

Jim Siegrist
Phone: 486-4397
Email: JLSiegrist@lbl.gov
Room (at LBL): 50-4055

Advice:

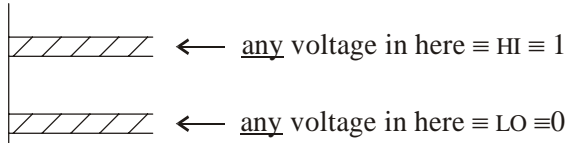
Today: lec 11 Digital II
 lec 12 TUE Apr 3 Digital III

Posted solutions, lab 3 & 4
Final project proposals due April 7 – see TA's (I sign)
Problems 8.12, 8.13 → supplementary.

Concepts

Digital Circuits

Digital Signals are required to be within one of 2 ranges – ‘HI’ & ‘LO’. What the range is depends on the type of logic – more later.



‘It’s all just 1’s & 0’s’

bit \equiv binary digit

Rate at which bits can be transferred by a digital system determines its speed (\equiv baud = bit/sec)
(gigabit easily achievable)

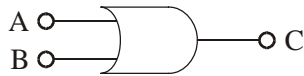
Other voltage levels appear during switching, and sometimes when outputs are not in use, but such levels are undefined.

Logic Building Blocks

- combinational logic
several inputs, one output
output at a given time depends on inputs at same time
- sequential logic
output depends on what inputs are now & at earlier times (flip-flop)

Combinational Logic Examples

OR gate:

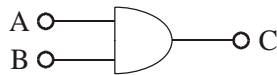


Truth Table →

Inputs		Output
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

Output is HI if & only if one or more of the inputs is high. (3 input truth table would have 8 entries)

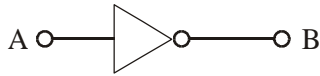
AND gate:



Inputs		Output
A	B	C
0	0	0
1	0	0
0	1	0
1	1	1

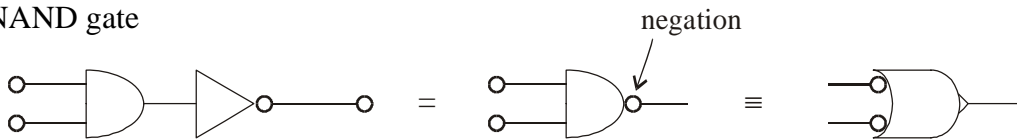
Output HI if & only if all inputs are HI

Inverter (Complement):

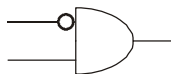


Input	Output
A	B
0	1
1	0

⇒ NAND gate

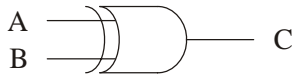


Also see circles on input sometimes:



Inputs		Output
A	B	C
0	0	1
1	0	1
0	1	1
1	1	0

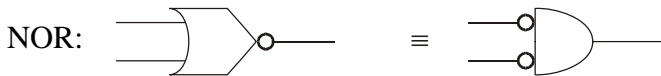
Exclusive OR



A	B	C
0	0	0
1	0	1
0	1	1
1	1	0

$$\begin{aligned} & \overline{\overline{A+B} + A \cdot B} \\ &= (A+B) \cdot (\overline{A \cdot B}) \\ &= (A+B) \cdot (\overline{A} + \overline{B}) \\ &= B \cdot \overline{A} + A \cdot \overline{B} \end{aligned}$$

output true if one and only one input is true



Logic Notation & Boolean Algebra

Complement	\overline{A}
OR	$A + B$
AND	$A \cdot B$

Two logical expressions are equivalent if their truth tables are the same.

$$\begin{aligned} 0 + 0 &= 0 & 0 \cdot 0 &= 0 \\ 0 + 1 &= 1 + 0 = 1 & 0 \cdot 1 &= 1 \cdot 0 = 0 \\ 1 + 1 &= 1 & 1 \cdot 1 &= 1 \\ \overline{0} &= 1, \overline{1} &= 0 \end{aligned}$$

De Morgan's Theorem

$$\left. \begin{aligned} \overline{(X + Y)} &= \overline{X} \cdot \overline{Y} \\ \overline{X \cdot Y} &= \overline{X} + \overline{Y} \end{aligned} \right\} \text{prove by comparison of truth tables}$$

e.g.

X	Y	X + Y	$\overline{X + Y}$	\overline{X}	\overline{Y}	$\overline{X} \cdot \overline{Y}$
0	0	0	1	1	1	1
1	0	1	0	0	1	0
0	1	1	0	1	0	0
1	1	1	0	0	0	0

Note distributed law also satisfied:

$$A \cdot (B + C) = A \cdot B + A \cdot C$$

e.g., mathematical algebra

A	B	C	$B + C$	$A \cdot (B + C)$	$A \cdot B$	$A \cdot C$	$A \cdot B + A \cdot C$
0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0
0	0	1	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	1	0	1	1	1	0	1
1	0	1	1	1	0	1	1
1	1	1	1	1	1	1	1

Logic Design

Achieve a specified truth table with least possible number of gates. Use Boolean algebra rules to simplify logic expressions.

Coding & Decoding – binary number representation

How do I represent a decimal number in binary?

Set of m binary number has up to 2^m distinct values. 10 possible values for a decimal digit $\Rightarrow 2^4 = 16 > 10$ bits required (8 bits = 1 byte; 4 bits = nibble)

Frequently, write bytes in Hex. 0–9ABCDEF

Concepts

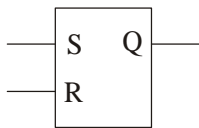
Sequential Logic (flip-flops)

Most common sequential logic circuit – bistable multivibrator – exists in (= flip-flop) one of 2 stable states (HI or LOW)

(other multivibrators – monostable – returns to same state
astable – oscillator)

Flip-Flop types: S-R, J-K, D

S-R Flop (set, reset)

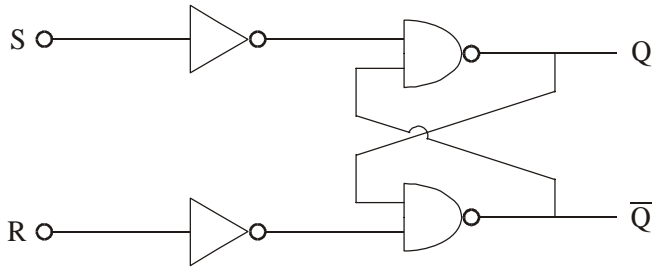
		S	R	Q_n	Q_{n+1}
	keep Q value {	0	0	0	0
		0	0	1	1
	reset {	0	1	0	0
		0	1	1	0
	set {	1	0	0	1
		1	0	1	1
	don't use, meaningless \rightarrow	1	1		

S, R both low \Rightarrow block 'remembers' Q

To set Q, use S = 1, R = 0; to 'reset' use S = 0, R = 1

Circuit Analysis

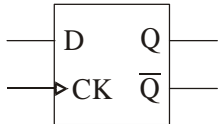
Circuit Realization:



S	R	Q_n	Q_{n+1}	
0	0	0	0	} same state
0	0	1	1	
1	0	0	1	} set
1	0	1	1	
0	1	0	0	} reset
0	1	1	0	

Concepts

D flip-flop



Flip-flop only changes state when a 'change' instruction is given (clock 'CK' input).

CK changes from 0 – 1 (makes a positive transition)

$\Rightarrow Q = D$ (Q set to D)

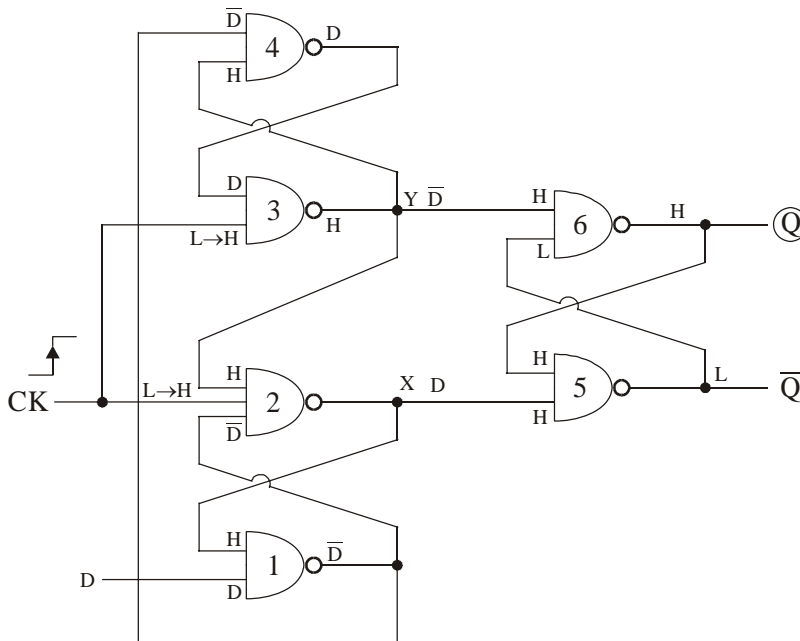
D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

D edge-triggered flip-flop

Also, master-slave version: Data sampled while CK HI, 'latched' when CK goes LO.

Circuit Analysis

Logical circuit realization.
D-type, positive edge-triggered.



CK LO \Rightarrow X = Y = HI $\rightarrow \Rightarrow$ Q, \bar{Q} outputs stable

CK HI enables 2 & 3 \Rightarrow

D	\bar{D}	Q_n	Q_{n+1}
0	1	—	0
		—	

At edge: $\left. \begin{array}{l} * D \rightarrow X \\ * \bar{D} \rightarrow Y \end{array} \right\} \rightarrow$

'latched' by output at this point

After 1 prop delay: (CK now HI)

1 \rightarrow false $(\overline{D \cdot D}) = 0$

\Rightarrow 2 \rightarrow HI \Rightarrow X \rightarrow HI

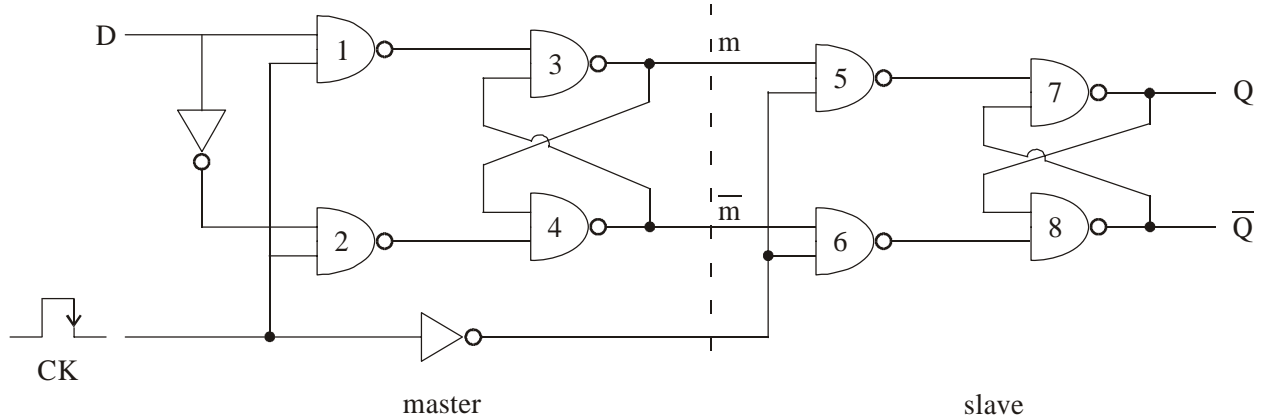
4 \rightarrow false $(\overline{\bar{D} \cdot \bar{D}}) = 0$

\Rightarrow 3 \rightarrow HI \Rightarrow Y \rightarrow HI

X HI, Y HI = back to initial condition, CK goes LO & output is stuck.

Circuit Analysis

Logical Circuit Realization, D flip-flop



Operation:

CK HI 1 & 2 enabled \Rightarrow master-slave version

- master flip-flop gets same state as D input (gates 3 & 4)
- gates 5 & 6 disabled, so output stable

CK goes LO inputs to master don't see D input, inputs of slave come from master \Rightarrow master transfers its state to slave

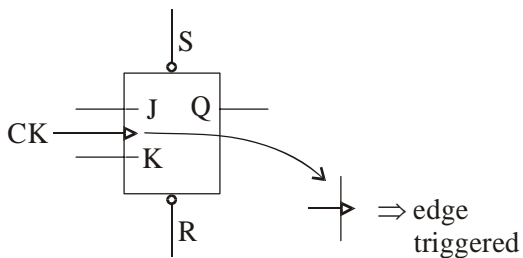
- master stuck, so no output changes

e.g.

D	CK	1	2	3	4	m
H	H	L	H	H	L	H
L	H	H	L	L	H	L

} ditto for slave section

J-K flop (like D, but 2 inputs)



J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

(reverse state after each clock pulse)

Jam set & reset also common:

