# Simple Pipelined Processor: Documentation & Performance Report

Christian Turjuman

**Project Overview:**

The purpose of this project was to improve the performance of a single cycle processor built in our Digital Systems course lab by redesigning it with a five-stage pipeline. The result of this is demonstrated on the nandland Go Board (Lattice iCE40 HX1K-VQ100). In the original single cycle implementation every instruction had to pass through the program counter, instruction ROM, register file, ALU and write back in one clock period. While simple to follow, it makes the instruction throughput very slow. The redesign adds pipeline registers at all the stages, so that five instructions can run in parallel, and at every rising edge of the clock one instruction is finished. This improved performance by around 30%, while using 77% less LUT/FF resources.

**Pipeline Architecture:**

The new datapath is illustrated in *Figure 1* (block diagram) and summarised in *Table 1*. The Instruction-Fetch stage increments the program counter and reads a 16-bit opcode from a 256-word ROM. During Instruction-Decode the opcode is split into control fields while two 8-bit operands are read from a 4x8 register file. Execute performs one of six ALU functions (ADD, SUB, NOT, AND, OR, XOR) using a single carry chain. The Memory stage currently forwards the ALU result unchanged but reserves a cycle for future data-RAM access. Finally, Write-Back stores the result into the destination register so that the next instructions can consume it in subsequent cycles. Using pipeline registers between the block stages allow the resources to be reused every cycle, which improves performance and resource usage.

Figure 1: Pipeline Block Diagram

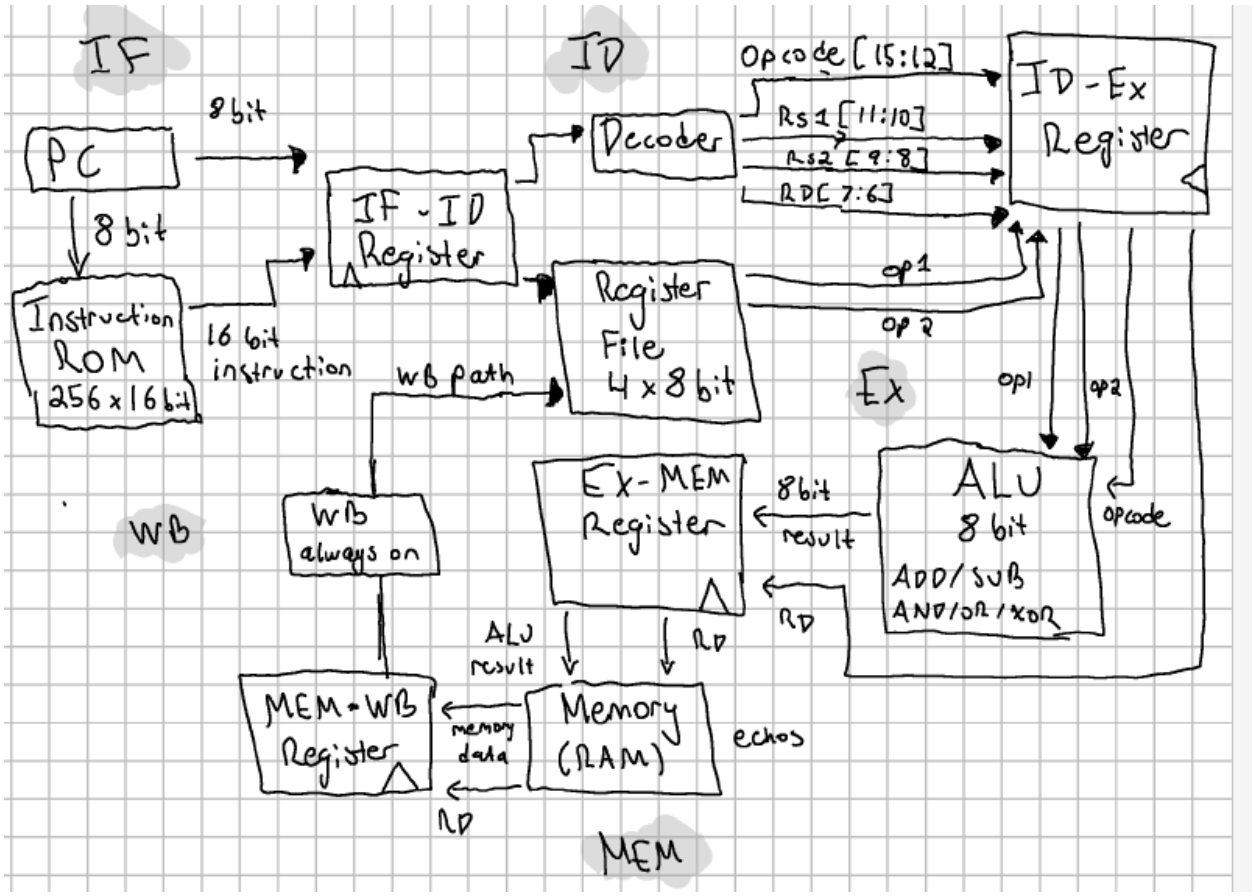



Table 1 : Pipeline stage responsibilities

| **Stage** | **RTL File** | **Function** |
|---|---|---|
| IF(instruction ffetch) | if_stage.v | Program counter increments, ROM fetch |
| ID(instruction decode) | id_stage.v | Decode, register reads two operands |
| EX(execute) | ex_stage.v | ADD/SUB/NOT/AND/OR /XOR |
| MEM(memory) | mem_stage.v | Slot for RAM(currently pass through) |

| | | |
|---|---|---|
| WB(write back) | Inside cpu.v | Write back to register file |

**Design Flow**

Development followed a conventional FPGA flow. Verilog RTL for each stage was written. Then the SystemVerilog testbench was written on *EDA Playground* runs 300 cycles of the full core, confirming that the program counter counts 0x00–0xFF and that the write-back-enable signal is never de-asserted. Both the reference single-cycle design and the new pipeline were then synthesised with and routed with IceCube2 using the same Go-Board pinout and 25 MHz clock SDC. Then flashed onto the board using DiamondProgrammer, the demonstration program was ran, and can be see in the demo video linked here:

demo video

**Performance Analysis**

The pipeline's advantages are seen in both resource utilization and performance. With an estimated clock speed of 128.4MHz vs 98.7MHz, there is roughly a 30% increase in clock speed performance for the pipelined design. In terms of resource utilization, the 5-stage design uses 46 LUTs, while the single cycle uses 200 LUTs, meaning this design uses almost 77% less resources.

Table 2: Performance(MIPS) Comparison
Single Cycle:

| Starting Clock | Requested Frequency | Estimated Frequency | Requested Period |
|---|---|---|---|
| i_Clk | 25.0 MHz | 98.7 MHz | 40.000 |

Pipelined:

| Starting Clock | Requested Frequency | Estimated Frequency | Requested Period |
|---|---|---|---|
| i_Clk | 25.0 MHz | 128.4 MHz | 40.000 |

Table 3: Resource Utilization Comparison

Single Cycle :

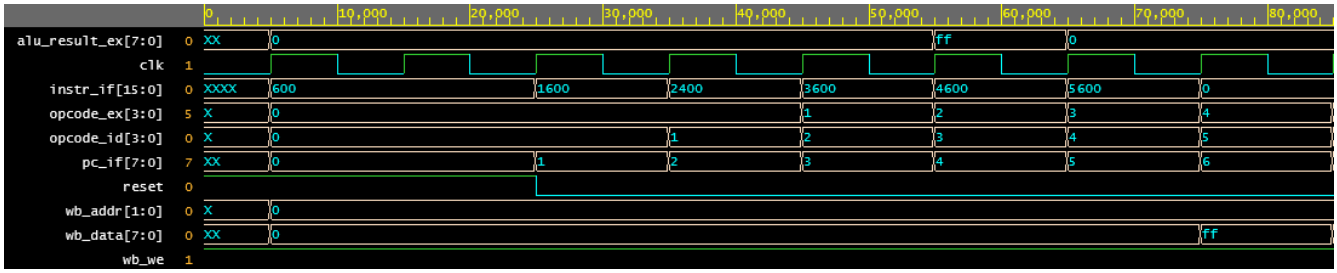
```
Resource Usage Report for simpleprocessor_top

Mapping to part: ice40hx1kvq100
Cell usage:
GND             8 uses
SB_CARRY        70 uses
SB_DFF          30 uses
SB_DFFE         19 uses
SB_DFFSR        72 uses
SB_GB           4 uses
VCC             8 uses
SB_LUT4         200 uses

I/O ports: 23
I/O primitives: 23
SB_GB_IO        1 use
SB_IO           22 uses

I/O Register bits:                  0
Register bits not including I/Os:   121 (9%)
Total load per clock:
   i_Clk: 1

@S |Mapping Summary:
Total  LUTs: 200 (15%)

Distribution of All Consumed LUTs = LUT4
Distribution of All Consumed Luts 200 = 200
```

Pipelined:

```
Resource Usage Report for simpleprocessor_top

Mapping to part: ice40hx1kvq100
Cell usage:
GND             6 uses
SB_CARRY        23 uses
SB_DFF          1 use
SB_DFFR         16 uses
SB_DFFSR        18 uses
SB_GB           1 use
VCC             6 uses
SB_LUT4         46 uses

I/O ports: 23
I/O primitives: 20
SB_GB_IO        1 use
SB_IO           19 uses

I/O Register bits:                  0
Register bits not including I/Os:   35 (2%)
Total load per clock:
   i_Clk: 1

@S |Mapping Summary:
Total  LUTs: 46 (3%)

Distribution of All Consumed LUTs = LUT4
Distribution of All Consumed Luts 46 = 46
```

**Testbench Review**

Timing waveforms in *Figure 2* taken from testbench show the overlapping execution of five successive instructions, highlighting that one result is committed every clock after the pipeline reaches steady state.




The waveform proves the one-instruction-per-clock steady state: after *reset* de-asserts, the fetch PC *pc_if* steps 0 -> 1 -> 2 -> 3 each cycle and the fetched word *instr_if* updates in the same cycle, while the decoded opcode *opcode_id* and executed opcode *opcode_ex* appear exactly one and two clocks later, showing the IF->ID->EX stages.

Two clocks after a valid opcode reaches EX the ALU begins producing results *alu_result_ex*, and the write-back enable *wb_we* is enabled continuously, confirming that once the pipeline registers are filled, every rising edge completes a new instruction.

**Hardware Demonstration**

To demonstrate this on hardware, the clock is way too fast to detect changes, so a 25-bit counter divides the board's 25 MHz crystal to about 1 Hz, making the program-counter value visible on the Go Board's dual seven-segment display. There is also a reset button on Switch 4, which resets the PC to 0x00. Then the digits then advance 00->01->02… etc each cycle. LED1 glows continuously, showing the write-back stage is active, while LEDs 2, 3 and 4 display PC[2:0] and therefore toggle at 1 Hz, 0.5 Hz, and 0.25 Hz respectively.