Design and Implementation of Embedded Complex Floating Point Hardware Accelerator

Darshini M B
Assistant professor, Dept. of Electronics and communication Engineering
Mysore, Karnataka, India

---***---

Abstract-*As circuit delays, reduction in power is also important design concern. A general idea is, the execution of large delays and power, by usage of co-processor in modern processor which executes the instructions in parallel with the main processor. Floating point can be loaded to a coprocessor to execute the part of complex operation with minimum power and delay. A pipelined, many input and many output arithmetic unit is typically handled to accelerate, to perform arithmetic floating point on complex operations. In our proposed Complex single precision Accelerated ALU (CAALU), the arithmetic unit has achieved a lower delay. Arithmetic module are designed in Verilog and are implemented on a FPGA device using Xilinx.*

Key Words: ALU, co-processor, single precision complex number, normalization.

1. INTRODUCTION

A cost efficient implementations are used in complex mathematical operations because it is not so familiar as that of the usage of real division, addition, multiplication and subtraction. Hence it is slow in the complex operation. The algorithms of operations in floating point implementation and operations in complex floating point on FPGA is hard. The enhancement of complex operations speed is will be helpful in the applications like multi antenna transceiver, GPS and signal processing. Arithmetic circuit which does the digital arithmetic operations has many applications in application circuits, co-processors [2]. As considering multiplication is important, where there is a use of less power signal application and speed. The various key factors for considering the floating point number system are computational capabilities required for the processor, application. The designers moved from the fixed numbers to single precision floating point due to wide range with the ability of numbers to represent a very large or very small.
In our approach, we are presenting an complex single precision accelerated ALU hardware for complex numbers which is in IEEE 754 binary 32 format contains all arithmetic operations [8]. IEEE 754 format single precision consists of three basic parts are sign, mantissa and exponent [3]. Sign indicates either a positive or negative number in 0 and 1 respectively [3]. Exponent of 8 bits see table 1 represent both positive and negative exponents. A 127 bias is added to exponent to store. Mantissa, is significand of 23bits comprises of fraction and leading digit represents the precision bits of the number [3].

TABLE-1 Single Precision Representation

32 Bits		
Sign	**Exponent**	**Mantissa**
1 bit	8 bits	23 bits

The rest part of the paper describes, Section II, the complex numbers arithmetic operations. Section III, the implementation of our single precision CAALU with efficient modified algorithms of arithmetic operations will improve latency. Section IV depicts the architecture of our CAALU. Section V presents the simulation results that have been simulated in Xilinx tool. Section VI presents the conclusion part.

1.1. Complex Number Mathematics

This section describes the complex algebra. A complex operation is written as:

$$C = A + i \times B \qquad (1)$$

Where A and B, real numbers, and i imaginary ($i = \sqrt{-1}$).
Let consider C1 and C2 are the two complex numbers;

$$C_1 = A_1 + i \times B_1 \qquad (2)$$

$$C_2 = A_2 + i \times B_2 \qquad (3)$$

Below shows the four basic operations using two complex numbers:

$$C_1 \pm C_2 = (A_1 \pm A_2) + i(B_1 \pm B_2) \qquad (4)$$

$$C_1 \times C_2 = (A_1A_2 + B_1B_2) + i(A_1B_2 + A_2B_1) \qquad (5)$$

$$C_1/C_2 = (A_1A_2 + B_1B_2) / (A_2^2 + B_2^2) + i\,[(A_2B_1 - A_1B_2) / (A_2^2 + B_2^2)] \qquad (6)$$

1.2. 32 Bit Complex Single Precision Arithmetic Unit

1.2.1. Adder / subtractor

In complex floating numbers division and multiplication operations are complicated than subtraction and addition. As given in equation 4, for the real and imaginary parts of the two complex numbers, that equation can be performed as two independent floating operations. In the mean while in order to result calculation of subtraction or addition of two floating numbers, exponents must be made equal in shifting the larger one. Considering section C, a combined adder-multiplier where subtraction and addition are handled efficiently by the block and the advantage is taken by our proposed CAALU.

1.2.2. Simple multiplier

Complex floating multiplication could be done by four real floating point multiplications plus two subtractions or summations of the type in equation 5 which is same as that of subtraction and addition operations. Using array multiplier implementation of a fast floating multiplier can be done in order to increase the speed. For this we have to distinguish exponent part from mantissa and some of the decisions must be made in order to shift the numbers with same power-two order. Instead of separate subtraction/addition or multiplication blocks, Multi-operational block called "Adder-Multiplier" will be shared by CAALU in order to reduce the area and power of the hardware.

1.2.3. Adder-multiplier module

Our proposed ALU adder-multiplier block mainly consists of two array multiplier and an adder blocks. This block is

$$\text{Out\_norm} = \text{input-A} \times \text{input-B} + \text{input-C} \times \text{input-D} \quad (7)$$

Where input-A, input-B, input-C and input-D are the floating point mantissa part.

The block diagram of adder-multiplier is shown in Figure 1. This pipeline block is fed with four 32 bit inputs. Exponent bits are added and normalized and sign bits are xored. This inputs are given to the prenormalizer in order to convert floating number into a predefined format i.e input-norm1, input-norm2, input-norm3, input-norm4(each of 23 bit) [1]. Now using an array multiplier we will multiply the inputs, inputs of array multiplier are zeros and input-norm1 and selection input was input-norm2. Then the output of array multiplier was out_margin(29 bits) and out1(23 bits) which is given as one of the input of mux. Another input of mux is input-norm2. Mux is mainly used to select whether operation is addition or multiplication. If it is addition output of mux is input-norm2 else out1 is output of mux. Now output of mux and output of margin are stored in register file which is used for further process. Then the output of register file is given to array multiplier again and input-norm3 and input-norm4 are another inputs of array multiplier. Now their output was sent to normalize and give output as output norm with 64 bit.

1.2.4. Divider

Unlike addition, subtraction, multiplication performing division operation is complicated function in ALU. Divider module consists of mainly four sub-blocks. Divident and divisor are the two 64 bit inputs of the divider module [4]. Then exponent bits are subtracted and normalized and sign bits are xored. This inputs are given to the preevaluator in order to convert floating point number into a predefined format i.e numerator and denominator (each of 52 bit). Now using array divider we will divide the inputs, and the quotient (23 bit) is stored in register file. Now their output was sent to normalize and give output as quotient with 32 bit.

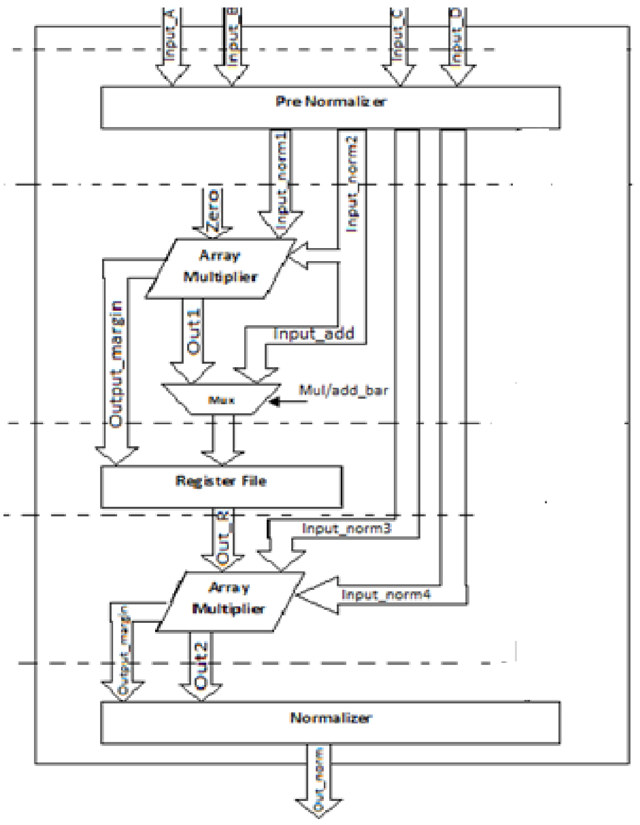

Fig -1: Block diagram of Adder-multiplier module

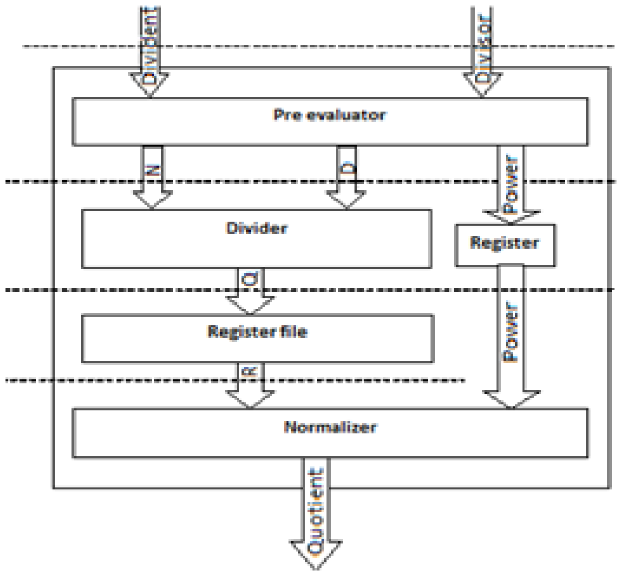

Fig -2: Block diagram of Divider module

2. ARCHITECTURE AND METHODOLOGY OF CAALU

In this section the new architectureof 32bit complex accelerated ALU with pipelined modules are implemented which includes all the arithmetic operations.

CAALU is composed of major four components are adder_multiplier, divider, registers and multiplexers.
The first block Adder-multiplier performs the addition, multiplication and subtraction operations [1]. Four 32 bit inputs is given to the Adder-Multiplier block. After performing this operations two 64 bit results are given to the divider module as inputs. At the division module, registers, adder-multiplier and division are all work together. Inputs to this two modules are provided by the multiplexers present in the input interface. Output interface selects the appropriate output at the final situation. Now by using both adder-multiplier and division module our CAALU performs all the complex floating operations by giving inputs as required and do the selection of operations like addition, subtraction, division, multiplication.

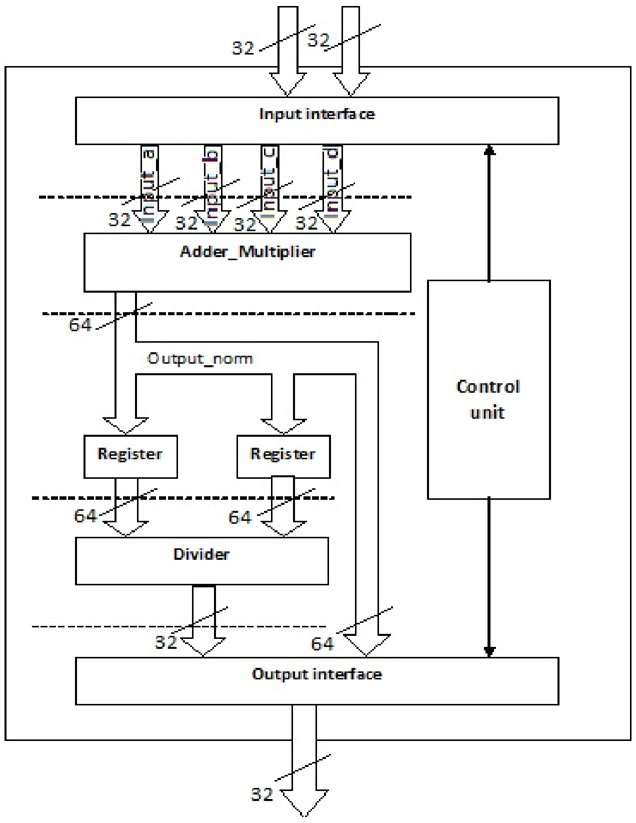

Fig -3: Block diagram of CAALU module

3. SIMULATION RESULTS

4. CONCLUSION

In this paper, a new 32 bit accelerated complex single precision ALU(CAALU) architecture is implemented successfully. The simulation with different modules is done in Xilinx.
In the speed critical complex number applications this presented CAALU can be accessed as a co-processor. CAALU architecture which is paralleled and pipelined and composed of blocks and are shared among the operations. The proposed CAALU will be integrated into NIOS II circuit where including subtract, add, divide and multiply operations for complex number, the four custom instructions will be added into the instruction set of the processor.

The 32 bit complex Adder-multiplier operation in Figure 4 and division operation in Figure 5, has been implemented in Xilinx tool.

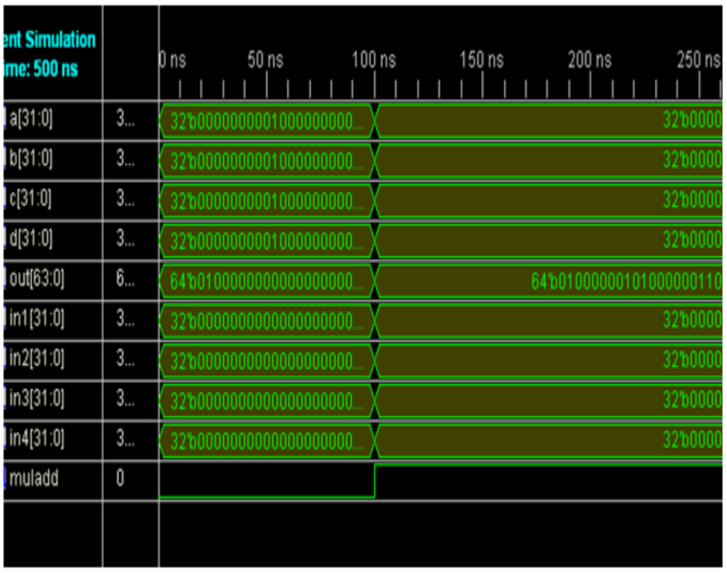

Fig -4: Output of 32 bit complex Adder-multiplier

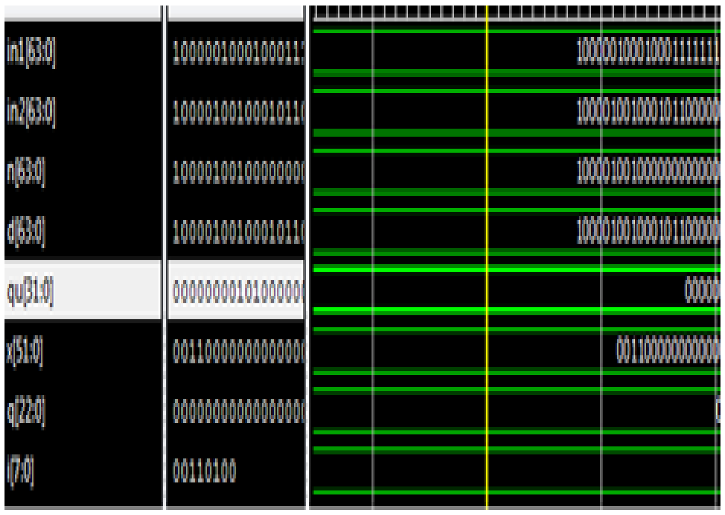

Fig -5: Output of 32 bit complex divider module
In the future work, we will be investigating the enhancement of the CAALU architecture in order to increase the speed of the

design by using architecture level techniques like pipelining or parallel processing. In the future, critical path delay of divider module can be reduced. In the meanwhile, the power consumption or lower area design can be optimized.

REFERENCES

[1] Rudolf Usselmann, “Open Floating Point Unit, The Free IP Cores Projects”.

[2] EdvinCatovic, Revised by: Jan Andersson, “GRFPU – High Performance IEEE754 Floating Point Unit”.

[3] David Goldberg, “What Every Computer Scientist Should Know About Floating-Point Arithmetic”, ACM Computing Surveys, Vol 23, No 1, March 1991.

[4] S.F. Oberman and M.J. Flynn, “Division algorithms and implementations”, IEEE Transactions on Computers, vol. 46, pp. 833-854, 1997.

[5] ANSI/IEEE Standard 754-1985, IEEE Standard for Binary Floating-Point Arithmetic, 1985.

[6] Guillermo Marcus, Patricia Hinojosa, Alfonso Avila and Juan NolazcoFlores, “A Fully Synthesizable Single-Precision, Floating-Point Adder/Subtractor and Multiplier in VHDL for General and Educational Use”, Proceedings of the 5thIEEE International Caracas Conference on Devices, Circuits and Systems, Dominican Republic, Nov.3-5, 2004.

[7] Carl Hamachar, ZvonkoVranesic, SafwarZaky “Computer Organization” 5th Edition, Tata McGraw-Hill Education, 2011.

[8] IEEE Standard 754 for Binary Floating Point arithmetic, IEEE, 1985.

[9] A book on “Verilog HDL: A Guide to Digital Design and Synthesis” by. Samir Palnitkar, second edition.