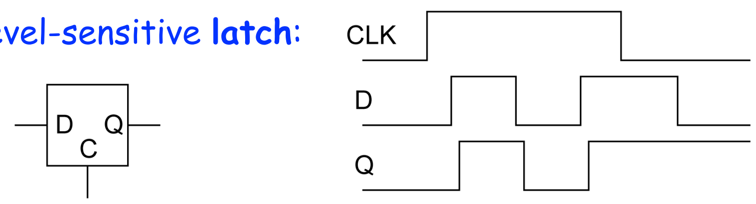
EECS150 - Digital Design
Lecture 25 - Latches & Flip-flops

April 23, 2012

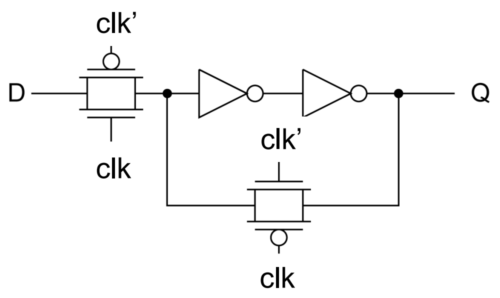
John Wawrzynek

CMOS "Transparent" Latches

Positive level-sensitive **latch**:

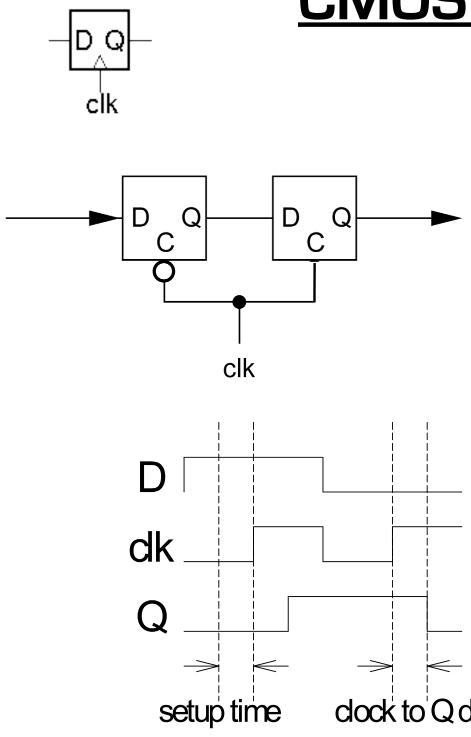


Latch Implementation:

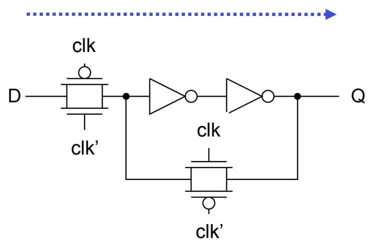


Designing with Latches

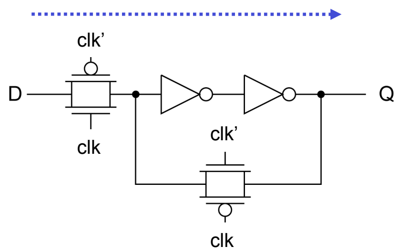
CMOS Flip-flops



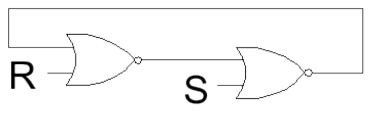
- Setup time results from delay through *first* latch.



Clock to Q delay results from delay through *second* latch.



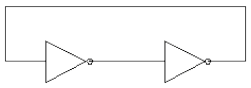
Cross-coupled NOR gates



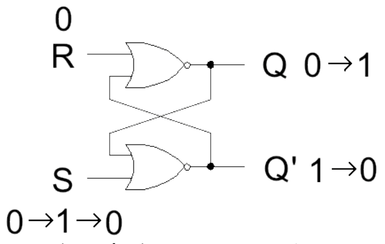
remember,

NOR	
00	1
01	0
10	0
11	0

- If both R=0 & S=0, then cross-couped NORs equivalent to a stable latch:



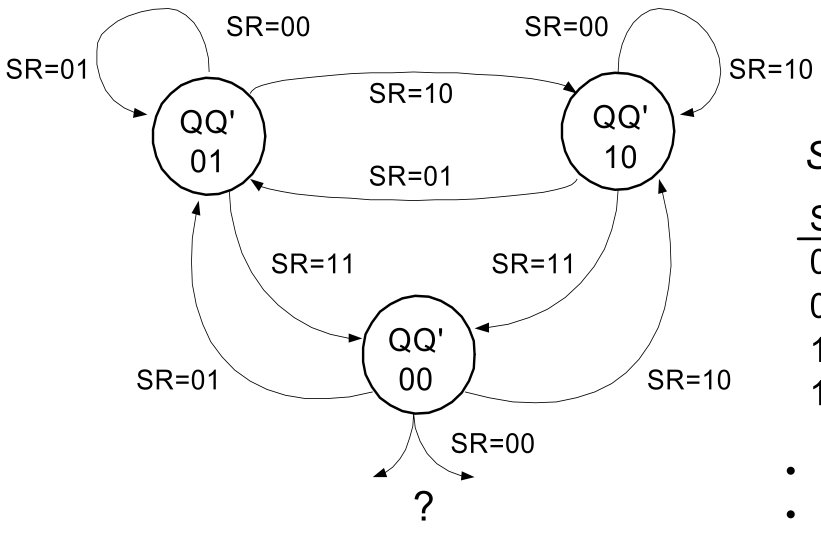
- If either R or S becomes =1 then state may change:

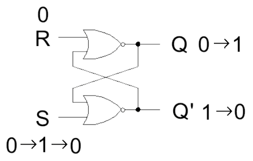


- What happens if R or S or both become = 1?

Asynchronous State Transition Diagram

Transitions triggered by input changes.





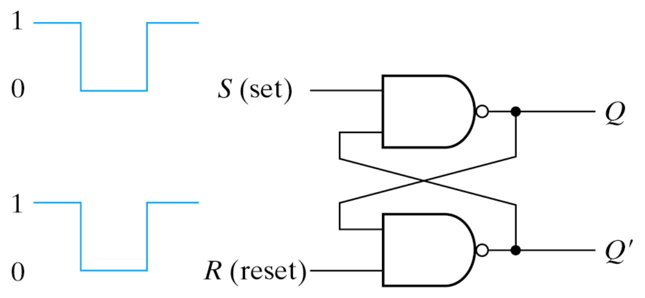
SR Latch:

SR	Q
00	hold
01	0
10	1
11	indeterminate

- S is “set” input
- R is “reset” input

QQ’=00 is often called a “forbidden state”

Nand-gate based SR latch



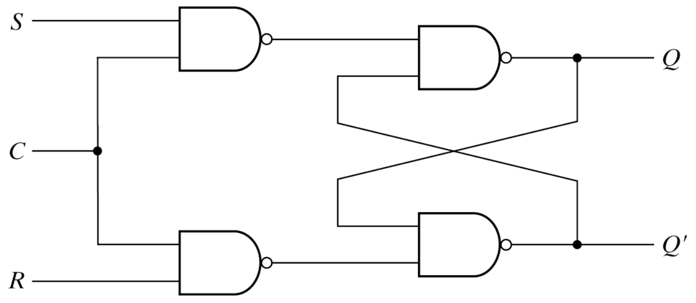
S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(after $S = 1, R = 0$)
0	1	1	0	
1	1	1	0	(after $S = 0, R = 1$)
0	0	1	1	

(a) Logic diagram (b) Function table

Fig. 5-4 *SR* Latch with NAND Gates

- Same behavior as cross-coupled NORs with inverted inputs.

Level-sensitive SR Latch



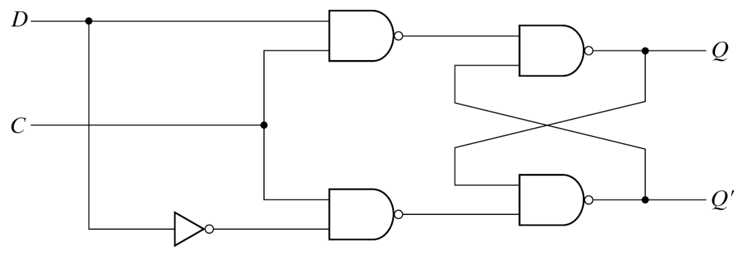
C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	$Q = 0$; Reset state
1	1	0	$Q = 1$; set state
1	1	1	Indeterminate

(a) Logic diagram (b) Function table

Fig. 5-5 SR Latch with Control Input

- The input “C” works as an “enable” signal, latch only changes output when C is high.
- Usually connected to **clock**.

D-latch



(a) Logic diagram

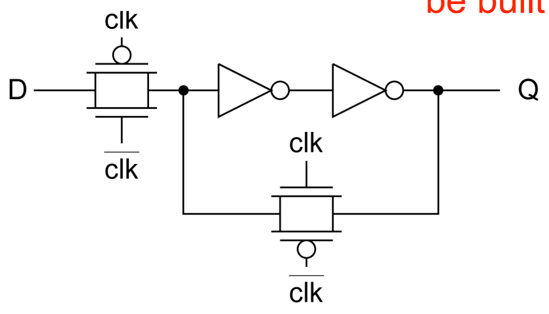
C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

(b) Function table

Fig. 5-6 D Latch

Compare to transistor version:

All state elements *could* be built using logic gates.



2 Alternative Flip-flops

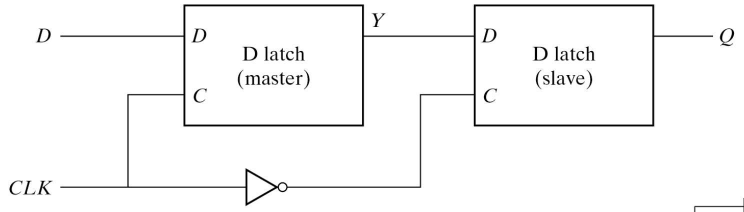


Fig. 5-9 Master-Slave *D* Flip-Flop

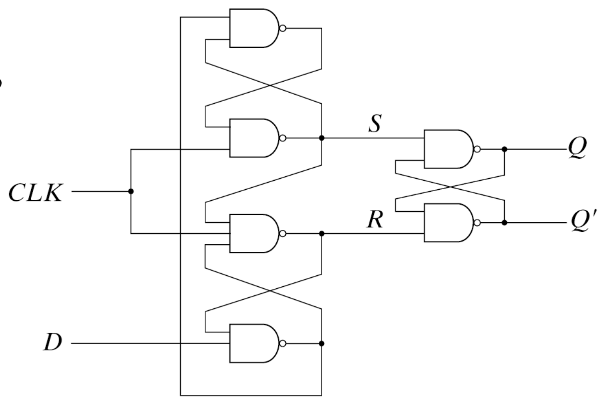
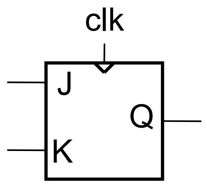


Fig. 5-10 *D*-Type Positive-Edge-Triggered Flip-Flop

J-K FF

- Add logic to eliminate "indeterminate" action of RS FF.
- New action is "toggle"
- J = "jam"
- K = "kill"



J	K	Q(t)	Q(t+Δ)	
0	0	0	0	hold
0	0	1	1	
0	1	0	0	reset
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	1	toggle
1	1	1	0	

Storage Element Taxonomy

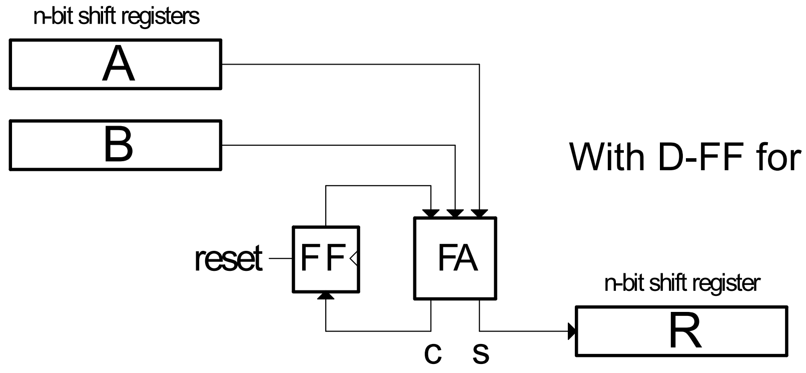
	synchronous		asynchronous
	level-sensitive	edge-triggered	
D-type	★	✓	n.a.
JK-type	✓	✓	n.a.
RS-type	✓	✓	★
	"latch"	"flip-flop"	"latch"

★ "natural" form

✓ "possible" form

Design Example with RS FF

- With D-type FF state elements, new state is computed based on inputs & present state bits - reloaded each cycle.
- With RS (or JK) FF state elements, inputs are used to determine conditions under which to set or reset state bits.
- Example: bit-serial adder (LSB first)



With D-FF for carry

Bit-serial adder with RS FF

- RS FF stores the carry:

a b c_i	c_{i+1}	s	
0 0 0	0	0	Carry kill a'b'
0 0 1	0	1	
0 1 0	0	1	
0 1 1	1	0	
1 0 0	0	1	
1 0 1	1	0	
1 1 0	1	0	Carry generate ab
1 1 1	1	1