Modeling and Execution of Floating Point Parallel Processing Operation for RISC Processor

Divya. D, Balasaraswathi. R, Harini kalyani. M, Vivek Anand. I

Abstract: ***The development of processors with sundry suggestions have been made regarding a exactitude definition of RISC, but the prosaic concept is that such a computer has a small set of simple and prosaic instructions, instead of an outsized set of intricate and specialized instructions. This project proposes the planning of a high speed 64 bit RISC processor. The miens of this processor consume less power and it contrives on high speed. The processor comprises of sections namely Instruction Fetch section, Instruction Decode section, and Execution section. The ALU within the execution section comprises a double-precision floating-point multiplier designed during a corollary architecture thus improving the speed and veracity of the execution. All the sections are designed using Verilog coding.***

Monotonous instruction format, cognate prosaic-purpose registers, and pellucid addressing modes were the other miens. RISC exemplified as Reduced Instruction Set Computer. For designing high-performance processors, RISC is considered to be the footing. The RISC processor has a diminished number of Instructions, fixed instruction length, more prosaic-purpose register which are catalogued into the register file, load-store architecture and facilitate addressing modes which make diacritic instruction execute faster and achieve a net gain in performance. Thus the cardinal intent of this paper is to consummate the veridicality by devouring less power, area and with merest delay and it would be done by reinstating the floating-point ALU with single precision section by floating- point double precision section. Video processing, telecommunications and image processing were the high end applications used by architecture.

Keyword: Double precision , RISC, Floating –point ALU,Instruction decoder.

I. INTRODUCTION

The floating-point operations were used in intensive digital signal processing (DSP) applications . Addition and subtraction were the basic floating-point operations. High precision operations require dynamic range and therefore require efficient floating-point processors.

Revised Manuscript Received on February 07, 2020.

D Divya, Undergraduate Student, Department of Electronics and Communication Engineering, National Engineering College, Tamil Nadu, India.

R Balasaraswathi, Undergraduate Student, Department of Electronics and Communication Engineering, National Engineering College, Tamil Nadu, India.

M Harini Kalyani, Undergraduate Student, Department of Electronics and Communication Engineering, National Engineering College, Tamil Nadu, India

I. Vivek Anand, Assistant Professor, Department of Electronics and Communication Engineering National Engineering College, Kovilpatti, India.

The main reason to choose the field-programmable gate array (FPGA) to implement arithmetic unit are its high performance, high integration density, and low price.

The predilection of design is positioned on accessible technology. Hardware, software, and their design alternatives emerge with technology.

Additionally the melioration in designing processors and exceptional system were also be designed.

For flourishing technology RISC architecture is a reverberation and it is amassing of philosophy from CISC designs. The main curs of processors were small and slower memories, to resolve this stile RISC processors were designed which were mainly accent on software with scant addressing modes. Large number of register, executed at single clock cycle, load, store operation which were used to access memory and easy pipelining these were all the other dominant miens of RISC[z^1].

The VLSI applications were becoming revolutionarily intricate and variegated due to continuous ameliorations. Higher precision, dynamic range and speed levels were demanded by digital signal and speech-processing for betterment. The floating-point arithmetic unit is the only one solution to fulfill bugging. Hence, the floating-point arithmetic unit was integrated parallel with fixed point unit.

The low power consumption and operating at high speed were the miens of this processor. The footing for designing high performance processors considered to be the RISC processor which have the capability to make individual instruction beheaded faster and to fulfill a net gain in performance. Thus the number of transistor required for RISC were very less which leads to occupies less area compared to CISC processor. Load-store instruction only used to access the memory no the arithmetic unit neither the logic unit nor the IO instructions cannot access the memory directly which acts as the main key to make single clock execution of instruction[^{2}].

In instruction set using fewer instructions which tends to produce the optimized compilers. Embedded and many portable applications uses this concept..

II. FLOATING POINT WITH DOUBLE PRECISION

The representation of numbers that would be overlarge or too small as integer which construed by floating-point. Comparing fixed point representation, floating-point representation have high resolution and veridicality which used in many applications one among that is signal processing.

The smallest change which will be represented in floating-point representation is named precision. The meaning of precision implies closeness or accuracy. There are two types of precision they are[3]

i) Single precision

ii) Double precision

Compared to single precision, double precision is assortment of numbers that has more precision (that is, more digits to the proper of the decimal point). The term construes that is some things of a contradiction because the precision isn't really double. The fact is that a double-precision number uses twice as many bit as a single precision were the word double is derived. For illustration, if 32-bit number required for single precision then double precision needed 64-bits long.

Thus the range of magnitude and the veridicality get increased by the extra bits. The pattern which the program is using to represent floating-point values which confide were the range and precision get increased. IEEE floating-point format was emblematical format referred by the most of computers[4].

The 64-bit word is the standard representation for double precision floating point which numbered from 0 to 63 left to right as shown in figure1.

Fig 1 Representation of Double Precision

Sign (1)	Exponent (11)	Mantissa (52)

Moving to Double-precision would potentially improve the accuracy of the calculations for games as the amount of processing per pixel goes up, but right now it's not worth the performance impact to get a slightly more accurate result.

The legitimate real-world applications that use Double Precision tend to be scientific in nature, they are working with the very large or very small and a high amount of calculation with compounding errors and hence double-precision makes a lot of sense to reduce that error component. SIMD floating-point instructions are used in modern graphic processors, multimedia processors, and general-purpose processors with multimedia extension.

III. RISC

It is known as Reduced Instruction Set Computer. It is a kind of microprocessor that features a limited number of instructions. They can execute their instructions in no time because instructions are very small and straight-forward.

RISC chips require fewer transistors which make them cheaper to style and produce. In RISC, the instruction set contains simple and basic instructions from which more complex instruction are often produced..

In this instructions are register based and deportation of data takes place from register to register.

It has small set of instruction with fixed format. Data transfer occurs between register to register. Instruction type must be register based. The memory access is less and which it includes single clock cycle. The RISC processors are used mainly to increase the execution speed of the system. RISC is employed in high-end applications like video processing, telecommunications and image processing.

IV. PROPOSED SYSTEM

This proposed system is much better than the existing one. In the existing system, it consists of floating-point ALU with single precision but in this paper, we introduced the double-precision floating-point ALU with low power and lower area consumption which increases and improves the approximation, accuracy and execution speed compared to the existing one.

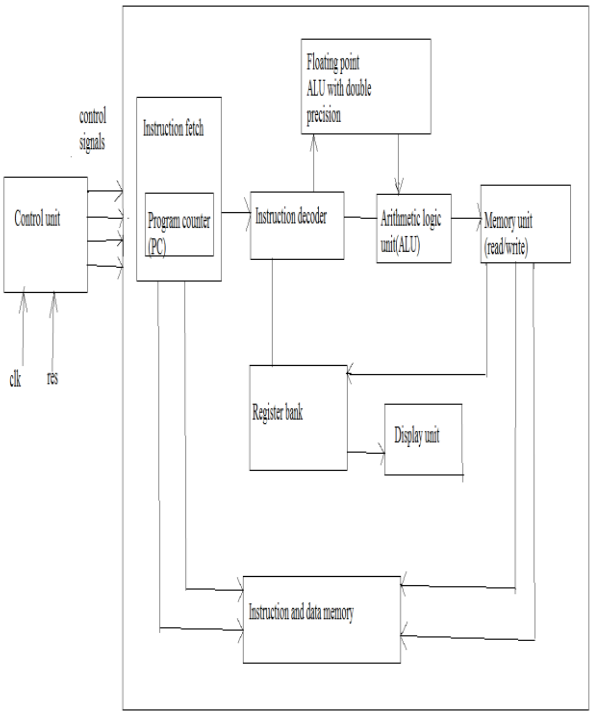

Fig 2 Architecture of RISC with double precision floating point unit

Fig 2 pageants the proposed system consists of 8 major blocks such as instruction-fetch, execution-unit, floating-point unit with double precision, memory-unit, control unit, instruction-decoder, register bank and instruction, and data memory.

a) Instruction-fetch

Instruction unit comprises of program counter with multipliers and adders. The repository area during in a computer processor that consists of address (location) of the instruction being executed at the continued time called a program counter. When each instruction fetched, it increases its stored value by 1. After each instruction is fetched, the program counter denotes the next instruction in the sequence. It administered the address of the instruction fetch. The jump or branch address was used then the address of the next instruction is obtained with the help of the control unit.

b) Arithmetic Logic Unit

The ALU is also called an execution unit [5], this unit comprises of arithmetic operations, relative operations, shifting operations and logic operations. The arithmetic operations comprise of addition, subtraction, multiplication, and division.

The relative operations such as comparison of values that it is checking the conditions such are greater than, less than, greater than equal, less than equal, equal to and not equal to. Shifting operations such are right shift, left shift, rotate left and rotate right. Then the logical operations include OR, AND, XOR, NOR, NAND, and XNOR.

c) Floating point unit with double precision

This unit undergoes the manipulations in floating-point numbers .as like as an execution unit it consists of arithmetic operations and logical operations. The arithmetic operations are addition and subtraction.

i) Floating-point addition with double precision

There are three major steps they are 1) Compare the sign bit (1-bit).2) Compare the exponent bits (11-bits), If the sign and exponents of the two inputs are the same then the final step is to add the mantissa part (52-bits) else assign the greater exponent value(11-bits) to the other exponent value. In this, our contribution is to reduce power and area consumption of the addition of floating-point with the double-precision block in the floating-point unit block which leads us to reduce the overall power and area consumption compared to existing work[6].

ii) Floating-point subtraction with double precision

In addition, floating-point subtraction with double precision also has the following three steps they are the first two steps are similar to the addition. Then the third step is to subtract the mantissa part.
As same as that of addition block subtraction comprises of three steps

d) Memory unit

A memory unit is nothing but a storage area with enough memory space required a system with 2^6 memory spaces. Memory unit interfaced with ALU and floating-point ALU, which used to store the final values of arithmetic operations done using execution unit that it is ALU and floating-point unit.

e) Control unit

The control unit plays vital role which controls the all over blocks by using control signals with clock and reset. This control unit decides which block that it is which operation to be performed.

f) Instruction decoder

All the internal control lines are the instruction decoder to convert op-code bits into settings. The operands offends a literal, the register address or program address ,which can be retained by instructions.

g) Register bank

The programmable registers used by assembly language programmers uses the register bank. It is observed as hardware equivalent of software array. The register bank comprises of ports for reading and writing data in inured an index. For reading and writing data have similar interfaces in main memory and caches. In register bank register number used to represent the index of the registers. In main memory, memory address used as an index and in cache a portion of memory address is used as an index. Thus the register bank was made up of BRAM[7].

Write port

The write port can afford write ingress to a tabbed word. There are 2 or 3 input signals such are one is to denote the index and another one is to enable the write operation.

Read port

The read port can afford write ingress to a tabbed word. There are 2 or 3 input signals such are one is to denote the index and another one is to enable the read operation. If the output is en-list towards bus which is shared with other devices then the read enable control input is needed.

h) Instruction and data memory

Instruction and data memory is nothing but read only memory (ROM).It is non-volatile memory and it used to store the data which can be permanently stored.

V. RESULT AND DISCUSSION

MEMORY BLOCK

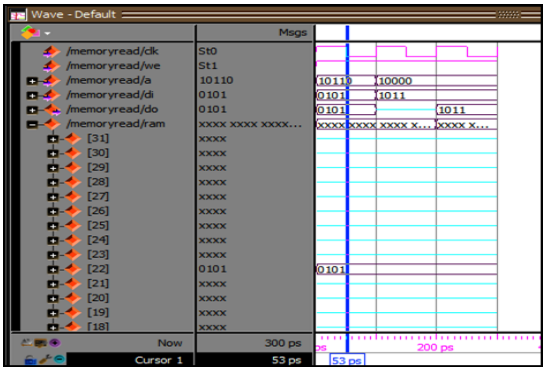

Fig 3 Waveform of memory block

As shown in Fig 3 we,a,di,clk,ram are inputs and d0,ram were the output. we=1(enable write),a=10100(location),di=0101 and so it write 0101 in denoted location (a) the output bis d0=0101 and in ram the ouput is stored in specified location.

ALU BLOCK

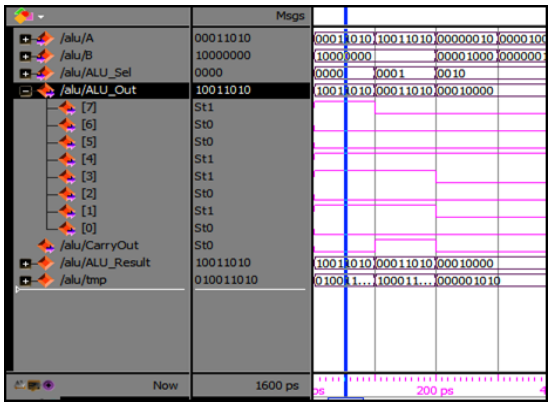

Fig 4 Waveform output of ALU block

As shown in Fig 4 A,B and ALU_Sel were the inputs and ALU_Out and ALU result are the outputs.
A=00011010,B=1000000 and ALU_Sel=0000.
The ALU_Sel values denotes addition.The output in ALU_ Result =1001101.

ALU AND MEMORY INTERFACE

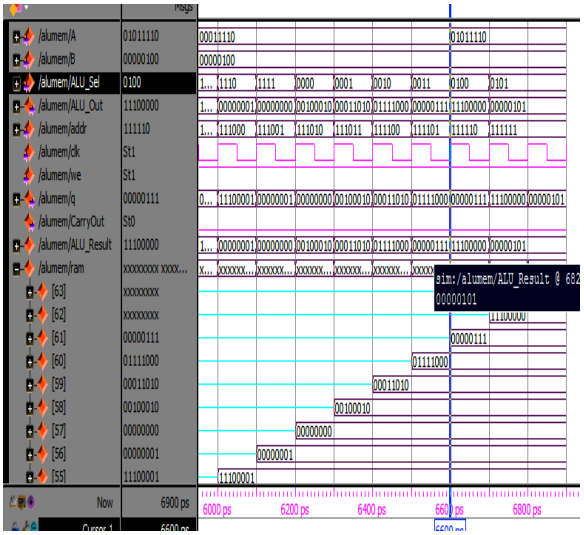

Fig 5 Waveform of ALU and memory interface

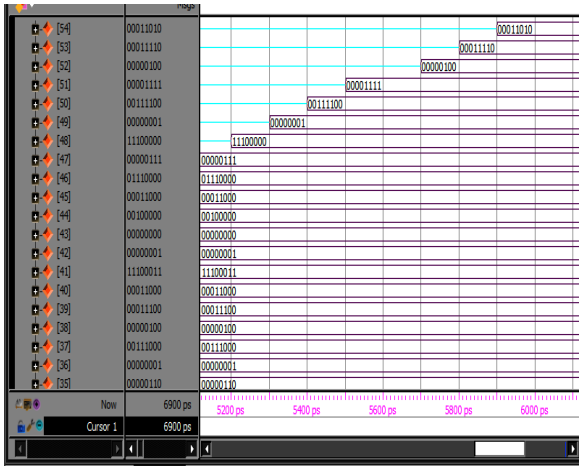

Fig 6 Waveform of ALU and memory interface

As shown in Fig 5 and Fig 6 ALU and memory get interfaced. The inputs are given, after arithmetic operations performed it full store the result in the specified location.

FLOATING POINT ADDITION WITH DOUBLE PRECISION

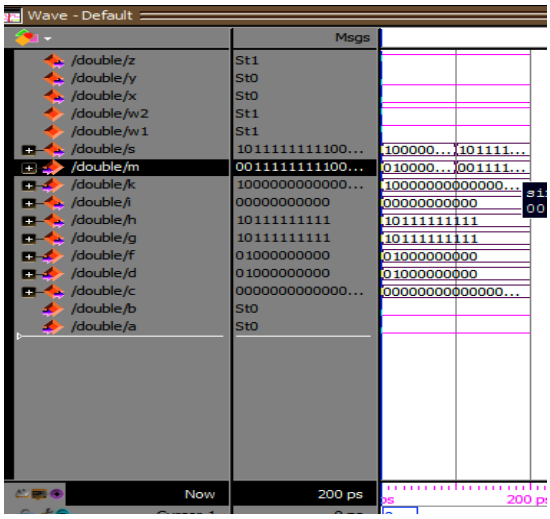

Fig 7 Waveform of floating point addition with double precision.

Fig 7 pageants the simulated output of Floating point addition with double precision a(sign), b(sign), e(exponent), f(exponent), m(mantissa), k(mantissa) are the inputs and s(sum) and c(carry) are the outputs

FLOATING POINT SUBTRACTION WITH DOUBLE PRECISION

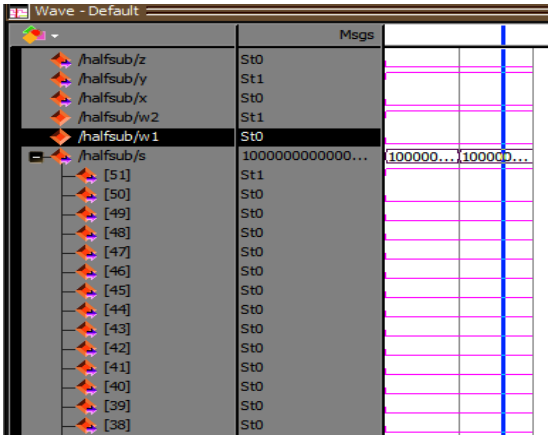

Fig 8 Waveform of floating point subtraction with double precision

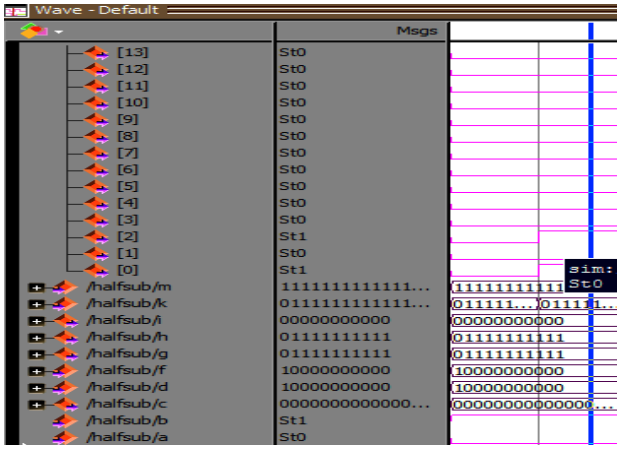

Fig 9 Waveform of floating point subtraction with double precision.

Fig 8 and Fig 9 pageants the simulated output of Floating point subtraction with double precision a(sign), b(sign), e(exponent), f(exponent), m(mantissa), k(mantissa) are the inputs and s(difference) and c(borrow) are the outputs.

FLOATING POINT ALU WITH DOUBLE PRECISION BLOCK

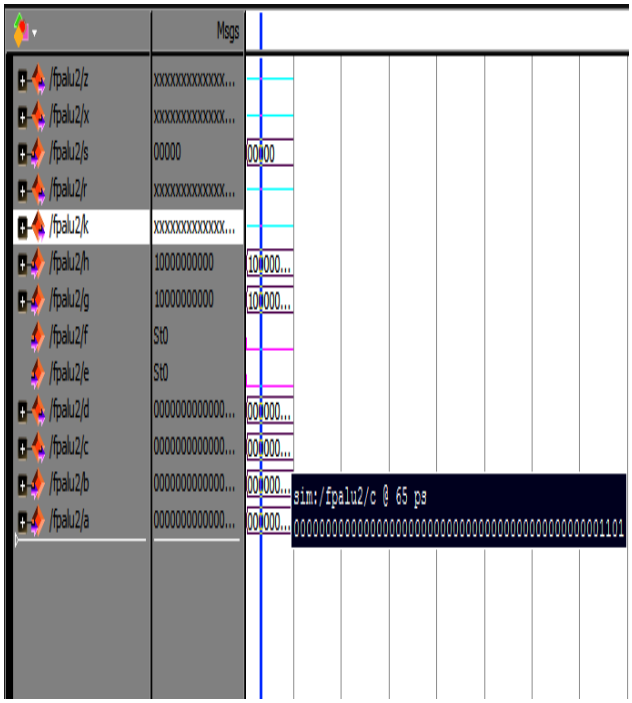

Fig 10 Waveform of floating point ALU with double precision.

Fig 10 shows Fig 9 the simulated output of Floating point ALU with double precision a(sign), b(sign), h(exponent), g(exponent), m(mantissa), k(mantissa) are the inputs and d(sum) , c(carry), x(difference),z(borrow)and r(output for other operations) are the outputs.

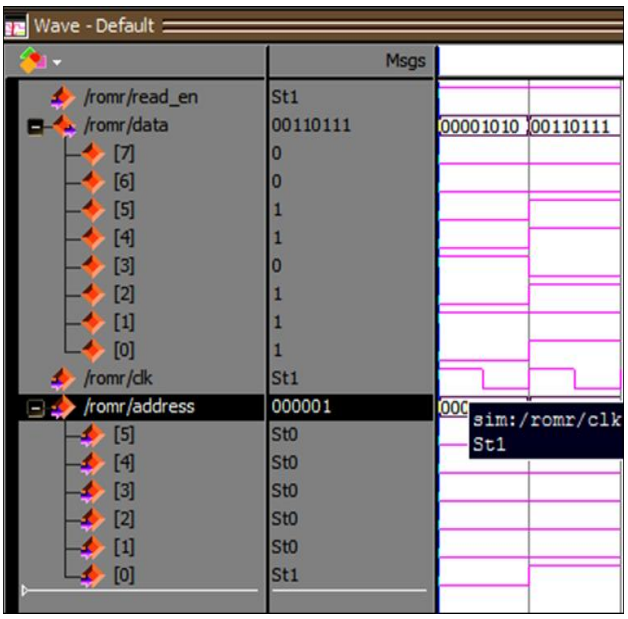

Fig 11 Waveform instruction and data memory block (ROM)

Fig 11 pageants the simulated output of ROM where read_en,clk and address are the input. The output is data, the data are already stored in the address to check this read_en is forced to be 1 and address is given as input.The output is shows the value which is already stored it the specified location.

PROGRAM COUNTER

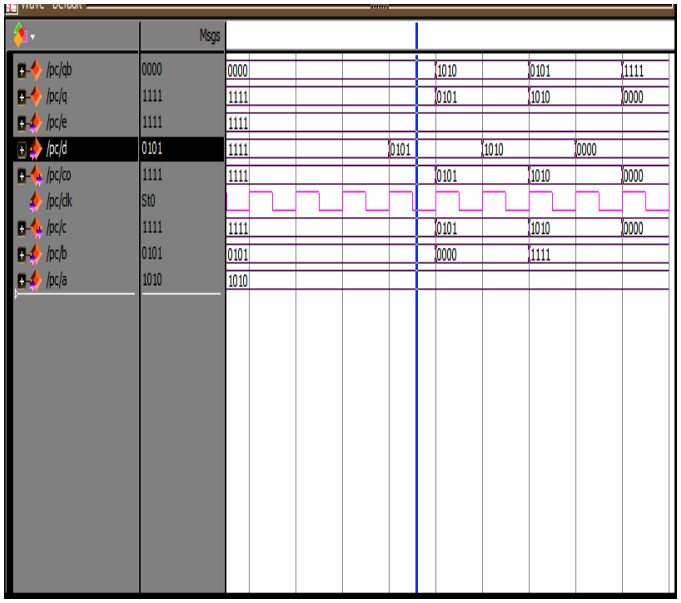

Fig 12 Waveform of program counter

Fig 12 shows simulated output of program counter

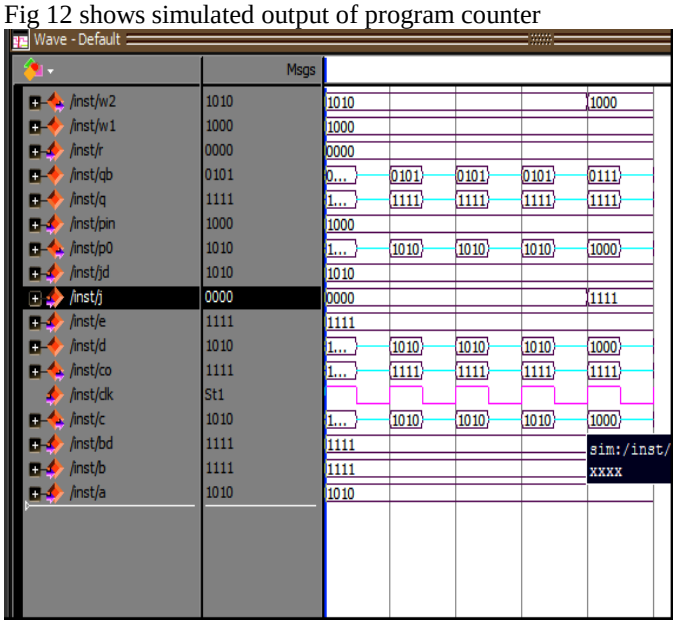

Fig 13 Waveform of instruction fetch block

Fig 13 pageants the simulated output of instruction fetch block

Bd,b(select line), pin, j(select line),jd,clk,e,t, and r(input for adder) are inputs q(output for flip-flop),qb(output for flip-flop),c(output from program counter),c0(output from program counter),p0(final input from instruction fetch) are output pins. The following parameters were measured using cadence tool which were shown below

FLOATING POINT ALU

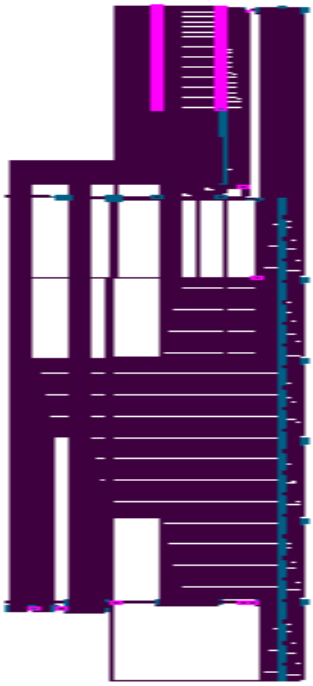

Fig 14 RTL View for Floating point ALU

```
  Instance     Cells  Cell Area  Net Area    Wireload
-----------------------------------------------------
palu2           2971      18686         0      <none> (D)
  add_124_5      757       3967         0      <none> (D)
  add_140_4      757       3967         0      <none> (D)
  add_125_8      332       1580         0      <none> (D)
  mux_c_12_7      52       1562         0      <none> (D)
  add_123_5      325       1544         0      <none> (D)
  g2              53        159         0      <none> (D)
  g1              52        156         0      <none> (D)
  mul_118_12       1          6         0      <none> (D)
  mul_116_12       1          6         0      <none> (D)
  add_117_20       1          6         0      <none> (D)
  add_117_13       1          6         0      <none> (D)
  mul_114_12       1          6         0      <none> (D)
  add_115_20       1          6         0      <none> (D)
  add_115_13       1          6         0      <none> (D)
  mul_112_12       1          6         0      <none> (D)
  add_113_20       1          6         0      <none> (D)
  add_113_13       1          6         0      <none> (D)
  mul_110_12       1          6         0      <none> (D)
  add_111_20       1          6         0      <none> (D)
  add_111_13       1          6         0      <none> (D)
  mul_108_12       1          6         0      <none> (D)
  add_109_20       1          6         0      <none> (D)
  add_109_13       1          6         0      <none> (D)
```

Fig 15 Area Report for Floating point ALU

```
                    Leakage    Dynamic     Total
  Instance   Cells Power(nW)  Power(nW)  Power(nW)
-------------------------------------------------
palu2         2971 53123.297 358662.683 411785.980
  add_140_4    757  4025.467  51604.368  55629.835
  add_124_5    757  4025.467  57179.279  61204.745
  add_125_8    332  1917.857  29832.706  31750.563
  add_123_5    325  1877.957  11376.133  13254.090
  mux_c_12_7    52  1093.332  13933.348  15026.679
  g2            53   248.258   7613.287   7861.545
  g1            52   243.574   3124.516   3368.090
  add_15_12      1    20.924    129.247    150.171
  add_17_12      1    20.924    140.676    161.600
  add_17_18      1    20.924    193.429    214.353
  add_19_12      1    20.924    140.676    161.600
  add_19_18      1    20.924    217.169    238.093
  add_21_13      1    20.924    129.247    150.171
  add_21_19      1    20.924    164.416    185.340
  add_23_11      1    20.924    131.883    152.808
  add_23_17      1    20.924    202.221    223.146
  add_25_12      1    20.924    140.676    161.600
  add_25_18      1    20.924    211.014    231.938
  add_27_12      1    20.924    140.676    161.600
```

Fig 16 Power Report for Floating point ALU with double precision

```
  Pin                Type         Fanout  Load Slew Delay Arrival
                                          (fF) (ps)  (ps)   (ps)
-------------------------------------------------------------------
b[0]          (u)  in port              8  40.0    0    +0      0 F
g1/A[0]
  g52/in_0                                            +0      0
  g52/z       (u)  unmapped_not         3  15.0    0   +44     44 R
g1/Z[0]
add_123_5/A[0]
  g2/in_0                                             +0     44
  g2/z        (u)  unmapped_nand2       1   5.0    0   +38     82 F
  g827/in_0                                           +0     82
  g827/z      (u)  unmapped_not         2  10.0    0   +37    119 R
  g109/in_0                                           +0    119
  g109/z      (u)  unmapped_nand2       1   5.0    0   +38    157 F
  g830/in_0                                           +0    157
  g830/z      (u)  unmapped_not         3  15.0    0   +44    201 R
  g214/in_1                                           +0    201
  g214/z      (u)  unmapped_nand2       1   5.0    0   +38    239 F
  g832/in_0                                           +0    239
```

Fig 17 Delay report for floating point ALU with double precision

Table I Comparison area of existent work and proffered work for individual blocks in RISC architecture

Blocks	Existent work's area(in terms of cell area)	Proffered work 's area (in terms of cell area)
Memory	**6102**	**4051**
ALU	**8204**	**7193**
ALU-Memory interface	**9720**	**7193**
Addition with double precision	**2048**	**1984**
Subtraction with double precision	**3099**	**2057**
ROM	**4237**	**3871**
Program counter	**98**	**48**
Instruction fetch	**114**	**72**

Table I pageants the area observed for blocks during the implementation of the RISC processor. Comparing to the existent work the area was reduced, for the betterment of the RISC processor.

Table II Comparison power of existent work and proffered work for individual blocks in RISC architecture

Blocks	Existing work's power(in terms nw)	Proposed work 's power (in terms nw)
Memory	3258.702	2228.200
ALU	184676.20	115661.82
ALU-Memory interface	215701.200	115601.182
Addition with double precision	41237.5	32197.665
Subtraction with double precision	4057.972	34079.434
ROM	16500.500	15931.785
Program counter	2078.595	1888.202
Instruction fetch	2134.315	3288.294

Table II pageants the power observed for blocks during the implementation of the RISC processor. Comparing to the existent work the area was reduced, for the betterment of the RISC processor.

Table III Comparison delay of existent work and proffered work for individual blocks in RISC architecture

Blocks	Existing work's delay(in terms ns)	Proposed work 's delay(in terms ns)
Memory	547	483
ALU	456	322
ALU-Memory interface	657	509
Addition with double precision	395	248
Subtraction with double precision	539	431
ROM	748	611
Program counter	100	70
Instruction fetch	200	108

Table III pageants the delay observed for blocks during the implementation of the RISC processor. Comparing to the existent work the area was reduced, for the betterment of the RISC processor.

Table IV Comparison of existent work and proffered work for floating-point ALU with double precision

Parameters	Existent work	Proffered work
Area (cell size)	20500	18686
Total power(in nw)	518311.720	411785.980
Delay (ps)	315	206

Table IV pageants the following parameters observed during the implementation of the floating point ALU with double precision.

VI. CONCLUSION

In this paper an area and power potent floating point ALU with double precision architecture for implementing RISC processor is presented. The proposed architecture is designed and several circumstances are taken into scrutiny like area, power and latency. In order to evaluate resource efficiency and power consumption, the proposed architecture is enforced on a Cadence. The obtained results are effectively reducing the power consumption, area and latency.

REFERENCES

1. Ritpurkar, S. P., Thakare, M. N., & Korde, G. D. (2015, January). Design and simulation of 32-Bit RISC architecture based on MIPS using VHDL. In 2015 International Conference on Advanced Computing and Communication Systems (pp. 1-6). IEEE
2. Mane, P. S., Gupta, I., & Vasantha, M. K. (2006, December). Implementation of RISC Processor on FPGA. In 2006 IEEE International Conference on Industrial Technology (pp. 2096-2100). IEEE.
3. Paldurai, K., & Hariharan, K. (2015, January). FPGA implementation of delay optimized single precision floating point multiplier. In 2015 International Conference on Advanced Computing and Communication Systems (pp. 1-5). IEEE.
4. Kaur, S., & Jassal, P. S. An Efficient Field Programmable Gate Array Implementation of Double Precision Floating Point Multiplier using VHDL.
5. Kumar, J. V., Swapna, C., Nagaraju, B., & Ramanjappa, T. (2014). FPGA Based Implementation of Pipelined 32-bit RISC Processor with Floating Point Unit. In IEEE International Conference on Communication and Signal Processing.
6. Gollamudi, P. S., & Kamaraju, M. (2013). Design Of High Performance IEEE-754 Single Precision (32 bit) Floating Point Adder Using VHDL. International Journal of Engineering Research & Technology, 2(7), 2264-2275.
7. Tomar, A. K. S., & Jain, R. (2013). 20-Bit RISC and DSP System Design in an FPGA. Computing in Science & Engineering, 16(2), 16-2

AUTHORS PROFILE

D Divya, is an undergraduate student in department of Electronics and communication engineering at National Engineering College, Tamil Nadu, India. Her research interest includes VLSI and networking.

R Balasaraswathi, is an undergraduate student in department of Electronics and communication engineering at National Engineering College, Tamil Nadu, India. Her research interest includes VLSI and networking.

M Harini Kalyani, is an undergraduate student in department of Electronics and communication engineering at National Engineering College, Tamil Nadu, India. Her research interest includes VLSI and networking.

I.Vivek Anand, received B.E degree in Electronics and Communication Engineering from Sethu Institute of Technology (2012) and M.E degree in VLSI from Mepco Schlenk Engineering College (2014). He is currently working in National Engineering College, Kovilpatti, India as Assistant Professor. His research interest includes Modeling and Simulation of Tunnel Field effect Transistors.