A RECONFIGURABLE VLSI ARCHITECTURE FOR MIXED RADIX FFT

[1]MUGA NAGESHWER, [2]YAKAIAH K

[1]Post graduate Student-Department of ECE, ACE Engineering College, Hyderabad, Telangana, India
[2]Associate Professor, Department of ECE, ACE Engineering College, Hyderabad, Telangana, India
E-mail:yakhoo4u@gmail.com, nagesmuga@gmail.com

Abstract- The squaring unit is an important one for different engineering branches like Digital Signal processing, Image processing and as well as the in the communication also. The low complexity and high speed hardware solution is essential one for now a day's communication system. The literature survey reports the either any one of the thing is taken into consideration for hardware solution like either low complexity or the high speed only. The proposed one aim is both factors are implemented simultationly using Vedic Mathematics. Direct computation of square of a given binary number consumes same amount of recourses of multiplication. Mathematical multiplication is an essential operation in any DSP calculation. In this paper, we propose an hardware architecture solution for the low complexity and high speed square unit using the Vedic mathematics principles.

Keywords- Binary number system, Square unit, Vedic Mathematics, VLSI architecture

I. INTRODUCTION

Fourier transformation is the basis for the majority of the digital signal processing applications. The Fourier transform is a principle analytical tool that has its roots in such diverse fields as linear systems, optics, probability theory quantum physics, antennas, and signal analysis. Even with the tremendous computing speeds available with modern computers, the slow computation of DFT found relatively few applications because of the exorbitant amount of computation time. However, with the development of the Fast Fourier Transform (FFT), many facts of scientific analysis have been completely revolutionized. FFT finds applications in biomedical engineering, mechanical analysis, analysis of stock market data, geophysical analysis, and the conventional radar communication field. The development of fast algorithms usually consisting of special properties of the interest to remove redundant or unnecessary operation of a direct implementation. The Discrete Fourier Transform) (k X of a complex sequence) (n x where • • 1 ,.... 2 , 1 , 0 , ~ • N k n can be defined as

$$X(k) = \sum_{n=0}^{N-1} x[n] \ W_N^{kn} \quad (1)$$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \ W_N^{-kn} \quad (2)$$

where,

$$W_N = e^{-j\frac{2\pi}{N}} \quad (3)$$

II. FFT USING FPGA

Vedic mathematics is mainly based on 16 sutras (or aphorisms) dealing with various branches of mathematics like arithmetic, algebra, geometry etc. Digital signal processing is one of the core technologies, in rapidly growing application areas, such as wireless communications, audio and video processing and industrial control. The number and variety of products that include some form of digital signal processing has grown dramatically over the last few years.

DSP has become a key component, in many of the consumer, communications, medical and industrial products which implement the signal processing using microprocessors, Field Programmable Gate Arrays (FPGAs), Custom ICs etc.

DSP techniques have been very successful because of the development of low-cost software and hardware support. For example, modems and speech recognition can be less Expensive using DSP techniques. DSP processors are concerned primarily with real-time signal processing. Real-time processing requires the processing to keep pace with some external event, whereas non-real-time processing has no such timing constrain

Fast Fourier transform (FFT) has an important role in many digital signal processing (DSP) systems. E.g., in orthogonal frequency division multiplexing (OFMD) communication systems, FFT and inverse FFT are needed.

The OFMD technique has become a widely adopted in several wireless communication standards.

Today, various FFT processors, such as pipelined or memory-based architectures, have been proposed for different applications. However, for long-size FFT processors, such as the 2048-point FFT, the pipelined

architecture would cost more area and power than the memory-based design. Hence, memory based approach as gained more and more attention recently in FFT processor designs for long-size DFT applications.

For the memory-based processor design, minimizing the necessary memory size is effective for area reduction since the memory costs a significant part of the processor. On the other hand, the FFT processor usually adopts on-chip static random access memory (SRAM) instead of external memory. The reason is the high-voltage I/O and the large capacitance in the printer-circuit-board (PCB) trace would increase power consumption for external memory.

III. PROPOSED ARCHITECTURE

Use of numerical methods is prevalent in most software algorithms. Such applications demand an efficient code for basic mathematical operations, one of them being multiplication. Real time systems demand instantaneous response to environmental variables and quick execution of taken decision. Multiplication algorithms find applications in Digital Signal Processing (DSP) for discrete Fourier transforms, fast Fourier transforms convolution, digital filters, etc. Therefore, any new multiplication algorithm opens up a new approach for improving existing schemes [2-6]. This calls for a 'time efficient' algorithm for 'multiplication' to improve processor speed. Proposed hardware architecture based on mixed radix FFT. The same architecture can be extended to N N. bit multiplication. In this paper, DFT of 16-samples are calculated using the DIT -FFT algorithm with proposed Vedic multiplier architecture. The samples of the data representation are shown in Fig 2.

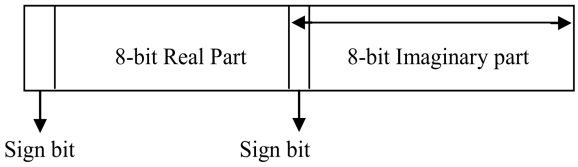


Fig 1.Register representation of sample value

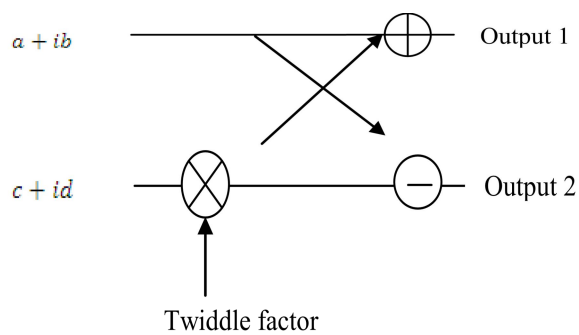


Fig 2.Butterfly Diagram of complex samples

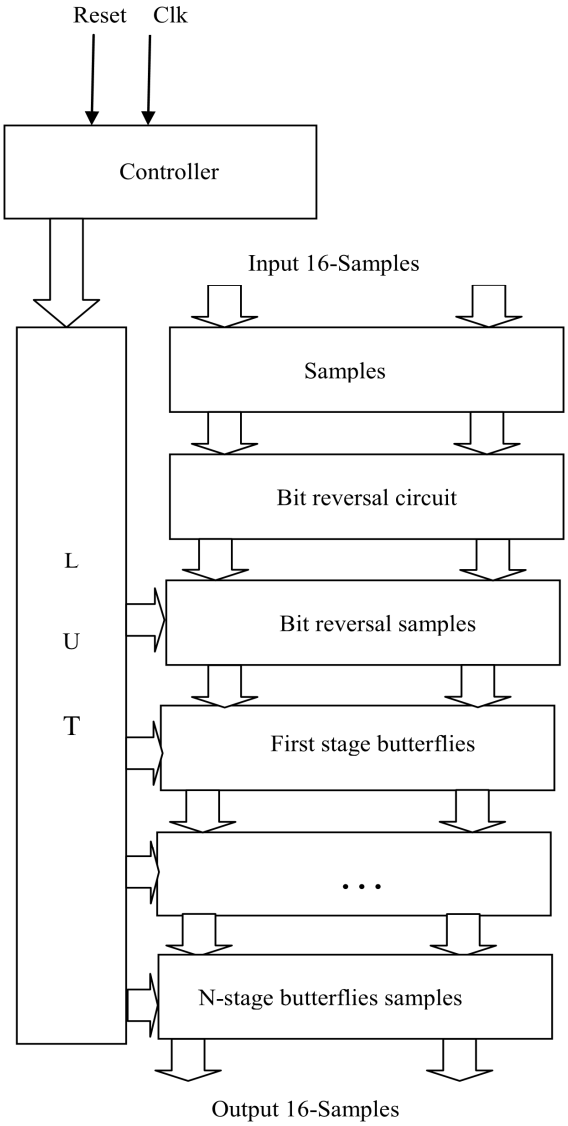


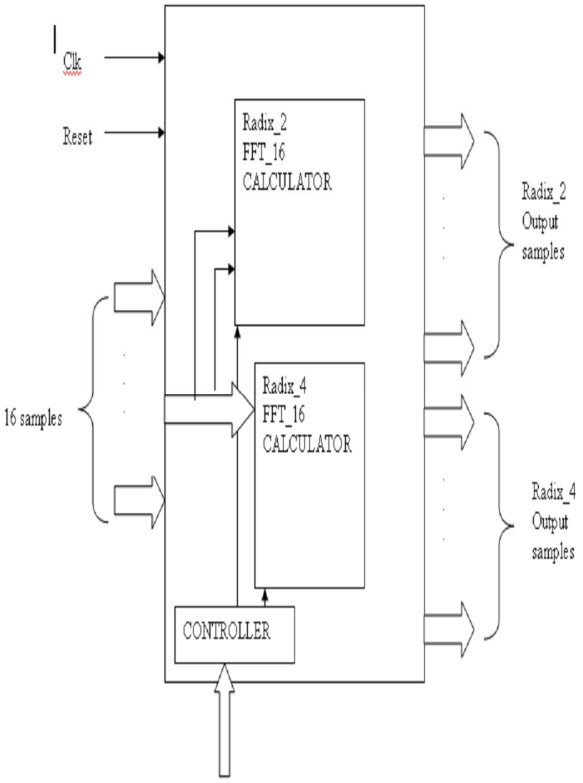
Fig 3. The proposed architecture for DIT-FFT calculation

The butterfly concept is one component used in FFT computation. This includes the multiplication, addition and subtraction operations as shown in the Fig 3. In this paper, the complex multiplication and addition operations of the butterfly shown in Fig 3 are implemented using Vedic mathematics procedure to obtain improved speed and minimum resource utilization.

As shown in Fig 4, the Clock and Reset are the inputs to the controller. The controller produces required signals for the look up table (LUT). The LUT contains the twiddle factors, these factors are accessed through the control signals derived from the controller.
The register set contains the 16 registers with 18-bits of information as shown in Fig 2. Depending on the control signal the bit reversal circuit generates bit reversed samples. The number of butterfly stages can be calculated by using the formula

s NoofSample ages NumberofSt 2 log. (4)

For the 16-samples, four stages of butterflies are required. The individual stage twiddle factors are accessed through the LUT concept.



Depending on the control signals the radix-2 and /or Radix-4 values can be calculated. If the controller value “00” nighter the radix-2 nor the radix-4 is calculated. If the controle value is “01” Radix-2 is calculated,”01” Radix-4 is calculated and “11” Radix-2 and Radix-4 values are calculated.

IV. RESULTS AND ANALYSIS

The 8 8 • bit multiplication operation in the normal procedure requires 64 multiplications and 77 additions. The proposed 8 8 • bit multiplier architecture of this paper requires 64 multiplications and 55 additions. The proposed multiplier architecture of this paper considerably reduces the number of additions. Table 1 shows the comparison of arithmetic operations of proposed multiplier architecture with normal multiplied architecture.

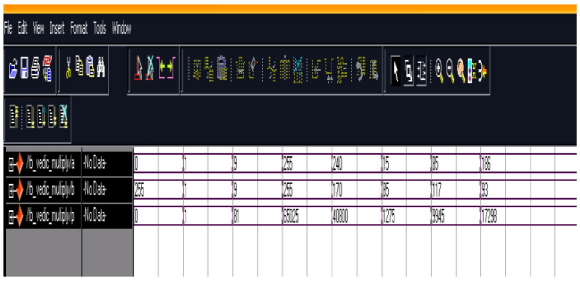


Fig 5.Simulation results for the 8X8 bit multiplication

Fig 5 Shows simulation results of 8 8 • bit (excluding sign bit) Multiplier using Vedic Mathematics principle (arrow multiplication method).

Fig 6 shows the simulation results of the complex multiplication operation)} (*) {(id c ib a. .Where c b a , ,and d are 8-bits. Fig 7 shows the simulation result of the 16-point DIT-FFT algorithm using architecture in Fig 4. As we discussed, the architecture shown in Fig 4 provides superior results as mentioned in Table 2 and Table 3.

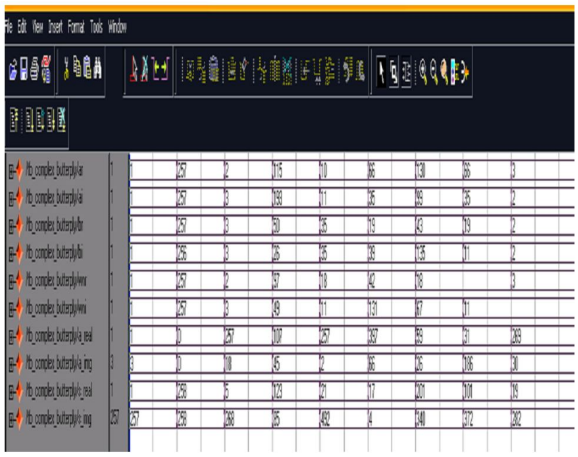


Fig 6.Simuation results for complex multiplication (8X8-bit)

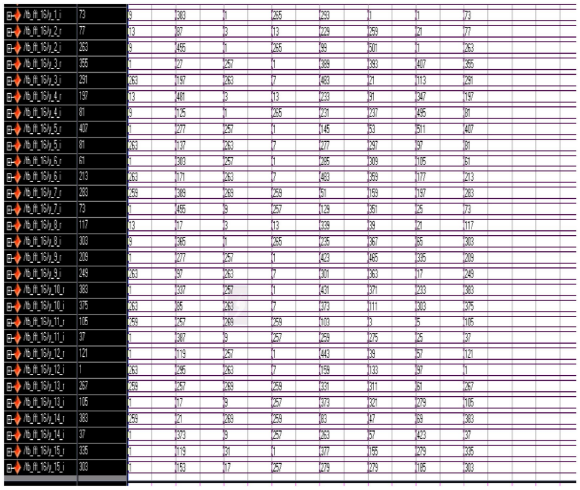


Fig 7.Simulation results for the 16-point FFT algorithm

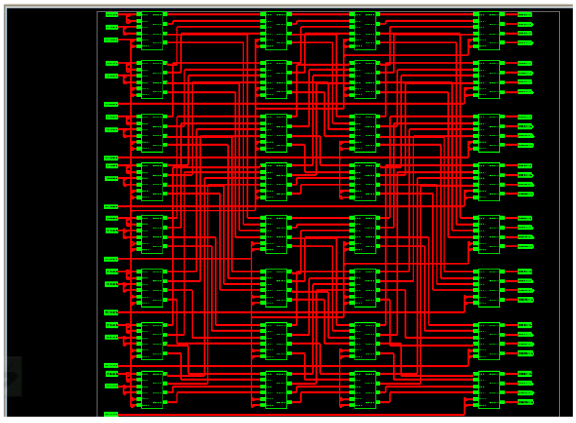


Fig 8. RTL schematics view for the 16-point FFT algorithm

V. TECHNOLOGY AND TOOLS

The architecture proposed in this paper has been authored in verilog HDL and its synthesis was done with Xilinx XST. Xilinx ISE Foundation 9.1i has been used for performing mapping, placing and routing. For behavioral simulation and place and route simulation, Modelsim10.d has been used. The Synthesis tool was configured to optimize for area of Vertex 2 pro, device xc2vp2, package fg256, speed:-6 [7-8].

CONCLUSION

Ancient Indian system of mathematics, known as Vedic mathematics, has been applied to various branches of engineering. In this paper an effort is made to use Vedic mathematics principles for the complex multiplier design which is used in DIT-FFT algorithm. The Fast Fourier Transform (FFT) is a critical block and widely used in digital signal processing algorithm. With the advent of semiconductor processing technology in VLSI System, it has enabled the performance of FFT design to increase steadily and applied in Portable application design. We have presented several algorithms for efficiently performing FFs of arbitrary length and dimension on CPUs. We choose the algorithm that provides the best performance for a given input size and hardware configuration. Our hierarchical FFT minimizes the number of memory access by combining operation with the FFT computation. The reported conventional radix-2 architecture observed to consume large chip area in conjunction with the increased number of hardware and power consumption corresponding to the in efficiency of the algorithm.

It is efficient and easier to use butterfly architecture in implementing the FFT computation. This butterfly computation can be done with two different approaches which are the decimation-in-time (DIT) and decimation-in-frequency (DIF). The difference between these architectures is that, DIT butterfly structure requires a multiplication process before addition is done. However, both algorithms require the same number of complex multiplication and addition operations. Work-steps include architecture Designing, Writing Verilog Code (Very high speed integrated circuit Hardware Descriptive Language), and simulating the code on "ModelSim 10.od.With the architecture implemented in this paper there has been reduction in number of additions required for DIT-FFT calculation. The proposed architecture of this paper is verified on FPGA and the results proved that this architecture has improved speed and minimum resource utilization. FFT could be instrumented to collect data on the huge space of transforms it evaluates, which could then be used to build more accurate models.

The existing FFT benchmarking infrastructure could be improved by detecting interruption by other system processes and re-running the affected results. Benchmarks could then be run on a much wider range of machines, under more controlled conditions, which would increase the accuracy of models built from the data. It might be possible to build a classifier that predicts whether a transform is likely, given some threshold, to be the fastest. The fastest is then selected from a subset of those that are likely to be the fastest, and thus the number of transforms that must be evaluated during calibration is reduced, while sacrificing little or no performance.

SFFT could be extended to multi-dimensional, multi-threaded, real, large (mega point and above) and arbitrary sized transforms. Additionally, support for other architectures such as POWER and Cell B.E. could be added. Code could be optimized between transforms in a library, which would reduce binary size, but there may be other effects.

REFERENCES

[1] Alan V. Oppenheim, Ronald W. Schafer and John R. Buck, Discrete- Time Signal Processing, Prentice Hal, second edition, 1999.

[2] Mandeep Singh Balwinder Singh Pawan Verma, Harpreet Kaur, “VHDL implementation of FFT/IFFT blocks for OFDM”, International

[3] Conference on Advances in Recent Technologies in Communication and Computing, 2009.

[4] James W. Cooley and John W. Tukey, “An algorithm for the machine calculation of complex fourier series”, Math Comput., pp. 297 – 301, 1965.

[5] Volnei A. Pedroni, Circuit Design with VDHL, MIT Press, ISBN 0- 262-16224-5, 2004.

[6] T. Starr, M. Sorbara, J. M. Cioffi and P. J. Silveramn, DSL Advances, Prentice Hall, 2003.

[7] Rd. J.; Ordaz-Moreno A.; Vite-Frias, J. A.; Romero-Troncoso, “VHDL

[8] Core for 1024-point radix-4 fft computation”, International Conference on Reconfigurable Computing and FPGAs 2005, ReConFig 2005, pp. 4 – 24, 9 2005.

[9] A. Álvarez-Marquina E. Martinez de Icaya C. Gonzaléz-Consejero, V.

[10] Rodellar and P. Gonzalez-Vilda, “A portable hardware design of a FFT algorithm”, Latin American Applied Research, 2007.

[11] Randy Yates, “Fixed-point arithmetic: An introduction”, 2009,

[12] http://www.digitalsignallabs.com.

[13] David Bishop, “Fixed point package user’s guide”.

[14] Frank Vahid, Digital Design with RTL Design, VDHL and Verilog,

[15] John Wiley and Sons, second edition, 2011.

[16] Altera, “Altera Corporation”, 2011, http://www.altera.com.

[17] Xilinx, Vertex II pro and Vertex II X platform FPGAs.

[18] Data sheets (http://www.xilinx.com/Verex II pro).

Table 4 Device utilization summary of the 16-DIT- FFT algorithm using the Vedic Multiplication

Design Summary		
Number of errors		0
Number of warnings		0
Logic Utilization	:	
Number of Slice Flip Flops	:	2 out of 66,560 1%
Number of 4 input LUTs	:	9,851 out of 66,560 14%
Logic Distribution	:	
Number of occupied Slices	:	5,131 out of 33,280 15%
Number of Slices containing only related logic	:	5,131 out of 5,131 100%
Number of Slices containing unrelated logic	:	0 out of 5,131 0%
Total Number of 4 input LUTs	:	10,115 out of 66,560 15%
Number used as logic	:	9,851
Number used as a route-thru	:	264
Number of bonded IOBs	:	434 out of 784 55%
IOB Flip Flops	:	434
Number of GCLKs	:	1 out of 8 12%
Total equivalent gate count for design	:	90,635
Additional JTAG gate count for IOBs	:	20,832
Number of BUFGMUXs	:	1 out of 8 12%
Number of External IOBs	:	434 out of 784 55%
Number of LOCed IOBs	:	0 out of 434 0%
Number of Slices	:	5131 out of 33280 15%
Number of SLICEMs	:	0 out of 16640 0%
Timing Summary	:	
----------	:	
Speed Grade: -5	:	
Minimum period	:	: 43.904ns (Maximum Frequency : 22.777MHz)
Minimum input arrival time before clock		: 5.149ns
Maximum output required time after clock		: 6.216ns

★★★