International Journal of Engineering Technology and Applied Science

(ISSN: 2395 3853), Vol. 5 Issue 4 April 2019

Efficient I/O order Radix-2 FFT Architecture using Vedic Multiplier

Ashish Raghuwanshi

Assistant Professor

Department of Electronics & Communication Engineering, IES, Bhopal

Abstract: The fast 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 data to frequency domain data with less hardware requirement and fast time utilization. The radix-2 Vedic multiplication algorithm is being used for the realization of the 256- point FFT. The input signals for the design have already been settled to 16 bits and the rest of the signals in the design are fixed-point format. The programming part is done by using Verilog. The Xilinx ISE Design Suite 13.1 is being used in the analysis of the design utilization part. The device utilization summary and result analysis of proposed design has presented.

Keywords: Radix-2, FFT, DFT, Vedic Multiplier, Verilog.

I. INTRODUCTION

Today's communication market faces strong competition and multiple new standards. For this reason, several systems require new embedded processors (EP). In the current technical environment, embedded processors (EP) and the necessary development tools are designed manually, with very little automation. This results in a long, labor-intensive process requiring highly skilled engineers with specialized know-how - a very scarce resource. Most of today's processor design is conducted by EP and IC vendors using a variety of development tools from different sources, typically lacking a well-integrated and unified approach. Engineers design the architecture, simulate it in software, design software for the target application, and test the implementation for hardware and software integration. These EPs can either be general purpose, such as microcontrollers (µC) and digital signal processors (DSP), or application specific, using application specific instruction set processors (ASIP). The decision between flexibility and high optimization to special applications is driven by the time intensive task to develop new architectures. In fact, there is only little margin for design exploration and finding the best-in-class solution. This results from the fact that the development process of new ASIPs is separated into several development phases, such as design exploration, software tools design, system integration and design implementation [1]. The development time can be decreased significantly by employing a Vedic multiplication approach using a machine description language. The Language for Instruction Set Architectures (LISA) [1][2] was developed for the automatic generation of consistent software development tools and synthesizable HDL code.

Fourier Transform is basis of many signal processing and communication applications. It is the analysis of the signal in its frequency domain. The Fourier transform has many applications, in fact any field of physical science that uses sinusoidal signals, such as engineering, physics, applied mathematics, and chemistry, will make use of Fourier series and Fourier transforms. Most of the fields nowadays make use of digital and discrete data. Thus the determination of Fourier Transform of discrete signals is of prime importance and such a transform is called Discrete Fourier Transform (DFT). Fast Fourier Transform (FFT) is an efficient algorithm to evaluate DFT. Fast Fourier transform (FFT) algorithms are computationally efficient algorithms that exploit these properties of Twiddle factor. It computes the DFT of N number of discrete data samples in O(N log2N) time as opposed to O(N2) in the direct method.

FFT is a basic technique for digital signal processing applicable to spectrum analysis, digital filter, speech recognition, image processing and so on. While application fields of FFT are growing rapidly, the amount of data to be transformed is increasing tremendously. In order to efficiently compute FFT, various parallel algorithms and their implementation to processor array have been developed. Some of the algorithms are Radix-2, Radix-4, Quick Fourier Transform and Split Radix Transform.

Despite the tremendous advancements made in digital computers during recent decades, the impact of the FFT continues to be felt. Many technologies enjoyed by the common public would as yet not be possible without the Cooley - Tukey FFT and its derivatives. Synthetic aperture radar (SAR), a type of imaging radar, operates at sampling rates of hundreds of Mega-

Hertz, or even Giga-Hertz. This is a tremendous computational load, even for modern digital computers. A typical general-purpose computer would be hard-pressed to sustain such a load in real-time. If implemented using the DFT, then the task would be impossible.

This example shows that some applications of the FFT are beyond any general purpose microprocessor, and even some of the latest multiprocessing systems. Considering that some DSP algorithms require multiple DFT calculations to be executed concurrently, and on a platform that is both small and low-power, it is clear that the demand for high-performance FFT implementations has only increased with time, and will continue for the foreseeable future.

II. VEDIC MULTIPLICATION ALGORITHMS

The word “Veda” has this derivational meaning i.e. the fountain-head and illimitable store-house of all knowledge. This derivation, In effect, means and implies that the Vedas should contain within themselves all the knowledge needed by mankind relating not only to the so called “spiritual”(or other wordly) matters but also to those usually described as purely “secular”, ”temporal” or ”wordly” and also to the means required by humanity as such for the achievement of all-round, complete and perfect success in all conceivable directions and that there can be no adjectival or restrictive epithet calculated (or tending) to limit that knowledge down in any sphere, any direction or in any respect (Maharaja 2009). In other words, it connects and implies that our ancient Indian Vedic lore should be all-round, complete and perfect and able to throw the fullest necessary light on all matters which any aspiring seeker after knowledge can possibly seek to be enlightened on.

Vedic mathematics shows its application in fast calculations (multiplication, division, squaring, cubing, square root, cube root), trigonometry, log and exponential. The basic sutras and upa sutras in the Vedic Mathematics helps to do almost all the numeric computations in easy and fast manner. The ancient Indian Vedic mathematics is now currently employed in our global silicon chip technology for easier and faster calculations (Paramasivam and Sabeenian 2010).

Vedic Multiplication

The multiplier architecture is based on the (Urdhva Triyagbhyam) Vertical and Crosswise algorithm of ancient Indian Vedic Mathematics. Consider two 4 bit numbers A and B. Divide bits of A and B into two parts, say a3a2&a1a0 of A and b3b2&b1b0 of B. Using the fundamentals of Vedic multiplication, taking two bit at a time and using 4 X 4 multiplier block, the following structure for multiplication is realized as shown in Figure 2.1.

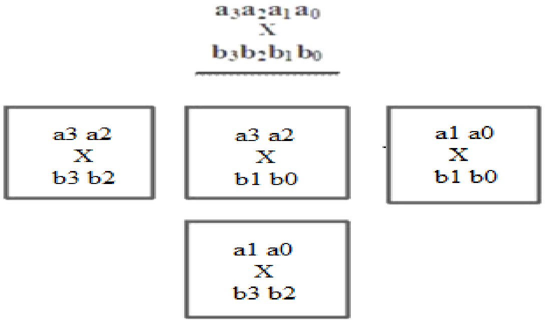


Figure 2.1 Basic 4X4 Vedic Multiplication.

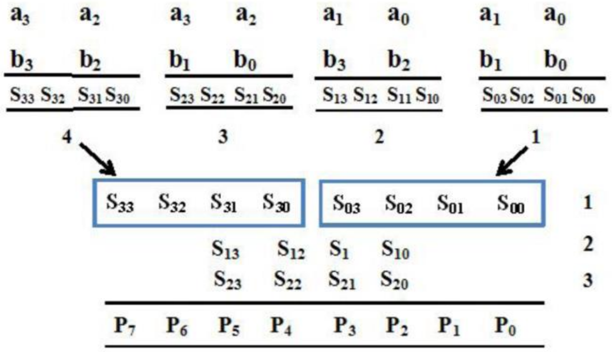


Figure 2.2 Partial Product Array of 4X4 Vedic Multiplication

The inputs for the rightmost 2 X 2 multiplier are a1a0 and b1b0 and for the leftmost 2 X 2 multiplier are a3a2 and b3b2. The mid-multiplier blocks have a3a2 & b1b0 and a1a0 & b3b2 as the inputs. The final result of multiplication which is of 8 bit p7,p6,p5,p4,p3,p2,p1,p0 can be interpreted as given in Figure 2.2.

III. PROPOSED APPROACH

The implementation has been made on a Field Programmable Gate Array (FPGA) as a way of obtaining high performance at economical price and a short time of realization. It can be used with segmented arithmetic of any level of pipeline in order to speed up the operating frequency. The radix-2 FFT algorithm is obtained by using the Vedic multiplier approach split the output sequence X(k) into two summations, one of which involves the sum over the first 2/ N data points and the second sum involves the last 2/ N data points.

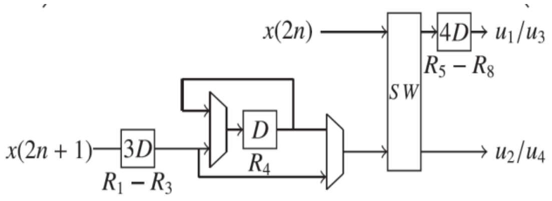


Figure 3.1 Structure of RSR Delay Commutator unit in L_1 and M_1

As shown in figure the Structure of RSR Delay Commutator unit in L_1 and M_1.further the detailed structure of RSR Delay Commutator unit in L_1 and M_1 has shown in figure 3.2. Figure 3.3 shows the utilization of proposed FFT used in N- Point FFT Architecture.

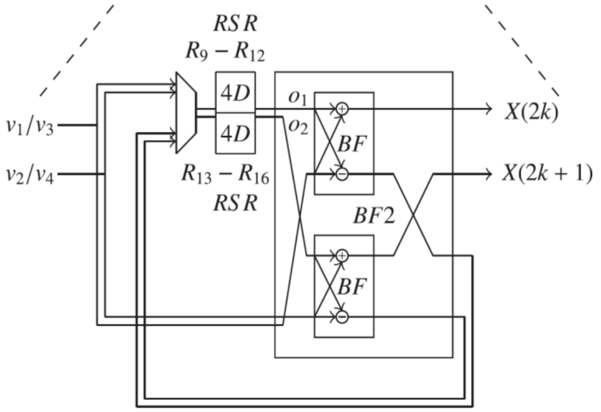


Figure 3.2 Detailed Structure of L_3, M_3

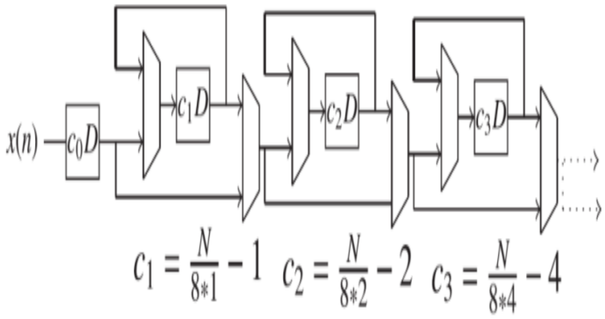

$x(n)$
c_0D
c_1D
c_2D
c_3D
$c_1 = \frac{N}{8*1} - 1$
$c_2 = \frac{N}{8*2} - 2$
$c_3 = \frac{N}{8*4} - 4$

Figure 3.3 Proposed FFT used in N- Point FFT Architecture

The internal structure of SW1 and SW2 has given in figure 3.4. Where u1, u2, u3, u4, are the input and v1, v2, v3, v4 are the outputs. As a fast computation algorithm, compared to DFT, the Fast Fourier Transform (FFT) is famous for decomposing the DFT computing module into small calculation blocks, which is called radix-2. By using that, the arithmetical complexity will be decreased from $O(N^2)$ to $O(N\log_2 N$), which will increase the computation speed and the total computational cost will be greatly reduced.

The radix-2 butterfly is used to construct FFT algorithms for operating on sequences of a size that is a power-of-two. A butterfly computes a DFT of size n, where n is the radix. So the radix-2 butterfly computes a 2-point DFT. The general rule is that as the radix of the butterflies increase, fewer twiddle factor. There exist dozens of other FFT butterflies of vary ingradices, each an atomic unit that computes some n-point DFT. Multiplications are required, but this is at the expense of less flexibility in available sizes.

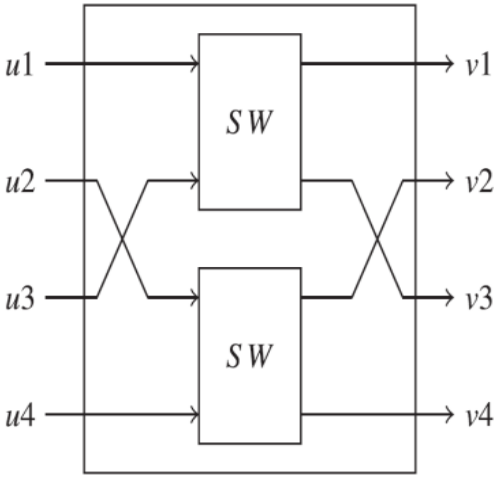


Figure 3.4 Internal Structure of $SW_1$and SW_2

From figure 3.4 the control signals to the multiplexer in proposed Vedic multiplier based algorithm are properly varied to interleave the data. Let log2 N is even, log2 N-2 multiplexers are required otherwise log2 N multipliers are required for bit reversal. It is found that proposed design using Vedic multiplication algorithm in radix-2 is a delay efficient high speed design from simulation outcome the comparison table has given to compare performance of proposed design.

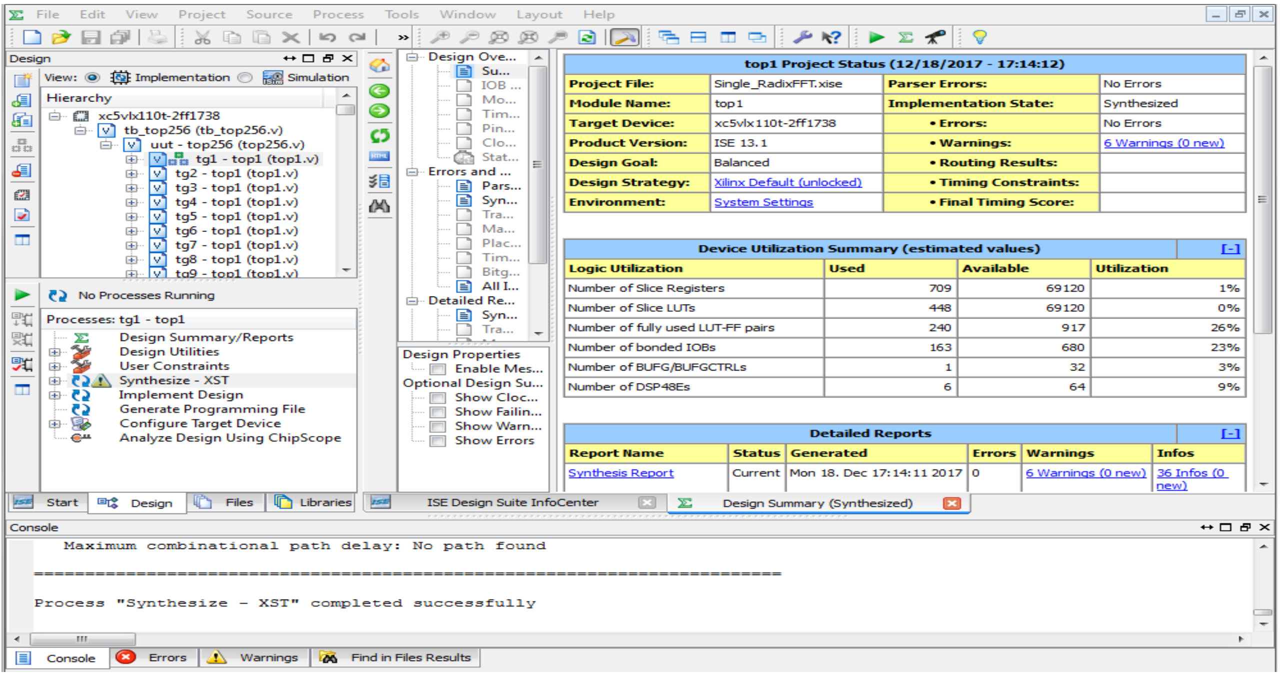
IV. SIMULATION OUTCOMES

The simulation of proposed work has done on Xilinx 13.1 Design suite. The simulation screen has shown in figure 4.1. The results of proposed work has been evaluated based on the performance parameters of device such as registers LUTs and Latency. And the obtained result parameters are compared with existing base work [1]. The comparison of result are shown in table 1.

Result comparison chart of proposed work with existing work in terms of number of register count, number of lookup tables count in Xilinx simulation device and latency in Nano-seconds has given in figure 4.2.

Table 1: Comparison of Area and Latency of 256 Point Architecture Single Stream.

Parameters	Previous	Proposed
Registers	1536	709
LUTs	2420	448
Latency	505 ns	20.703 ns



Hierarchy
- xc5vlx110t-2ff1738
  - tb_top256 (tb_top256.v)
    - uut - top256 (top256.v)
      - tg1 - top1 (top1.v)
      - tg2 - top1 (top1.v)
      - tg3 - top1 (top1.v)
      - tg4 - top1 (top1.v)
      - tg5 - top1 (top1.v)
      - tg6 - top1 (top1.v)
      - tg7 - top1 (top1.v)
      - tg8 - top1 (top1.v)
      - tg9 - top1 (top1.v)

No Processes Running

Processes: tg1 - top1
- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
- Implement Design
- Generate Programming File
- Configure Target Device
- Analyze Design Using ChipScope

top1 Project Status (12/18/2017 - 17:14:12)			
Project File:	Single_RadixFFT.xise	**Parser Errors:**	No Errors
Module Name:	top1	**Implementation State:**	Synthesized
Target Device:	xc5vlx110t-2ff1738	**• Errors:**	No Errors
Product Version:	ISE 13.1	**• Warnings:**	6 Warnings (0 new)
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values) [-]

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	709	69120	1%
Number of Slice LUTs	448	69120	0%
Number of fully used LUT-FF pairs	240	917	26%
Number of bonded IOBs	163	680	23%
Number of BUFG/BUFGCTRLs	1	32	3%
Number of DSP48Es	6	64	9%

Detailed Reports [-]

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Mon 18. Dec 17:14:11 2017	0	6 Warnings (0 new)	36 Infos (0 new)

```
    Maximum combinational path delay: No path found

=========================================================================

Process "Synthesize - XST" completed successfully
```

Figure 4.1 Simulation Screen of Proposed Design on Xilinx13.1 Design Suite.

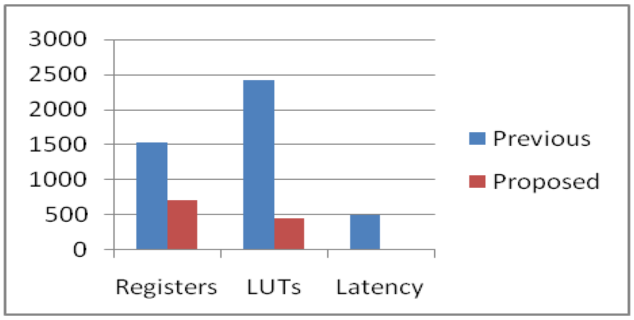


Figure 4.2 Comparison Chart proposed work with previous work.

V. CONCLUSION

Fixed Point FFTs in radix 2 are implemented in this work. The results are validated from the input signal and twiddle factor assigned in the test bench. The outputs obtained from verilog code are in close agreement confirming satisfactory results. The implementation and simulation of proposed FFT using Vedic multiplier has done on Xilinx 13.1 ISE design suite HDL programming platform. The comparison between different coding techniques for radix 2 method are studied implementation of FFT using Radix 2 and Vedic multiplier has found to be efficient. The use of “states” in place of port mapping reduces the device resource utilization to a great extent. Hence, similar improvement in code can lead to a large variation in resource utilization.

REFERENCES

[1] A. X. Glittas, M. Sellathurai and G. Lakshminarayanan, "A Normal I/O Order Radix-2 FFT Architecture to Process Twin Data Streams for MIMO," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 24, no. 6, pp. 2402-2406, June 2016.

[2] Shousheng He and M. Torkelson, "A new approach to pipeline FFT processor," Parallel Processing Symposium, 1996., Proceedings of IPPS '96, The 10th International, Honolulu, HI, , pp. 766-770, 1996.

[3] Y. Chen, Y. W. Lin, Y. C. Tsao and C. Y. Lee, "A 2.4-Gsample/s DVFS FFT Processor for MIMO OFDM Communication Systems," in IEEE Journal of Solid-State Circuits, vol. 43, no. 5, pp. 1260-1273, May 2008.

[4] S. N. Tang, C. H. Liao and T. Y. Chang, "An Area- and Energy-Efficient Multimode FFT Processor for WPAN/WLAN/WMAN Systems," in IEEE Journal of

Solid-State Circuits, vol. 47, no. 6, pp. 1419-1435, June 2012.

[5] P. P. Boopal, M. Garrido and O. Gustafsson, "A reconfigurable FFT architecture for variable-length and multi-streaming OFDM standards," 2013 IEEE International Symposium on Circuits and Systems (ISCAS2013), Beijing, pp. 2066-2070, 2013.

[6] K. J. Yang, S. H. Tsai and G. C. H. Chuang, "MDC FFT/IFFT Processor With Variable Length for MIMO-OFDM Systems," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 4, pp. 720-731, April 2013.

[7] M. Ayinala, M. Brown, and K. K. Parhi, "Pipelined parallel FFT architectures via folding transformation," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 20, no. 6, pp. 1068–1081, Jun. 2012.

[8] M. Garrido, J. Grajal, and O. Gustafsson, "Optimum circuits for bit reversal," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 58, no. 10, pp. 657–661, Oct. 2011.

[9] S.-G. Chen, S.-J.Huang, M. Garrido, and S.-J.Jou, "Continuous-flow parallel bit-reversal circuit for MDF and MDC FFT architectures," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 61, no. 10, pp. 2869–2877, Oct. 2014.