International Journal of Industrial Engineering *Vol. 1, No. 8, 2017. Page 239-243.*
http://ijie.gjpublications.com/ ***ISSN: 2456-8449.***

Research Article

An Efficient Approach for Design of Low Power Split Radix FFT Processor with Reduced Computational Complexity

A. Praveena*, P. Dhilipmohan

Department of Electronics and Communication Engineering, Arasu Engineering College, Kumbakonam – 612501. India.

*Corresponding author's e-mail: veena3011@gmail.com

Abstract

The purpose of Fast Fourier Transform (FFT) is to compute the frequency domain sequence from its time domain sequence. The Fast Fourier Transform is improved version of Discrete Fourier Transform (DFT), which used to perform the computations faster than DFT approach. Our proposed technique has modified architecture of FFT processor in such a way that it has least number of arithmetic operations to perform the same computation. Whenever dealing with FFT algorithms, the address generation schemes need to be done for both input data and twiddle factors. In this approach the multipliers are enabled whenever necessary, which reduces the dynamic power consumption. Generally the number of arithmetic operations such as multiplications and additions decides the computational complexity of the algorithm. In this approach the numbers of complex multiplications are significantly reduced as compared to Radix-2 FFT algorithm. Similarly the number of used flipflops, Look Up Tables, slices and memory are reduced comparing with previous design. Hence the proposed architecture consumes less dynamic power, have reduced number of multiplications and area efficient.

Keywords: Fast Fourier Transform; Split Radix FFT; Twiddle Factors; Multiplier gating.

Introduction

The 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 whereas the shared memory architecture uses less hardware resources while giving the slower throughput, which is shown in [2].

Since the frequency domain signal is more prominent than the time domain signal, it is preferred widely in many signal processing applications. 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. Fourier transform cannot be calculated on any digital processor since it is continuous in nature. This problem can be solved by evaluating Fourier transform at only discrete points using [3].

In this approach, the shared memory architecture of split radix FFT processor is implemented. It requires two address generation schemes for both input data and twiddle factors as in [4]. As mentioned earlier, the SRFFT algorithm has modified butterfly unit as compared to radix-2 butterfly unit of FFT Algorithm. Initially two input data and twiddle factors are provided by memory banks. The butterfly unit performs the computations according to the equations described. The output sequence is obtained in bit reversed order and stored back to the memory banks by replacing the old data. This is the important operation performed by butterfly architecture at each clock cycle. Hence the butterfly unit is represented as core of the algorithm.

Previous work

Previous work of the project is that the FFT processor designed using radix-2 butterfly architecture. Radix-2 FFT algorithm categorized into two methods: Decimation In Time(DIT) and

Received: 01.11.2017; Received after Revision: 10.11.2017; Accepted: 12.11.2017; Published: 02.12.2017.

Decimation In Frequency(DIF). In our previous work [4] Radix-2 DIF algorithm is preferred. Radix-2 means that the number of samples must be an integral power of two and it is generally based on divide and conquer approach. The decimation is done in frequency domain, hence called as decimation in frequency algorithm.

The N-point Discrete Fourier Transform is given by eq. 1.

$X[K] = \sum_{n=0}^{N-1} x[n].e^{-2\pi ikn/N}$ (1)

Where k=0, 1... N-1 and $W^{nk}_N = e^{-2\pi ikn/N}$. If we split X[k] into even and odd terms, the equations of radix-2 FFT can be derived as eq. 2 and 3.

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

$X(2K+1) = \sum_{n=o}^{\frac{N}{2}-1} [(x(n) - x\left(n+\frac{N}{2}\right)]\ W^{n}_{N}\, W^{nk}_{N/2}$ (3)

Based on above equations the butterfly architecture is designed as shown in fig. 1 and the stages of decimation is shown in fig. 2, which are used to process the input data in conjuction with twiddle factors as in [5]. 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 2 and 16 for 2-point and 16-point FFT architectures respectively. This input includes both real part and imaginary parts and therefore the output also interpreted in real part as well as imaginary parts.

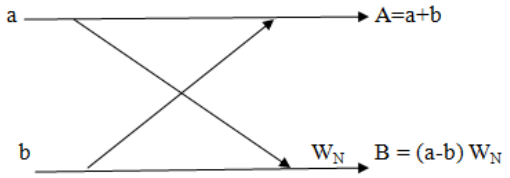

Fig. 1. Simple flow graph of 2-point Radix-2 DIF FFT

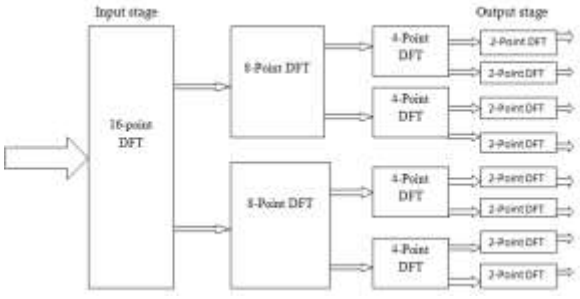

Fig. 2. Decimation Stages of 16-point Radix-2 DIF FFT

There are two important drawbacks of the previous design, which are identified in [6] and [7]. First, it could not use the multiplier gating technique to reduce unnecessary switching activities hence the dynamic power consumption is high. Second, it involves trivial multiplication of twiddle factors with input data, which leads to more number of complex multiplications. Our proposed technique solves these two major issues.

Proposed methodology

In this section, the methodology of Split Radix FFT Algorithm was discussed briefly. Unlike radix-2 FFT algorithm the split radix FFT Architecture makes use of both radix-2 and radix-4 computations. 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 [8] and [9]. The equation for even index term is same as that of Radix-2 FFT technique and is given by eq. 4.

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

Whereas the equation for odd index term is further splitted into two equations, which are distinct from Radix-2 FFT technique, given as eq. 5 and 6.

$X(4K+1) = \sum_{n=0}^{\frac{N}{4}-1} [x(n) - x\left(n+\frac{N}{2}\right) - j\left(x\left(n+\frac{N}{4}\right) - x\left(n+\frac{3N}{4}\right)\right)]\ W^{n}_{N}\, W^{nk}_{N/4}$ (5)

$X(4K+3) = \sum_{n=0}^{\frac{N}{4}-1} [x(n) - x\left(n+\frac{N}{2}\right) + j(x\left(n+\frac{N}{4}\right) - x\left(n+\frac{3N}{4}\right))]\ W^{n}_{N}\, W^{nk}_{N/4}$ (6)

Based on the above equations the flow graph of 4-point Split radix FFT is constructed, which is shown in fig. 3. In this way the basic flow graph of Split Radix FFT differs from conventional Radix-2 FFT technique, which can be identified using fig. 3.

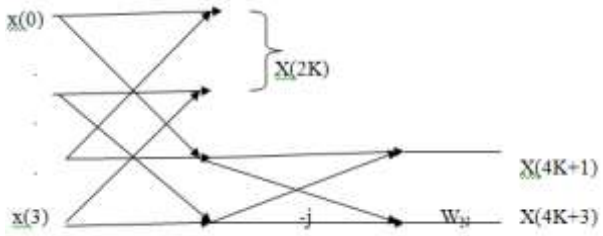

Fig. 3. Simple Flow Graph of 4-point Split Radix FFT

Shared memory processor

The block diagram of shared memory processor is shown in Fig. 4. It includes two memory banks called RAM and ROM to store the FFT data and twiddle factors respectively. It is shown that the butterfly architecture of split radix algorithm is same as radix-2 FFT except the locations and values of twiddle factors on each leg using [9] and [10].

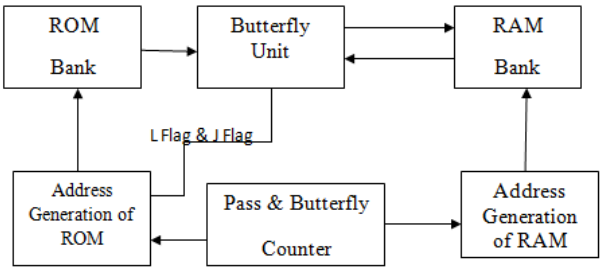

Fig. 4. Architecture of shared memory processor

As mentioned before 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 calculated result is stored back to the memory, which replaces the old data. The butterfly counter keeps the track of addresses, which have been used. The address generation logic is almost same for both Radix-2 FFT method and Split Radix FFT method. Due to mixed radix property of split radix algorithm, the location of twiddle factors is irregular.

Butterfly architecture of Split Radix FFT

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 involve 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 whenever required; it is referred to as multiplier gating technique. As Split Radix FFT uses mixed radix property, the equations defined above results in L shaped architecture. There is totally five L shaped butterfly architecture for four passes. In this approach 16-point FFT is considered and the data word length is 32 bit. Each input sequence comprised of real and imaginary parts and the output sequence is also a complex one.

Address generation logic

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 [12]. 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 previous design, the twiddle factors need to be multiplied for each butterfly; therefore the ROM banks are always enabled. In the proposed design, the L block signal enables the ROM banks only when the twiddle factors required as shown in [13]. This is an added advantage in this technique, which reduces further power consumption.

It is important to mention that in conventional implementations, the twiddle factors are mandatory for each and every butterfly unit, hence the ROM banks are enabled at every time. But proposed technique shows that the L Flag signal could be used as the enable signal for the ROM memory, because if the butterfly belongs to the L block, no multiplication is required. This will reduces the power consumption significantly. Because of this reason the ROM banks are enabled whenever required, our modified butterfly architecture leads to lesser number of switching activities and low dynamic power consumption. This factors leads to efficient utilization of memory banks.

Results and discussion

The conventional radix-2 FFT and SRFFT algorithms were developed in verilog code. Both are synthesized and simulated using Xilinx ISE 9.1i targeting for Spartran3E family with XC3S500E device under the constraint of 100 MHz. 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. 5 and fig. 6 respectively. We can interpret the output sequence as a combination of real and imaginary parts from the simulation results. Compared with the radix-2 addressing schemes, our addressing method requires additional 2S−1-bit memory. 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. It is obvious that the proposed algorithm requires

significantly less memory than the lookup table approach.

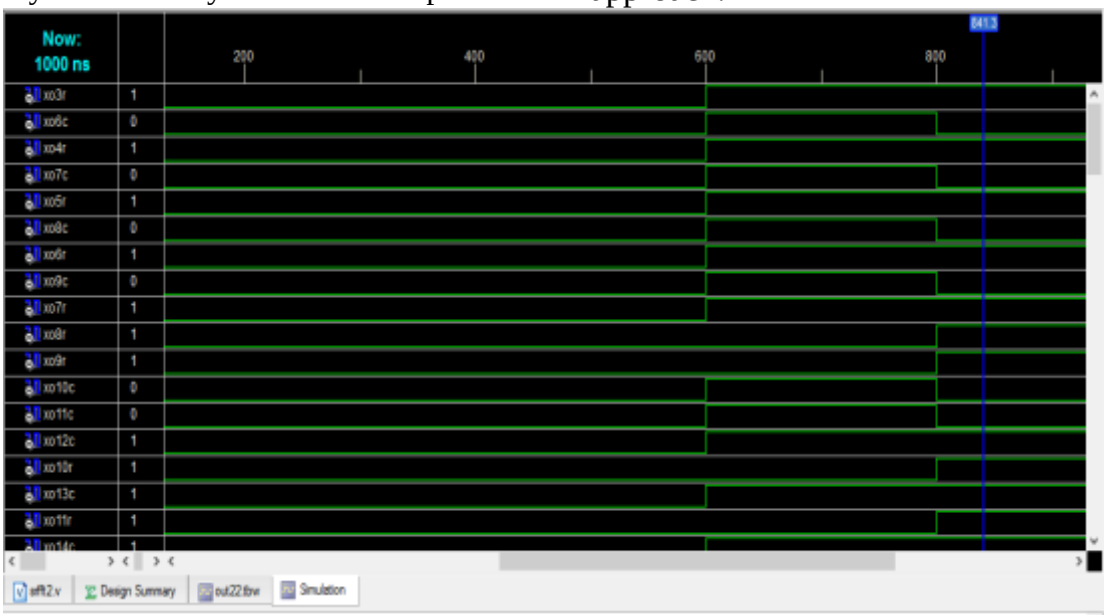

Fig. 5. Output Waveform of 16-point Split Radix FFT (1 bit)

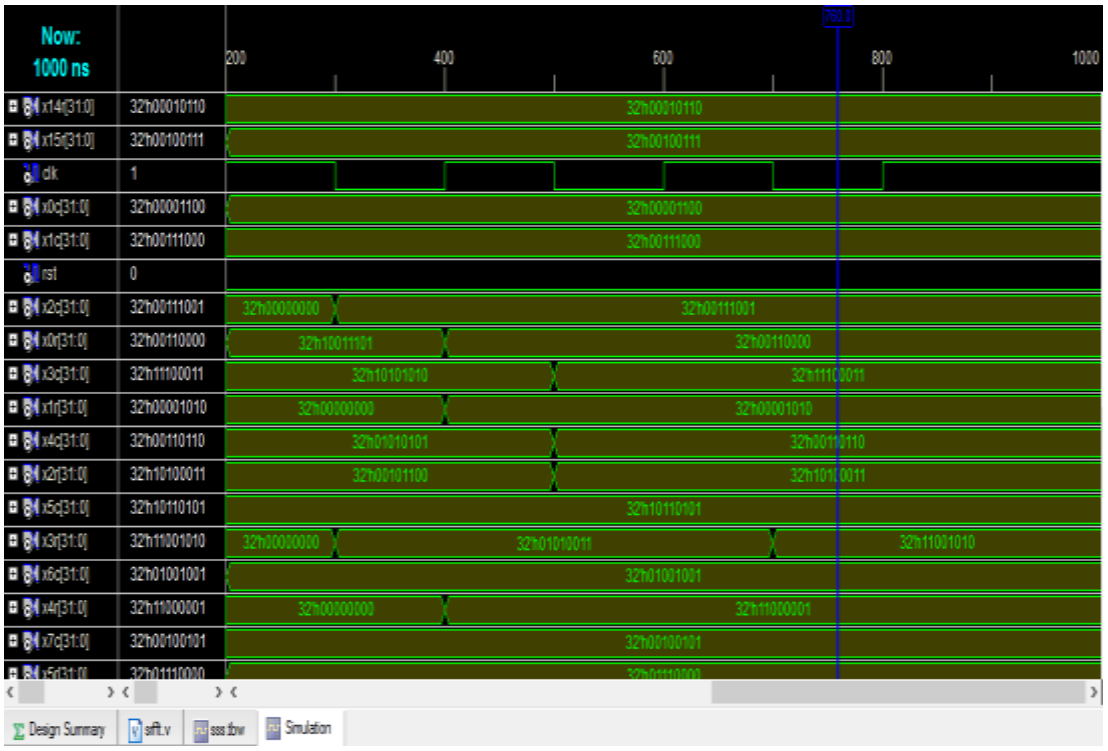

Fig. 6. Output waveform of 16-point Split Radix FFT (32 bit)

The power consumption is analyzed using Xilinx Xpower Estimator, which summarize the power consumption in terms 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 1 gives the comparison results of previous and proposed technique in terms of number of Slices, Flip Flops, BRAM and LUTs. The multiplicative complexity is calculated using the mathematical equations and number of complex multiplications involved in the computation. Table 2 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.

Table 1. Comparison of each component for 16-point computation

Components	Radix-2 FFT (Existing System)	Split Radix FFT (Proposed System)
Flip Flops	2734	2576
LUTs	8278	8151
BRAMs	12	8

Table 2. Comparison of simulation results for power, area and multiplicative complexity

Parameter	Radix-2 FFT (Existing System)	Split Radix FFT (Proposed system)
Dynamic Power	31.4 mW	23.7 mW
Device Utilization	80%	72%
Number of complex multiplications	32	18

Conclusions

In brief, the Split Radix FFT Processor is designed to reduce the dynamic power consumption at the cost of reduced hardware resources. The static power consumption of SRFFT remains the same as conventional technique. In the proposed architecture the number of non-trivial multiplications are reduced, which in turn reduces the multiplicative complexity of the algorithm. The switching activities of the design are significantly reduced due to its multiplier gating technique. All these above factors makes the proposed design to achieve over 18% lower power consumption comparing to radix-2 architecture when computing a complex valued transform.

Conflicts of interest

Authors declare no conflict of interest.

References

[1] Xiao X, Oruklu E, Saniie J. An efficient FFT engine with reduced addressing logic. IEEE Trans Circuits Syst II 2008:55(11):1149-1153.

[2] Yang CH, Yu TH, Markovic D. Power and area minimization of reconfigurable FFT processors. IEEE J Solid-State Circuits 2012: 47(3):757-768.

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

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

[5] Suto J, Oniga S, Hegyesi G. A simple fast Fourier transformation algorithm to microcontrollers and mini computers. 18th International Conference on Intelligent Engineering Systems, Tihany, 2014; pp. 61-65.

[6] Johnson LG. Conflict free memory addressing for dedicated FFT hardware. IEEE Trans Circuits Syst II 1992:39(5):312-316.

[7] Chang YN. An efficient VLSI architecture for normal I/O order FFT design. IEEE Trans Circuits Syst II 2008:55(12):1234-1238.

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

[9] Zhuo Q, Martin M. Low power Split Radix FFT Processors using Radix-2 Butterfly units. IEEE Trans Very Large Scale Integr Syst 2016;24:3008-3012.

[10] Qian Z, Nasiri N, Segal O, Margala M. FPGA implementation of low-power split-radix FFT processors. Proceedings of the 24th International Conference on Field Program Logic Applications, 2014.

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

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

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

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

[15] García J, Michell JA, Burón A. VLSI configurable delay commutator for a pipeline split radix FFT architecture. IEEE Trans Signnal Process 1999:47:3098-3107.

****

