High-Performance Computing (HPC)

Pipelining

Design a datapath in Verilog to perform the operation

$$Y = (A + B) \times C + D$$

where each arithmetic operation (+, ×) is completed within a single clock cycle.

Write a **testbench** that applies multiple sets of input values (A, B, C, D) at regular intervals and records the corresponding outputs Y.

Clock Cycle	Non-Pipelined Output
1	Start (A_1,B_1,C_1,D_1) → compute Y_1
2	Output Y_1
3	Start Y_2
4	Start Y_3
5	—

Throughput Improvement using Pipelining

- **Stage 1:** Perform addition → $S_1 = A + B$
- **Stage 2:** Perform multiplication → $S_2 = S_1 \times C$
- **Stage 3:** Perform final addition → $Y = S_2 + D$

Compare the **non-pipelined** and **pipelined** versions in a table for:

- Clock period
- Latency
- Throughput
- Number of registers used
- Maximum operating frequency

Assumptions (for analysis)

- Adder delay = 10 ns
- Multiplier delay = 25 ns

Clock Cycle	Pipelined Stage-1	Pipelined Stage-2	Pipelined Stage-3	Pipeline Output
1	(A_1+B_1)	—	—	—
2	(A_2+B_2)	$(A_1+B_1)\times C_1$	—	—
3	(A_3+B_3)	$(A_2+B_2)\times C_2$	$((A_1+B_1)\times C_1)+D_1$	**Y_1 ready**
4	(A_4+B_4)	$(A_3+B_3)\times C_3$	$((A_2+B_2)\times C_2)+D_2$	**Y_2 ready**
5	—	—	$((A_3+B_3)\times C_3)+D_3$	**Y_3 ready**

Design and compare two versions of the arithmetic circuit:

$$Y = (A + B) \times C + D$$

1. **Non-pipelined design** – all operations in a single clock cycle.
2. **3-stage pipelined design** – stages for addition, multiplication, and final addition.
3. Compute and compare **latency** and **throughput** assuming
   - $t_{add} = 10\,\text{ns}$,
   - $t_{mul} = 25\,\text{ns}$.

Timing and Performance

- Total combinational delay

$$T_{clk} = t_{add} + t_{mul} + t_{add} = 10 + 25 + 10 = 45\,\text{ns}$$

- Max frequency $f_{max} = 1/45\,\text{ns} = 22.2\,\text{MHz}$
- **Throughput** = 1 output every 45 ns = **22.2 M results/s**
- **Latency** = 1 cycle (45 ns)

Example: 3-Stage Arithmetic Pipeline

Compute

$$Y = (A + B) \times C + D$$

with one operation per stage:

Stage	Operation	Register Output
Stage 1	Add	`sum_reg <= A + B`
Stage 2	Multiply	`mul_reg <= sum_reg * C`
Stage 3	Add	`Y <= mul_reg + D`

Each stage completes one partial result per clock cycle.

$$Y = (A + B) \times C + D$$

and

$$t_{add} = 10\,\text{ns}, \quad t_{mul} = 25\,\text{ns}$$

Pipeline stages:

Stage	Operation	Delay (ns)
1	Add (A + B)	10
2	Multiply by C	25
3	Add D	10

Therefore,

$$T_{clk} = \max(10, 25, 10) = 25\,\text{ns}$$

$$f_{max} = \frac{1}{T_{clk}} = 40\,\text{MHz}.$$

- **General Rule**

$$T_{clk} \geq \max(T_{stage1}, T_{stage2}, T_{stage3}, \dots)$$

where

- T_{stagei} = combinational delay of stage *i*,

Timing and Performance

Stage	Operation	Delay (ns)
1	$A + B$	10
2	× C	25
3	+ D	10

- $T_{clk} = \max(10, 25, 10) = 25\text{ns}$
- $f_{max} = 1/25\text{ns} = 40\text{MHz}$
- **Latency** = 3 cycles = 75 ns
- **Throughput** = 1 result every 25 ns = **40 M results/s**

Parameter	Non-Pipelined	3-Stage Pipelined	Improvement
Clock period	45 ns	25 ns	1.8× faster
Clock frequency	22.2 MHz	40 MHz	—
Latency	1 cycle (45 ns)	3 cycles (75 ns)	↑ latency
Throughput	22.2 M ops/s	40 M ops/s	**1.8× higher**
Hardware cost	1 reg	3 regs	Slightly higher

- **Non-pipelined design**: Simpler, but the entire path delay (45 ns) limits clock speed.
- **Pipelined design**: Divides computation into smaller stages; despite higher latency, it **improves throughput** because each stage works on a different input simultaneously.