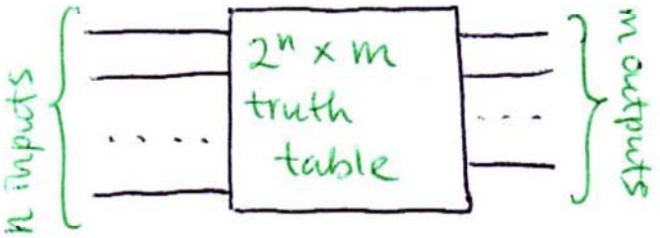
P3360/AEP3630

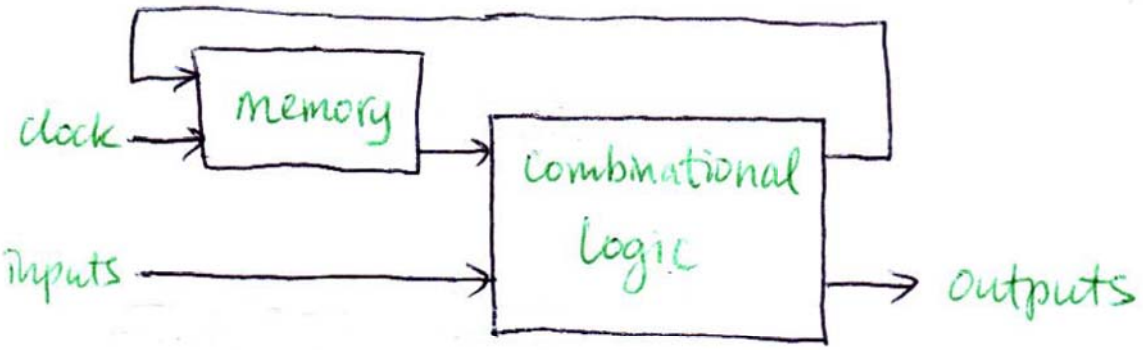
Lecture 27

Combinational Logic



- outputs depend only on present values of inputs

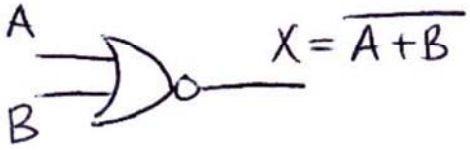
Sequential Circuit (simple example)



- outputs depend on both present and states of memory (past) elements
- clock determines when info stored in memory can change

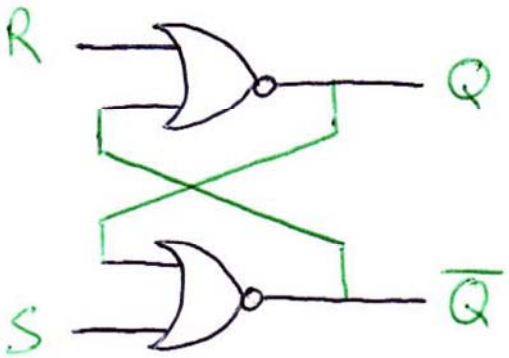
RS latch

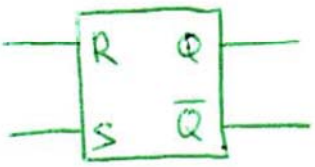
recall NOR gate

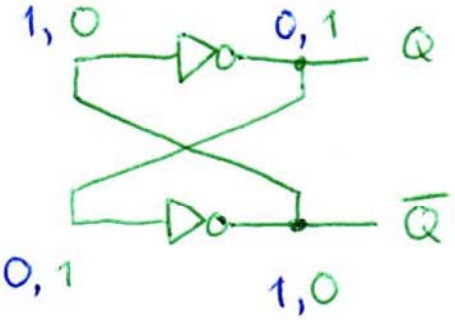


if $A = 0$, $X = \overline{0+B} = \overline{B}$,

$\Rightarrow$ NOR becomes







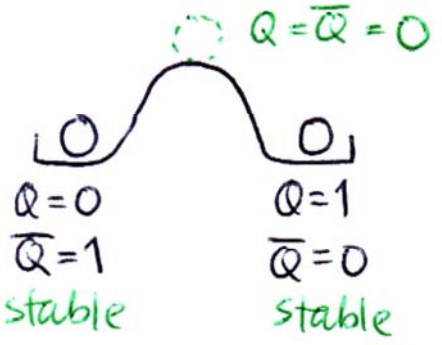
	S	R	Q	$\overline{Q}$	
	0	0	1	0	} two stable states
	0	0	0	1	} two stable states
step 1)	0	1	0	1	reset
2)	0	0	0	1	after S=0, R=1
	1	0	1	0	set
	0	0	1	0	after S=1, R=0
	1	1	0	0	← contrary to the names ($Q = \overline{Q}$?)
	0	0	?	?	

summary

S	R	Q	$\overline{Q}$
0	0	no change	
0	1	0	1
1	0	1	0
1	1	not allowed	

- latch has asynchronous response, i.e. output changes whenever the input changes
- contrary to "flip-flop", which requires a clock.

Two stable states



- similar to ball analogy, need a strong enough force to send the latch from one stable state into another
- there exists a min pulse length needed for switching (typically ~10's ns)

Flip-flop (i.e. clocked or synchronized clock)

problems with latches : asynchronous circuits hard to combine / integrate

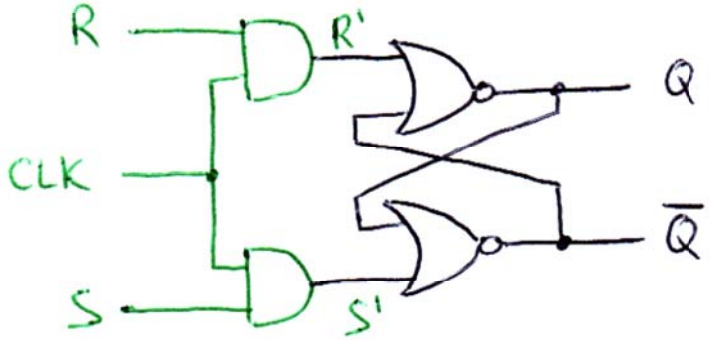
E.g. sequence of operations depends on exact values of prop. delays between gates

Synchronous circuits (= clocked) are usually preffered

Inputs/outputs change @ well-defined times

Improvements

1) clocked RS flip-flop

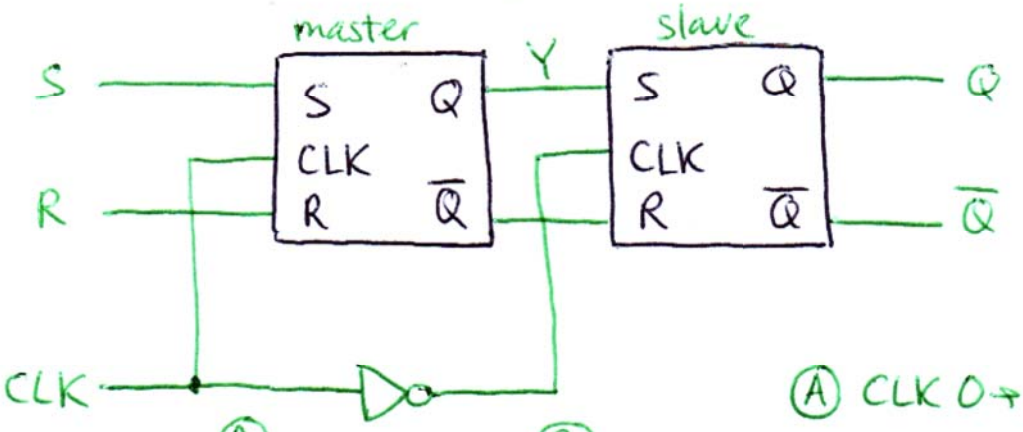


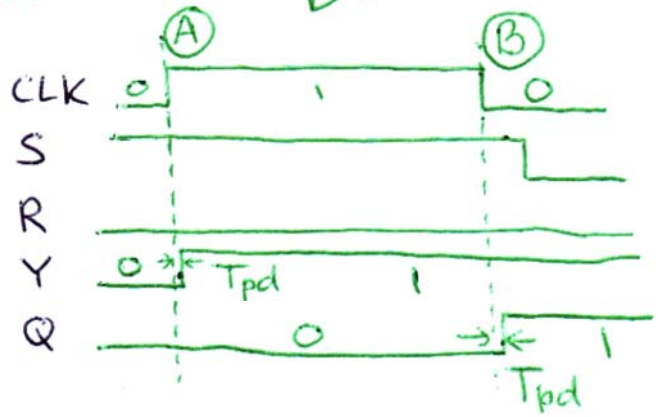
State can change only when CLK = 1

2) triggering : level-sensitive or transition-sensitive (better)

CLK (EN) = 1 latch "transparent" when CLK ↓ or ↑

Master-slave configuration



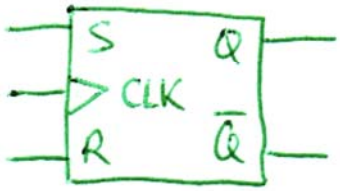


(A) CLK 0→1 master enabled & slave disabled

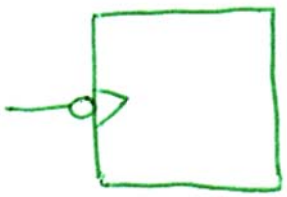
(B) CLK 1→0 master disabled & slave enabled

- Q ends up controlled by S, R when CLK ↓ only
- switch for multiple FF's can occur at the same time

Symbol

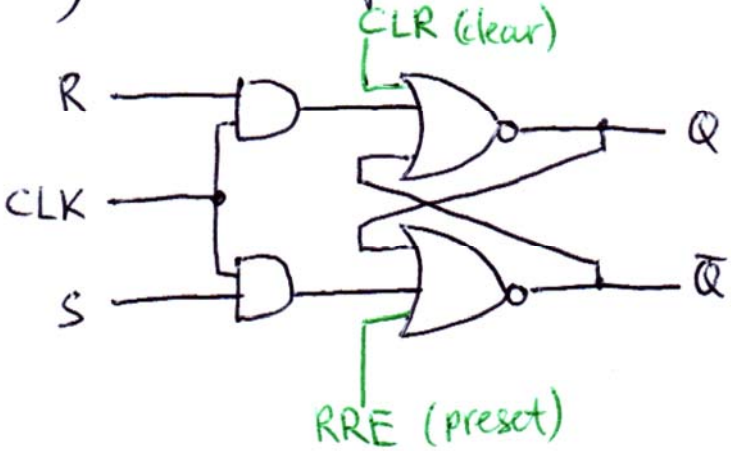


"+" edge ↑ triggered



"−" edge ↓ triggered

3) direct inputs : used to set FF state asynchronously (= force a state)



PRE → 1 : Q → 1
CLR → 1 : Q → 0
} regardless of CLK, S, R