ISSN: 2395-1680 (ONLINE)
DOI: 10.21917/ijme.2026.0394

SPFAS: DESIGN OF SEMI PARALLEL FOLDED ADD-SHIFT BASED DISCRETE TCHEBICHEF ARCHITECTURE WITH REDUCED COMPONENTS

M. Kiruba[1], V. Sumathy[2] and K.N. Vijeyakumar[3]
[1]*Department of Information Technology, SNS College of Technology, India*
[2]*Department of Electronics and Communication Engineering, Government College of Engineering, Dharmapuri, India*
[3]*Department of Electronics and Communication Engineering, Dr. Mahalingam College of Engineering and Technology, India*

Abstract

FP-Floating point arithmetic unit is the widely used application in Digital Signal Processing (DSP), which is an alternative of the logarithmic number system. When compared to the square root operations, the addition, subtraction, multiplication and division are more challenging tasks. In accordance with this reason, few structures were developed in the existing literatures concerning the DSP. But, it has some drawbacks which includes increased memory usage, delay and power consumption. To overwhelm these issues, a Semi Parallel Folded Add-Shift (SPFAS) architecture is carefully designed by making use of minimized components is introduced. Here, the folded Discrete Tchebichef Transformation (DTT) multiplier is integrated with SPFAS forming the novelty in the circuit used in this work in order to find the output register. Moreover, there is no need to add any separate block for performing subtraction operation because of the integration undertaken. The main advantage of this concept is that it reduces the size of Digital Signal Processing (DSP) unit with minimum usage of logical components and registers. Also, the structure of shifter and adder are changed to reduce the accumulator size, Flip Flop (FF) and Look Up Table (LUT). In experiments, the performance of the proposed SPFAS with DTT architecture is evaluated and compared in terms of delay, area, latency, cycles, RAM, LUTs, slice and FFs. On investigating, we could infer that this devised architecture design need only lesser usage components of 5979 instead of 8702.

Keywords:

SPFAS-Semi Parallel Folded Add-Shift architecture, DTT-Discrete Tchebichef Transformation, FPPE-Floating Point Processing Element, DSP-Digital Signal Processing, and FFs-Flip Flops

1. INTRODUCTION

Multimedia Utilizations and processing of digital signal need huge amount of information, more computational power and processing ability. Floating Point Processing Element (FPPE) is an important unit for addition, subtraction and multiplication of binary data [1-3]. In arithmetic calculation, most of the system struggles in the floating point number estimation due to its exponential terms. It is applicable in most of the Digital Processing Unit (DPU) for calculating the Milli or Micro scale value. It analyzes the clear range of variation in the applications of signal processes such as, Bio-medical signal and speech signal analysis. Majority of trades, entertainment cum technical utilizations deployed the floating point processes and computation logics in accordance with the consistent advancements in VLSI technology. Here, the architecture of FPU-Floating Point Unit is implemented in different ways, which improves the overall performance by reducing area and power consumption. Moreover, the floating point numbers are widely used in different fields due to its great dynamic range, easy process protocols and optimal rate of precision. The floating point is defined as a method that represents an approximation to real number based on the fixed number of bits.

Discrete Tchebichef Transform (DTT) is broadly utilized transformation algorithm in field of image processing but not suitable for image watermarking. DTT is transform method which utilize Techebichef polynomial, has better energy compaction properties similar to DCT and function good for few class of image. Because of its higher energy compaction DTT is widely in application of image processing like image feature extraction and compression. Additionally, DDT has capacity to examine algebraic expression, whereas DCT wants trigonometric function for computation. DDT's algebraic form won't allow for recursive minimization of polynomial order as in DCT. DTT has high energy compactness and also some other properties similar to DCT like orthogonally, symmetry and separability. Additionally, DTT had potential of minimizing used memory while executing image processing program.

In VLSI design architecture, many applications suffer in size of components usage in logical operations. In the Digital signal Processing (DSP) system, there may be some errors can be occurred in calculating the transform values such as DWT [4, 5], DCT [6] and FFT [7, 8]. To solve this problem, some of the architecture designs are introduced in the existing works. But, it increases the difficulties in designing it along with delay in time. The main aim of designing a floating point architecture [9-11] is to reduce the power loss and area. For this purpose, the FPPE model is employed to predict the derivation and integration operation of the given signal. In FPPE model, different techniques are developed to estimate the mantissa value and exponential term with minimum usage of logical components. However, it has some drawbacks such as increased memory usage and delay.

1.1 CONTRIBUTIONS

To solve these problems, this work focused to design Semi Parallel Folded Add-Shift (SPFAS) architecture with reduced delay and area consumption under the benefit that this proposed methodology gives with the novel integration of Folded DTT.

In this architecture, the processing element is implemented in the DSP processing unit. The proposed architecture reduces the size of DSP unit with minimum usage of logical components and the amount of registers.

The structure of shifter and adder are changed to reduce the accumulator size, number of Flip Flops (FFs) and LUTs. At each condition, there is one register that produces the logical output by holding the pair of blocks in every phase. Bit sequence cross overs from source to the destination lessens the delay block because of development of the links in the considered register. It reduced the usage of components to reduce the power consumption.

Remainder of the manuscript are categorized into the following sections: Section 2 presents some of the existing works related to floating point arithmetic unit design and implementation. Section 3 depicts regarding the proposed Semi Parallel Folded Add-Shift (SPFAS) architecture in an expanded manner. Section 4 gives the investigation made between the current and devised methodologies. At last, Section 5 will be giving our conclusion inferred out of this work along with the futuristic views extending the devised work.

2. RELATED WORKS

This section poses few earlier literatures with reference to its structure and design of floating point arithmetic unit in VLSI. Lalita Gangwar and Rajan Choudhary [12] designed a floating dependent point methodology comprising augmented varying span with the benefit of not requiring the scaling operation. This become more fond for sophisticated methodologies. Arithmetic functions on floating point numbers consist of adding, subtracting, multiplying and dividing operations. The fraction part of the sum is the sum of fractions, and the exponent part of the sum is remain same. It states that F1 and F2 are properly normalized, if they are not normalized than first normalization, the fraction part of sum. In Float 32 Add is a block name that is provided for free in the form of Verilog code. The code is difficult to read because of removed text form adders and identifiers replaced with automatically generated strings. It can be simulate it or synthesize it to FPGA or ASIC technology. Modifications of the code are extremely difficult. The floating-point adder gets 3 sources like 2 floating point operands along with one maintaining field. The control field is 5 bits long. In this designed the single-precision multiplier for floating-point numbers. Use 23 bit fractions and 9 bit exponents, with negative numbers represented in 2's complement.

Sohn and Swartzlander [13] designed a fused floating-point 4 element dot unit for improvising the area and consumption made in terms of the power. The aim of the work was to lessen the latency by handling the cases (positive and negative) of the significant sum. The normalization process was applied to reduce the significant addition by determining the sign bit. Moreover, the Leading Zero Anticipation (LZA) was applied to predict the normalization shift based on encoding. The major advantage of this method was, it simultaneously performed the significant addition and subtraction, but it was computationally more complex. Coleman and Ismail [14] described a co-transformation procedure to eliminate the ROM space to allow easy synthesis of the remainder. The main drawback of this method was, it required few percent of additive operations to necessitate the re-use of interpolator.

Rosa, et al [15] suggested an out of the box methodology for improving the operation of transforming methodologies. This makes the transformation to the fixed-point from the floating-point type. The major objective of the devised work was to solve the resolution concerning the bit-length estimation problem. Here, the significant amount of time required to estimate the bit length of Evolutionary Optimize (EO) was saved. This algorithm avoided the maximum error that leads to variable underflow and overflow. In this paper, the variables like power usage along with the size were considered for validating their devised methods' outcomes.

Joldes, et al [16] recommended some arithmetic algorithms based on the floating point expansions for extended precision. The main objective of this work was to renormalize the FP expansions by ensuring certain precision related requirements. Moreover, a clear investigation about error analysis and effective error bounds were provided in this paper. Another objective of this concept was to renormalize the FP expansions, which was fast in practice. The Error Free Transforms (EFT) was utilized to estimate both the result and rounding error. The major disadvantages of this paper were high memory consumption and area consumption. Jaiswal, et al [17] introduced two different architectures such as QPdDP and QPdDPqSP based on the fully pipelined format for the multi modal floating pointed adding operators. This paper focused on the design of configurable multi-mode floating arithmetic architecture design. The main objective of this work was to provide high precision computational aid towards the estimations with lesser precision. The advantage of this paper was, the architectures provided entire estimating aid to both ordinary and abnormal operators that deals with several iterations efficiently. The drawback that this work came across was in utilizing the FMA-Fused Multiplier Adder for designing the arithmetic unit. Canto, et al [18] designed a Vector Floating-Point Unit (VFPU) for the flexible biometric online speaker verification. This architecture contains two different phases such as, enrollment and classification. The system was trained with various utterances provided by the user in the enrollment phase. The features of any individual from the genuine user were extracted in the classification or matching phase. The success of these models was based on the estimation of parameters and the likelihood of the data, which were the drawbacks of this paper. Wang, et al [19] considered a generic logic level architecture to verify the particular architecture of End-Around-Carry (EAC) adder. The suggested architecture contains a kernel and auxiliary component, which executes the functionality of unsigned addition. Moreover, it was fully based on the component and parameters of customization, which was the main limitation of this paper. It identified and verified some basic adders based on the factors of scalability, modularization and fidelity.

Rajapaksha, et al [6] introduced a VLSI computational architecture to address the following problems:

- To obtain the mean value of a given method based on preposition
- To design an efficient architecture for calculating the eight-point DCT

Here, the low complexity architectures were employed by considering the non-uniformly sampled inputs. The registers were inserted at the internal nodes that lead the low critical path delay. Swartzlander and Saleh [20] described two fused floating point operations to implement the FFT architecture. It utilized the butterfly operations by considering the addition, multiplication and subtraction of complex valued data. For this purpose, the radix 2 and radix 4 butterflies were implemented in this paper that efficiently performed the floating point operations. In experiments, it was observed that the obtained results from this paper were slightly accurate, but it need to use fewer rounding operations. Dorrance, et al [21] determined the use of FPGA dependent dispersed matrix vector multiplying kernel for using

the allocated computing sources and memory bandwidth in an efficient manner. Here, a separate additional processor was utilized to balance the system specific sources. Moreover, the available memory bandwidth was matched by scaling the number of Processing Elements (PEs). The major advantage of this paper was, it transformed the non-abnormal arbitrary storage authentication to a more stabilized and normal serial storage authentication. But, the drawback of this concept was, it required more number of resources to design the architecture.

Zhang, et al [22] analyzed the computing memory bandwidth and throughput of CNN-Convolutional Neural Network design on an FPGA platform. In this paper, the basics of CNN, roofline model and accelerator design model are investigated. Here, the data sharing relations between different loop iterations were classified into three categories, which include: irrelevant, independent and dependent.

Also, the loop pipelining based optimization technique was used to improve the system throughput by executing different loop iterations. Jovanovic and Milutinovic [23] designed a floating point matrix multiplication method for minimizing the resource utilization and maximizing the clock frequency. The suggested architecture was simple that utilized the block matrix multiplication algorithm. Here, the FPU was implemented in a much improvised manner comprising the 8 FPGA 25 × 18 integer multiplying operating blocks that were embedded. It allowed the realization of large number of processing elements. Also, the multipliers and adders were pipe-lined in nature and utilized various FPGA methods for minimizing the required area and to improve high clock frequency. The limitations of this paper were augmented storage usage and complexities in the computing operations. Purohit, et al [24] used individualistic precision specific FPPE-Floating Point Processing Element to evaluate the huge fulfilment processing components in the systems that were enabling the post configuration of the settings in it. The fulfilment of it was influenced by: Individual processing cell and Top-level array and interconnection scheme.

The proposed data path performs the adding, subtracting, multiplying and accumulating processes. Matam and Hsieh [25] evaluated an energy efficient FP matrix multiplication method on the latest FPGAs. In this paper, an optimum structure was designed for computing the maximum energy efficiency contributed by the multiplying the considered matrixes. The beneficial outcomes got from this work were:

- The parameters such as problem size and type of memory were utilized for parameterized floating point matrix multiplication.
- The energy efficiency of matrix multiplication was implemented on a given target.
- The operation of the entire design was coordinated with the help of control unit.

The major limitations of this paper were, long access latency and periodic references were required to retain. Rosa, et al [26] solved the floating-point to fixed-point bit length problem by using an evolutionary approach multiple-objective compact genetic algorithm. The major objective defined by this work was to reduce the energy consumption and to improve the total performance. Moreover, the operation of transforming methodologies facilitated the acceleration of transformation to the fixed point from the taken FP. Wang, et al [27] introduced a reconfiguring capable pipe-lined processor for implementing the varied length individual precision FPU. The major objective of the devised work was to lessen both the power usage and hardware overhead. Also, RR4BF-Reconfigurable Radix-Four Butterfly architecture was designed for reducing \frac{3}{4} of adding operators, when compared to the traditional parallel radix-4 butterfly architecture. Manolopoulos, et al [28] designed an efficient multiple precision floating-point multiply-add fused unit to execute a quadruple precision. This architecture reduced the latency of Floating-point Add (FADD) instruction by utilizing the minimum number of operating components. However, it required more number of components to perform the arithmetic operations, which was the major limitation of this architecture design. Rony Antony and Joseph [29] implemented a Double Precision Floating Point for efficient floating point comparison. Here, the decision modules utilized 2 OR operators to obtain the best outcome. It compared the optimum bit along with the less important bit at the instant at which the correlated bits were alike. The main advantage of this paper was, it represented a large range of values and it includes the gains of parallel prefix tree structure. Moreover, it consumed large amount of memory and time for performing the floating point operation. Thiruvenkadam, et al [30] designed a quadruple precision FP adding operator supporting 4 separate parallel precision process. Thus utilized increased computing operations incurring reduced source area exchanging intermediary to the several precision enabling operators. Moreover, the multiple modal quadruple precision adding operator aids the ordinary and abnormal processes. Peesapati, et al [31] introduced the fixed point and FP difference based evolution methodology to provide chip based solution. Here, an IIR-Infinite Impulse Response dependent regularity detection operation was applied on PSoC-Programmable System-on-Chip based on the fixed point and FP DE-Differential Evolution. The advantages and disadvantage of this paper were that it improved both the accelerating circumstance and resource usage increasing the complexity of the issue.

3. FPPE INTEGRATED FOLDED DTT DESIGN ARCHITECTURE

This section presents the detailed description of the proposed Semi Parallel Folded Add-Shift (SPFAS) architecture of the parallel butterfly configuration. The main aim of this paper is to design FPPE architecture with minimum logical components and number of registers. For this purpose, the SPFAS architecture is implemented in the Arithmetic Logical Unit (ALU) processing unit, which efficiently reduces the size of DSP. This SPFAS is integrated with Folded DTT to make the optimization of delay, power, and area usage to make this work a novel and improvised one by eliminating the need for the separate block for performing subtraction by deploying the adder block with same XOR operation. This entire flow of proposed system is indicated in the below Fig.1. In the architecture, used registers are chosen as input for doing arithmetic logic operation. Input register's values are multiplied by doing multiplication process and the delay time is because of complication of multiplier block.

Instead of multiplier block addition operation is used to minimize operational complexity. Thus, block use input register

to do arithmetic operation which is examined to count components involved within the process. By using adder operation, count of selection line get increased respect to input registers. Here, the selected lines were controlled to verify needed delay time for multiplication/arithmetic process. If there occurs delay time for particular selected line which is matched with the predefined time, then registers which involves in particular selected line were merged for further process. Otherwise, operational phase is moved to next in way to determine reduced delay time. In this manner, delay time needed for process is examined till it attains minimized components along with minimized delay. Finally, output registers is created as floating point processing element by merging register along with matched delay.

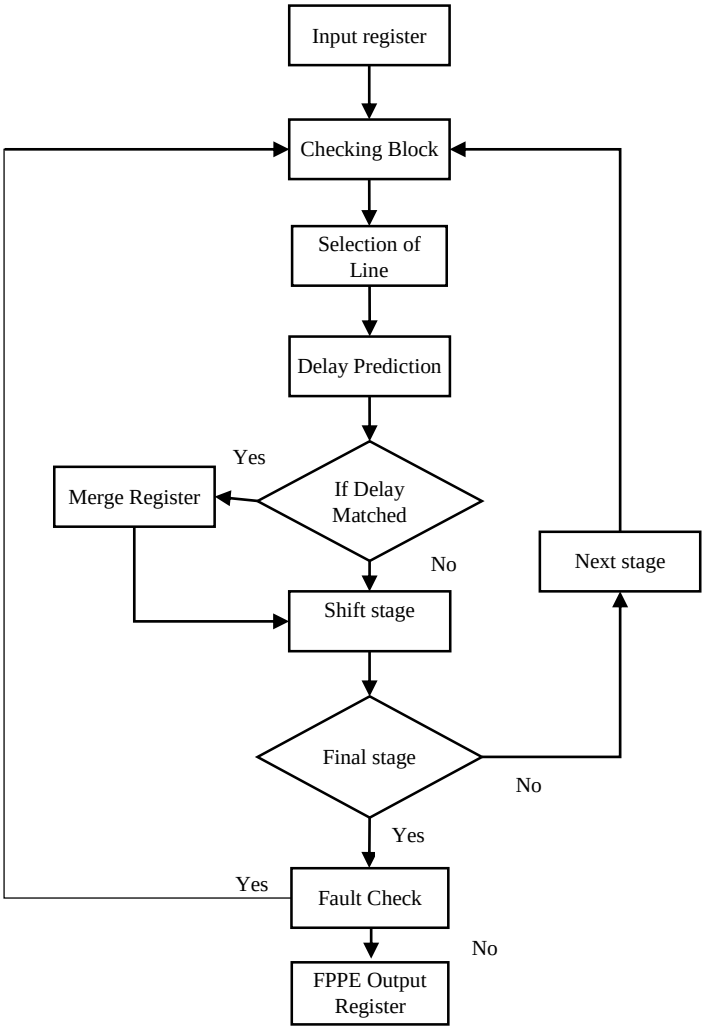


Fig.1. Proposed system flow

This FPPE is developed as final phase to acquire arithmetic value along with minimized component and delay. Suppose if output of FPPE is not minimized value then arithmetic operation was processed by doing next phase of block element validation. While doing division operation, separate subtraction block is needed to add results after completing division operation in conventional architecture.

For multiplication operation, with support of XOR, separate block for the subtraction is not needed in proposed SPFAS architecture. While predicting delay, count of used registers were computed by using Discrete Tchebichef Transform. Thus, proposed model use folded kind multiplier along with Tchebichef filter and it's installed to effectively minimize the components. Calculating Static and Dynamic Power to lessen the leaks:

3.1.1 Addition/Subtraction:

The addition and subtraction of two floating point numbers are as follows:

- For the operations of addition or subtraction of two mantissa, the exponents of numbers must be equal.
- These operations are performed by finding the difference between two mantissas and right shifting of mantissa of the number.
- Based on the obtained result, both significant are added or subtracted.
- Then, the significant should be handled, if there is an overflow or underflow during addition/subtraction operations.

3.1.2 Multiplication:

The multiplication of two floating point numbers requires the following:

- At first, the exponents of two operands are added to obtain the resultant exponent.
- Then, the mantissa of two operands is multiplied to obtain the result significant.
- The signs of these operands are XORed to attain a resultant sign.

Still some approximations are round up are performed to attain the final results. After multiplying two mantissas, the resultant significant is double than the original operands. So, it requires some adjustments in the resulting exponents on the intermediate results.

At first, the data including sign bit, exponent and mantissa bits are given as the input. For exponential bits A and B, the XOR operation is applied for performing addition and subtraction operations. For mantissa bits A and B, the arithmetic operations are applied with the controlling block. It includes both the multiplication and division operations. Then, the output of exponent and mantissa are given to the input of multiplexer, which produces the result of exponential register R. After that, XOR operation between the original mantissa bits and exponential bits are performed. It will be given to adder, multiplier and divider. Finally, the output of M_R and E_R are given to the output register, which produces the output R with sign bit, exponential E_R and mantissa M_R. Three components make up the total required power for each supply source.

3.2 FOLDED DTT DESIGN ARCHITECTURE

The Discrete Tchebichef Transform (DTT) is a direct transformation technique that utilizes the moments of Tchebichef to provide a basis matrix. Typically, it is derived from the orthonormal Tchebichef polynomials that exhibit similar energy compaction properties. A set of orthogonal features are based on the discrete Tchebichef moments.

The Fig.3 illustrates schematic figure of proposed FDTT architecture. Count of multiplication stage is reduced by crossing and folding of data lines with BF (Butterfly) blocks. Complex

input points from basic matrix are transmitted to complex output points by processing lines in BF blocks. Transformation process requires complex multiplication that needs new architecture referred as FPPE (Floating Point Processing Element).

Placement of proposed DTT on every BF do complex multiplication. SPFAS contains 4 intermediate stages among input and output. DTT consists of four stages such as:

- In stage 1, the inputs of x(n), x(n+1) are given to the adder circuit, and the inputs x(n+2), x(n+3) are given to another adder circuit.
- In stage 2: XOR operation is performed, and the value is given to the multiplexer.
- In stage 3: The output of MUX is given to register R1. Similarly, this operation in performed in another adder circuit and its output is given to register R2.
- In stage 4: The output weight W1 is obtained. Likewise, the weight values of W2, W3 and W4 are obtained in each stage, which are multiplied with the crossing of register value. Finally, the output Y is obtained in the output stage.

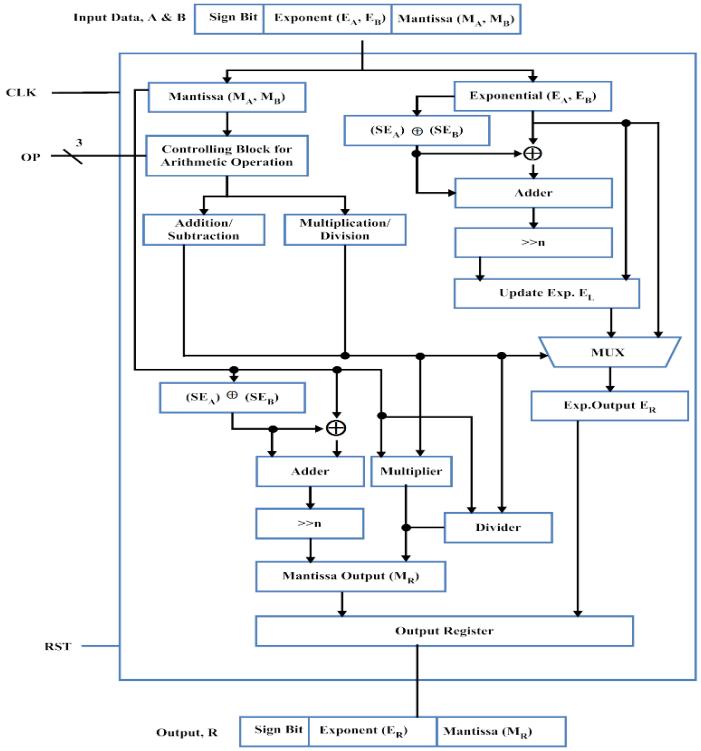


Fig 2. Architecture of FPPE

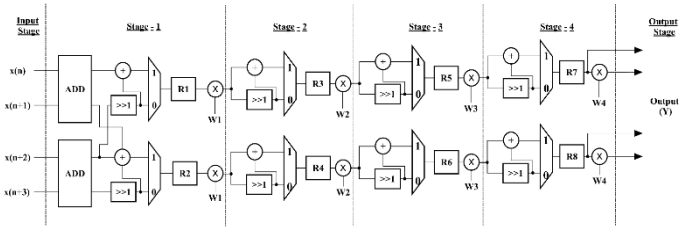


Fig 3. Folded DTT architecture

Combination of multiplexers and registers of BF block is responsible for forwarding and storage of data. Registers denotes crossing of data for every 'n' value on every stage and denotes relation between current stage with previous stage. Input is applied to R1 and R2 registers as: $x(n)$, $x(n+1)$, $x(n+2)$ and $x(n+3)$. Sequential manipulation through registers, BF blocks and multiplexers gives Y as final output. Registers in even and odd position were linked to BF blocks and multiplexers directly.

Gadget static (dissipation) power: This power will be used by any gadget for processing it for further making it programmable. Much power will be dissipated in the transistors which are being deployed for maintaining the configuration of the gadget under consideration.

- Clock power is the power used up by the clock propagating signals
- IO power is the power used up by IOs via IOBs
- Logic power is the power used up by the predefined logical blocks
- Leakage power is the standby power contained by FPGA known as quiescent power.
- Signal power is the power used up by the inner wires

The objectives of the proposed architecture design are listed as follows:

- To work with minimum number of components,
- To reduce area and delay, and
- To minimize the power consumption.

4. PERFORMANCE ANALYSIS

4.1 PRELIMINARIES

This section presents the results of both existing and proposed techniques. The effectiveness of this devised SPFAS-DTT structure is investigated and correlated with simulation i. The results are evaluated in terms of delay, area, latency, cycles, LUTs, slice and FFs. Appropriate existing techniques are considered in this performance comparison to make prove the effectiveness of the proposed SPFAS with the novel integration of FPPE and DTT.

4.2 COMPARATIVE ANALYSIS OF EXISTING AND PROPOSED FACTORS

Table.1. Number of components vs Used

Number of components	Used
BUFG/BUFGCTRL$_S$ Counts	1
Entirely consumed LUT-FF sets Counts	0
Slice registers counts	153
bonded IOBs counts	44
Slice LUTs counts	117

Table.2. Power consumption vs Power

Methods	Power (W)
RBBE	0.9
PPRB	0.146
Proposed	0.027

Above Table.1 shows the overall summary of all the hardware that are being used in this proposed work. On observing the Table.2, we could infer that this devised SPFAS yields lesser power consumption readings against the corresponding parameters that have been prior set. The power consumed by devised SPFAS was found to be 0.027 watts against the 0.9 and 0.146 yielded by RBBE and PPRB respectively.

Table.3. Summary for timing when under simulation

Parameter	Yielded value
Peak resulting period after clock	7.842
Less source arrival period before clock	0.342

From the above Table.3, we can infer that the proposed method had taken optimum level of minimum and maximum time periods correspondingly by yielding 0.342 and 7.842 respectively.

4.3 PERFORMANCE EVALUATION OF FAULTS AWARE SPFAS

The performance evaluation of the proposed SPFAS in the Isim tool is to show the overall contribution made by the proposed in the optimizing the circuit specific configuration. Fig.4 depicts performance of fault checking.



Fig.4. Performance of Fault checking

4.4 NUMERICAL EXAMPLE

4.4.1 Input Bits (63 to 0 bits):

1111111100000000111111110000000011111111000000001111
111100000000

4.4.2 Output Bits (41 to 0bits):

00000010001111111000000001000000100000001

4.4.3 Register

EA=> input bit from 62 to 55 =11111110, SA=1

EA=0000 0010

EB=> input bit from 30 to 23 =11111110, SA=1

EB=0000 0010

4.4.4 Adder:

ADD EA+EB=> SV=0000 0100

4.4.5 Register:

Eout=> inp=SV=0000 0100,SA=0;

Eout=0000 0100

4.4.6 FPPE Multiplier:

Input bit from 54 to 32 =MA=00000000111111110000000

Input bit from 22 to 0 =MB=00000000111111110000000

Out=W1=10000000100000001000000011111110

4.5 FINAL OUTPUT

- SB=1, SA=1, xor of SA, SB is '0'
- 0 00000100 01111111000000001000000100000001
- Output bits consist of 0 to 32 bits (10000000100000001000000011111110) from FPPE multiplier
- Next eight bits are from 33 to 40 the register which is from the input adder
- Last bit 41 bit is xor of SA, SB is '0'

4.6 VIRTEX-6 IMPLEMENTATION RESULTS OF EXISTING AND PROPOSED TECHNIQUES

Table 4 and Table 5 shows the virtex-6 implementation results of existing and proposed techniques for existing [32] FFT-M3 and proposed SPFAS techniques with respect to 1028 and 1038 bits. The results are evaluated in terms of product's mean latency, LUTs, latency, RAM, cycles, slice, time, and DSP. Here, the configured variable would have lesser area latency towards each aimed operators. Moreover, the real implementation results and the area latency-product sophisticated investigation counts are shown in the following tables. Based on the product's sophisticated area, clock time, and cycle count, the area latency is calculated.

Table.4. Evaluation of Virtex-6 utilization outcomes for bit sizes of 1028

Methodologies	DSP	LUTs	Area-latency in product	Latency	Cycles	RAM	Time (ns)	Slice
SPFAS	61	20651	47497	2.185	143	30	2.546	5979
FFTM3	135	25309	96174	3.8	387	44	9.37	8702

Table.5. Evaluation of Virtex-6 utilization outcomes for bit sizes of 1038

Methodologies	DSP	LUTs	Area-latency product	Latency	Cycles	RAM	Time (ns)	Slice
SPFAS	30	12175	30438	2.375	310	15	2.2895	2435
FFTM3	54	13573	52527	3.87	553	33	6.59	4786

From the analysis, it is found out that the sophisticated investigation goes hand in hand with the actual outcomes after implementing it. The Table.6 depicts the resource usage on Xilinx XC6VLX240T IFFG 1156 devices for existing [33] exact DTT, approximate DTT and proposed SPFAS methods. The results are simulated using Matlab Simulink, which are observed directly on the considered FPGA equipment and then verified with the loop investigation with the hardware combo. These approximations are verified many times of ten thousand iteration vectors with exact replica of the empirical readings. Various performance measures were compared and correlated accordingly. Moreover, the static Qp and dynamic Dp power usage are computed with the help of analyzer named Xilinx XPower. When compared to the results of app. DTT and exact DTT, the proposed SPFAS-DTT provides the improvised results.

Table.6. Resource Usage on Xilinx-XC6VLX240T – IFFG1156 equipment

Methodologies	CLB	CPD (ns)	Fmax (MHz)	Qp	FF	Dp
SPFAS-DTT	115	1.77	562.04	3.09	241	2.52
Approximate DTT	144	2.29	438.68	3.44	396	2.81
Exact DTT	408	2390	418.41	3.44	1370	5.1

5. CONCLUSION AND FUTURE WORK

This paper presented a novel SPFAS with integration being done between DTT and FPPE architecture with reduced logical components for reducing the power consumption, delay and area. The main contribution of this paper was to reduce the size of DSP unit and change the structure of shifter and adder for reducing the accumulator size, FF and LUT. The novel concept of this paper is, it does not require any separate block for performing subtraction and it uses the adder block with same XOR operation. Here, this circuit is implemented by using the Tchebichef filtering technique. The proposed work modified the design strategy of traditional DTT architecture by using SPFAS. It uses the same block for performing arithmetic operation, which results in an efficient reduction of delay rate. The performance of the proposed SPFAS was simulated and evaluated in terms of area, delay, cycle, LUT, FF and latency. Moreover, its results were compared with some of the existing floating point architecture models. From this analysis, we could conclude that the proposed SPFAS provides the best results when compared to the compared architecture designs. The future work aims to investigate the proposed architecture with many hardware so that the comprehensive case studies could be made with it.

REFERENCES

[1] M. Langhammer and B. Pasca, "Floating-Point DSP Block Architecture for FPGAS", *Proceedings of ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 117-125, 2015.

[2] D. Pani, C. Sau, F. Palumbo, and L. Raffo, "Computing Swarms for Self-Adaptiveness and Self-Organization in Floating-Point Array Processing", *ACM Transactions on Autonomous and Adaptive Systems*, Vol. 10, pp. 1-16, 2015.

[3] S. Arish and R. Sharma, "Run-Time-Reconfigurable Multi-Precision Floating-Point Matrix Multiplier Intellectual Property Core on FPGA", *Circuits, Systems, and Signal Processing*, Vol. 32, pp. 1-29, 2016.

[4] Y. Zhang, H. Cao, H. Jiang and B. Li, "Memory-Efficient High-Speed VLSI Implementation of Multi-Level Discrete Wavelet Transform", *Journal of Visual Communication and Image Representation*, Vol. 38, pp. 297-306, 2016.

[5] C.H. Hsia, J.H. Yang and W. Wang, "An Efficient VLSI Architecture for Discrete Wavelet Transform", *Proceedings of Asia-Pacific International Association Annual Summit and Conference on Signal and Information Processing*, pp. 684-687, 2015.

[6] N. Rajapaksha, A. Madanayake, R.J. Cintra, J. Adikari and V.S. Dimitrov, "VLSI Computational Architectures for the Arithmetic Cosine Transform", *IEEE Transactions on Computers*, Vol. 64, pp. 2708-2715, 2015.

[7] A.R. Raju, N.S. Murthy and M. Kamaraju, "FPGA Implementation of CORDIC Multiplier based Redundant Floating-Point Butterfly Architecture for Signal Processing Applications", *Architecture*, Vol. 3, pp. 1-13, 2016.

[8] A. Kaivani and S. Ko, "Floating-Point Butterfly Architecture Based on Binary Signed-Digit Representation", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 24, pp. 1208-1211, 2016.

[9] F. Ren, W. Xu and D. Markovic, "Scalable and Parameterised VLSI Architecture for Efficient Sparse Approximation in FPGAs and SoCs", *Electronics Letters*, Vol. 49, pp. 1440-1441, 2013.

[10] A. Pedram, A. Gerstlauer, and R.A. Van De Geijn, "Floating Point Architecture Extensions for Optimized Matrix Factorization", *Proceedings of International Conference on Computer Arithmetic*, pp. 49-58, 2013.

[11] K. Pande, A. Parkhi, S. Jaykar and A. Peshattiwar, "Design and Implementation of Floating Point Divide-Add Fused Architecture", *Proceedings of International Conference on Communication Systems and Network Technologies*, pp. 797-800, 2015.

[12] L. Gangwar and R. Chaudhary, "Floating Point Arithmetic Unit using Verilog", *Advances in Electronic and Electric Engineering*, Vol. 3, No. 8, pp. 1013-1018, 2013.

[13] J. Sohn and E.E. Swartzlander, "A Fused Floating-Point Four-Term Dot Product Unit", *IEEE Transactions on Circuits and Systems I: Regular Papers*, Vol. 63, pp. 370-378, 2016.

[14] J. Coleman and R.C. Ismail, "LNS with Co-Transformation Competes with Floating-Point", *IEEE Transactions on Computers*, Vol. 65, pp. 136-146, 2016.

[15] L.D.S. Rosa, C.F.M. Toledo and V. Bonato, "Accelerating Floating-Point to Fixed-Point Data Type Conversion with Evolutionary Algorithms", *Electronics Letters*, Vol. 51, pp. 244-246, 2015.

[16] M. Joldeş, O. Marty, J.M. Muller and V. Popescu, "Arithmetic Algorithms for Extended Precision using Floating-Point Expansions", *IEEE Transactions on Computers*, Vol. 65, pp. 1197-1210, 2016.

[17] M.K. Jaiswal, B.S.C. Varma, H.K.H. So, M. Balakrishnan, K. Paul and R.C.C. Cheung, "Configurable Architectures for Multi-Mode Floating Point Adders", *IEEE Transactions on*

Circuits and Systems I: Regular Papers, Vol. 62, pp. 2079-2090, 2015.

[18] E. Cant and L.M.G. Navarro, "Flexible Biometric Online Speaker-Verification System Implemented on FPGA using Vector Floating-Point Units", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 23, pp. 2497-2507, 2015.

[19] Q. Wang, X. Song, W.N.N. Hung, M. Gu and J. Sun, "Scalable Verification of a Generic End-Around-Carry Adder for Floating-Point units by Coq", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 34, pp. 150-154, 2015.

[20] E.E. Swartzlander and H.H.M. Saleh, "FFT Implementation with Fused Floating-Point Operations", *IEEE Transactions on Computers*, Vol. 61, pp. 284-288, 2012.

[21] R. Dorrance, F. Ren and D. Markovic, "A Scalable Sparse Matrix-Vector Multiplication Kernel for Energy-Efficient Sparse-Blas on FPGAs", *Proceedings of ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 161-170, 2014.

[22] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao and J. Cong, "Optimizing FPGA-Based Accelerator Design for Deep Convolutional Neural Networks", *Proceedings of ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 161-170, 2015.

[23] Z. Jovanovic and V. Milutinovic, "FPGA Accelerator for Floating-Point Matrix Multiplication", *IET Computers and Digital Techniques*, Vol. 6, pp. 249-256, 2012.

[24] S.S. Purohit, S.R. Chalamalasetti, M. Margala and W.A. Vanderbauwhede, "Design and Evaluation of High-Performance Processing Elements for Reconfigurable Systems", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 21, pp. 1915-1927, 2013.

[25] K.K. Matam and M. Hsieh, "Evaluating Energy Efficiency of Floating Point Matrix Multiplication on FPGAs", *Proceedings of ACM/SIGDA International Conference on High Performance and Computing*, pp. 1-6, 2013.

[26] L.S. Rosa, A.C.B. Delbem, C.F.M. Toledo and V. Bonato, "Design and Analysis of Evolutionary Bit-Length Optimization Algorithms for Floating to Fixed-Point Conversion", *Applied Soft Computing*, Vol. 49, pp. 447-461, 2016.

[27] M. Wang, F. Wang, S. Wei and Z. Li, "A Pipelined Area-Efficient and High-Speed Reconfigurable Processor for Floating-Point FFT/IFFT and DCT/IDCT Computations", *Microelectronics Journal*, Vol. 47, pp. 19-30, 2016.

[28] K. Manolopoulos, D. Reisis and V.A. Chouliaras, "An Efficient Multiple Precision Floating-Point Multiply-Add Fused Unit", *Microelectronics Journal*, Vol. 49, pp. 10-18, 2016.

[29] P. Rony Antony and A.M. Joseph, "Design and Implementation of Double Precision Floating Point Comparator", *Procedia Technology*, Vol. 25, pp. 528-535, 2016.

[30] K. Thiruvenkadam, J. Ramesh and V. Kalaiyarasi, "An Area Efficient Multi-Mode Quadruple Precision Floating Point Adder", *Microprocessors and Microsystems*, Vol. 45, pp. 310-323, 2016.

[31] R. Peesapati, K.K. Anumandla, S. Kudikala and S.L. Sabat, "Comparative Study of System on Chip based Solution for Floating and Fixed Point Differential Evolution Algorithm", *Swarm and Evolutionary Computation*, Vol. 19, pp. 68-81, 2014.

[32] D.D. Chen, G.X. Yao, R.C. Cheung, D. Pao and C.K. Koc, "Parameter Space for the Architecture of FFT-Based Montgomery Modular Multiplication", *IEEE Transactions on Computers*, Vol. 65, pp. 147-160, 2016.

[33] P.A. Oliveira, R.J. Cintra, F.M. Bayer, S. Kulasekera and A. Madanayake, "A Discrete Tchebichef Transform Approximation for Image and Video Coding", *IEEE Signal Processing Letters*, Vol. 22, pp. 1137-1141, 2015.

[34] L. Gangwar and R. Chaudhary, "Floating Point Arithmetic Unit using Verilog", *Advance in Electronic and Electric Engineering*, Vol. 3, No. 8, pp. 1013-1018, 2013.

[35] J.N. Coleman and R.C. Ismail, "LNS with Co-Transformation Competes with Floating-Point", *IEEE Transactions on Computers*, Vol. 65, pp. 136-146, 2015.

[36] L. Rosa, C. Toledo and V. Bonato, "Accelerating Floating-Point to Fixed-Point Data Type Conversion with Evolutionary Algorithms", *Electronics Letters*, Vol. 51, pp. 244-246, 2015.

[37] M. Joldes, O. Marty, J.M. Muller and V. Popescu, "Arithmetic Algorithms for Extended Precision using Floating-Point Expansions", *IEEE Transactions on Computers*, Vol. 65, pp. 1197-1210, 2015.

[38] M.K. Jaiswal, B.S.C. Varma, H.K.H. So, M. Balakrishnan, K. Paul and R.C. Cheung, "Configurable Architectures for Multi-Mode Floating Point Adders", *IEEE Transactions on Circuits and Systems I: Regular Papers*, Vol. 62, pp. 2079-2090, 2015.

[39] E. Canto Navarro, M. Lopez-Garcia, R. Ramos Lara and R. Sanchez-Reillo, "Flexible Biometric Online Speaker-Verification System Implemented on FPGA using Vector Floating-Point Units", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 23, pp. 2497-2507, 2014.

[40] Q. Wang, X. Song, W.N. Hung, M. Gu and J. Sun, "Scalable Verification of a Generic End-Around-Carry Adder for Floating-Point Units by Coq", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 34, pp. 150-154, 2014.

[41] E.E. Swartzlander and H.H. Saleh, "FFT Implementation with Fused Floating-Point Operations", *IEEE Transactions on Computers*, Vol. 61, pp. 284-288, 2010.

[42] Z. Jovanovic and V. Milutinovic, "FPGA Accelerator for Floating-Point Matrix Multiplication", *IET Computers and Digital Techniques*, Vol. 6, pp. 249-256, 2012.

[43] S.S. Purohit, S.R. Chalamalasetti, M. Margala and W.A. Vanderbauwhede, "Design and Evaluation of High-Performance Processing Elements for Reconfigurable Systems", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 21, pp. 1915-1927, 2012.

[44] K.K. Matam, H. Le and V.K. Prasanna, "Evaluating Energy Efficiency of Floating Point Matrix Multiplication on FPGAs", *Proceedings of International Conference on High Performance Extreme Computing*, pp. 1-6, 2013.

[45] L. Rosa, A.C. Delbem, C.F.M. Toledo and V. Bonato, "Design and Analysis of Evolutionary Bit-Length Optimization Algorithms for Floating to Fixed-Point

Conversion", *Applied Soft Computing*, Vol. 49, pp. 447-461, 2016.

[46] P.R. Antony and A.M. Joseph, "Design and Implementation of Double Precision Floating Point Comparator", *Procedia Technology*, Vol. 25, pp. 528-535, 2016.

[47] K. Thiruvenkadam, J. Ramesh and V. Kalaiyarasi, "An Area Efficient Multi-Mode Quadruple Precision Floating Point Adder", *Microprocessors and Microsystems*, Vol. 45, pp. 310-323, 2016.