Digital VLSI Architectures:

Pipelining & Parallel Processing

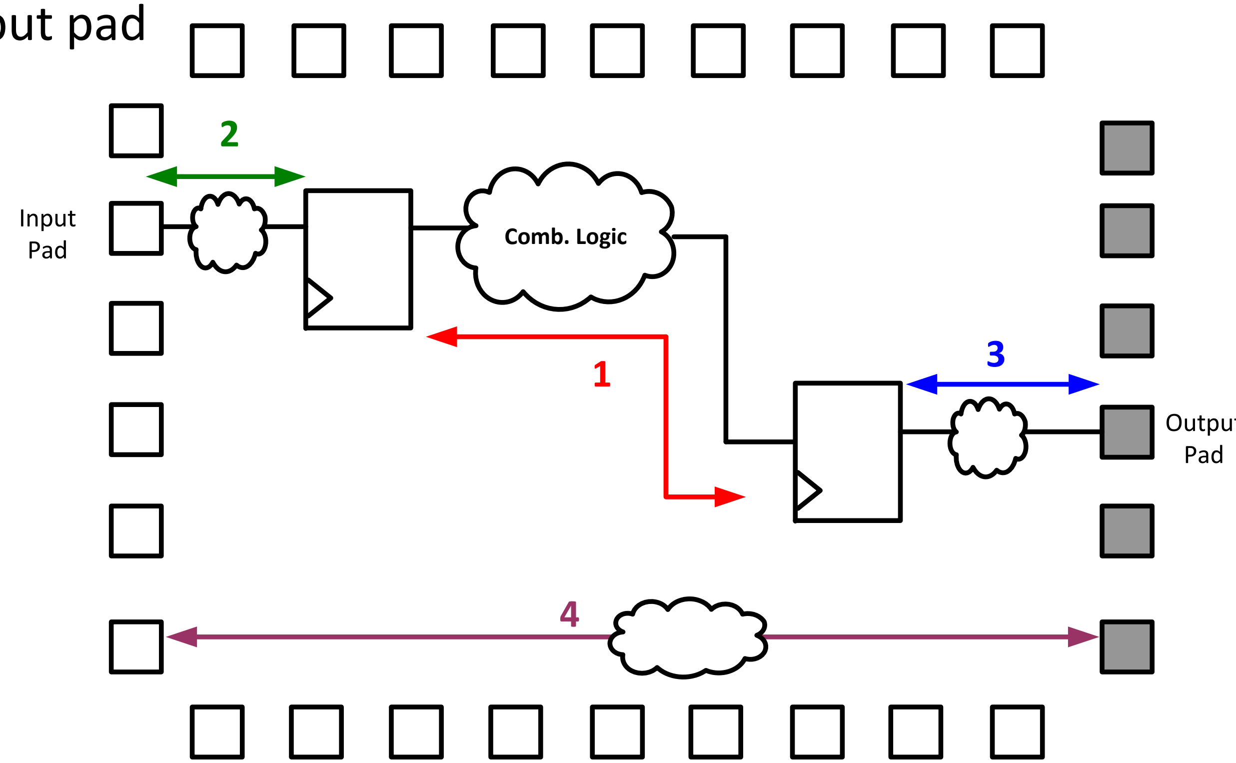
Mahdi Shabany
Sharif University of Technology

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)

The maximum delay between any two sequential elements in a design will determine the max clock speed



Digital Design Metrics

- Three primary physical characteristics of a digital design:
  - **Speed**
    - Throughput
    - Latency
    - Timing
  - **Area**
  - **Power**

Digital Design Metrics

- **Speed**
  - **Throughput** :
    - The amount of data that is processed per clock cycle (bits per second)
  - **Latency**
    - The time between data input and processed data output (clock cycle)
  - **Timing**
    - The logic delays between sequential elements (clock period)
    - When a design does not meet the timing it means the delay of the critical path is greater than the target clock period

Maximum Clock Frequency: Critical Path

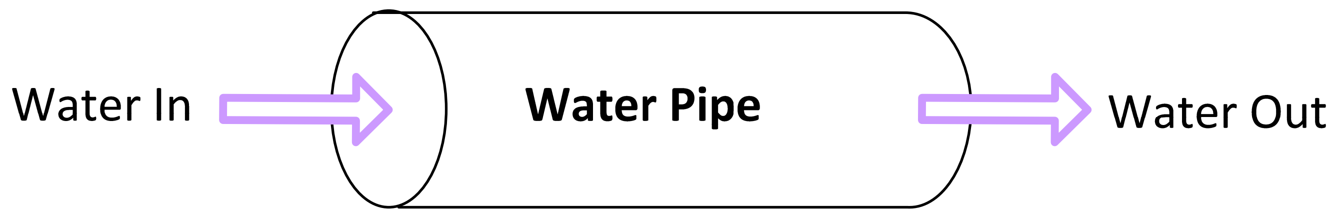
- Maximum Clock Frequency:

$$F_{max} = \frac{1}{T_{clk-q} + T_{logic} + T_{setup} + T_{routing} - T_{skew}}$$

- T_{clk-q} : time from clock arrival until data arrives at Q
- T_{logic} : propagation delay through logic between flip-flops
- $T_{routing}$: routing delay between flip-flops
- T_{setup} : minimum time data must arrive at D before the next rising edge of clock
- T_{skew} : propagation delay of clock between the launch flip-flop and the capture flip-flop.

Pipelining (to Improve Throughput)

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



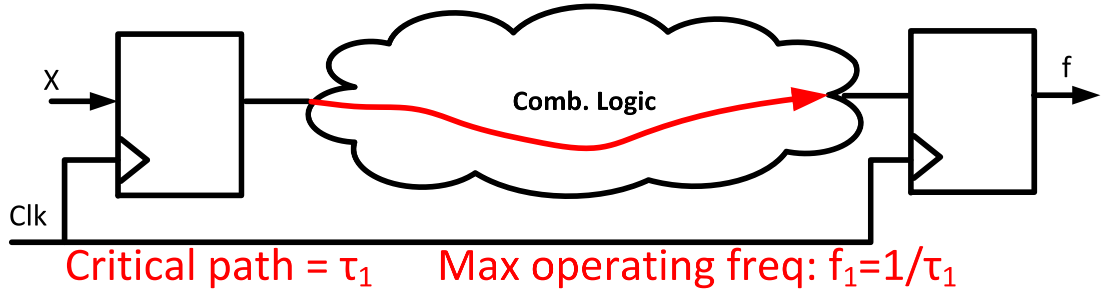
- **Advantageous**:
  - Reduction in the critical path
  - Higher throughput (number of computed results in a give time)
  - Increases the clock speed (or sampling speed)
  - Reduces the power consumption at same speed

Architectural Techniques :Pipelining

- **Pipelining:**
  - Very similar to the assembly line in the auto industry
  - The beauty of a pipelined design is that new data can begin processing before the prior data has finished, much like cars are processed on an assembly line.

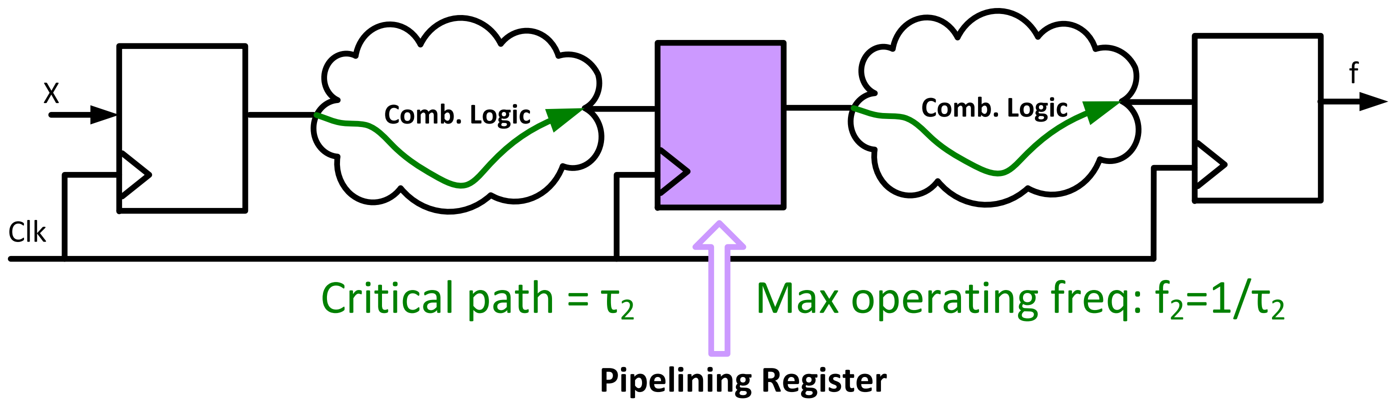
Architectural Techniques :Pipelining

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



2 cycles later

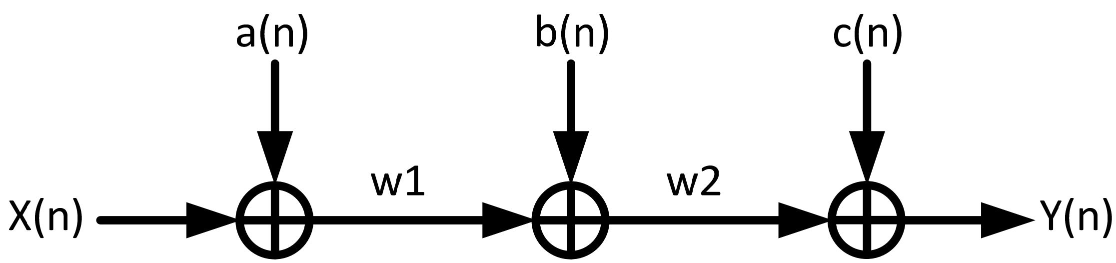
- **Pipelined version: (**Critical path = τ_2 Max operating freq: $f_2=1/\tau_2$**)**
  - Smaller Critical Path → higher throughput ($\tau_2<\tau_1$ → $f_2>f_1$)
  - Longer latency



3 cycles later

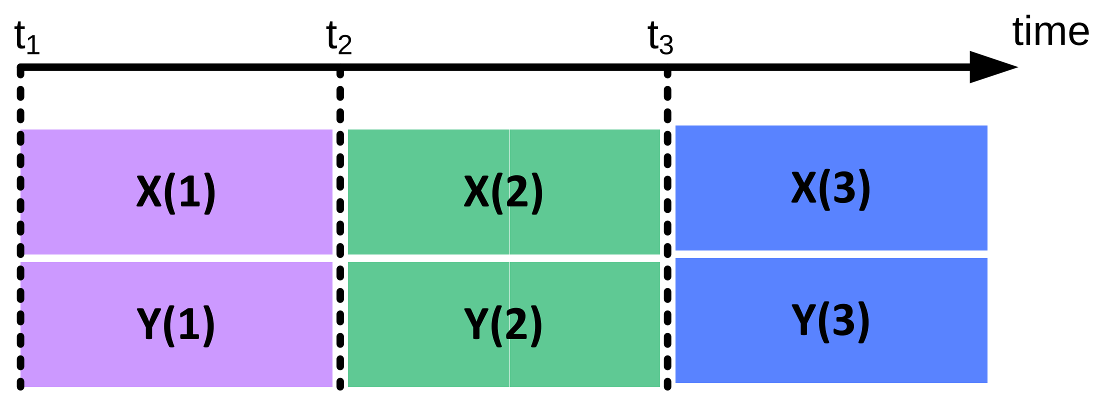
Architectural Techniques : Pipeline depth

- Pipeline depth: 0 (No Pipeline)
  - Critical path: 3 Adders



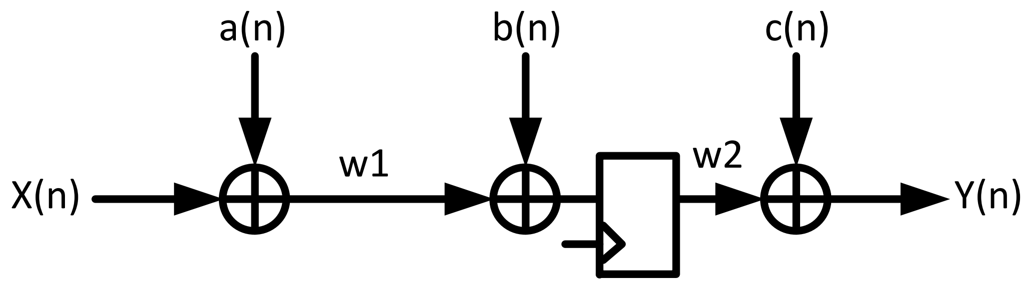

```
wire  w1, w2;
assign  w1 = X + a;
assign  w2 = w1 + b;
assign  Y = w2 + c;
```

- Latency : 0



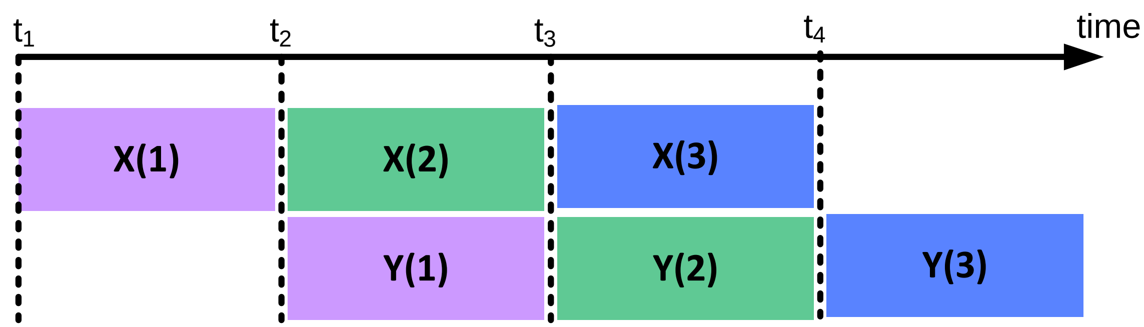
Architectural Techniques : Pipeline depth

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



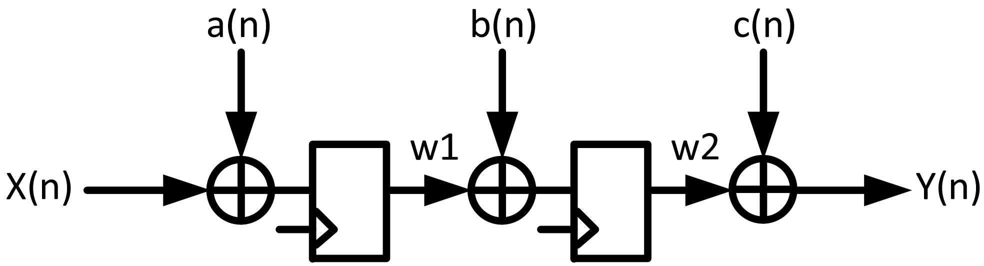

```
wire  w1;
reg   w2;
assign  w1 = X + a;
assign  Y = w2 + c;

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

- Latency : 1



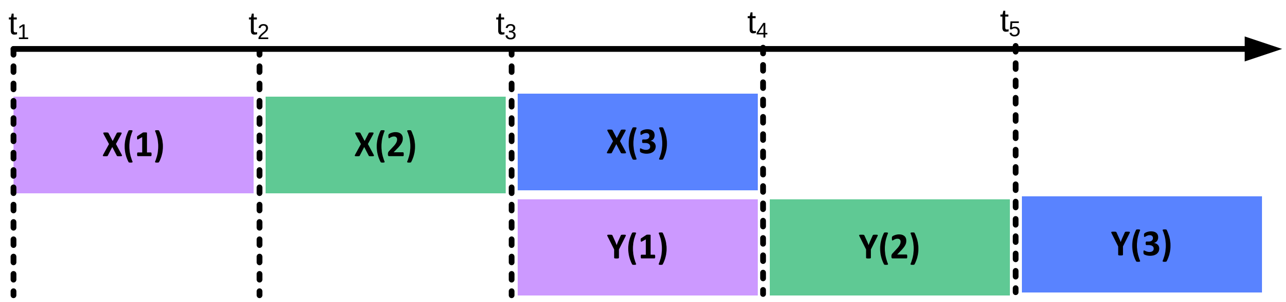
Architectural Techniques : Pipeline depth

- 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
```

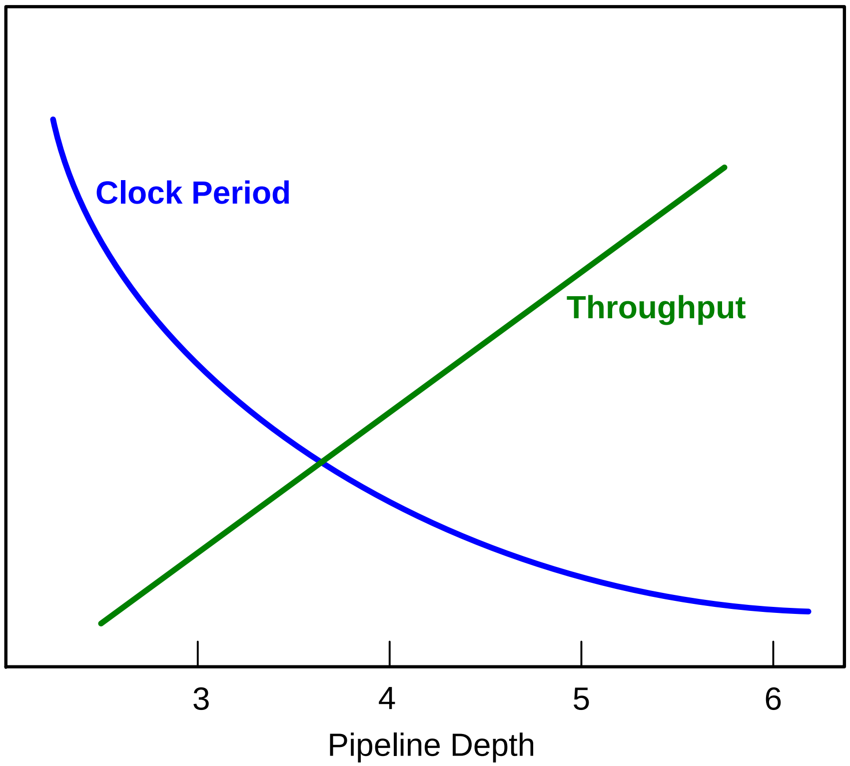
- Latency : 2



Architectural Techniques :Pipelining

- Clock period and throughput as a function of pipeline depth:
  - Clock period : $\tau_{\text{Clk}} \propto \frac{1}{n}$
  - Throughput: $\text{T} \propto n$

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.

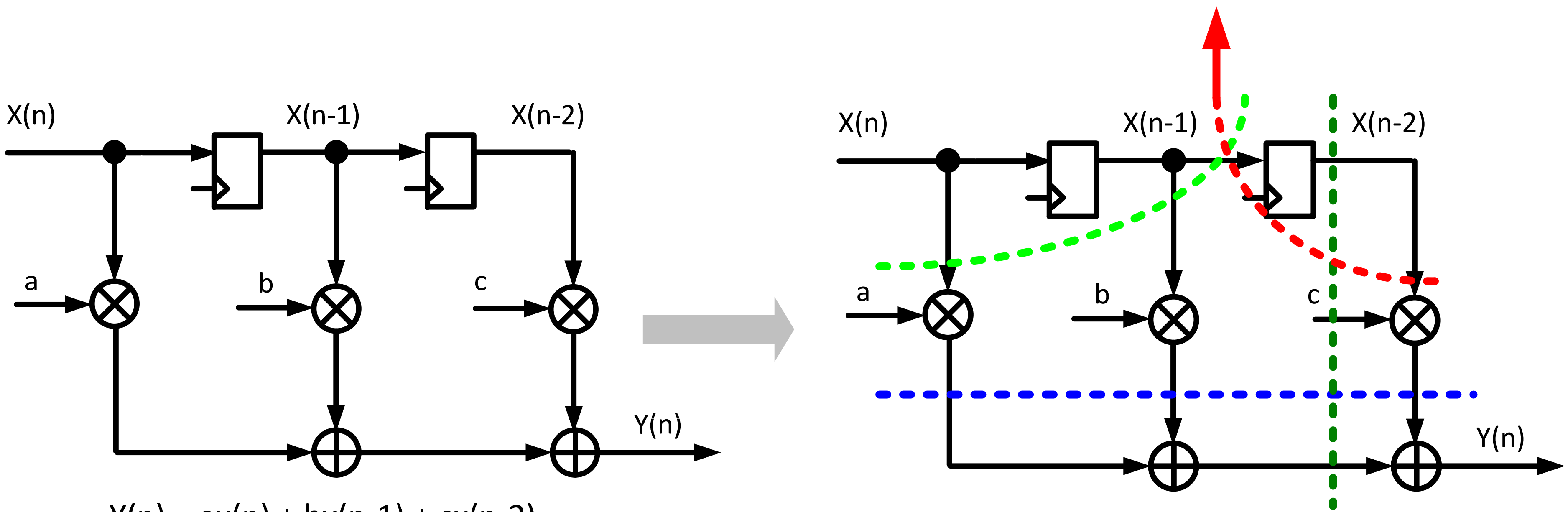
- **Cutset:**
  - A set of paths of a circuit such that if these paths are removed, the circuit becomes disjoint (i.e., two separate pieces)

- **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:**

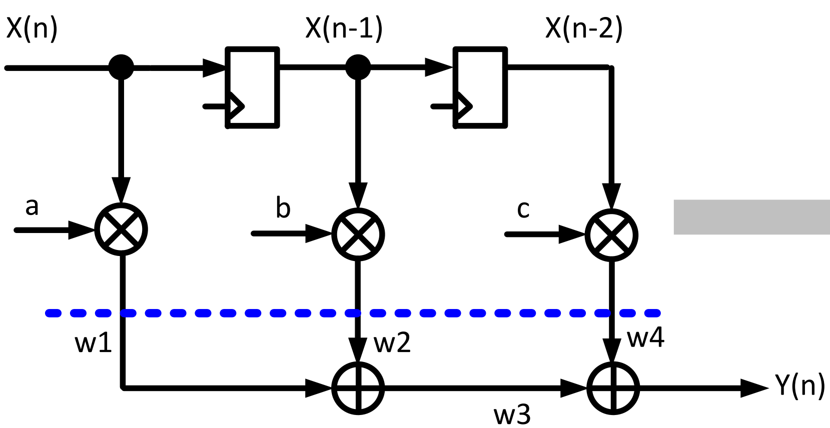
- FIR Filter
- Three feed-forward cutsets are shown



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

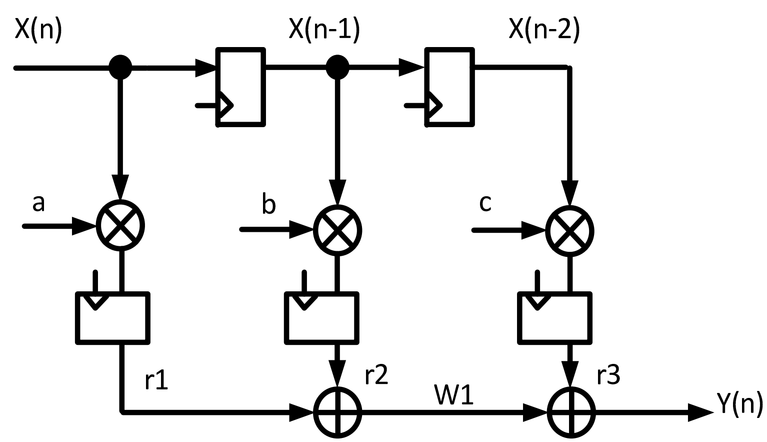
Architectural Techniques : Pipelining

Critical Path: 1M+2A



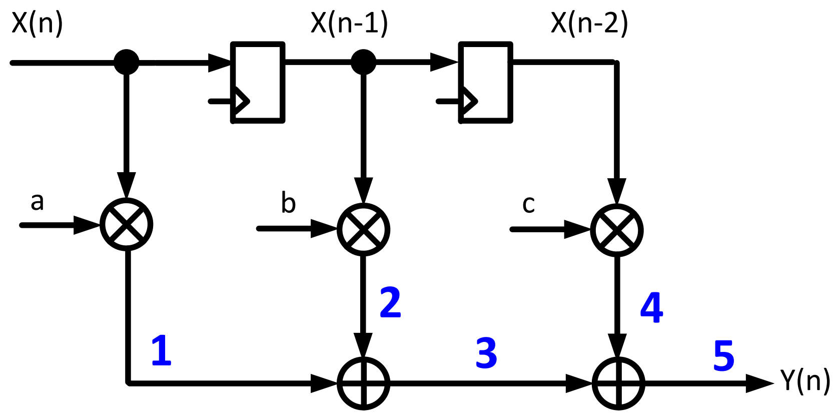

```
assign  w1 = a*Xn;
assign  w2 = b*Xn_1 ;
assign  w3 = w1 + w2;
assign  w4 = c*Xn_2;
assign  Y = w3 + w4;
always @(posedge Clk)
  begin
     Xn_1 <= Xn;
     Xn_2 <= Xn_1;
  end
```

Critical Path: 2A



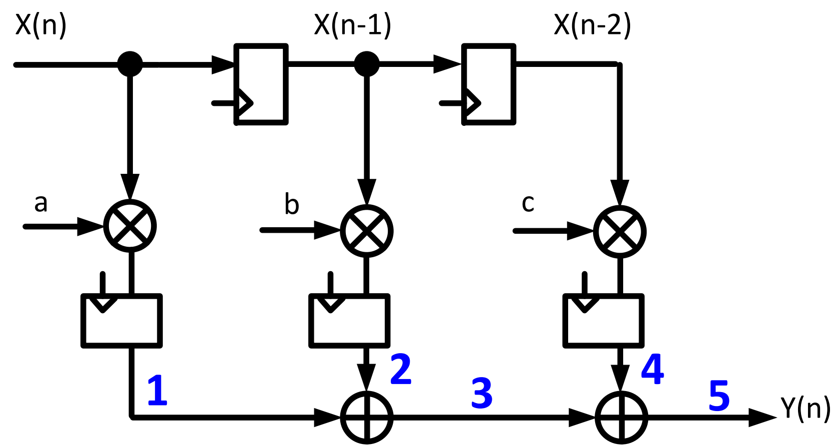

```
assign  Y = r3 + w1;
assign  w1 = r1 + r2;
always @(posedge Clk)
  begin
     Xn_1 <= Xn;
     Xn_2 <= Xn_1;
     r1 <= a*Xn;
     r2 <= b*Xn_1;
     r3 <= c*Xn_2;
  end
```

Architectural Techniques : Pipelining



Clock	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)

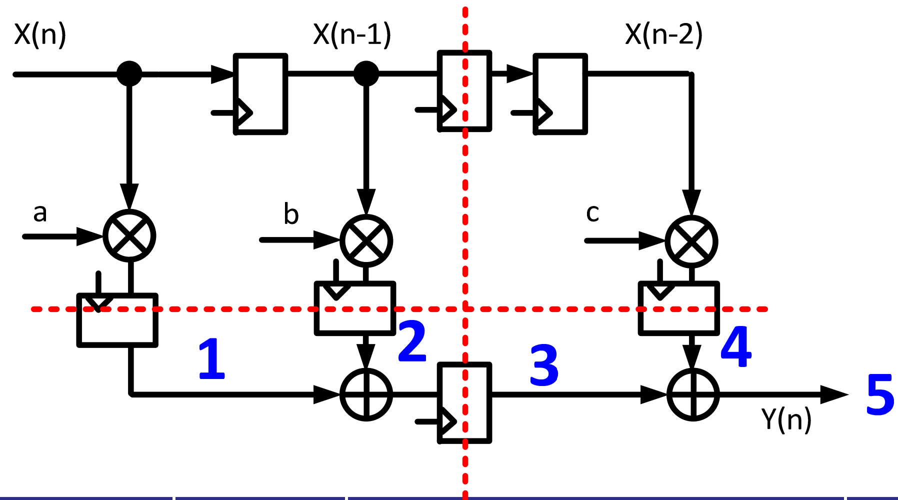
Architectural Techniques : Pipelining



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

Architectural Techniques : Pipelining

- Even more pipelining

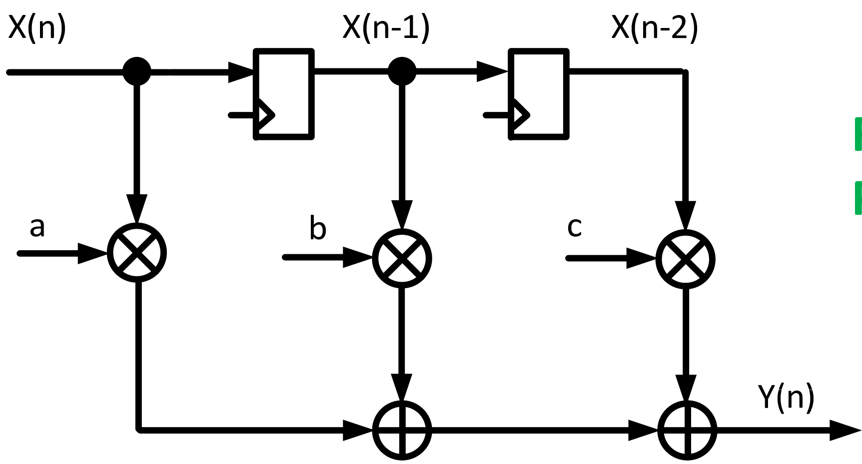


Clock	Input	1	2	3	4	5	Output
0	X(0)	-	-	-	-	-	-
1	X(1)	aX(0)	-	-	-	-	-
2	X(2)	aX(1)	bX(0)	aX(0)	-	aX(0)	Y(0)
3	X(3)	aX(2)	bX(1)	aX(1)+bX(0)	-	aX(1)+bX(0)	Y(1)
4	X(3)	aX(2)	bX(1)	aX(2)+bX(1)	cX(0)	aX(2)+bX(1)+cX(0)	Y(2)

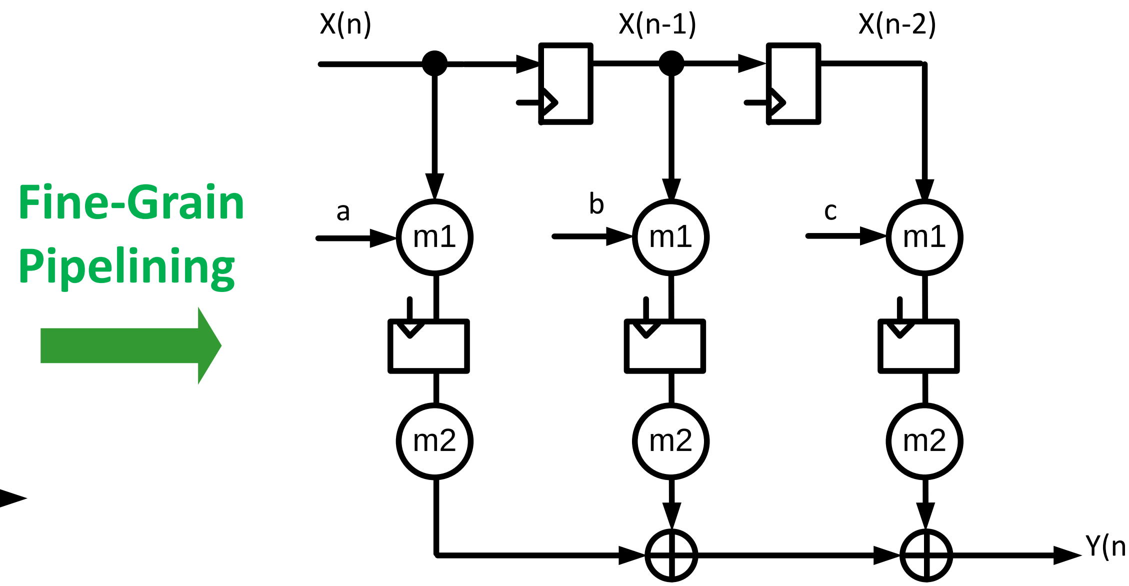
Architectural Techniques : Fine-Grain Pipelining

❖ Pipelining at the operation level

   ➢ Break the multiplier into two parts



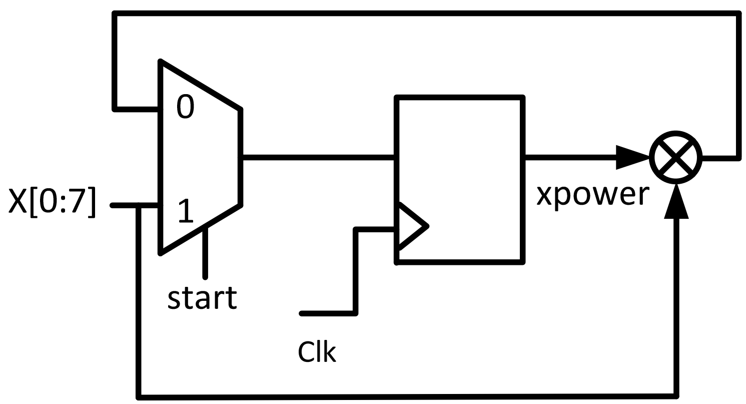
$Y(n) = ax(n) + bx(n-1) + cx(n-2)$



Unrolling the Loop Using Pipelining

- Calculation of X^3
  - Throughput = 8/3, or 2.7 bits/clock
  - Latency = 3 clocks
  - Timing = One multiplier in the critical path

- Iterative implementation:
- No new computations can begin until the previous computation has completed

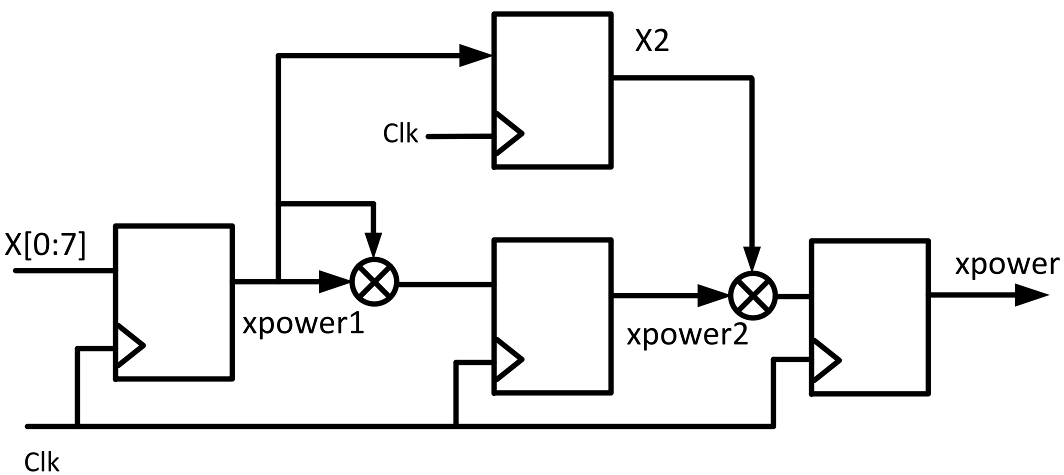



```
module power3(
output reg [7:0] X3,
output finished,
input [7:0] X,
input clk, start);
reg [7:0] ncount;
reg [7:0] Xpower, Xin;
assign finished = (ncount == 0);
        always@(posedge clk)
        if (start) begin
        XPower <= X; Xin<=X;
        ncount <= 2;
        X3 <= XPower;
        end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

Unrolling the Loop Using Pipelining

- Calculation of X^3
  - Throughput = 8/1, or 8 bits/clock (3X improvement)
  - Latency = 3 clocks
  - Timing = One multiplier in the critical path
- **Penalty**: More Area

Unrolling an algorithm with n iterative loops increases throughput by a factor of n



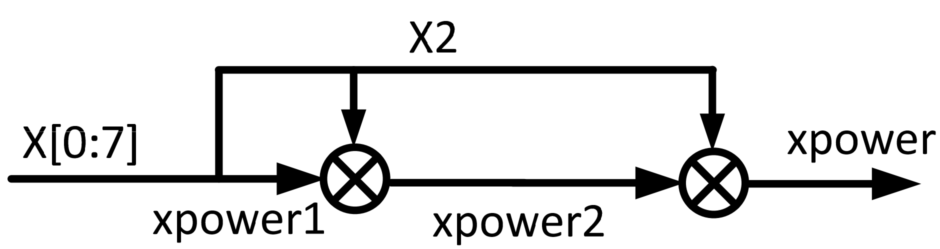

```
module power3(
output reg [7:0] XPower,
input clk,
input [7:0] X);
reg [7:0] XPower1, XPower2;
reg [7:0] X2;
always @(posedge clk) begin
// Pipeline stage 1
XPower1 <= X;
// Pipeline stage 2
XPower2 <= XPower1 * XPower1 ;
X2 <= XPower1 ;
// Pipeline stage 3
XPower <= XPower2 * X2;
end
endmodule
```

Removing Pipeline Registers (to Improve Latency)

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

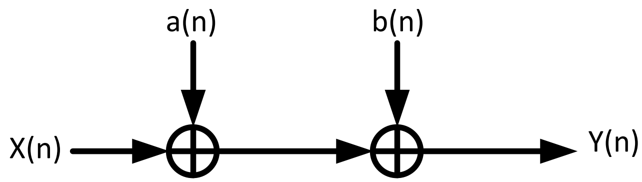
Latency can be reduced by removing pipeline registers

```
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
```



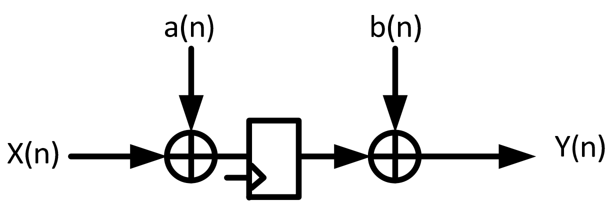
Architectural Techniques : Parallel Processing

- In parallel processing the same hardware is duplicated to
  - Increases the throughput without changing the critical path
  - Increases the silicon area



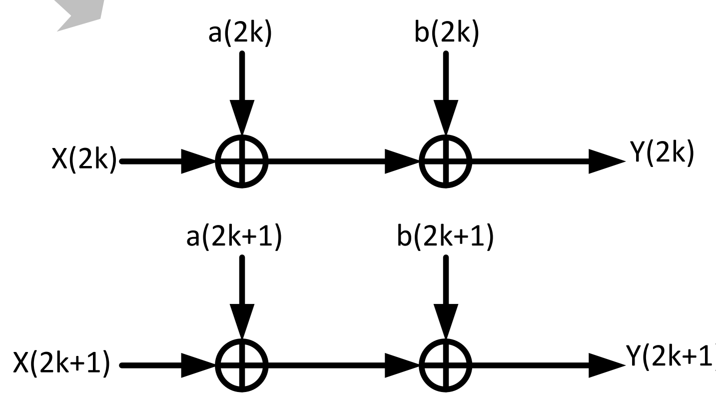
Clock Freq: f
Throughput: M samples

Pipelining



Clock Freq: 2f
Throughput: 2M samples

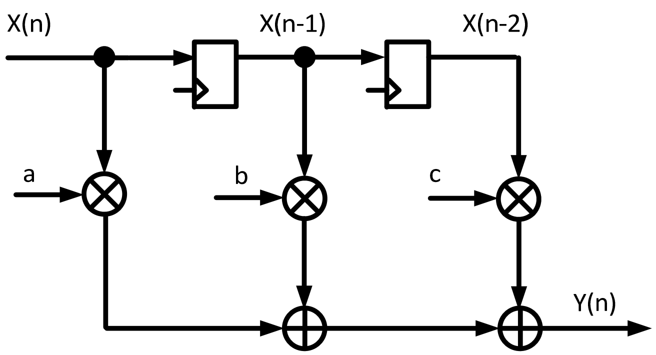
Parallel Processing



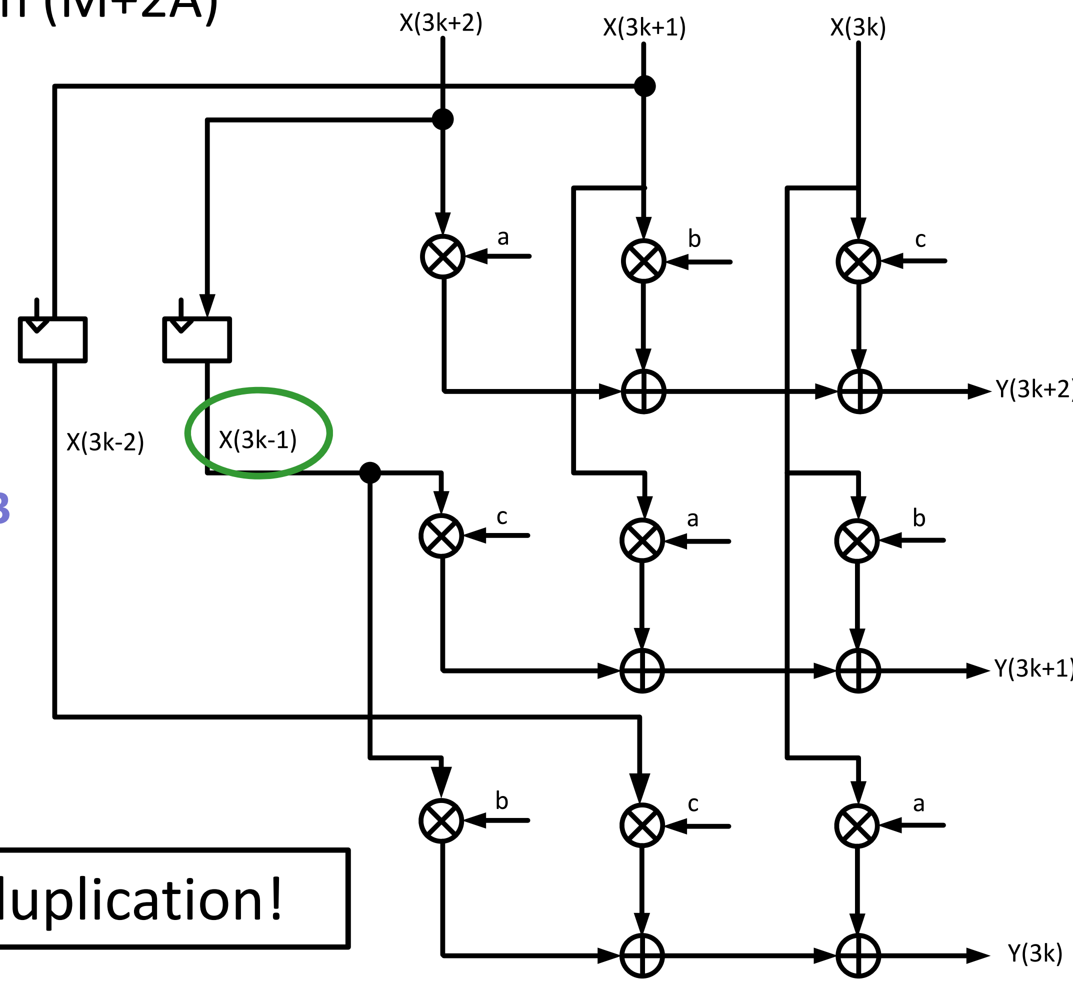
Clock Freq: f
Throughput: 2M samples

Architectural Techniques : Parallel Processing

- ❑ Parallel processing for a 3-tap FIR filter
  - ➢ Both have the same critical path (M+2A)



Parallel Factor:3



$$y(3k) = ax(3k) + bx(3k-1) + cx(3k-2)$$

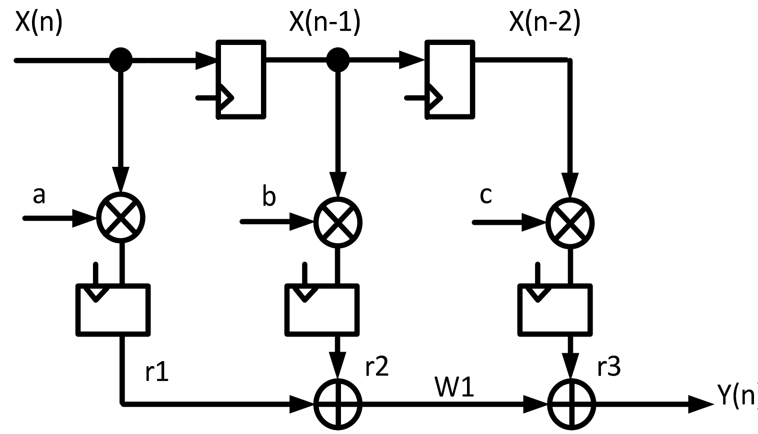
$$y(3k+1) = ax(3k+1) + bx(3k) + cx(3k-1)$$

$$y(3k+2) = ax(3k+2) + bx(3k+1) + cx(3k)$$

Not a simple duplication!

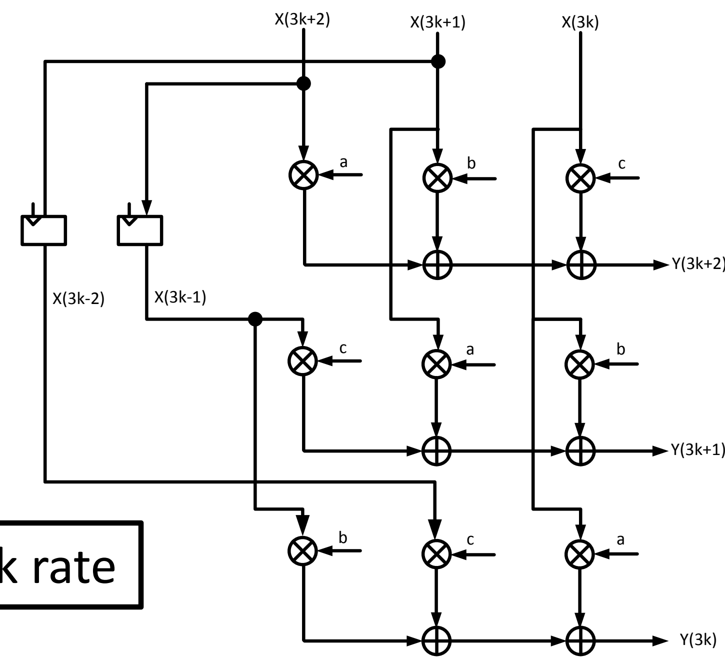
Sample Period vs. Clock Period

- Pipelined system: $T_{clk} = T_{sample}$



$$T_{sample} = T_{Clk} = T_M$$

- Parallel System: $T_{clk} \neq T_{sample}$

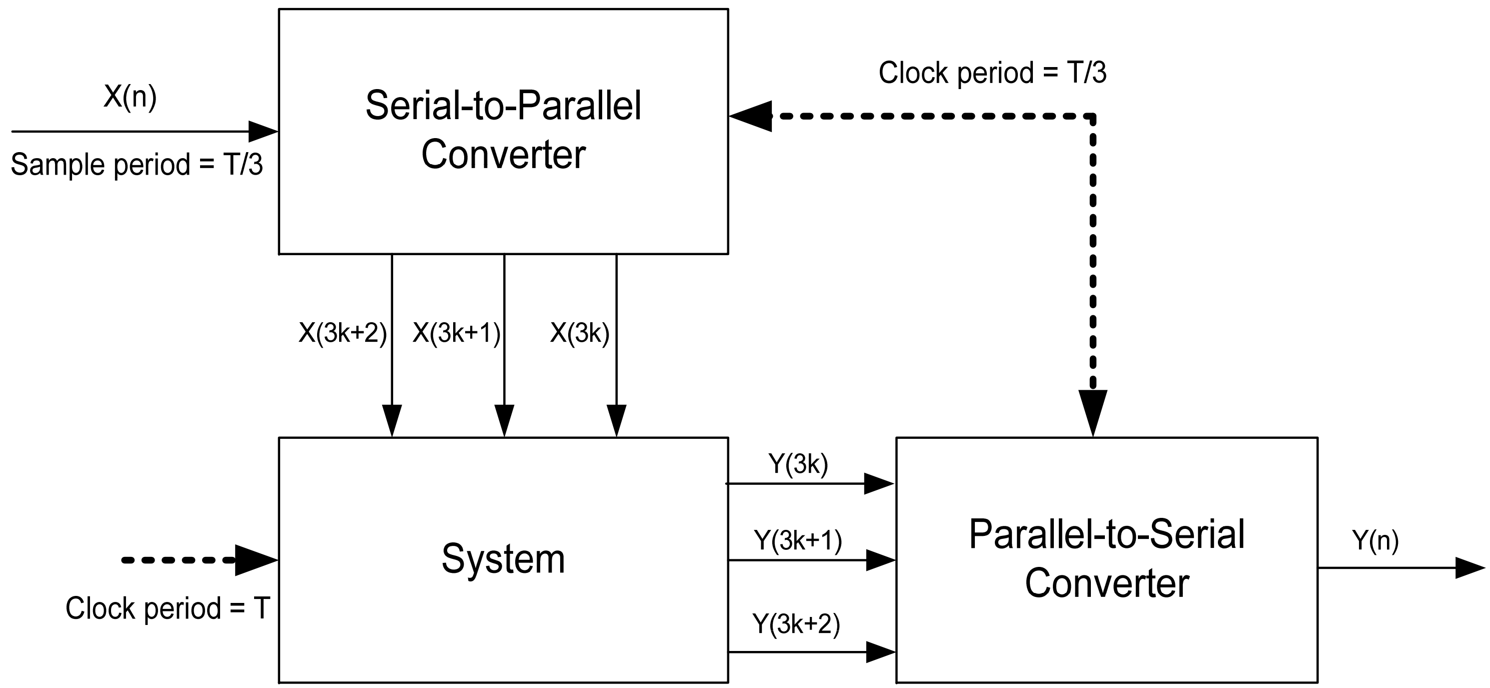


$$T_{sample} = \frac{1}{3} T_{Clk} = \frac{1}{3}(T_M + 2T_A)$$

Higher Sample rate than the clock rate

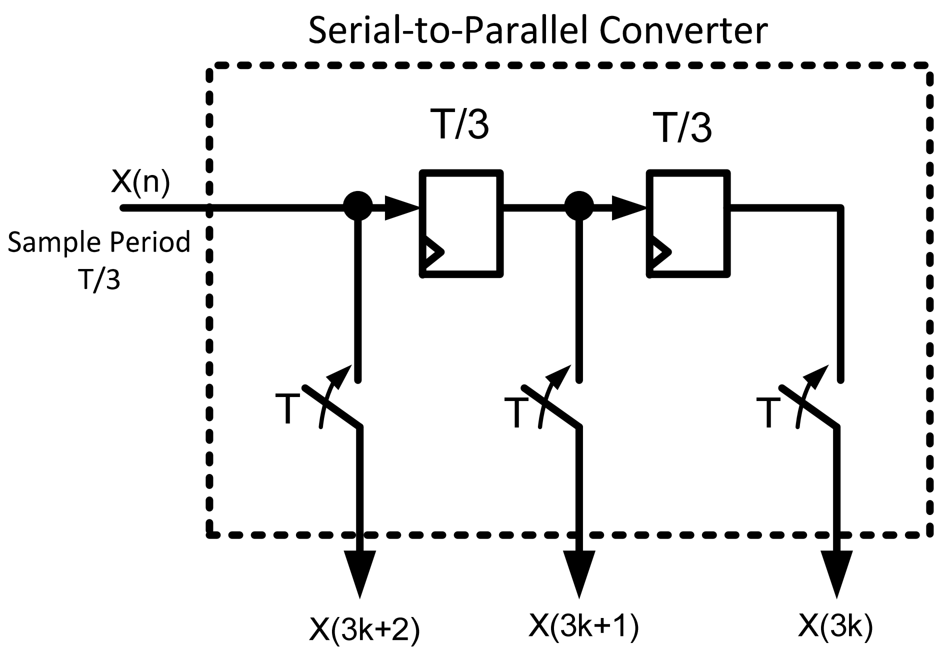
Complete Parallel System with S/P and P/S

- A Complete Parallel System:

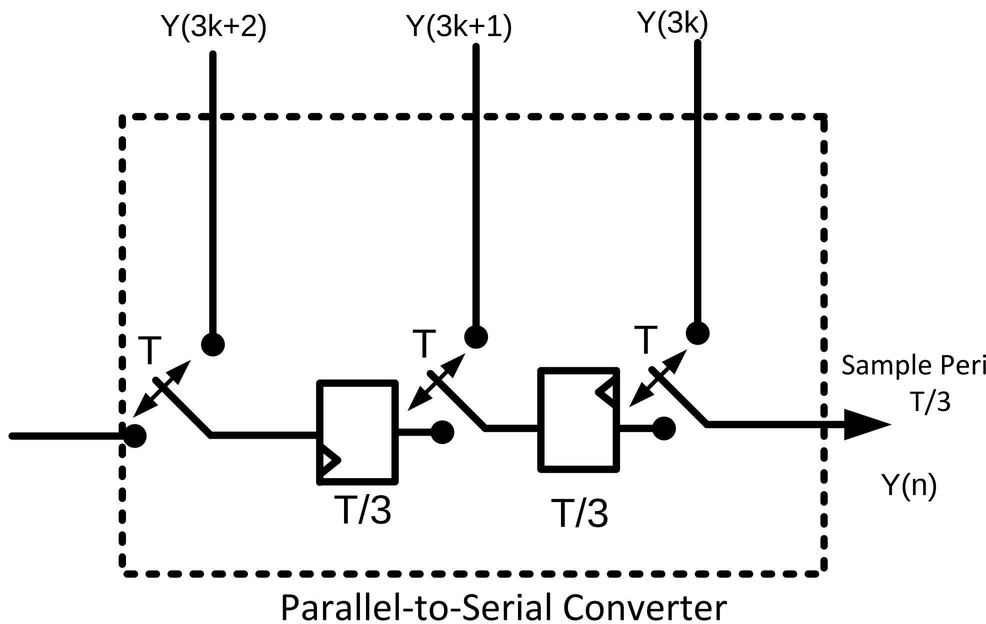


S/P and P/S Blocks

- S/P Converter:



- P/S Converter:



When Pipelining When Parallelism?

- Pipeline technique is used when the critical path is in the design (Number 1, 2, 3, 4)

- Parallelism is used when the critical path is bounded by the communication or I/O bound. (Number 5)
  - Pipelining does not help in this case!
  - a.k.a Communication Bounded