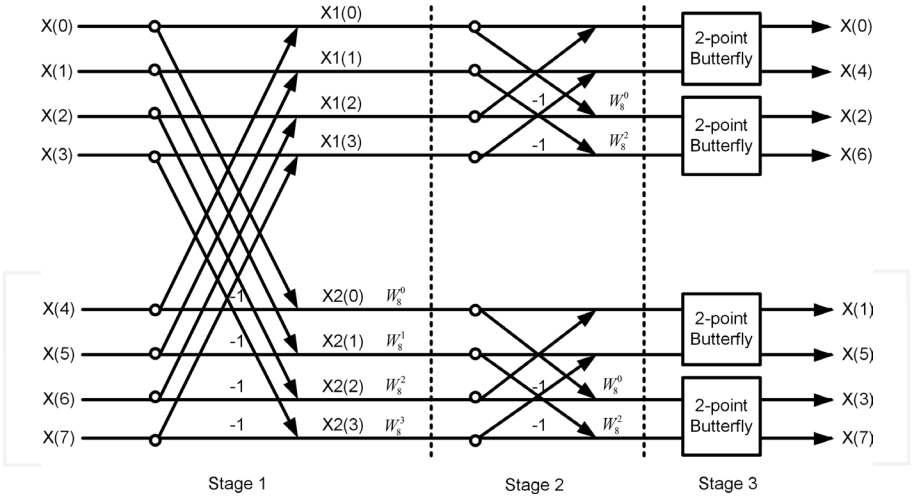

Figure 4. Internal calculation of 8-point DIF-FFT processor.

Unlike the DIF structure, input data in DIT-FFT is in bit-reverse format while the output is sorted. On the other hand, both the DIT and DIF can go from normal to shuffled data or vice versa. In order to apply Radix II FFT structures, DIT and DIF algorithms require the same number of operations and bit-reversal to compute the FFT calculation. The overall performance of the FFT processor is dependent on the application, hardware implementation, and convenience. If the design is focused on high speed structure, the processor has to take the most efficient approach and algorithm to perform the FFT calculation accordingly. In this chapter DIT-FFT architecture is considered for floating-point implementation.

2.2. Floating point FFT algorithm

Measured frequency by FFT will be subjected to quantization noise error with respect to the real frequency. This is caused by the fact that the FFT only computes the spectrum at discrete frequencies. This error is said to affect the accuracy. In addition, spectral leakage effect becomes very significant when small amplitude harmonics are close to large amplitude ones since they become hidden by the energy distribution of the larger harmonics. Furthermore, the fixed internal arithmetic calculation generates white noise in frequency domain. To reduce the generated noise effect and enhance signal strength, floating-point technique is designed and implemented. The floating-point technique allows numbers to be represented with a large dynamic range. Therefore, floating-point arithmetic enables the reduction of overflow problems that occur in fixed-point arithmetic. Although it is at the expense of throughput and chip area size, the new architecture is designed and investigated to avoid undesired effects in floating-point FFT algorithm. Floating-point arithmetic provides higher precision and a much larger dynamic range under IEEE 754 standard [10]. Therefore, floating-point operations support more accurate DSP operations. **Table 2** compares the efficiency between fixed-point and the floating-point FFT processor.

Fixed-point FFT	Floating-point FFT
16-bit or 24-bit	32-bit
Limited dynamic range	Large dynamic range
Overflow and quantization errors	Less error
Higher frequency	Low frequency
Less silicon area	More silicon area
Cheaper	More expensive
Low power consumption	High power consumption

Table 2. Fixed-point and floating-point FFT processor properties.

In floating-point format, the data are translated based on power and mantissa in the decimal system. This notation can be expanded into the binary system. Representing the data in power and mantissa system gives the data the capability of storing a much greater range of numbers than if the binary points were fixed. Floating point refers to the "truth" of the Radix point, which refers to the decimal point or in computers it is known as the binary point that has the capability to float. This entails the event to occur anywhere that is relative to the significant digit of the number. Thus, a floating-point representation, with its position indicated separately in the internal representation, is a computer's recognition of a scientific concept. Although the benefit of floating-point representation over fixed-point (and integer) representation is much wider in range of values, but the floating-point format needs more storage. Hence, the implementation of high performance system requires applying efficient and fast floating-point processor, which is competitive with the fixed-point processor. Various types of floating-point representation have been used in computers in the past. However, in the last decade, the IEEE 754 standard [10] has defined the representation. According to the IEEE 754 standard [10], the single precision is chosen to represent the floating-point data. The IEEE standard specifies a way in which the three values described can be represented in a 32-bit or a 64-bit binary number, referred to single and double precision, respectively [11, 12]. In this project, single precision is selected to function. For the 32-bit numbers, the first bit (MSB) specifies the sign, followed by 8 bits for the exponent, and the remaining 23 bits are used for the mantissa. This arrangement is illustrated in **Figure 5**. The sign bit is set to zero if the number is positive, and the bit is set to 1 if the number is negative. The mantissa bits are set to the fractional part of the mantissa in the original number in bits 22 to 0.

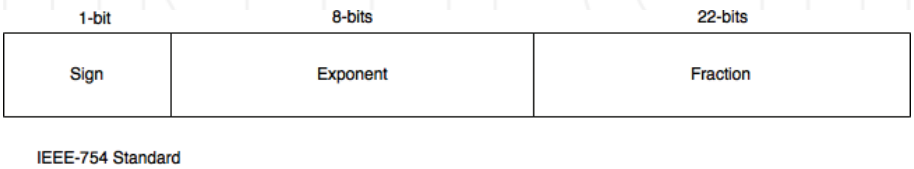

Figure 5. Floating-point structure in IEEE 754 standard [10].

Floating-point algorithm finds huge demand in industry. To conclude this section, **Table 3** summarizes the FFT algorithm application in fixed-point and floating-point architectures.

http://dx.doi.org/10.5772/66745

Fixed-point FFT	Floating-point FFT
Low resolution disk drive	Radar, Image processing
Consumer audio application	High-end audio application, ambient acoustics simulators
Channel coding	Professional audio encoding/decoding and audio mixing
Communication device	Sound synthesis in professional audio and video coding/decoding
	Prototyping
	4G OFDM Transceiver
	High resolution motor monitoring

Table 3. Fixed and floating-point FFT application.

2.3. Pipeline/parallel FFT algorithm

In 2009, Xilinx Logic core [13] introduced the FFT processor using the Radix structure on a chip. The introduced FFT processors were designed to offer a trade-off between core sizes and transform time. These architectures are classified below:

- FFT Processor with Radix II pipelined serial I/O architecture
- FFT Processor with Radix IV, parallel I/O (burst) architecture
- FFT Processor with Radix II, parallel I/O (burst) architecture
- FFT Processor Radix II lite, parallel I/O (burst) architecture

The pipeline serial I/O allows to continue data processing, whereas the burst parallel I/O loads and processes data separately by using the iterative approach. It is smaller in size than the parallel but has a longer transform time. In the case of Radix II algorithm, it uses the same iterative approach as Radix IV with the difference of smaller butterfly size that differentiates it. Yet, the transformation time is longer. Finally, for the last category, based on Radix II architecture, this variant uses a time multiplexed approach to the butterfly for an even smaller core, at the expense of longer transformation time. **Figure 6** shows the throughput versus resource among the four architectures.

2.3.1. FFT processor with Radix II pipelined, serial I/O

In this design, *n*-stage of Radix II butterfly is connected as a serial structure. Each unit of Radix II butterfly has its own RAM memory to upload and download data. The input data are stored in the RAM while the processor simultaneously performs transform calculations on the current frame of data and loads input data for the next frame of data and unloads the result of the previous frame of data. Input data are presented in sorted order. The unloaded output data can either be in bit-reversed order or in sorted order. When sorted output data are selected, an additional memory resource is utilized. **Figure 7** illustrates the architecture of the pipeline serial I/O with individual memory bank, which connects in a serial structure.

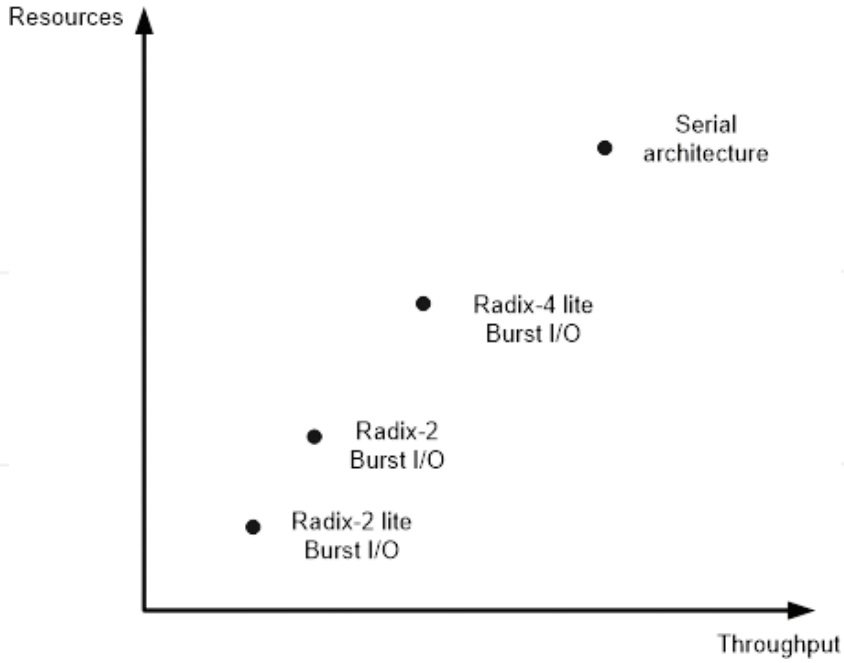

Figure 6. FFT architecture resources vs. throughput.

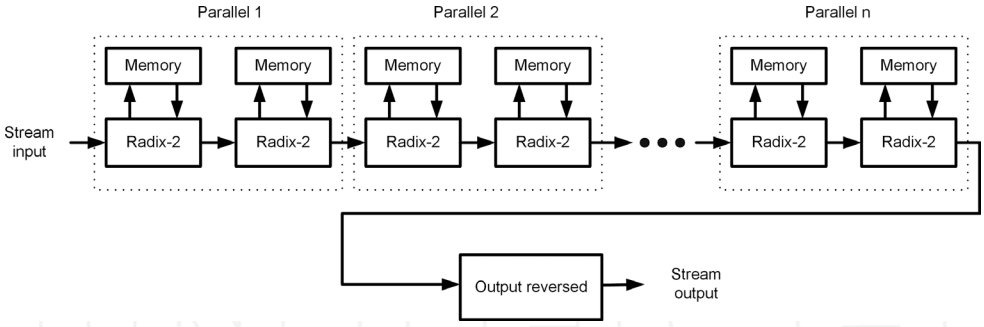

Figure 7. FFT processor with Radix II pipelined, serial I/O [13].

2.3.2. FFT processor with Radix IV, burst I/O

Radix IV structure accepts 4 input data simultaneously whereas Radix II takes only 2 input data at the time to perform FFT calculation. Radix IV input data uploaded into the FFT processor, cannot be uploaded while the calculation is underway. When the FFT is started, the data are loaded. After a full frame has been loaded, the core computes the transformation. The result can be downloaded after the full process is over. The data loading and unloading processes can be overlapped if the data are unloaded in digit-reversed order. **Figure 8** shows the Radix IV structure when 4 input data are loaded for FFT calculation.

http://dx.doi.org/10.5772/66745

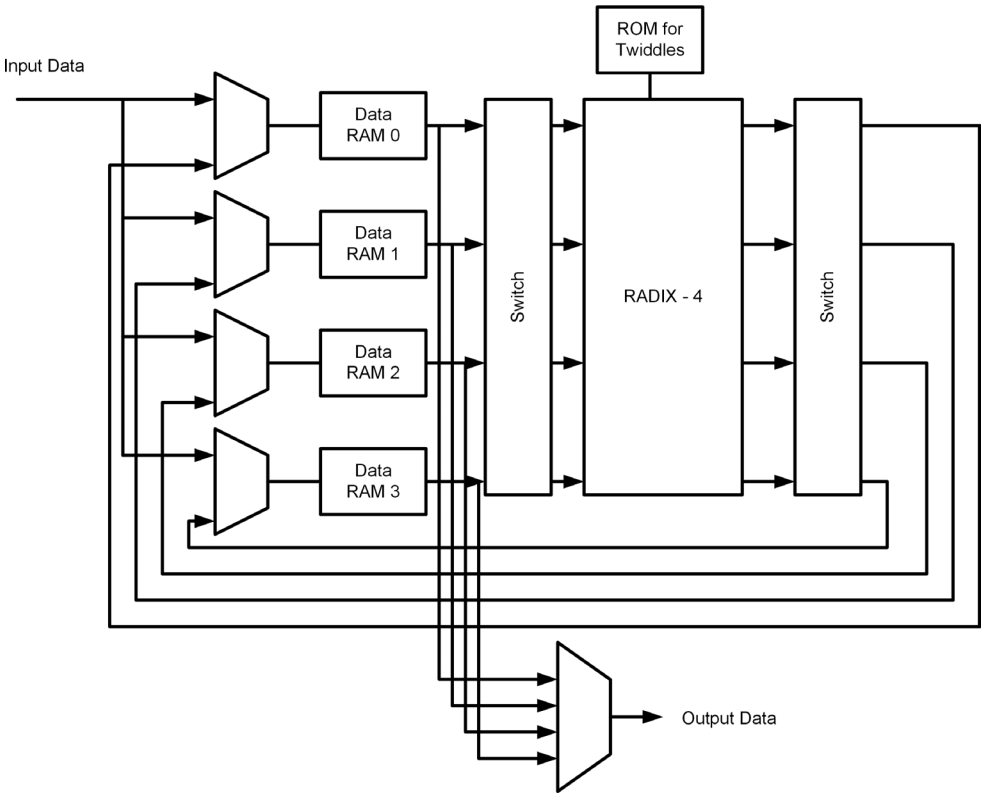

Figure 8. FFT processor with Radix IV architecture [13].

2.3.3. FFT processor with Radix II, burst I/O

FFT processor with burst I/O architecture utilizes Radix II butterfly calculation to execute the arithmetic structure. In spite of Radix IV with burst I/O processor, which the input data cannot be loaded and unloaded simultaneously, the Radix II processor accepts the input data during the FFT processor and data can be used concurrently when the output samples are in bit-reversed order. The twiddle factors are stored in the ROM blocks while the output and input data are stored in a separate or mixed RAM blocks. **Figure 9** shows the Radix II structure when 2 input data are loaded for FFT calculation.

2.3.4. FFT processor with Radix II lite, burst I/O

FFT processor with Radix II lite architecture uses one shared RAM, hence reducing resources at the expense of an additional delay per butterfly calculation. The multiplier in this structure multiplies the real part of complex number in one clock cycle and the imaginary in the next. In this architecture, the data can be simultaneously loaded and unloaded if the output samples are in bit-reversed order. In this architecture, sine and cosine twiddle factor coefficient will be saved in the ROM and the output data will be saved in a single RAM. Although this proposed architecture saves the resources, the throughput is significantly limited by the FFT structure

due to the sequence calculations. **Figure 10** shows the Radix II lite structure when 2 input data are loaded for FFT calculations.

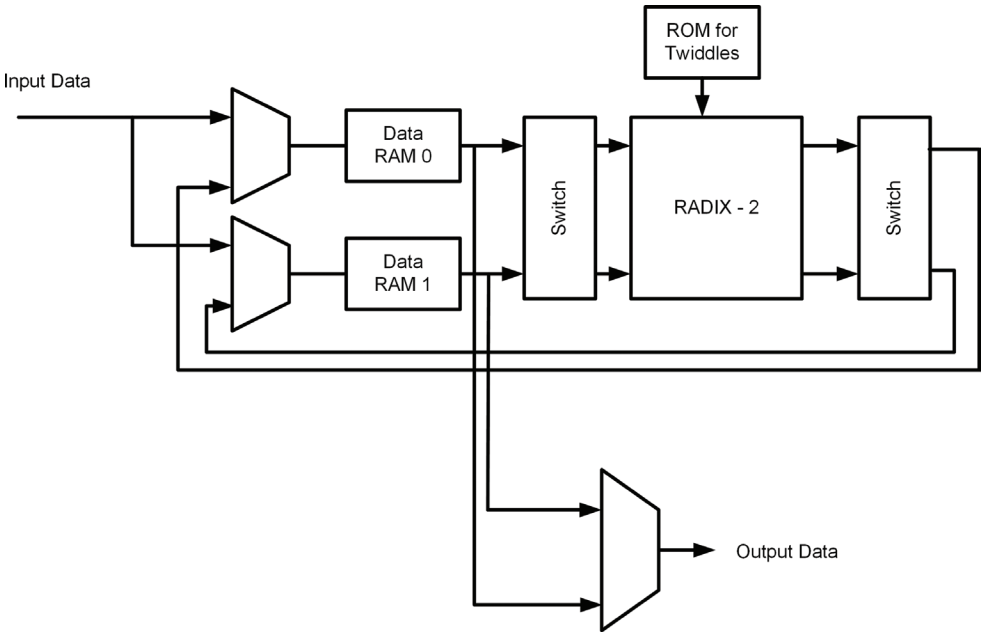

Figure 9. FFT processor with Radix II burst I/O architecture [13].

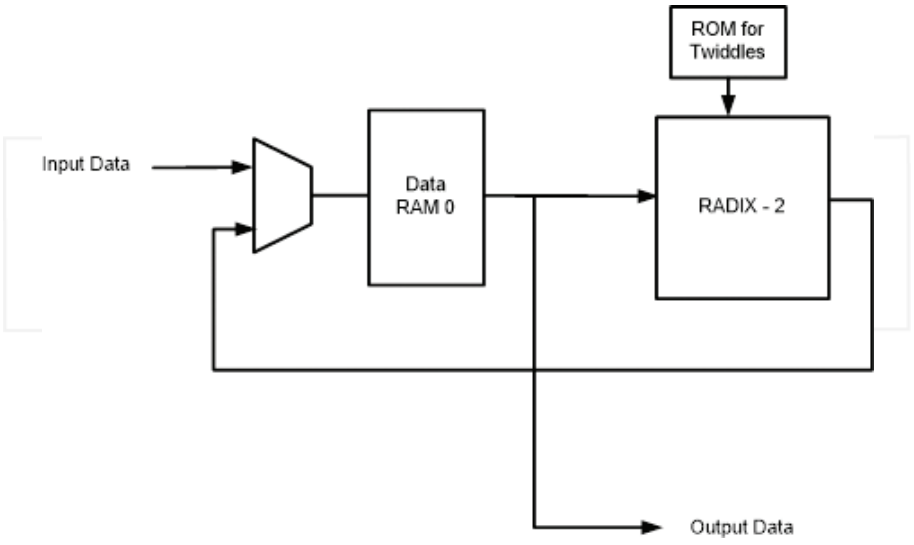

Figure 10. FFT processor with Radix II lite burst I/O architecture [13].

http://dx.doi.org/10.5772/66745

3. Advanced high-tech fast Fourier transform algorithm

In Section 2, FFT fundamental was discussed and elaborated. Furthermore, different FFT architectures were provided with the detail on IO configuration. Here, advance FFT processor with the focus on 1024 floating-point parallel architecture for high performance application is provided.

3.1. Stage realization of 1024-point parallel pipeline FFT structure

High-tech FFT principle is based on Radix II algorithm in floating-point format to conduct 1024 point FFT structure. **Figure 11** illustrates the main block diagram of the 1024-point Radix II floating-point parallel pipeline (FPP) FFT processor in detail.

As shown in **Figure 11**, there are six major subprocessor units in the high-tech 1024 point Radix II FPP-FFT algorithm. These units are shared memory, bit reverse, butterfly arithmetic, smart controller, ROM, and finally address generator unit. The floating-point input data act as a variable streaming configuration into the processor. The variable streaming configuration allows continuous streaming of input data and produces continuous stream of output data. **Figure 12** shows the internal schematic of the pipeline butterfly algorithm with the parallel architecture at a glance.

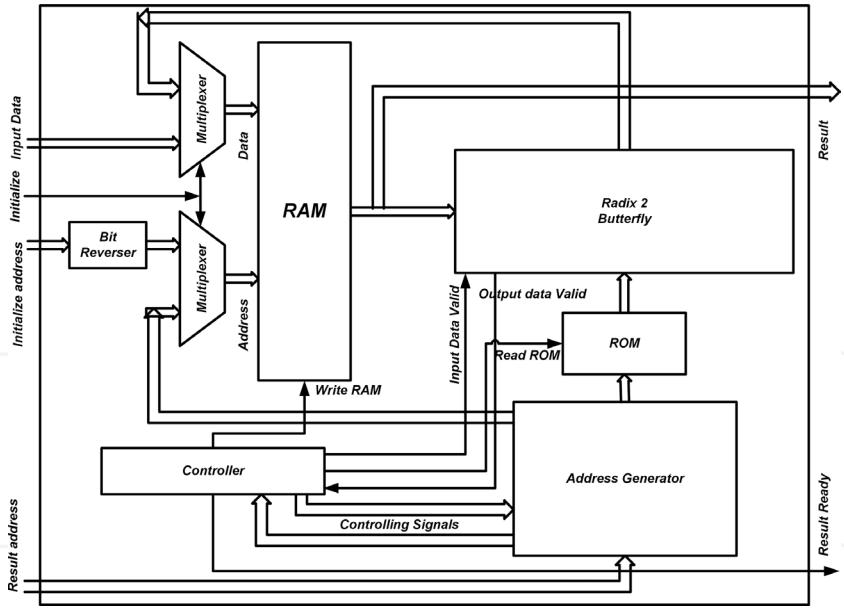

Figure 11. 1024 point Radix II FPP-FFT block diagram.

To enhance the speed of calculation in Radix II butterfly algorithm, the pipeline registers are located after each addition, subtraction, and multiplication subprocessors. Hence, the pipeline butterfly algorithm keeps the final result in the register to be transferred into the

RAM by the next clock cycle. Additionally, the parallel architecture splits the data in real and imaginary format and increases the speed of FFT calculation by 50%. As a result of the design algorithm, Radix II FPP-FFT processor calculates 1024 point floating-point FFT exactly after $O\left(N/2\log_2^N\right)+11$ clock a pulse which proves the performance improvement in comparison with similar Radix II FFT architecture. The existence of 11 clock pulses delay is due to 11 pipeline registers in adder, subtraction, and multiplier in a serial butterfly block. Additionally, parallel design of the FFT algorithm decreases the calculation time significantly.

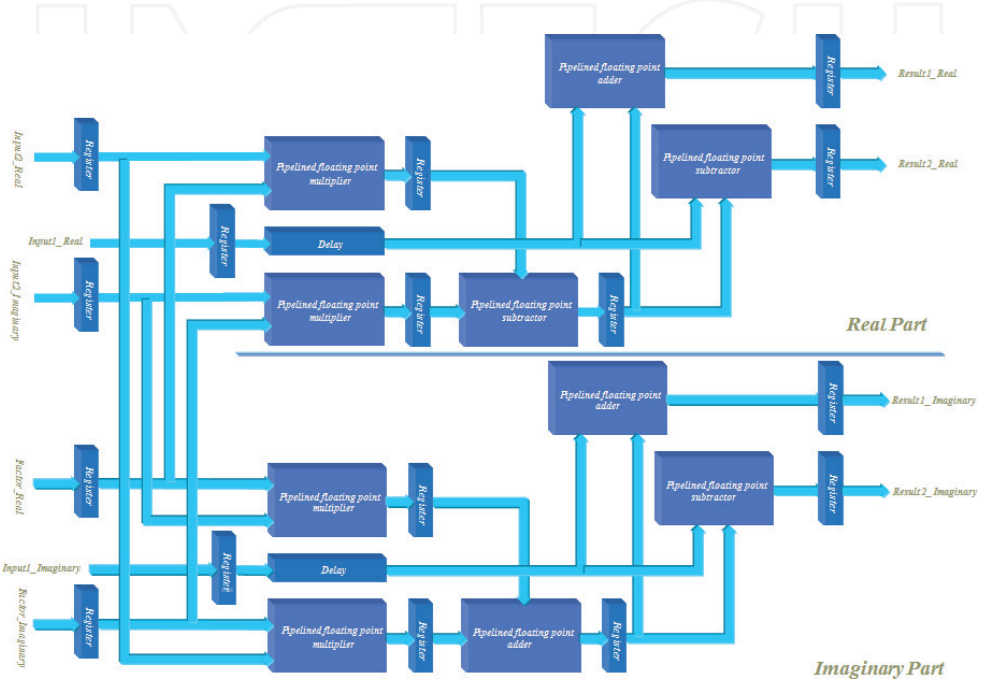

Figure 12. Designed FPP Radix II butterfly structure.

Radix II butterfly unit is responsible for calculating the complex butterfly equations as $\text{output1} = \text{input1} + W^k \times \text{input2}$ and $\text{output2} = \text{input1} - W^k \times \text{input}$. To calculate the butterfly equation, it is necessary to initiate the RAM with bit-reverse format and the external processor loads the data in the RAM. Since butterfly equation deals with complex data, thus each butterfly requires four multiplication units (two for the real and two for the imaginary) and six additional units (three for the real and three for the imaginary part). Fixed point implementation of such complex calculation does not satisfy high-tech application of FFT processor due to the generated noise of round-off, overflow, and coefficient quantization errors [14]. Consequently in order to reduce the error as well as to achieve high-resolution output, the floating-point adders and subtractions are used to replace the fixed-point arithmetic units.

http://dx.doi.org/10.5772/66745

3.1.1. Floating point adder/subtraction

Butterfly processor efficiency greatly depends on its arithmetic units, and high-speed floating-point adder is the bottle neck of butterfly calculation. Based on IEEE-754 standard [10] for floating-point arithmetic, 32-bit data register is considered to allocate mantissa, exponent, and sign bit in a portion of 23, 8, and 1 bits, respectively. The advantages of floating-point adder are that the bias power is applied to complete the calculation and avoid using unsigned value. Additionally, the floating-point adder unit performs the addition and subtraction using substantially the same hardware as used for the floating-point operations. This functionality minimizes the core area by minimizing the number of elements. Furthermore, each block of floating-point adder/subtraction operates the arithmetic calculation within only one clock cycle that results high-throughput and low latency for the entire FFT processor. **Figure 13** shows the novel structure of the floating-point adder when it is divided into four separate blocks while detail algorithm is presented in **Figure 14**.

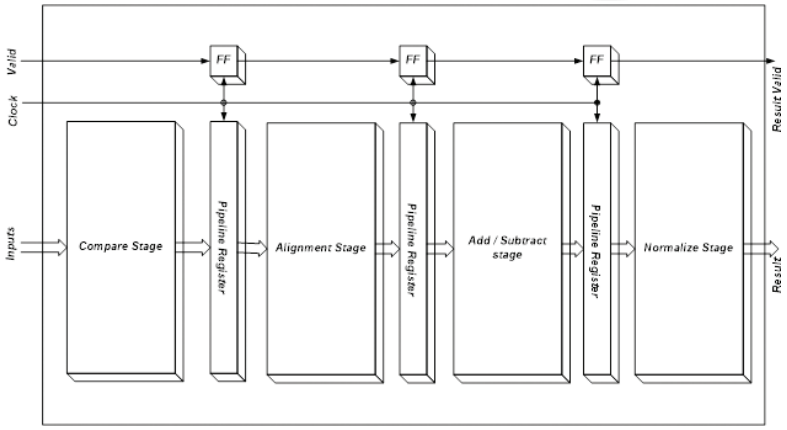

Figure 13. Schematic diagram of advance floating-point adder.

The purpose of having separate blocks is to share the total critical path delay into three equal blocks. These blocks calculate the arithmetic function within one clock cycle. However, the propagation delay can be associated with continuous assignment to increase the overall critical path delay and for the slowing down of the throughput. Based on combinational design, the output of each stage depends on its input value at the time. The unique structure of floating-point adder enables feeding of the output result in the pipeline registers after every clock cycles. Hence, the sequential structure is applied for the overall pipelined add/subtraction algorithm to combine the stages. The processing flow of the floating-point addition/subtraction operation consists of comparison, alignment, addition/subtraction, and normalization stages.

The comparison stage compares two input exponents. This unit compares two exponents and provides the result for the next stage. The comparison is made by two subtraction units and the result is revealed by *compare_sign* bit.

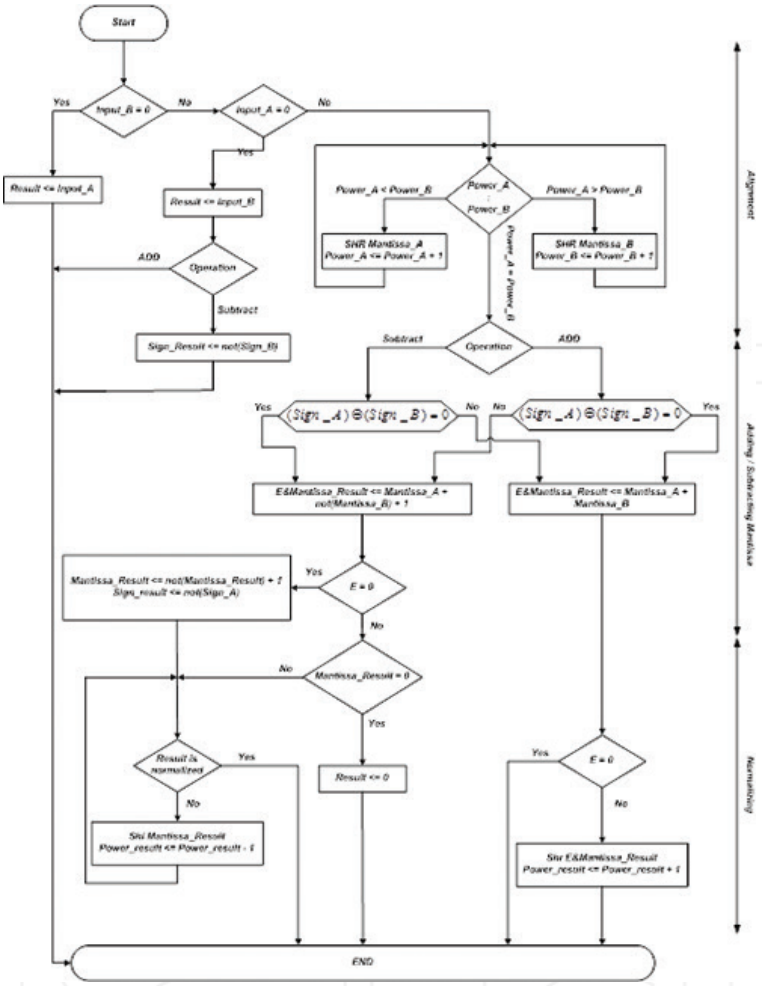

Figure 14. Flowchart of advance floating-point adder.

According to the results of the comparison stage, the alignment stage shifts the mantissa and transfers it to the adder/subtraction stage. The number of shifting will be selected by the comparison stage output. Consequently, each stage of the floating-point adder algorithm is executed within one clock cycle. Floating-point adder/subtraction unit satisfies high speed and efficiency of arithmetic unit in cost of die area size. The floating-point arithmetic unit is designed to calculate entire numbers regardless of the number sign. As shown in **Figure 15**, there is a logic gate involved with the stages, which cause higher delay propagation through the circuit.

Floating-point numbers are generally stored in registers as normalized numbers. This means that the most significant bit of the mantissa has a nonzero value. Employing this method allows the most accurate value of a number to be stored in a register. For this purpose, the

http://dx.doi.org/10.5772/66745

normalized stage is required. This unit is located after the add/sub stage. The output signal representing the add/sub block leads to zero digits of an unnormalized result of the calculation operation. The normalized block ignores the digital value of zero from the MSB of the mantissa and shifts the mantissa to imply value of one in digital as MSB in mantissa.

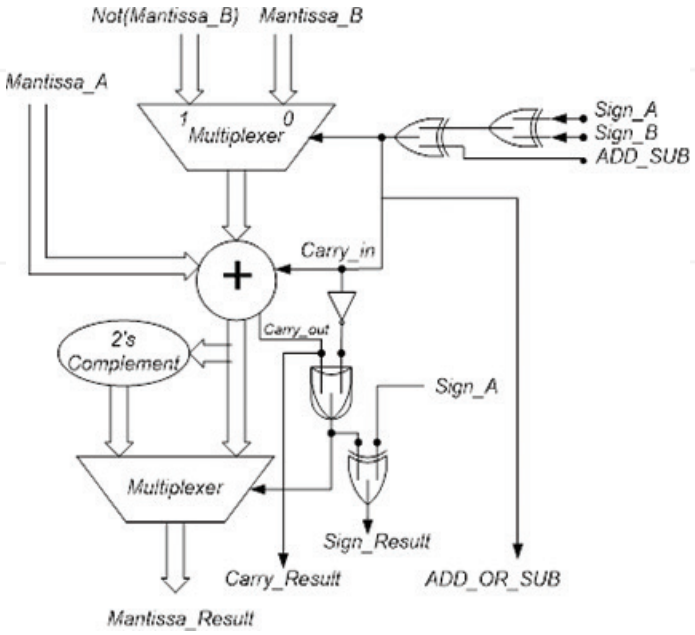

Figure 15. Addition/subtraction structure.

3.1.2. Floating-point multiplier

In a floating-point multiplier, numbers are represented in single-precision normalized mantissa and 8-bit exponent format defined by the IEEE 754 standard. This structure has developed the architecture for partial-product reduction for the IEEE standard floating-point multiplication, leading to a structured high-speed floating-point multiplier. The shortening of the data path is desirable because they require shorter wires and therefore support faster operation. The former approach uses a reduction scheme based on combination unit and connects it as parallel architecture. Implementing floating-point multiplier is simpler than floating-point adder since it does not require alignment stage. The processing flow of the floating-point multiplication operation consists of multiple stage and normalized stage. **Figure 16** shows the overall block diagram of the floating-point multiplier while the flowchart of the functionality of the multiplier is shown in **Figure 17**.

In a floating-point multiplier, the bias power format is applied to avoid having negative exponent in the data format. Additionally, the multiplier is designed as pipelined structure to

enhance speed calculation, with the intention of the initial result appearing after the latency period where the result can then be obtained after every clock cycle. The multiplier offers low latency and high throughput and is IEEE 754 compliant. This design allows a trade-off between the clock frequency and the overall latency by adding the pipeline stage.

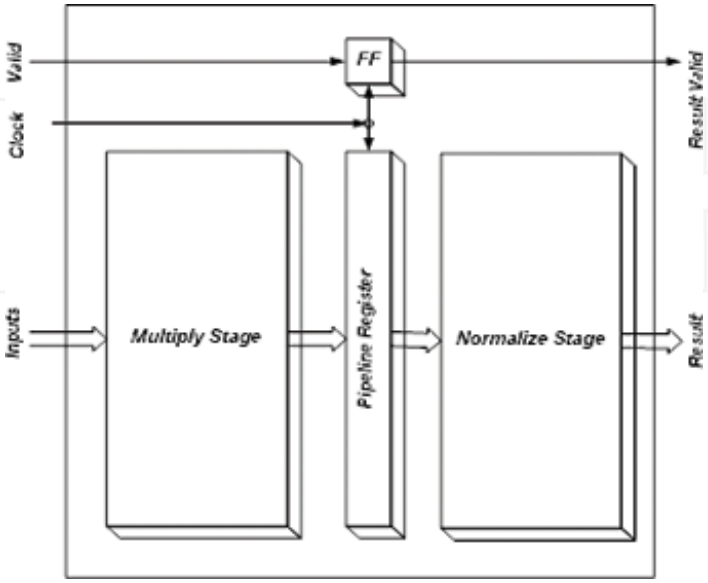

Figure 16. Floating-point multiplier block diagram.

3.1.3. Smart controller structure

Smart controller unit significantly affects the efficiency of the 1024 Radix II FPP-FFT processor. As such, small die area can be achieved by designing high performance controller for the FFT processor. In this architecture, FFT controller is designed with the pipeline capability. The global controller unit provides the signal control to the different parts of the FFT processor. Additionally, several paths are switched between the data input and data output in architecture design and the data path is controlled. To calculate the 1024 point Radix II FFT processor, it is necessary to have $\log_2 N$ stages, which are 10 stages for 1024-point data. Furthermore, each stage calculates $\frac{N}{2}$ butterfly that is 512 butterfly calculations in the design. Hence, there are two counter in corporation with the controller to count the stage number of the processor and the number of butterfly calculation. Smart controller with collaboration of address generator unit calculates 1024 point floating-point FFT by using only one butterfly structure. This functionality has great contribution on power supply as well as saving die area size. **Figure 18** shows the smart controller state machine, which controls the flow of the 1024 floating-point Radix II FFT processor.

http://dx.doi.org/10.5772/66745

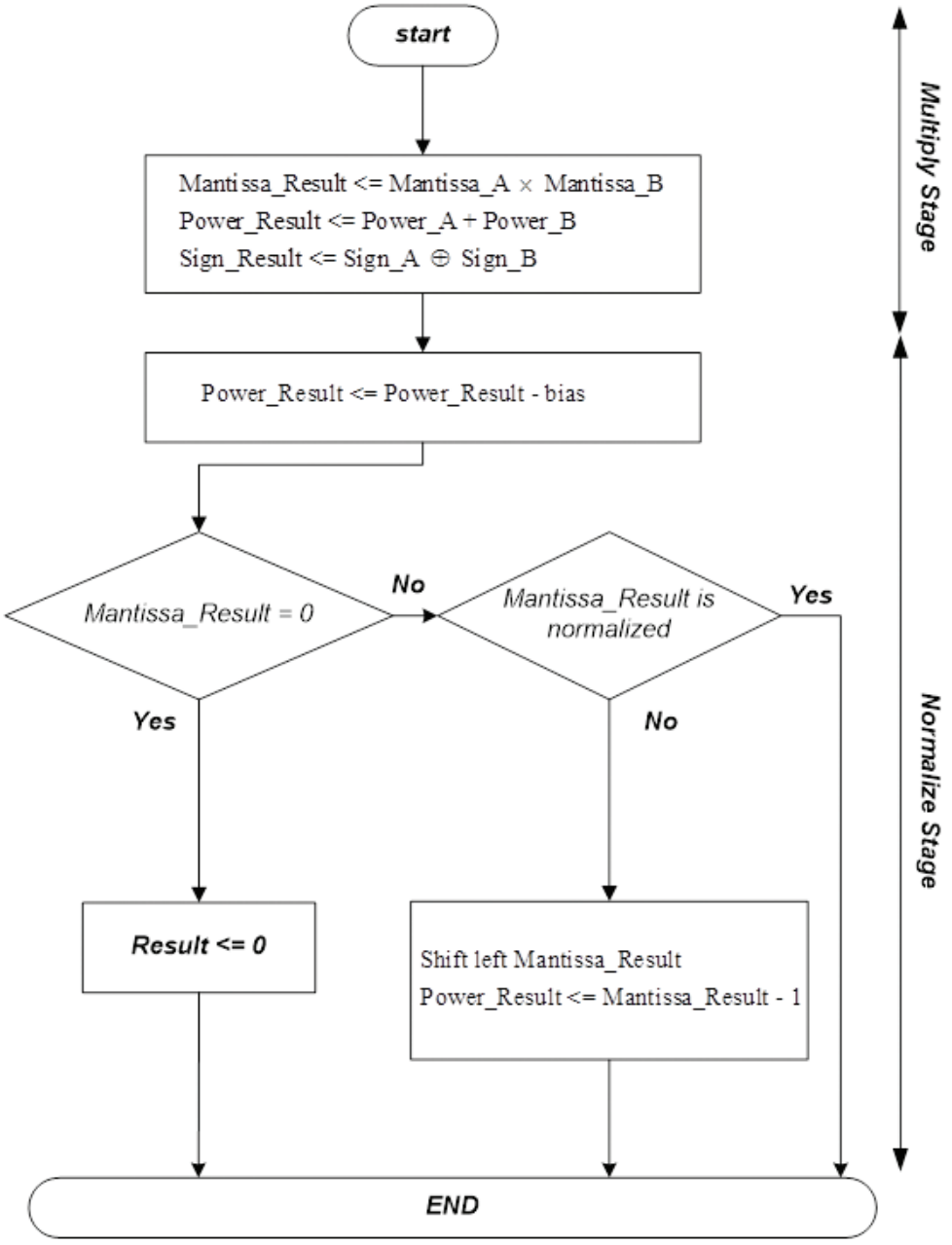

Figure 17. Floating-point multiplier flow chart.

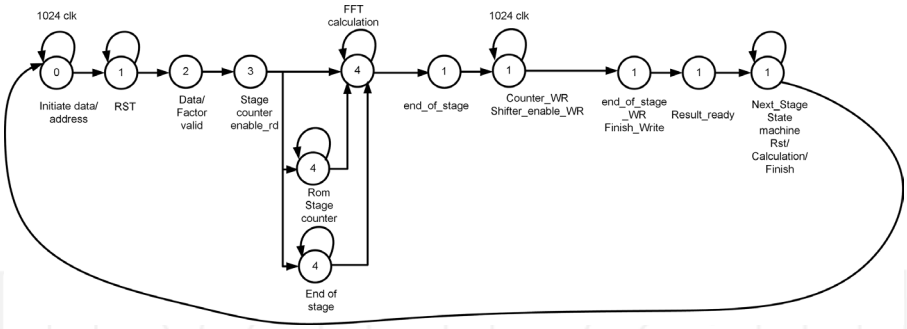

Figure 18. Smart controller state machine.

There are several control signals in smart controller to clarify the presence of correct output after finishing the current cycle of FFT calculation. The control signals transfer information through the RAM, ROM, butterfly preprocessor, and address generator. The designed controller operates according to the provided state machine (**Figure 18**) and makes the high performance FFT calculation feasible for implementation. The controller unit is structured into the subblocks such as in sequential and combination units. Sequential unit is responsible for updating the state of the processor, while the combinational unit performs the states individually. The state machine waits for processor core to complete the entire FFT calculations and then records data points into the memory. Reset state is received every time the reset input is asserted then holds the entire calculation. The processor gets activated after the reset input signal is removed.

3.1.4. Memory and address generator

Address generator has a significant task in Radix II FFT processor, since it delivers the address of the input/output data for each computational stage in an appropriate way. Address generator architecture consists of ROM address generator, Read address generator, and Write address generator. ROM address generator produces the reading address for the ROM module. The reading address represents the address of the twiddle factor, which must be taken to feed the butterfly structure. This address generator is designed to select the specific twiddle factor for the butterfly calculations. Meanwhile, the Write address generator is designed to save the result of the butterfly calculation in the proper location in the complex RAM. The proposed smart address generator is designed to provide the correct result for the next stage of the butterfly in 1024-point Radix II FFT calculations. The architecture of the Read address generator is similar to the Write address generator. The butterfly will save the data result after reading from the certain address and input it to the butterfly, in the previous address line. The reading RAM select control signal ensures the correct location of data in the complex RAM. On the other hand, memory modules are used for the storing input and output results with 1024 complex long words of 32-bit registers. The implemented architecture for the memory is shown in **Figure 19**. The capacity of the memory is 1024-point data for real and imaginary data. In high-tech implementation, shared RAM architecture is designed and implemented in a single-chip FFT processor. The high-tech design makes the Radix II FFT architecture entirely independent of the type of FPGA board since it has on

http://dx.doi.org/10.5772/66745

board memory system. Furthermore, each complex RAM has the capability of saving real and imaginary input data separately. The module is programed with a dual-in-line header to provide the appropriate location for storing input and output result in each stage consequently. It is composed of two delay memories and multiplexer, which allows straight through or crossed input-output connection as required in the pipeline algorithm. Memory unit similarly contains the controller trig. The controller, which is connected directly to the memory modules, takes the responsibility of transferring data through the memory and arithmetic blocks ensuring that no data conflict occurs within the complete process of the FFT calculations. This is another advantage of high-tech smart memory modules, by which data can be read and written in the memory simultaneously without sending bubble data in the FFT processor.

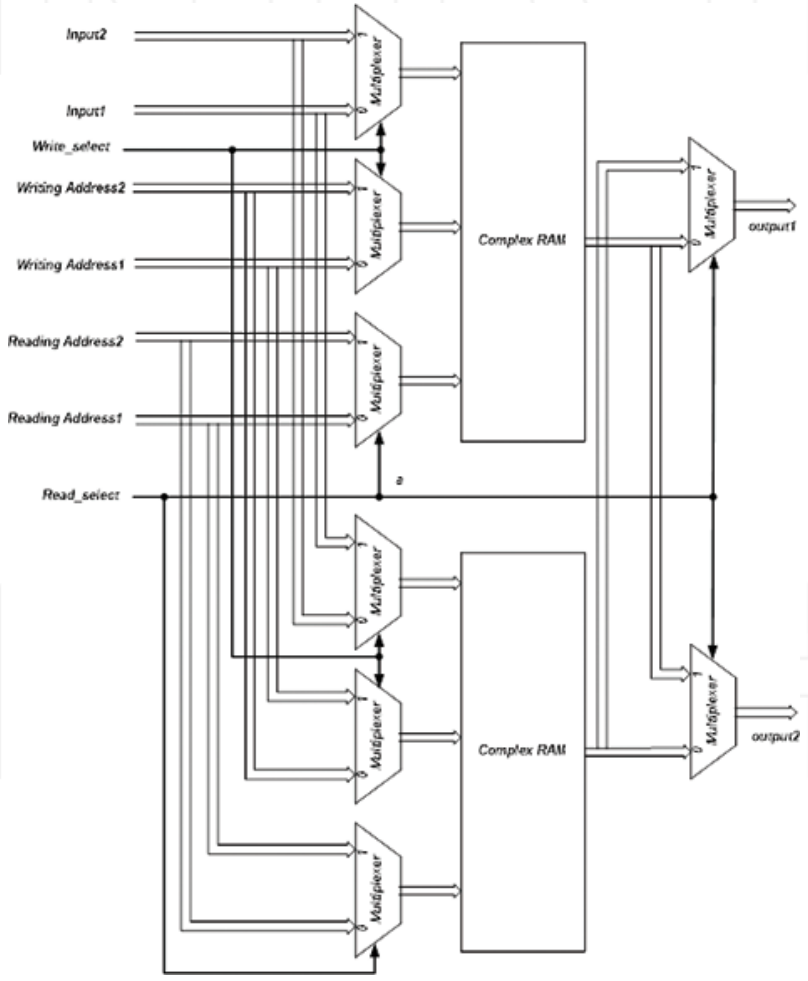

Figure 19. RAM internal architecture.

3.2. Advantages of 1024-point parallel pipeline FFT structure

Design algorithm of the 1024 point Radix II FPP-FFT processor was based on the smart sub-blocks where the result was optimized accordingly. The designed processor takes the advantages of (i) shared memory to store the input and output data and makes the system as single chip. Hence, it reduces hardware complexity. Furthermore, (ii) the entire individual arithmetic unit is designed to operate within one clock cycle to increase the maximum clock frequency. Additionally, (iii) the butterfly structure is in parallel and pipelined architecture to minimize delay caused by the FFT calculations, and finally, (iv) the strong controller with collaboration of address generator unit ignores the need of using N numbers of butterfly unit, since Radix II calculation is carried out within one butterfly unit that results reduction of power consumption, area, and avoid system complexity. The high performance processor is implemented with optimizing the architecture to enable the system in maintaining a reasonable clock rate and with low latency of $\left(N/2\,\mathrm{Log}_{2}{}^{N}\right) + 11$. The throughput of the operation is limited by the amount of available logic in the target device.

4. 1024 point FPP-FFT implementation

Section 4 details the implementation of introduced 1024-point floating-point parallel pipeline Radix II FFT algorithm. Hardware implementation of the algorithm as system on chip (SOC) is presented here.

4.1. Hardware implementation

In order to verify the functionality of the 1024-point FPP-FFT processor, the VHDL code for the overall processor is developed. Register transfer level (RTL) behavior description of the processor is generated for downloading into FPGA prototyping. The procedure is continued by attaching the library cell and constraint file for ASIC implementation. High performance FFT is transferred into the gate level synthesis to complete postsimulation stage. The design moves forward to the back-end implementation by 0.18 μm Silterra technology and 0.35 Mimos technology library. Generated netlist with constraint file is transferred to complete floor planning and place and route stage. The implementation process is summarized in **Figure 20**.

The high-tech 1024-point FPP-FFT specification generated by Xilinx ISE synthesis report is provided in **Table 4**.

As stated in **Table 4**, high-tech FFT processor operates with the maximum clock frequency of 227.7 MHz and the total latency of 5131 clock cycles (**Figure 21**) to prove the computation complexity derived from (N/2log2N) + 11 when N = 1024.

Place and route (PAR) process was completed and the processor routed successfully on silicon chip (**Figure 22**).

Later, the 1024-point FPP-FFT processor was optimized in Silterra 0.18 μm and Mimos 0.35 μm technology for power consumption and die size measurement in maximum clock frequency.

http://dx.doi.org/10.5772/66745

Table 5 shows the optimization result of FFT processor implementation in Silterra 0.18 μm and Mimos 0.35 μm technology library.

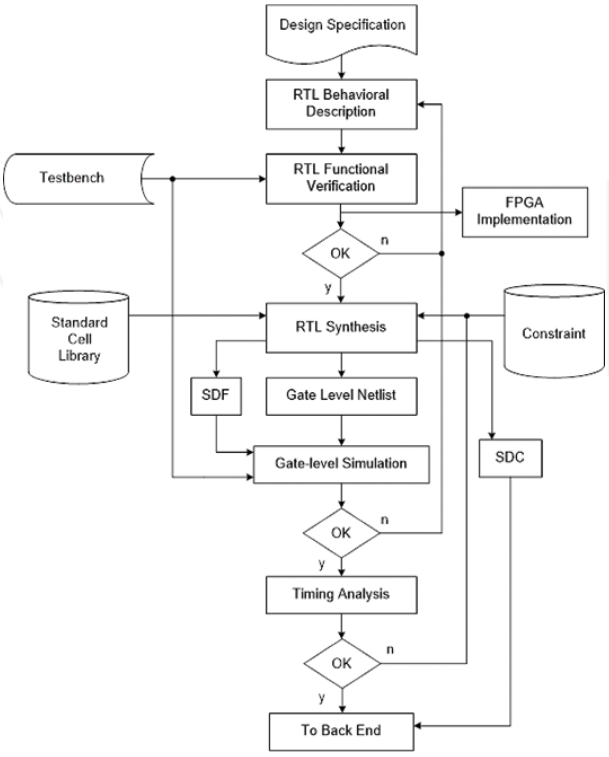

Figure 20. Flowchart of hardware implementation.

HDL synthesis report		Timing summary	
Registers flip-flops	1175	Minimum period (ns)	4.391
Shift registers	43 (6%)	Maximum frequency (MHz)	227.747
LUTs slice	4419 (23%)	Min. input arrival time (ns)	3.788
Logic slice	2584 (13%)	Max. output required time (ns)	6.774
RAM cells	1835 (35%)	Total equivalent gate count	998678
IOs	88 (40%)	Total number of path	220310
Memory usage (MB)	254 (40%)	Total number of destinations	5926
Multiplexers	77		
Tri-states	98		

Table 4. 1024-point FPP-FFT specification.

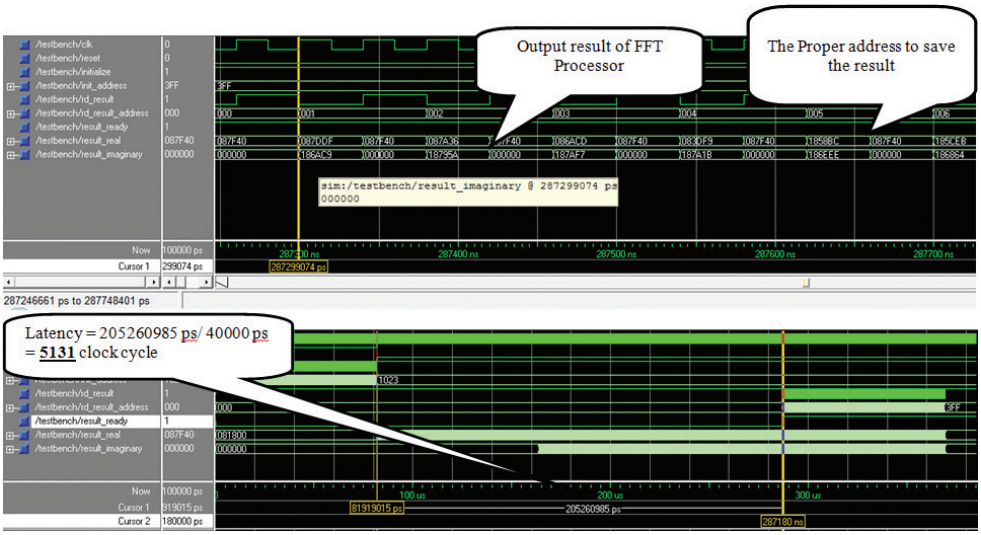

Figure 21. 1024-point FPP-FFT processor output signal.

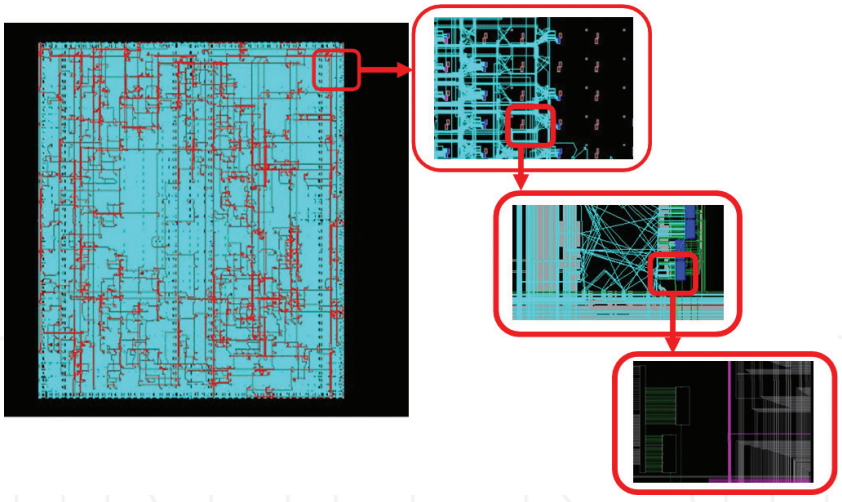

Figure 22. Chip layout of high-tech FFT processor.

FPP-FFT specification	Silterra 0.18 μm technology	Mimos 0.35 μm technology
Active core area (mm^2)	2.32 × 2.32	4.256 × 4.256
Power consumption (mW)	640	1198

Table 5. Optimized power consumption and die area size in different technology library.

http://dx.doi.org/10.5772/66745

To conclude, after FPGA implementation and ASIC optimization and with considering available software and hardware resources, the high-tech 1024-point Radix II FPP-FFT processor was implemented and tested in FPGA prototyping under Xilinx ISE software and CAD tools in synopsis. **Figure 23** shows relevant FPGA board, and **Table 6** summarizes the design property.

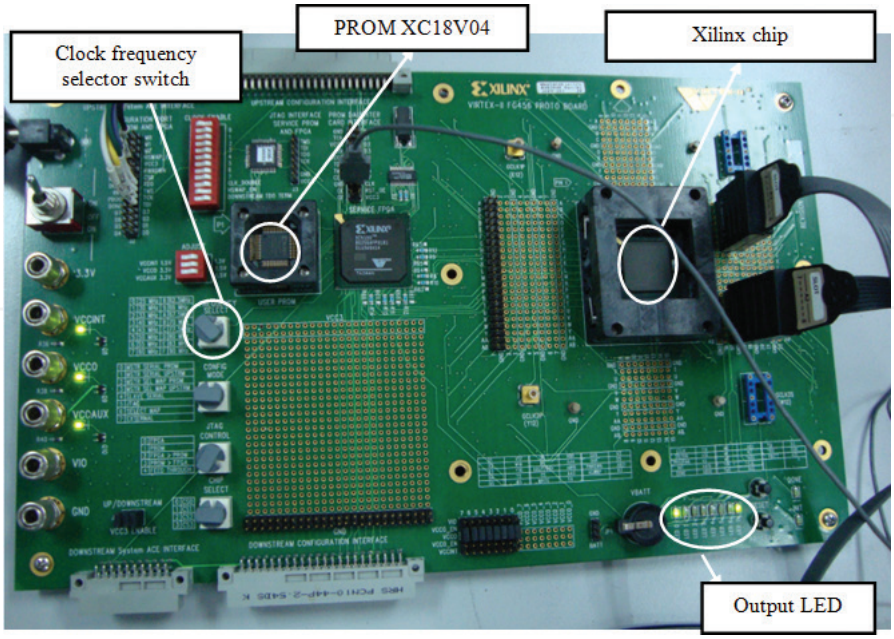

Figure 23. FPGA implementation of high-tech FFT processor.

Parameters	Unit	specification
Processor machine		Radix II
Calculation type		Floating-point
Latency (μs)		22
Maximum precision		32-bit
No. of input data		1024
Data rate (ms/s)		25
Max. clock frequency	$f_{s.max}$	227 MHz
Signal to noise ratio	SNR	192 dB
Power consumption (Silrerra 0.18 μm library)	P_o	640 mW
Active core area (Silrerra 0.18 μm library)	mm	2.32 × 2.32
Accuracy		≤0.01

Table 6. High-tech 1024 point FFT specification.

5. Summary and conclusion

In this chapter, high-tech 1024-point Radix II FFT processor was implemented. The design was launched with introducing 32-bit data single precision floating-point parallel pipeline architecture. Then, it was followed by implementing the subcomponents such as Radix II butterfly and smart controller. The implementation result of high-tech 1024-point Radix II FPP FFT processor was provided accordingly. Designing high speed floating-point arithmetic unit such as adder/subtraction (278 MHz), multiplier (322 MHz), implementing smart controller to save area and increase system efficiency, design processor as single chip by implementing complex dual memory, and providing pipeline and parallel architecture lead to present a high-tech 1024-point Radix II FPP FFT processor. In addition, the processor was synthesized using the Xilinx ISE platform. From synthesis report, it was found that the FPP FFT processor shows the maximum clock frequency of 227 MHz. The latency for calculating 1024-point FFT is 22 μs. After FPGA implementation, the proposed processor was optimized in ASIC under Silterra 0.18 μm and Mimos 0.35 μm technology libraries. The estimation power consumption was reported 640 mW in Silterra and 1.198 W in Mimos technology library with sample rate of 25 ms/s. The procedure was followed by defining the constraints and the netlist (gate level) to produce the ASIC layout. The design compiler result shows the die size of 2.32×2.32 mm^2 in Silterra 0.18 μm technology and 4.256×4.256 mm^2 in Mimos 0.35 μm technology. From the given specification, it was found that the high-tech 1024-point Radix II FPP FFT processor is suitable for high performance DSP application.

Author details

Rozita Teymourzadeh

Address all correspondence to: rozita.teymourzadeh@neonode.com

Neonode Inc., San Jose, California, USA

References

[1] Bergland, G. D. A guided tour of the fast Fourier transform. IEEE Spectrum Conference. 1969; pp. 41–52.

[2] Gold, B., Radar, C. Digital Processing of Signals. New York: McGraw-Hill; 1969.

[3] Smith, J. O. Mathematics of the Discrete Fourier Transform (DFT) with Audio Applications. 2nd ed. W3K Publishing; Stanford, California; 2007. DOI: ISBN 978-0-9745607-4-8

[4] Alegre P. Low Power QDI Asynchronous FFT. 2016 22nd IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC). 2016;978-1-4673-9008-8; pp. 87–88. DOI: http://doi.ieeecomputersociety.org/10.1109/ASYNC.2016.17

[5] Kuo, S. M., Gan, W.-S. Digital Signal Processors, Architecture, Implementations and Applications. Pearson Education International: Prentice Hall; Singapore; 2005. DOI: ISBN: 0131277669

[6] Cooley, J. W., Tukey, J. W. An algorithm for the machine computation of complex Fourier Series. Mathematics of Computation Journal. 1965;**19**:297–301.

[7] Hemmert, K. S., Underwood, K. D. An Analysis of the Double-Precision Floating-point FFT on FPGAs. In: 13th IEEE Symposium on Field-Programmable Custom Computing Machines; 2005. pp. 171–180.

[8] Zheng, S., Yu, D. Design and Implementation of a Parallel Real-Time FFT Processor. In: 7th IEEE Conference on Solid-State and Integrated Circuits Technology; IEEE; 2004. Vol. 3. pp. 65–168.

[9] Thulasiram, R. K., Thulasiraman, P. Performance evaluation of a multithreaded fast Fourier transform algorithm for derivative pricing. Journal of Supercomputing. 2003;**26**(1):43–58.

[10] IEEE Std. 1985. For Binary Floating-Point Arithmetic. IEEE Standard 754-1985. 1985;1–17.

[11] Madeira, P. A Low Power 64-point Bit-Serial FFT Engine for Implantable Biomedical Applications. IEEE 2015 Euromicro Conference on Digital System Design (DSD) (2015). 2015**;** 978-1-4673-8034-8; pp. 383–389. DOI: http://doi.ieeecomputersociety.org/10.1109/DSD.2015.30

[12] Ahmedabad. Implementation of Input Data Buffering and Scheduling Methodology for 8 Parallel MDC FFT. 2015 19th International Symposium on VLSI Design and Test (VDAT) (2015). 2015;978-1-4799-1742-6; pp. 1–6. DOI: http://doi.ieeecomputersociety.org/10.1109/ISVDAT.2015.7208107

[13] Xilinx Logic core. Fast Fourier transform. Xilinx; 2009; Version 7.0.DS260; pp. 1–64.

[14] Ifeachor, E. C., Jervis, B. V. Digital Signal Processing: A Practical Approach. 2nd ed. Prentice Hall; 2002.

INTECH