LOW- POWER SPLIT-RADIX FFT PROCESSORS USING CARRY SELECT

G. Renuka[1], R. Ranjeetha[2], K. Divya[3], C. R. Aarthi Chandran[4]

[1] *Assistant Professor, Department of ECE, Prince Dr. K. Vasudevan College of Engineering and Technology,Tamil nadu, India*
[2] *Assistant Professor, Department of ECE, Prince Dr. K. Vasudevan College of Engineering and Technology,Tamil nadu, India*
[3] *Assistant Professor, Department of ECE, Prince Dr. K. Vasudevan College of Engineering and Technology,Tamil nadu, India*
[4] *Assistant Professor, Department of ECE, Prince Dr. K. Vasudevan College of Engineering and Technology,Tamil nadu, India*

ABSTRACT

Split-Radix Fast Fourier Transform (SRFFT) is mainly for the implementation of a low-power FFT processor. In FFT algorithms, SRFFT has less number of arithmetic operations. Twiddle factor is required in FFT addressing processors. The signal flow graph of SRFFT is the same as radix-2 FFT and so conventional addressing schemes is used in of SRFFT. However, it has improper arrangement of twiddle factors and denies application of radix-2 address generation methods.This brief presents a shared-memory low-power SRFFT processor architecture. We show that SRFFT can be computed by using a butterfly unit. The butterfly unit exploits the multiplier-gating technique to save dynamic power. In addition, two novel address generation algorithm for both trivial and non-trivial twiddle factors are developed. The proposed design achieves over a less delay, higher speed and 20% dynamic low power consumption.

Keyword: - *SRFFT, Twiddle factor, Butterfly unit, etc….*

1. INTRODUCTION

The fast Fourier transform (FFT) is one of the most important and fundamental algorithms in the digital signal processing area. Since the discovery of FFT, many variants of the FFT algorithm have been developed, such as radix-2 and radix-4 FFT. Since arithmetic operations significantly contribute to overall system power consumption, SRFFT is a good candidate for the implementation of a low-power FFT processor. In general, all the FFT processors can be categorized into two main groups: pipelined processors or shared-memory processors.

ErdalOruku implemented "An Efficient FFT engine with reduced addressing logic" In this technique including butterfly structure and address generation method for fast fourier transform. It uses reduce logic to generate the address, avoiding the parity check and barrel shifter commonly used in FFT implementations. A 16-point FFT with a 32 bit complex number is synthesized using a CMOS 0.18micro meter technology. The methods reorder the addresses of the butterfly inputs and outputs to realize the parallel accessing of the memory. The circuit gate count analysis indicates the significant logic reduction can be achieved with improved throughput compared to the conventional implementation. A highly efficient FFT memory addressing scheme with significant logic reduction and delay improvements compared to existing shared-memory based FFT methods [1].

Kwong implemented " High performance split radix FFT with constant geometry architecture ". A new parallel FFT architecture which combines the split-radix algorithm with a constant geometry interconnect structure . It exploit the lower arithmetic complexity of split-radix to lower dynamic power , by gating the multipliers during trivial

multiplications . It achieves 46% lower power. It proposes pair of radix-2 like datapath with shorter latencies. The conventional "L-shaped" split-radix datapath has uneven latencies and is thus not suited for high throughput operation. Using these datapath, the split-radix signal flow graph can be reorganized to have a constant geometry structure. The proposed architecture enables power reduction in two ways , it achieves multiplicative complexity then radix-4 algorithm while using a shorter datapath reduces glitch power and the complex multipliers can be gated to saves dynamic power[3].

Rikard Andersson" Multiplierless unity-gain SDF FFTs". A novel approach to implement multiplierless unity-gain single delay feedback fast fourier transforms(FFTs). This reduces the amount of hardware resources of the FFT architecture, while having high accuracy in the calculations. It achieve unity gain, but also require the smallest number of adders among current SDF FFTs. The SDF FFT is one of the most attractive and widely used FFT architectures. It allows for high throughput and requires a low amount of resources. The proposed architectures are not only multiplierless and achieve unity gain, but also require the smallest number of adders among current SDF FFTs[5].

2. EXISTING METHODLOGY

A pipelined architecture provides high throughputs, but it requires more hardware resources at the same time. One or multiple pipelines are often implemented, each consisting of butterfly units and control logic. In contrast, the shared-memory-based architecture requires the least amount of hardware resources at the expense of slower throughput. The scope of this brief is limited to the shared-memory architecture.

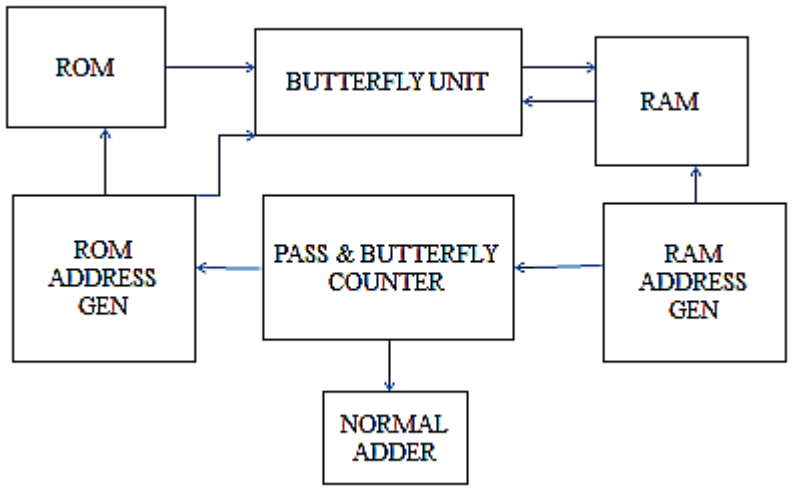

Fig -1 Existing Block Diagram

The above figure 1 shows in the radix-2 shared memory architecture, the FFT data are organized into two memory banks. At each clock cycle, two FFT data are provided by memory banks and one butterfly unit is used to process the data. At the next clock cycle, the calculation results are written back to the memory banks and replace the old data.

2.1 Butterfly Unit

Butterfly is a portion of the computation that combines the result of smaller discrete transform into a larger DFT or vice versa. The name butterfly comes from the shape of the data flow diagram in the radix-2. The same structure can also be found in the Viterbi algorithm, used for finding the most likely sequence of hidden states. The term butterfly appears in the context of cooley-Tukey FFT algorithm, which recursively breaks down a DFT of composite size n=rm into r smaller transforms of size m where r is the radix of the transform. These smaller DFTs are then combined via size-r butterflies, which themselves are DFTs of size r(performed m times on corresponding outputs of

the sub transforms) pre-multiplied by roots of unity(knows as twiddle factor). This is the decimation in time case; one can also perform the steps in reverse, known as decimation in frequency.

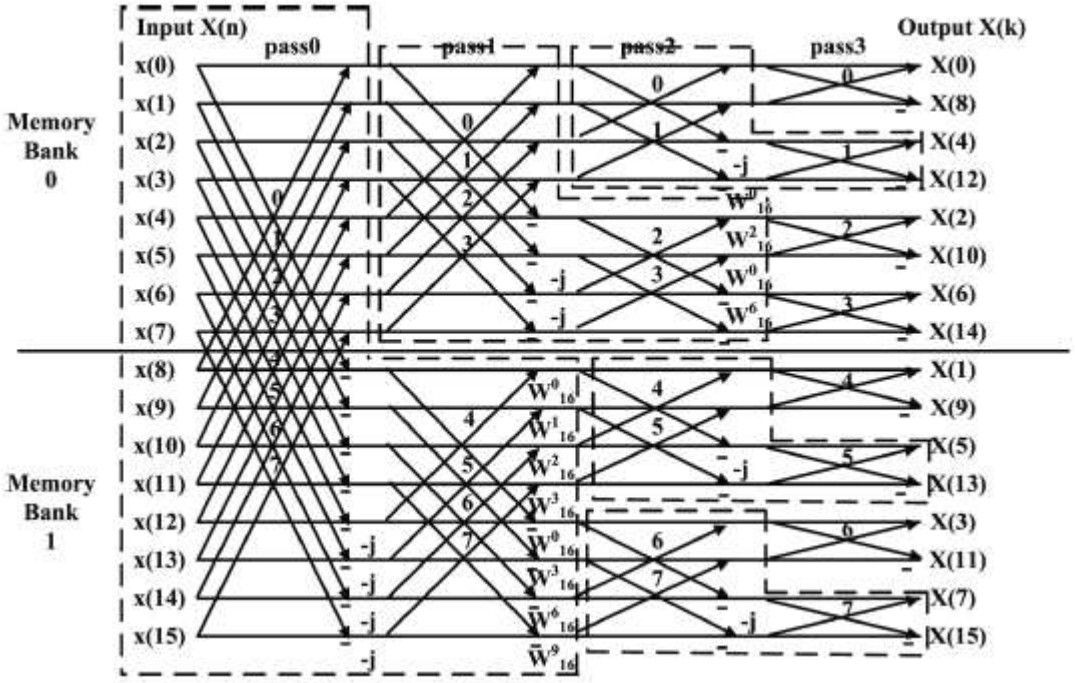

Fig. 2 Signal flow Graph

In figure shows the signal flow graph is divided into two memory banks and each stage is divide into the pass stage of zero, one, two and three. Here, the time domain is converted into frequency domain**.** It requires more multiplication, with high power and more delay. These disadvantages are overcome by the following proposed method.

3. PROPOSED METHOD

For split-radix FFT, it conventionally involves an L-shaped butterfly datapath whose irregular shape has uneven latencies and makes scheduling difficult. In this brief, we show that the SRFFT can be computed by using a radix-2 butterfly structure. Our contribution consists of mapping the split-radix FFT algorithm to the shared-memory architecture, leveraging the lower multiplicative complexity of the algorithm to reduce the dynamic power and developing two novel twiddle factor addressing schemes for the split-radix FFT. In figure 3 shows the proposed block diagram when compared to existing block diagram we include carry select adder in this structure.The CSLA is simulated and synthesized, the area and power is less in the CSLA compared to BEC as the delay is not reduced. So we can improve the above structure in terms of less delay and higher speed by replacing the BEC with a D-latch.

The architecture of shared-memory processor. The FFT data and the twiddle factors are stored in the RAM and ROM banks, respectively. We observed that the flow graph of split radix algorithm is the same as radix-2 FFT except for the locations and values of the twiddle factors and therefore, the conventional radix-2 FFT data address generation schemes could also be applied to SRFFT (RAM address generator). However, the mixed-radix property of SRFFT algorithm leads to the irregular locations of twiddle factors and forbids any conventional address generation algorithm (ROM address generator).

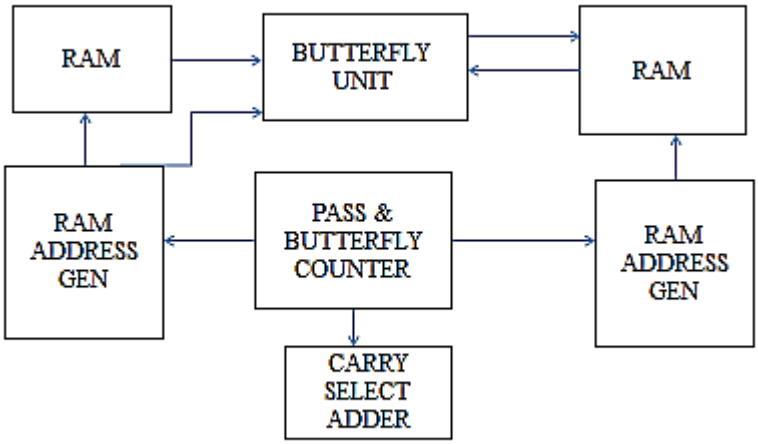

Fig 3 Proposed Block Diagram

In figure 4 shows shared memory architecture, the entire memory, i.e., main memory and disks, is shared by all processors. A special fast interconnection network allow any processor to access any part of the memory in parallel. All processors are under the control of a single operation system which makes it easy to deal with load balancing.

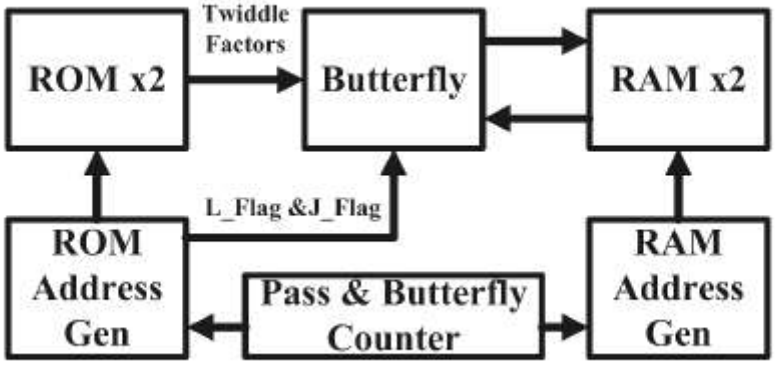

Fig -4 Shared Memory Architecture

3.1 Carry Select Adder (CSLA)

In figure 5 shows the carry select adders reduce the computation time by pre-computing the sum for all possible carry bit values (ie „0" and „1"). After the carry becomes available the correct sum is selected using multiplexer. Latches are used to store one bit information. Instead of using two separate adders in the regular CSLA, in this method one adder is used to reduce the area and other adder is used to reduce the power consumption. When clock goes high addition for carry input one is performed. When clock goes low then carry input is assumed as zero and sum is stored in adder itself. The latch is used to store the sum and carry for cin=1and cin=0. CSLA is one of the fastest adders to perform arithmetic operations comparing to all conventional adders. The sum of each bit position in an elementary adder is generated sequentially only after the previous bit position has been summed and a carry propagated into the next position. Ripple carry adders are simplest and most compact full adders, but their performances is limited by a carry that must propagate from the least significant bit to the most significant bit.

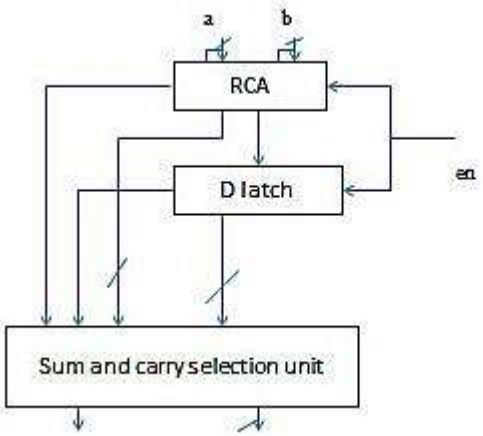

Fig- 5 Carry Select Adder with D- Latch

3.3 FFT DESIGN

FFT is widely used in modern digital signal processing. The goal is to design a FFT IP as an accelerator of the OR1200 CPU. Digital signals that needs to be processed using fast fourier transform algorithm can directly go into this IP. The final delivery is expected to be a layout and a netlists of the FFT accelerator. In figure 6 shows FFT design to compute the discrete fourier transform at n points with an optimal speed-up as long as the memory is large enough. The fast Fourier transform [FFT] is one of the most important and fundamental algorithms. The fourier transform is used to deliver a fast approach for the processing of data in the wireless transmission. The fast fourier transform is one of the methods of converting the time domain to frequency domain data with less hardware requirement and fast time ultilization. The conventional signal and image processing applications requires high computational power based on fast fourier transform in addition to the ability to choose the algorithm and architecture. The implementation has been made on FPGA as a way of obtaining high performance at economical price and a short time of realization.

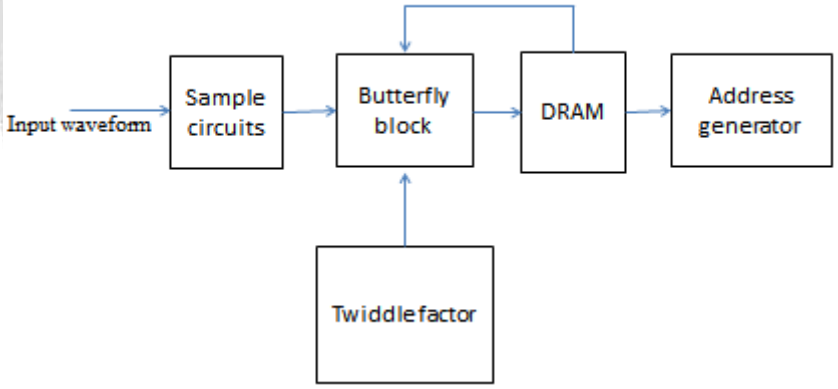

Fig-6 FFT Design

4. SIMULATION RESULT

The following figure shows the simulation results .

The counter can be displayed. According to the logic of 1 and 0 in the L_flag and J_flag. Figure 7 the counts is done both up and down counter according to the command of a control unit.

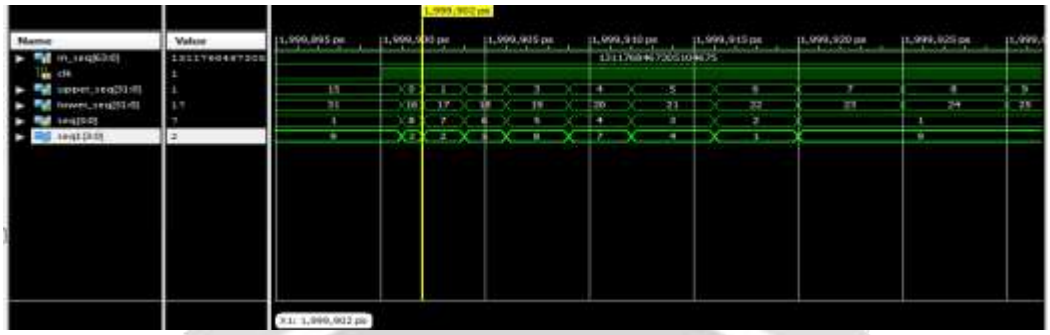

Fig 7 Counter

In the butterfly structure, there are two stages upper leg and bottom leg in that performing trivial and non-trivial multiplications are performed and the corresponding twiddle factors are stored. Figure 8 shows the twiddle factor storage for corresponding input.

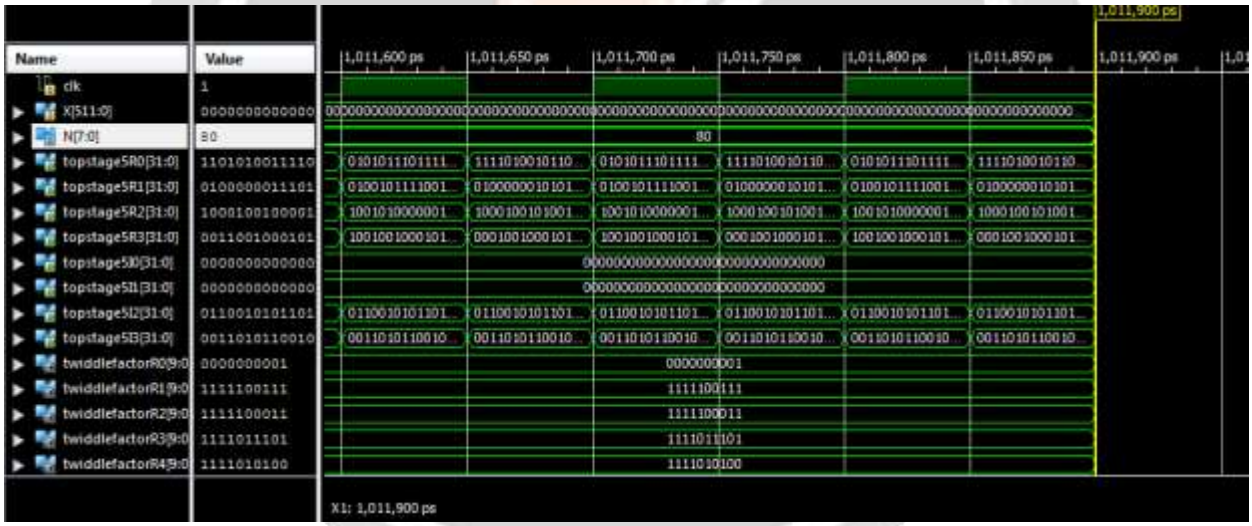

Fig 8 Butterfly Unit

The following figure 9 shows result CSLA using D - LATCH

Fig 9 CSLA using D- Latch

4. CONCLUSIONS

This method reduces the dynamic power consumption at the expense of more hardware resources. It also present two addressing schemes for both the trivial and non trivial twiddle factors. Since SRFFT has the minimum number of multiplications compared with other types of FFT, the result would be more optimal in the sense of floating point operations. The FFT can be implemented in 256 points FFT processor with efficient algorithm and technique to improve the efficiency of the proposed system

5. REFERENCES

1. D. Cohen(Dec. 1976), "Simplified control of FFT hardware," IEEE Trans. Acoust., Speech, Signal Process., vol. 24, no. 6, pp. 577–579.
2. J. Chen, J. Hu, S. Lee, and G. E. Sobelman(Feb. 2015) , "Hardware efficient mixed radix-25/16/9 FFT for LTE systems," IEEE Trans. Very Large Scale Integration (VLSI) Syst., vol. 23, no. 2, pp. 221–229.
3. P. Duhamel and H. Hollmann(Jan. 1984), "'Split radix' FFT algorithm," Electron, Lett., vol. 20, no. 1, pp. 14–16.
4. L. G. Johnson(May .1992), "Conflict free memory addressing for dedicated FFT hardware," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.,vol. 39, no. 5, pp. 312–316
5. J. Kwong and M. Goel(Mar.2012), "A high performance split-radix FFT with constant geometry architecture," in Proc. Design, Autom. Test Eur. Conf.Exhibit. (DATE), Dresden, Germany, pp. 1537–1542.
6. Z. Qian, N. Nasiri, O. Segal, and M. Margala(Sep.2014), "FPGA implementation of low-power split-radix FFT processors," in Proc. 24th Int. Conf. Field Program. Logic Appl., Munich, Germany, pp. 1–2.
7. A.Richards(Oct.1998), "On hardware implementation of the split-radix FFT," IEEE Trans. Acoust., Speech Signal Process., vol. 36, no. 10,pp. 1575–1581.
8. N. Skodras and A. G. Constantinides(Feb.1992), "Efficient computation of the split-radix FFT," IEE Proc. F-Radar Signal Process., vol. 139, no. 1, pp. 56–60.
9. H. V. Sorensen, M. T. Heideman, and C. S. Burrus(Feb.1986), "On computing the split-radix FFT," IEEE Trans. Acoust., Speech Signal Process., vol. 34, no. 1, pp. 152–156.
10. X. Xiao, E. Oruklu, and J. Saniie,(Nov.2008) "An efficient FFT engine with reduced addressing logic," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 55,no. 11, pp. 1149–1153.
11. W.C. Yeh and C.W. Jen(Mar.2003), "High-speed and low-power split-radixFFT," IEEE Trans. Signal Process., vol. 51, no. 3, pp. 864–874.