International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6135
***Article Received**: 24 December 2022 **Revised**: 18 January 2023 **Accepted**: 24 January 2023*

A Brand-New, Area - Efficient Architecture for the FFT Algorithm Designed for Implementation of FPGAs

Gutti Naga Swetha[1], Dr. Anuradha M. Sandi[2]
[1]ECE department, Guru Nanak Dev Engineering College, Bidar, Karnataka, India
Email - nswethag@gmail.com
[2]ECE department, Guru Nanak Dev Engineering College, Bidar, Karnataka, India
Email - anu29975@gmail.com

Abstract:

Elliptic curve cryptography, which is more commonly referred to by its acronym ECC, is widely regarded as one of the most effective new forms of cryptography developed in recent times. This is primarily due to the fact that elliptic curve cryptography utilises excellent performance across a wide range of hardware configurations in addition to having shorter key lengths. A High Throughput Multiplier design was described for Elliptic Cryptographic applications that are dependent on concurrent computations. A Proposed (Carry-Select) Division Architecture is explained and proposed throughout the whole of this work. Because of the carry-select architecture that was discussed in this article, the functionality of the divider has been significantly enhanced. The adder carry chain is reduced in length by this design by a factor of two, however this comes at the expense of additional adders and control. When it comes to designs for high throughput FFT, the total number of butterfly units that are implemented is what determines the amount of space that is needed by an FFT processor. In addition to blocks that may either add or subtract numbers, each butterfly unit also features blocks that can multiply numbers. The size of the region that is covered by these dual mathematical blocks is decided by the bit resolution of the models. When the bit resolution is increased, the area will also increase. The standard FFT approach requires that each stage contain $N/2$ times as many butterfly units as the stage before it. This requirement must be met before moving on to the next stage.

Keywords: ECC, Cryptography, FFT processor, FPGA .

I. INTRODUCTION:

Through the process of encrypting the messages, cryptography not only helps to maintain the messages' integrity but also provides an additional layer of protection for the confidentiality of conversations. It is generally agreed upon that the use of cryptography is essential for the protection of data by software companies, devices that are linked to the internet of things (IoT), and services that need privacy, trust, and integrity. In contrast, however, assaults on electronic devices, user information, and digital transactions have dramatically improved in the recent years, creating a tremendous challenge for those who are responsible for the implementation of security measures. Data security is equally as important to modern electronic devices as having a low power consumption and having a rapid speed performance in that device. A trustworthy cryptographic system is thus very much required as a means of providing protection against the types of attacks and manipulation described above. Nevertheless, the most challenging task is to keep the system's integrity intact while simultaneously guaranteeing that it is always safe to use. Cryptography may either be robust or vulnerable. The effectiveness of a cryptographic system may be evaluated based on the amount of time and resources necessary to decrypt the encrypted data. Strong cryptography results in the generation of cypher text that is extremely challenging to decipher for persons who do not have access to the appropriate decoding instrument. Even with all of the processing power that is available today and all of the time that is available, it is not possible to decrypt the result of using strong cryptography before the end of the universe. This is because strong cryptography is so complex that it cannot be broken even with all of the time and processing power that is currently available. This is the case despite the fact that there is an abundance of time at our disposal. This is true even if a one trillion computers were doing a billion checks each second. The conclusion that can be drawn from this is that robust cryptography should be able to withstand even the most dogged of cryptanalysts with relative ease. PGP's use of robust cryptography, on the other hand, is now the industry standard and the finest that can be found. However, rather than making claims of impenetrability, you would be better off practicing vigilance and conservatism.

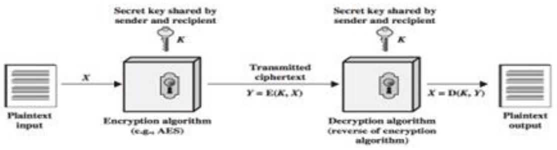
Secret key shared by sender and recipient
K
Secret key shared by sender and recipient
K
X
Transmitted ciphertext
Y = E(K, X)
X = D(K, Y)
Plaintext input
Encryption algorithm (e.g., AES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

Fig1: Conventional Encryption, Expressed as a Simplified Model

There are two prerequisites that must be met before using conventional encryption securely:

We need a robust algorithm for the encryption process. At a bare minimum, we would want the algorithm to be designed in such a way that an adversary who is aware of the procedure and has access to one or more ciphertexts is unable to decrypt the ciphertext or determine the key. This is one of our primary goals for the algorithm. In most situations, the obligation will be presented in a more forceful manner. Even if the adversary is in possession of a number of ciphertexts in addition to the plain text that produces each ciphertext, he or she should not be able to decode ciphertext or find the key. Both the sender and the receiver are responsible for maintaining the key's confidentiality and must have gotten a copy of the secret key in an uncompromising manner. All of the information that is encrypted with this key can be read by anybody who can find the key and figure out the algorithm.

There are two different platforms on which cryptography can be implemented: a software platform or a hardware platform. These two approaches each offer their own set of benefits as well as drawbacks. Both of these options are good ones that should be considered. These two approaches each have some positive aspects to them, but they also each have some drawbacks to consider. Both of these strategies have a number of benefits that may be associated with them, but on the other hand, each of them also has a number of negatives that can be associated with them. Both of these strategies have their merits in their own right. The usage of cryptography on a hardware platform is not only more effective but also results in the processing being done considerably more quickly. The study of cryptography can be broken down into two independent subfields, which are respectively referred to as symmetric cryptography and asymmetric cryptography. The application of cryptography relies heavily on the study of both of these subfields. Elliptic curve cryptography and the Rivest-Shamir-Adleman (RSA) cryptosystem are two examples of asymmetric cryptographic methods that have gained widespread notoriety among all state-of-the-art approaches. Both of these techniques are considered to be state-of-the-art. The use of elliptic curve cryptography and the RSA cryptosystem are two examples of methods that are considered to be cutting edge. Asymmetric cryptography can be approached in a number of different ways, two examples of which are presented here. Both of these instances are examples of state-of-the-art techniques. Cryptography based on elliptic curves is one example of these methods (ECC). When it comes to asymmetric cryptography, the ECC method is much more effective than any other state-of-the-art technique, and this is due to the fact that it has a shorter key length and a high capacity of resolving security concerns. Moreover, the ECC approach has a lower key length. The Elliptic Curve Encryption (ECC) technique has been shown effective for use with both symmetric and asymmetric cryptography standards.

In the most recent few years, one can clearly see a substantial shift occurring in the production of electronic products. In order to keep the cost of manufacturing down and improve communication across appliances, a number of electronic device manufacturers are shrinking the size of their products to levels below the micro and tiny categories. As a result, these sophisticated electronic devices are capable of performing calculations in a shorter amount of time and storing their information [1]. All of these data tell us something about the user's behaviour and the environment they're in. As a result, the information on the user's actions and the surroundings is the most vital piece of data, and it must be guarded with a high level of security. As a result, in this age of digitization, information is the most important item, and maintaining the confidentiality of this information is the most difficult task. As a direct result of this, there has been a discernible increase in the demand for the utilisation of cryptography as a means of guaranteeing the safety of user information, digital transactions, and electronic equipment. Cryptography is a method of encoding and decrypting information using a combination of mathematical algorithms and secret keys. This is a direct result of the fact that those who commit cybercrime are growing more sophisticated all the time. Cryptography is applied to offer security to communications via the process of encrypting the messages, and it also serves to keep the messages' integrity intact. The use of cryptography is widely considered to be essential for software companies, devices connected to the internet of things (IoT), and services that require privacy, trust, and integrity to protect their data. However, in recent years, cyberattacks on electronic devices, user information, and digital transactions have vastly improved, posing a significant challenge for those who are responsible for implementing security measures. Data security is just as crucial to current electronic devices as low power consumption and fast speed performance in that device. To protect against these kinds of assaults and manipulation, a cryptographic system that is reliable is thus quite desired. Nevertheless, the most difficult duty is to preserve the system's integrity while also ensuring that it is constantly secure. Because of this, the Rivest-Shamir-Adleman (RSA)

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6135
***Article Received**: 24 December 2022 **Revised**: 18 January 2023 **Accepted**: 24 January 2023*

cryptosystem and elliptic curve cryptography (ECC) are now the most widely used forms of encryption in the contemporary age [2, 3].

An electronic device that is composed of a large number of different types of components [1] and whose computing performance is determined by parameters such as processing speed, storage, time delay, device area, key lengths, and the amount of power used. Elliptic Curve Cryptography (ECC), on the other hand, has much shorter key lengths and requires much less complexity and power than the Rivest-Shamir-Adleman like RSA cryptosystem, yet it offers the same level of protection as RSA does. RSA, for instance, necessitates a key length of $2048\, bits$ for a certain security action, while ECC only necessitates 233 bits of key length for the identical operation over a binary field [6-8]. [Note: RSA is a more secure algorithm.] In addition to this, Elliptic Curve Cryptography, often known as ECC, offers carry-free operations, which enables hardware implementations. The Elliptic Curve Cryptography (ECC) cryptosystem is more advantageous than the Rivest-Shamir-Adleman (RSA) cryptosystem as a result of the benefits described above. And ECC may be particularly helpful for applications that need significant computational power as well as embedded devices. As a result, this ECC system is applicable to the fields of network security, health-care device operations, and smart-grid operations, all of which place an emphasis on maintaining a high level of security.

Therefore, a large amount of effort has been contributed by a number of scholars in order to create an effective ECC system, and some of the relevant literature is going to be described in the paragraph that is following this one. In this paper, a hard processor architecture for elliptic curve cryptography (ECC) on FPGA over Galois Fields is proposed. This approach promotes flexibility in the body. However, the amount of time needed to complete tasks is significantly increased while employing this approach. A Montgomery multiplier design is suggested for use in Elliptic Curve Cryptography (ECC) in order to improve the efficiency with which flaws may be found. In addition to this, the scalar multiplication model that was developed offers protection against fault assaults as well as duplicating schemes. The Xilinx ISE FPGA serves as the platform for the implementation of this design. In order to offer high speed and decrease space, an efficient Vedic multiplier that can be used for elliptic curve cryptography (ECC) on FPGA architecture has been presented. Xilinx FPGA is being used for the implementation of this synthesis process. However, in the previously described methods, problems with Elliptic Curve Cryptography (ECC) employing point multiplication remain unaddressed. These problems include computational complexity, time delay, processing speed, and excessive power consumption.

For this reason, the purpose of this paper is to propose a novel High Throughput Concurrent Computation (HTCC) technique that makes use of point- multiplier architecture for Elliptic Curve Cryptography (ECC) core on Xilinx Virtex-5 and Xilinx Virtex-7 Field-programmable gate arrays (FPGA) with high efficiency. Since the suggested HTCC point-multiplier architecture is built with low latency and gives more flexibility in comparison to existing hardware devices, it is possible to simply update ECC algorithms on FPGA. Due to the insignificant cost of manufacturing, Field-programmable gate arrays (FPGA) are much more cost-effective than application-specific integrated circuits (ASIC) for sample production in lower numbers. As a result, an HTCC point-multiplier architecture is provided over binary fields $GF(2^n)$ which is then synthesised in FPGA. This is done in order to alter the algorithms of ECC cores by mixing concurrent computing techniques. In this article, a method known as HTCC is presented in order to improve the processing speed as well as the effectiveness of the point-multiplier architecture. The High Throughput Concurrent Computing (HTCC) methodology that has been developed is an example of a concurrent calculation method that can finish its computation in only one clock cycle. As a result, the suggested HTCC method exhibits a level of efficiency that is much greater than that of the different group operations of elliptic curve design. Therefore, the suggested HTCC point-multiplier architecture achieves superior results in terms of the area-time product and requires less time to synthesise than the research efforts that were previously discussed.

II. IMPLEMENTATION

The ECC method may be implemented in a variety of fields, such as over real numbers, prime Galois fields, and binary fields, respectively. Prime Galois fields, on the other hand, function well for software executions, although binary fields perform excellently with hardware implementations. Prime Galois fields also work well for generating prime numbers.

ECC, or Elliptic Curve Cryptography, is a technique for public-key encryption that helps to ensure high levels of security, secrecy, and validity in transmitted data. It is generally agreed upon that ECC is the most trustworthy and effective cryptosystem that can presently be used. In spite of this, there is still opportunity for improvement in the efficacy of the ECC cryptosystem across the whole of its actual implementation.

In addition to calculating a variety of various group operations, the bulk of the researchers have focused the majority of their attention on modifying the finite-field arithmetic model. On the other side, it can lead to an increase in the latency of the group. Taking this into consideration, there is a chance that it may slow down the network. In addition to this, the high-speed multiplier systems are only concerned with a single area of the device at any one time. As a consequence of this, the results of this research offer a design called a High Throughput Multiplier that may be used in elliptic cryptographic applications. This architecture will be able to maintain a high degree of efficiency while performing concurrent computations on $Xilinx\ Virtex - 5\ and\ Xilinx\ Virtex - 7$ Field-programmable gate arrays (FPGA).

III. OBJECTIVE

The goal is to develop new computational architecture with the intention of lowering the number of clock cycles required for computing (latency). Design and construct an architecture for High Throughput Concurrent Computation (HTCC) point multipliers that has low latency and gives better flexibility in comparison to existing hardware devices. This enables ECC algorithms to be readily changed on FPGA. The purpose of this endeavour is to apply in the area of binary for bit lengths of $2, 163, and\ 2, 283\ bits$ with the intention of improving speed, throughput, and efficiency. The last step of the mission is to synthesise utilising $vertex\ 5\ and\ vertex\ 7$, and then evaluate the provided inputs in terms of their area, speed, and efficiency. Develop a full elliptic curve encryption system architecture in the firmware and use FPGA.

1. To design arithmetic handling in ECC Core.
   a) Point Multiplication Architecture.
   b) Point division Architecture.
2. Area efficient realization of the architecture on FPGA
3. Architecture design targeting low latency.
4. Power efficient architecture for FPGA.

FPGAs, which are also known as field programmable gate arrays, are able to achieve higher levels of performance while consuming less power than general purpose processors. This is made possible by the fact that FPGAs are configured specifically for the problem that needs to be solved. Because of the ease with which they can be reconfigured, FPGAs make this kind of thing feasible. Field-programmable gate arrays, abbreviated to FPGAs for short, are a method of implementation that is appropriate for computationally demanding applications such as signal, image, and network processing jobs [1]. This is possible because FPGAs can be altered through programming.

The Fast Fourier Transform, or FFT, is one of the operations that is used most frequently in digital signal processing algorithms [2]. It also plays an important part in a wide variety of signal processing applications, including image processing, speech processing, and software defined radio, amongst others. FFT processors should be of better throughput with decreased calculation time. Because of this, if we want to compute a greater number of data samples, we will need to give some consideration to the size of the FFT processor, since the number of stages required for each FFT calculation rises by a factor that is proportional to N. Maximizing performance while adhering to power dissipation limits may be accomplished with the use of energy-efficient design strategies in the process of designing high throughput FFT designs.

One of the possible high throughput designs is a spatial and parallel FFT architecture, which is also known as array architecture [3]. This architecture is based on the Cooley-Tukey algorithm layout. Despite this, the implementation of the array design requires a significant amount of hardware. Through the use of spatial parallelism, it is able to achieve great performance, despite the increased routing resources this requires. However, when the scale of the problem increases, expanding the architecture physically is not practical owing to major power and area issues brought on by intricate linkages. These issues originate from the fact that the problem requires more space.

When performing FFTs that demand a high data flow, pipelined designs are helpful [4, 5], [6], [7]. The rows are often collapsed as part of the pipelined architecture design philosophy. For the pipeline implementation of the $radix - 2$ FFT method, the $radix - 2$ multi-path delay commutator [8] [9] was arguably the most conventional solution. The addition of memories results in an increase in area, and there is a delay that is directly connected to the amount of memory that is being used [10].

In this study, we suggest an innovative design for an area-efficient FFT by reusing $N/2$ numbers of butterfly units more than once rather than employing $(N/2) \log_2 N$ butterfly units only once [11]. This is accomplished via a time control unit that recycles the butterfly units to finish the FFT calculation after sending back the data that it had previously calculated for $N/2$ butterfly units to itself for $(\log_2 N)$ times. In contrast to the array architecture and pipelined designs, the pipelined architecture has a space need that is just $N/2\ radix - 2$ elements, which is clearly a far lower footprint.

IV. ALGORITHM FOR THE TRADITIONAL FFT

The Cooley-Tukey Fast Fourier Transform (FFT) method is by far the most frequent algorithm used when generating FFT. In order to solve FFT problems of any arbitrary size N, this technique employs a recursive approach. The approach reduces the complexity of the algorithm by dividing the main

FFT into many smaller FFTs, which are then processed separately. This method makes $N = N1.N2$, where $N1$ and $N2$ are the sizes of the smaller FFTs. If the size of the FFT is N, then this algorithm makes $N\ equal\ to\ N$. The most frequent implementation of the Cooley-Tukey technique, known as $radix - 2$ decimationin-time (DIT), may be used for numbers of any arbitrary size N. It is possible to write N as a $power\ of\ 2$, which would be written as $N = 2^M$, where M is an integer. This method is referred to as a decimation-in-time algorithm because at each step, the input sequence is split into smaller sequences, which is another way of saying that the input sequences are decimated at each stage. The fast Fourier transform (FFT) of an N-point discrete-time complex sequence $x(n)$, where $n = 0,1 \ldots\ldots N-1$, is defined as follows:

$$Y(k) = \sum_{n=0}^{N-1} x(n)W_N^{nk}, k = 0,1, \ldots\ldots\ldots\ldots N-1 \qquad (1)$$

where $W_N = e^{-j2\pi/N}$ and the FFT is partitioned into two equal pieces by $radix - 2$. In the first step of this process, the Fourier transform of the even index numbers is calculated. The other component determines the Fourier transform of the odd index numbers, combines the results of this calculation, and then applies it to the full sequence to get the Fourier transform. Separating the $x(n)$ into odd and even indexed values of $x(n)$, we obtain

$$\frac{N}{2}-1 \qquad \frac{N}{2}-1$$

$$Y\ (k) = X\frac{xe(n)WN}{nk2} + WNk\ X\frac{xo(n)WN}{nk2} \qquad (2)$$

$$n = 0 \qquad n = 0$$

PROPOSED FFT ALGORITHM

The overall number of butterfly units used in an FFT processor determines how much space the device takes up. Each butterfly unit has multiplier blocks in addition to blocks that may add or subtract numbers. The bit resolution of the samples determines the size of the area of these two mathematical blocks. The higher the bit resolution, the greater the area. The conventional FFT technique dictates that each stage must have $N/2$ times as many butterfly units as the stage before it. As a result, the total number of butterfly units for a conventional FFT processor may be calculated as

$$BU_{Traditional}FFT = \left(\frac{N}{2}\right) log2\ N \qquad (3)$$

In the approach that has been suggested, there are $N/2$ times as many butterfly units that are reused as there are $\log_2 N$ times. As a result, the revised design of the FFT processor calls for $BU_{Proposed}FFT$ number butterfly units, which can be calculated using the formula:

$$BU_{Proposed}FFT = \frac{N}{2} \qquad (4)$$

The proposed architecture of FFT processor reduces the number of butterfly units by a factor of (α), which is given by

$$\propto = \frac{\frac{N}{2}}{\frac{N}{2}\log_2 N}$$

$$\propto = \log_2 N^{-1}$$

$$\propto = \log_N 2 \qquad (5)$$

Table I shows that the number of multipliers and adders/subtractions for the proposed FFT is less compared to that of the traditional FFT.

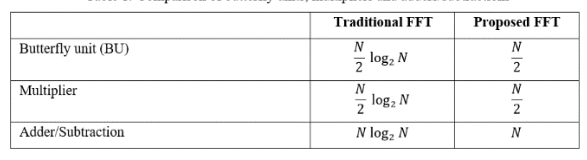

Table 1. Comparison of butterfly units, multipliers and adders/subtractions

	Traditional FFT	Proposed FFT
Butterfly unit (BU)	$\frac{N}{2}\log_2 N$	$\frac{N}{2}$
Multiplier	$\frac{N}{2}\log_2 N$	$\frac{N}{2}$
Adder/Subtraction	$N\log_2 N$	N

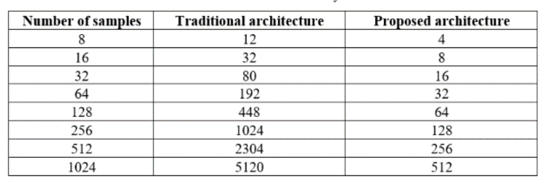

Table 2. Number of butterfly units

Number of samples	Traditional architecture	Proposed architecture
8	12	4
16	32	8
32	80	16
64	192	32
128	448	64
256	1024	128
512	2304	256
1024	5120	512

ARCHITECTURE OF PROPOSED FFT PROCESSOR

The low area is the most important feature of the proposed FFT processor, although it has a number of other important features as well. The architecture that is being considered at this time recycles $\frac{N}{2}$ times the number of butterfly units for a grand total of $\log_2 N$ times. A block diagram of the proposed fast fourier transform (FFT) processor is shown in the following illustration. Figure 3 contains this particular diagram for your perusal. It is made up of several different components, the most notable of which are a control unit, a butterfly unit, and a routing network. An input output enable block is also a part of this package.

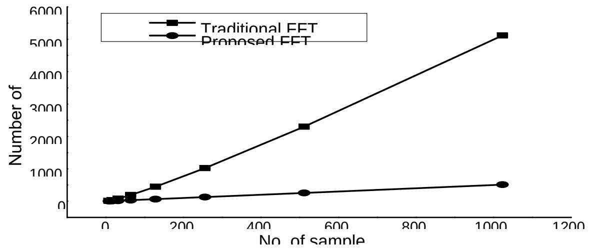

Fig. 2. A comparison of the required number of multipliers for a traditional FFT processor and the one proposed here

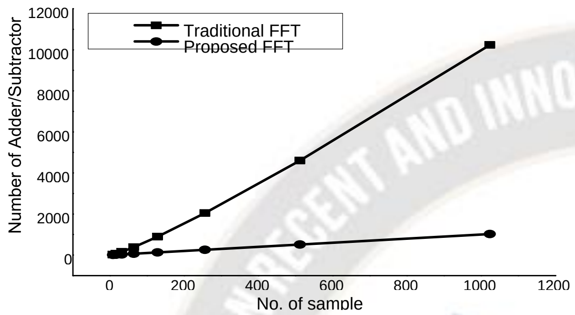

Fig. 3 A comparison of the required number of adders and subtractors blocks in the traditional FFT processor and the proposed FFT processor

In this case, the control unit is what is responsible for synchronizing all of the FFT processor's blocks. It also controls the input, output, and feedback data bus, in addition to counting the number of stages. The control block is responsible for the generation of three signals: the multi-bit stage bus, the one-bit Input Select Line (ISL), and the output select line (OSL) (SB). This stage bus (SB) is where the FFT calculation is at its current stage. The rising edge of the clock signal triggers an increase in the number of stages that are performed by the control unit. After the first stage, set $ISL\ =' \ 0'$ to pick data from an external source. After that, set $ISL\ =' \ 1'$ to select data from the feedback path or register array for $\log_2 N - 1$ times. Finally, set $OSL\ =' \ 0'\ for\ (\log_2 N) - 1$ times to retrieve the output data of the butterfly unit to register array. At the $\log_2 N$th time, $OSL =' \ 1'$ must equal in order to activate the FFT processor's output data route.

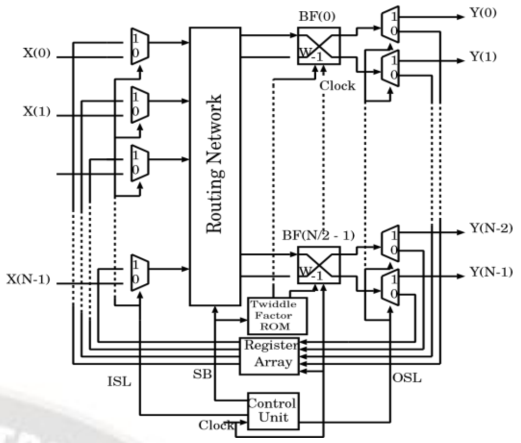

Fig. 4. The design of the FFT processor that was put forward for consideration

Butterfly unit (BU)

According to the mathematical diagram, the output samples of the butterfly unit are produced after an addition and subtraction operation with the product of an even data sample and a twiddle factor. The twiddle factor is the variable that causes the output samples to be unpredictable. The randomness that is introduced into the output samples is a result of a variable known as the twiddle factor. The twiddle factor is a variable that determines how much randomness is introduced into the output samples. This variable is responsible for the introduction of the randomness. You can find an illustration of this idea in the fifth figure of the figure, which is provided for your review. These butterfly-shaped devices are able to keep accurate time, which enables them to function as clocks. The rising edge of the clock signal signifies the beginning of the multiplication process, while the falling edge of the clock signal indicates the beginning of the addition or subtraction operation. As can be seen in Figure 5, the whole of the butterfly unit's function may be completed in the space of a single clock cycle.

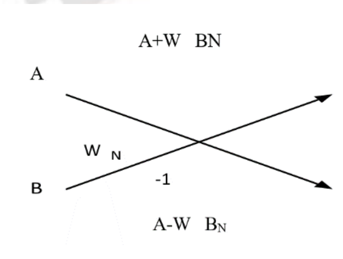

Fig. 5. Architecture of butterfly unit

Twiddle factor ROM

The twiddle factor coefficients are saved in the ROM that the twiddle factor uses. This ROM unit's size may be expressed as $\log_2 NX(\frac{N}{2})$. This block has an output signal count of $\frac{N}{2}$, and

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6135
***Article Received**: 24 December 2022 **Revised**: 18 January 2023 **Accepted**: 24 January 2023*

those $\frac{N}{2}$ output signals are coupled to an equal number of butterfly units. A connection has been made between the stage bus (SB) and the address bus of the ROM.

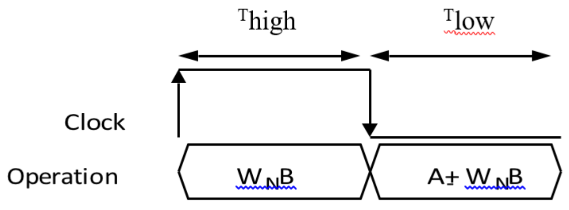

Fig. 6. Timing diagram of butterfly unit

The correct sequences of input samples are sent by the routing network unit to the butterfly units so that the various stages of calculations may take place. The significance of the stage bus (SB) value determines the output of this unit. At the beginning of the process, the routing network produces a sample sequence that is bit-reversed based on the input samples. In the next phases, the routing network will reorder the feedback samples such that they are separated by a distance of 2^{m-1}, where m is defined as m=2, 3,... $\log_2 N$. Figure.7 illustrates the data route configuration for an 8-point fast Fourier transform. The feedback samples are denoted by the dashed arrows. Arrows that cross one other represent the butterfly units. Register array [12] is responsible for storing the data from the butterfly units' last iteration and for transmitting that data at the rising edge of the clock.

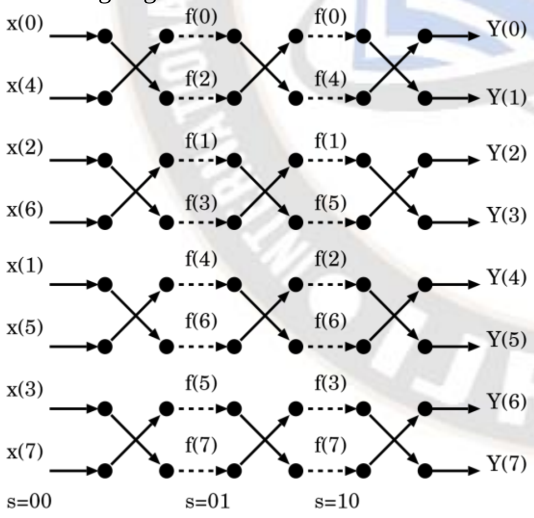

Fig. 7.Data path layout of 8-point FFT processor

DEPLOYMENT & OUTCOMES:

In accordance with the suggested FFT processor, the architecture of an 8-point FFT processor is shown in Figure 8. The timing diagram for this CPU may be seen in figure 9. $X(n)$ *and* $Y(K)$ stand for input and output samples, respectively, while f(k) represents the output samples from the stage that came before. VHDL is used for coding the architecture of the proposed 8-point FFT processor, and $Xilinx$ is used for emulating and synthesising the design.

The architecture of an 8-point FFT processor is presented graphically in Figure 8, which is consistent with the FFT processor that was suggested earlier. Figure 9 illustrates the CPU timing diagram while also providing a graphical representation of the data. Figure 9 also provides some context. The samples that are denoted by $X(n) and\ Y(K)$respectively, represent the input and output in this notation. The samples that are denoted by f(k), on the other hand, represent the output samples that were generated by the stage that came before it. Emulating and synthesising the design are both handled by $Xilinx$, while VHDL is the language of choice for coding the architecture of the proposed 8-point FFT processor. $Xilinx$ is the tool that should be used for either of these two tasks.

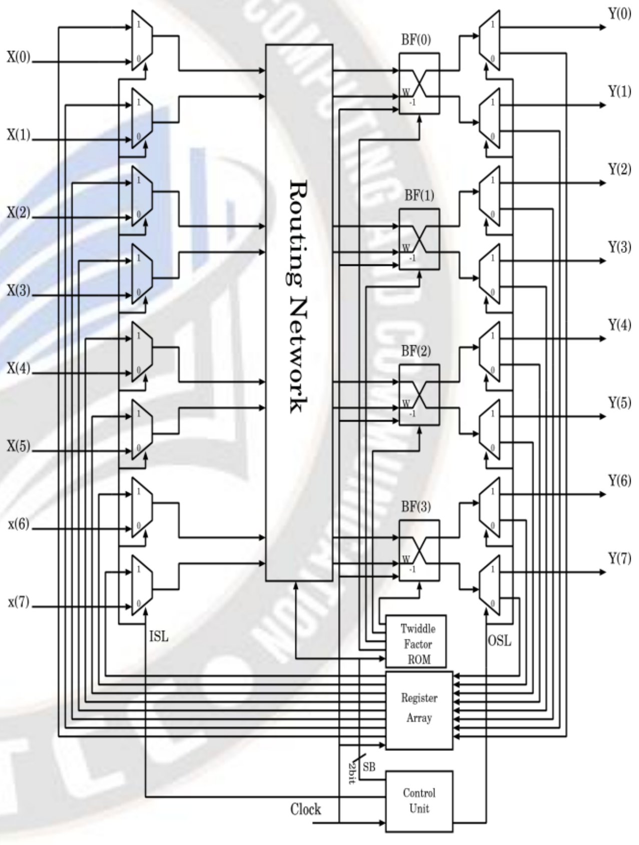

Fig 8. Architecture of an 8-point FFT processor that has been proposed

Virtex-6 support added to ISE 14.2 for FPGAs. The results of a comparison between advanced HDL synthesis reports and conventional FFT are shown in Table 3. Figure.10 shows the created comprehensive RTL diagram of the 8-point FFT processor that was suggested. Figure.9 and Table 4 provide a comparison of the needs for the number of DSP slices and LUTs, respectively, while Figure.9 presents a comparison of the timing delay with the conventional FFT processor.

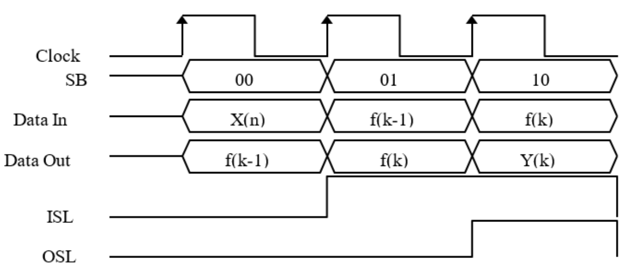

Fig. 8. Timing Diagram of an 8-point Fast Fourier Transform

Table 3. Comparison of ADVANCED HDL Synthesis reports

Hardware	Traditional FFT	Proposed FFT
MACs	24	8
Multipliers	24	8
Adder/Subtractions	72	25
Multiplexers	360	136
XORs	24	8
Registers	–	288
Counter	–	1

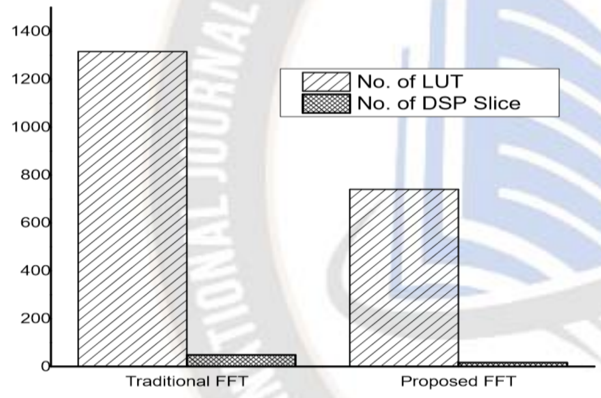

Fig. 9. A comparison of the number of LUTs and DSP slices between the conventional FFT processor and the suggested one

Table 4. A comparison of the delays caused by the traditional FFT processor and the proposed one

Algorithm	Delay (nsec)
Traditional FFT	29.111
Proposed FFT	29.397

Table 5.Device Utilization and Timing Summary

Device Utilization Summary	
Selected Device	6vsx475tff1759-2
Number of Slice Registers	301 out of 595200
Number of Slice LUTs	748 out of 297600
Number of DSP48E1s	16 out of 2016
Timing Summary	
Minimum period	19.598ns
Maximum Frequency	51.025MHz
Minimum input arrival time before clock	9.384ns
Maximum output required time after clock	0.665ns

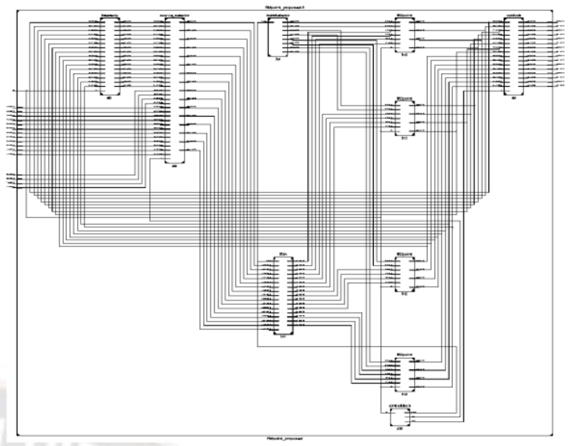

Fig. 10. Specifications of the RTL diagram for an 8-point fast Fourier transform processor

V. CONCLUTION

The architecture that was built demonstrates a Radix2 FFT processor that utilises its space in an effective and efficient manner. The fact that the butterfly units from one stage are reused several times by the algorithm leads to a considerable decrease in the amount of space required for the overall pattern. Both the emulation of the architecture and a performance analysis of the operation of the FPGA's operation in terms of overall response time and consumption of hardware resources have both been carried out. An in-depth investigation reveals that the proposed layout greatly reduces the space while yet preserving the same response time. This is the result of the reduction in reaction time. Both the design of the silicon layout and an examination of the performance trade-off that takes place after the layout have the potential to be explored further in the development process.

REFERENCES

[1]. C. A. Lara-Nino, A. Diaz-Perez and M. Morales-Sandoval, “Elliptic Curve Lightweight Cryptography: A Survey,” in IEEE Access, vol. 6, pp. 72514-72550, 2018, doi: 10.1109/ACCESS.2018.2881444.

[2]. Rivest RL, Shamir A, Adleman L. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. Commun. ACM. Feb. 1978; 21(2): 120-126. https://doi.org/10.1145/359340.359342

3. Kong Y, Asif S, Khan Mohammad AU. Modular multiplication using the core function in the residue number system. AAECC. Springer Berlin Heidelberg. Jan. 2015; 27(1): 1±16.

[3]. N. Koblitz, “Elliptic curve cryptosystems,” Math. Comput., vol. 48, no. 177, pp. 203–209, 1987.

[4]. V. S. Miller, “Use of elliptic curves in cryptography,” in Proc. Conf. Theory Appl. Cryptograph. Techn. Santa Barbara, CA, USA: Springer, 1985, pp. 417–426.

[5]. Hankerson Darrel, Menezes Alfred J, Vanstone Scott. Guide to Elliptic Curve Cryptography. Springer-Verlag New York, Inc. Jan. 2003.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6135
***Article Received**: 24 December 2022 **Revised**: 18 January 2023 **Accepted**: 24 January 2023*

[6]. IEEE Standard Specifications for Public-Key Cryptography. IEEE Standard 1363-2000. Aug. 2000; pp:1±228.
[7]. NIST- National Institute of Standards and Technology, Digital Signature Standard. FIPS Publication 186-2. 2000.
[8]. J. W. Bos, J. A. Halderman, N. Heninger, J. Moore, M. Naehrig, and E. Wustrow, “Elliptic curve cryptography in practice,” in Proc. Int. Conf. Financial Cryptogr. Data Secur. Christ Church, Barbados: Springer, 2014, pp. 157–175.
[9]. G. M. de Dormale and J.-J. Quisquater, “High-speed hardware implementations of elliptic curve cryptography: A survey,” J. Syst. Archit., vol. 53, nos. 2–3, pp. 72–84, 2007.
[10]. H. Asshidiq, A. Sasongko and Y. Kurniawan, “Implementation of ECC on Reconfigurable FPGA Using Hard Processor System,” 2018 International Symposium on Electronics and Smart Devices (ISESD), Bandung, 2018, pp. 1-6, doi: 10.1109/ISESD.2018.8605444.
[11]. M. Bedoui, B. Bouallegue, B. Hamdi and M. Machhout, “An Efficient Fault Detection Method for Elliptic Curve Scalar Multiplication Montgomery Algorithm,” 2019 IEEE International Conference on Design & Test of Integrated Micro & Nano-Systems (DTS), Gammarth-Tunis, Tunisia, 2019, pp. 1-5, doi: 10.1109/DTSS.2019.8914743.
[12]. P. Ahuja, H. Soni and K. Bhavsar, “High Performance Vedic Approach for Data Security Using Elliptic Curve Cryptography on FPGA,” 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, 2018, pp. 187-192, doi: 10.1109/ICOEI.2018.8553721
[13]. V. Maruthi Prasad, B. Bharathi, “A Novel Trust Negotiation Protocol for Analysing and Approving IoT Edge Computing Devices Using Machine Learning Algorithm”, International Journal of Computer Networks and Applications (IJCNA), 9(6),PP: 712-723, 2022, DOI: 10.22247/ijcna/2022/217704.
[14]. M. M. Islam, M. S. Hossain, M. Shahjalal, M. K. Hasan and Y. M. Jang, “Area-Time Efficient Hardware Implementation of Modular Multiplication for Elliptic Curve Cryptography,” in IEEE Access, vol. 8, pp. 73898-73906, 2020, doi: 10.1109/ACCESS.2020.2988379.
[15]. Rashid, Muhammad & Imran, Malik & Jafri, Atif & Kashif, Muhammad. (2019). A Throughput/Area Optimized Pipelined Architecture for Elliptic Curve Crypto Processor. IET Computers & Digital Techniques. 13. 10.1049/iet-cdt.2018.5056.
[16]. M. M. Islam, M. S. Hossain, M. K. Hasan, M. Shahjalal and Y. M. Jang, “FPGA Implementation of High-Speed Area-Efficient Processor for Elliptic Curve Point Multiplication Over Prime Field,” in IEEE Access, vol. 7, pp. 178811-178826, 2019, doi: 10.1109/ACCESS.2019.2958491.
[17]. H. N. Almajed and A. S. Almogren, “SE-Enc: A Secure and Efficient Encoding Scheme Using Elliptic Curve Cryptography,” in IEEE Access, vol. 7, pp. 175865-175878, 2019, doi: 10.1109/ACCESS.2019.2957943.
[18]. R. Salarifard and S. Bayat-Sarmadi, “An Efficient Low-Latency Point-Multiplication Over Curve25519,” in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 66, no. 10, pp. 3854-3862, Oct. 2019, doi: 10.1109/TCSI.2019.2914247.
[19]. P. Choi, M. Lee, J. Kim and D. K. Kim, “Low-Complexity Elliptic Curve Cryptography Processor Based on Configurable Partial Modular Reduction Over NIST Prime Fields,” in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 65, no. 11, pp. 1703-1707, Nov. 2018, doi: 10.1109/TCSII.2017.2756680.
[20]. Orlando G and Paar C. A High-Performance Reconfigurable Elliptic Curve Processor for GF(2m). In:Proc. CHESS. 2000. pp: 41±56.
[21]. D. F. P. Gallagher and C. Director, “FIPS PUB 186-3 federal information processing standards publication digital signature standard (DSS),” Federal Inf. Process. Standards Publication, 2009.
[22]. C. Rebeiro, S. S. Roy, and D. Mukhopadhyay, “Pushing the limits of high-speed GF(2m) elliptic curve scalar multiplication on FPGAs,” in Proc. Int. Workshop CHES, 2012, pp. 494–511.
[23]. R. Azarderakhsh and A. Reyhani-Masoleh, “Efficient FPGA implementations of point multiplication on binary Edwards and generalized Hessian curves using Gaussian normal basis,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 20, no. 8, pp. 1453–1466, Aug. 2012.
[24]. S. S. Roy, C. Rebeiro, and D. Mukhopadhyay, “Theoretical modelling of elliptic curve scalar multiplier on LUT-based FPGAS for area and speed,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 21, no. 5, pp. 901–909, May 2013.
[25]. G. D. Sutter, J. Deschamps, and J. L. Imaña, “Efficient elliptic curve point multiplication using digit-serial binary field operations,” IEEE Trans. Ind. Electron., vol. 60, no. 1, pp. 217–225, Jan. 2013.
[26]. K. C. C. Loi, S. An, and S.-B. Ko, “FPGA implementation of low latency scalable elliptic curve cryptosystem processor in GF(2m),” in Proc. ISCAS, Jun. 2014, pp. 822–825.
[27]. Z.-U.-A. Khan and M. Benaissa, “Throughput/area-efficient ECC processor using montgomery point multiplication on FPGA,” IEEE Trans. Circuits Syst. II, Express Briefs, vol. 62, no. 11, pp. 1078–1082, Nov. 2016.
[28]. L. Li and S. Li, “High-performance pipelined architecture of elliptic curve scalar multiplication over GF(2m),” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 24, no. 4, pp. 1223–1232, Apr. 2016.
[29]. T. T. Nguyen and H. Lee, “Efficient algorithm and architecture for elliptic curve cryptographic processor,” J. Semicond. Technol. Sci., vol. 16, no. 1, pp. 118–125, 2016.
[30]. B. Rashidi, R. R. Farashahi, and S. M. Sayedi, “High-speed hardware architecture of scalar multiplication for binary elliptic curve cryptosystems,” in Proc. IEEE Conf. Inf. Secur. Cryptol. (ISCISC), Sep. 2014, pp. 15–20.
[31]. Z. U. Khan and M. Benaissa, “High-speed and low-latency ECC processor implementation over GF(2m) on FPGA,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 1, pp. 165–176, Jan. 2017.
[32]. R. Salarifard, S. Bayat-Sarmadi and H. Mosanaei-Boorani, "A Low-Latency and Low-Complexity Point-Multiplication in ECC," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 65, no. 9, pp. 2869-2877, Sept. 2018, doi: 10.1109/TCSI.2018.2801118.