Journal of Harbin Engineering University
ISSN: 1006-7043

Vol 45 No. 11
November 2024

Comparative Analysis of 16-bit and 32-bit RISC Pipelined Processors by Verilog Design

[1]Tejaswini D, [2]Mrs. Shrisha M R,

Department of Electronics and Communication Engineering
[1]Student, BMS College of Engineering, Bangalore
[2]Assistant Professor, BMS College of Engineering, Bangalore

Abstract

The evolution and advancement of microprocessors has driven the quest for enhanced performance and efficiency, particularly within the domain of Very-Large- Scale Integration (VLSI) design. The Reduced Instruction Set Computer (RISC) architecture is designed to execute small, uniform instructions efficiently, aiming to minimize the complexity, cost, cycle time, and energy usage of microprocessors. Although 16-bit RISC processors emerged in the 1970s, they encountered significant technical challenges, leading to the evolution of more advanced RISC processors with the use of pipelining architectures. This paper presents a comparative study of 16-bit and 32-bit RISC processors, each integrated with Floating Point Units (FPUs) and distinct instruction sets. The RISC processor with 16 bits utilizes a non- pipelined Harvard architecture employing independent data and instruction storage memories, while the 32-bit RISC microprocessor features a pipelined design inspired by the MIPS architecture. Both processors include General Purpose Registers (GPRs) and Flag registers (e.g., Carry, Zero). The study involves simulating and optimized floating-point units (FPU) and focusing on optimizing the data path, as arithmetic and logical operations significantly impact power consumption and execution delay. Therefore, comparing the two models, theirperformance and instruction sets were assessed, including speedup and power dissipation. Each model was designed in Verilog, simulated, and then integrated into a top-level component with the ISE Design Suite XILINX 14.4, with a detailed power analysis provided and also synthesized in Cadence genus and obtained timing analysis.

Keywords— VLSI Design, RISC Architecture, Floating PointUnits (FPUs), Verilog, XILINX ISE Design Suite 14.4, Genus.

1. Introduction

In the evolving landscape of digital design, the Reduced Instruction Set Computing (RISC) architecture stands out for its streamlined instruction sets and emphasis on performance efficiency. RISC processors are fundamental to a variety of uses, including the applications, from embedded systems to high-performance computing. This review paper provides a comprehensive comparative study of 16-bit and 32-bit RISC pipelined processors, focusing on their design and implementation using Hardware Description Language (HDL) in Verilog. The decision between utilizing 16-bit and 32-bit architectures is crucial and involves understanding the inherent trade-offs. A 16-bit processor typically offers advantages in terms of reduced complexity, lower power consumption, and cost-effectiveness, making it ideal for resource-constrained environments and applications requiring moderate computational power. Conversely, a 32-bit processor delivers superior computational capabilities, larger addressable memory space, and enhanced performance, essential for advanced embedded systems, real-time processing, and scientific computations.

A novel aspect of this is the unification of floating-point operators within the comparative framework. Floating-point operations are increasingly vital in applications requiring a high degree of precision and dynamic range, like digital signal processing, graphics, and scientific simulations. By incorporating floating-point units (FPUs) in both 16-bit and 32-bit RISC processors, we assess their impact on performance, resource utilization, and power efficiency. This paper delves into the architectural intricacies of 16-bit and 32-bit RISC pipelined processors, with a particular emphasis on their floating-point processing capabilities. We explore key design elements, including pipeline stages, instruction sets, data paths, and control mechanisms, to investigate the role of these factors influence overall processor performance and resource utilization. The application of Verilog HDL as a design and simulation tool enables precise modeling and verification, providing valuable perceptions of the operational characteristics of these processors.

Our analysis is grounded in detailed simulations and synthesis results obtained using Cadence Genus, highlighting the strengths and weaknesses of each processor type. By examining metrics such as speed, power consumption, scalability, and the efficiency of floating- point operations, we aim to offer a balanced perspective that can inform the selection of appropriate processor configurations for specific applications.

The paper is organized in the subsequent way: We begin with an outline of RISC architecture, pipelining concepts, and the importance of floating-point operators. Following this, a detailed comparison of 16-bit and 32-bit processors according to their design and performance indicators, such as floating- point capabilities. Subsequent sections discuss the implications of our research for real-world uses and future trends in processor design. Through this comparative analysis, we seek to bolster the ongoing discourse on processor design optimization, providing a valuable reference for researchers and engineers in the field of digital design.

16-Bit RISC PROCESSOR

The design of the 16-bit RISC processor presented here is according to the Harvard architectural design and features 8 general-purpose registers. It includes a basic Arithmetic Logic Unit (ALU) that performs fundamental operations like shift operations and addition, along with data memories and an instruction set consisting of fourteen instructions. The processor supports a load-store architecture, meaning every action is carried out inside the registers. Being a non-pipelined processor, it offers simplicity in understanding and implementation.

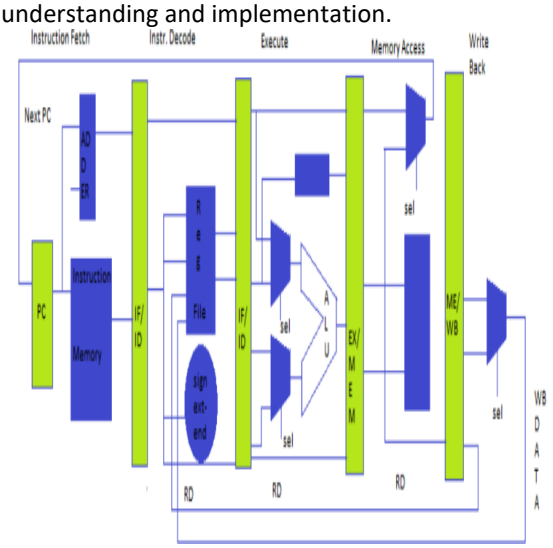


Fig 1. 16-Bit Non-Pipelined RISC Processor Architecture

Fig 1. provides a detailed depiction of the 16-Bit RISC processor. The building design is primarily composed of five logical blocks: the memory unit, program counter (PC), execution unit, control unit, and arithmetic logic unit (ALU). All operands, retrieved from the Memory Unit, are stored in the register file. The ALU performs activities, including read and write operations as needed, upon receiving the proper signal from the Control Unit. After that, the Program Counter moves on to the following instruction.

A. PC- Program Counter

The 16-bit latch is responsible for retrieving the location of the upcoming directive from memory. As each once a command is carried out, the program counter increments to prepare the next step in the sequence for execution. However, the new instruction cannot be loaded until the current one is fully processed. The latch is linked to the unit of control because many instructions that modify the counter for the program are part of the set of instructions. Although this connection is crucial, it introduces additional hardware complexity and increases power dissipation. Therefore, the latch is ideally suited for a load/store architecture.

B. ALU- Arithmetic and Logic Unit

The ALU executes the majority of instructions within the set of instructions, controlled by the ALU Control unit's ALU_Op signal. It does mathematical operations, including addition and subtraction, logical operations like AND and NOT (invert), and rotate operations, which are particularly useful in signal processing tasks such as correlation. The quantity of registers or data elements utilized based on the kind of operation. For instance, a basic addition operation utilizes three registers (two source and one destination), whereas an add immediate operation requires only two registers (one source and one destination) along with a binary data element directly available in the opcode. The outcomes among these procedures affect the ALU's output as well as the flags indicate the outcomes of the carried-out procedures. Table I offers a comprehensive summary of the instructions followed by the ALU.

C. CU- Control Unit

The control modules serve as the processor's central control, interpreting input derived from opcodes kept in stored in memory. Dependent on these opcodes, it enables or disables signals to determine which operation should be performed. As Fig. 2, illustrates the instruction format can be categorized into:

Memory Access, Data Processing, Branch, Jump

Table 1: ALU Control Design

ALU Control				
ALU Op	Opcode (Hex)	ALUcnt	ALU Operation	Instruction
10	Xxxx	000	LW/SW	Load/Store
00	0002	000	ADD	Addition
00	0003	001	SUB	Subtract
00	0004	010	INVERT	Invert
00	0005	011	LSL	Logical Shift Left
00	0006	100	LSR	Logical Shift Right
00	0007	101	AND	Logical And
00	0008	110	OR	Logical Or
00	0009	111	SLT	Set Less Than

Table 2: Format of guidelines in RISC Processor

Memory Access: Load

Opcode	*Source Register Rs*	*Destination Register Rd*	*Offset*
4	3	3	6

Memory Access: Store

Opcode	*Source Register Rs1*	*Source Register Rs2*	*Offset*
4	3	3	6

Data Processing

Opcode	*Source Register Rs1*	*Source Register Rs2*	*Destination Register Rd*	*Offset*
4	3	3	3	3

Branch:

Opcode	*Source Register Rs*	*Source Register Rs2*	*Offset*
4	3	3	6

Jump:

Opcode	*Offset*
4	12

32-Bit RISC PROCESSOR

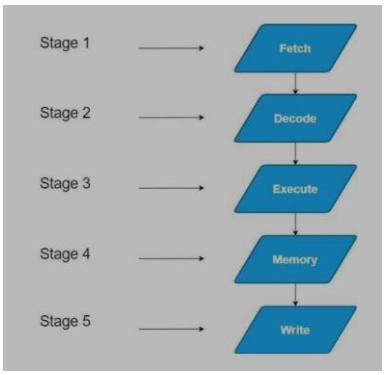
The 32-bit RISC-based processor, employing a 3-stage pipeline and Harvard architecture, represents a successful implementation of the MIPS processor. It offers several advantages over a 16-bit machine, such as addressing twice the quantity of unique memory locations and handling larger data chunks. Consequently, in comparable conditions, a 32- bit CPU processor typically achieves nearly double the speed of a 16-bit processor.

The increase in data width from 16 to 32 bits has enabled new application areas, like images and handling extensive data structures, while also enhancing the capability to work in fields like convolution. Intensive operations, like computing the nth power of a number, require substantial effort and time, rendering a 16-bit processor unsuitable for these tasks.

Fig. 3 Sequence of Operations

Fig 3. demonstrates the execution process of an operation within the suggested framework. This process includes retrieving the directive from memory, carrying out the required computations, writing the results back to memory, and obtaining the final outcome of the operation.

A. Pipelined



Pipelining, as its name implies, facilitates the sequential saving and running the instructions. Traditional CPUs encountered substantial delays because no other operations could occur while one was being processed, akin to a system software being idle when an I/O apparatus is active, which leads to considerable time delays. Pipelining mitigates this problem by allowing parallel execution; for instance, as opposed to opcode 1 is being fetched and after which opcode 2 can be simultaneously be obtained and interpreted concurrently, enabling multiple directives to be processed at once.

The benefits of pipelining for the processor include:

- Less Cycles Per Instruction (CPI): Pipelining lowers the CPI and theoretically enhances speed by the factor proportional to the amount of pipeline stages.
- Increased Throughput: By processing multiple instructions concurrently, pipelining shortens the interval between finished instructions, which reduces the average duration per instruction.
- Support for Complex Instructions: The improved speed of pipelined processors supports advanced instructions within the ALU, expanding the

processor's application scope, such as in DSP (Digital Signal Processing) tasks like convolution.

- Higher Operating Frequency: Pipelining allows CPUs to perform at higher frequencies than that of the RAM, significantly enhancing overall computer performance. This improvement is due to the simplification of the combinational circuits, which reduces the overall delay.

Fig 4. 32-Bit Non-Pipelined RISC Processor Architecture

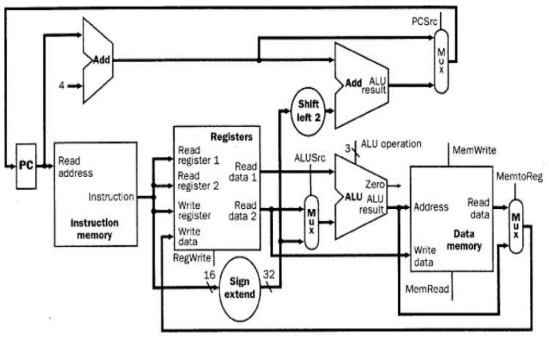


Fig 4. illustrates the comprehensive structure of the implemented 32-bit MIPS processor, which features a 3-stage pipelining process. The stages are as follows:

Instruction Fetch: Registers are used to store and retrieve instructions from memory. The address for the instruction to be executed is provided by the program counter (PC).

Instruction Decode: The operation code along with the operands is retrieved and matched against the set of instructions stored within the registry file.

Execute: The computation occurs in this stage, with operands provided to the ALU, and ALU_out is the computed result that is output.

Memory Access: The ALU output is recorded within Data Memory at the specified location.

Write Back: The last operational code specifies the target position at which is written back with the calculated data to the register. The last opcode specifies the target position at which is written back with the calculated data to the register.

The objective behind reaching maximum processing capability speed, versus a comparable non-pipelined processor, is realized by fully leveraging hardware resources. This approach enhances performance even when executing more complex instructions.

B. Flag Registers

Registers for flags are found in 16-bit RISC processors, but their functionality and use are limited by less complicated calculations and a more constrained instruction set. During instruction execution, modifications done to the flag register are applicable to evaluate the ALU output [13]. Of the 32 bits available, 5 bits are allocated for flag purposes, as illustrated in Fig. 5.

1. Flag of Parity (PF): The first bit within the flag registry is designated as PF. It determines how many 1s are in the outcome. PF is set to 1 if the total of 1s is even; otherwise, it stays at 0.

2. The Zero Flag (ZF): The second bit within flag registry is ZF. It indicates if the outcome of an ALU operation or instructions like BEQZ or BNE is zero. Assume that the outcome is 0, ZF is set; otherwise, it continues to be reset.

3. Sign Flag (SF): The third bit within flag registry is SF. It determines whether the result, considered as a signed number, it is encompassed by the range of signed data. The most significant bit (MSB) is applied for this check. If the MSB is 1, the outcome is negative; if 0, the outcome is positive.

4. Auxiliary Carry Flag (ACF): The fourth bit within flag registry is ACF. It records the carry produced during arithmetic operations (addition, subtraction) that occurs except for the MSB. This flag captures the additional carry over or borrow bit and updates itself during bit-by-bit computation.

5. Carry Flag (CF): The fifth bit within flag register is CF. It holds the carry created by the MSB or the borrow for the MSB. If a carry or borrow occurs, CF is set; otherwise, it remains cleared.

4	3	2	1	0
CF	ACF	SF	ZF	PF

Fig 5. Flag Register

C. Modified Instruction Set

With the presence of additional all-purpose registries and an extended opcode length, 6 bits are employed to specify the type of instruction, allowing for up to 63 possible instructions. The suggested 32-bit MIPS architecture features advanced instructions, including multiplication and comparison operations, such as confirming whether aregister's contents are zero.

As shown in Fig.6, the processor features three primaryinstruction varieties encoding:

1. R-Type Instruction Encoding: In this format, the ALU executes operations solely on the both the origin and the destination registers. The set of 5 bits from positions 10 to 6 indicate the shift quantity (shamt) for rotation and shifting operations. The final 6 bits, from positions 5 to 0, define the opcode extension (funct), specifying extra operations to be carried out.
2. I-Type Instruction Encoding: This format includes ALU operations that use data from a source register combined with immediate data defined by the opcode. The destination register contains the result. The last set of 15 bits are allocated for the current data used in the operation, as outlined by the instruction.

R-Type:

Opcode	*Source Register Rs1*	*Source Register Rs2*	*Destination Register Rd*	*Shamt*	*Funct*
6	5	5	5	5	5

I-Type:

Opcode	*Source Register Rs*	*Destination Register Rd*	*Immediate Data*
6	5	5	16

J-Type:

Opcode	*Immediate Data*
6	26

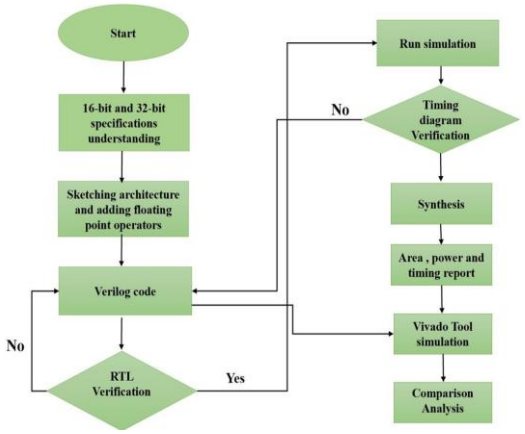
Fig 6. Instruction Template

3. J-Type Instruction Encoding: Jump instructions, may alter the program sequence's execution path by pointing it to a specific memory address, employ this encoding. The directive is 32 bits long in this format. The operation's opcode is contained in the first six bits, while the program's current data address is found in the next twenty-six bits.

2. Objectives

This project focuses on the implementation of 16-bit and 32-bit RISC processor architectures, with the aim of enhancing computational speed through the addition of floating-point operators and pipeline stages. The architectures are optimized by incorporating floating-point operations, and their performance is evaluated using the Cadence synthesis tool. Additionally, a comparative analysis of the architectures is conducted using Vivado Xilinx for ASIC synthesis, assessing factors such as speed and efficiency. The goal is to improve overall performance by reducing execution delays and enhancing computational accuracy.

3. Methods



The methodology for designing and analyzing 16-bit and 32-bit RISC processors with floating-point operations follows a structured approach. It begins with understanding the specifications for both architectures, which includes defining the instruction sets, data widths, and computational requirements. After this, the processor architecture is sketched, incorporating floating-point units to enhance computational abilities. The next step involves coding the design using Verilog HDL, followed by RTL (Register Transfer Level) verification to ensure the design functions correctly. If issues arise during verification, the design is refined and retested. Once verified, the processor design is simulated to evaluate its behavior, and timing diagrams are checked to ensure correct operation. After that, the design is synthesized, transforming it into a gate-level model suitable for hardware implementation. Metrics such as area, power consumption, and timing are analyzed to evaluate the design's efficiency. Following this, the design undergoes further simulation using Vivado tools to ensure accuracy. Finally, a comparative analysis is conducted to assess the performance of the 16-bit and 32-bit processors, focusing on factors like execution speed, power consumption, and area efficiency, especially with the inclusion of floating-point operations. This iterative process ensures a well-optimized, high-performance processor design.

4. IMPLEMENTED RESULTS

To effectively compare the functionality of both processors, we simulate a related program on each and evaluate them based on various criteria. To achieve this, we implemented a basic addition program on both the 16-bit and 32-bit RISC processors, each featuring Floating Point Units

(FPUs). The Xilinx ISE Design Suite 14.4 was utilized for the implementation process and also synthesized in Cadence by Genus tool.

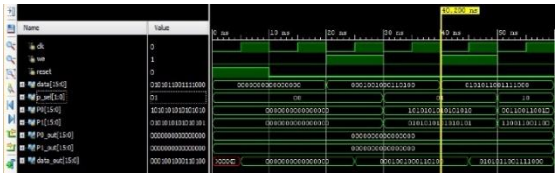
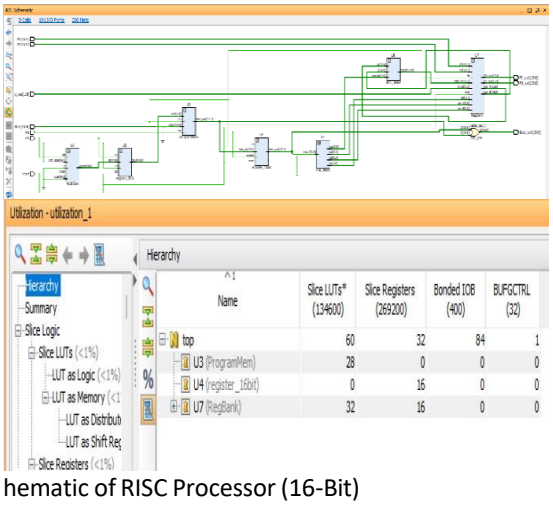


Fig 7. 16-Bit RISC Simulation Results

```
@genus:root: 5> report_power
Info    : Joules engine is used. [RPT-16]
        : Joules engine is being used for the command report_power.
Info    : ACTP-0001 [ACTPInfo] Activity propagation started for stim#0 netlist
        : top
Info    : ACTP-0009 [ACTPInfo] Activity Propagation Progress Report : 100%
Info    : ACTP-0001 Activity propagation ended for stim#0
Info    : PWRA-0001 [PwrInfo] compute_power effective options
        : -mode : vectorless
        : -skip_propagation : 1
        : -frequency_scaling_factor : 1.0
        : -use_clock_freq : stim
        : -stim :/stim#0
        : -fromGenus : 1
Info    : ACTP-0001 Timing initialization started
Info    : ACTP-0001 Timing initialization ended
Info    : PWRA-0002 [PwrInfo] Skipping activity propagation due to -skip_ap
        : option....
Warning: PWRA-0302 [PwrWarn] Frequency scaling is not applicable for vectorless
        : flow. Ignoring frequency scaling.
Warning: PWRA-0304 [PwrWarn] -stim option is not applicable with vectorless mode
        : of power analysis, ignored this option.
Info    : PWRA-0002 Started 'vectorless' power computation.
Info    : PWRA-0009 [PwrInfo] Power Computation Progress Report : 100%
Info    : PWRA-0002 Finished power computation.
Info    : PWRA-0007 [PwrInfo] Completed successfully.
        : Info=6, Warn=2, Error=0, Fatal=0
Instance: /top
Power Unit: W
PDB Frames: /stim#0/frame#0
  ------------------------------------------------------------------------
    Category       Leakage     Internal    Switching       Total    Row%
  ------------------------------------------------------------------------
      memory    0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
    register    2.59371e-08  1.43391e-05  3.39228e-07  1.47042e-05  17.06%
       latch    2.41030e-09  3.55051e-06  7.06180e-07  4.25910e-06   4.94%
       logic    6.96955e-08  4.41409e-05  2.30124e-05  6.72230e-05  78.00%
        bbox    0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
       clock    0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
         pad    0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
          pm    0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
  ------------------------------------------------------------------------
    Subtotal    9.80429e-08  6.20305e-05  2.40578e-05  8.61864e-05 100.00%
  Percentage         0.11%       71.97%       27.91%      100.00% 100.00%
  ------------------------------------------------------------------------
```

Fig 8. RTL Sc



hematic of RISC Processor (16-Bit)

Fig 9. Utilization of 16-Bit RISC Processor

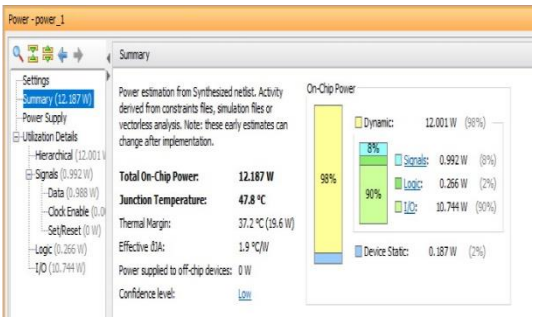
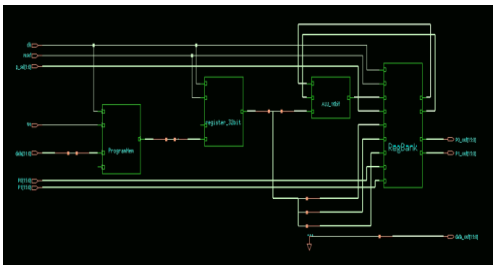
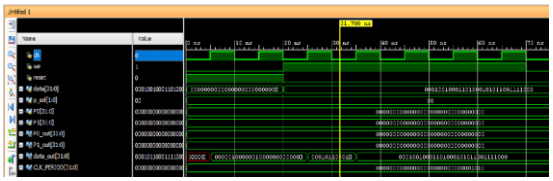


Fig 10. Power Analysis Results for 16-Bit RISC Processor OperationSYNTHESIS RESULTS FROM CADENCE GENUS




```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 20.10-p001_1
  Generated on:           Jul 24 2024  06:36:40 pm
  Module:                 top
  Operating conditions:   PVT_0P9V_125C (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================


Path 1: MET (1761 ps) Setup Check with Pin U7/R1/reg_data_reg[7]/CK->SI
          Group: clk
     Startpoint: (R) U7/R6/reg_data_reg[2]/CK
          Clock: (R) clk
       Endpoint: (F) U7/R1/reg_data_reg[7]/SI
          Clock: (R) clk

                     Capture       Launch
        Clock Edge:+   10000            0
         Src Latency:+     0            0
         Net Latency:+     0 (I)        0 (I)
             Arrival:=   10000            0

               Setup:-    145
       Required Time:=   9855
        Launch Clock:-      0
           Data Path:-   8094
               Slack:=   1761
```

```
@genus:root: 4> report_area
============================================================
  Generated by:           Genus(TM) Synthesis Solution 20.10-p001_1
  Generated on:           Jul 25 2024  05:22:54 pm
  Module:                 top
  Operating conditions:   PVT_0P9V_125C (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

      Instance          Module      Cell Count  Cell Area  Net Area   Total Area   Wireload
-------------------------------------------------------------------------------------------
top                                       1923   4705.578     0.000     4705.578 <none> (D)
  U3                ProgramMem             32    240.768     0.000      240.768 <none> (D)
  U6                ALU_16bit            1418   2860.146     0.000     2860.146 <none> (D)
    U0              ieee_16add            531   1016.424     0.000     1016.424 <none> (D)
    U3              ieee_16mul            743   1558.152     0.000     1558.152 <none> (D)
  U7                RegBank               457   1582.776     0.000     1582.776 <none> (D)
    R0              Reg_16_bit             17    132.012     0.000      132.012 <none> (D)
    R1              Reg_16_bit_260         17    132.012     0.000      132.012 <none> (D)
    R2              Reg_16_bit_259         17    132.012     0.000      132.012 <none> (D)
    R3              Reg_16_bit_258         17    132.012     0.000      132.012 <none> (D)
    R4              Reg_16_bit_257         17    132.012     0.000      132.012 <none> (D)
    R5              Reg_16_bit_256         17    132.012     0.000      132.012 <none> (D)
    R6              Reg_16_bit_255         17    132.012     0.000      132.012 <none> (D)
    R7              Reg_16_bit_254         17    132.012     0.000      132.012 <none> (D)
    U0              mux_8to1              137    215.460     0.000      215.460 <none> (D)
    U1              mux_8to1_104          138    217.170     0.000      217.170 <none> (D)
  decoder_3to8_inst decoder_3to8           14     17.442     0.000       17.442 <none> (D)
(D) = wireload is default in technology library
```



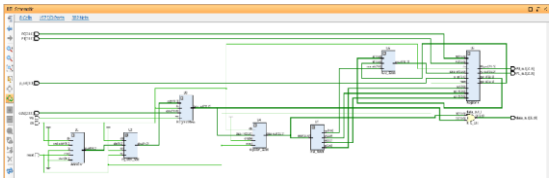


Fig 13. RTL Schematic of RISC Processor (32-Bit)

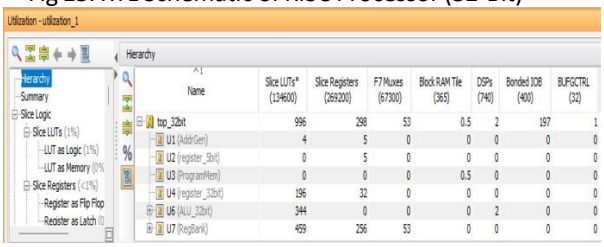


Fig 14. Utilization of 32-Bit RISC Processor

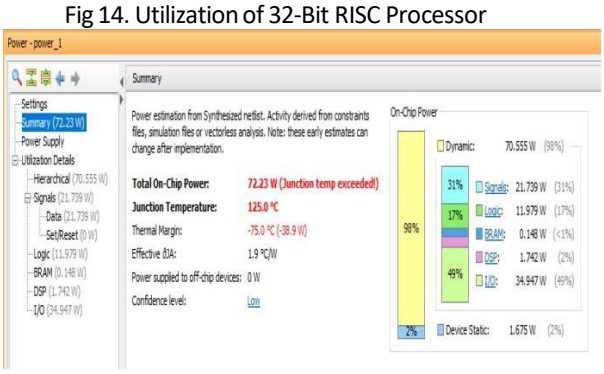
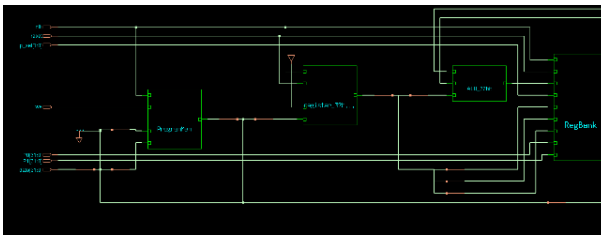


Fig 15. Power Analysis of 32-Bit RISC Processor

SYNTHESIS RESULTS



Objectives

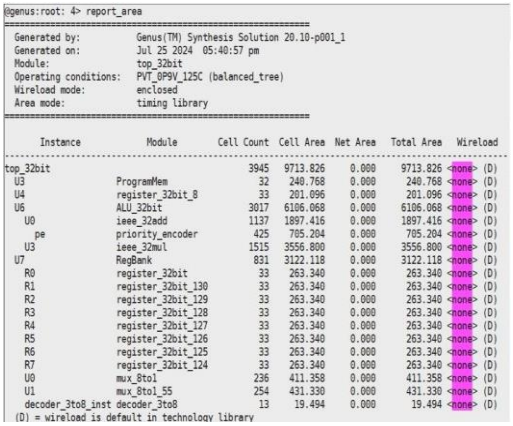
This project focuses on the implementation of 16-bit and 32-bit RISC processor architectures, with the aim of enhancing computational speed through the addition of floating-point operators and pipeline stages. The architectures are optimized by incorporating floating-point operations, and their performance is evaluated using the Cadence synthesis tool. Additionally, a comparative analysis of the architectures is conducted using Vivado Xilinx for ASIC synthesis, assessing factors such as speed and efficiency. The goal is to improve overall performance by reducing execution delays and enhancing computational accuracy.

```
@genus:root: 5> report_power
Info    : Joules engine is used. [RPT-16]
        : Joules engine is being used for the command report_power.
Info    : ACTP-0001 [ACTPInfo] Activity propagation started for stim#0 netlist
        : top_32bit
Info    : ACTP-0009 [ACTPInfo] Activity Propagation Progress Report : 100%
Info    : ACTP-0001 Activity propagation ended for stim#0
Info    : PWRA-0001 [PwrInfo] compute_power effective options
        : -mode : vectorless
        : -skip_propagation : 1
        : -frequency_scaling_factor : 1.0
        : -use_clock_freq : stim
        : -stim :/stim#0
        : -fromGenus : 1
Info    : ACTP-0001 Timing initialization started
Info    : ACTP-0001 Timing initialization ended
Info    : PWRA-0002 [PwrInfo] Skipping activity propagation due to -skip_ap
        : option....
Warning: PWRA-0302 [PwrWarn] Frequency scaling is not applicable for vectorless
        : flow. Ignoring frequency scaling.
Warning: PWRA-0304 [PwrWarn] -stim option is not applicable with vectorless mode
        : of power analysis, ignored this option.
Info    : PWRA-0002 Started 'vectorless' power computation.
Info    : PWRA-0009 [PwrInfo] Power Computation Progress Report : 100%
Info    : PWRA-0002 Finished power computation.
Info    : PWRA-0007 [PwrInfo] Completed successfully.
        : Info=6, Warn=2, Error=0, Fatal=0
Instance: /top_32bit
Power Unit: W
PDB Frames: /stim#0/frame#0
  ------------------------------------------------------------------------
   Category       Leakage     Internal    Switching       Total    Row%
  ------------------------------------------------------------------------
     memory   0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
   register   5.28894e-08  1.19301e-04  3.09883e-06  1.22453e-04  35.54%
      latch   0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
      logic   1.56572e-07  1.46741e-04  7.00190e-05  2.16916e-04  62.96%
       bbox   0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
      clock   0.00000e+00  0.00000e+00  5.18400e-06  5.18400e-06   1.50%
        pad   0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
         pm   0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
  ------------------------------------------------------------------------
   Subtotal   2.09461e-07  2.66042e-04  7.83018e-05  3.44553e-04 100.00%
 Percentage         0.06%       77.21%       22.73%      100.00% 100.00%
  ------------------------------------------------------------------------
```

5. CONCLUSION AND FUTURE SCOPE

This analysis provides an in-depth survey of various RISC Processor architectures, internal data pathways and fewer transistors, 16-bit computers are costlier but are limited to use in a growing number of niche applications. Notwithstanding their increased cost and power consumption, 32-bit processors offer noticeably improved performance, economy, and operational speed particularly in demanding applications like floating-point arithmetic. 32-bit processors use more dynamic power due to its elevated operating frequency, which permits for more operations to be carried out in a cycle, resulting in quicker processing rates and less combinational delays. 32-bit CPUs' pipelined architecture optimizes the instruction cycle, which further improves speed.

```
Path 3: MET (660 ps) Setup Check with Pin U7/R1/reg_data_reg[29]/CK->SI
          Group: clk
     Startpoint: (R) U4/reg_data_reg[24]/CK
          Clock: (R) clk
       Endpoint: (R) U7/R1/reg_data_reg[29]/SI
          Clock: (R) clk

                    Capture       Launch
      Clock Edge:+    10000            0
      Src Latency:+       0            0
      Net Latency:+       0 (I)        0 (I)
          Arrival:=   10000            0

            Setup:-     206
     Required Time:=   9794
      Launch Clock:-      0
         Data Path:-   9134
             Slack:=    660
```



```
@genus:root: 4> report_area
============================================================
  Generated by:           Genus(TM) Synthesis Solution 20.10-p001_1
  Generated on:           Jul 25 2024  05:40:57 pm
  Module:                 top_32bit
  Operating conditions:   PVT_0P9V_125C (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

      Instance            Module       Cell Count  Cell Area  Net Area   Total Area   Wireload
------------------------------------------------------------------------------------------------
top_32bit                                    3945   9713.826     0.000     9713.826 <none> (D)
  U3                  ProgramMem               32    240.768     0.000      240.768 <none> (D)
  U4                  register_32bit_8         33    201.096     0.000      201.096 <none> (D)
  U6                  ALU_32bit              3017   6106.068     0.000     6106.068 <none> (D)
    U0                ieee_32add             1137   1897.416     0.000     1897.416 <none> (D)
      pe              priority_encoder        425    705.204     0.000      705.204 <none> (D)
    U3                ieee_32mul             1515   3556.800     0.000     3556.800 <none> (D)
  U7                  RegBank                 831   3122.118     0.000     3122.118 <none> (D)
    R0                register_32bit           33    263.340     0.000      263.340 <none> (D)
    R1                register_32bit_130       33    263.340     0.000      263.340 <none> (D)
    R2                register_32bit_129       33    263.340     0.000      263.340 <none> (D)
    R3                register_32bit_128       33    263.340     0.000      263.340 <none> (D)
    R4                register_32bit_127       33    263.340     0.000      263.340 <none> (D)
    R5                register_32bit_126       33    263.340     0.000      263.340 <none> (D)
    R6                register_32bit_125       33    263.340     0.000      263.340 <none> (D)
    R7                register_32bit_124       33    263.340     0.000      263.340 <none> (D)
    U0                mux_8to1                236    411.358     0.000      411.358 <none> (D)
    U1                mux_8to1_55             254    431.330     0.000      431.330 <none> (D)
  decoder_3to8_inst decoder_3to8              13     19.494     0.000       19.494 <none> (D)
(D) = wireload is default in technology library
```

Therefore, how many steps there are in the pipeline are considered as 3.

Future scope should focus on further optimizing 32-bit processor designs to balance performance and power consumption. Investigating advanced power-saving techniques, such as dynamic voltage and frequency scaling (DVFS) and clock gating, lower power consumption without compromising performance. Additionally, exploring hybrid architectures that combine the benefits of 16-bit and 32-bit processors could offer a versatile solution for a wider range of applications. Another promising priority is the growth of adaptive processors that can effortlessly alternate between 16-bit and 32-bit modes derived from the computational demands of the application, thereby optimizing power usage and processing efficiency. Finally, further research into the integration of specialized co-processors for specific tasks, such as DSP and AI acceleration, can enhance the overall capabilities within the processor while maintaining energy efficiency.

References

[1] R. Bhat, D. Pandey, F. Ahmad and P. Gupta, "Analysis and Optimization of 16-bit RISC Processor and 32-bit MIPS Processor: A Review," 2023 3rd International Conference on Intelligent Technologies (CONIT), Hubli, India, 2023, pp. 1-6, doi: 10.1109/CONIT59222.2023.10205898

[2] Mrs. Rupali S. Balpande, Mrs.Rashmi S. Keote, Design of FPGA based Instruction Fetch & Decode Module of 32-bit RISC (MIPS) Processor, 2011 International Conference on Communication Systems and Network Technologies, 978-0-7695-4437-3/11, 2011 IEEE.

[3] Pravin S. Mane, Indra Gupta, M. K. Vasantha, Implementation of RISC Processor on FPGA, 1-4244-0726-5/06, 2006 IEEE.

[4] A. Kulshreshtha, A. Moudgil, A. Chaurasia and B. Bhushan, "Analysis of 16-Bit and 32- Bit RISC Processors," 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2021, pp. 1318-1324, doi: 10.1109/ICACCS51430.2021.9441873.

[5] Yagnesh Savaliya, Jenish Rudani “Design and Simulation of 32-Bit Floating Point Arithmetic Logic Unit using VerilogHDL”2020 International Research Journal of Engineering and Technology (IRJET), Nadiad, India.

[6] C. Venkatesan, M. T. Sulthana, M. G. Sumithra and M. Suriya, "Design of a 16-Bit Harvard Structure RISC Processor in Cadence 45nm Technology," 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), Coimbatore, India, 2019, pp. 173-178, doi: 10.1109/ICACCS.2019.8728479.

[7] S. Palekar and N. Narkhede, "32-Bit RISC processor with floating point unit for DSP applications," 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, 2016, pp. 2062-2066, doi: 10.1109/RTEICT.2016.7808202.

[8] M. B. I. Reaz, M. S. Islam and M. S. Sulaiman, "A single clock cycle MIPS RISC processor design using VHDL," ICONIP '02. Proceedings of the 9th International Conference on Neural Information Processing. Computational Intelligence for the E-Age (IEEE Cat. No.02EX575), Penang, Malaysia, 2002, pp. 199-203, doi: 10.1109/SMELEC.2002.1217806.

[9] Mr. Sagar P. Ritpurkar, Prof. Mangesh N. Thakare, Prof. Girish D. Korde, “Review on 32- bit MIPS RISC Processor using VHDL,” 2014 IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE), Wardha, India.

[10] Pantazi-Mytarelli, "The history and use of pipelining computer architecture: MIPS pipelining implementation," 2013 IEEE Long Island Systems, Applications and Technology Conference (LISAT), Farmingdale, NY, USA, 2013, pp. 1-7, doi: 10.1109/LISAT.2013.6578243.

[11] V. Patil, A. Raveendran, P. M. Sobha, A. David Selvakumar and D. Vivian, "Out of order floating point coprocessor for RISC V ISA," 2015 19th International Symposium on VLSI Design and Test, Ahmedabad, India, 2015, pp. 1-7, doi: 10.1109/ISVDAT.2015.7208116.

[12] V. Jain, A. Sharma and E. A. Bezerra, "Implementation and Extension of Bit Manipulation Instruction on RISC-V Architecture using FPGA," 2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT), Gwalior, India, 2020, pp. 167- 172, doi: 10.1109/CSNT48778.2020.9115759

[13] S. P. Ritpurkar, M. N. Thakare and G. D. Korde, "Design and simulation of 32-Bit RISC architecture based on MIPS using VHDL," 2015 International Conference on Advanced Computing and Communication Systems, Coimbatore, India, 2015, pp. 1-6, doi: 10.1109/ICACCS.2015.7324067.

[14] John Bunda, Don Fussell, W. C. Athas, Roy Jenevein," 16-bit vs. 32-bit instructions for pipelined microprocessors," 1993 ACM SIGARCH Computer Architecture News, doi: doi.org/10.1145/173682.165159

[15] J. -Y. Lai, C. -A. Chen, S. -L. Chen and C. -Y. Su, "Implement 32-bit RISC-V Architecture Processor using Verilog HDL," 2021 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), Hualien City, Taiwan, 2021, pp. 1-2, doi: 10.1109/ISPACS51563.2021.9651130.

[16] Sivarama P. Dandamudi, "A Guide to RISC Processor For Programmers And Engineers" in, Springer

[17] Preetam Bhosle, Hari Krishna Moorthy, "FPGA Implementation of low power pipelined 32-bit RISC Processor", International Journal of Innovative Technology and Exploring Engineering (IJITEE), August 2012.

[18] M. N. Topiwala and N. Saraswathi, "Implementation of a 32-bit MIPS based RISC processor using Cadence," 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies, 2014, pp. 979-983.

[19] Patra, S., Kumar, S., Verma, S., Kumar, A. (2020). Design and Implementation of 32-bit MIPS-Based RISC Processor. In: Dutta, D., Kar, H., Kumar, C., Bhadauria, V. (eds) Advances in VLSI, Communication, and Signal Processing. Lecture Notes in Electrical Engineering, vol 587. Springer, Singapore.

[20] P. Trivedi and R. P. Tripathi, "Design & analysis of 16-bit RISC processor using low power pipelining," International Conference on Computing, Communication & Automation, 2015, pp. 1294-1297.