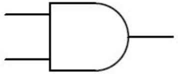
Review of digital logic design

Background

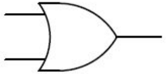
- Logic gates (AND, OR, NOT, XOR, etc.)
- Boolean algebra
- Truth tables
- Logic functions
- Logic function synthesis by
  - Sum of Products (SOP)
  - Product of Sums (POS)
  - K-maps
- Logic blocks (MUX, DEMUX)
- Sequential elements (Latch, Flip-flop)

Logic gates



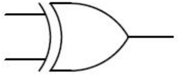
AND

A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1



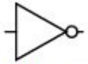
OR

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1



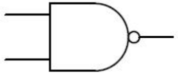
XOR

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0



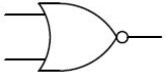
NOT

Input	Output
0	1
1	0



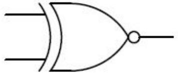
NAND

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0



NOR

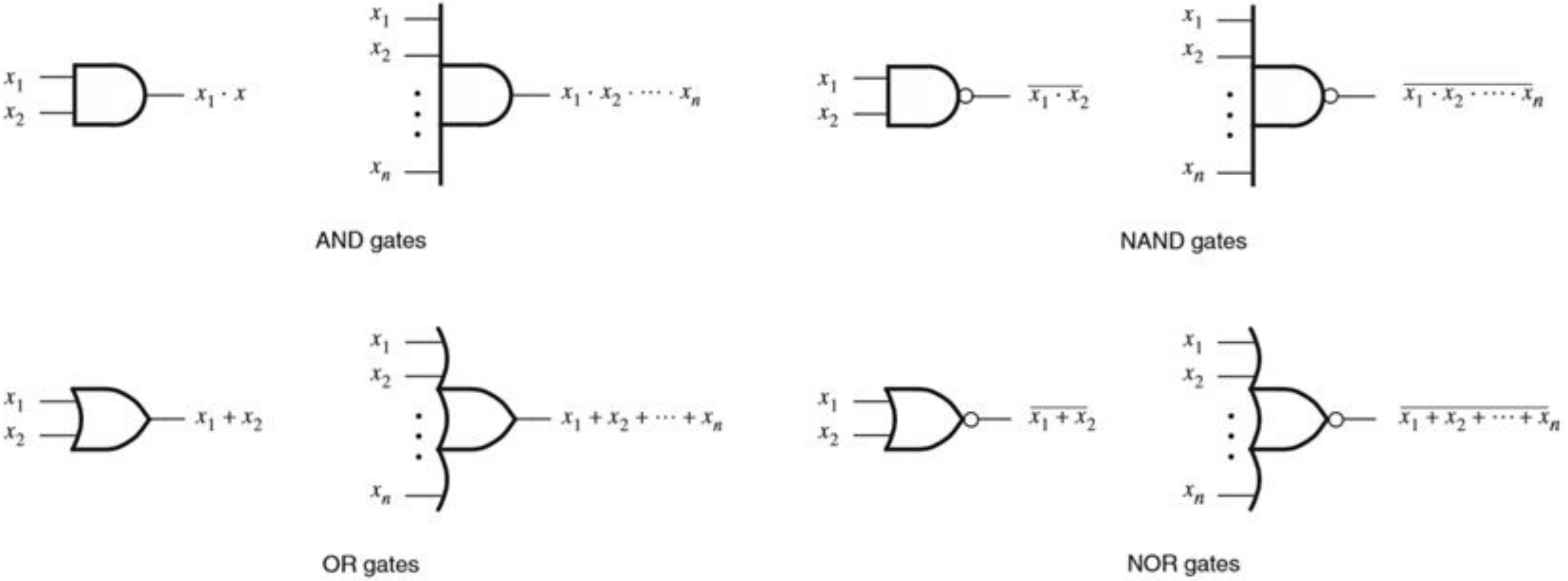
A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0



XNOR

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

Generalized n-input logic gates



Axioms of Boolean Algebra

- 1a. $0 \cdot 0 = 0$
- 1b. $1 + 1 = 1$
- 2a. $1 \cdot 1 = 1$
- 2b. $0 + 0 = 0$
- 3a. $0 \cdot 1 = 1 \cdot 0 = 0$
- 3b. $1 + 0 = 0 + 1 = 1$
- 4a. If $x = 0$, then $x' = 1$
- 4b. If $x = 1$, then $x' = 0$

Boolean Algebra - Single Variable Theorems

- 5a. $x \cdot 0 = 0$
- 5b. $x + 1 = 1$
- 6a. $x \cdot 1 = x$
- 6b. $x + 0 = x$
- 7a. $x \cdot x = x$
- 7b. $x + x = x$
- 8a. $x \cdot x' = 0$
- 8b. $x + x' = 1$
- 9. $(x')' = x$

- Boolean Algebra - Two Variable Properties

- 10a. $x \cdot y = y \cdot x$ Commutative
- 10b. $x + y = y + x$
- 11a. $x \cdot (y \cdot z) = (x \cdot y) \cdot z$ Associative
- 11b. $x + (y + z) = (x + y) + z$
- 12a. $x \cdot (y + z) = x \cdot y + x \cdot z$ Distributive
- 12b. $x + y \cdot z = (x + y) \cdot (x + z)$
- 13a. $x + x \cdot y = x$ Absorption
- 13b. $x \cdot (x + y) = x$
- 14a. $x \cdot y + x \cdot y' = x$ Combining
- 14b. $(x + y) \cdot (x + y') = x$

Boolean Algebra - Two & Three Variable Properties

- **15a. $(x \cdot y)' = x' + y'$ DeMorgan's theorem**
- **15b. $(x + y)' = x' \cdot y'$**
- 16a. $x + x' \cdot y = x + y$
- 16b. $x \cdot (x + y) = x \cdot y$
- 17a. $x \cdot y + y \cdot z + x' \cdot z = x \cdot y + x' \cdot z$ Consensus
- 17b. $(x + y) \cdot (y + z) \cdot (x' + z) = (x + y) \cdot (x' + z)$

Logic Function Synthesis - Three variable SOP & POS

- Function synthesis from truth table

Row number	x_1	x_2	x_3		Minterm	Maxterm
0	0	0	0		$m_0 = \bar{x}_1\bar{x}_2\bar{x}_3$	$M_0 = x_1 + x_2 + x_3$
1	0	0	1		$m_1 = \bar{x}_1\bar{x}_2 x_3$	$M_1 = x_1 + x_2 + \bar{x}_3$
2	0	1	0		$m_2 = \bar{x}_1 x_2\bar{x}_3$	$M_2 = x_1 + \bar{x}_2 + x_3$
3	0	1	1		$m_3 = \bar{x}_1 x_2 x_3$	$M_3 = x_1 + \bar{x}_2 + \bar{x}_3$
4	1	0	0		$m_4 = x_1\bar{x}_2\bar{x}_3$	$M_4 = \bar{x}_1 + x_2 + x_3$
5	1	0	1		$m_5 = x_1\bar{x}_2 x_3$	$M_5 = \bar{x}_1 + x_2 + \bar{x}_3$
6	1	1	0		$m_6 = x_1 x_2\bar{x}_3$	$M_6 = \bar{x}_1 + \bar{x}_2 + x_3$
7	1	1	1		$m_7 = x_1 x_2 x_3$	$M_7 = \bar{x}_1 + \bar{x}_2 + \bar{x}_3$

Logic Function Synthesis - 2/3/4 variable k-map

- Function synthesis using k-maps

x_1	x_2	
0	0	m_0
0	1	m_1
1	0	m_2
1	1	m_3

$x_2 \backslash x_1$	0	1
0	m_0	m_2
1	m_1	m_3

x_1	x_2	x_3	
0	0	0	m_0
0	0	1	m_1
0	1	0	m_2
0	1	1	m_3
1	0	0	m_4
1	0	1	m_5
1	1	0	m_6
1	1	1	m_7

$x_3 \backslash x_1 x_2$	00	01	11	10
0	m_0	m_2	m_6	m_4
1	m_1	m_3	m_7	m_5

$x_3 x_4 \backslash x_1 x_2$	00	01	11	10
00	m_0	m_4	m_{12}	m_8
01	m_1	m_5	m_{13}	m_9
11	m_3	m_7	m_{15}	m_{11}
10	m_2	m_6	m_{14}	m_{10}

Logic Function Synthesis - 2/3/4 variable k-map

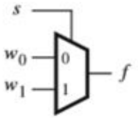
Function synthesis using k-maps

1. No zeros allowed.
2. No diagonals.
3. Only power of 2 number of cells in each group. (2^0=1, 2^1=2, 2^2=4, 2^3=8, etc.)
4. Groups should be as large as possible.
5. Every 1 must be in at least one group.
6. Overlapping allowed.
7. Wrap around allowed.
8. Fewest number of groups possible.

Visit: http://www.ee.surrey.ac.uk/Projects/Labview/minimisation/karrules.html

Multiplexer

- Multiple inputs, single output. Output is chosen by selector pin/s

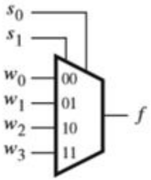


Graphical symbol

s	f
0	w_0
1	w_1

Truth table

2x1 Multiplexer



Graphical symbol

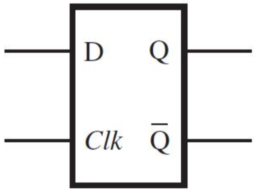
s_1	s_0	f
0	0	w_0
0	1	w_1
1	0	w_2
1	1	w_3

Truth table

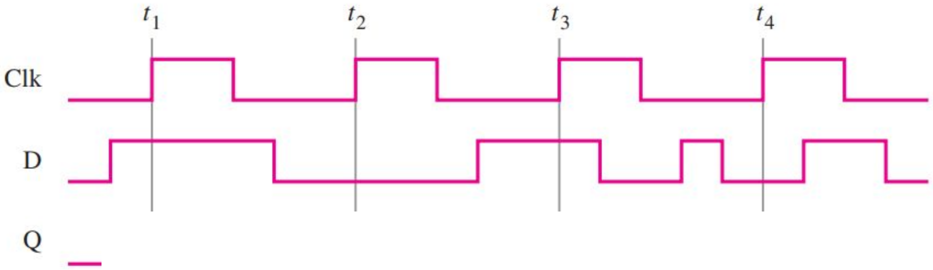
4x1 Multiplexer

D Latch

- Level sensitive element
- A *positive level triggered* D latch
  - copies D to output Q, if Clock=1, else preserves the previous output
- A *negative level triggered* D latch
  - copies D to output Q, if Clock=0, else preserves the previous output



Graphical symbol

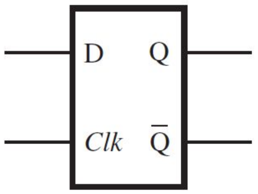


Clk	D	$Q(t+1)$
0	x	$Q(t)$
1	0	0
1	1	1

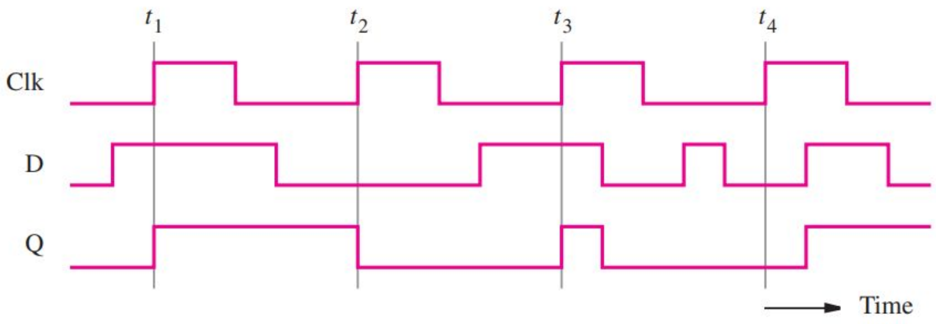
Characteristic table

D Latch

- Level sensitive element
- A ***positive level triggered*** D latch
  - copies D to output Q, if Clock=1, else preserves the previous output
- A ***negative level triggered*** D latch
  - copies D to output Q, if Clock=0, else preserves the previous output



Graphical symbol

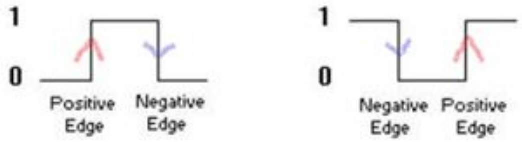


Clk	D	$Q(t+1)$
0	x	$Q(t)$
1	0	0
1	1	1

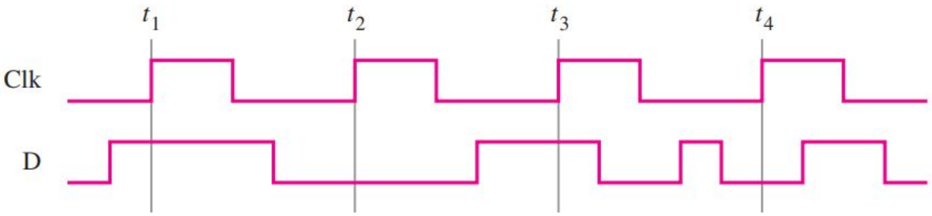
Characteristic table

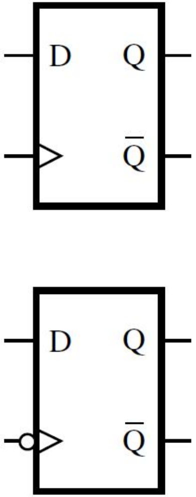
D Flip-flop

- Edge sensitive element



- A ***positive edge*** *triggered* D flip-flop
  - Sets Q=D at all positive edges (rising edges) of the clock, retains the old value of Q otherwise
- A ***negative edge*** *triggered* D flip-flop
  - Sets Q=D at all negative edges (falling edges) of the clock, retains the old value of Q otherwise

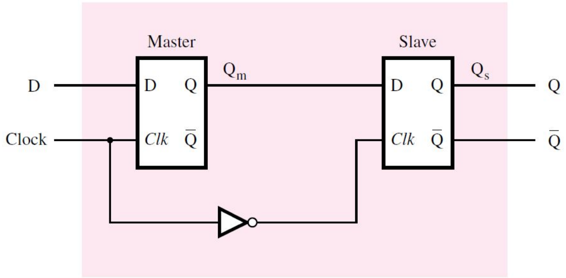


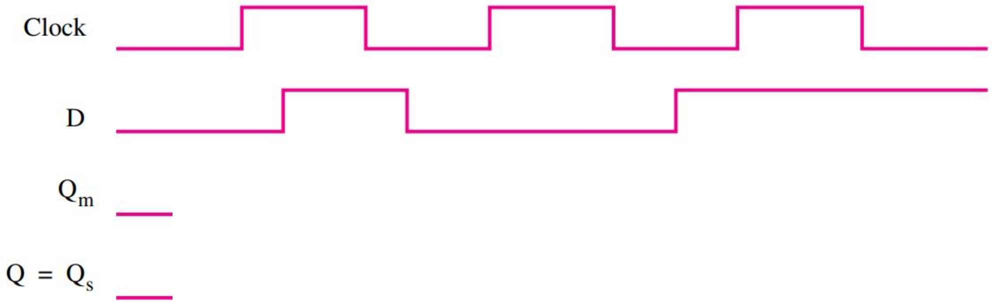


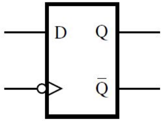
Graphical symbol

Building D Flip-flops using D Latches

- By cascading a positive level triggered D latch and a negative level triggered D latch we can build a negative edge triggered D flip-flop

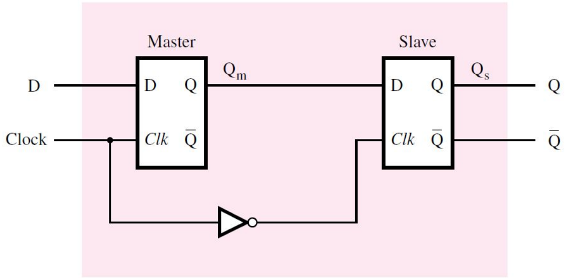


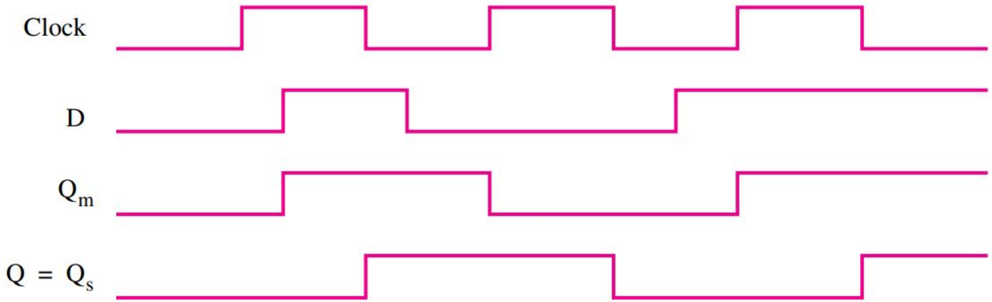


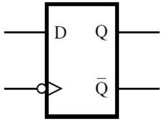


Building D Flip-flops using D Latches

- By cascading a positive level triggered D latch and a negative level triggered D latch we can build a negative edge triggered D flip-flop







symbol of negative edge Flipflop

follow the Q_m, clock_bar signals to draw Q_s ie, Q of the second latch

Level triggered vs. Edge triggered

- In level triggered elements
  - output is affected by the clock levels *(high/low)*
- In edge triggered elements
  - output is affected by the clock edges (positive edge/negative edge) (rising edge/falling edge)