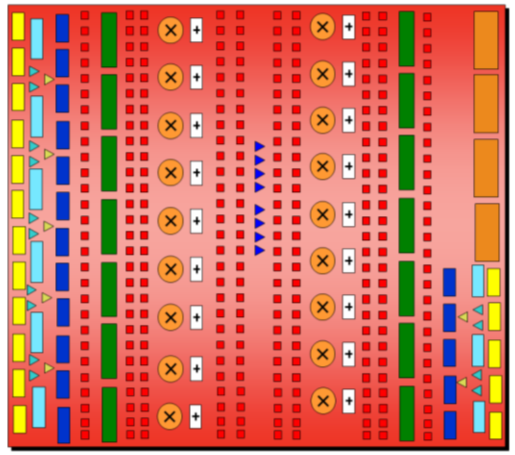
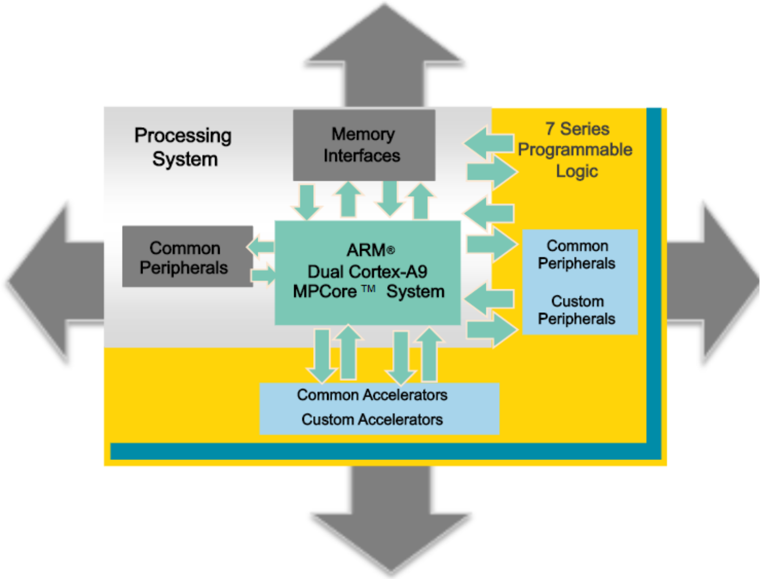
DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

A2A
Algorithms to Architecture

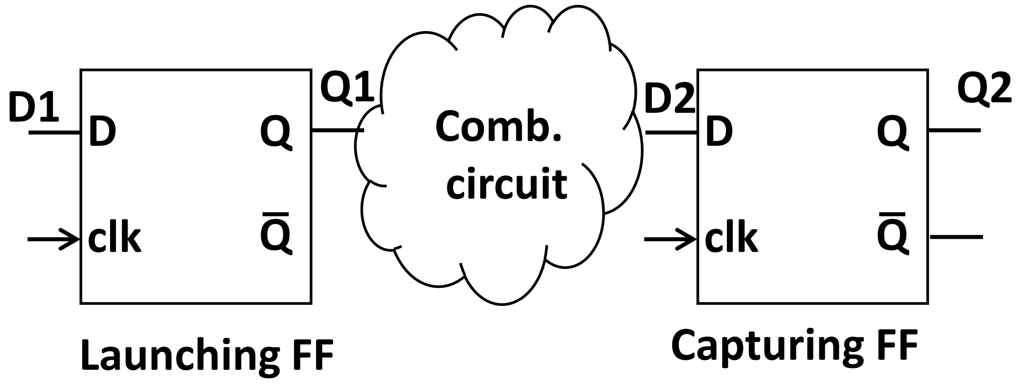
ECE 270: Embedded Logic Design





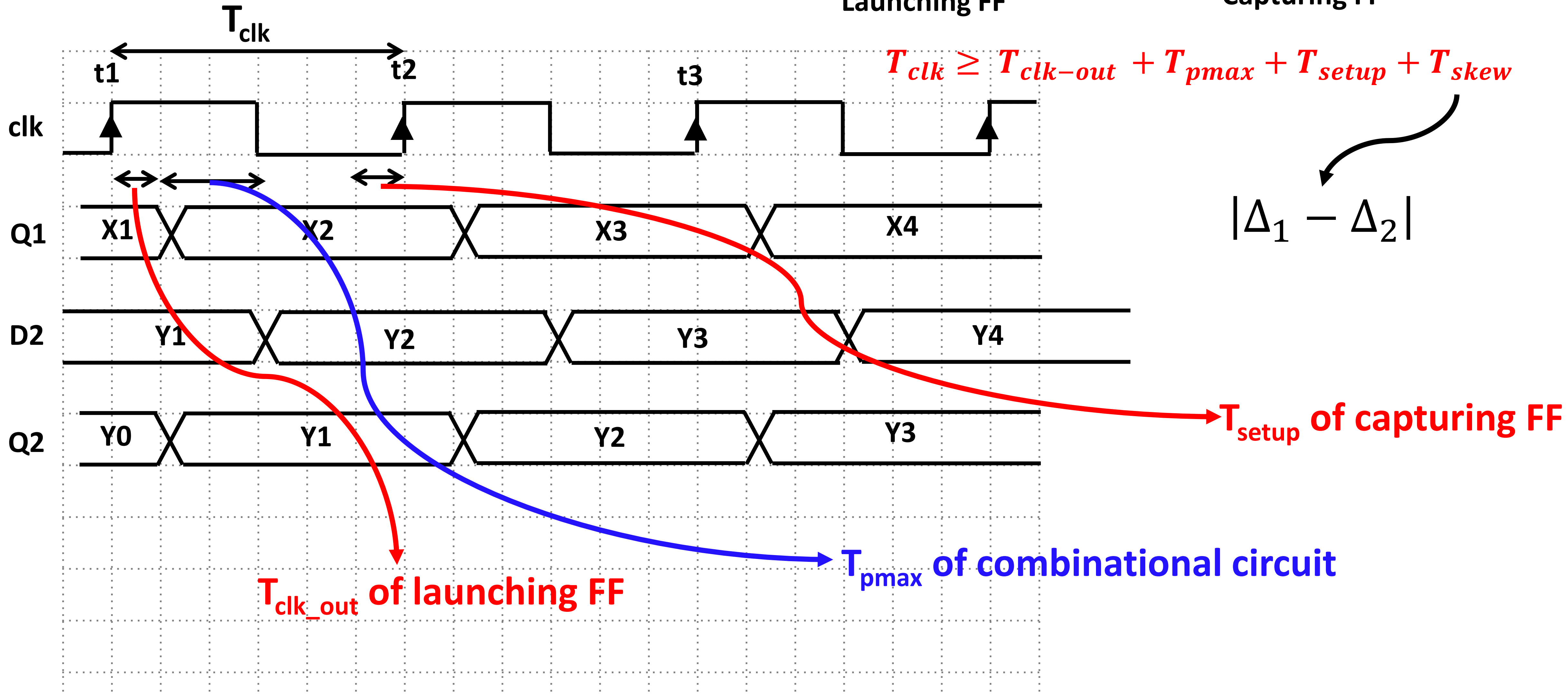
Pipelining

FF Timing Constraints



$$T_{clk} \geq T_{clk-out} + T_{pmax} + T_{setup} + T_{skew}$$

$$|\Delta_1 - \Delta_2|$$

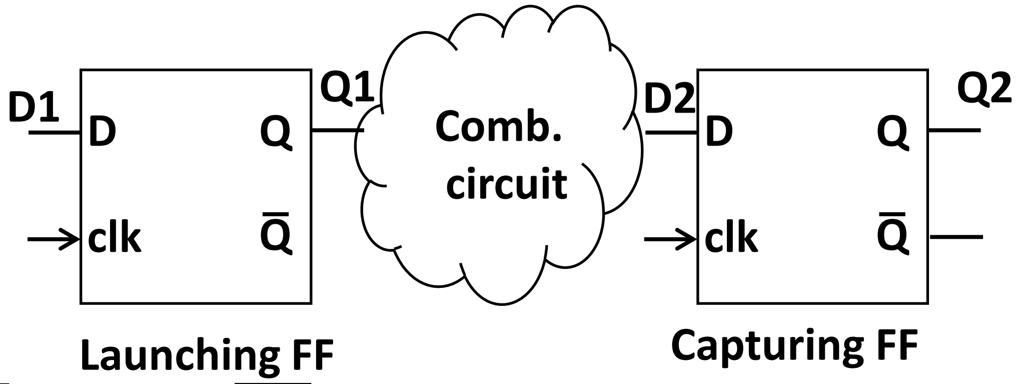


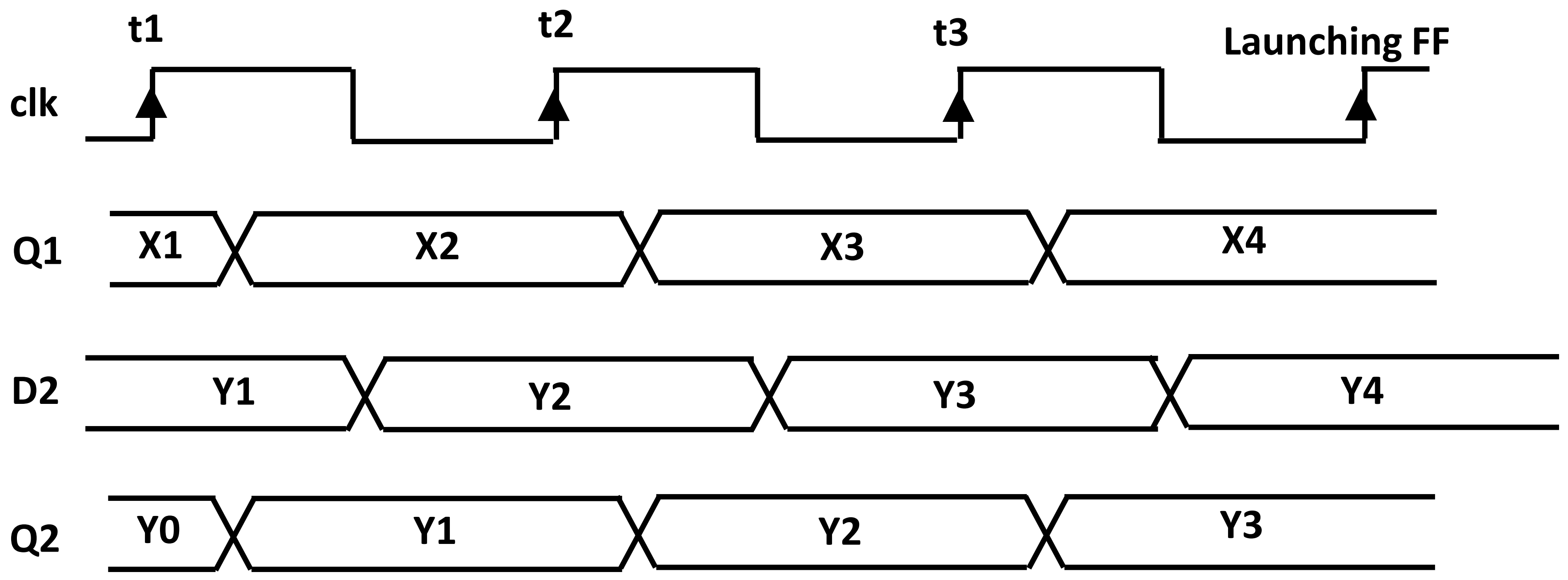
T_{setup} **of capturing FF**

T_{pmax} **of combinational circuit**

T_{clk_out} **of launching FF**

FF Timing Constraints



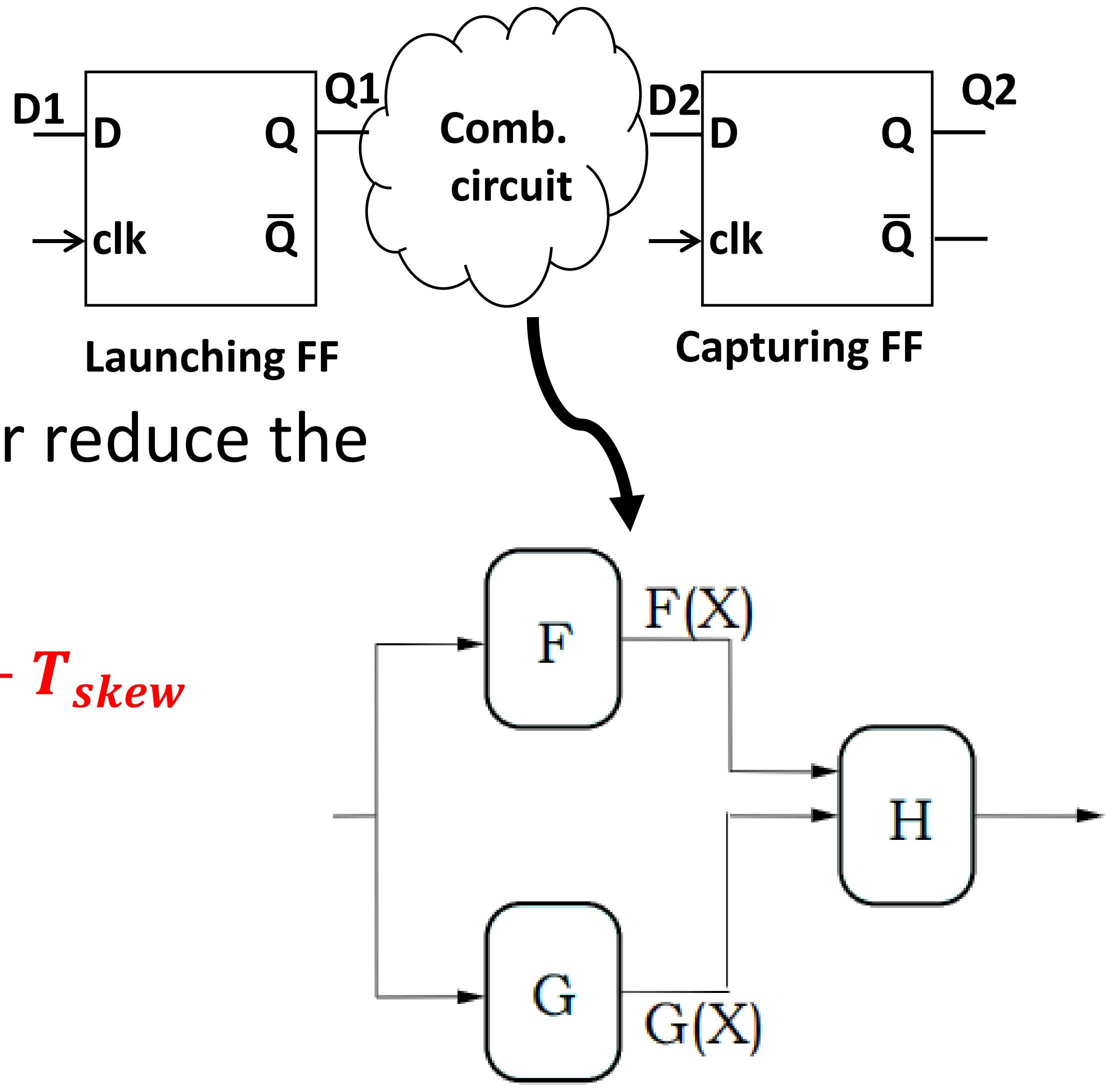


$$T_{clk} \geq T_{clk-out} + T_{pmax} + T_{setup} + T_{skew}$$

- This is very important equation since it puts limit on the minimum value of clock period and hence, maximum value of clock frequency.
- Limit on maximum value of clock frequency means limit on the speed at which the circuit can operates.

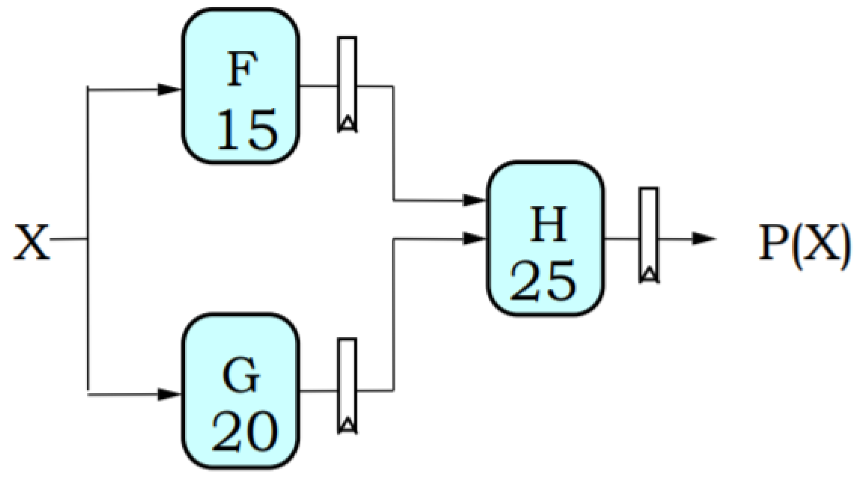
- How to increase the clock frequency or reduce the clock period?

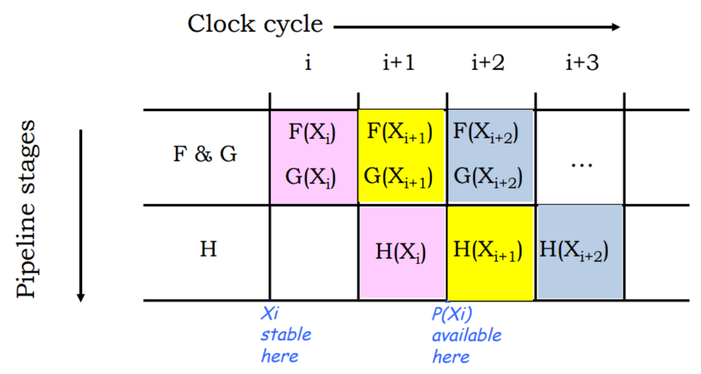
$$T_{clk} \geq T_{clk-out} + T_{pmax} + T_{setup} + T_{skew}$$



Pipelining

- Now, F and G can be working on input Xi+1 while H is performing its computations on Xi.
- This is 2-stage pipeline i.e. if we have valid input X during clock cycle j, P(X) is valid during clock cycle j+2.
- Assuming F, G and H have propagation delay of 15, 20 and 25 ns, and ideal registers i.e. FFs, then





	latency	throughput
unpipelined	45	1/45
2-stage pipeline	50	1/25

Pipelining (Summary)

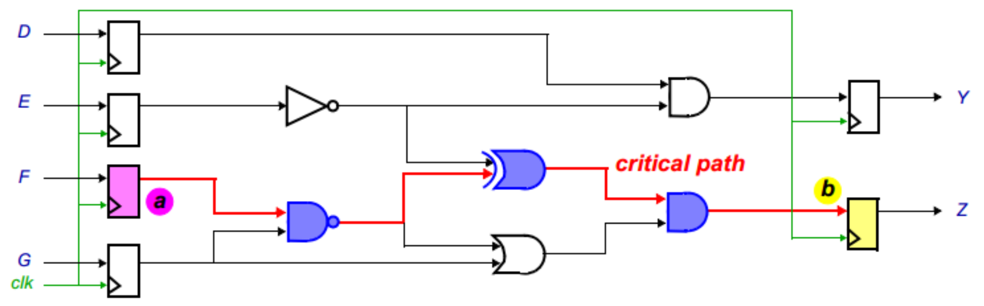
- A well-formed K-Stage Pipeline ("K-pipeline") circuit have exactly K registers on every path from an input to an output.
- A COMBINATIONAL CIRCUIT is thus a 0-stage pipeline.
- Every pipeline stage, hence every K-Stage pipeline, has a register on its OUTPUT (not on its input).
- The CLOCK common to all registers must have a period sufficient to cover propagation delay of combinational circuit, clock-to-output delay of launching FF, setup time of capturing FF and clock skew.
- The LATENCY of a K-stage pipeline is K times the period of the system's clock. The THROUGHPUT of a K-stage pipeline is the frequency of the clock.

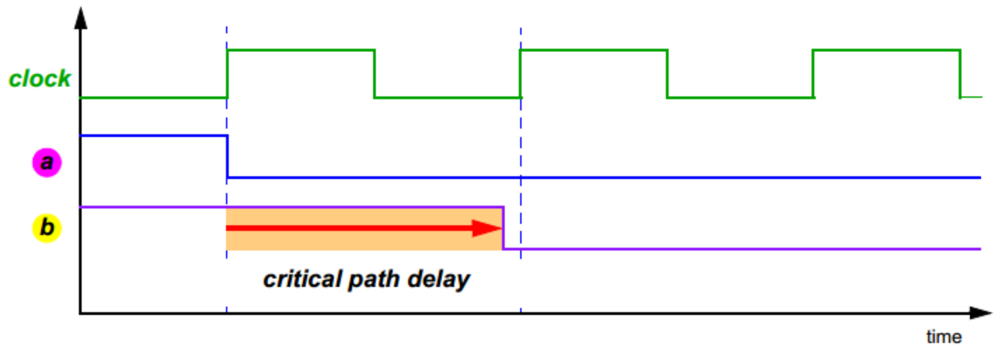
$$T_{clk} \geq \cancel{T_{clk-out}} + T_{pmax} \cancel{+ T_{setup}} \cancel{+ T_{skew}}$$

$$T_{clk} \geq \cancel{T_{clk-out}} + T_{pmax} \cancel{+ T_{setup}} \cancel{+ T_{skew}}$$

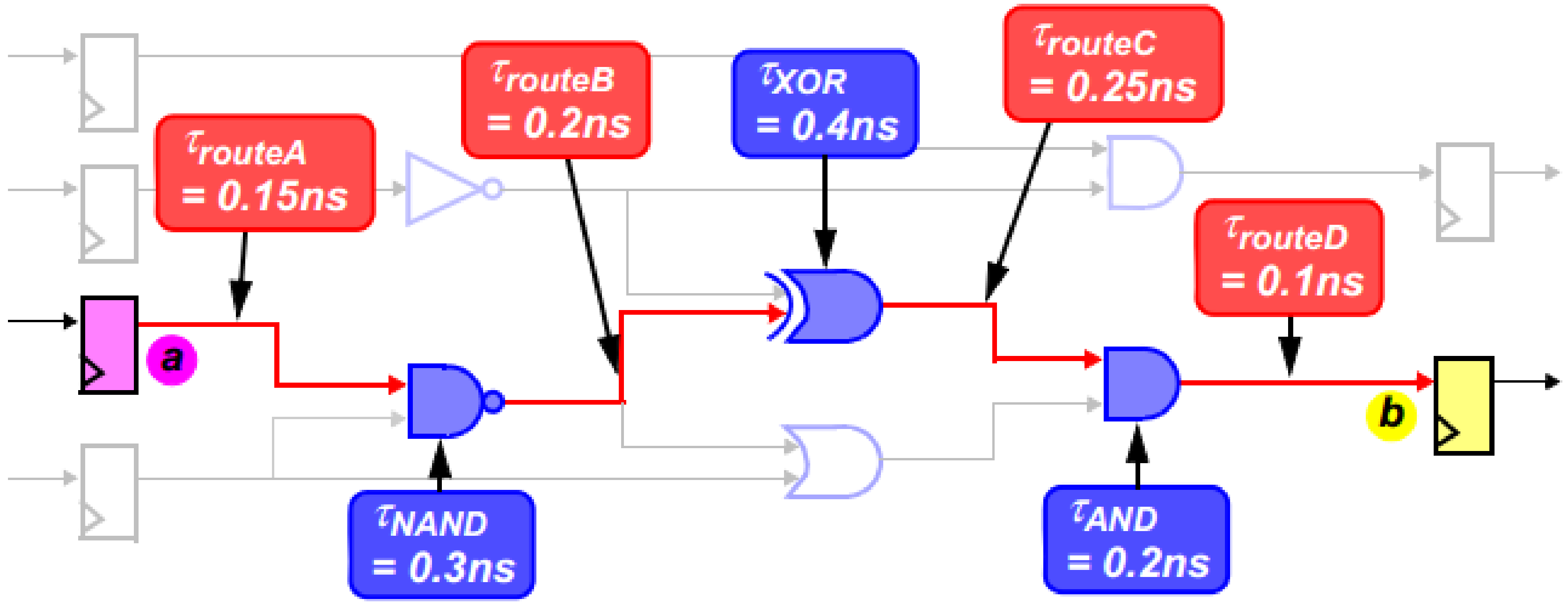
Critical Path Delay

- Signals experiences logic and routing delays through all logic paths, as they propagate from one clocked register (FF) to the next.
- Critical path delay is the delay along the critical path i.e. the longest combinatorial propagation delay through the circuit.
- Critical path delay must be shorter than one clock period in order to guarantee the correct operation of the circuit.

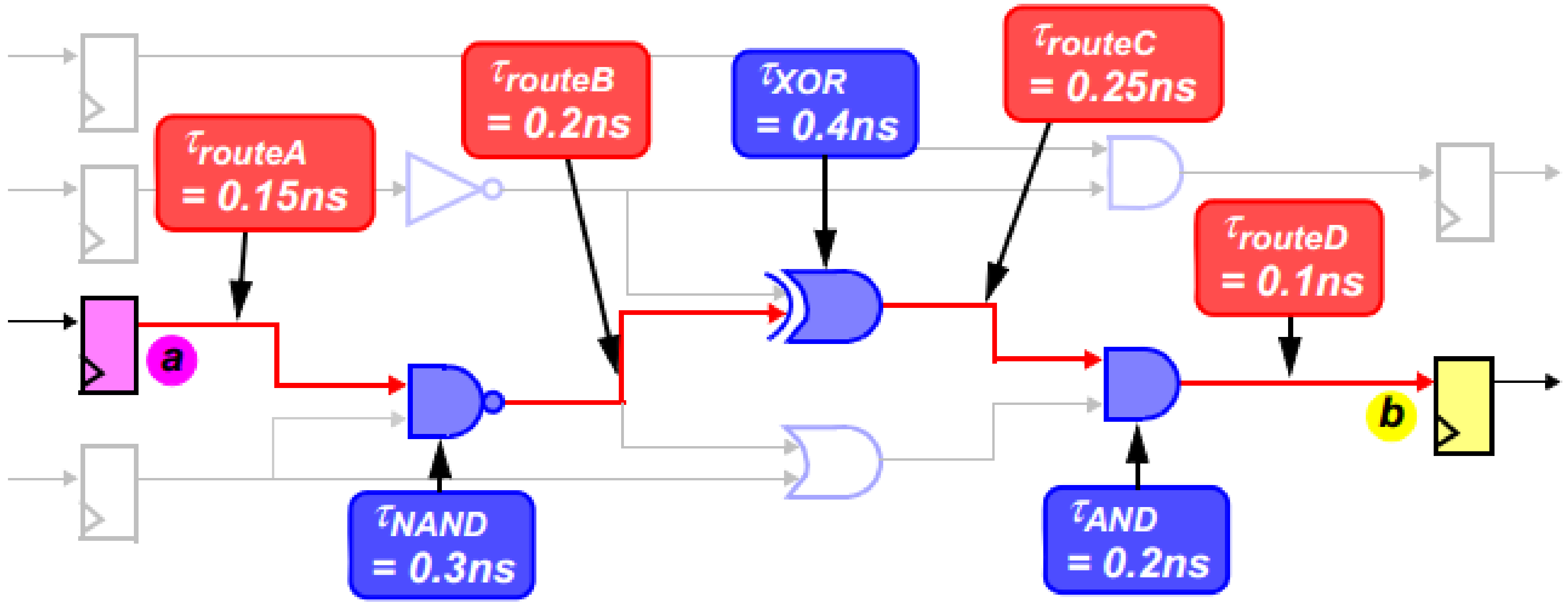




Critical Path Delay



Critical Path Delay



$$\tau_{CPD} = \underbrace{\tau_{NAND} + \tau_{XOR} + \tau_{AND}}_{\text{logic delays}} + \underbrace{\tau_{routeA} + \tau_{routeB} + \tau_{routeC} + \tau_{routeD}}_{\text{routing delays}}$$

$$\tau_{CPD} = \underbrace{0.3\text{ns} + 0.4\text{ns} + 0.2\text{ns}} + \underbrace{0.15\text{ns} + 0.2\text{ns} + 0.25\text{ns} + 0.1\text{ns}}$$
$$= \boxed{1.6\text{ns}}$$

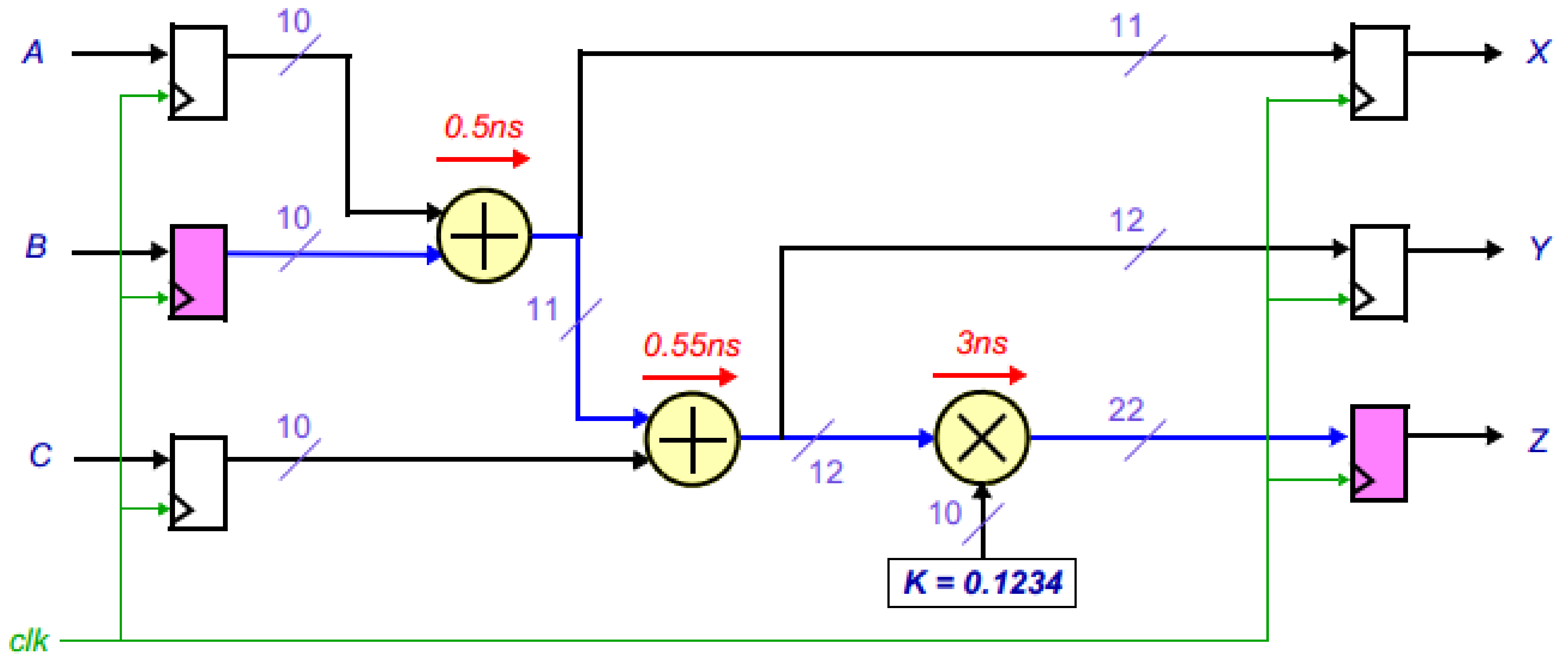
Maximum Clock Frequency

- ❖ Maximum clock frequency depends on critical path delay as given below

$$f_{clk_{max}} = \frac{1}{\tau_{CPD}} = \frac{1}{1.6ns} = 625MHz$$

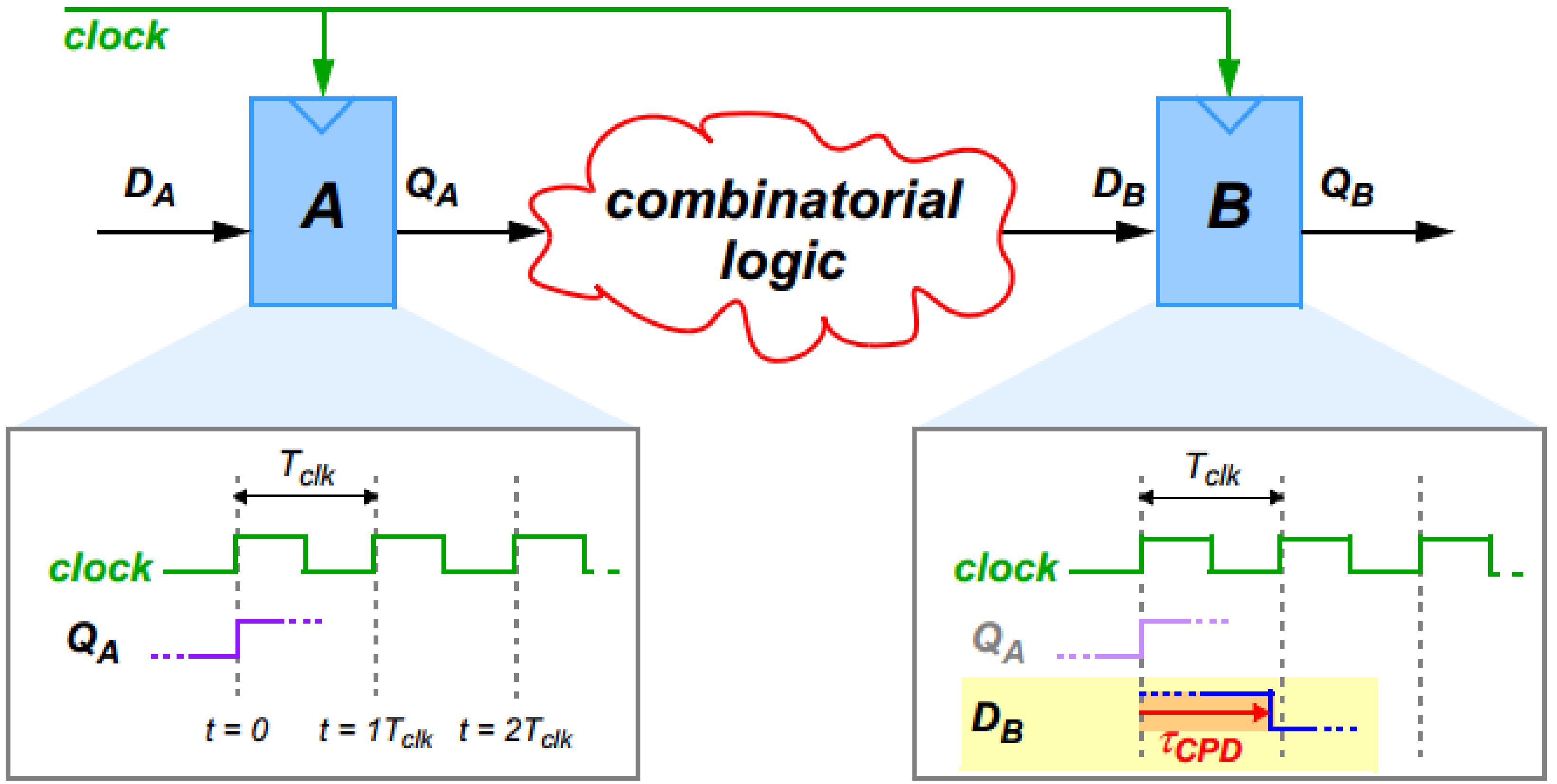
- ❖ If the clock frequency applied is less than 625 MHz, then a signal leaving one register arrives at the next register within one clock period.

Critical Path Delay



Maximum Clock Frequency

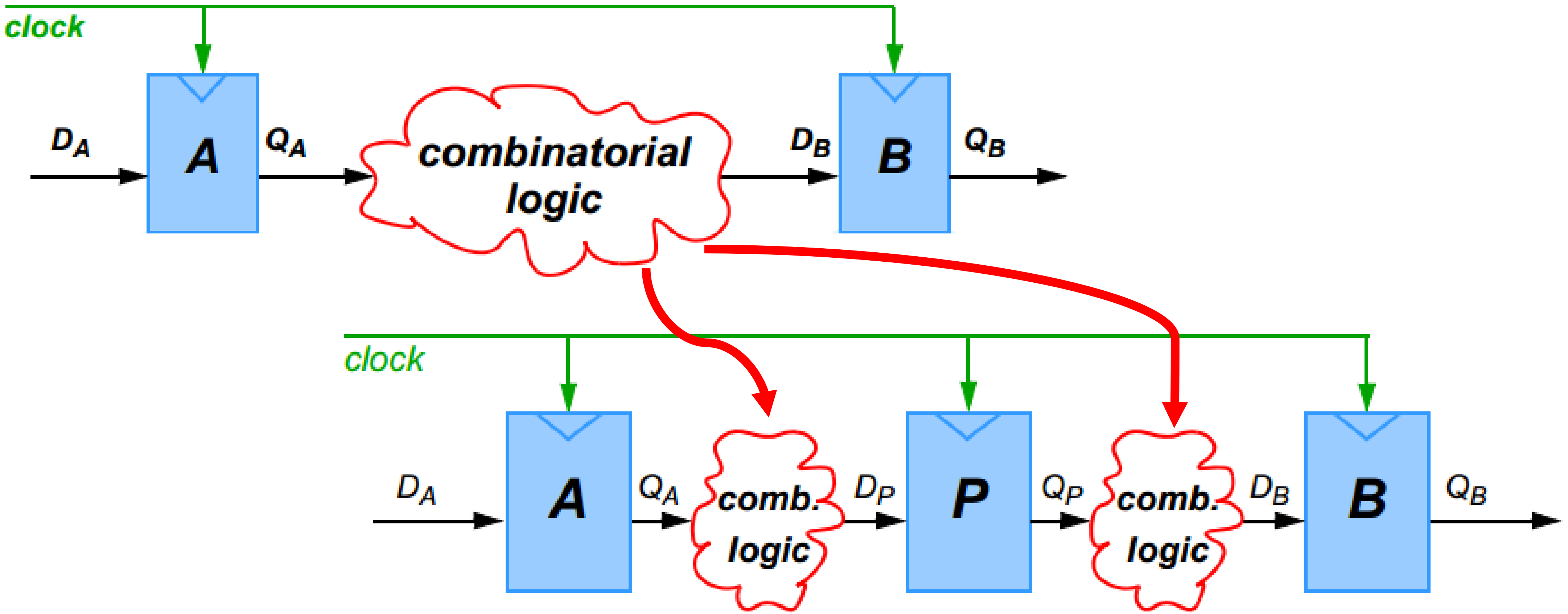
- Maximum clock frequency should be as high as possible



$$\boxed{T_{clk} > \tau_{CPD}}$$

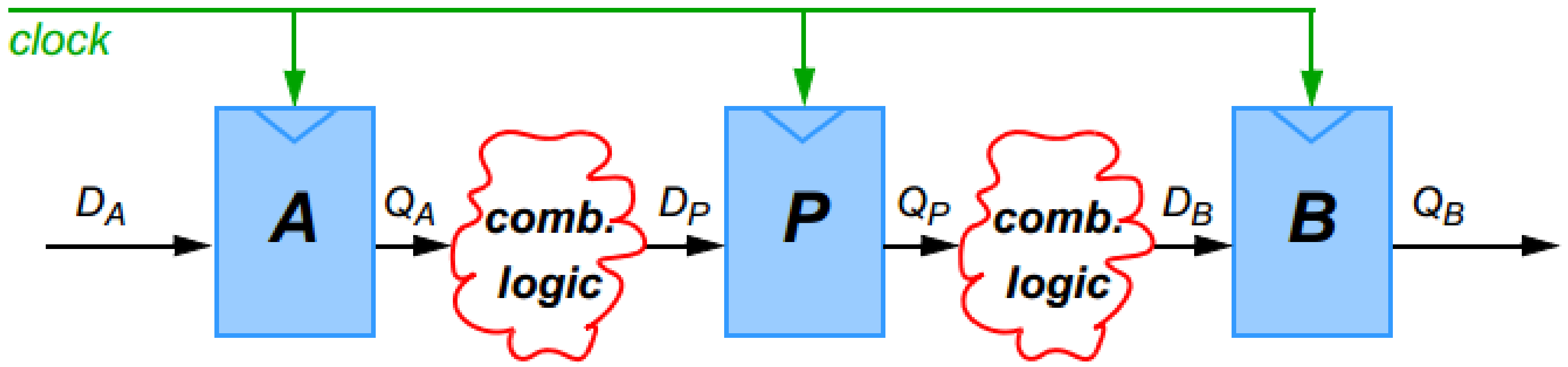
Pipelining

❖ Pipelining: Breaking of critical path by inserting additional registers



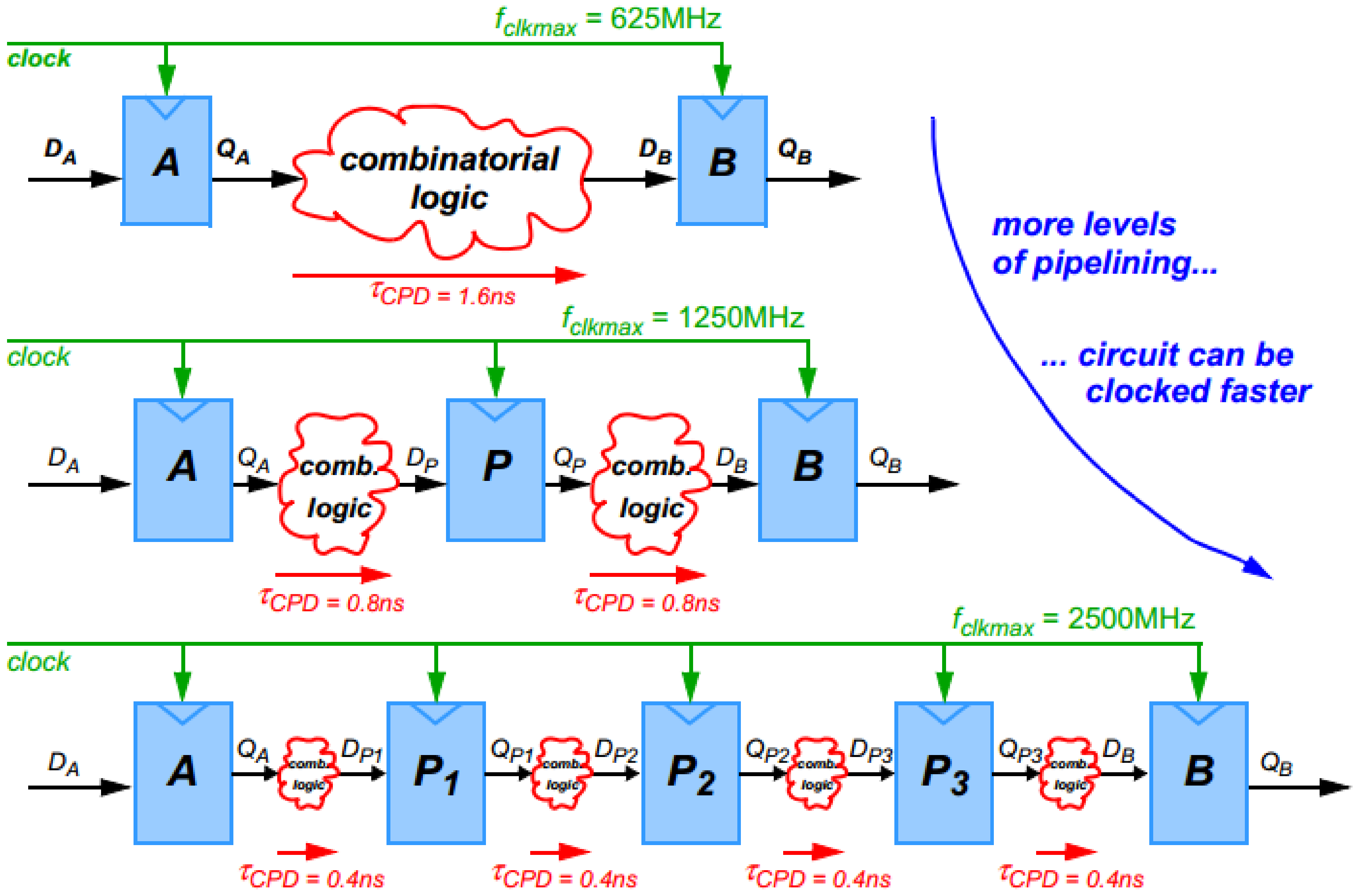
Pipelining

- ❖ Pipelining: Breaking of critical path by inserting additional registers



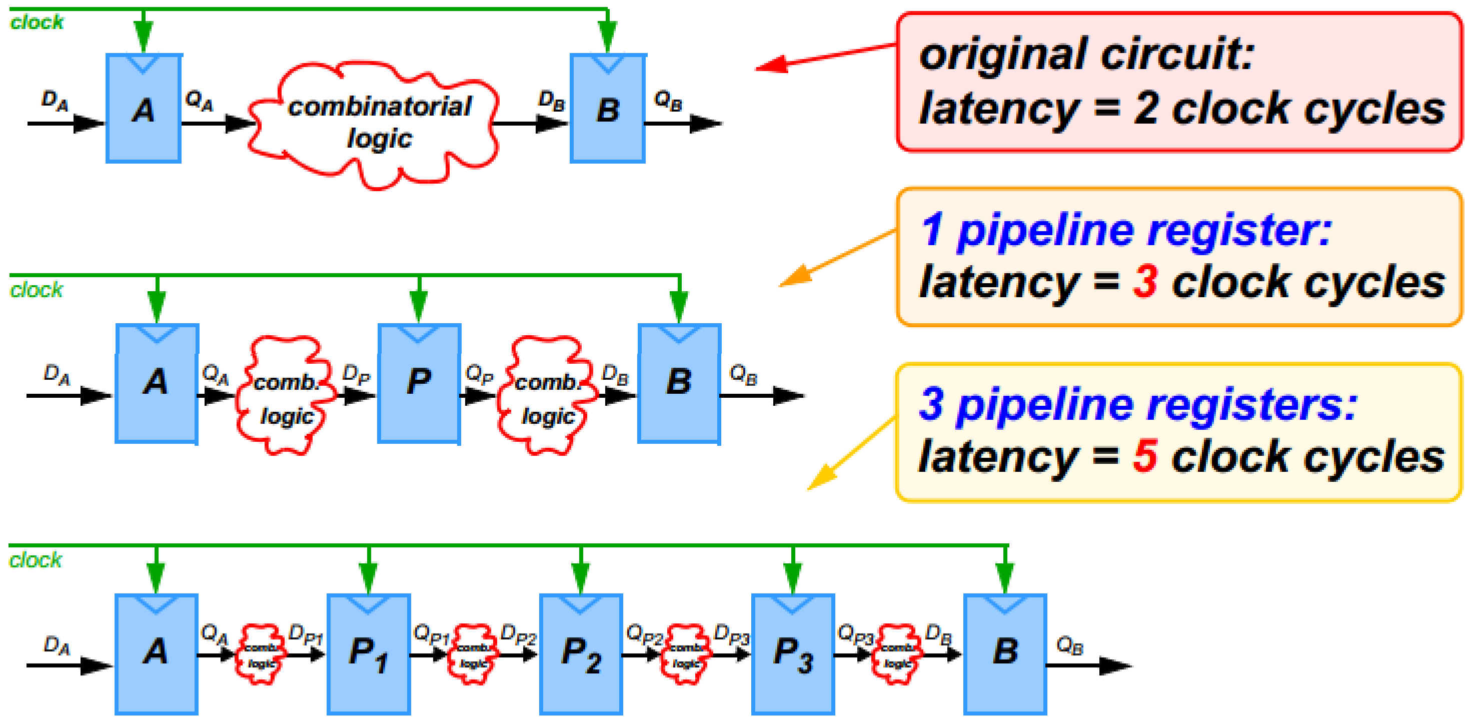
- ❖ The effect of adding the additional pipeline register is to increase the maximum clock frequency by a factor of 2 **(Is it true?)**

Pipelining

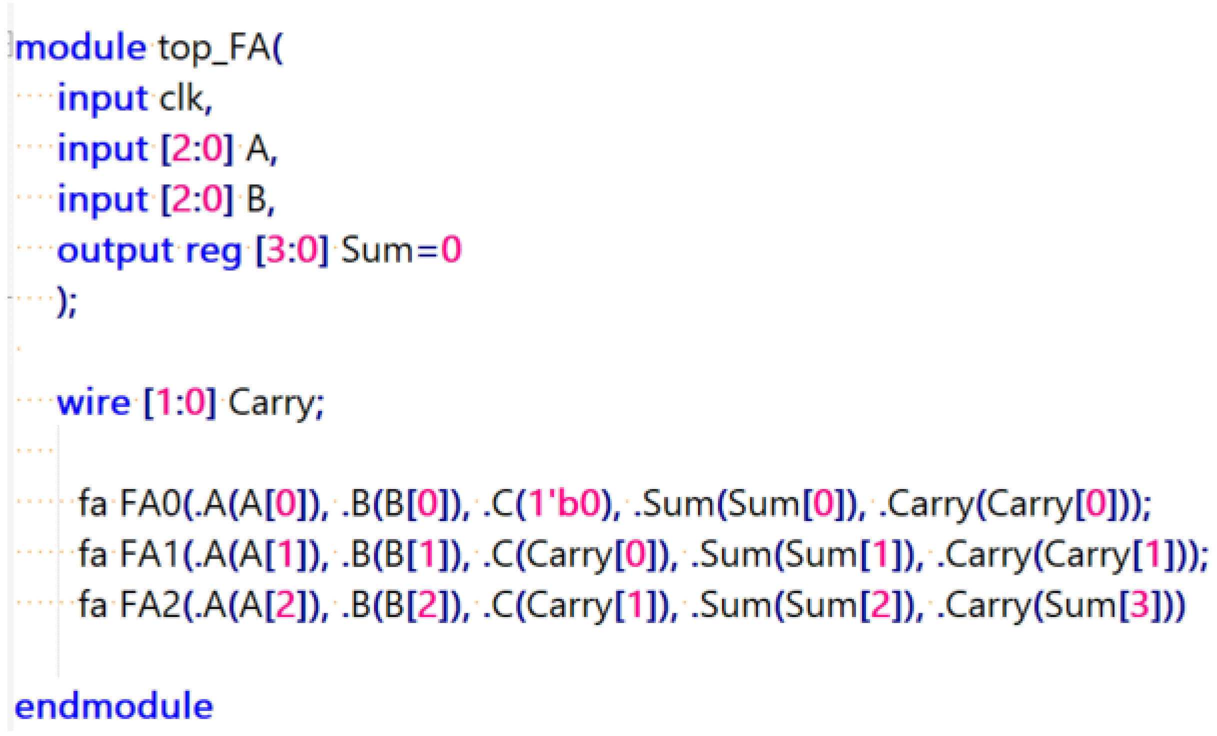

f_{clkmax} = 625MHz
clock
D_A
A
Q_A
combinatorial logic
D_B
B
Q_B
τ_{CPD} = 1.6ns
f_{clkmax} = 1250MHz
clock
D_A
A
Q_A
comb. logic
D_P
P
Q_P
comb. logic
D_B
B
Q_B
τ_{CPD} = 0.8ns
τ_{CPD} = 0.8ns
f_{clkmax} = 2500MHz
clock
D_A
A
Q_A
comb. logic
D_{P1}
P_1
Q_{P1}
comb. logic
D_{P2}
P_2
Q_{P2}
comb. logic
D_{P3}
P_3
Q_{P3}
comb. logic
D_B
B
Q_B
τ_{CPD} = 0.4ns
τ_{CPD} = 0.4ns
τ_{CPD} = 0.4ns
τ_{CPD} = 0.4ns
more levels of pipelining...
... circuit can be clocked faster

Pipelining

- Inserting pipeline registers clearly has an effect on the latency of the circuit i.e. the number of clock cycles from input to output.
- For every pipeline register, the latency increases by 1 clock cycle

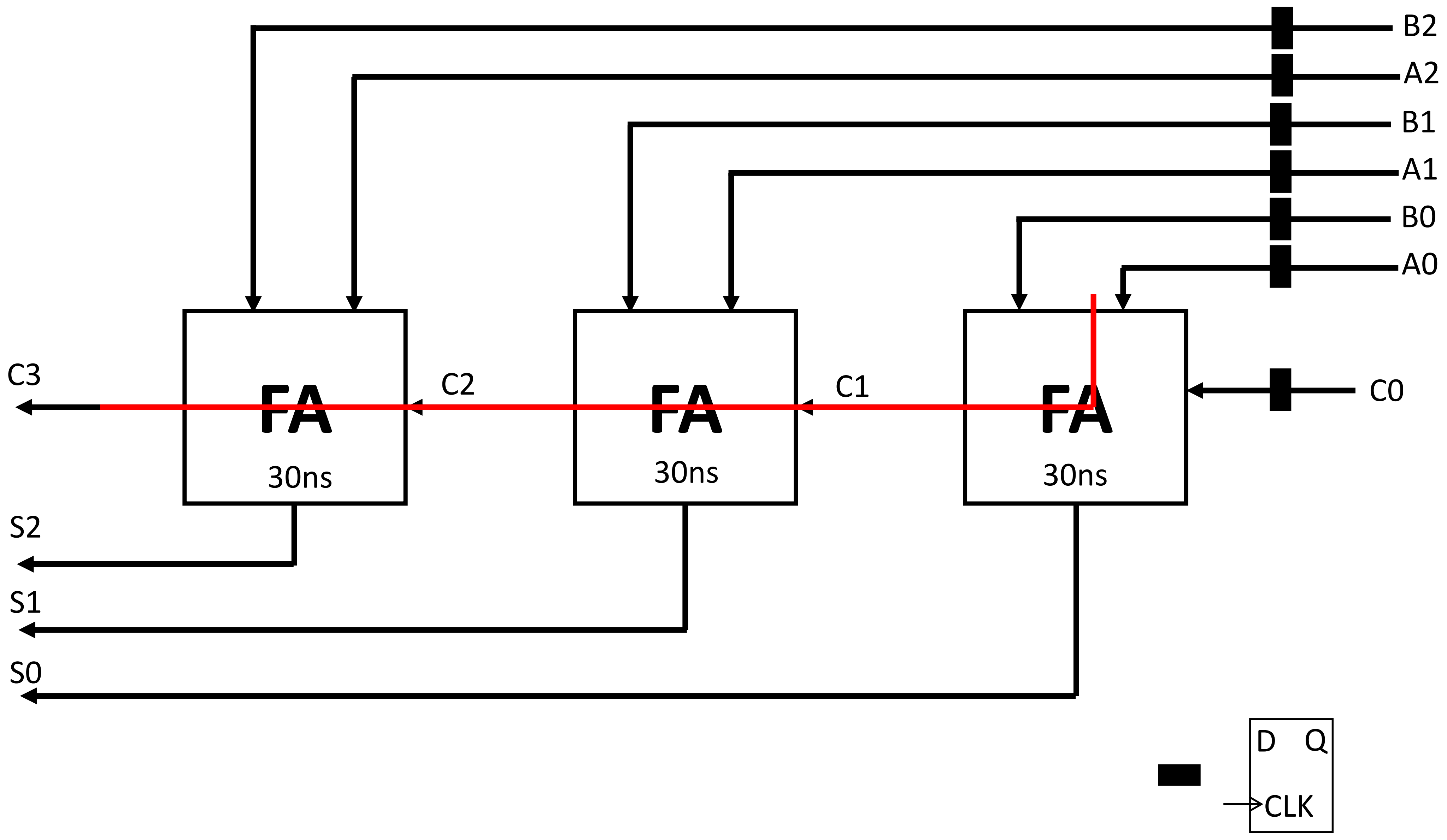


Pipelining



```
module top_FA(
    input clk,
    input [2:0] A,
    input [2:0] B,
    output reg [3:0] Sum=0
    );

    wire [1:0] Carry;

     fa FA0(.A(A[0]), .B(B[0]), .C(1'b0), .Sum(Sum[0]), .Carry(Carry[0]));
     fa FA1(.A(A[1]), .B(B[1]), .C(Carry[0]), .Sum(Sum[1]), .Carry(Carry[1]));
     fa FA2(.A(A[2]), .B(B[2]), .C(Carry[1]), .Sum(Sum[2]), .Carry(Sum[3]))

endmodule
```

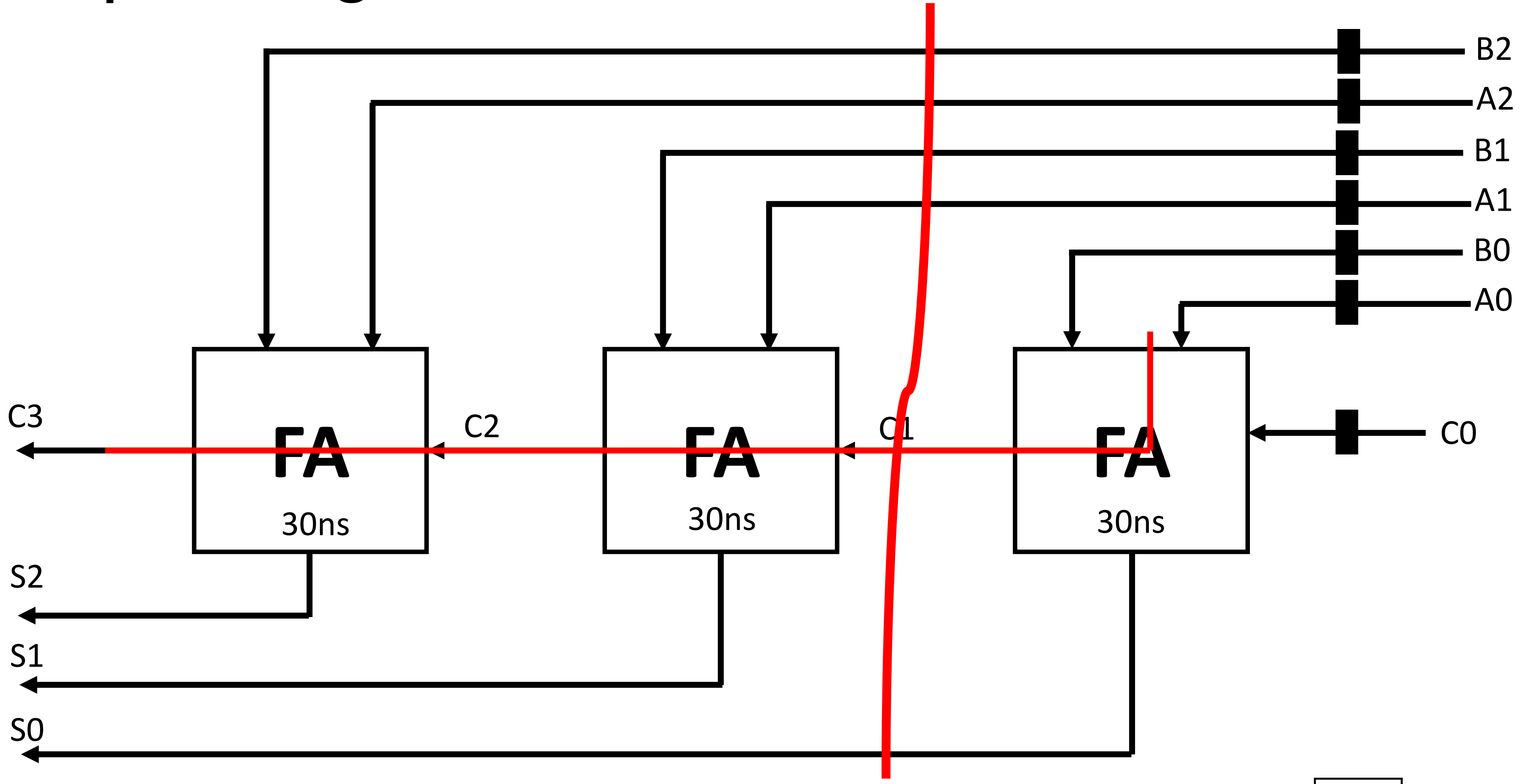
Pipelining

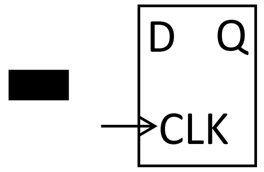


Critical Path Delay

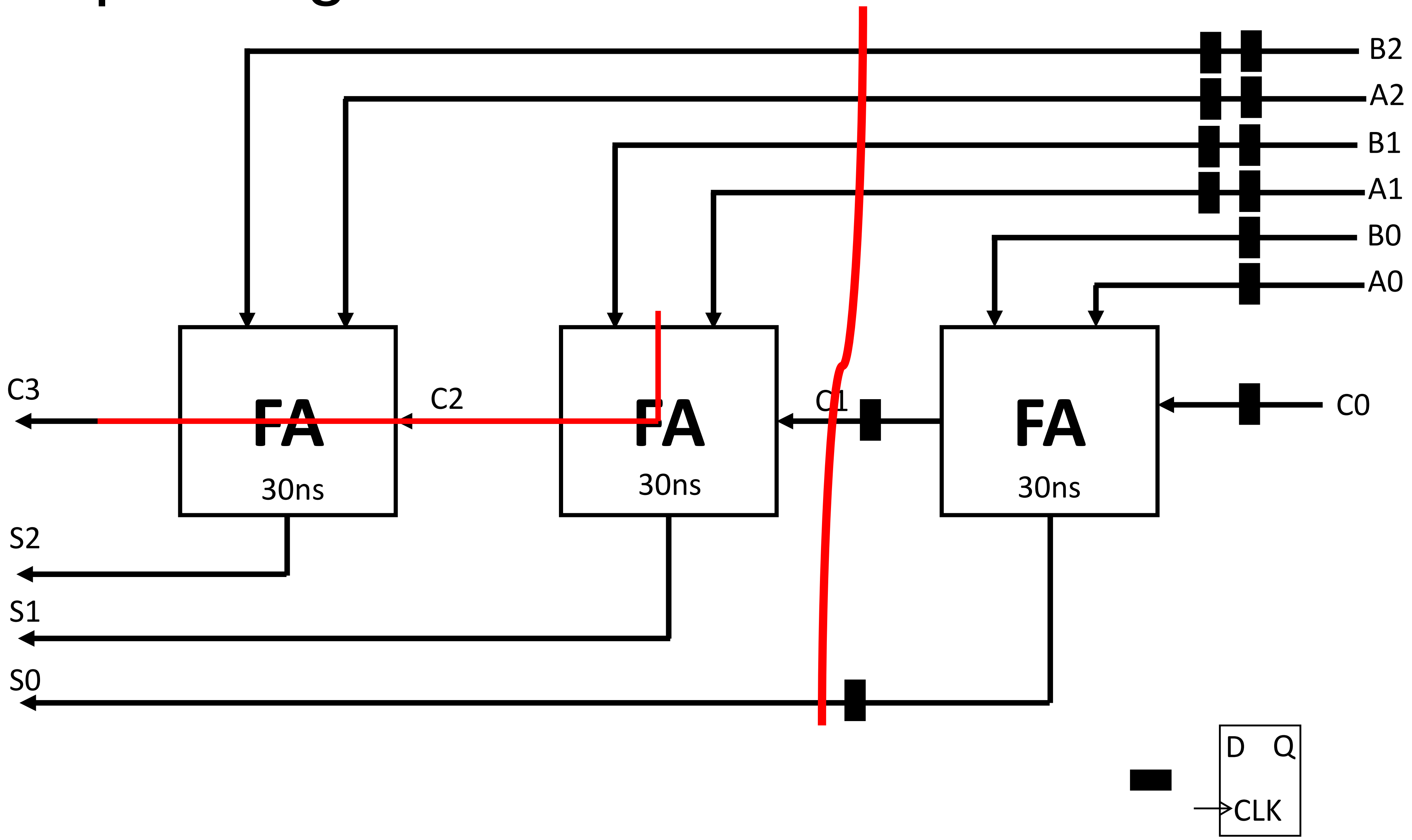
- Does arithmetic word lengths affect critical path delay???
- The last bit of result is not available until the calculation is complete and all the necessary carries have propagated from LSB to MSB.
- Hence, longer the arithmetic word length, longer is the critical path delay.

Pipelining

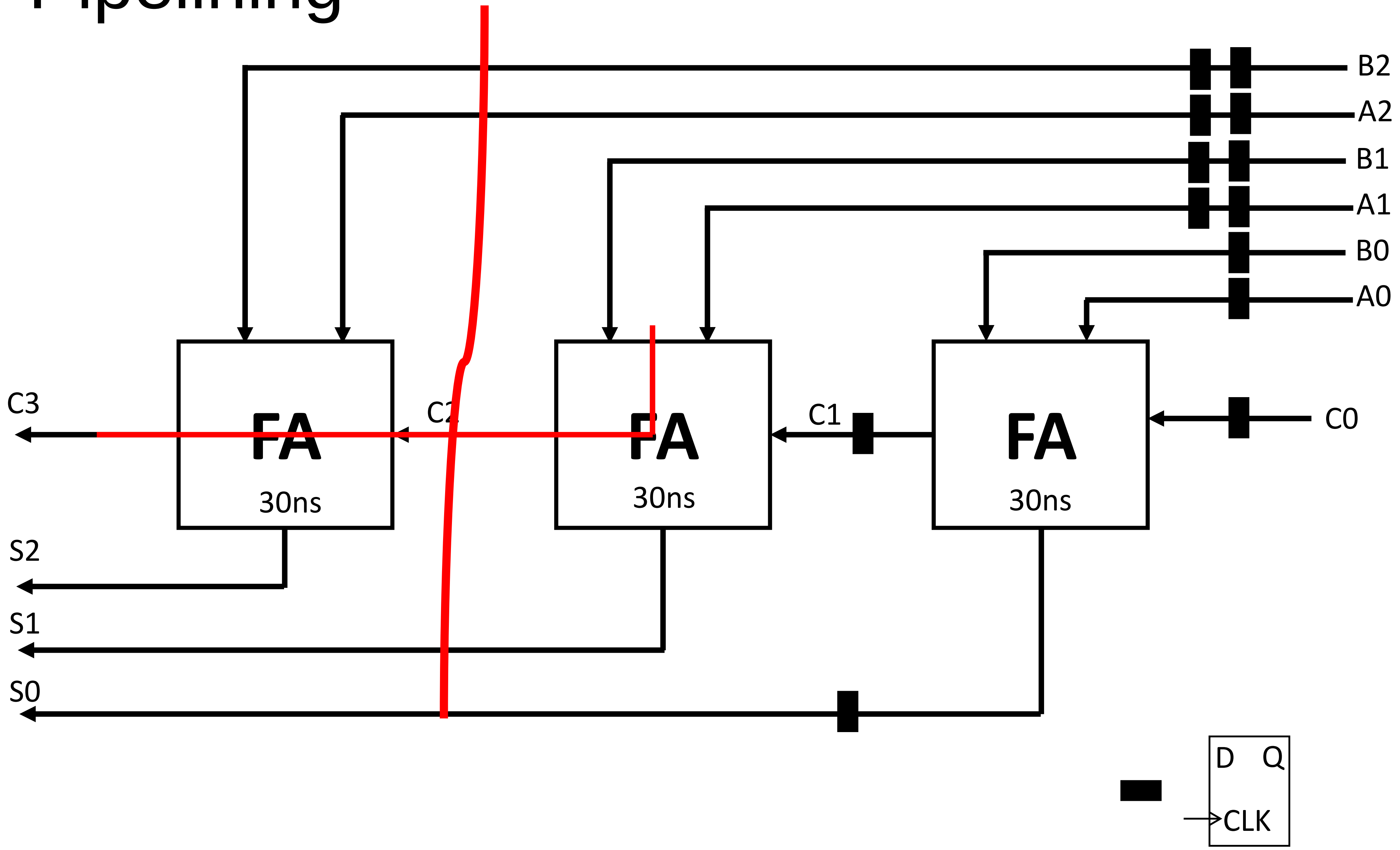




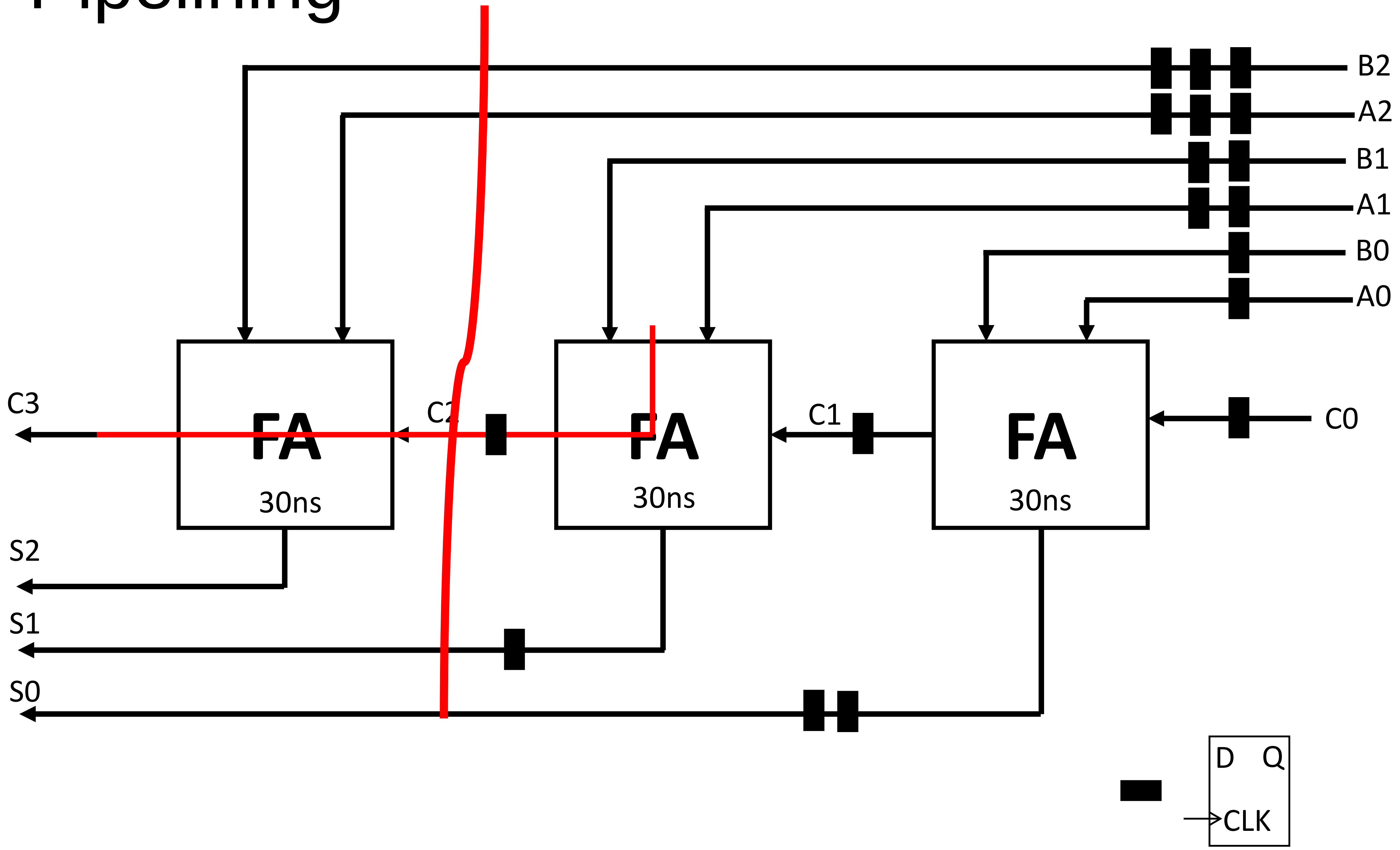
Pipelining



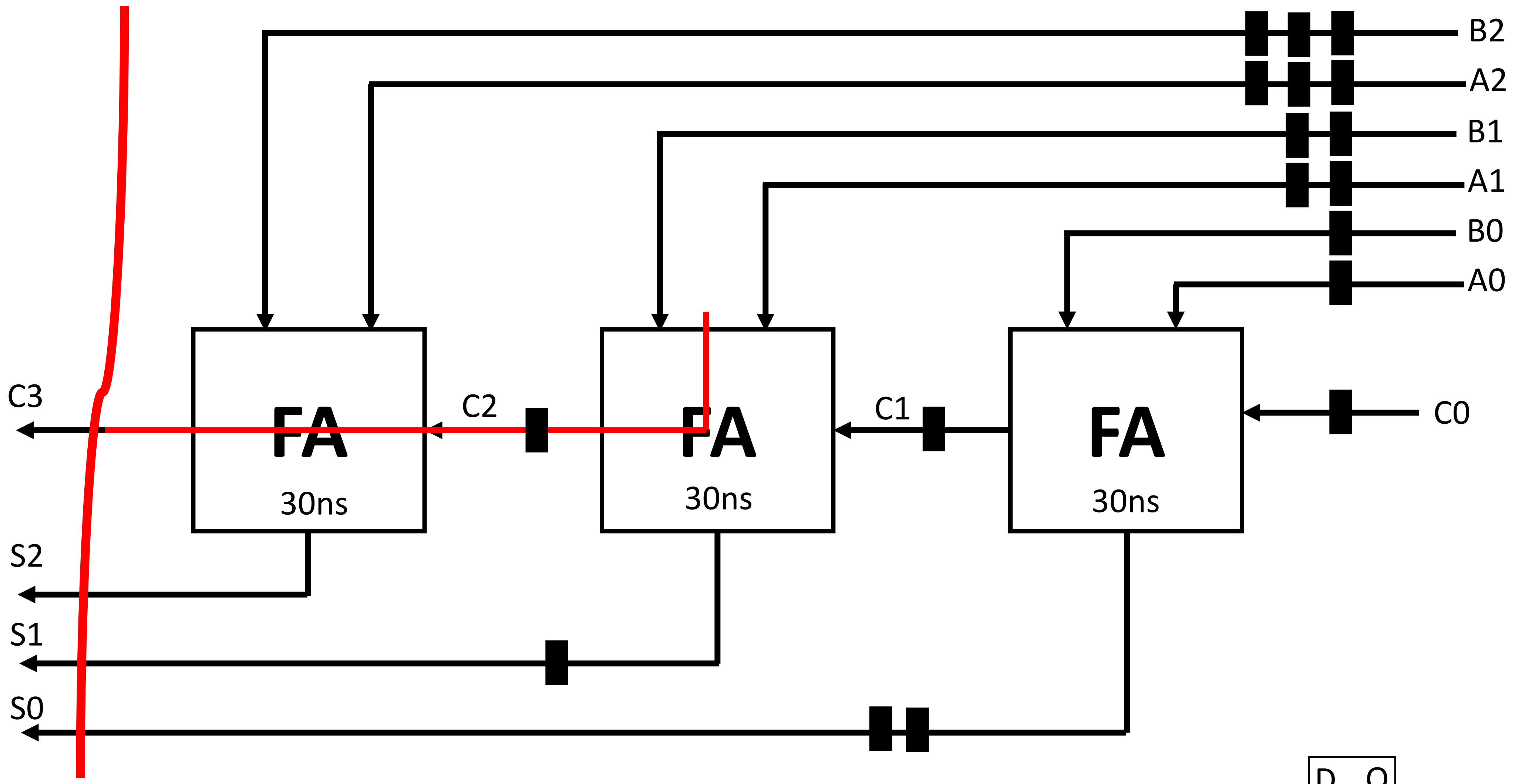
Pipelining

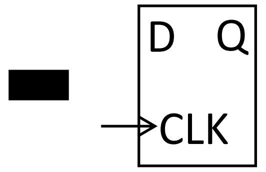


Pipelining

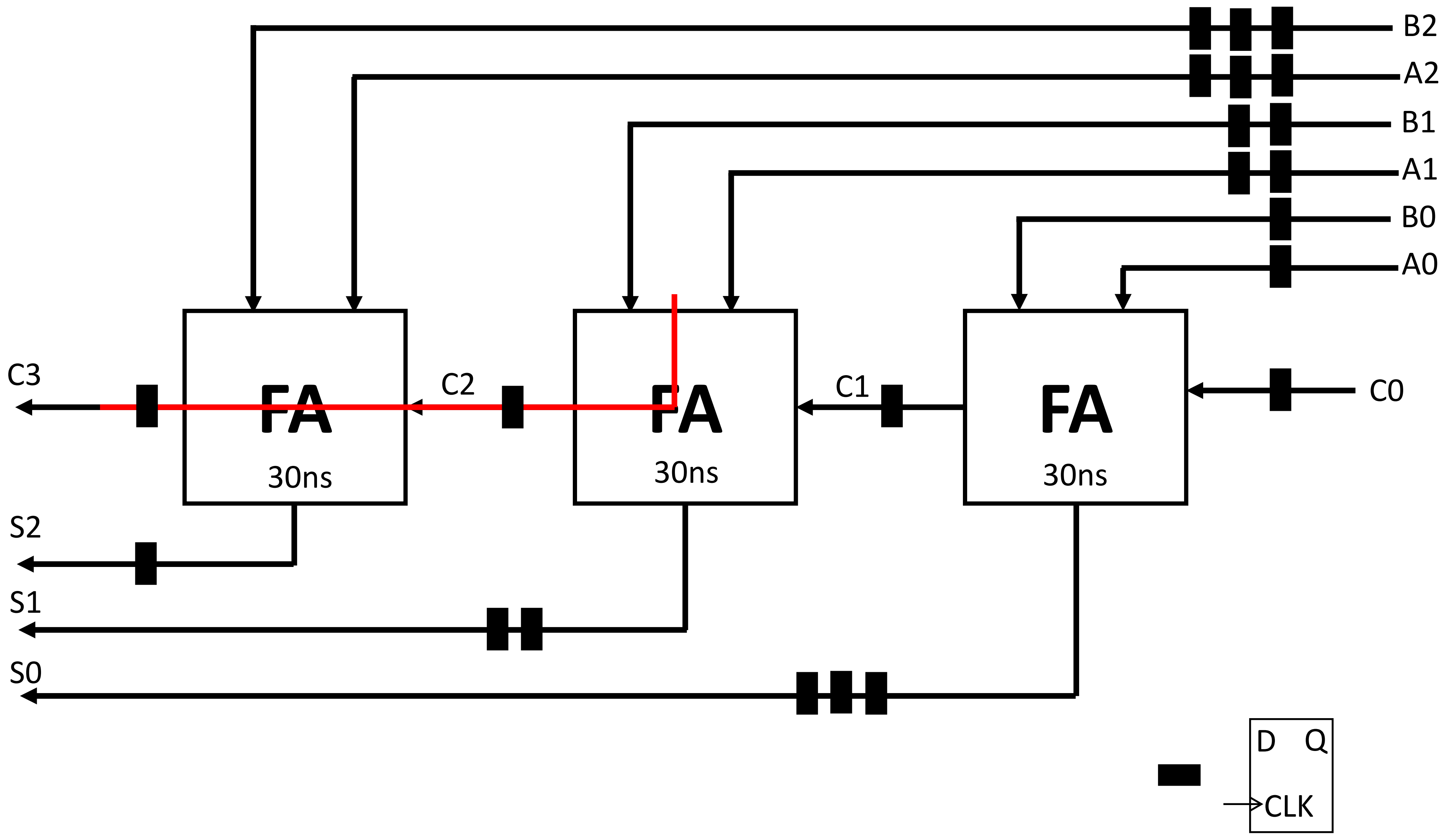


Pipelining

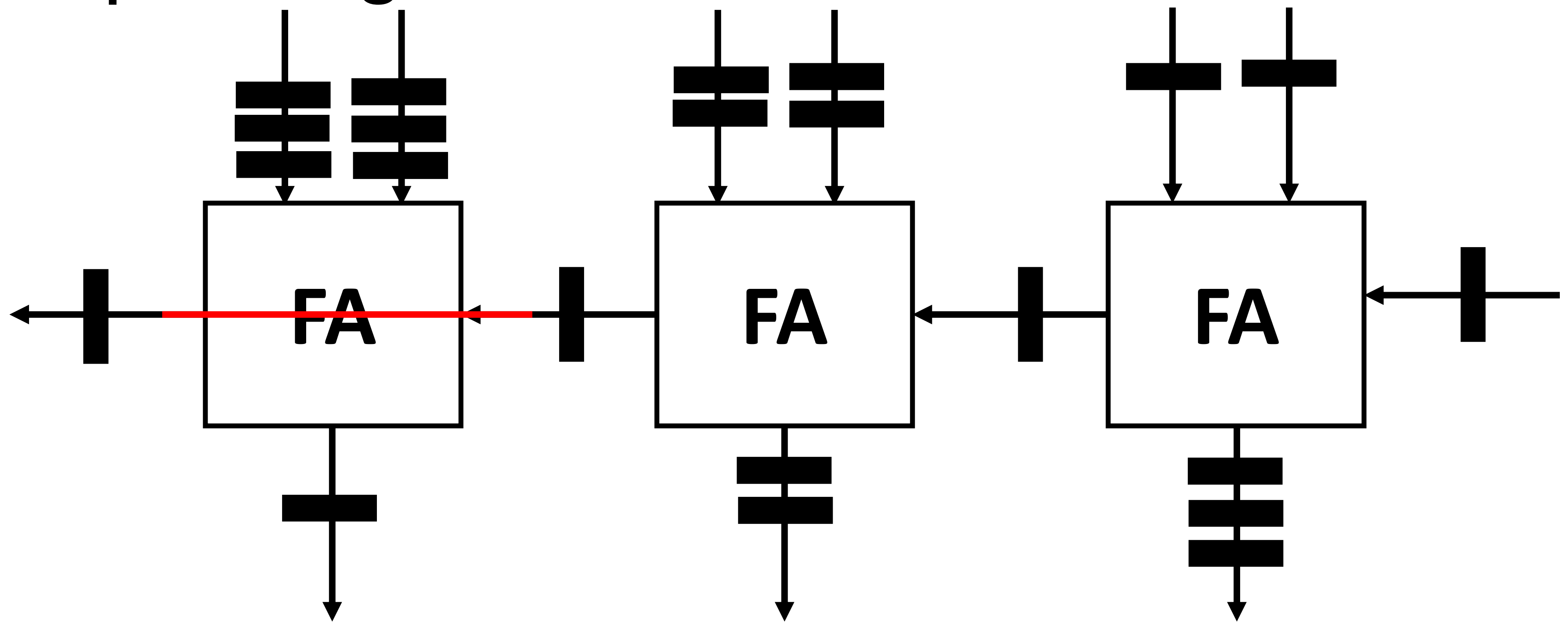


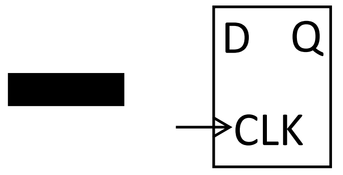


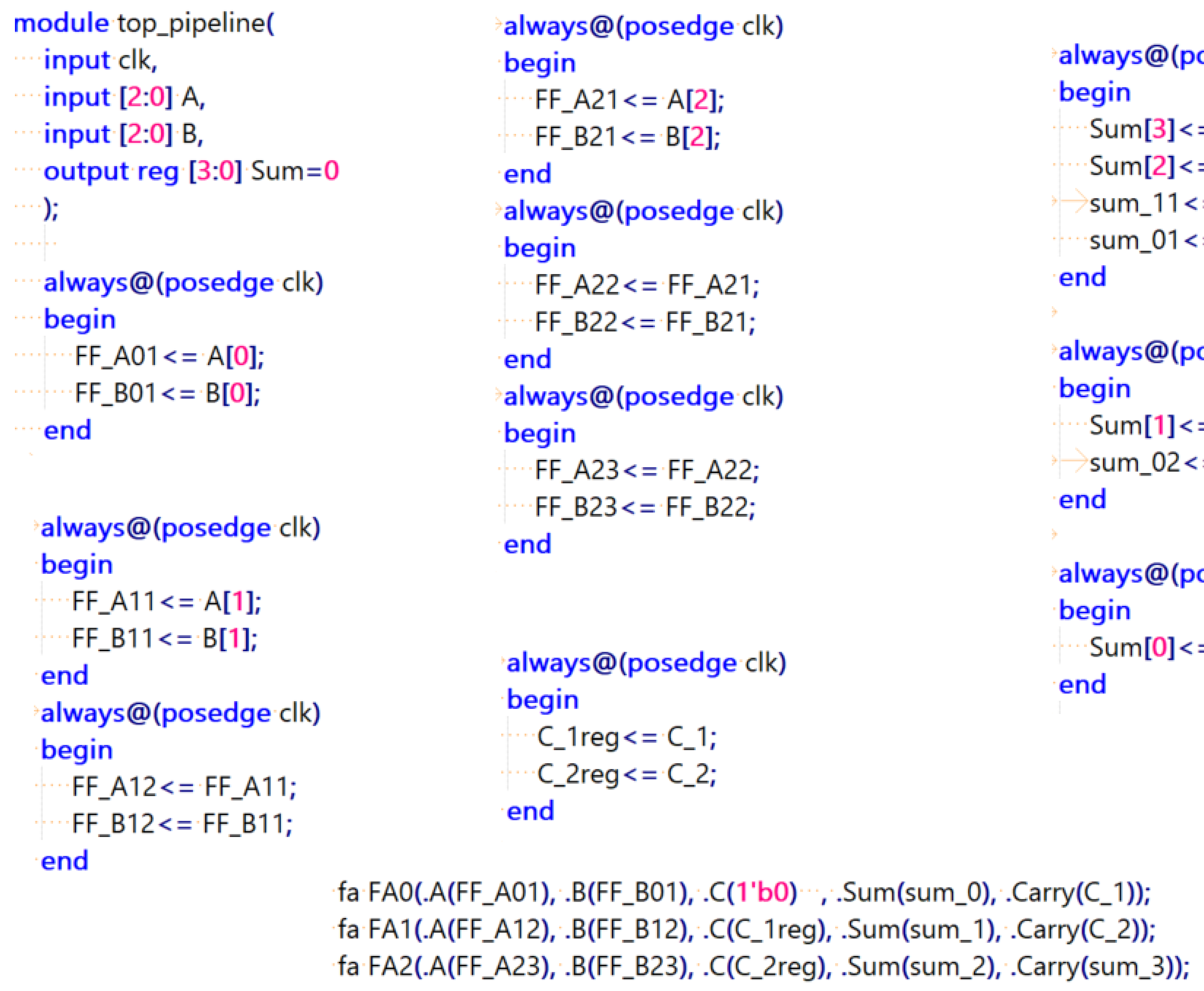
Pipelining



Pipelining

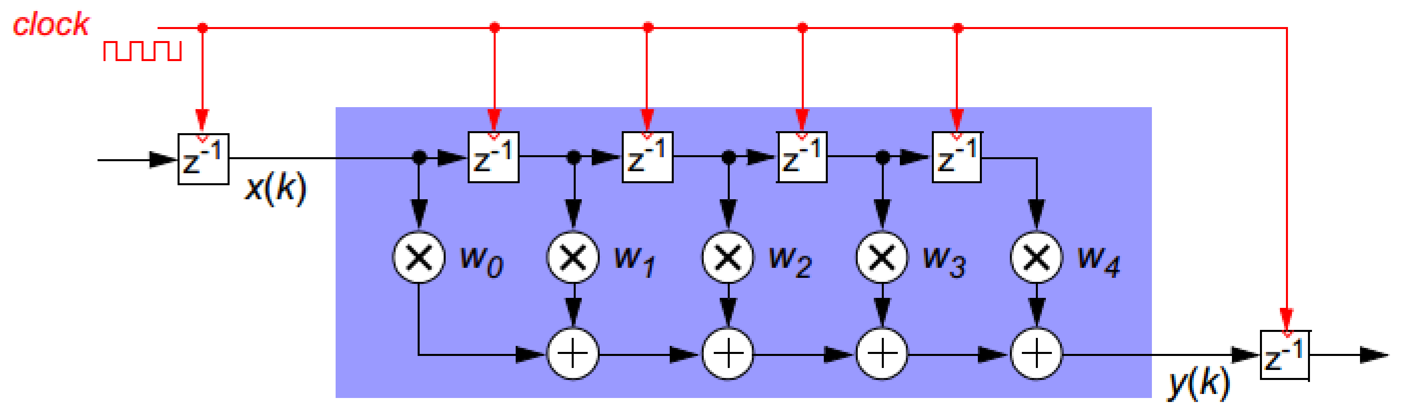






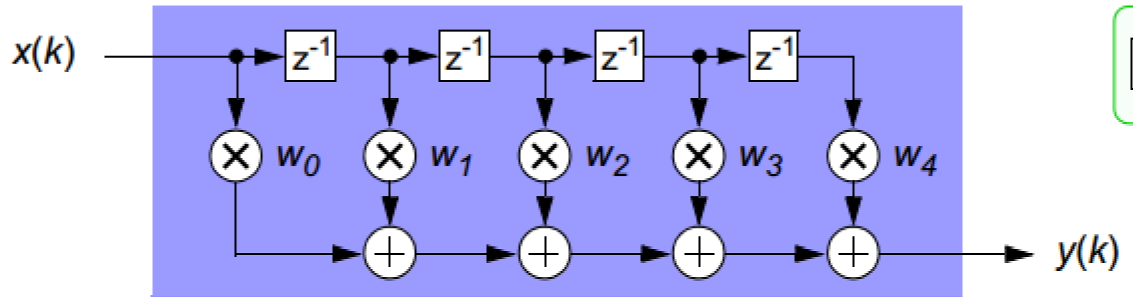
```verilog
module top_pipeline(
    input clk,
    input [2:0] A,
    input [2:0] B,
    output reg [3:0] Sum=0
    );

    always@(posedge clk)
    begin
        FF_A01<= A[0];
        FF_B01<= B[0];
    end


  always@(posedge clk)
  begin
      FF_A11<= A[1];
      FF_B11<= B[1];
  end
  always@(posedge clk)
  begin
      FF_A12<= FF_A11;
      FF_B12<= FF_B11;
  end

always@(posedge clk)
begin
    FF_A21<= A[2];
    FF_B21<= B[2];
end
always@(posedge clk)
begin
    FF_A22<= FF_A21;
    FF_B22<= FF_B21;
end
always@(posedge clk)
begin
    FF_A23<= FF_A22;
    FF_B23<= FF_B22;
end


always@(posedge clk)
begin
    C_1reg<= C_1;
    C_2reg<= C_2;
end

always@(posedge clk)
begin
    Sum[3]<= sum_3;
    Sum[2]<= sum_2;
    sum_11<= sum_1;
    sum_01<= sum_0;
end

always@(posedge clk)
begin
    Sum[1]<= sum_11;
    sum_02<= sum_01;
end

always@(posedge clk)
begin
    Sum[0]<= sum_02;
end

fa FA0(.A(FF_A01), .B(FF_B01), .C(1'b0)   , .Sum(sum_0), .Carry(C_1));
fa FA1(.A(FF_A12), .B(FF_B12), .C(C_1reg), .Sum(sum_1), .Carry(C_2));
fa FA2(.A(FF_A23), .B(FF_B23), .C(C_2reg), .Sum(sum_2), .Carry(sum_3));
```

FIR Filter



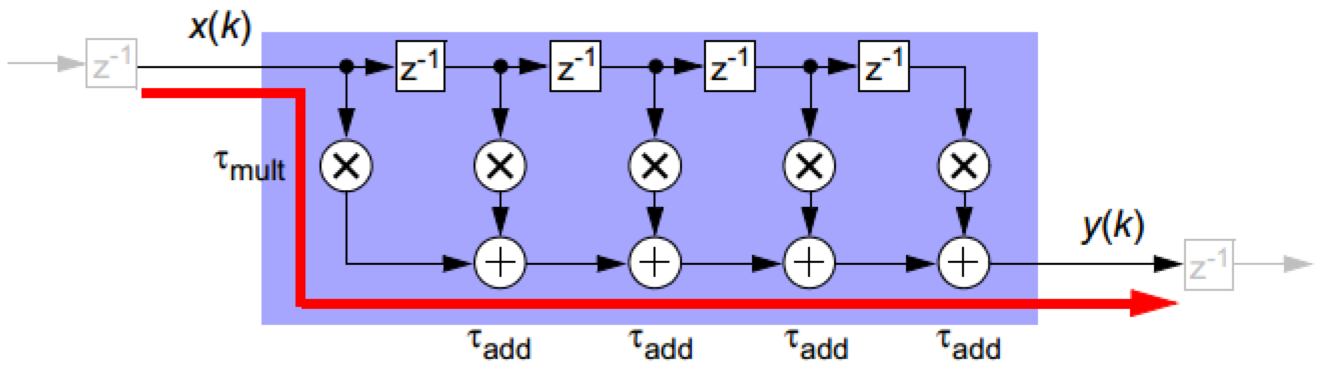
_N_-weight Finite Impulse Response (FIR)

$$y(k) = \sum_{n=0}^{N-1} x(k-n)w_n$$



z^{-1} 1 sample delay

FIR Filter



***N*-weight Finite Impulse Response (FIR)**

$$y(k) = \sum_{n=0}^{N-1} x(k-n) w_n$$

$$\textbf{critical path delay} = \tau_{\text{mult}} + 4\tau_{\text{add}}$$

$$f_{\text{clk(max)}} = \frac{1}{\tau_{\text{mult}} + 4\tau_{\text{add}}}$$

$\tau_{\text{add}} = 0.1\text{ns}$ $\tau_{\text{mult}} = 1\text{ns}$

$$f_{\text{clk}} = \frac{1}{1 + (4 \times 0.1)} \text{ x } 10^9 \approx \textbf{714MHz}$$

$$f_{\text{clk}} = \frac{1}{1 + (9 \times 0.1)} \text{ x } 10^9 \approx \textbf{526MHz}$$

zero sample latency between input and output

How to Pipeline Architecture?