COMPARATIVE ANALYSIS AND DESIGN IMPROVEMENT

Table of Content

Introduction

In modern computing, the performance of a processor plays a pivotal role in determining the efficiency of a system. Analyzing and enhancing processor performance is critical to meet the growing demands for faster and more reliable computations. By evaluating various architectures and optimizing key components, such as multipliers, designers can significantly improve processing speed and efficiency while addressing bottlenecks.

This chapter focuses on comparing different processor architectures, including single-cycle and pipelined designs, to analyze their performance under varying configurations. Special attention is given to the integration of multipliers, where two designs—the Carry Ripple Adder and the Carry Save Adder—are examined. The goal is to demonstrate how these multiplier implementations impact the performance of single-cycle and pipelined processors, and to identify improvements that optimize overall efficiency.

Performance of a Processor

The performance of a processor is a crucial factor in determining the efficiency and speed of a computing system. It reflects how well the processor executes instructions within a given period. Analyzing processor performance allows architects to identify bottlenecks and improve design efficiency. Performance is typically evaluated using metrics such as clock frequency, instructions per cycle, and execution time.

Key Metrics for Measuring Performance

Clock Frequency (MHz or GHz):

The operating speed of the processor, representing the number of cycles it can execute per second. Higher frequencies generally indicate faster processors, but performance also depends on architectural efficiency.

CPI (Clock per Instruction)

The average number of clock cycles required to execute a single instruction. CPI accounts for factors like instruction complexity, pipeline stalls, and memory latency. Lower CPI values indicate better performance.

$$\textbf{CPI} = \frac{\textbf{Total Clock Cycles}}{\textbf{Total Instructions Executed}}$$

Instruction per Second (IPS)

The number of instructions the processor executes in one second. It is derived from the clock frequency and CPI:

$$\textbf{IPS} = \frac{\textbf{Clock Frequency}}{\textbf{CPI}}$$

Execution Time

The time taken to execute a specific number of instructions, often measured for benchmarks or real workloads. For example, the time to execute one billion instructions can be calculated as:

$$\textbf{Execution Time} = \frac{\textbf{Number of Instructions}}{\textbf{IPS}}$$

Performance of Single Cycle vs Pipelined Processor

Single Cycle Processor

For single-cycle processors, CPI is typically 1, as each instruction completes in one cycle. The Maximum Frequency on which the design can run properly is **98.1Mhz** calculated through timing analyzer of Quartus Prime Software. So IPS for Single cycle processor will be

IPS = 98.1M (instruction per second)

5 Stage Pipelined Processor

The Maximum clock frequency for the 5 stage Pipelined processor is **139Mhz** calculated through timing analyzer of Quartus Prime Software. For the CPI lets calculate it so lets assume a case.

There are approximately 22% loads, 13% stores, 12% branches, 3% jumps, and 50% R- or I-type ALU instructions out of which 30% load instruction follows with a register used in load as destination, 20% of the branches are taken, 20% of R-Type and I-Type follows with the same register used as destination by the R and I type and 30% of R-Type and I-Type instruction following after one instruction contain the same register used as destination by the R and I type. Now the CPI would be.

Introduce **1 stall cycle** for 30% of the loads.
Effective stalls = 22% × 30% = 6.6%

When branches are taken, **2 stall cycles** are introduced.
Effective stalls = 12% × 20% × 2 = 2.4%

No stall Cycle for R-type and I-type as there is a forwarding unit present.

CPI = 1 + 0.066 + 0.024 = 1.07
IPS = 139/1.07 = 129.9M (instruction/second)

6 Stage Pipelined Processor

The Maximum clock frequency for the 5 stage Pipelined processor is **206.9Mhz** calculated through timing analyzer of Quartus Prime Software. For the CPI lets calculate it so lets assume a case.
Taking the same example for this as well so

Introduce 2 **stall cycle** for 30% of the loads.
Effective stalls = 22% × 30% x 1 = 13.2%

When branches are taken, 3 **stall cycles** are introduced.
Effective stalls = 12% × 20% × 2 = 3.6%

Introduce 1 **stall cycle** for the R and I-type follow with a same register instruction
Effective stalls = 50% × 20% × 1 = 10%

No stall cycle for the R and I-type followed by the second instruction with same resister.

CPI = 1 + 0.132 + 0.036 + 0.1= 1.268
IPS = 206.9/1.268 = 163.17M (instruction/second)

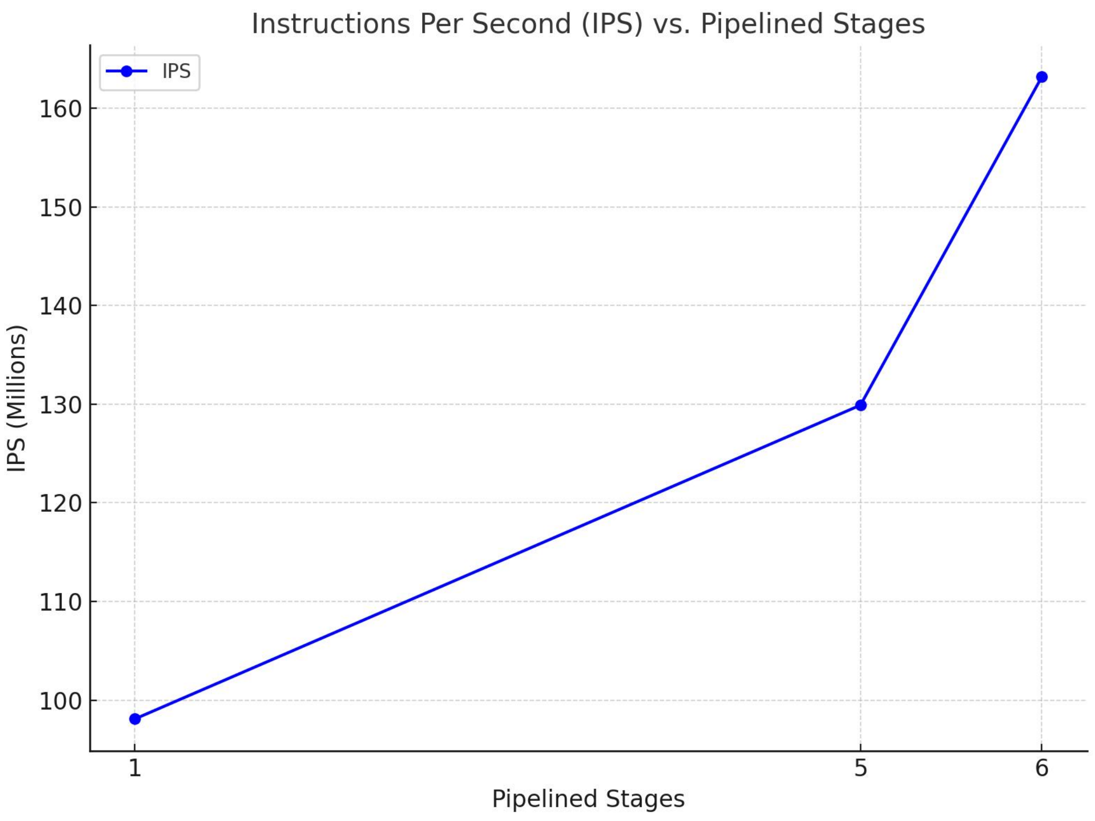


Figure 1 : Performance Graph IPS vs Number of Pipeline stage

The 6-stage pipelined processor demonstrates superior performance compared to its counterparts, achieving an IPS of 163.17 million, which significantly surpasses the 5-stage (129.9M IPS) and single-cycle (98.1M IPS) architectures. This improvement is attributed to enhanced instruction throughput facilitated by deeper pipelining, which reduces the execution time per instruction. Despite the added complexity of an additional stage, the 6-stage design effectively balances instruction latency and overall system performance, making it the most efficient architecture among the three. The architectural diagram of the Six stage Pipelined Processor is attached below where ALU stage is pipeline in 2 namely Execute and X stage.

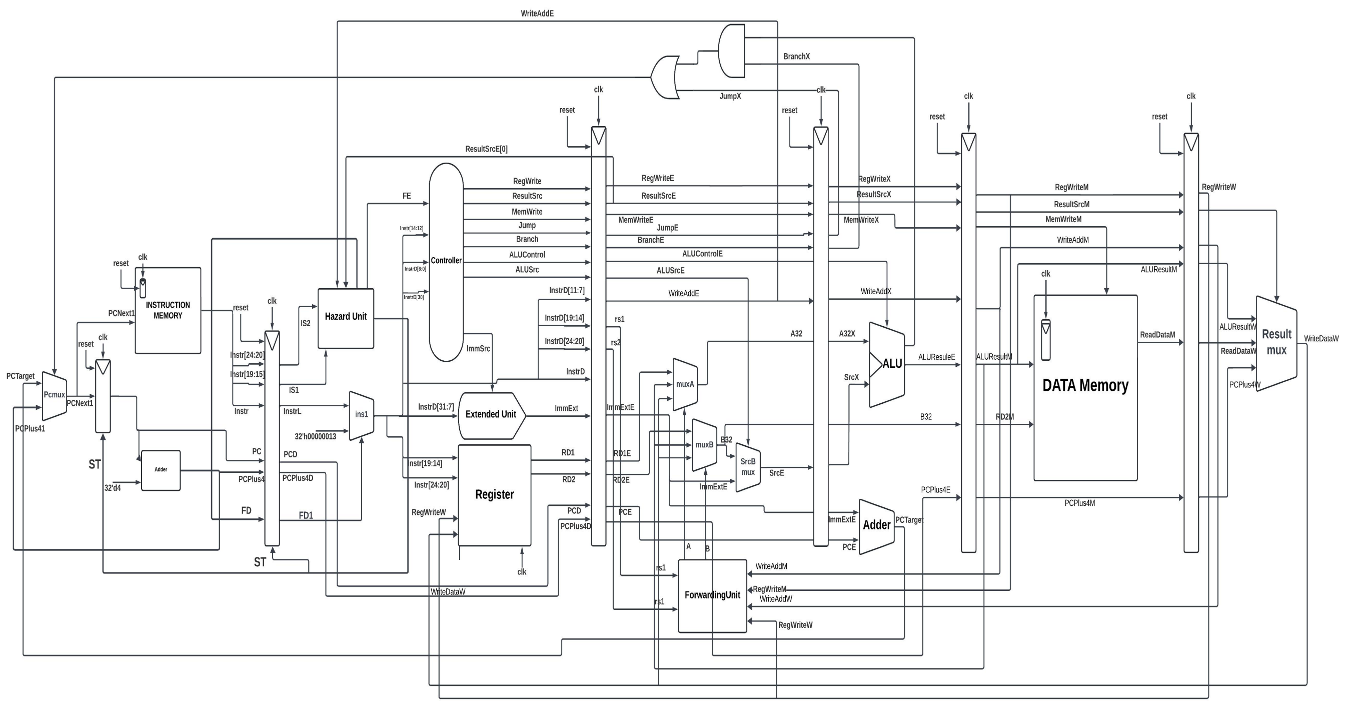


Figure 2 : 6 stage Pipelined Processor Design

While the 6-stage pipelined processor strikes an optimal balance between performance and complexity, extending the pipeline to 7 or more stages introduces diminishing returns. To achieve a 7-stage pipeline, stages such as instruction fetch or memory access would need to be split further. This division increases the number of intermediate stages, which, in turn, leads to higher stall cycles due to additional data hazards and control dependencies. As a result, the overall throughput may not improve significantly and could even degrade due to the increased penalty of pipeline stalls. Therefore, going beyond 6 stages is not advisable, as the performance gains plateau while the complexity and latency penalties escalate.

4-Stage Pipelined Multiplier

The 4-stage pipelined multiplier is a highly efficient architecture for performing multiplication operations, optimized for a maximum operating frequency of 137 MHz. This design leverages pipeline registers to divide the multiplication process into distinct stages, ensuring improved throughput and reduced latency by allowing multiple multiplication operations to be executed simultaneously in a pipelined manner.

Architecture Overview

Stage 1: Partial Product Generation

The inputs, A[31:0] and B[31:0], are used to generate 32 partial products using bitwise AND operations. Each bit of operand B is ANDed with the entire operand A, resulting in 32 rows of partial products.

Stage 2: First-Level Addition

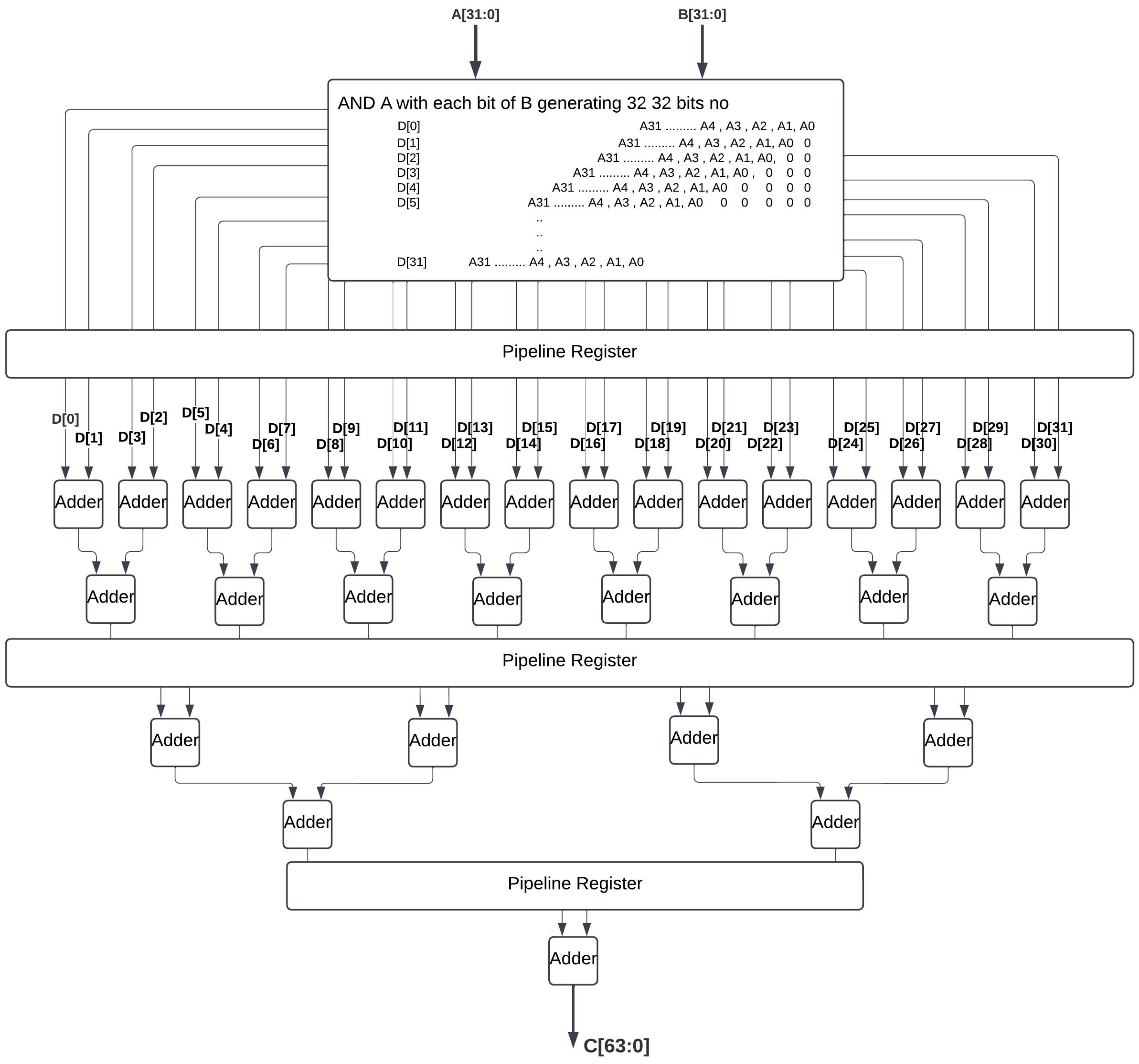
The generated partial products are passed to a set of carry ripple adders in parallel, which reduce the 32 rows into intermediate sums. This stage organizes the addition in a hierarchical structure to minimize delay and balance the workload across the pipeline.

Stage 3: Intermediate Reduction

The intermediate sums are further processed through additional adders, reducing the number of operands while maintaining accuracy. Pipeline registers are strategically placed to store intermediate results and ensure proper timing alignment.

Stage 4: Final Addition

In the final stage, the reduced partial products are combined to generate the final 64-bit product, C[63:0]. This stage completes the computation, with the result ready at the output after all pipeline stages have completed their tasks.



A[31:0]
B[31:0]
AND A with each bit of B generating 32 32 bits no
D[0] A31 .... A4 , A3 , A2 , A1, A0
D[1] A31 .... A4 , A3 , A2 , A1, A0 0
D[2] A31 .... A4 , A3 , A2 , A1, A0, 0 0
D[3] A31 .... A4 , A3 , A2 , A1, A0 , 0 0 0
D[4] A31 .... A4 , A3 , A2 , A1, A0 0 0 0 0
D[5] A31 .... A4 , A3 , A2 , A1, A0 0 0 0 0 0
..
..
..
D[31] A31 .... A4 , A3 , A2 , A1, A0
Pipeline Register
D[0] D[1] D[2] D[3] D[4] D[5] D[6] D[7] D[8] D[9] D[10] D[11] D[12] D[13] D[14] D[15] D[16] D[17] D[18] D[19] D[20] D[21] D[22] D[23] D[24] D[25] D[26] D[27] D[28] D[29] D[30] D[31]
Adder
Pipeline Register
Adder
Pipeline Register
Adder
C[63:0]

Figure 3 : 4-Stage Multiplier Design

Performance Analysis

Max Frequency

The 4-stage pipelined multiplier achieves a maximum frequency of 137 MHz, indicating the time required for a single stage is well-optimized.

Throughput

The pipelined nature of the design ensures high throughput, as each stage operates independently, allowing for a new multiplication operation to be initiated every clock cycle.

Latency

While the design introduces a latency of 4 cycles, the increased throughput compensates for the delay, making it suitable for high-performance computing applications.

This architecture balances complexity and performance, providing a robust solution for multiplication operations in pipelined processor designs. Its effectiveness can be attributed to the use of pipeline registers and an organized hierarchical addition structure.

Integration into a 6-Stage Pipelined Processor

When integrated into the 6-stage pipelined processor, the 4-stage pipelined multiplier operates at a maximum frequency of 137 MHz. This demonstrates the seamless compatibility of the multiplier with the existing 6-stage architecture, ensuring high performance without compromising the processor's efficiency.

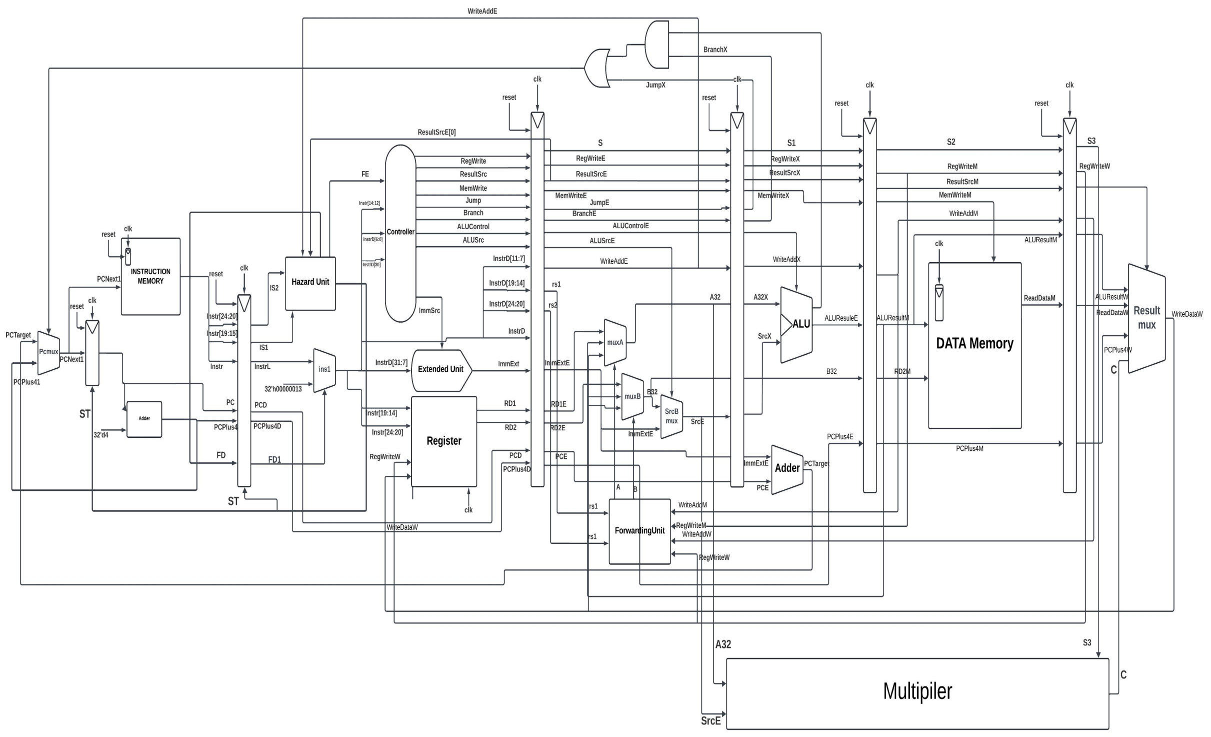


Figure 4 : 6-Stage Pipelined Processor with 4 stage Multiplier

However, this multiplier cannot be directly implemented in a 5-stage pipelined processor due to architectural constraints. To adapt it for a 5-stage pipeline, the multiplier design would need to be reduced to 3 stages. While this modification allows integration, it results in a significant reduction in performance, with the maximum frequency dropping to 89 MHz. This highlights the trade-offs involved in adapting multiplier designs for different pipeline architectures and underscores the superiority of the 6-stage processor in maintaining high frequencies with advanced components.

3-Stage Pipelined Multiplier with Carry Save Adder

This design include carry save adders (CSAs) to optimize the partial product accumulation process, significantly reducing carry propagation delays. The multiplier achieves a maximum frequency of 234 MHz, making it faster than the previous 4-stage design. It operates by performing AND operations on the inputs, generating partial products, and efficiently summing them through CSA stages, followed by a final addition to produce the 64-bit output. While this approach slightly increases area consumption due to the added CSA logic, the performance gain justifies the trade-off, especially in high-speed application.

Design Overview

The 3-stage pipelined multiplier incorporates carry save adders (CSAs) to optimize the performance of the multiplication operation. The use of CSAs reduces the carry propagation delay by breaking the addition process into smaller, faster stages, thereby enabling higher operating frequencies.

Input Handling

The multiplier begins by performing AND operations between the bits of inputs A[31:0] and B[31:0], generating partial products.

Carry Save Adder Stages

The partial products are processed through multiple levels of carry save adders to accumulate intermediate sums and carries efficiently.

Final Summation

At the final stage, the accumulated sum and carry values are passed through a pipeline register and combined using an adder to produce the 64-bit output C[63:0].

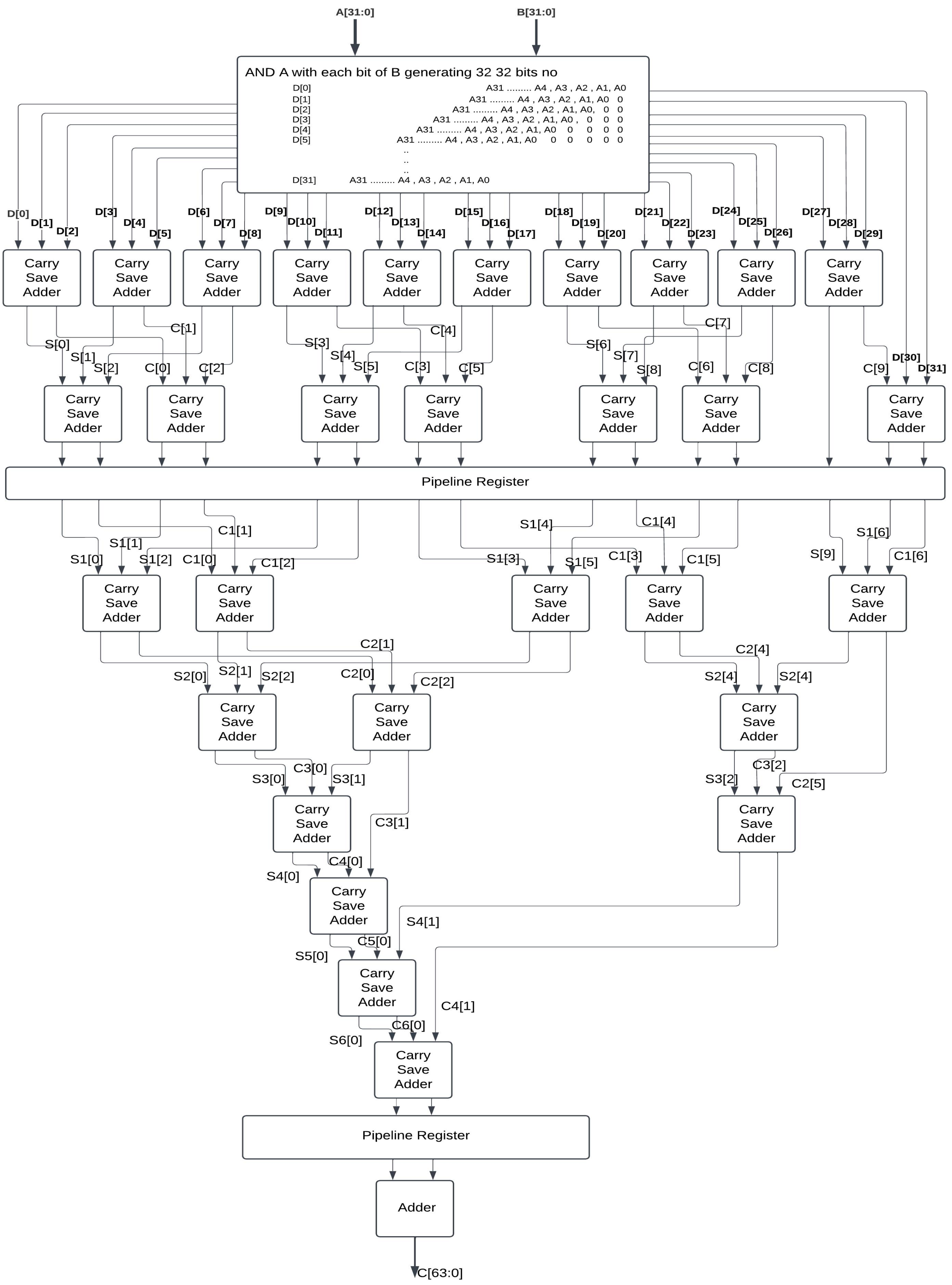

A[31:0]
B[31:0]
AND A with each bit of B generating 32 32 bits no
D[0] A31 .... A4 , A3 , A2 , A1, A0
D[1] A31 .... A4 , A3 , A2 , A1, A0 0
D[2] A31 .... A4 , A3 , A2 , A1, A0, 0 0
D[3] A31 .... A4 , A3 , A2 , A1, A0 , 0 0 0
D[4] A31 .... A4 , A3 , A2 , A1, A0 0 0 0 0
D[5] A31 .... A4 , A3 , A2 , A1, A0 0 0 0 0 0
D[31] A31 .... A4 , A3 , A2 , A1, A0
Carry Save Adder
Pipeline Register
Adder
C[63:0]

Figure 5 : Architecture of 3 stage Pipelined Multiplier

Performance Metrics

Maximum Frequency

The multiplier achieves a peak frequency of **234 MHz**, significantly higher than the previously discussed 4-stage design.

Area Consumption

The use of carry save adders introduces a slight increase in area consumption due to the additional logic elements required for carry-save operations. However, this trade-off is justified by the substantial performance gain.

Integration into Pipelines

When integrated into a **6-stage** or **5-stage pipelined processor**, the 3-stage pipelined multiplier with carry save adders does not impact the maximum frequency of the processors. The frequencies remain at **207 MHz** for the 6-stage design and **139 MHz** for the 5-stage design. This indicates that the multiplier's frequency capabilities are well above the processor's operational limits, ensuring seamless integration without introducing bottlenecks. This compatibility highlights the multiplier's efficiency and adaptability across different pipelined architectures.

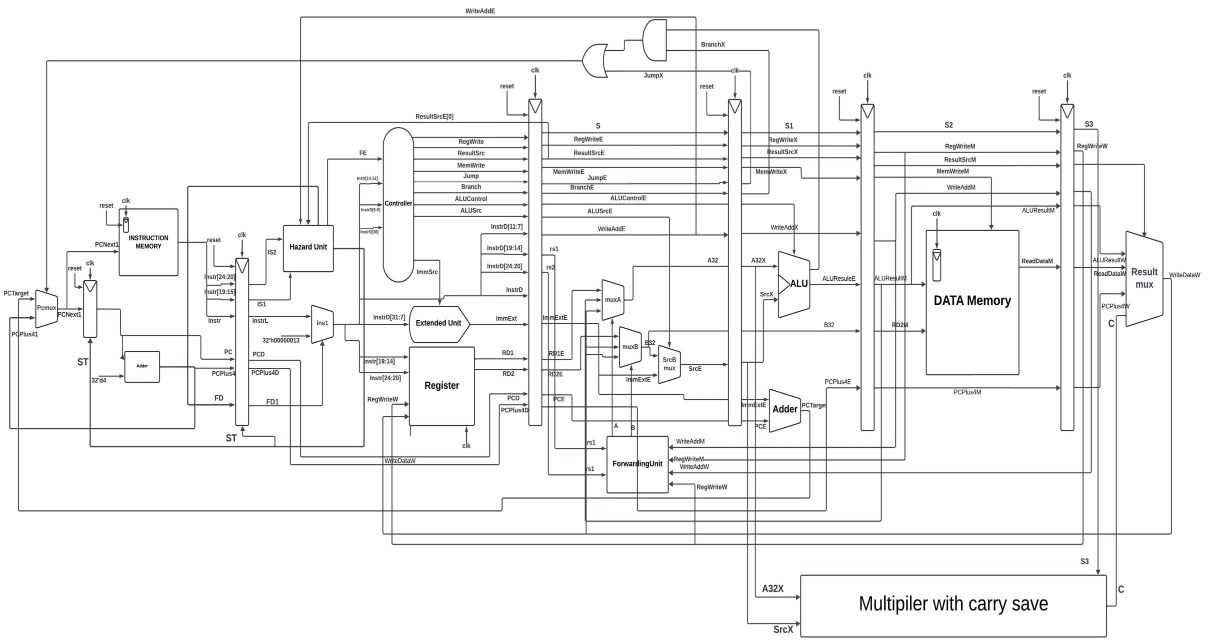


Figure 7 : 6-Stage Pipelined Processor with 3 stage Multiplier

The 3-stage pipelined multiplier with carry save adders is a superior choice compared to the 4-stage multiplier with carry ripple adders. While it has slightly higher area consumption, its significantly higher speed makes it more suitable for our design. Since we are implementing the processor on an FPGA, where achieving higher performance is critical, the 3-stage carry save multiplier is the preferred option to ensure optimal speed and efficiency in the overall system.

Performance of 6 Stage Pipelined Processor with Different Multiplier

To evaluate the performance of a 6-stage pipelined processor, we calculate the CPI (Cycles Per Instruction) and IPS (Instructions Per Second) for two multiplier designs: a 3-stage pipelined multiplier and a 4-stage pipelined multiplier. The performance is compared using instruction mix statistics, stalls due to hazards, and clock frequencies.

With 4 stage Multiplier

Consider the Same example that was discussed in performance of a processor that was There are approximately 20% loads, 13% stores, 12% branches, 3% jumps, and 50% R- or I-type ALU instructions out of which 30% load instruction follows with a register used in load as destination, 20% of the branches are taken, 15% of R-Type and I-Type follows with the same register used as destination by the R and I type, 27% of R-Type and I-Type instruction following after one instruction contain the same register us destination by the R and I type and 10% multiply instruction with 4% followed by the same register used as destination by the Mult. Now the CPI would be.

Introduce 2 **stall cycle** for 30% of the loads.
Effective stalls = 20% × 30% x 2 = 12%

When branches are taken, 3 **stall cycles** are introduced.
Effective stalls = 12% × 20% × 3 = 3.6%

Introduce 1 **stall cycle** for the R and I-type follow with a same register instruction
Effective stalls = 50% × 15% × 1 = 7.5%

introduce 3 **stall cycles** are when a Mult instruction is follow by same destination register
Effective stalls = 10% × 4% × 3 = 1.2%

No stall cycle for the R and I-type followed by the second instruction with same reggister.

CPI = 1 + 0.12 + 0.036 + 0.075 + 0.012 = 1.243
IPS = 137/1.243 = 110.22 Million (instruction/second)

With 3 stage Multiplier

Considering the same exapmle CPI would remain same but with the Max frequency IPS would be different so

IPS = 207/1.243 = 166.53 Million (instruction/second)

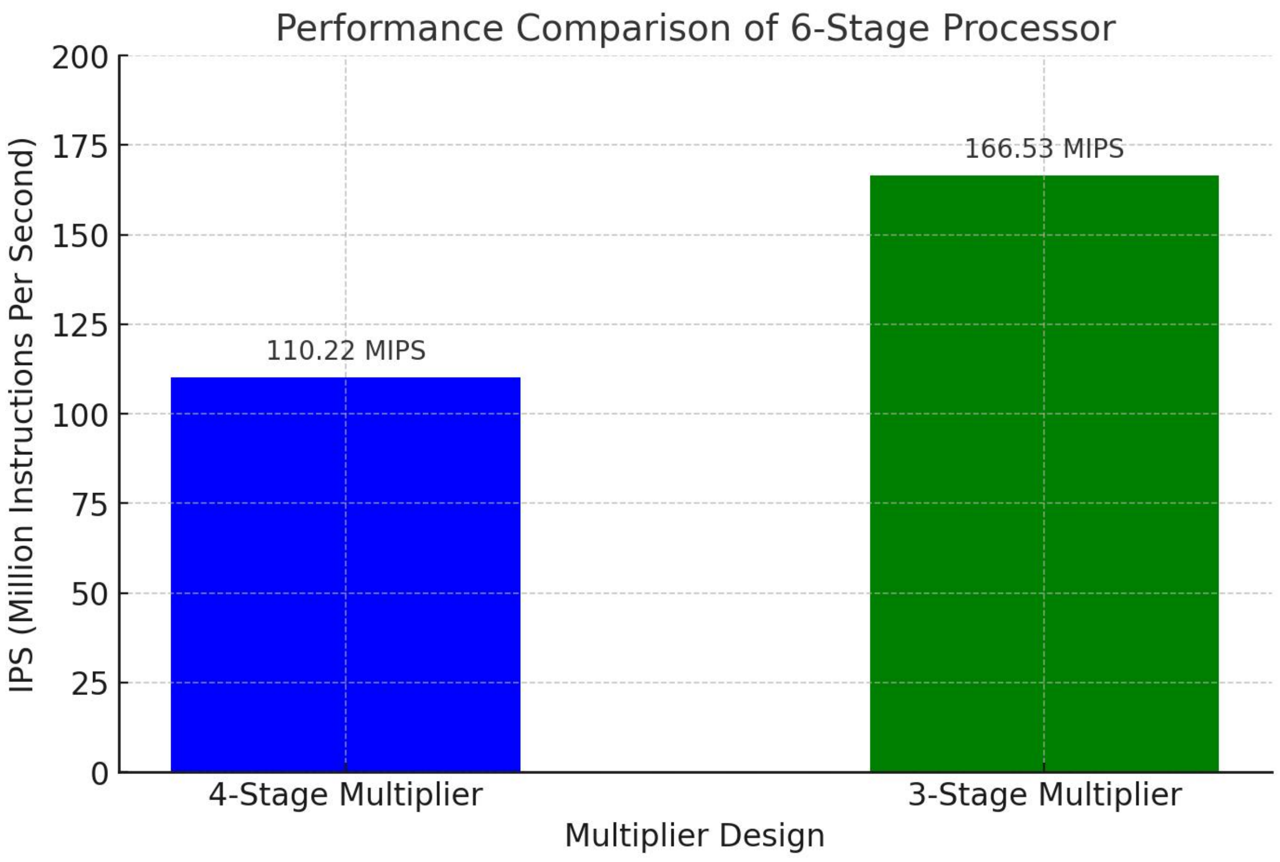


Figure 8 : Performance Comparision of 6-Stage processor with different Multipliers

Here is the performance comparison plot of a 6-stage pipelined processor using 4-stage and 3-stage multipliers. The 3-stage multiplier achieves a significantly higher IPS (166.53 MIPS) compared to the 4-stage multiplier (110.22 MIPS), reflecting the superior speed of the 3-stage design for FPGA implementations where performance is the primary concern.