# ENHANCING THE PERFORMANCE OF SPLIT RADIX FFT PROCESSOR USING APPROXIMATE ARITHMETIC CIRCUITS

**A.PRAVEENA[1], P.DHILIPMOHAN[2]**

*[1]PG Student[VLSI Design], Dept of ECE, Arasu Engineering College, Kumbakonam, Tamilnadu, India*
*[2]Assistant Professor [O.G], Dept of ECE, Arasu Engineering College, Kumbakonam, Tamilnadu, India*

Veena3011@gmail.com

**ABSTRACT**

Split Radix Fast Fourier Transform (SRFFT) is an efficient technique to design FFT processor, which consumes less dynamic power. The SRFFT algorithm is improved method of Fast Fourier Transform (FFT), which provides faster computation among various FFT algorithms. SRFFT Technique has modified architecture of FFT processor in such a way that it has least number of arithmetic operations to perform the same computation. In proposed methodology the approximate arithmetic circuits are used rather than accurate arithmetic circuits to perform FFT computation. Use of approximate circuits reduces the design complexity and also increases the performance interms of power consumption and area requirement. Generally the number of arithmetic operations such as multiplications and additions decides the computational complexity of the algorithm. Approximate circuits avoids use of many XOR gates, which results in faster computation and reduced logic gate count. This brief gives a comparative analysis of various hardware components such as FlipFlops, Look Up Tables (LUTs) used in both existing and proposed design.

*Keywords:* Accurate arithmetic, Approximate arithmetic, Computational complexity, Fast Fourier Transform (FFT), Split Radix FFT.

## I.INTRODUCTION

**T**he Digital Signal Processors found many applications as it deals with operations on signals. Many real time applications involves digital signal processors called DSPs. Such processor seems to be fast, have less chip area and low power consumption. These constraints leads to VLSI implementation of FFT processors as described in [1]. The FFT Processors can have either shared memory architecture or pipelined architecture. The difference is that the pipelined architecture offers increased throughput at the expense of more hardware resources where as the shared memory architecture uses less hardware resources while giving the slower throughput. In this approach Split Radix FFT of shared memory architecture is preferred.

Since the frequency domain signal is more prominent than the time domain signal, it is preferred widely in many signal processing applications as mentioned in [2]. The frequency domain signal gives information about both magnitude and phase components. It additionally involves harmonics and hence the error analysis seems to be easier. The Split Radix FFT has many advantages in Digital Signal Processing with better performance interms of Power consumption, Area efficiency and Speed. This SRFFT algorithm have entirely modified butterfly architecture as compared to Radix-2 FFT algorithm as shown in [2]. The main difference is that the Split Radix FFT uses two kind of Twiddle Factors called ‘j’ and ‘Wn’ whereas the Radix-2 FFT uses only one kind of Twiddle Factor called ‘Wn’.

Approximate arithmetic circuits offers reduced design complexity and better power efficiency. These approximate circuits provides slightly erroneous output, which is referred to as inexact computation as described in [3]. For example, the approximate full adder results in incorrect output for some input combinations. In Digital Signal Processing applications the slightly inexact output will not affect the result of the algorithm as in [4], because the Quantization process in DSP itself is an Approximation

technique. Hence the Approximate computing is well suited for DSP systems. Simulation results show that the approximate adder saves power upto 55%.

In this approach, the accurate full adder and full subtractor are replaced by approximate full adder and approximate full subtractor respectively. Our ultimate aim is to reduce the logic complexity and time complexity of the design with these approximate arithmetic circuits as shown in [5]. The mathematical expressions for approximate full adder and approximate full subtractor are represented, from which the corresponding arithmetic modules of the algorithm also changed.

## II. PREVIOUS WORK

Previous work of the project is that the Split Radix FFT processor designed with efficient addressing scheme. SRFFT algorithm categorized into two methods: Decimation In Time (DIT) and Decimation In Frequency(DIF). In this brief Split Radix DIF FFT algorithm is preferred. The decimation is done in frequency domain, hence called as Decimation In Frequency algorithm. The basic principle behind the SRFFT algorithm is that the radix-2 index is mapped to even index terms and radix-4 index is mapped to odd index terms using [6]. Hence SRFFT algorithm makes use of both Radix-2 FFT and Radix-4 FFT methods.

The N-point Discrete Fourier Transform is given by equation(1)

$$X[K] = \sum_{n=0}^{N-1} x[n].\, W_N^{nk} \quad (1)$$

Where k=0, 1... N-1 and $W_N = e^{-2\pi j/N}$ . The expression for even index term of Split Radix FFT is given by following equation (2)

$$X(2K) = \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) + x\left(n+\frac{N}{2}\right)\right] W_{N/2}^{nk} \quad (2)$$

Whereas the notation for odd index term is further splitted into two equations (3) and (4), they are represented as

$$X(4K+1) = \sum_{n=0}^{\frac{N}{4}-1} \left[x(n) - x\left(n+\frac{N}{2}\right) - \left(x\left(n+\frac{N}{4}\right) - x\left(n+\frac{3N}{4}\right)\right)\right] W_N^{n} W_{N/4}^{nk} \quad (3)$$

$$X(4K+3) = \sum_{n=0}^{\frac{N}{4}-1} \left[x(n) - x\left(n+\frac{N}{2}\right) + \left(x\left(n+\frac{N}{4}\right) - x\left(n+\frac{3N}{4}\right)\right)\right] W_N^{n} W_{N/4}^{nk} \quad (4)$$

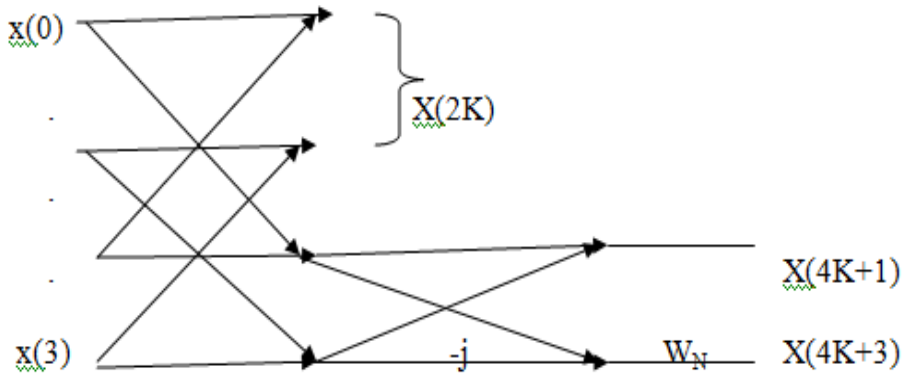


Fig1: Simple Flow Graph of 4-point Split Radix FFT

Based on the above equations the flow graph of 4-point Split radix FFT is constructed, which is shown in Fig.1. The SRFFT algorithm described in this approach have Shared Memory Processor architecture as shown in Fig.2. Two memory banks namely ROM and RAM are used to store the data and twiddle factors respectively. As mentioned in [6] and [7], at each clock cycle two input data and a twiddle factor are fetched from memory bank, the butterfly unit used to process

them based on addition and multiplications at each pass. The output is obtained in bit reversed order and is stored back to the memory, which replaces the old data. The butterfly counter indicates that which butterfly is under operation and Pass counter indicates that which pass is currently under operation. The efficient address generation logic is applied for both RAM and ROM banks.

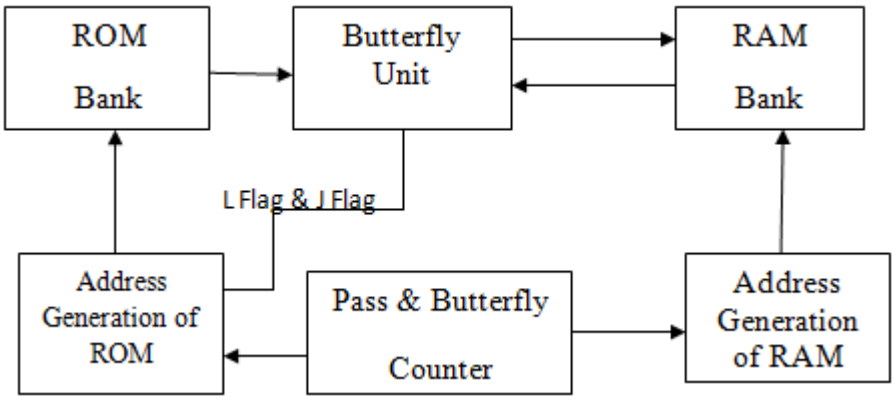


Fig 2: Architecture of Shared Memory Processor

Twiddle Factor is any of the trigonometric constant coefficient, which includes sine and cosine terms. Hence we are able to convert the time domain sequence into frequency domain sequence only with the help of twiddle factors as described in [8]. The algorithm mainly involves arithmetic operations to perform the computation. The address generation techniques are used to fetch the data and twiddle factors from RAM and ROM respectively. The number of inputs will be 4 and 16 for 4-point and 16-point SRFFT architectures respectively. This inputs includes both real part and imaginary parts and therefore the output also interpreted in real part as well as imaginary parts.

The address generation of twiddle factors is an important aspect in the proposed technique. Butterfly counter, pass counter, L Flag and J Flag are fundamental units used in this logic. The address generation is based on following two assumptions using [8]. First, if one butterfly unit is not within the L block in current pass, it will be definitely in L block in next pass. Second, if one butterfly unit need to be multiplied with 'j' in current pass, then the same butterfly unit should be multiplied with 'Wn' in next pass. In some approach, the twiddle factors need to be multiplied for each butterfly, therefore the ROM banks are always enabled. In this design, the L block signal enables the ROM banks only when the twiddle factors required as shown in [9]. This is an added advantage in this technique, which reduces further power consumption.

There are two important drawbacks of the previous design as described in [10]. First, the accurate adder and subtractor modules comprised of more number of XOR gates, which contribute to high area and delay in the algorithm. Second, the number of addition and multiplication performed in the arithmetic modules of the algorithm is significantly increased. Our proposed technique solves this two major issues and it is briefly discussed in following section.

## III. PROPOSED METHODOLOGY

In this section, the methodology of Approximate arithmetic circuits is included in Split Radix FFT algorithm to improve the performance of the Processor. The adder and Subtractor modules of the design are replaced with Approximate Adder and Approximate Subtractor modules. Their corresponding mathematical expressions, truth table and schematic view also derived and included in proposed technique. In this way approximate arithmetic is incorporated in Split Radix FFT design to perform the FFT computation.

### *1) Architecture Of SRFFT Algorithm*

. The main difference of butterfly architecture of Split Radix FFT is that it involves both trivial and nontrivial multiplications of twiddle factors. Here the trivial multiplications involves the twiddle factor 'Wn' and this contributes to actual multiplications. Whereas the nontrivial multiplication involves the twiddle factor 'j' and this multiplication is just swapping real and imaginary parts instead of an actual multiplication as mentioned in [11]. This portion of nontrivial

multiplication leads to less number of complex multiplications. The multipliers are enabled only whenever required, it is referred to as multiplier gating technique.

As Split Radix FFT uses mixed radix property, the equations defined in previous section results in L-shaped butterfly architecture as described in [12]. The following Fig.3 represents the Stages of Decimation for SRFFT algorithm. In this approach 16-point SRFFT is considered and the data word length is 32bit. First stage comprised of a 16-point Discrete Fourier Transform (DFT) computation. Second stage involved two 8-point DFT computation. Third Stage comprised of four 4-point DFT computation. Final stage also called output stage consists of eight 2-point DFT computation. Each input sequence comprised of real and imaginary parts and the output sequence is also a complex one.

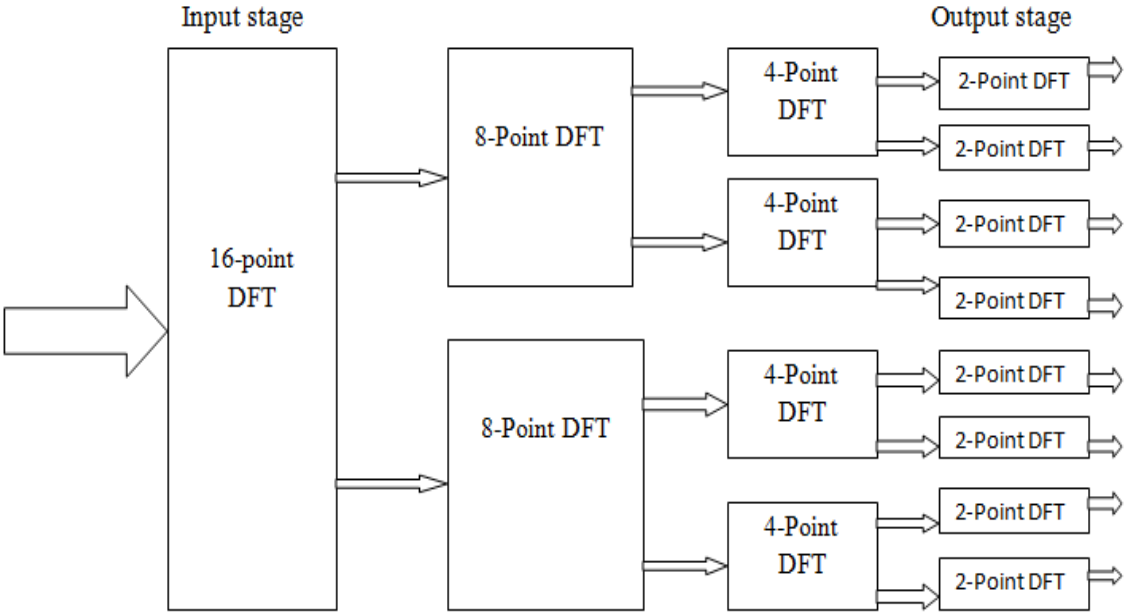


Fig 3: Decimation Stages of 16-point Split Radix DIF FFT

***2)Approximate Arithmetic Circuits in SRFFT***

In this section we need to understand the difference between the equations, truth table and logic diagram of both accurate arithmetic circuit and approximate arithmetic circuits. The equations for Sum and Carry are approximated as in equations (5) and (6). In approximate full adder one of the two XOR gate is replaced with an OR gate in Sum calculation as found in [13]. It is shown that the Sum of the approximate full adder results in incorrect output for last two cases out of eight cases. Similarly, the carry output results in error for one case out of eight cases.

Let W = A+B

Sum = W^C (5)

Carry =W.C (6)

The schematic diagram of Approximate full adder is given below in Fig.4. From this Figure We can see that the number of XOR gate is reduced by one when comparing to Accurate full adder. Likewise the total number of gates is three in proposed adder, whereas the gate count seems to be five in Accurate adder. This factor leads to logic complexity reduction in the design, which can be found in [13]. It is also proved that the proposed full adder eliminates four number of multiplication and one addition at each step. Hence the total number of operations in the algorithm is significantly reduced as compared to the existing design.

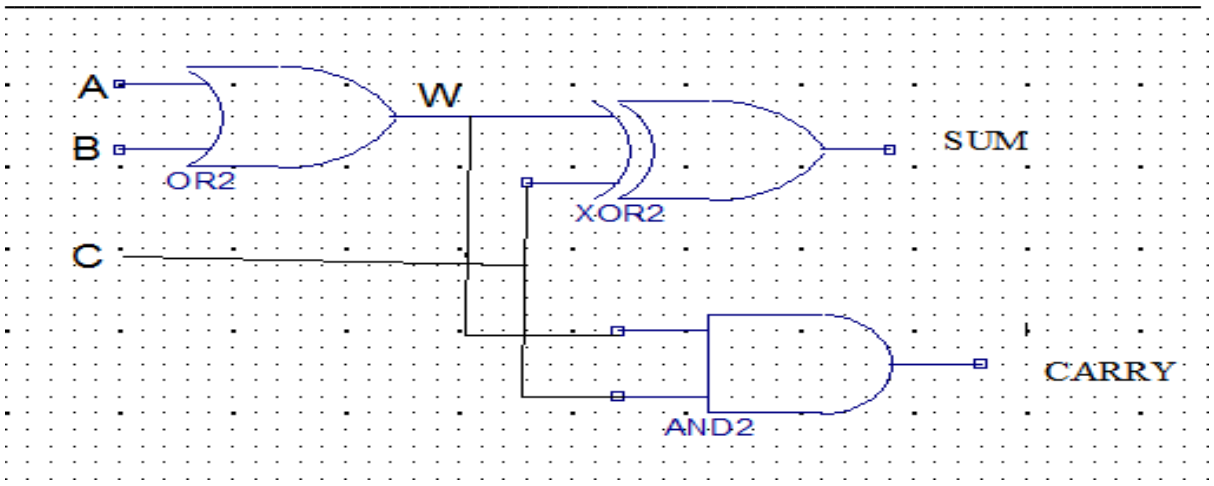


Fig 4:Schematic diagram of Approximate Full Adder

Table I shows truth table representation of Approximate full adder. From the table we can check the results of both accurate and approximate arithmetic. For some input combination Sum and Carry of approximate adder mismatches with accurate full adder. It is concluded that the carry outputs are approximated only for the cases, where the Sum is approximated as mentioned in [14]. A chain of four one bit full adder is used to design an four bit full adder. The chain of this eight four bit full adder makes the 32-bit adder, which is otherwise known as 32bit Ripple Carry Adder.

TABLE-I

TRUTH TABLE REPRESENTATION OF APPROXIMATE FULL ADDER

| Input | | | Accurate Output | | Approximate Output | | Output Status |
|---|---|---|---|---|---|---|---|
| A | B | C | Sum | Carry | Sum | Carry | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | Correct |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | Correct |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | Correct |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | Correct |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | Correct |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | Correct |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | Incorrect |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | Incorrect |

In addition to approximate full adder, the full subtractor of SRFFT algorithm also approximated using the equations (7) and (8),

Diff = ((A|B)^C) (7)

Borrow = ((~A|B)&C) (8)

The schematic diagram of Approximate full subtractor is shown in Fig.5. Using the chain of single bit full subtractor a 32-bit Approximate full

subtractor is designed. This subtractor also leads to logic complexity reduction similar to approximate full adder. This proposed subtractor avoids one of the two XOR gates. The total gate count is decreased by two when comparing to Accurate full subtractor. Due to this factor, switching at the output node of each gate is reduced, which results in reduced dynamic power consumption.

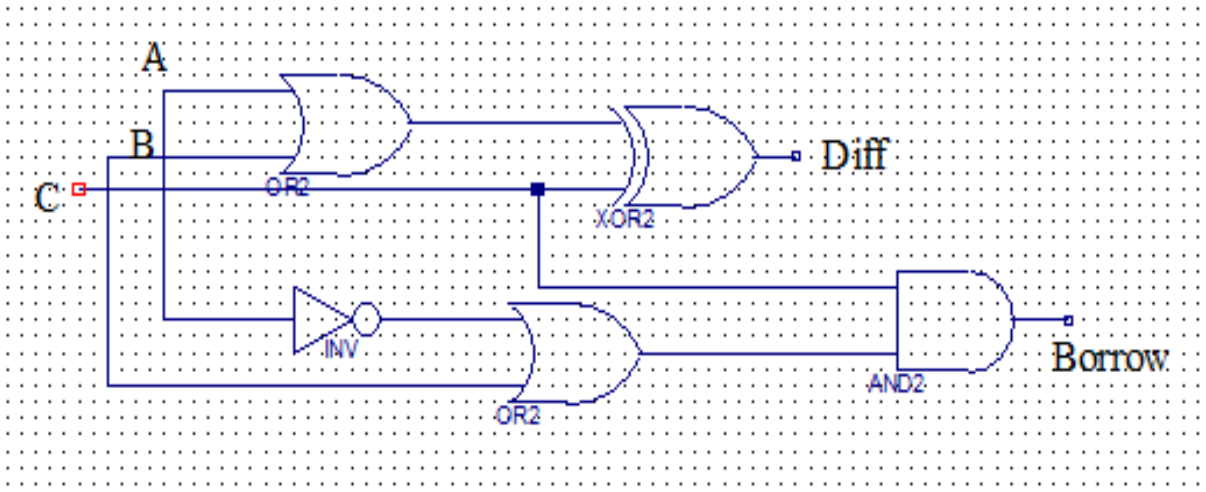


Fig 5:Schematic Diagram of Approximate Full Subtractor

. Each adder and subtractor modules were replaced with approximate full adder and approximate full subtractor respectively. As discussed in this section the SRFFT algorithm has improved performance interms of power efficiency, design complexity and computational complexity with the help of these approximate arithmetic circuits. The following section shows the simulation results of the proposed technique. A comparison analysis is also carried out to indicate the performance of proposed system.

## IV.RESULTS AND DISCUSSION

SRFFT algorithm for both previous and proposed design is developed in verilog code. The algorithm is synthesized and simulated using Xilinx ISE 9.1i targeting for Spartran3E family with XC3S500E device under the constraint of 100MHz. The test bench waveforms were simulated for 1-bit as well as 32-bit data word length for both the cases, which are shown in Fig.6 and Fig.7 respectively. We can interpret the output sequence as a combination of real and imaginary parts from the simulation results. However, the SRFFT algorithm has the irregular signal flow graph and makes the control of such processors more difficult than the fixed-radix ones. Although a software solution for the indexing problem is given, the indexing scheme is designed for the L butterfly structure, which is not suitable for the hardware implementation due to its uneven latencies. Some previous works use lookup tables to solve the indexing problem.

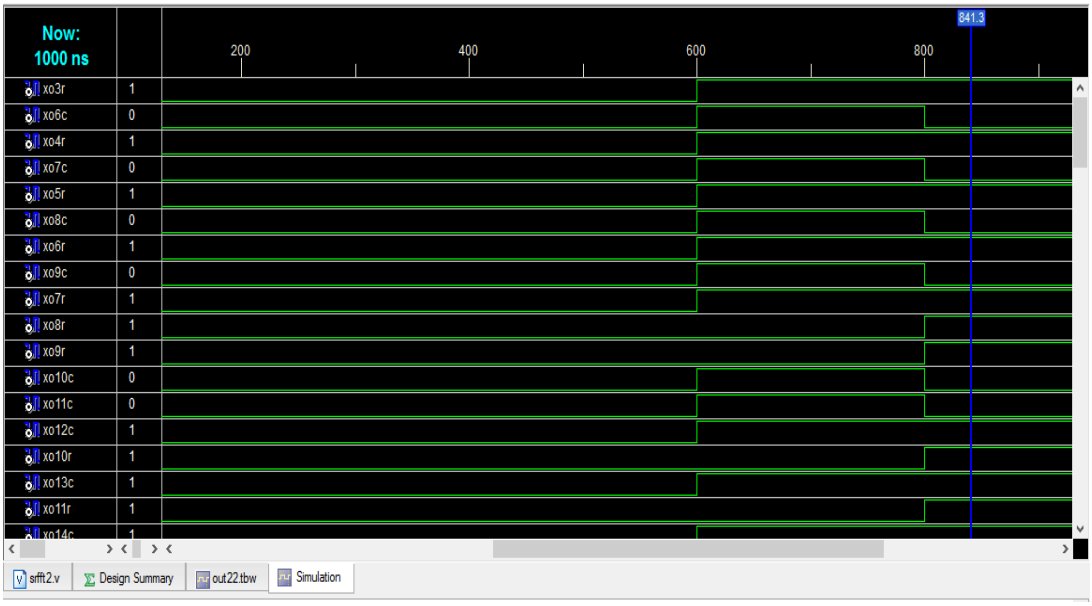


Fig 6:Output Waveform of 16-point Split Radix FFT (1 bit)


Fig 7:Output Waveform of 16-point Split Radix FFT (32 bit)

The power consumption is analyzed using Xilinx Xpower Estimator, which summarizes the power consumption interms of total on chip power and junction temperature. Total on-chip power is comprised of clock, Logic, RAM and DSP units. The device utilization summary of Xilinx ISE 9.1i provides detailed description about the ratio of number of used devices to number of available devices in the design. Table-II gives the comparison results of previous and proposed technique interms of number of Slices, Flip Flops, BRAM and LUTs. The multiplicative complexity is calculated using the derived mathematical equations.

TABLE-II

COMPARISON OF EACH COMPONENT FOR 16-POINT COMPUTATION

| Components | SRFFT using Accurate Arithmetic (Existing System) | SRFFT using Approximate Arithmetic (Proposed System) |
|---|---|---|
| Flip Flops | 2576 | 1184 |
| LUTs | 694 | 382 |
| BRAMs | 12 | 8 |

TABLE-III

COMPARISON OF SIMULATION RESULTS FOR POWER, AREA AND MULTIPLICATIVE COMPLEXITY

| Parameter | SRFFT with Accurate circuits (Existing System) | SRFFT with Approximate circuits (Proposed system) |
|---|---|---|
| Dynamic Power | 32.4mW | 24.7mW |
| Device Utilization | 80% | 62% |
| Number of complex multiplications | 32 | 18 |

This Table-III compares the power consumption, number of complex multiplications and device utilization percentage of Existing technique and proposed technique. From the above table, we can conclude that our proposed design achieves low power consumption, utilizes the components efficiently and involves reduced number of multiplications as compared to existing technique. The reduction in these three parameters makes our proposed design more efficient than previous technique.

## CONCLUSIONS

In this brief, the Split Radix FFT algorithm with approximate arithmetic circuits is designed to reduce the dynamic power consumption at the cost of reduced hardware resources. The static power consumption of proposed design remains the same as existing technique. In the proposed architecture the number of non trivial multiplications are reduced, which in turn reduces the multiplicative complexity of the algorithm. Besides lesser gate count the switching activities of the design are significantly reduced, which leads to low dynamic power consumption. All these above factors makes the proposed design to achieve over 19% lower power consumption than the existing design.

## REFERENCES

[1] Zhuo Q, Martin M. Low power Split Radix FFT Processors using Radix-2 Butterfly units. IEEE Transactions on very Large Scale Integrated Systems. 2016:25-36.

[2] Qian Z, Nasiri N, Segal O, and Margala M. FPGA implementation of low-power split-radix FFT processors. Proc 24th Int Conf Field Program Logic Appl,2014:1-2.

[3] Vaibhav G and Anand Raghunathan. Low-Power Digital Signal Processing Using Approximate Adders. IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems, 2013:32(1):1-50.

[4] Ashim G and Vinay kumar. Design of Low Power, Area Efficient and High Speed Approximate Adders for Inexact Computing. IEEE Transaction on Very Large Scale Integrated systems. 2016:1-30.

[5] Milos D. Ercegovac. Approximate Arithmetic for low power signal processing applications. IEEE transactions on VLSI systems, 2013: 32(2): 30-75.

[6] Yeh W.C and Jen C.W. High-speed and low-power split-radix FFT. IEEE Trans Signal Process, 2003:51(3):864-874.

[7] Hsiao C.F, Chen Y, Lee C.Y. A generalized mixed-radix algorithm for memory-based FFT processors. IEEE Trans Circuits Syst II Exp. Briefs 2010:57(1):26-30.

[8] Cheng C and Parhi K.K. Low-cost fast VLSI algorithm for discrete Fourier transform. IEEE Trans Circuits Syst I, Reg. Papers, 2007:54(4):791-806.

[9] Suto J, Oniga S and Hegyesi G. A simple fast Fourier transformation algorithm to microcontrollers and mini computers. 18th Int Conf on Intelligent Engineering Systems, Tihany, 2014, pp. 61–65.

[10] Johnson L.G. Conflict free memory addressing for dedicated FFT hardware. IEEE Trans Circuits Syst. II, Analog Digit. Signal Process., 1992:39(5):312-316.

[11] Chang Y.N. An efficient VLSI architecture for normal I/O order FFT design. IEEE Trans Circuits Syst II, Exp. Briefs, 2008: 55(12):1234-1238.

[12] Chen J, Hu J, Lee S, and Sobelman G.E. Hardware efficient mixed radix-25/16/9 FFT for LTE systems. IEEE Trans Very Large Scale Integr (VLSI) Syst., 2015: 23 (2): 221 - 229.

[13] Kwong J and Goel M. A high performance split-radix FFT with constant geometry architecture. Proc Design, Autom Test Eur Conf Exhibit, Dresden, Germany, 2012: 1537–1542.

[14] Duhamel P and Hollmann H. Split radix FFT algorithm., Electron Lett, 1984:20(1):14-16.

[15] Balasubramanian P and Dang C. Approximate Carry look ahead adder and Ripple carry adder A comparison analysis. IEEE Transaction on circuit and systems II, 2017:24(3):50-86.

[16] Richards M.A. On hardware implementation of the split-radix FFT. IEEE Trans Acoust Speech Signal Process., 1988:36(10): 1575-1581.