Hands-on Introduction to Computer Architecture with RISC-V

Adding a Pipeline Stage to the Single-Cycle Processor

Assignment 04

N.G.Kaween Newmal

SKF2400193

2026/01/04

Objective

- The given single - cycle RV32I processor was converted into a 2 - stage pipelined design.
    - ✓ Stage 1: Instruction Fetch (IF) + Instruction Decode (ID) + Execute (EXE)
    - ✓ Stage 2: Memory Access (MEM) + Writeback (WB)

- A pipeline register is inserted between Stage 1 and Stage 2. All pipeline registers are cleared to 0 on reset. The original single-cycle processor is kept in a separate source file for comparison.

Signals stored in the pipeline registers (Stage1 → Stage2)

The following signals are written into pipeline registers at every clock edge (unless reset):

Pipeline Reg Signal	*Width*	*Purpose (used in Stage 2)*
aluOut_s2	32	ALU result from Stage 1. Used as arithmetic result or load/store address
storeData_s2	32	Store data (rs2 value) for SB/SH/SW in Stage 2
pc_plus4_s2	32	PC + 4 for JAL/JALR writeback into rd
funct3_s2	3	Load/store size and sign control (LB/LH/LW/LBU/LHU, SB/SH/SW)
rd_s2	5	Destination register index for writeback
isMemRead_s2	1	Control: indicates Stage 2 should perform a load
isMemWrite_s2	1	Control: indicates Stage 2 should perform a store
isJAL_s2	1	Control: selects PC + 4 as writeback data (JAL)
isJALR_s2	1	Control: selects PC + 4 as writeback data (JALR)
regWriteEn_s2	1	Control: enables register file write in Stage 2

Hazards in a 2-stage pipelined design

Data hazards (RAW): Yes.

- A Read-After-Write hazard can occur when an instruction in Stage 1 reads a register that is being written by the previous instruction in Stage 2.
- Hazards are handled either by inserting NOPs in software (baseline) or by enabling forwarding (optional).

Control hazards: Limited.

- Branch/JAL/JALR targets are computed in Stage 1 and the PC is updated at the end of Stage 1, so the next fetch uses the correct PC.
- However, data dependencies can affect branch compares.

Structural hazards: Possible if instruction and data access share a single-port memory.

- This RTL uses one array mainMemory for both instruction fetch and data access; real hardware would typically use separate I-MEM and D-MEM or multi-port memory.

RISC-V program (Fibonacci n = 12) and hazard avoidance

- The required program stores n = 12 to data memory location 0x80 and loads it back , computes Fibonacci F(12) = 144 and writes the final result to register x10 (a0).

- Baseline hazard avoidance: The pipelined version inserts a single NOP after addi x2, x0, 12 before sw x2, 0(x1) to avoid a RAW hazard.

Hazard-avoidance snippet (program_pipe.txt):

```
addi x2, x0, 12
addi x0, x0, 0    // NOP (wait 1 cycle)
sw   x2, 0(x1)
```

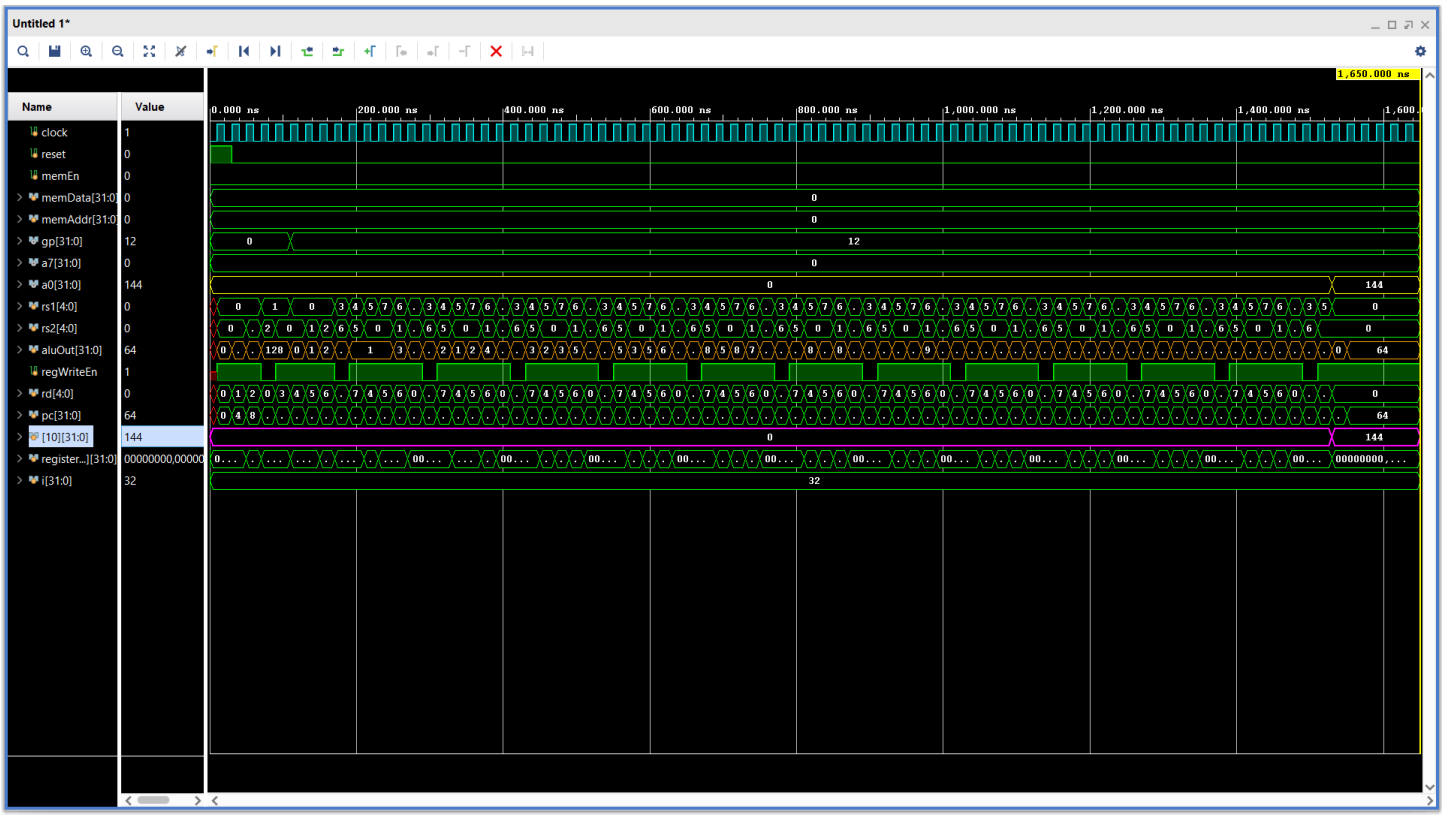
Simulation results and execution time comparison

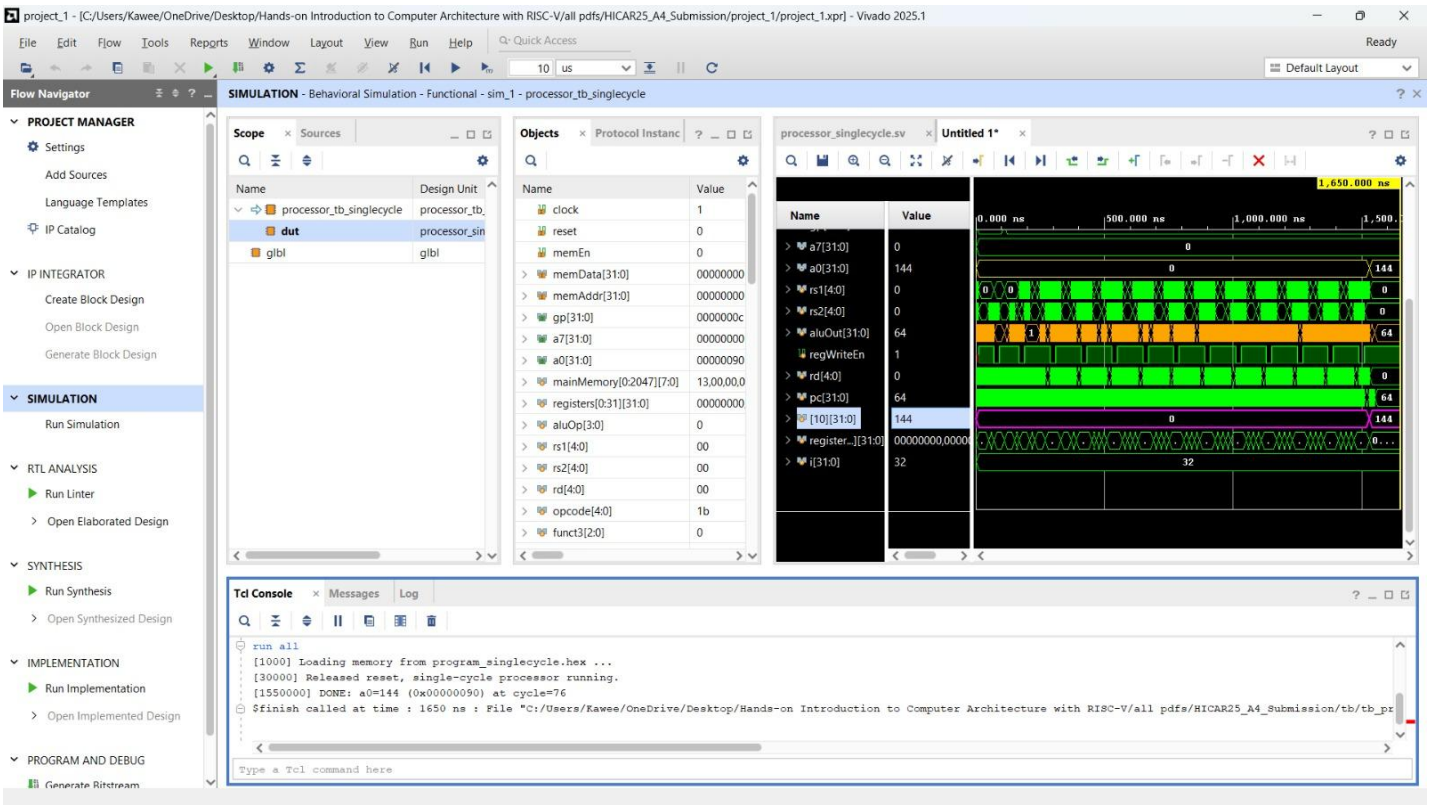
Expected final output: a0 (x10) = 144 (0x00000090).

Implementation	*Program instructions executed (until first JAL)*	*Clock period*	*Total cycles*	*Total time*
Single-cycle	66	20 ns	66	1320 ns
2-stage pipeline (no forwarding)	67	10 ns	67 + 1 = 68	680 ns
2-stage pipeline (forwarding enabled)	66	10 ns	66 + 1 = 67	670 ns

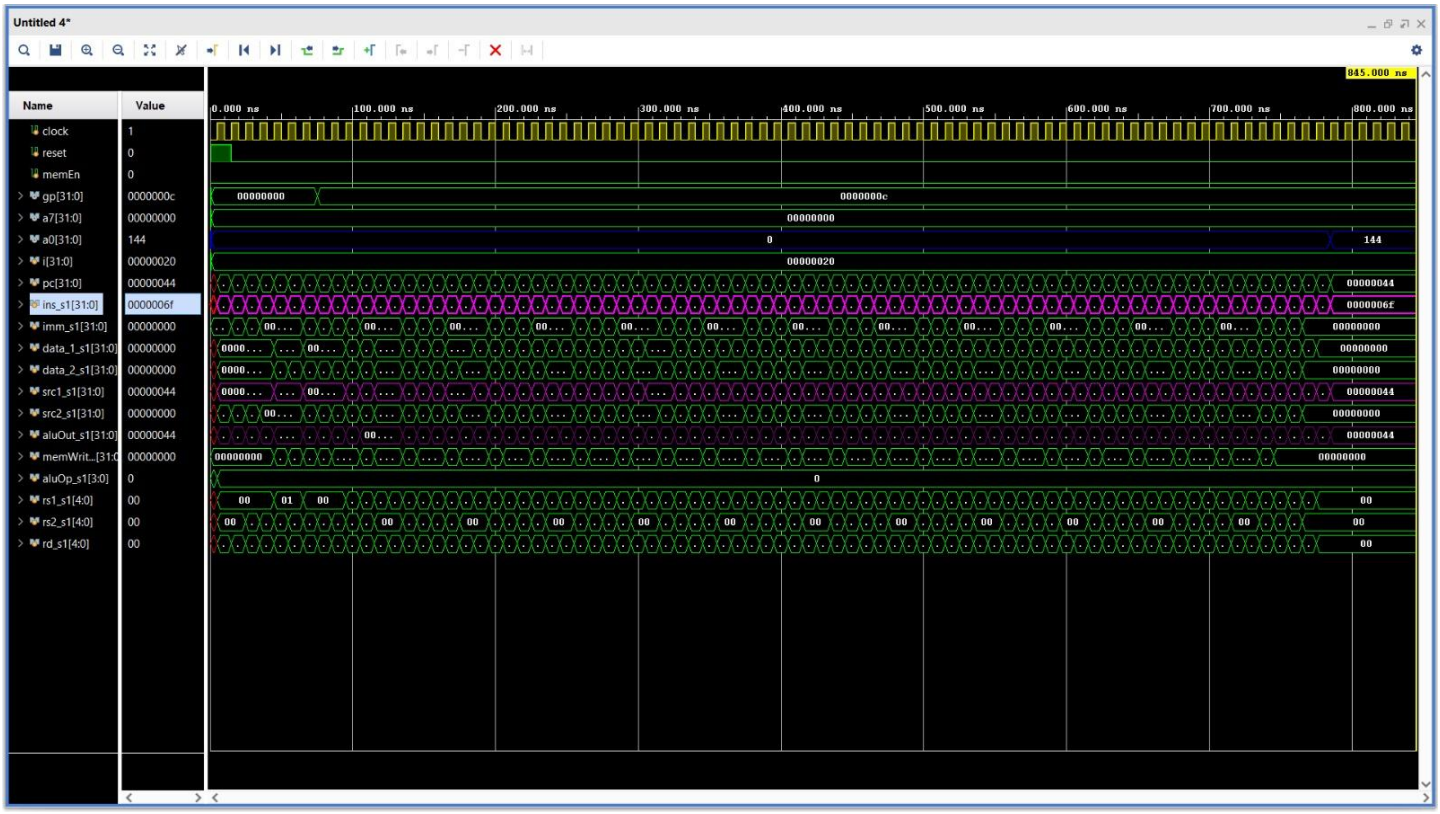
Simulation waveforms

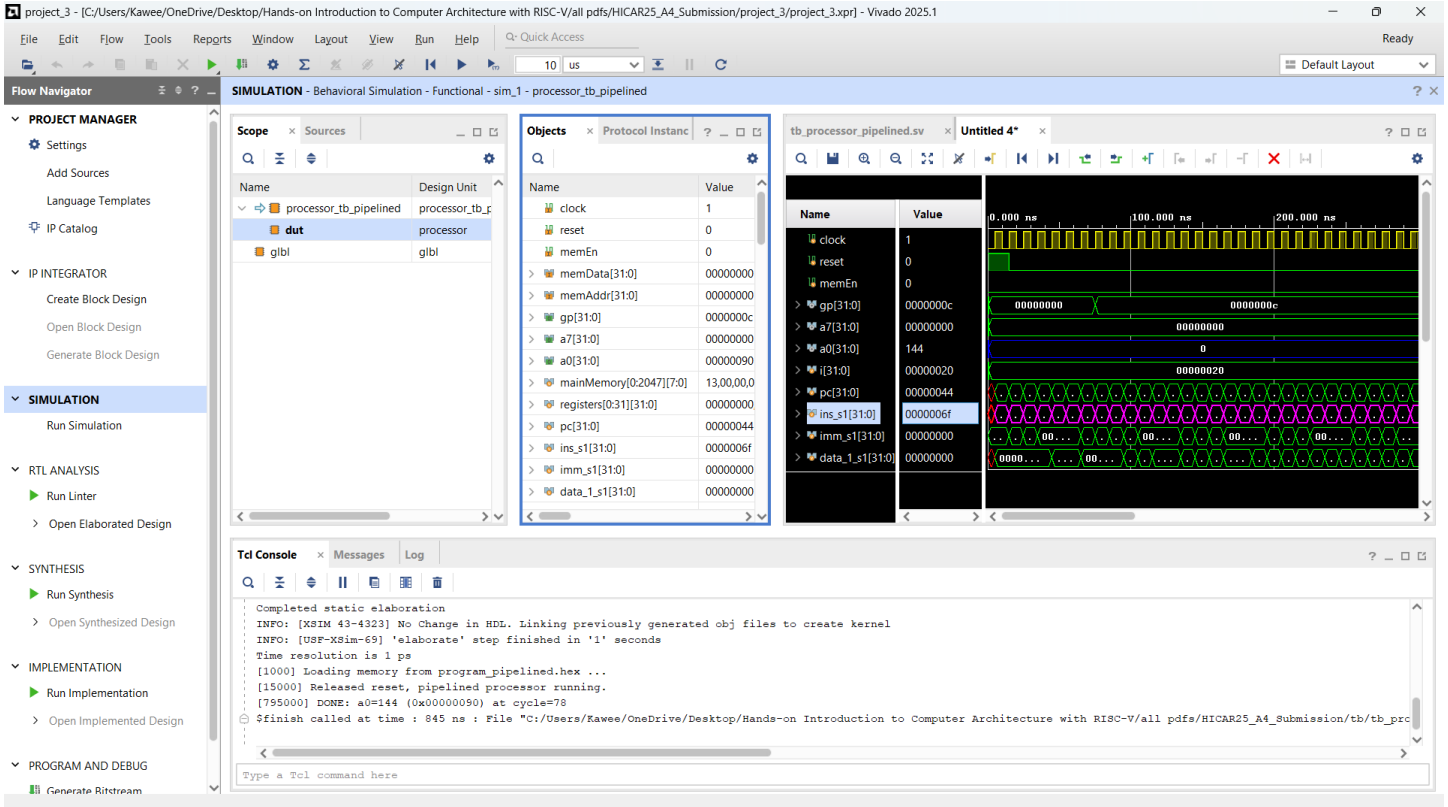
- *Single-cycle CPU*





➢ *Pipelined CPU (NO forwarding)*





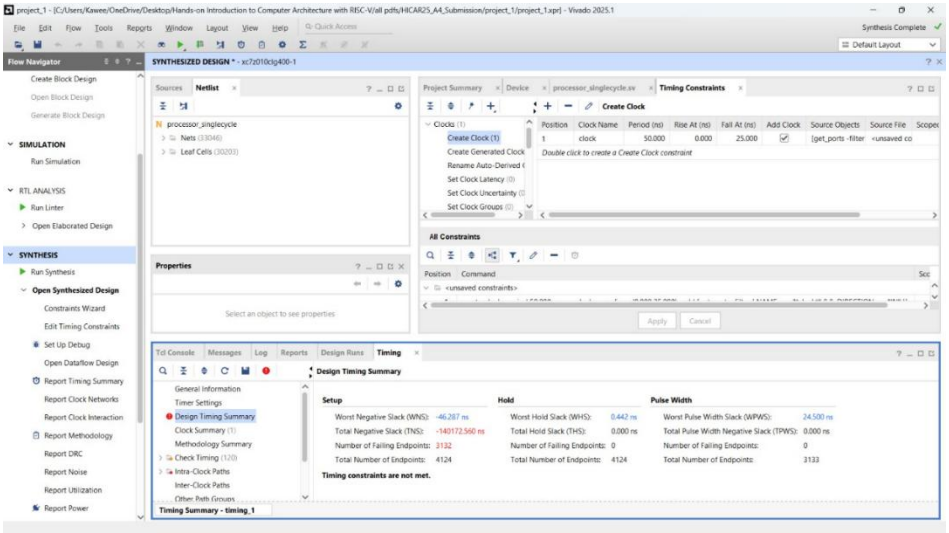
Maximum clock frequency comparison (Vivado Synthesis)

❖ Synthesize both designs with MEM_DEPTH = 256 and a 50 ns clock constraint. Record the WNS (Worst Negative Slack) from Vivado and compute:

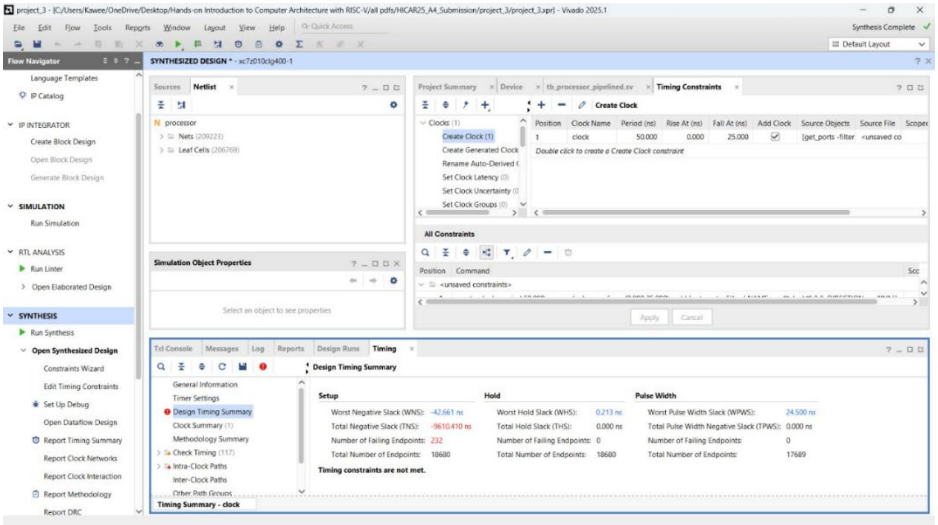
$$\text{max\_clk\_freq} = \frac{1}{\text{clk\_period} - \text{WNS}}$$

Design	*Clock period constraint (ns)*	*WNS (ns) from Vivado*	*max clock period*	*Max frequency f_max*
Single-cycle	50	− 46.287 ns	96.287	10.385618 MHz
2-stage pipeline	50	− 42.661 ns	92.661	10.793 MHz
2-stage pipeline (forwarding)	50	− 40.282 ns	90.282	11.0765 MHz

Single-cycle Timing report



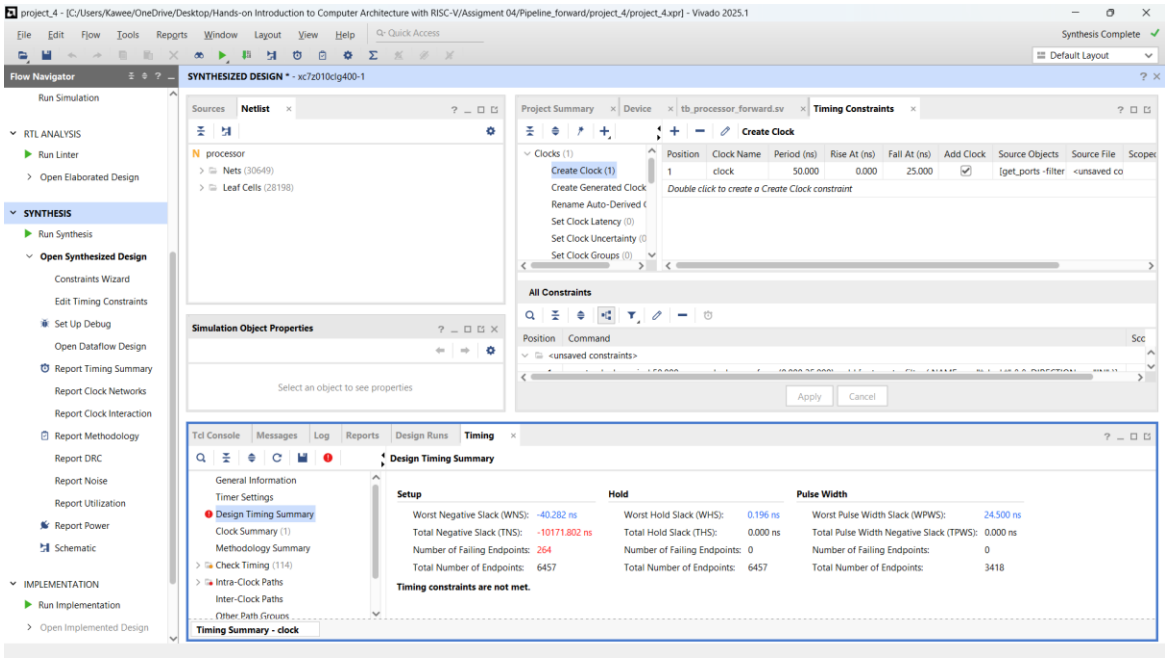
2-stage pipeline Timing report



Optional: Data forwarding

- Forwarding compares Stage-1 source regs with Stage-2 destination reg when writeback is enabled.
- With forwarding enabled, the program runs without inserted NOPs and still produces the correct final output.

2-stage pipeline (forwarding) Timing report



Performance comment

The pipelined processor improves through put by overlapping Stage 1 and Stage 2, and it typically allows a higher maximum clock frequency because each stage has a shorter critical path. Speedup is reduced by pipeline hazards and any NOPs inserted for hazard avoidance; forwarding reduces the need for software-level NOPs.

References

- Course slides