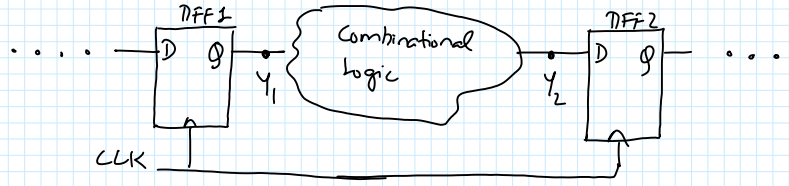
Lecture - 12

* Constraints / Restrictions on clock period / frequency and combinational delay

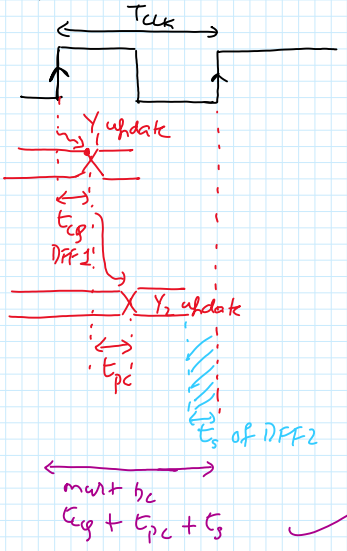
* Any design implemented will be a finite state machine (FSM) with inputs, registers, combinational logic and outputs.



* The combinational logic delay is input dependent and has a minimum & maximum delay based on the input conditions.
  ↳ $t_{pc(min)}$ and $t_{pc(max)}$

* DFF-1 and DFF-2 : t_s, t_h and t_{cq}.

* To avoid any setup violations:-

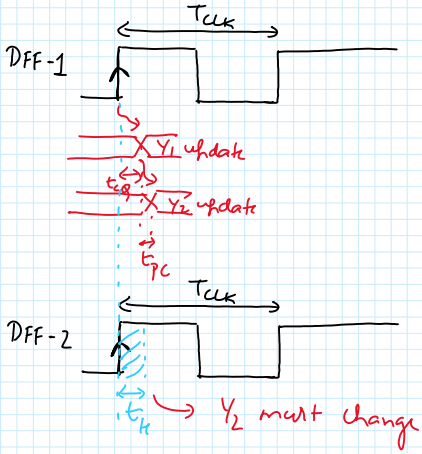


$$\Rightarrow T_{CLK} > t_{cq} + t_{pc(max)} + t_s$$

↓

Setup-time violation puts restriction on max clock frequency

* To avoid any hold violations -



↳ Y_2 must change after t_h to avoid race conditions (Non-logic voltages)

t_H → Y_2 must change after t_H to avoid race conditions (Non-logic voltages)

$$\therefore \quad t_H < t_{cq} + t_{PC(min)}$$

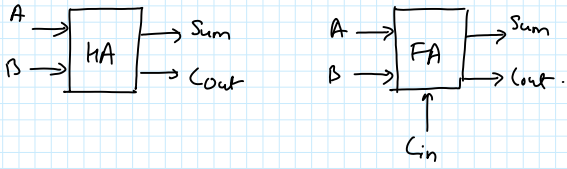
$$\Rightarrow \boxed{t_{PC(min)} > t_H - t_{cq}}$$

↓

Hold-time violation puts limit on minimum combinational delay

---

* Two important building blocks of any digital IC is an "Adder" and a "Multiplier".
* Used in ALU in a processor, FIR filters (DSP), MAC in ML engines etc.
* The basic standard cells required to build an adder and multiplier are
  - Half adder (HA)
  - Full adder (FA)



* Half adder (HA)

A	B	Sum	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

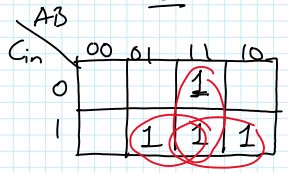
$Sum = \bar{A}B + A\bar{B}$ → Either static MUX or static CMOS logic

$Cout = AB$ → NAND + NOT

* Full adder

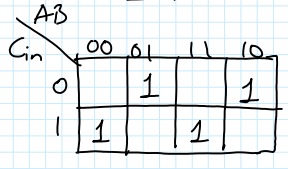
A	B	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Cout



$Cout = AB + BCin + CinA$

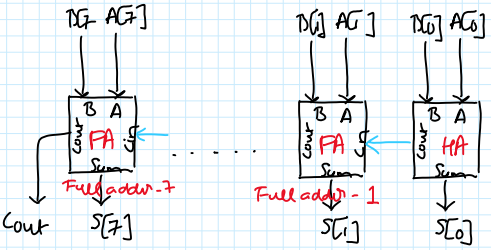
Sum



$Sum = \bar{A}\bar{B}Cin + \bar{A}B\bar{Cin} + A\bar{B}\bar{Cin} + ABCin$

$$Sum = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C_{in}} + A\bar{B}\bar{C_{in}} + ABC_{in}$$

* Let's say we implement an 8-bit adder. (A[7:0] added with B[7:0])



* In the worst-case scenario, carry out will propagate from HA to FA-1 → FA2 . . → FA7

⇒ The critical path (worst-case delay) will be because of "Cout" logic and not "Sum" logic

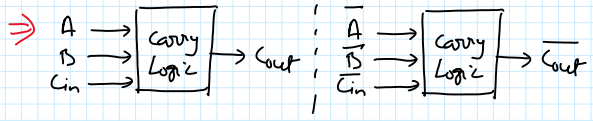
* : What if we generate "Cout" first and generate "Sum" from "Cout": ?

↳ This will mean while Cout is propagating, Sum will be simultaneously generated

* Consider,

$$C_{out} = AB + BC_{in} + C_{in}A$$

$$\overline{C_{out}} = \bar{A}\bar{B} + \bar{B}\bar{C_{in}} + \bar{C_{in}}\bar{A}$$



* Speciality of logic !!! If $A\bar{B}$, C_{in} are passed as input, O/p = Cout
If $\bar{A}$, $\bar{B}$, $\bar{C_{in}}$ are passed as input, O/p = $\overline{C_{out}}$.

* Now Consider $\overline{C_{out}}(A+B+C_{in}) = (\bar{A}\bar{B} + \bar{B}\bar{C_{in}} + \bar{C_{in}}\bar{A})(A+B+C_{in})$

$$= \bar{A}\bar{B}C_{in} + A\bar{B}\bar{C_{in}} + \bar{A}B\bar{C_{in}}$$

$$\Rightarrow Sum = \overline{C_{out}}(A+B+C_{in}) + ABC_{in}.$$

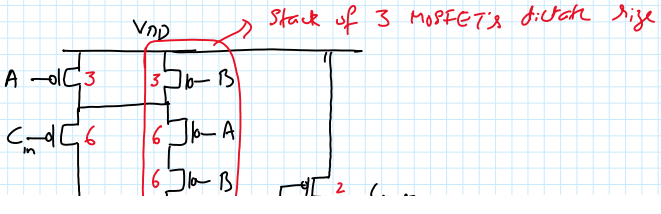
↓

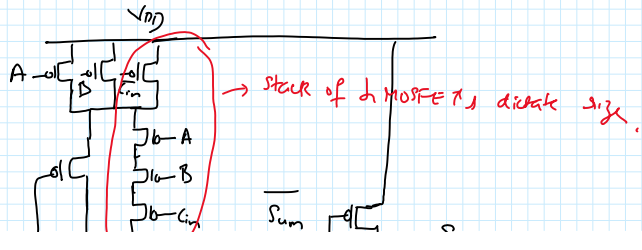
Can generate Sum from Cout ⇒ Cout will go to next stage first while Sum of present stage is getting computed !!

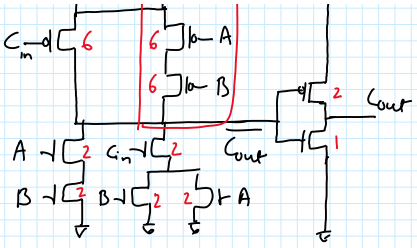
* Static CMOS implementation:

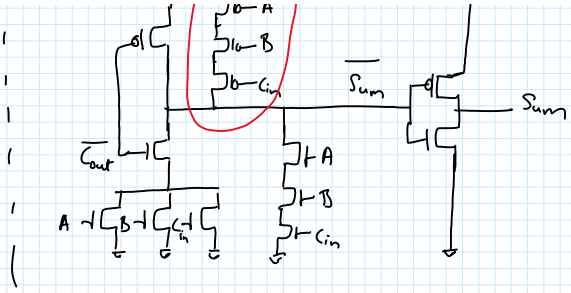
$$C_{out} = AB + BC_{in} + C_{in}A = AB + C_{in}(A+B)$$

$$Sum = \overline{C_{out}}(A+B+C_{in}) + ABC_{in}$$









* Recall!-

$$C_{out} = AB + BC_{in} + C_{in}A$$
$$\overline{C_{out}} = \bar{A}\bar{B} + \bar{B}\bar{C_{in}} + \bar{C_{in}}\bar{A} \quad \} \ (1)$$

* It so happens, this is also true for "Sum" logic

i.e $\quad Sum = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C_{in}} + A\bar{B}\bar{C_{in}} + ABC_{in}$

& $\quad \overline{Sum} = AB\bar{C_{in}} + A\bar{B}\bar{C_{in}} + \bar{A}BC_{in} + \bar{A}\bar{B}\bar{C_{in}}$ $\} \ (2)$

* (1) & (2) Indicate that the "Sum" and "Carry" functions are their own complements!!

* In CMOS we interchange series & parallel configurations (on NMOS & PMOS side)
  ↳ This ensures that pull-up (PMOS) & pull-down (NMOS) networks are complementary!
  ⇒ For NMOS and PMOS we can use the same configuration on PMOS and NMOS side.