SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

19ECB302–VLSI DESIGN

III YEAR/ V SEMESTER

UNIT 3 –SEQUENTIAL LOGIC CIRCUITS

TOPIC 4 –PIPELINES

OUTLINE

- Introduction
- Architectural techniques : critical path
- Synchronous timing
- Self-timed pipelined data path
- Completion signal using current sensing
- Architectural techniques :pipelining
- Activity
- Architectural techniques : fine-grain pipelining
- Unrolling the loop using pipelining
- Architectural techniques : parallel processing
- Assessment
- Summary & thank you

INTRODUCTION

Combinational logic

output depends on current inputs

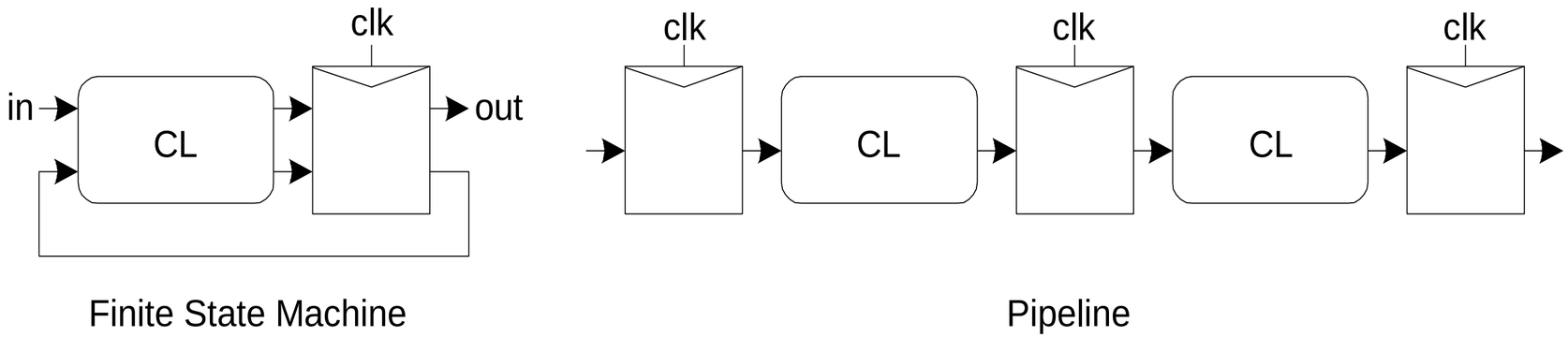
Sequential logic

output depends on current and previous inputs

Requires separating previous, current, future

Called state or tokens

Ex: FSM, pipeline

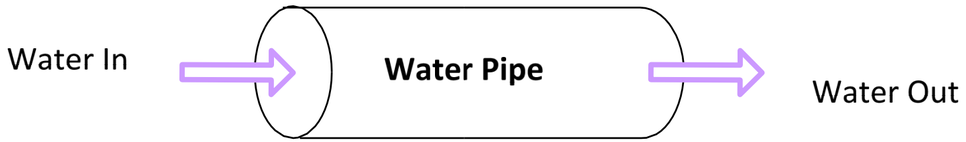


ARCHITECTURAL TECHNIQUES : CRITICAL PATH

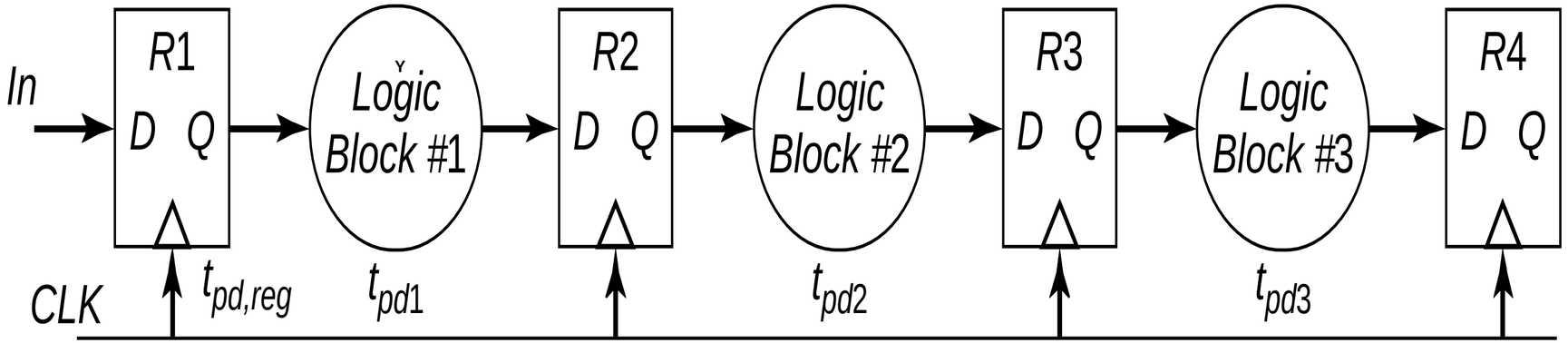
Critical path in any design is the longest path between

1. Any two internal latches/flip-flops
2. An input pad and an internal latch
3. An internal latch and an output pad
4. An input pad and an output pad

•Use FFs right after/before input/out pads to avoid the last three cases (off-chip and packaging delay)



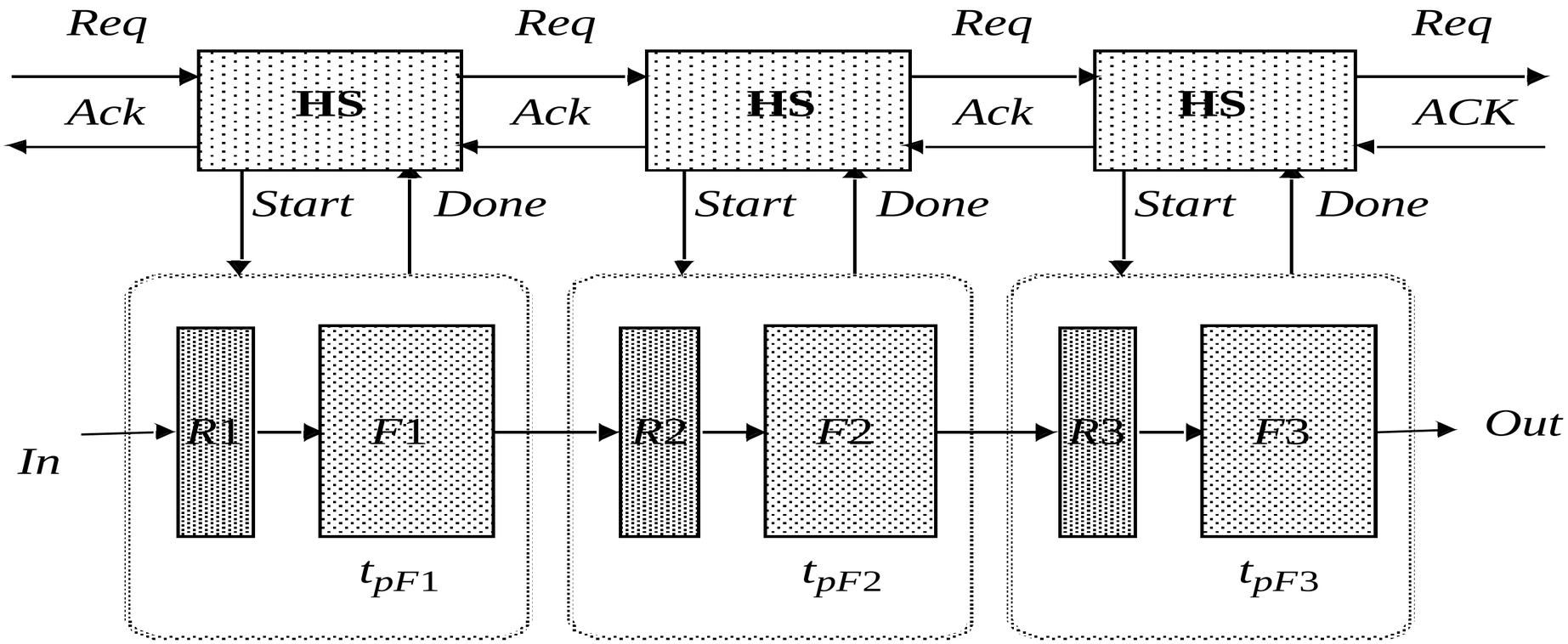
SYNCHRONOUS TIMING



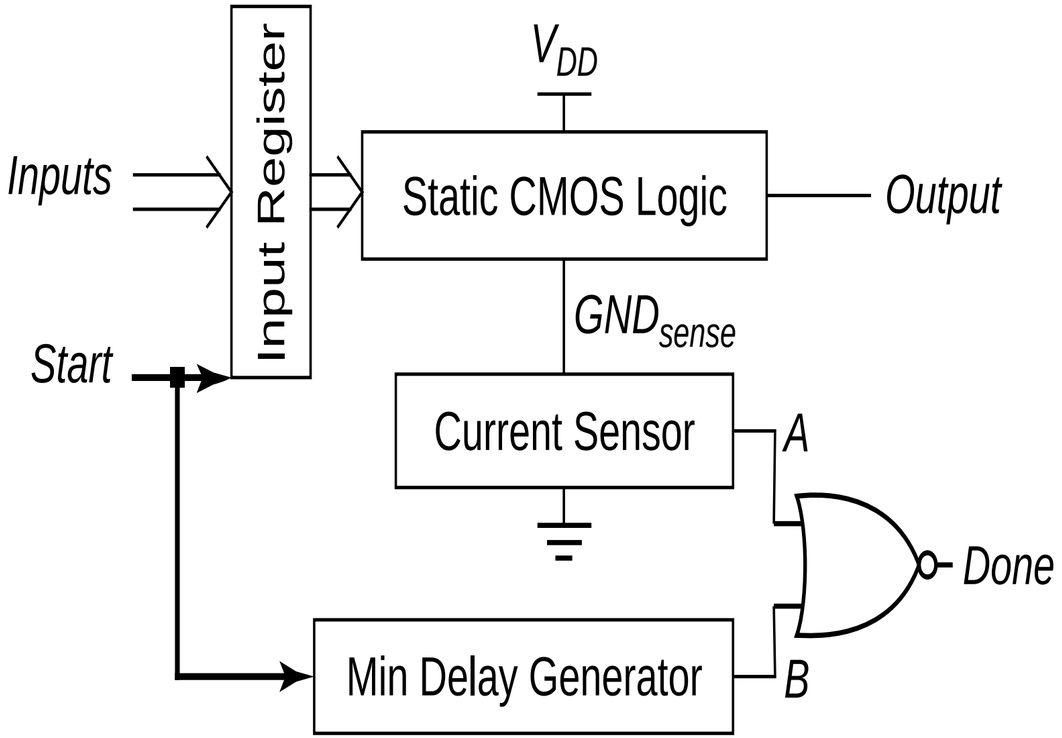
Pipelining:

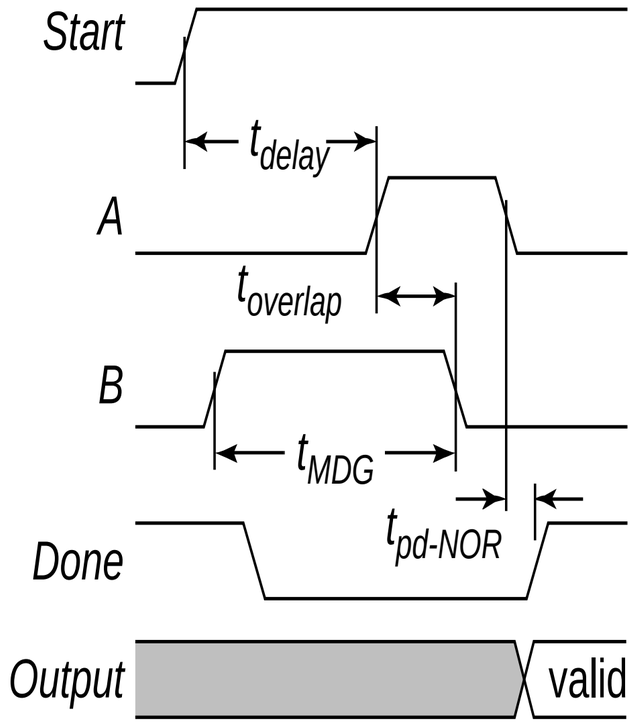
- Comes from the idea of a water pipe: continue sending water without waiting the water in the pipe t be out
- Used to reduce the critical path of the design

SELF-TIMED PIPELINED DATA PATH



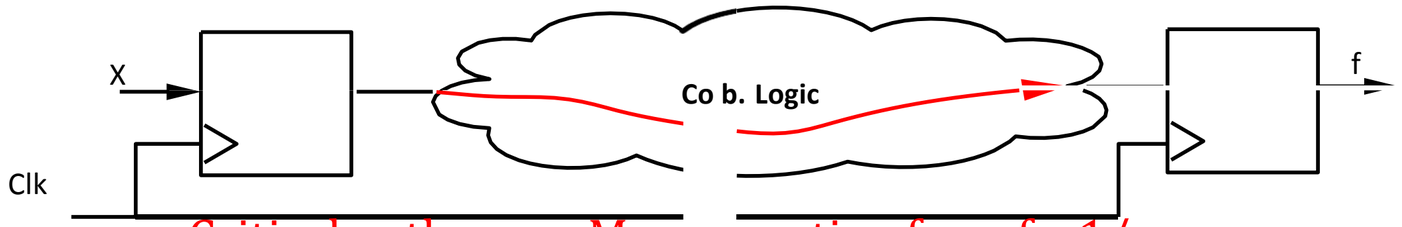
COMPLETION SIGNAL USING CURRENT SENSING





ARCHITECTURAL TECHNIQUES :PIPELINING

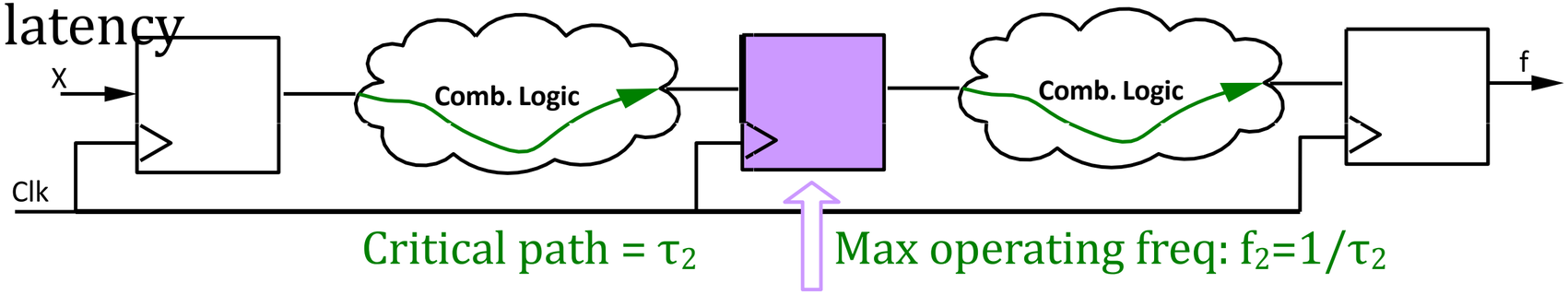
Original System: (Critical path = τ_1 Max operating freq: $f_1=1/\tau_1$**)**



2 cycles later

Pipelined version: (Critical path = τ_2 Max operating freq: $f_2=1/\tau_2$**)**

- Smaller Critical Path ➡ higher throughput ($\tau_2<\tau_1$ ➡ $f_2>f_1$)
- Longer latency

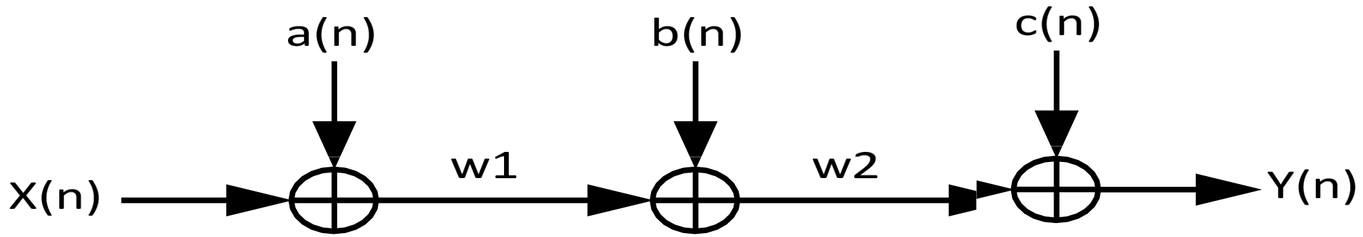


3 cycles later

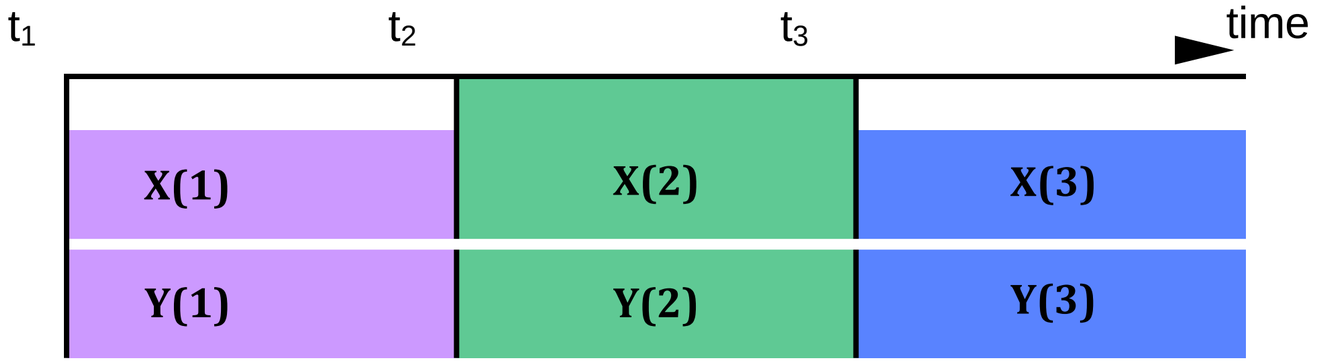
ARCHITECTURAL TECHNIQUES : PIPELINE DEPTH

Pipeline depth: 0 (No Pipeline)

Critical path: 3 Adders

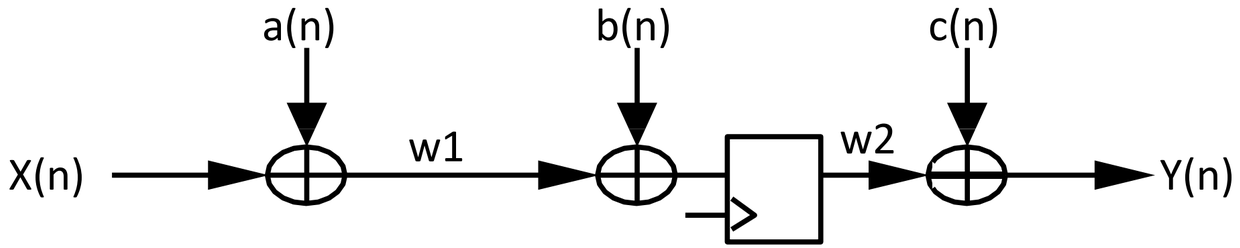



```
wire w1, w2;
assign w1 = X + a;
assign w2 = w1 + b;
assign Y = w2 + c;
```



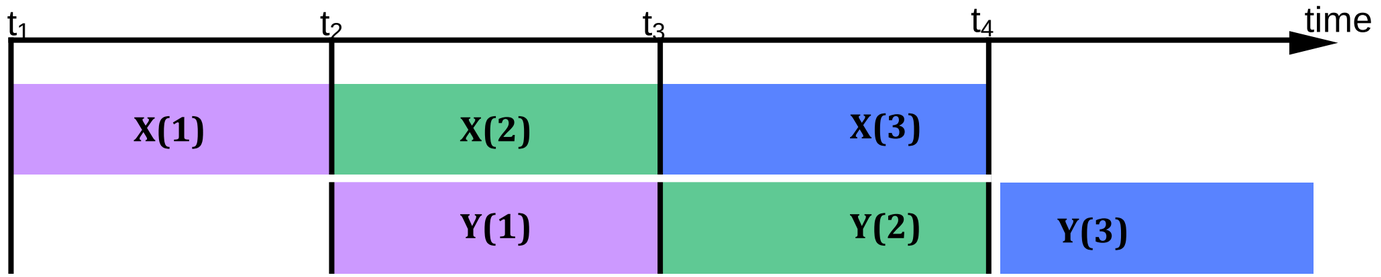
ARCHITECTURAL TECHNIQUES : PIPELINE DEPTH

- Pipeline depth: 1 (One Pipeline register Added)
  - Critical path: 2 Adders



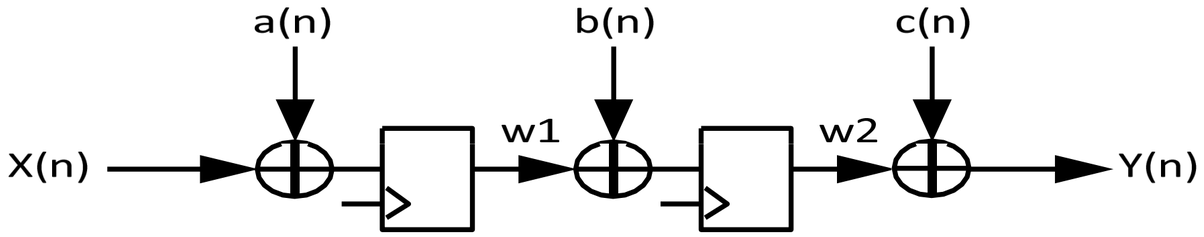

```
wire w1;
reg  w2;
assign w1 = X + a;

always @(posedge Clk)
    w2 <= w1 + b;
```

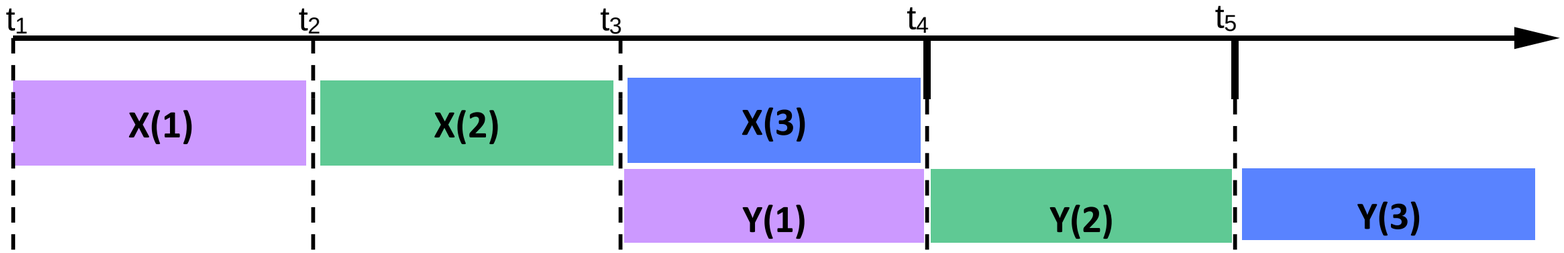


ARCHITECTURAL TECHNIQUES : PIPELINE DEPTH

- Pipeline depth: 2 (One Pipeline register Added)
  - Critical path: 1 Adder




```
reg  w1, w2;
assign Y = w2 + c;
always @(posedge Clk)
  begin
     w1 <= X + a;
     w2 <= w1 + b;
  end
```

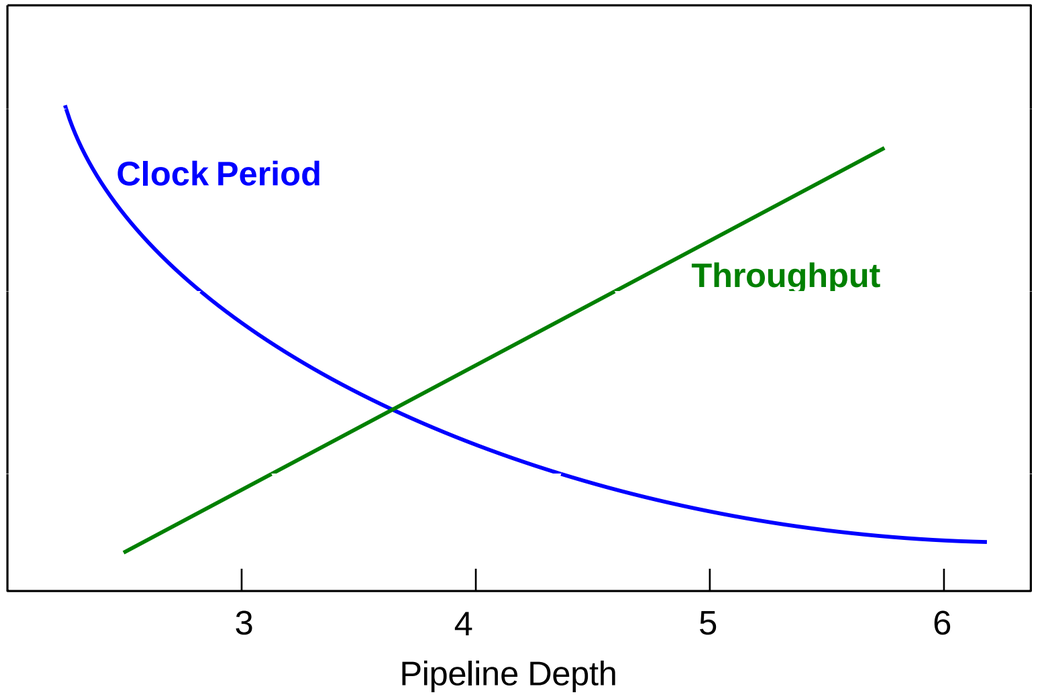


ARCHITECTURAL TECHNIQUES :PIPELINING

- Clock period and throughput as a function of pipeline depth:

- Clock period : $\tau_{\text{Clk}} \propto \frac{1}{n}$
- Throughput: $\mathrm{T} \propto n$

Adding register layers improves timing by dividing the critical path into two paths of smaller delay



Architectural Techniques : Pipelining

General Rule:

- Pipelining latches can only be placed across **feed-forward cutsets** of the circuit.

Cut set:

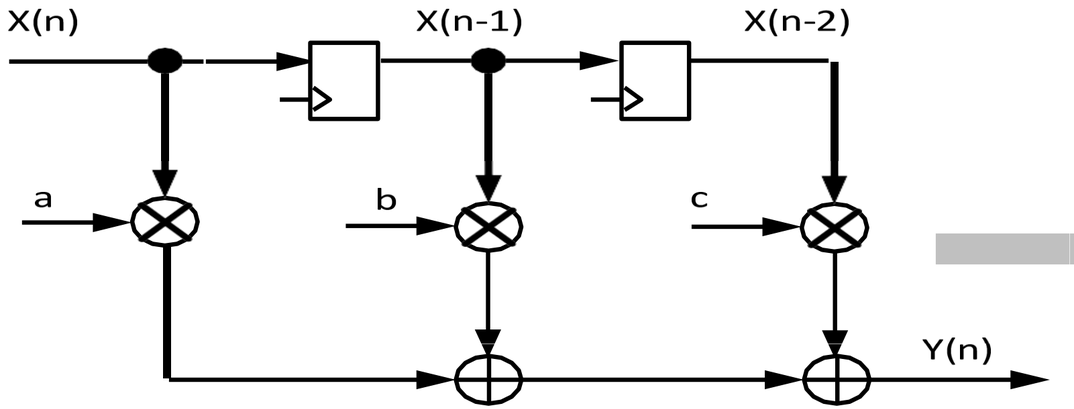
- A set of paths of a circuit such that if these paths are removed, the circuit becomes disjoint (i.e., two separate pieces)

Feed-Forward Cutset:

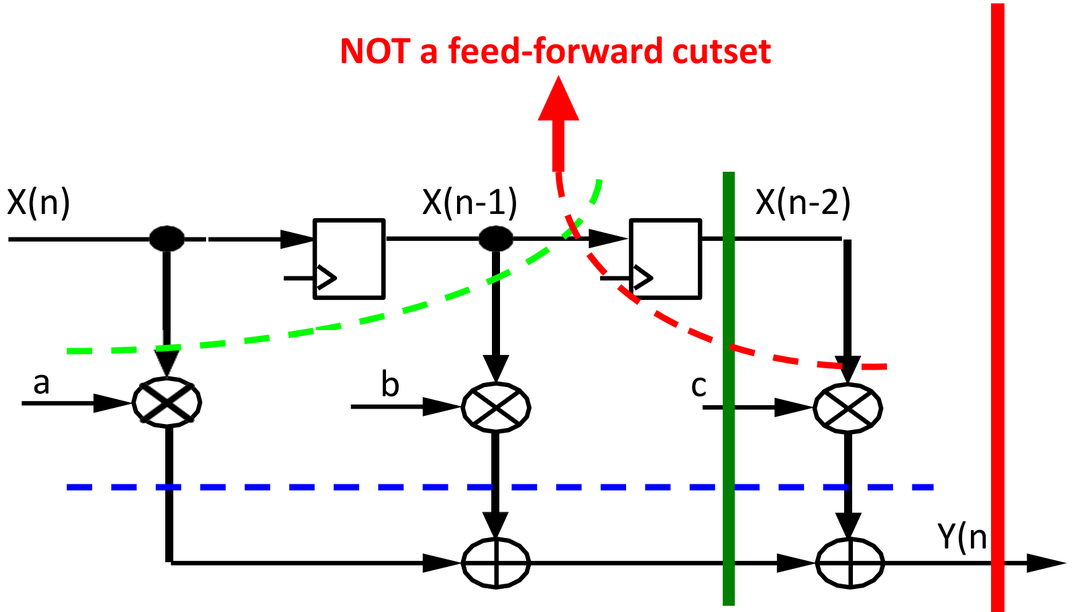
- A cutset is called feed-forward cutset if the data move in the forward direction on all the paths of the cutset

ARCHITECTURAL TECHNIQUES : PIPELINING

- **Example:**
  - FIR Filter
  - Three feed-forward cutsets ar



$Y(n) = ax(n) + bx(n-1) + cx(n-2)$



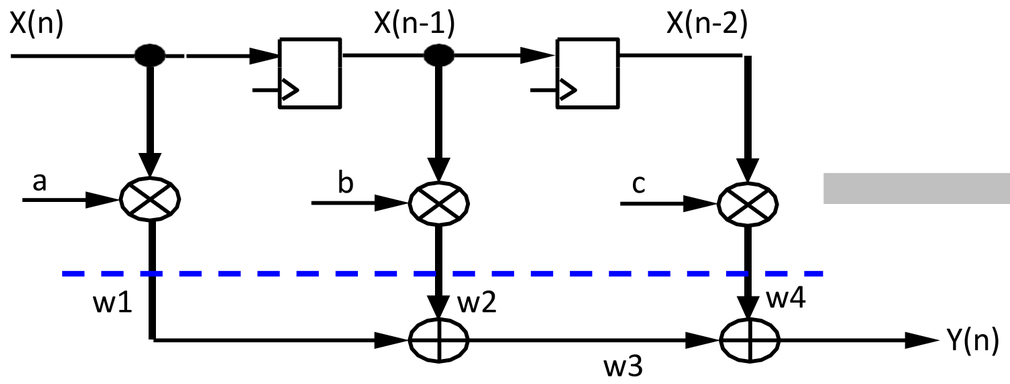
CLASS ROOM ACTIVITY

Group Discussion & Debate

15

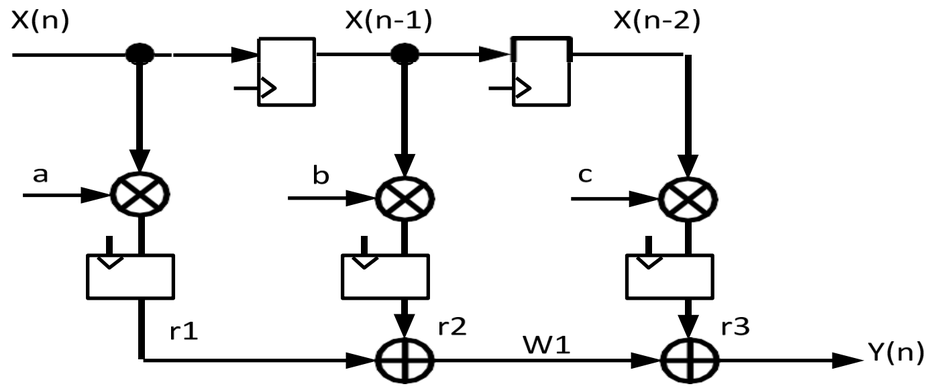
ARCHITECTURAL TECHNIQUES : PIPELINING

Critical Path: 1M+2A



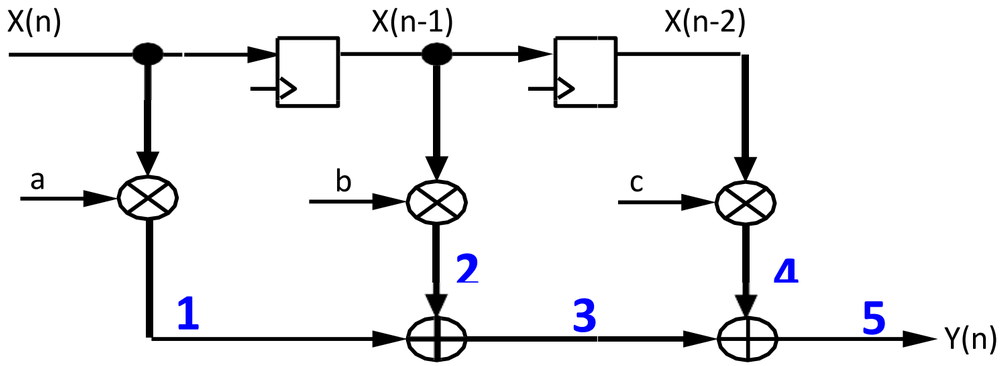

```
assign w1 = a*Xn;  assign
w2 = b*Xn_1 ;  assign w3
= w1 + w2;  assign w4 =
c*Xn_2;  assign Y = w3 +
w4;  always @(posedge
Clk)  begin
     Xn_1 <= Xn;
     Xn_2 <= Xn_1;
  end
```

Critical Path: 2A




```
assign Y = r3 + w1;  assign
w1 = r1 + r2;  always
@(posedge Clk)  begin
     Xn_1 <= Xn;
     Xn_2 <= Xn_1;  r1
     <= a*Xn;
     r2 <= b*Xn_1;  r3
     <= c*Xn_2;
  end
```

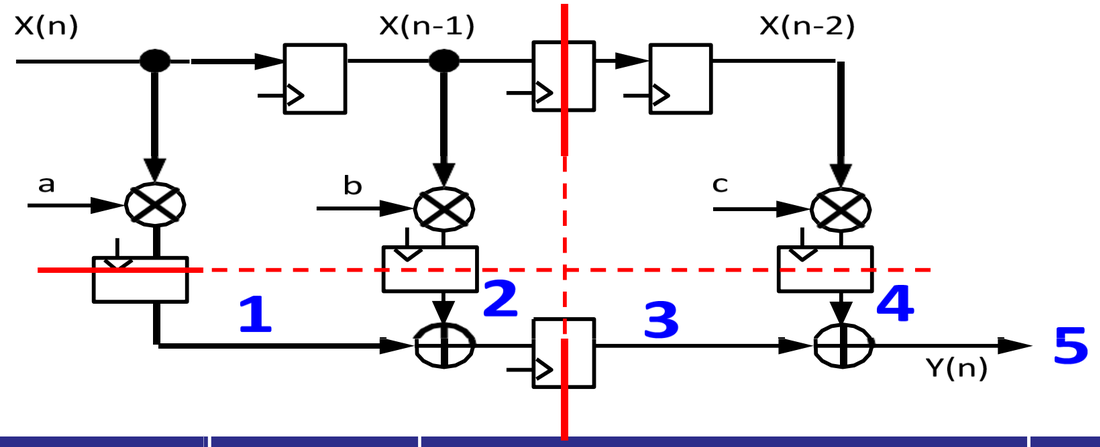
ARCHITECTURAL TECHNIQUES : PIPELINING



Cloc k	Input	1	2	3	4	5	Output
0	X(0)	aX(0)	-	aX(0)	-	aX(0)	Y(0)
1	X(1)	aX(1)	bX(0)	aX(1)+bX(0)	-	aX(1)+bX(0)	Y(1)
2	X(2)	aX(2)	bX(1)	aX(2)+bX(1)	cX(0)	aX(2)+bX(1)+cX(0)	Y(2)
3	X(3)	aX(3)	bX(2)	aX(3)+bX(2)	cX(1)	aX(3)+bX(2)+cX(1)	Y(3)

ARCHITECTURAL TECHNIQUES : PIPELINING

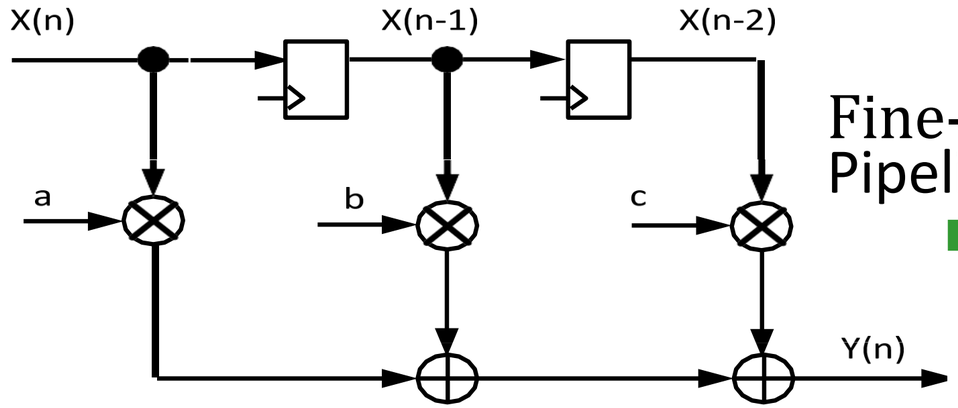
Even more pipelining



Clock	Input	1	2	3	4	5	Output
0	X(0)	-	-	-	-	-	-
1	X(1)	aX(0)	-	-	-	-	-
2	X(2)	aX(1)	bX(0)	aX(0)	-	aX(0)	Y(0)
3	X(3)	aX(2)	bX(1)	aX(1)+bX(0)	-	aX(1)+bX(0)	Y(1)
4	X(3)	aX(2)	bX(1)	aX(2)+bX(1)	cX(0)	aX(2)+bX(1)+cX(0)	Y(2)

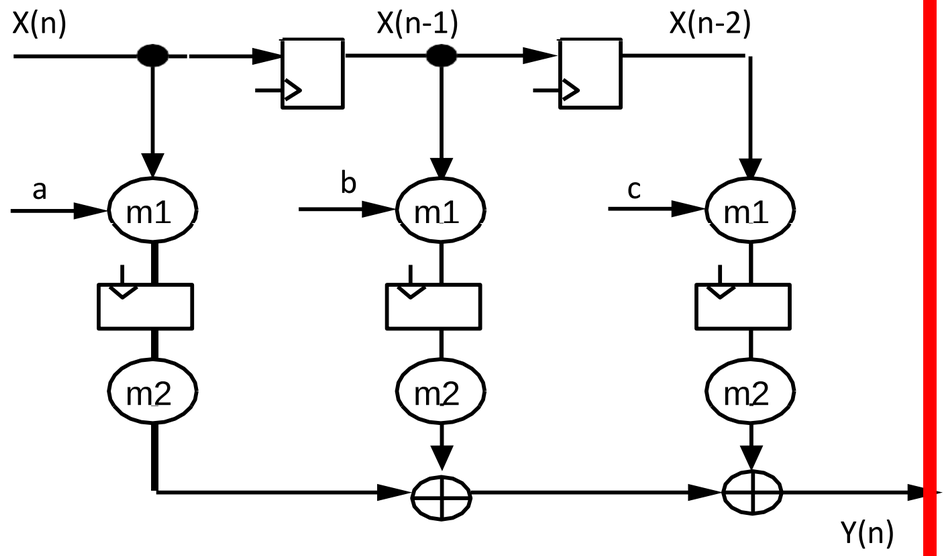
ARCHITECTURAL TECHNIQUES : FINE-GRAIN PIPELINING

- Pipelining at the operation level
  - Break the multiplier into two parts



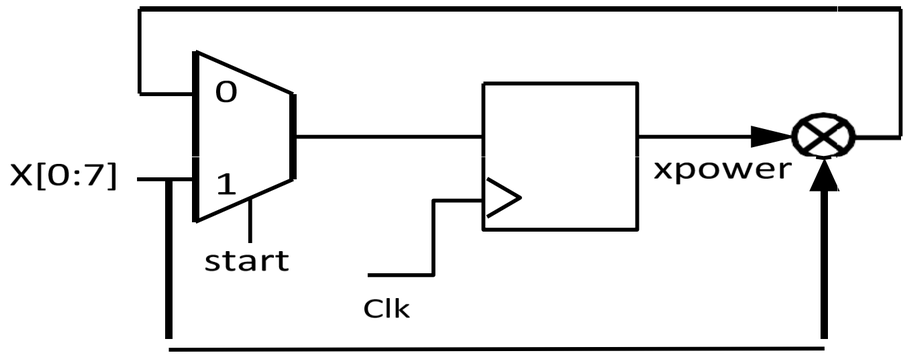
Y(n) = ax(n) + bx(n-1) + cx(n-2)

Fine-Grain Pipelining



UNROLLING THE LOOP USING PIPELINING

- Calculation of X^3
  - Throughput = 8/3. or 2.7
- Timing = One multiplier in the critical path
- Iterative implementation:
- No new computations can begin until the previous computation has completed



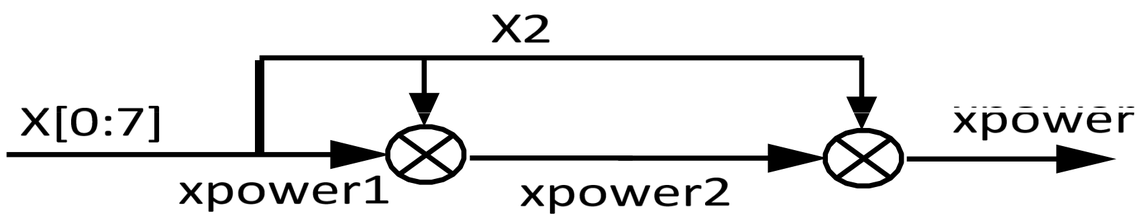

```
module power3( output reg [7:0] X3, output finished,
input [7:0] X, input clk, start); reg [7:0] ncount;
reg [7:0] Xpower, Xin;
assign finished = (ncount == 0);
        always@(posedge clk)
        if (start) begin XPower <= X;
        Xin<=X; ncount <= 2;
        X3 <= XPower; end
        else if(!finished) begin ncount <= ncount - 1; XPower <= XPower * Xin;
        End
endmodule
```

REMOVING PIPELINE REGISTERS (TO IMPROVE LATENCY)

Calculation of X^3

- Throughput = 8 bits/clock (3X improvement)
- Latency = 0
- ➢Timing = Two multipliers in the critical path

Latency can be reduced by removing pipeline registers

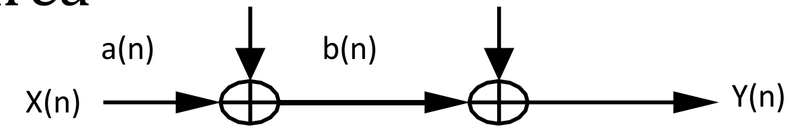



```
module power3(
Output [7:0] XPower,
input [7:0] X);
reg [7:0] XPower1, XPower2;  reg
[7:0] X1, X2;
always @*
   XPower1 = X;
always @(*)
begin
   X2 = XPower1;
   XPower2 = XPower1*XPower1;
end
assign XPower = XPower2 * X2;
endmodule
```

ARCHITECTURAL TECHNIQUES : PARALLEL PROCESSING

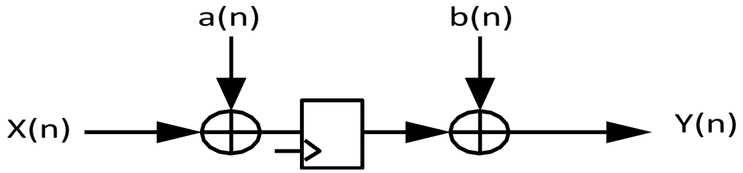
In parallel processing the same ha dware is duplicated to

- Increases the throughput without changing the critical path
- Increases the silicon area



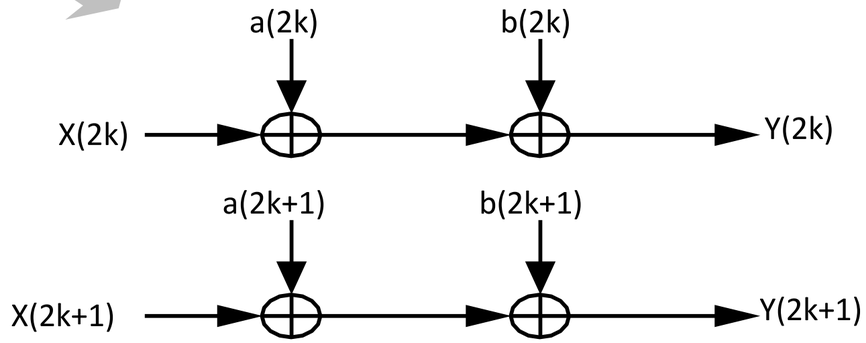
Clock Freq: f
Throughput: M samples

Pipelining



Clock Freq: 2f
Throughput: 2M samples

Parallel Processing



Clock Freq: f
Throughput: 2M samples

ADVANTAGEOUS

- Reduction in the critical path
- Higher throughput (number of computed results in a give time)
- Increases the clock speed (or sampling speed)
- Reduces the power consumption at same speed

ASSESSMENT

- Define critical path
- List out the needs of Pipeline
- Draw the FIR filter using pipeline processing
- Compare pipeline and parallel processing

SUMMARY & THANK YOU