ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 10, October 2013

Area and Speed efficient Floating Point Unit design on Hybrid FPGA

Ankit Kumar Kusumakar, Utsav Malviya

Abstract— **Implementation of area and speed efficient design of Floating Point Unit on Hybrid FPGA is gradually replacing the conventional slower FPUs which have lower speed while computing complex calculations includes Digital Signal Processing. Existing FPGA devices are not optimized for floating-point computations, and for this reason, floating-point operators consume a significant amount of FPGA resources. Implementation of area optimized FPU on a hybrid Field Programmable Gate Arrays (FPGAs) with new feature multiplication and addition. For multiplication Peasant Multiplication algorithm, also known Ancient Egyptian multiplication and proposed tree adder for addition purpose which is designed with VHDL, synthesized using Xilinx ISE 9.2i Webpack, simulated using ModelSim simulator and then implemented on Xilinx Virtex 2E FPGA.**

Index Terms—**Field Programmable Gate Arrays (FPGAs), Floating Point Unit (FPU), Integrated Software Environment (ISE), Digital Signal Processing (DSP), Configurable Logic Block (CLB), Look up Table (LUT).**

I. Introduction

In modern Field Programmable Gate Arrays (FPGAs), coarse-grained elements such as processors, memories and DSPs are embedded into the fine-grained programmable fabric. A hybrid FPGA consists of a combination of coarse-grained and fine-grained reconfigurable elements. It provides a high-throughput and cost effective platform for designers to develop applications. Coarse-grained units are more efficient than fine-grained programmable logic for implementing specific word-level operations. However they are less flexible, and only benefit applications that can make use of them. Given this limitation, optimization of coarse-grained elements becomes a critical issue. Modern commercial FPGAs consist of commonly used coarse-grained elements such as DSPs and memories. Application which demands high performance floating point computation can achieve better speed and density by incorporating embedded floating point units (FPUs).

In this thesis work an optimized design of a 16 bit Peasant multiplier and proposed Tree Adder through efficient coding is done. The design uses ancient Egyptian Multiplication Algorithm and Tree Adder so as to increase the speed of multiplication and addition respectively, thereby reducing the number of logic blocks used to design it.

Ankit Kumar Kusumakar, *Student, Department of Electronics and Communication, Gyan Ganga Institute Of Technology And Science, Rajeev Gandhi Technical University, Jabalpur, India*

Utsav Malviya, *Asst. Professor, Department of Electronics and Gyan Ganga Institute Of Technology And Science, Rajeev Gandhi Technical University, Jabalpur, India*

A. FPGA Architecture For Floating Point

In general, FPGA-based floating-point application circuits can be divided into control and data path portions. The data path typically contains floating-point operators such as adders, subtractors, and multipliers, and occasionally square root and division operations. The datapath often occupies most of the area in an implementation of the application. Existing FPGA devices are not optimized for floating-point computations, and for this reason, floating-point operators consume a significant amount of FPGA resources. The control circuit is usually much simpler than the data path, and therefore, the area consumption is typically lower. Control is usually implemented as a finite-state machine and most FPGA synthesis tools can produce an efficient mapping from the Boolean logic of the state machine into fine-grained FPGA resources.

B. Hybrid FPGA

A hybrid FPGA consists of coarse-grained and fine-grained components, which are connected by routing tracks. Our fine grained fabric consists of an array of identical configurable logic blocks (CLBs), each containing N basic logic elements (BLEs). This architecture is similar to the Xilinx Virtex II slice. [3]

Coarse-grained units are more efficient than fine-grained programmable logic for implementing specific word-level operations. For example, an application which demands high performance floating point computation can achieve better speed and density by incorporating embedded floating point units (FPUs). Floating point adders/subtracters (FAs) and floating point multipliers (FMs) contain several basic functional elements such as barrel shifters, adders and multipliers. Word blocks (WBs) are used for the bitwise operation of the floating point number such as comparison, shifting, latch and logical operation. [3]

We know that a hard core ASIC will gives us a better throughput over the software based library routine if our application is specific if our application is defined. A Hybrid FPGA fabric consists of a combination of logic modules and a unique combination of interconnect. In general, a Hybrid-FPGA can be thought of as an application specific FPGA where there is some degree of field programmability and alterability. Regardless of the chosen implementation, a top view comparing a Hybrid-FPGA fabric with conventional FPGA shows the main difference is the amount of switching fabric required relative to the number of logic modules. [7] Architecture optimizations show that although high density FPUs are slower, they have the advantages of improved area, area-delay product, and throughput. In short the main problem is the flexibility, timing result and area consume.

II. BASIC IDEA OF IMPLEMENTATION

Proposing methodology to optimize coarse-grained floating point units (FPUs) in a hybrid field-programmable gate array (FPGA), where the FPU consists of a number of interconnected floating point adders (FAs) and floating point adders multipliers (FMs). The word blocks include registers and lookup tables (LUTs) which can implement fixed point operations efficiently. We propose to employ method based on Peasant algorithm for an optimized FM, proposed Tree adder for addition and study the area, speed and utilization trade off over a set of floating point benchmark circuits. We will then explore the system impact of FPU density and flexibility in terms of area, speed, and routing resources. Finally, we have planned to derive an optimized coarse-grained FPU by considering both architectural and system-level issues. This proposed methodology would be used to evaluate a variety of FPU architecture optimizations.

A. *Floating Point Adder*

For Floating Point Addition we have used Tree Adder, there are many cases where it is desired to add more than two numbers together. The straightforward way of adding together m numbers (all n bits wide) is to add the first two, then add that sum to the next using cascading full adders.

In Peasant Algorithm it is observed that the numbers which are being added comes in ascending order. So that the number of bits also improves for these numbers we proposed a new addition structure for fast addition and less number of gate counts required, as shown below:

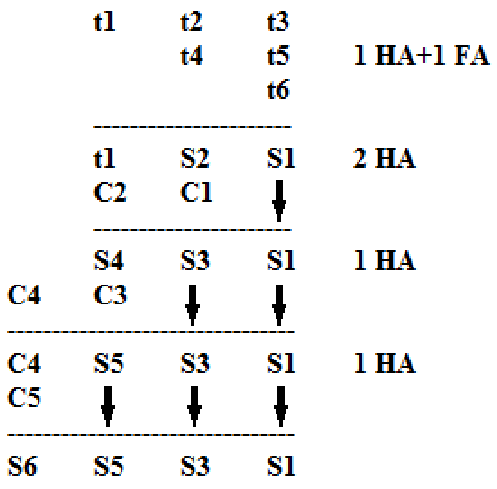

Figure1. Tree Adder Structure

Where, "t1 t2 t3", "t4 t5", "t6" are binary variables arranged in a tree form, Sx and Cx are Sum and Carry respectively.

B. *Floating Point Multiplier*

For Floating Point multiplier Peasant Multiplication also known as ancient Egyptian multiplication method is proposed for multiplying two numbers that does not require the multiplication table, only the ability to multiply and divide by 2, and to add is required.

It decomposes one of the multiplicands (generally the larger) into a sum of powers of two and creates a table of doublings of the second multiplicand. This method may be called mediation and duplation, where mediation means halving one number and duplation means doubling the other number.

In base 2 operation, for each '1' bit in the multiplier, shift the multiplicand an appropriate amount and then sum the shifted values. Depending on computer processor architecture and choice of multiplier, it may be faster to code this algorithm using hardware bit shifts and adds rather than depend on multiplication instructions, when the multiplier is fixed and the number of add required is small.

III. RESULT

Simulation result of 16-bit FPU design is shown below:

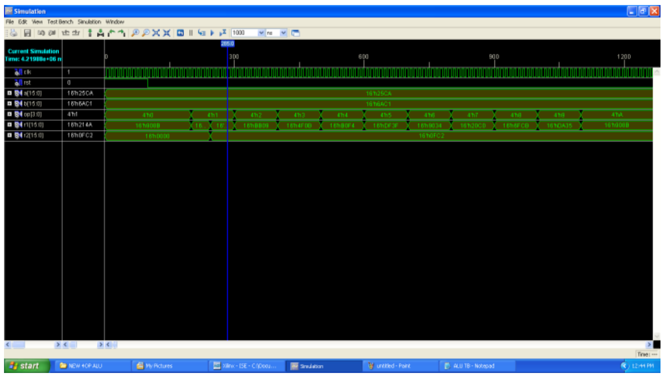

Figure2. 16-bit FPU Waveform

Comparative results are shown below:

Parameters	Base 1	Base 2	Base 3	Proposed
No. of Slices	576	564	564	128
No. of 4 bit LUT	--	1099	1099	321
IOBs	--	99	99	70
Power (mw)	--	151.31	--	333
Delay (ns)	2.95	--	--	10.642
Max Frequency (MHz)	338.98	50.180	50.180	93.967

As shown in above table our proposed multiplier is better in Area as number of slices required are less then base papers, and our speed is also batter as our Maximum frequency is higher than base papers.

IV. CONCLUSION

Proposed work estimated the number of slices and 4 input LUTs is decreased by using Peasant Multiplication Algorithm and Addition Tree. I have come to a conclusion that our design for Floating Point Unit in Hybrid FPGA in which we are using Peasant Multiplier for multiplication purpose and proposed Adder Tree requires lesser amount of area (gates) & delay (ns) as compared to the reference base papers. These two techniques are able to decrease the area up to a great extent by increasing the Power.

ACKNOWLEDGMENT

I am grateful to Mr. Utsav Malviya, my Project Advisor & Mentor, for his valuable guidance and encouragement. I was greatly benefited from the technical discussions shared with him which helped me in understanding the finer aspects of my work.

REFERENCES

[1] ChiWai Yu, Alastair M. Smith, Wayne, Philip, H. W. Leong and Steven J. E. Wilton, “Optimizing Floating Point Units in Hybrid FPGAs”, IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 20, JULY 2012, Pages 1295-1300.

[2] Z. Babic, A. Avramovic, P. Bulic, “An iterative Logarithmic Multiplier” SCIENCE DIRECT, MICROPROCESSORS AND MICROSYSTEMS, VOLUME 35, ISSUE 1, FEBRUARY 2011, Pages 23-33.

[3] Patricio Bulic, Zdenka Babic and Aleksej Avramovi, “A Simple Pipelined Logarithmic Multiplier”, IEEE COMPUTER DESIGN (ICCD), October 2010, Pages 235-240.

[4] Rathindra Nath Giri, M.K.Pandit, “Pipelined Floating-Point Arithmetic Unit (FPU) for Advanced Computing Systems using FPGA”, International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-1, Issue-4, April 2012.

[5] Chun Hok Ho, ChiWai Yu, Philip Leong, Wayne Luk, and Steven J. E. Wilton, “Floating-Point FPGA: Architecture and Modeling”, IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 17, NO.12, DECEMBER 2009, pp 1709-171.

[6] Chi Wai Yu, Alastair M. Smith, Wayne Luk, Philip H.W. Leong, Steven J.E. Wilton, “OPTIMIZING COARSE-GRAINED UNITS IN FLOATING POINT HYBRID FPGA”, IEEE, 2008, pp57-64.

[7] Mustafa Gok, “Enhanced-functionality multipliers”, SCIENCE DIRECT, 2 February 2008.

[8] Chi Wai Yu, “A tool for exploring hybrid FPGAs”, in Proc. FPL, 2007, pp. 509–510. [5] Siddhartha, Gopal Krishna, Bahar Jalali-Farahani, “A Fast Settling Slew Rate Enhancement technique for Operational Amplifiers”, 978-1-4244-7773-9/10/$26.00 ©2010 IEEE 2008, pp57-64.

[9] Alireza Kaviani and Stephen Brown, “Hybrid FPGA Architecture”, IEEE, 1996.

[10] VHDL Programming By Example doughlas perry, McGraw-Hill, 497 pages.

Ankit Kumar Kusumakar received the Bachelor Engineering degree from Electronics and Communication Engineering Department, Rajeev Gandhi Technology University, India in 2011.

Utsav Malviya received the M.tech.degree from, Embedded Systems, Electronics and Communication Engineering Department, Devi Ahilya University, India in 2009.