DESIGN AND IMPLEMENTATION OF COMPLEX FLOATING POINT PROCESSOR USING FPGA

Murali Krishna Pavuluri[1] and T.S.R. Krishna Prasad[2] and Ch.Rambabu[3]

[1]Embedded Systems, Gudlavalleru Engineering College, Gudlavalleru
[2]Associate professor, Department of E.C.E, Gudlavalleru Engineering College
[3]Assistant professor, Department of E.C.E, Gudlavalleru Engineering College

ABSTRACT

This paper presents complete processor hardware with three arithmetic units. The first arithmetic unit can perform 32-bit integer arithmetic operations. The second unit can perform arithmetic operations such as addition, subtraction, multiplication, division, and square root on 32-bit floating point numbers. The third unit can perform arithmetic operations such as addition, subtraction, multiplication on complex numbers. The specific advancement in this processor is the new architecture introduced for complex arithmetic unit. In general complex floating point arithmetic hardware consists of floating to fixed and fixed to floating conversions. But using such hardware will lead to compromise between accuracy and number of bits used to represent the fixed point equivalent of floating point numbers. The proposed architecture avoids that compromise and it is implemented with less number of look-up tables to save around 5500 logic gates. The complex numbers are represented using a subset of IEEE754 standard floating point format, 16-bits for real part and 16-bits for imaginary part. The floating point arithmetic unit works on 32-bit IEEE754 single precision numbers. The instruction set is specially designed to support integer, floating point and complex floating point arithmetic operations. The on-chip RAM is 8kBytes and is extendable up to 64kBytes. As the processor is designed to implement on FPGA, the embedded block RAMs are utilized as RAM.

KEYWORDS

Processor hardware, Complex floating point arithmetic hardware, look-up tables, IEEE754 standard floating point format, Single precision, Instruction set, On-chip RAM, and Embedded block RAM.

1. INTRODUCTION

The present day processor design technology is becoming very adaptive. Wide verities of architectures with optimized instruction sets are readily available. The present work proposes such a flexible architecture with an optimized instruction set. The architecture includes an efficient complex floating point arithmetic unit in order to perform the computations on the complex numbers with high degree of the accuracy. This processor can also be used as a co-processor to share the work load on the main processor in the applications where intensive computations are needed. Generally floating point representation is used in digital signal processing applications. However most of the times fixed point DSP processors can satisfy the requirements. One will choose the floating point architectures if they require high precision and dynamic range. Several significant factors need to be considered while selecting DSP architecture [1]. The proposed architecture can be called as hybrid architecture since it consists of floating point arithmetic units like DSP processors, but it does not consist of any instruction level parallelism, and any specialized or dedicated hardware units, and also does not consists of hardware looping mechanism. But it has load and store architecture similar to most of the DSPs.

DOI : 10.5121/vlsic.2013.4504

The floating point arithmetic unit present in the processor can compute addition, subtraction, multiplication, division, as well as square root of the numbers that are represented in IEEE754 standard single precision floating point format [2]. The complex floating point arithmetic unit present in the processor can compute the addition, subtraction, multiplication on the complex numbers that are represented in a subset of IEEE754 standard format with 16-bits (1-sign bit, 5-exponent bits, 10-mantissa bits) for real part and 16-bits for imaginary part (1-sign bit, 5-exponent bits, 10-mantissa bits) [3]. This is a completely new architecture when compared with the one proposed in [3], and it also requires a less number of look-up tables.

Complex number arithmetic is very common and important requirement in almost all DSP algorithms and hence most of the modern DSP processors are coming with complex number arithmetic modules. However DSP processors uses fixed to floating and floating to fixed conversions which will support a less dynamic range and less precision in order to compromise with the number of bits used to represent a fixed point equivalent of floating point counterpart while floating to fixed conversion. If a less number of bits are used to represent fixed- point equivalent of floating point numbers, obviously there will be a compromise for accuracy. To avoid that compromise and to achieve less number of look-up tables, it is better to choose the proposed architecture which can directly perform arithmetic operations on the complex numbers that are represented using 16-bit subset of IEEE floating point format. Some works also tried to perform complex arithmetic using resource sharing and pipelining concepts, by using a single floating point adder and floating point multiplier for processing of both real and imaginary parts to implement a typical DSP benchmark like scalable FIR filter [4].

Several architectures have been developed for IEEE floating point arithmetic, which were concentrated to reduce latency [5], Examples of complex domain floating point DSP processors are described in [6], [7] and [8]. The complex domain VLIW DSP core with microprocessor interface was presented in [6]. But the main theme of this work is implementing an efficient processor that can process integer and floating point numbers, as well as complex numbers also. Most of the design issues are having close similarities with the DSP processor design issues.

Paper Overview: The paper can be organized as follows:

Section 2 describes the critical design issues in the processor design and it also describes the Top level functional block diagram and the major modules present in it. Section 3 describes the Simulation results of various modules and synthesis report. Section 4 concludes the work and finally section 5 gives various possible future extensions of the work.

2. PROCESSOR DESIGN ISSUES AND IMPLEMENTATION

Common modules to construct any processor are data path and control path. The hardware that performs arithmetic and logic operations on data will come under data path. The hardware that controls the sequence of actions to be performed by data path, by issuing some control signals will come under control path. Several methods are in use to design control path. Some of the well known methods are using FSMs or using micro-coded programming. The functional block diagram of complex floating point processor is shown in Fig.1. Selecting the instruction set is the key design issue that will decide the interconnections among all modules [9] [10]. Selecting the op-code length will also be the critical factor. The present work uses 32-bit op-codes. Choosing the appropriate size of on-chip memories will improve the memory accessing speed but one must keep area constraints in mind. However FPGA vendors are providing optimized embedded memories that can be used very easily. The present processor consisting of 8KB on-chip RAM and is extendable up to 64KB. This facilitates the use of the processor for fast and real time needs. The register bank consists of 32 registers each of 32-bit wide and the two operands can be

fetched simultaneously. The bus multiplexer manages the data flow between register bank and arithmetic units, by multiplexing the buses which intern controlled by controller. The single precision floating point arithmetic unit is designed based on conventional algorithms similar to other floating point processors. The complex floating point arithmetic unit, which is major contribution of this work, uses a new architecture and is described in the following sub-sections.

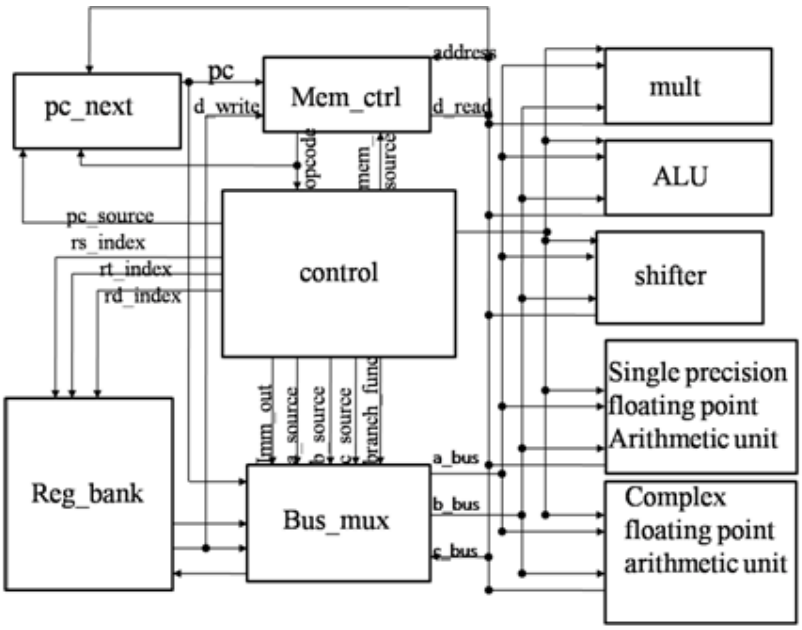

Figure.1 Block diagram of complex floating point processor

2.1. Data Path

Data path consists of integer arithmetic unit and a single precision floating point arithmetic unit and complex floating point arithmetic unit. Several design issues need to be considered while designing complex arithmetic unit. Complex numbers addition results another complex number and is performed by the hardware whose architecture is shown in Fig.2. The pre-normalization for addition/ subtraction unit, checks the presence of NaNs (not a numbers) in the inputs and aligns the exponents of the operands given to the add/sub unit which performs addition/ subtraction on fraction parts depending on the sign bits of the operands and actual operation that is intended to perform. Finally the post-normalization unit checks whether the result is a valid IEEE floating point number or not. If the result is near to a valid floating point number then it is rounded according to the rounding mode selected. Addition is explained in this section with an example.

$$a: 0\ 10001\ 0000000000 = (4)\ \textit{in decimal}$$
$$b: 0\ 10010\ 0000000000 = (8)\ \textit{in decimal}$$
$$c: 0\ 10001\ 1000000000 = (6)\ \textit{in decimal}$$
$$d: 0\ 10000\ 1100000000 = (3.5)\ \textit{in decimal}$$

The result of addition of the complex numbers is shown below and the simulation results for the same example are presented in Section 3.

$$X + jY = (0\ 10010\ 0100000000) + j(0\ 10010\ 0111000000) = 10 + j11.5$$

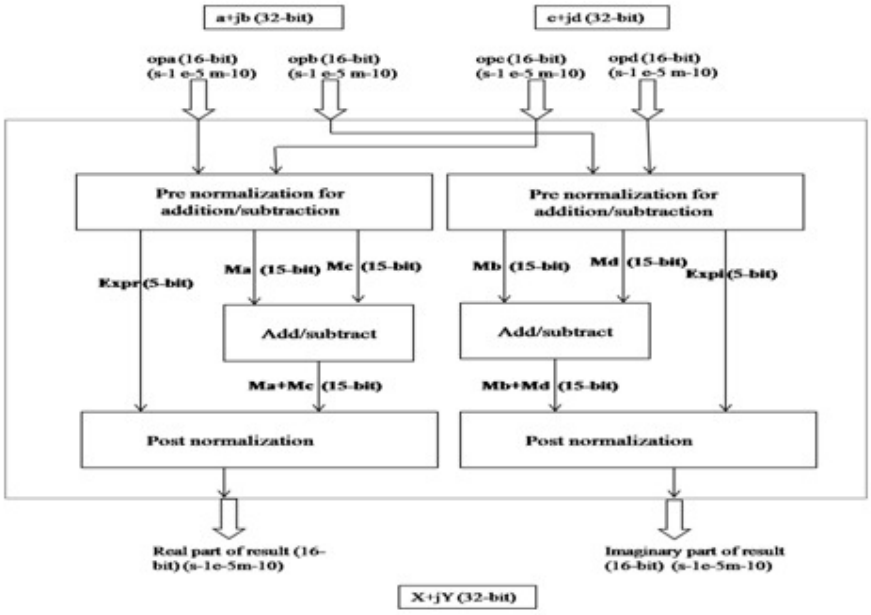

Figure.2 Complex floating point addition architecture

An example for complex number multiplication is given below and the simulation results are shown in section 3.

The result of multiplication is given below.

$a: 0\ 10001\ 1000000000 = (6)$ *in decimal*

$b: 0\ 10001\ 0100000000 = (5)$ *in decimal*

$c: 0\ 10010\ 0100000000 = (10)$ *in decimal*

$d: 0\ 10001\ 0010000000 = (4.5)$ *in decimal*

$$X + jY = (0\ 10100\ 0010110000) + j(0\ 10101\ 0011010000) = 37.5 + j77$$

The functionality of pre-normalization for multiplication is to check for NaNs (not a numbers) in the inputs and adding the required exponents. To perform the product of real parts (a*c), the pre-normalization unit adds the exponents of a and c and then subtracts the bias value 15. The multiplication unit then multiplies the mantissa or fraction parts of the two operands. The result of multiplication of fractions will be of 22-bit. Hence the post-normalization unit will truncate it and then rounds the result according to the rounding mode chosen, and finally packs the sign, exponent, mantissa bits of the result of multiplication of operands a and c. A similar procedure will be followed to evaluate the products (b*c), (a*d), (b*d). The products are then given to floating point addition /subtraction unit that is shown in Fig.2. The architecture proposed for complex multiplication is shown in Fig.3.

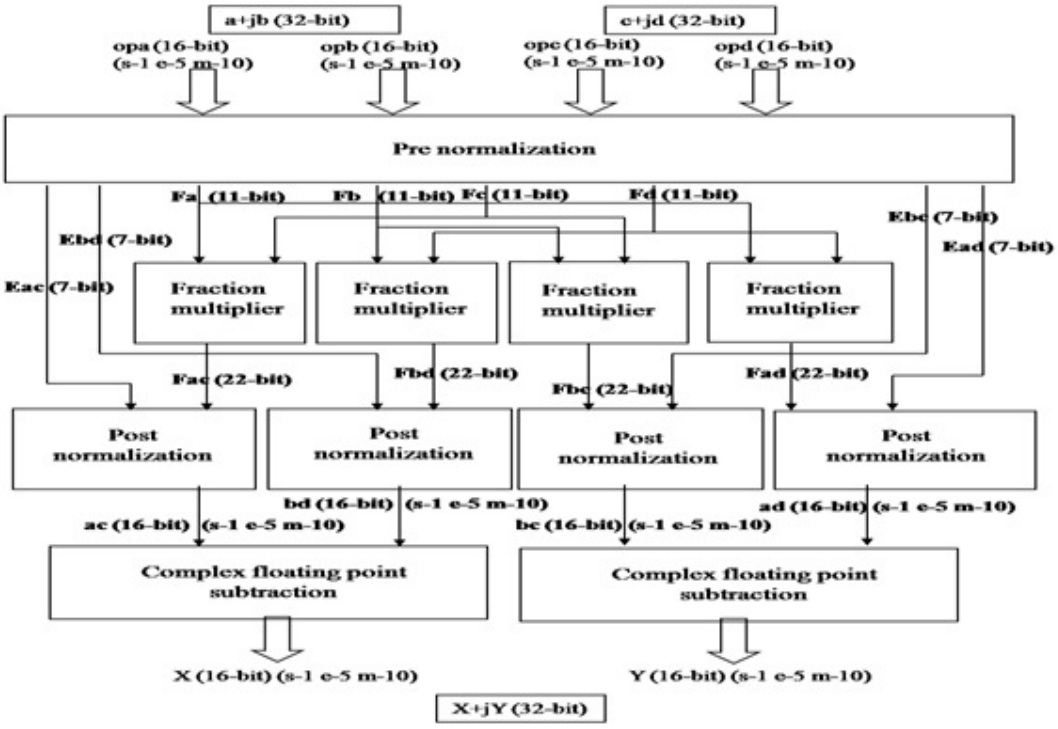

Figure.3 Complex floating point multiplication architecture

2.2. Control Path Design

Control path of a processor consists of a decoder which takes the op-codes from memory and decodes them and generates the necessary control signals that will enable the corresponding modules of data path to perform the required action. The addressing modes will specify the sources of operands on which the operation is to be performed. The sources of operands for ALUs are also selected by the decoder. There may be multiple decoders in the processors that support instruction level parallelism like VLIW architectures. Some additional hardware will also be used to avoid dependencies in such architectures. Op-code mapping is one of the major challenges after the instruction set selection. These two tasks will decide the performance and efficient utilization of the hardware. Efficient compiler design task will entirely depends on careful selection of instruction set and bench marking and op-code mapping steps. The proposed architecture supports two level pipelining.

The width of the program counter is 32-bit. The decoder itself will generate the necessary control signals to load the program counter with appropriate value. The control unit takes the 32-bit op-code from the memory which we call fetching of the instruction. The op-code is then decoded for generating various control signals to various modules. If the operands are present in registers and the result is to be stored in the register then the control unit generates source register index, target register index, destination register indexes to select the respective registers from the register bank. Consider an example, if the instruction being fetched is CFADD rd,rs,rt which is to perform complex floating point addition between the complex data present in source and target registers and places the result of addition in destination register. The 32-bit op-code corresponding to complex floating point addition is “000000 rs,rt,rd 00000 001010”. Where rs,rt,rd are the 5-bit indexes of respective registers in the register file. Fig.5 shows the simulation result for such example. When the decoder receives this op-code, it fetches the operands and enables the complex floating point arithmetic unit by specifying the operation to be performed by it. The

signal alu3_func[2:0] specifies the operation to be performed by the complex floating point arithmetic unit. If it is “000” means addition is performed on the available complex operands.

2.3. Register Bank

The Xilinx embedded dual port memories are used as register bank. The register bank consists of 32 registers and each of 32-bit wide. The decoder generates the read/write addresses according to the op-code. The register bank consists of two read ports and one write port in order to read the two operands simultaneously and write the result of operation in the specified destination register. Some registers in the bank are reserved for special purpose they can’t be used for general purpose.

2.4. On-chip RAM

The size of on-chip RAM is 8KB and can be extendable up to 64KB by simply setting the generic parameter named as block_count. Xilix Embedded block RAMs of 2kB each are used to construct total of 64KB memory. The block RAMs can be instantiated using ‘RAMB16_S9’ and each block RAM is initialized with zeros.

2.5. Single Precision Floating Point Arithmetic Unit

The single precision floating point arithmetic unit can perform addition, subtraction, multiplication, division and square root operations on the given 32-bit floating point operands based on the control signals given by the decoder and presents the result again in IEEE single

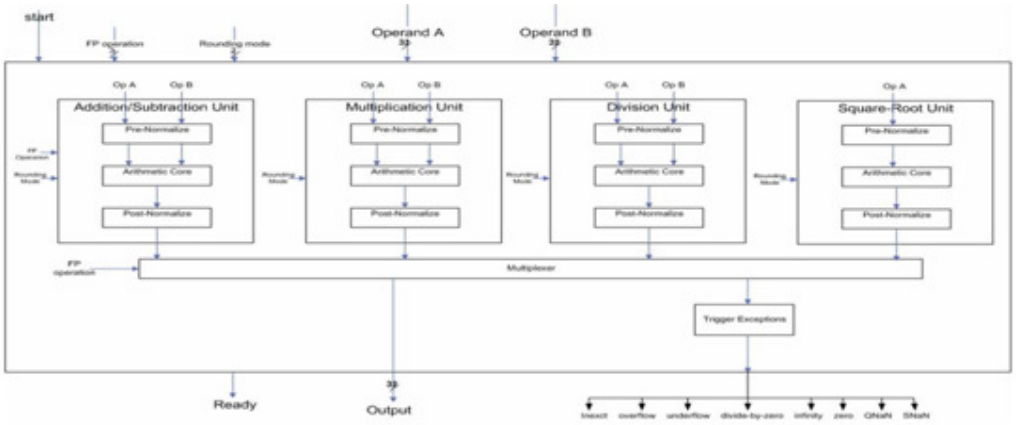

Figure.4 single precision floating point arithmetic unit

precision floating point format. In case any exceptions like inexact, overflow, underflow, divide-by-zero, infinity, zero, QNaN, SNaN occurs the corresponding signals will be raised. The values of Not a Number (NaNs), zero, infinity are defined in VHDL package. For example the value of Quite Not a Number (QNaN) is defined as "11111111100000000000000000000000" without sign bit. Similarly infinity (INF) is defined as "11111111000000000000000000000000". The single precision floating point unit is designed based on conventional floating point algorithms. The pre normalization for addition/subtraction and pre normalization for multiplication units performs the same work as in the case of complex floating point arithmetic unit except the length of the operands is 32-bit each, and the operands are real numbers. All the rounding modes defined in IEEE754 standard can be performed by this module.

3. SIMULATION AND SYNTHESIS REPORT ANALYSIS

This section describes the simulation results of the major modules of the design like decoder and complex floating point arithmetic units. It also analyzes the synthesis report summary of complex floating point arithmetic unit which is the major contribution of work.

3.1. Decoder Simulation Result Analysis

A 32-bit op-code is given as input to the decoder. The op-code includes the information of addressing mode, source register index (5-bit), target register index (5-bit), and destination register index (5-bit) in case of register addressing mode, and the operation to be performed. The decoder decodes this information and generates control signals to enable the corresponding modules. The signals alu1_func, alu2_func, alu3_func specifies the operation to be performed by the three arithmetic units present in the processor. Fig.5 shows the decoder simulation results for complex floating point addition.

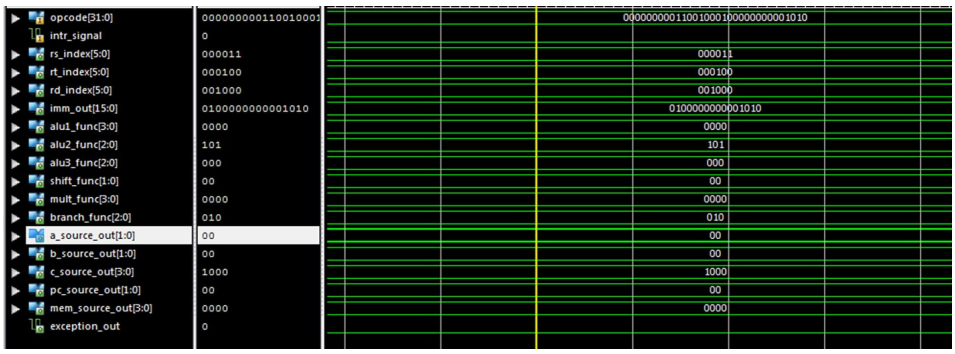

Figure.5 Decoder simulation results

3.2. Complex Addition Simulation Result Analysis

The signal fpu_op_i specifies the operation to be performed on the given complex numbers whether addition or subtraction or multiplication, which is connected to the decoder. The complex numbers are given in 16-bit floating point format as explained in section 2. The outputs of this module consist of real and imaginary parts which are in 16-bit floating point format, and the exception signals. Fig.6 shows the complex floating point addition results of complex floating point arithmetic unit.

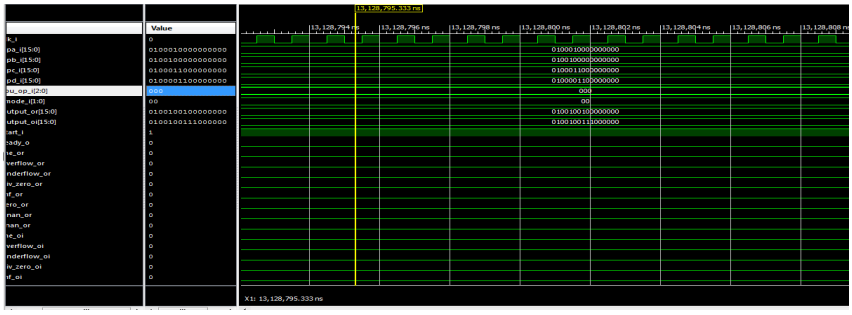

Figure.6 Complex floating point addition simulation result

3.3. Complex Multiplication Simulation Result Analysis

The summary of synthesis report for complex floating point arithmetic unit is given in TABLE1 and Fig.7 shows the multiplication results of complex floating point arithmetic unit.

Figure.7 Complex floating point multiplication simulation result

TABLE-1 Synthesis report summary of complex floating point arithmetic unit

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	2765	768	360%
Number of Slice Flip Flops	1086	1536	70%
Number of 4 input LUTs	5164	1536	336%
Number of bonded IOBs	120	124	96%
Number of MULT18X18s	4	4	100%
Number of GCLKs	1	8	12%

The synthesis report summary table shows that the number of 4-input LUTs required to implement the complex floating point arithmetic unit with proposed architecture are 5164, where as the existing architecture (using floating to fixed and fixed to floating conversions) proposed in [3] requires gate count of 36545 (sum of gate counts of 32-bit floating point adder, multiplier) to implement as ASIC. But one 4-input LUT is equivalent to 6-gates, so the gate count of proposed architecture is approximately 30984. It is obvious that around 5500 gates were saved with the proposed architecture without compromising at dynamic range and precision.

4. CONCLUSION

This work presents a complete processor hardware that includes three arithmetic units. It can process integer data and 32-bit floating point data as well as complex data that is represented using subset of IEEE 754 floating point format. The architecture of complex floating point arithmetic unit presented here is a completely new architecture that will avoid the compromise of using limited number of bits to represent fixed point equivalent of floating point numbers and there by limited dynamic range. The processor presented here can be used in real time applications and also can be used as a co-processor along with a main processor.

5. FUTURE WORK

There are many possibilities to extend this work. Some of the ways are mentioned in this section. An efficient compiler can be developed for this processor. This processor can be used as a co-processor in the implementation of several DSP bench mark algorithms which need complex computations. Some additional modules can be included to still enhance the advanced features and to develop the complete system on same platform. The ILP (instruction level parallelism) or

VLIW architecture or SIMD or superscalar concepts can be introduced for the same data path modules. The same architecture can be implemented for the complex numbers that are represented using other subsets of IEEE floating point formats or entirely new floating point formats. One can also concentrate on common VLSI optimization parameters like area, power consumption, speed.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their constructive comments and suggestions.

REFERENCES

[1] C. Inacio and D. Ombres, "The DSP decision: Fixed point or floating?," IEEE Spectrum, vol. 33, no. 9, pp. 72–74, Sep. 1996.
[2] IEEE Standard for Floating-Point Arithmetic, IEEE Std 754-2008, Aug. 29, 2008, pp. 1–58.
[3] Nadav Cohen and Shlomo Weiss," Complex Floating Point—A Novel Data Word Representation for DSP Processors" IEEE transactions on circuits and systems—I, VOL. 59, no. 10,pp.2252-2262 oct2012.
[4] Allison L. Walters, "A Scaleable FIR Filter Implementation Using 32-bit Floating-Point Complex Arithmetic on a FPGA Based Custom Computing Platform," M.S. thesis, Dept. Electrical. Eng., Virginia,1998.
[5] A.Beaumont-Smith,N.Burgess,S.Lefrere,andC.C.Lim,"Reduced latency IEEE floating-point standard adder architectures," inProc. IEEE Symp. Comput. Arithmetic, 1999, pp. 35–42.
[6] P. S. Paolucci, "Complex Domain Floating Point VLIW DSP With Data/Program Bus Multiplexer and Microprocessor Interface," U.S. Patent 7 437 540, Oct. 14, 2008.
[7] S. Katayanagi, "Complex Vector Operation Processor With Pipeline Processing Function and System Using the Same," U.S. Patent 20030009502, Jan. 9, 2003.
[8] R. G. Cox, M. W. Yeager, and L. L. Flake, "Single Chip Complex Floating Point Numeric Processor," U.S. Patent 4996661, Feb. 26, 1991.
[9] Dake Liu "Embedded DSP Processor Design" application specific instruction set processors, ELSEVIER.
[10] Mazen A. R. Saghir "Application Specific Instruction-set architectures for embedded DSP Applications" Ph.D thesis, Dept., of electrical and computer engineering, university of Toronto, canada.
[11] Author: Jidan Al-eryani, "Floating point unit" http://opencores.org/
[12] Auther: Rhoads,steve, "plasma-most MIPS I (TM): opcodes:: overview", http://opencores.org/

Authors

Murali Krishna Pavuluri1 is M.Tech student from Gudlavalleru engineering college, Andhra pradesh. He has completed B.Tech from Sasi institute of technology and engineering.

T.S.R Krishna Prasad2 is Associate professor in E.C.E department of Gudlavalleru Engineering College. He has about 10 years of teaching experience and presented papers in several international and national conferences and published papers in international journals.