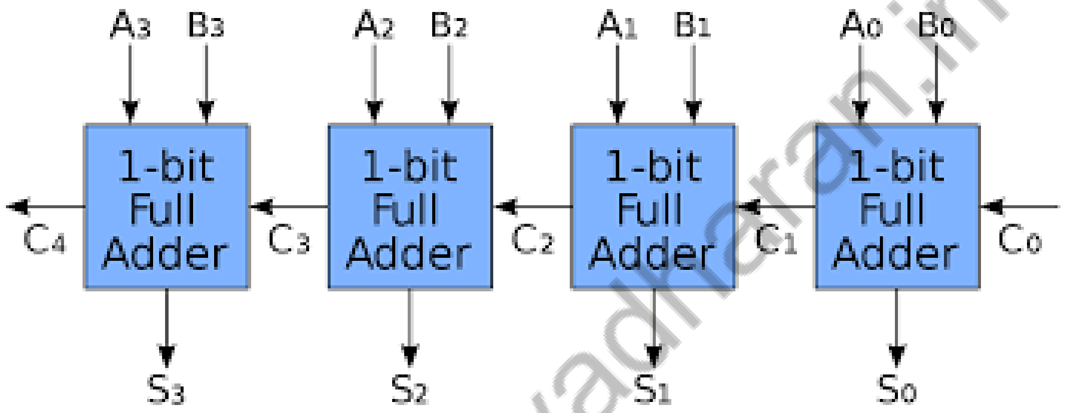
Advanced VLSI Design: 2021-22
Lecture 4A
Pipelined Registers

By Dr. Sanjay Vidhyadharan

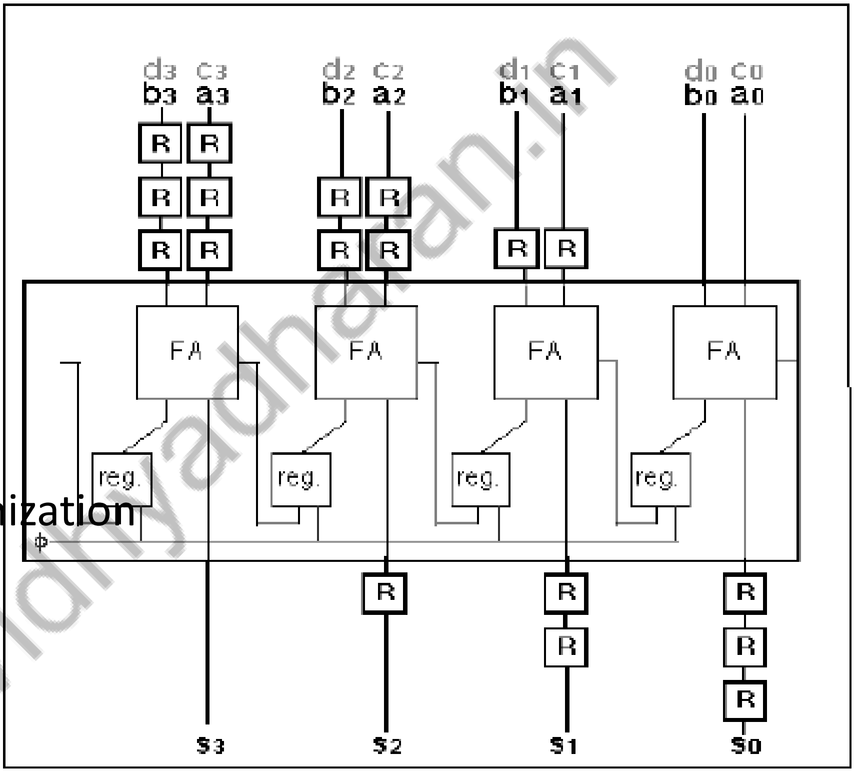
4-Bit Adder



Performance Figure	Value
Function Delay	$3 * \tau_{pd\_adder(Carry)} + \tau_{pd\_adder(Sum)}$
Power	$4 * P_{adder}$
Throughput	$@\ (3 * \tau_{pd\_adder(Carry)} + \tau_{pd\_adder(Sum)})$
Gate complexity	$4 * G_{adder}$

Pipelined 4-Bit Adder

Principle of Operation:

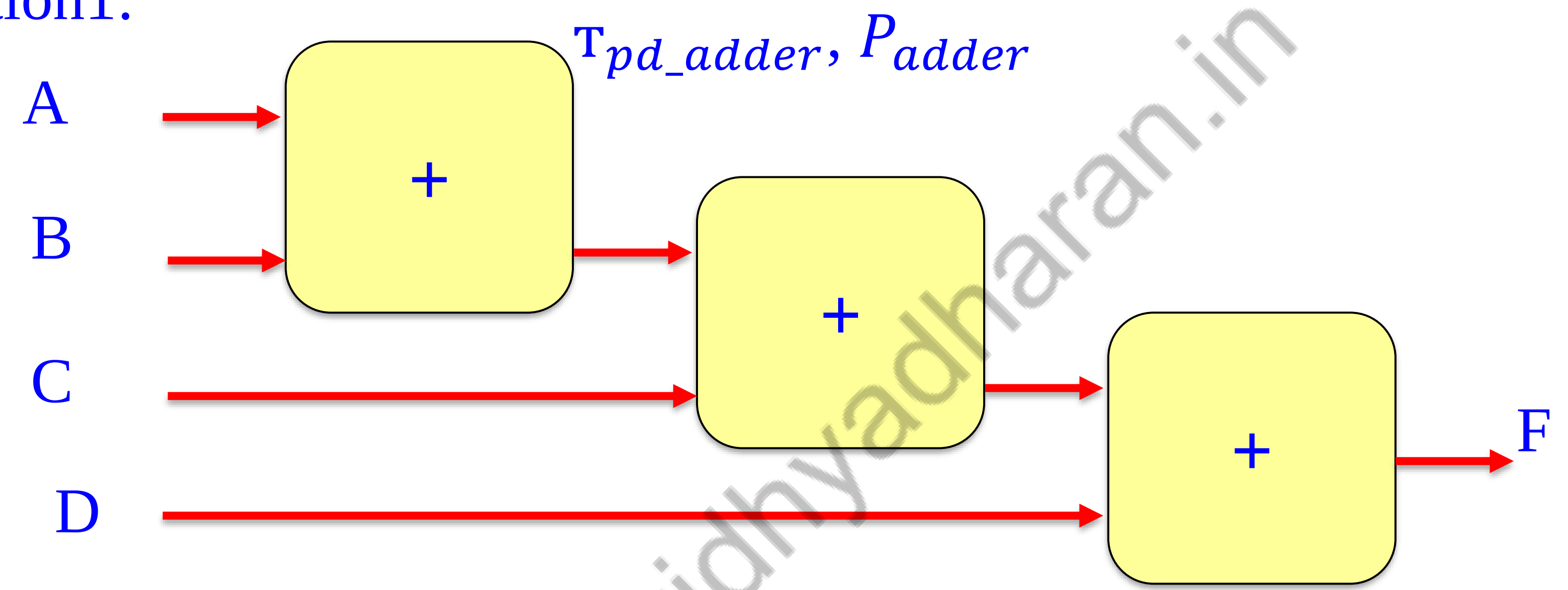


Computer Organization

Performance Figure	Value
Function Delay	$4 * T_{clock}$ $(T_{clock} > T_{pd\_adder} + T_{setup} + T_{clk-Q})$
Power	$4 * P_{adder} + 22 * P_{Regsiter}$
Throughput	@ T_{Clock}
Gate complexity	$4 * G_{adder} + 22 * G_{Register}$

Example 2

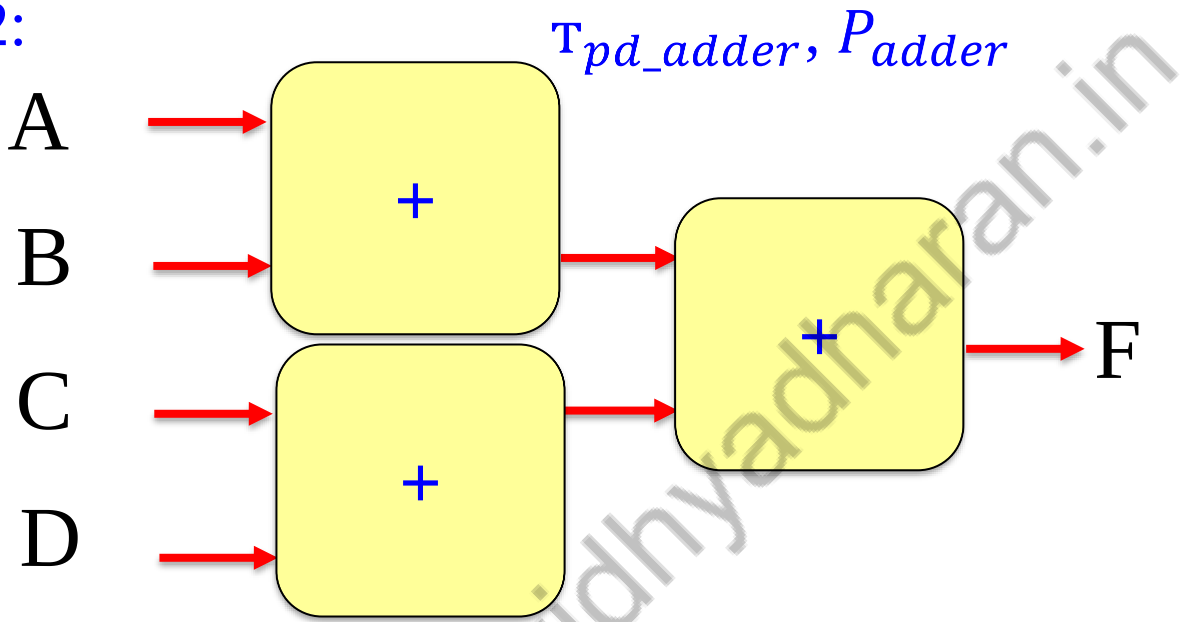
Option1:



Performance Figure	Value
Function Delay	$3 * \tau_{pd\_adder}$
Power	$3 * \ P_{adder}$
Throughput	@ $3 * \tau_{pd\_adder}$
Gate complexity	$3 * G_{adder}$
Functional Flexibility	Nil
Function Expandability	Nil

Example 2

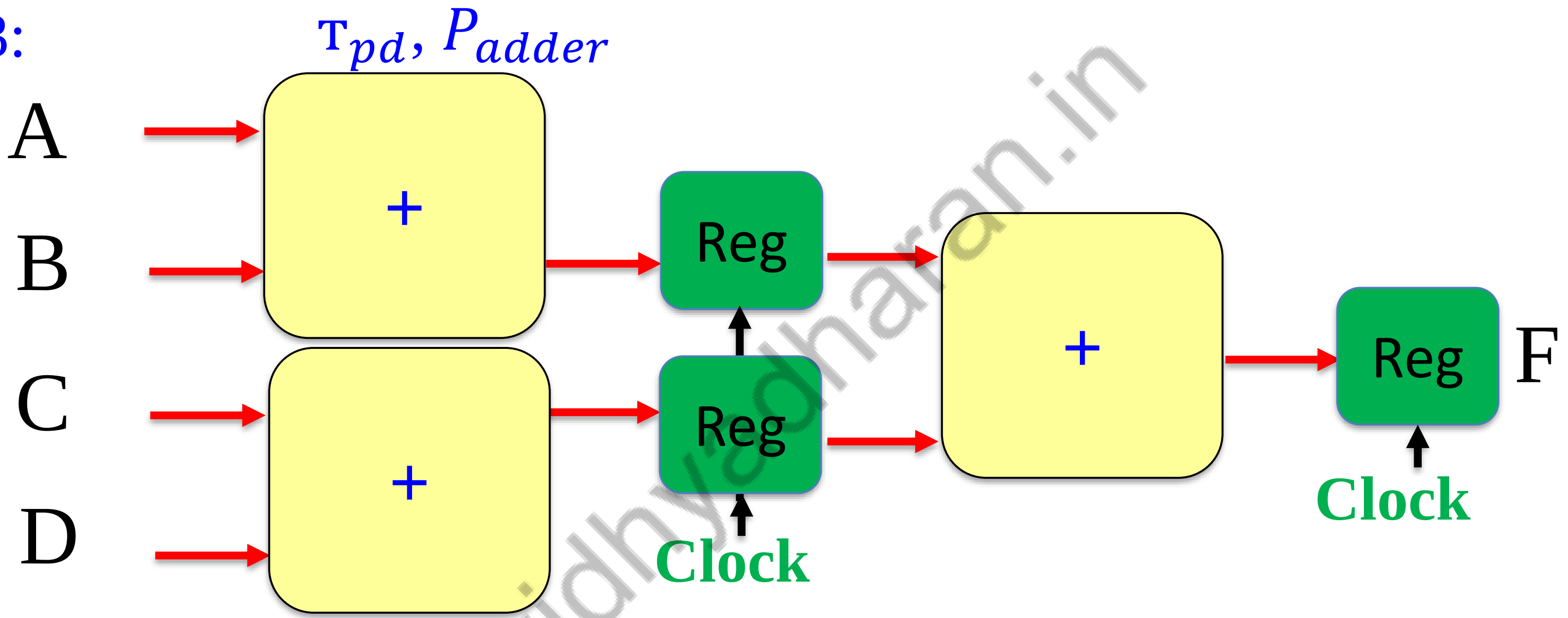
Option 2:



Performance Figure	Value
Function Delay	$2 * T_{pd\_adder}$
Power	$3 * P_{adder}$
Throughput	$@ 2 * T_{pd\_adder}$
Gate complexity	$3 * G_{adder}$
Functional Flexibility	Nil
Function Expandability	Nil

Example 2

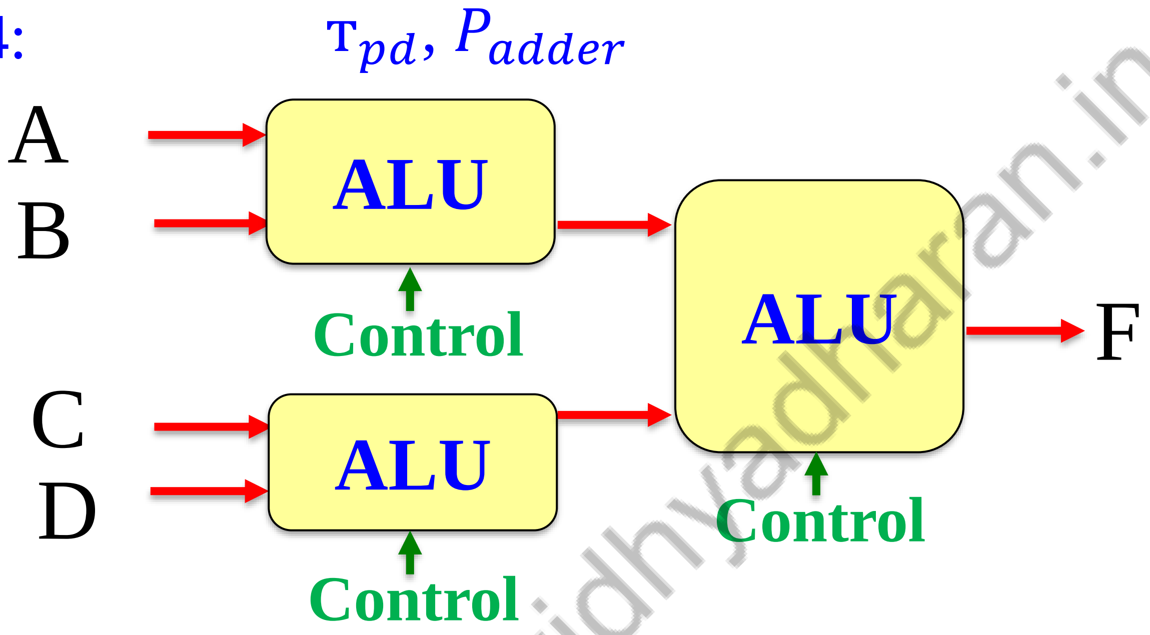
Option 3:



Performance Figure	Value
Function Delay	2*T$_{\text{Clock}}$
Power	$3 * P_{adder} + 3 * P_{Regsiter}$
Throughput	@ T$_{\text{Clock}}$ $(\tau_{clk-Q} + \tau_{pd\_adder} + \tau_{setup})$
Gate complexity	$3 * G_{adder} + 2 * G_{Register}$
Functional Flexibility	Nil
Function Expandability	Nil

VLSI Architecture Example 1

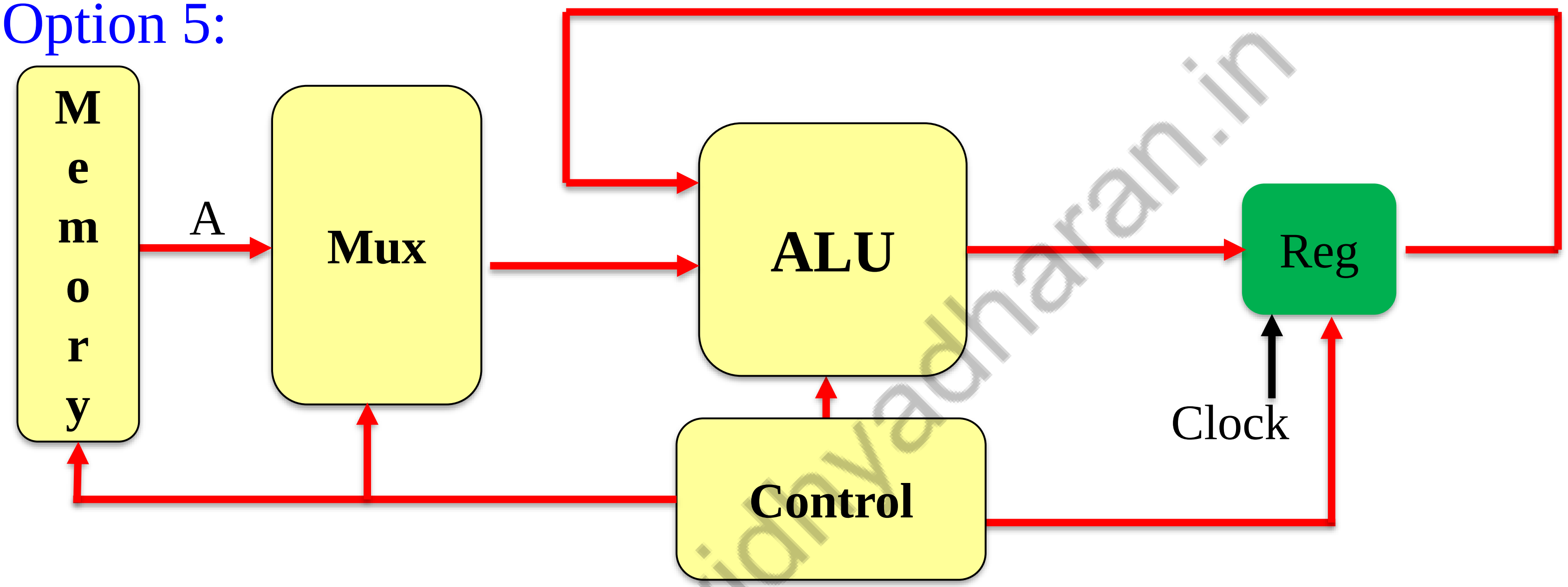
Option 4:



Performance Figure	Value
Function Delay	$2 * \tau_{ALU}$
Power	$3 * P_{ALU}$
Throughput	@ $2*\mathrm{T}_{\mathrm{ALU}}$
Gate complexity	$3 * G_{ALU}$
Functional Flexibility	Yes
Function Expandability	Nil

VLSI Architecture Example 1

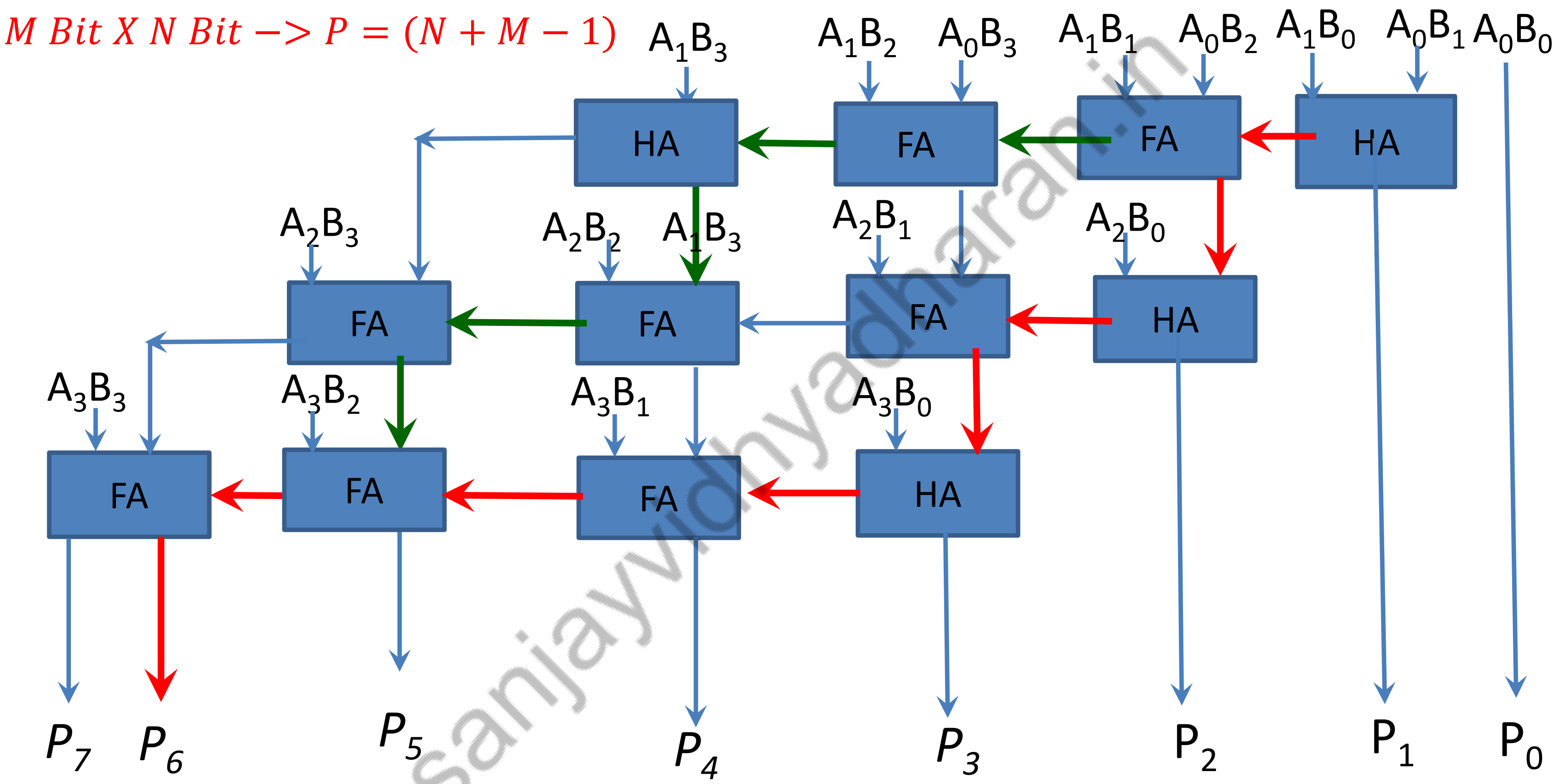
Option 5:



Performance Figure	Value
Function Delay	$\tau_{MUX} + T_{ALU} + \tau_{clk-Q}$
Power	$P_{MUX} + P_{ALU} + P_{Regsiter}$
Throughput	@ 4 * T_{CLK}
Gate complexity	$G_{MUX} + G_{ALU} + G_{Regsiter}$
Functional Flexibility	Yes
Function Expandability	Yes

The Array Multiplier

$M\ Bit\ X\ N\ Bit \rightarrow P = (N + M - 1)$

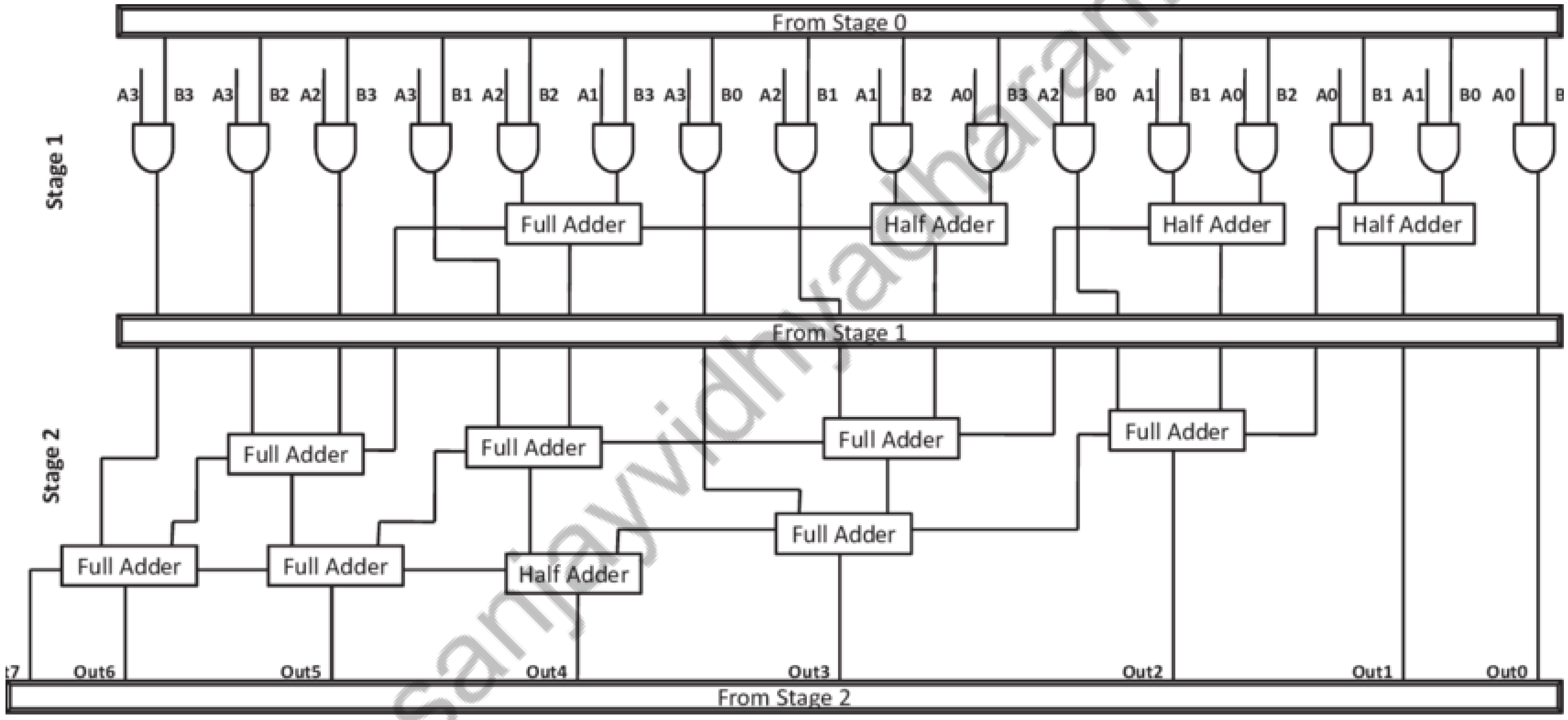


Require $(N-1)\ M - Bit\ Adders$

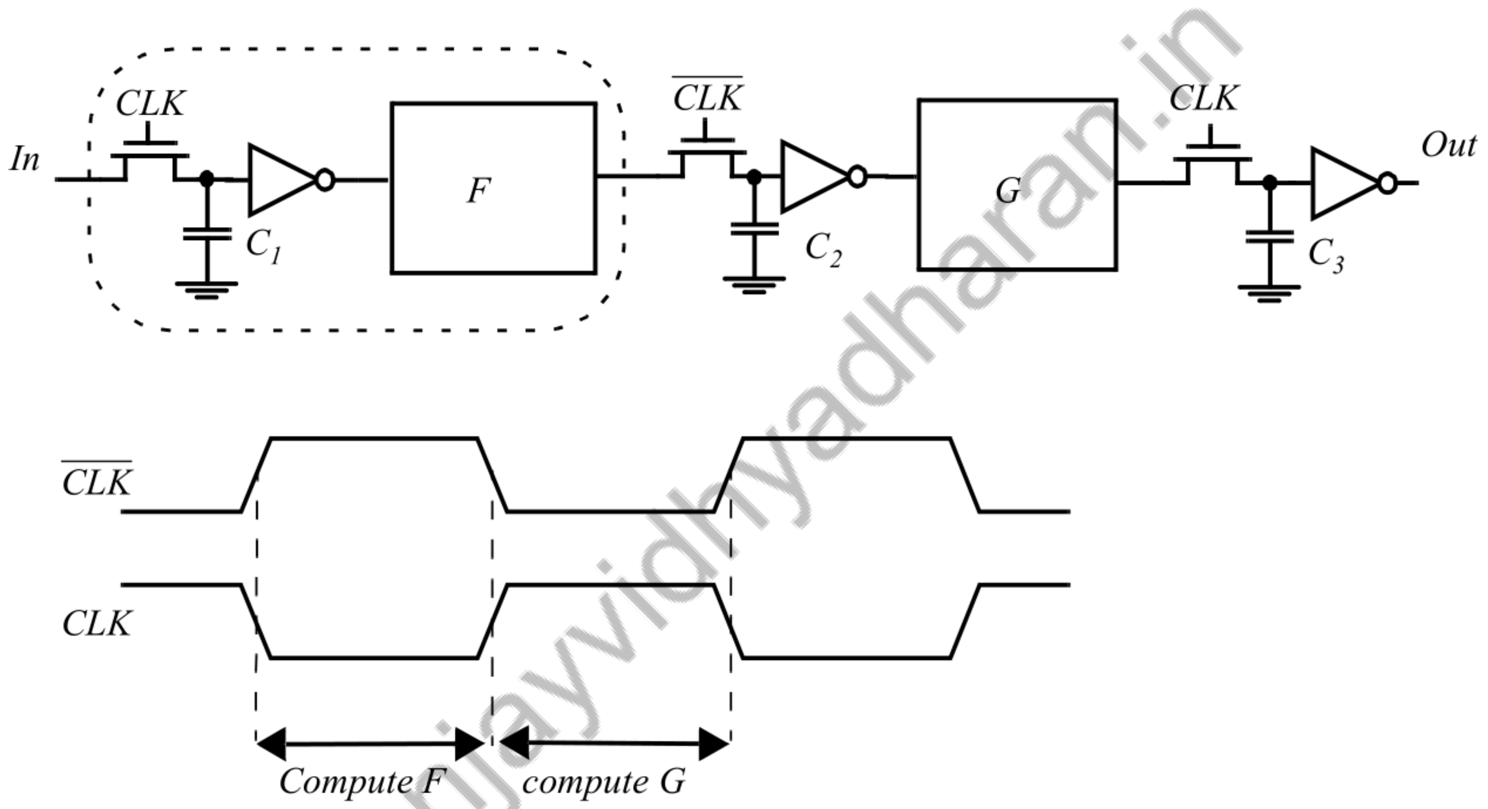
Require $M\ X\ N\ AND\ Gates$

$$t_{mul} = [(M-1) + (N-2)]t_{carry} + (N-1)t_{sum} + t_{and}$$

4-Bit Pipelined Multiplier



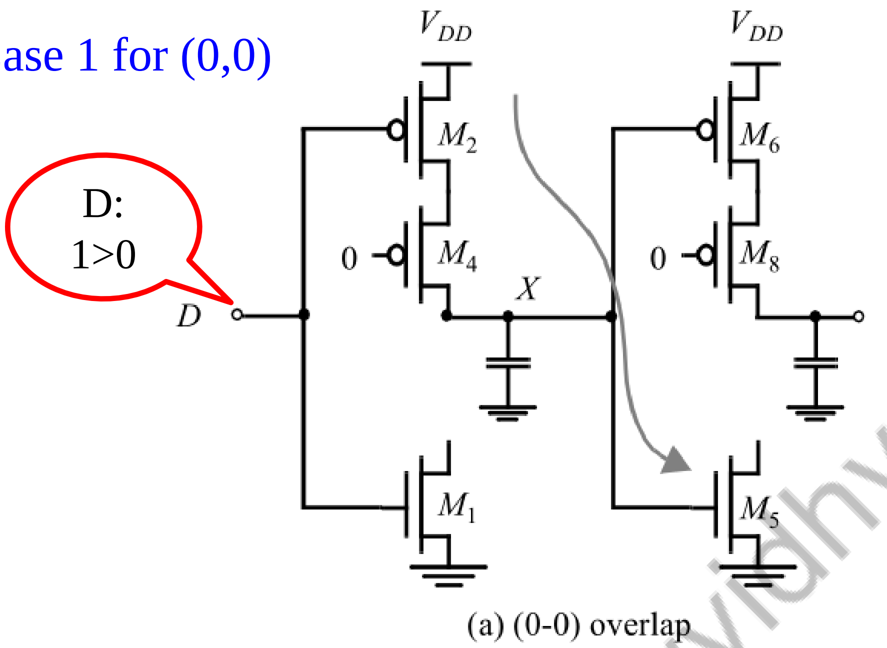
Pipelining with Latches



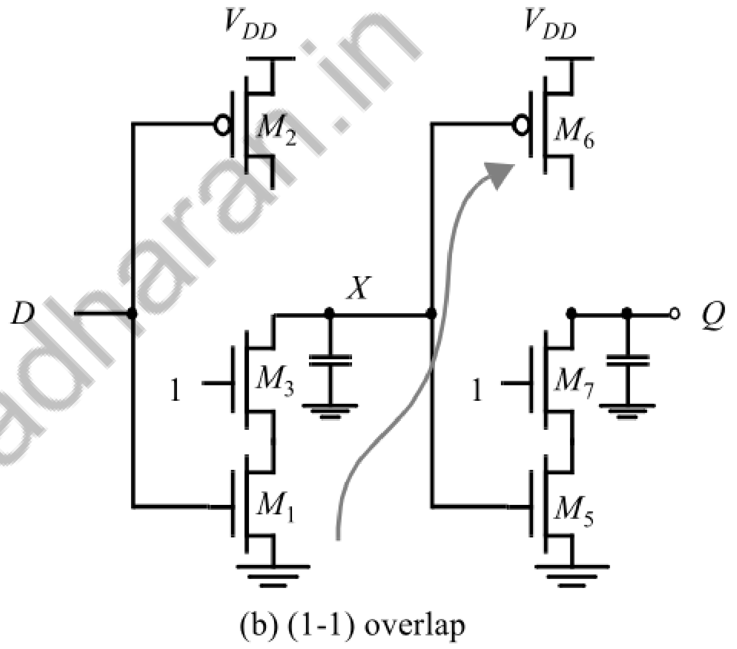
A non-overlapping clock essential for correct operation. Else there will be race around

Clock-Skew Insensitive C²MOS Register

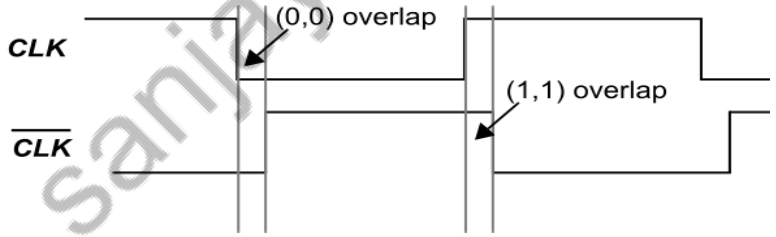
Case 1 for (0,0)



(a) (0-0) overlap

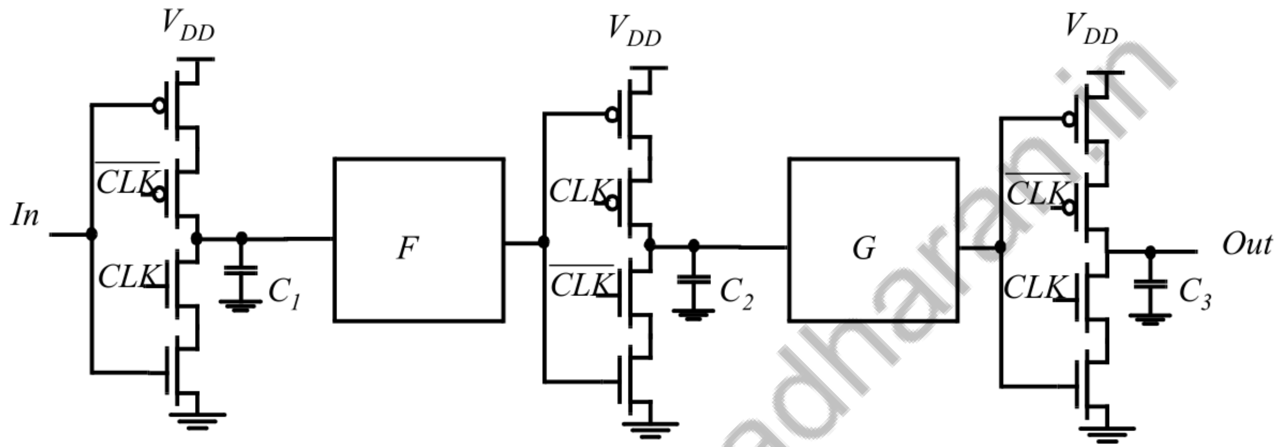


(b) (1-1) overlap

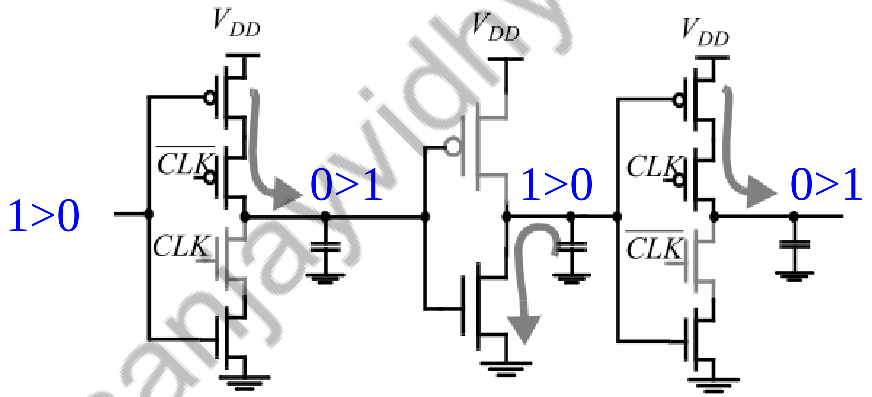


If the D input changes during the overlap period, node X can make a transition, but cannot propagate to the output.

Pipelined Logic using C^2MOS



Potential race condition during (0-0) overlap in C^2MOS-based design

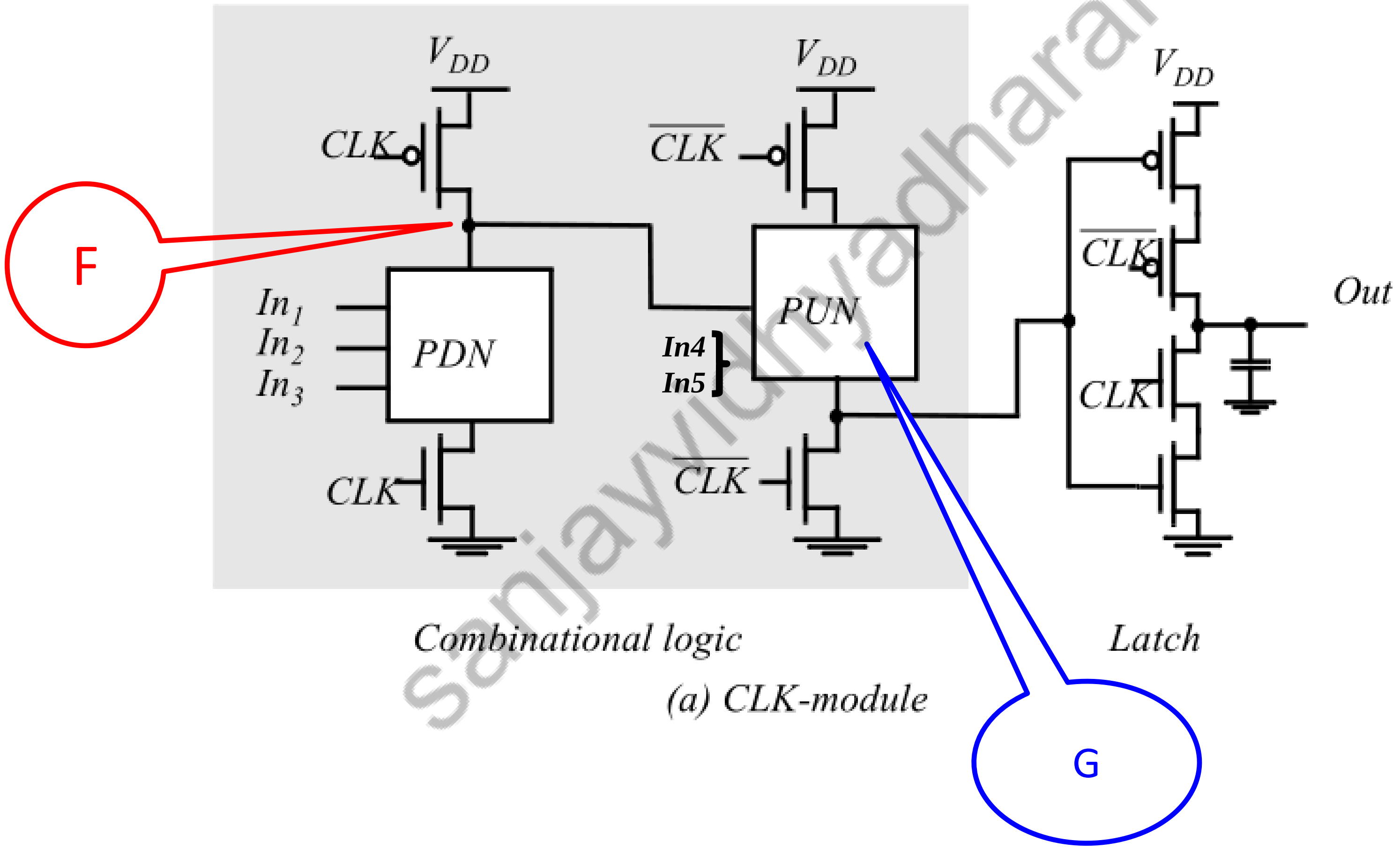


Similar considerations are valid for the (1-1) overlap.

A C^2MOS-based pipelined circuit is race-free as long as all the logic functions *F* (implemented using static logic) between the latches are non-inverting

NORA CMOS

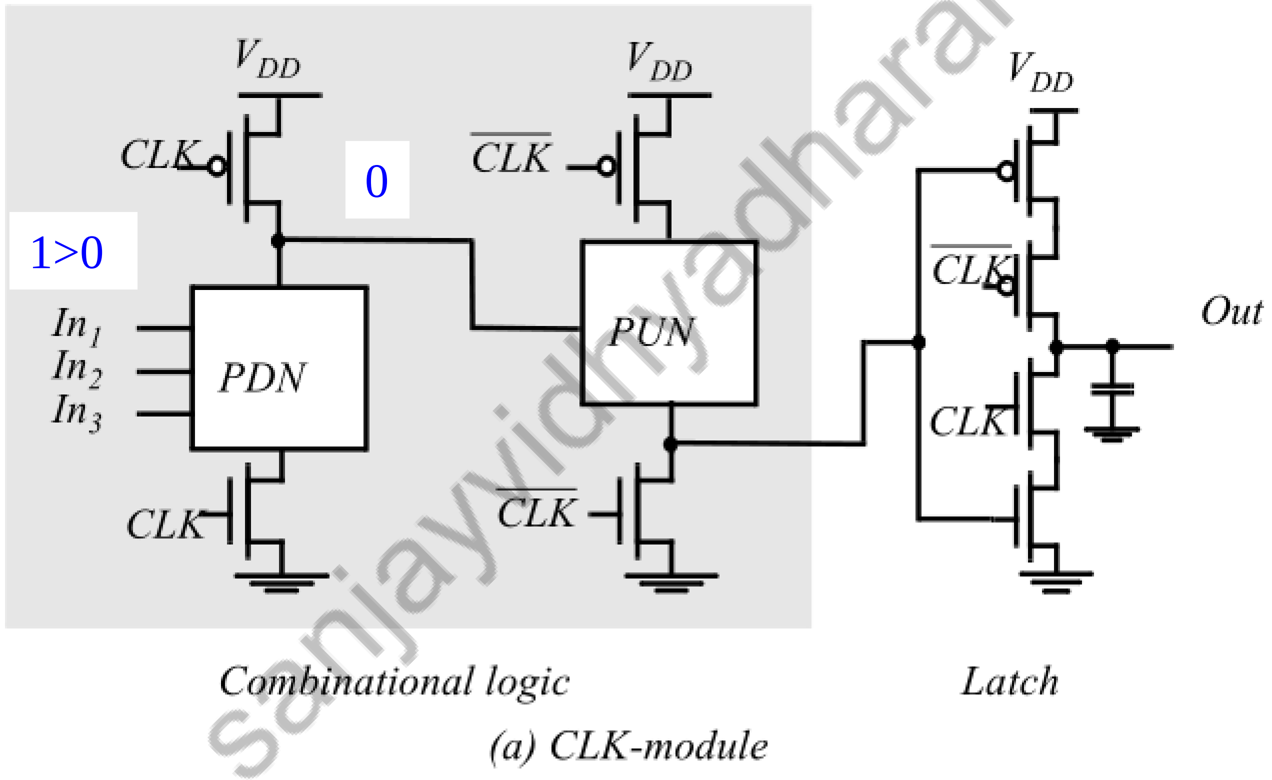
It combines C^2MOS pipeline registers and dynamic logic functional blocks.



(a) CLK-module

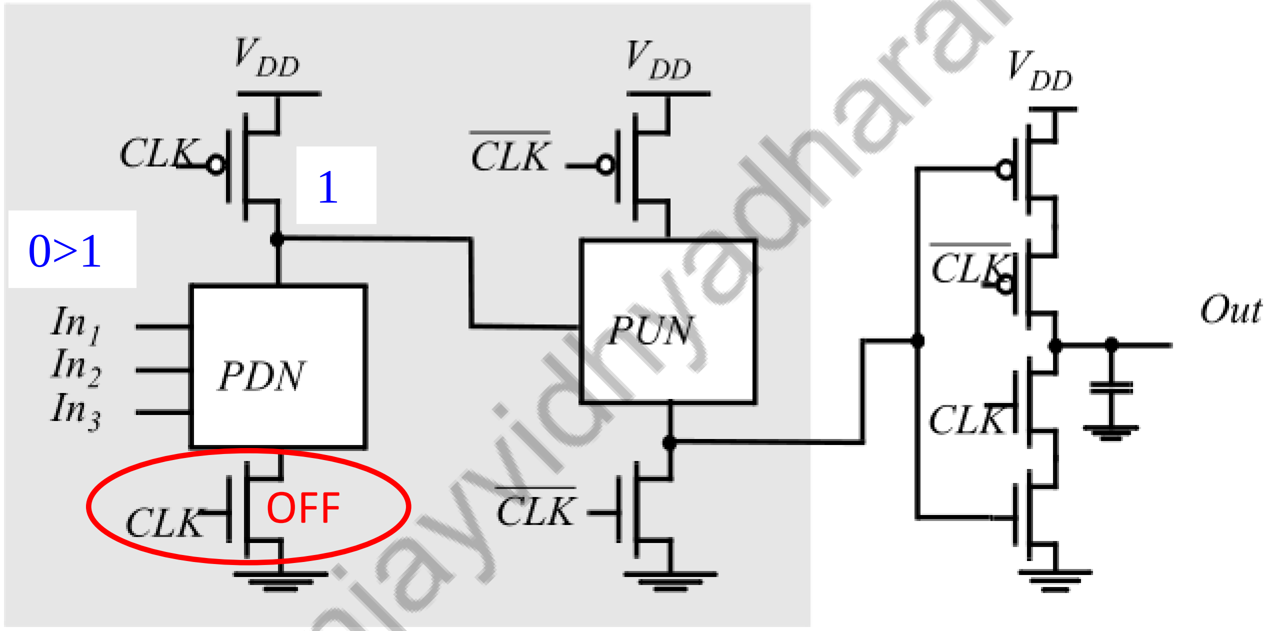
NORA CMOS

It combines C²MOS pipeline registers and dynamic logic functional blocks.



NORA CMOS

It combines C^{2}MOS pipeline registers and dynamic logic functional blocks.

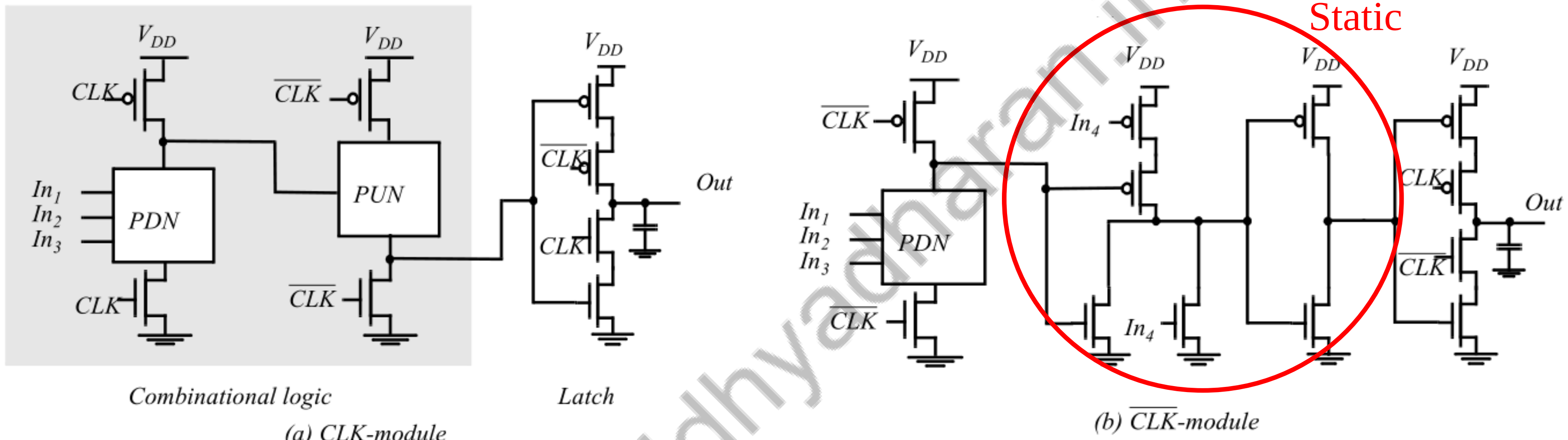


Combinational logic *Latch*

(a) CLK-module

NORA CMOS

It combines C^2MOS pipeline registers and dynamic logic functional blocks.



(a) CLK-module

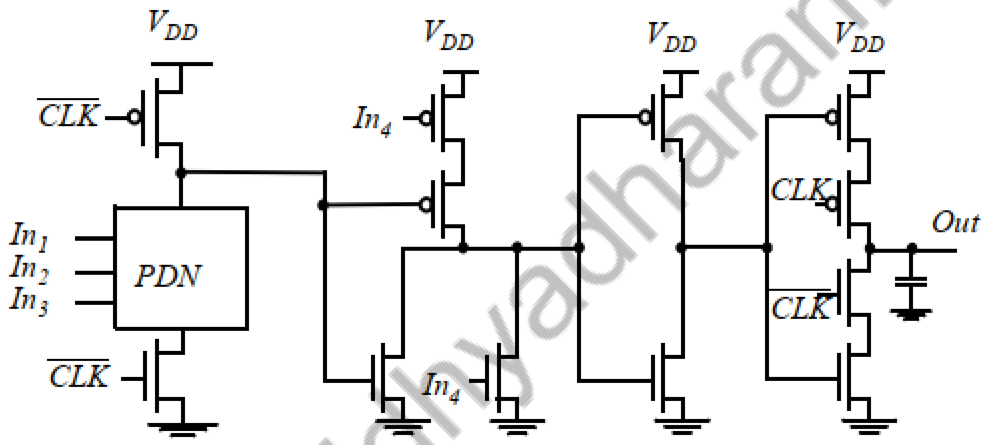
(b) $\overline{CLK}$-module

	Φ-block		$\overline{\Phi}$-block	
	Logic	Latch	Logic	Latch
Φ = 0	Precharge	Hold	Evaluate	Evaluate
Φ = 1	Evaluate	Evaluate	Precharge	Hold

NORA offers designers a wide range of design choices. Dynamic and static logic can be mixed freely. A NORA datapath consists of a chain of alternating CLK and CLK modules. While one class of modules is precharging with its output latch in hold mode, preserving the pre-vious output value, the other class is evaluating. Data is passed in a pipelined fashion from module to module.

NORA CMOS

Example of a NORA CLK' Module



In order to ensure correct operation, two important rules should always be followed:

• **The dynamic-logic rule:** Inputs to a dynamic CLK_n (CLK_p) block are only allowed to make a single $0 \rightarrow 1$ ($1 \rightarrow 0$) transition during the evaluation period.

• **The C$_2$MOS rule:** In order to avoid races, the number of static inversions between C$_2$MOS latches should be even.

Thank you