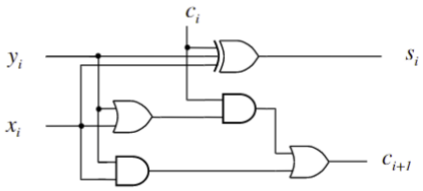
1 AND 2's COMPLEMENT:

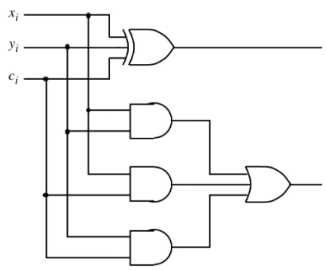
- 1's: simply invert all bits.
    - -2^(n - 1) - 1 to +2^(n - 1) - 1
- 2's: start on right until first 1: invert remaining bits.
    - -2^(n - 1) to +2^(n - 1) - 1

CIRCUITS FOR ADDERS AND FAST ADDERS:

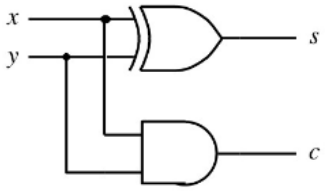
- Fast-Adder



Full-Adder



Half-Adder



SINGLE & DOUBLE PRECISION IEEE FLOATING POINT FORMATS:

- Decimal to float:
    - Convert to binary (abs. value), move to mantissa and keep track of of power (<- negative, -> positive), exponent = 127 + power from above, find binary of this #, then place mantissa behind this # and add sign in front of both.
- Fraction to float:
    - Convert to binary (take # * 2, keep track of # before decimal, continue multiplying decimal until decimal = 0). MSB is at the top, LSB at bottom. Opposite of normal.
    - Repeat same steps from above.
- IEEE Float to Decimal:
    - Mark off sign, mark off and find exponent, find value (2^(exponent - 127)), find mantissa (1 + ½ + ¼… the 1 is always included), multiply by value (2^(exponent - 127)), adhere to sign.

MULTIPLEXERS:

- 2^n to 1, n select lines. # of MUX = size of each number. Size of MUX = amount of numbers.

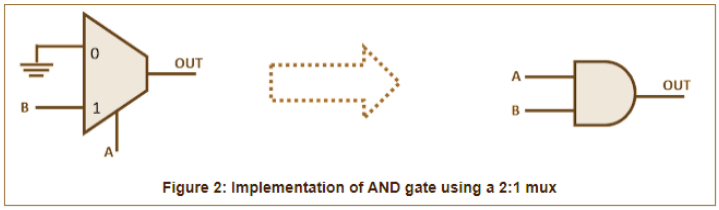


Figure 2: Implementation of AND gate using a 2:1 mux

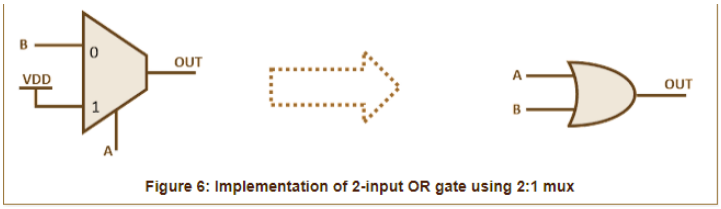


Figure 6: Implementation of 2-input OR gate using 2:1 mux

Figure 14: Implementation of NOT gate using 2x1 mux

SHANNON'S EXPANSION:

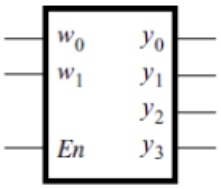
- Always start by pulling out MSB after writing entire function.
- F = ~MSB() + MSB()
- Continue until contents are single value.
    - Values pulled out are select line values.
    - Single values within are data inputs.

DECODERS:

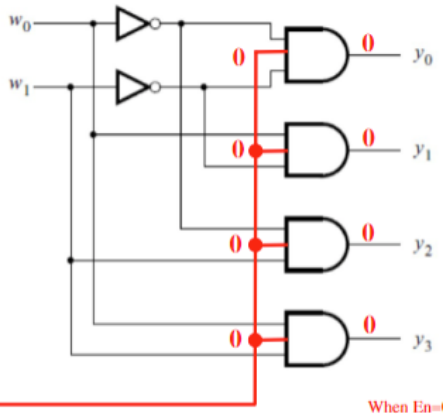
- n to 2^n

En	w_1	w_0	y_0	y_1	y_2	y_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	x	x	0	0	0	0

(a) Truth table



(b) Graphical symbol



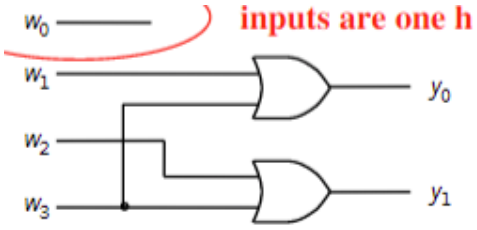
DEMULTIPLEXERS:

- Used to reverse an input to 2^n outputs:
  - 1-4 deMUX can make 2-4 decoder.
  - 1-8 deMUX can make 3-8 decoder.

ENCODERS:

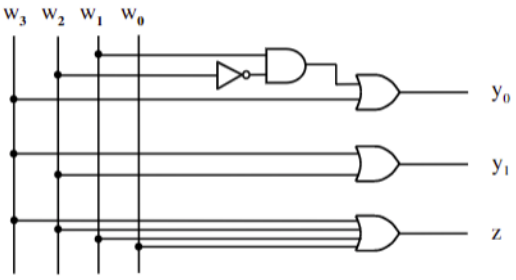
- 2^n to n
- Binary (“one-hot” encoded):

w_3	w_2	w_1	w_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



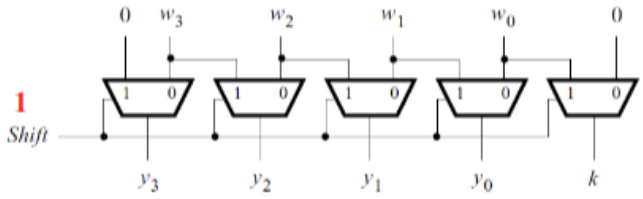
- Priority:

w_3	w_2	w_1	w_0	y_1	y_0	z
0	0	0	0	d	d	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1



CODE CONVERTERS:

- Shifter:



K-MAPS FOR 2, 3, & 4 VARIABLES:

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

x_2 \ 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 \ x_1x_2	00	01	11	10
0	m_0	m_2	m_6	m_4
1	m_1	m_3	m_7	m_5

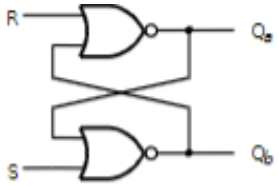
(b) Karnaugh map

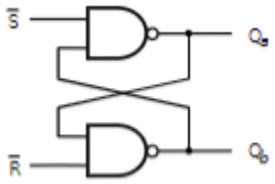
x1	x2	x3	x4	
0	0	0	0	m0
0	0	0	1	m1
0	0	1	0	m2
0	0	1	1	m3
0	1	0	0	m4
0	1	0	1	m5
0	1	1	0	m6
0	1	1	1	m7
1	0	0	0	m8
1	0	0	1	m9
1	0	1	0	m10
1	0	1	1	m11
1	1	0	0	m12
1	1	0	1	m13
1	1	1	0	m14
1	1	1	1	m15

x_3x_4 \ x_1x_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}

LATCHES (SR, GATED SR, GATED D):

- SR w/ NOR (if using NAND, swap and negate R and S, characteristics table gets inverted):

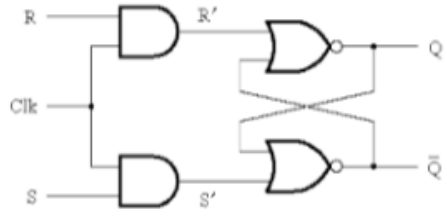




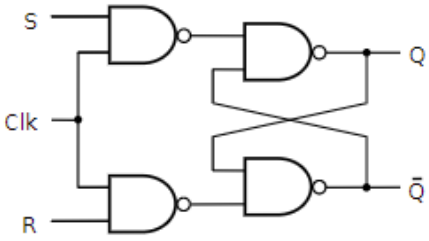
$\bar{S}$	$\bar{R}$	Q_a	Q_b	
0	0	1	1	
0	1	1	0	
1	0	0	1	
1	1	0/1	1/0	(no change)

S	R	Q_a	Q_b	
0	0	0/1	1/0	(no change)
0	1	0	1	
1	0	1	0	
1	1	1	1	

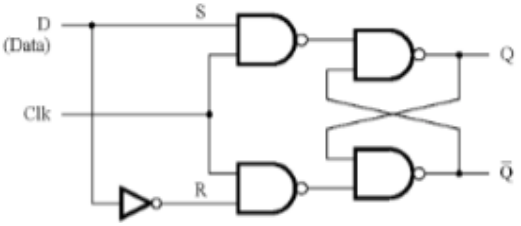
- Gated SR:



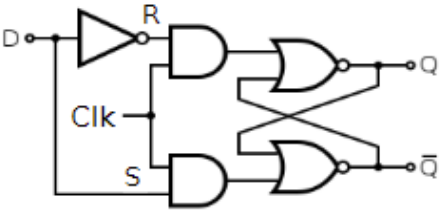
Clk	S	R	$Q(t+1)$
0	x	x	$Q(t)$ (no change)
1	0	0	$Q(t)$ (no change)
1	0	1	0
1	1	0	1
1	1	1	x



- Gated D:

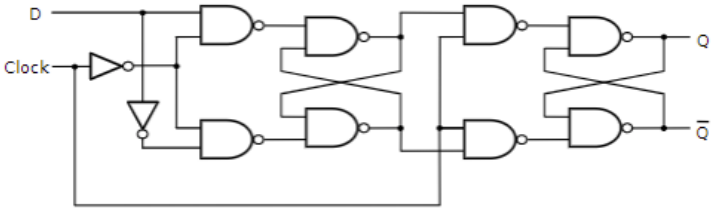


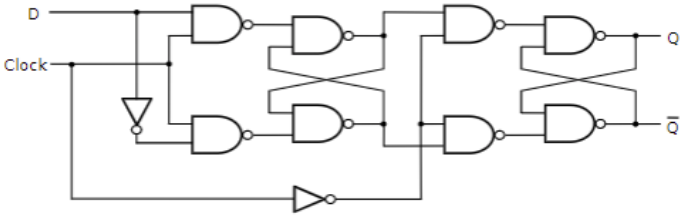
Clk	D	$Q(t+1)$
0	x	$Q(t)$
1	0	0
1	1	1



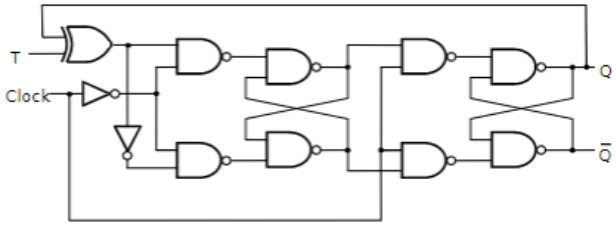
FLIP-FLOPS (D F.F., T F.F., AND JK F.F.):

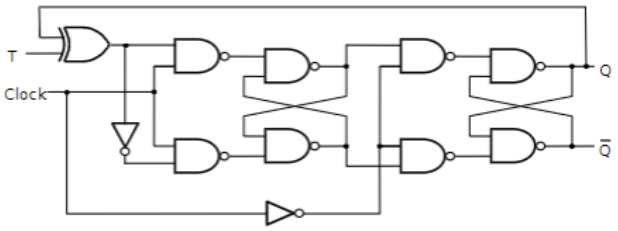
- D (Positive-edge, Negative-edge):





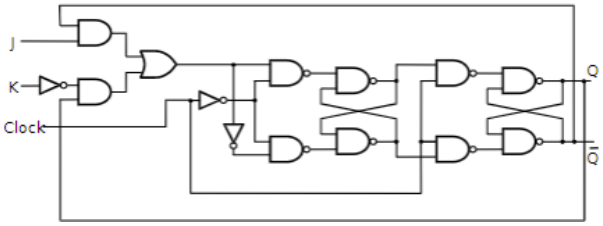
- T (Positive-edge, Negative-edge):

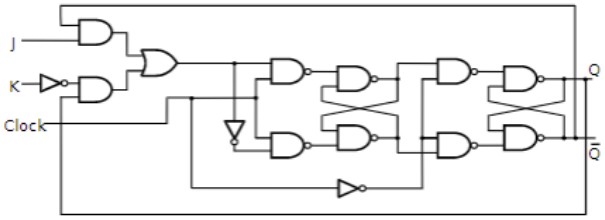




T	Q(t + 1)
0	Q(t)
1	$\overline{Q}(t)$

- JK (Positive-edge, Negative-edge):

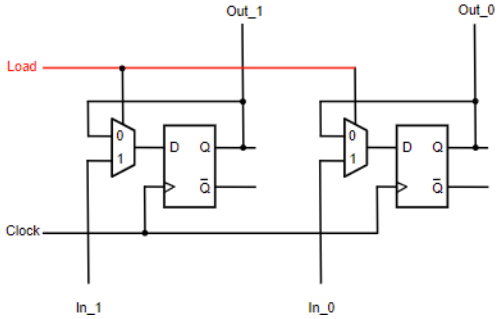




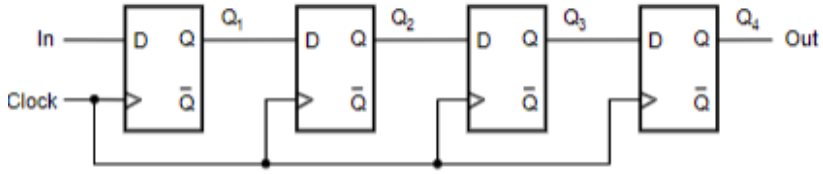
J	K	Q (t + 1)
0	0	Q (t)
0	1	0
1	0	1
1	1	$\overline{Q}$ (t)

REGISTERS & REGISTER FILES:

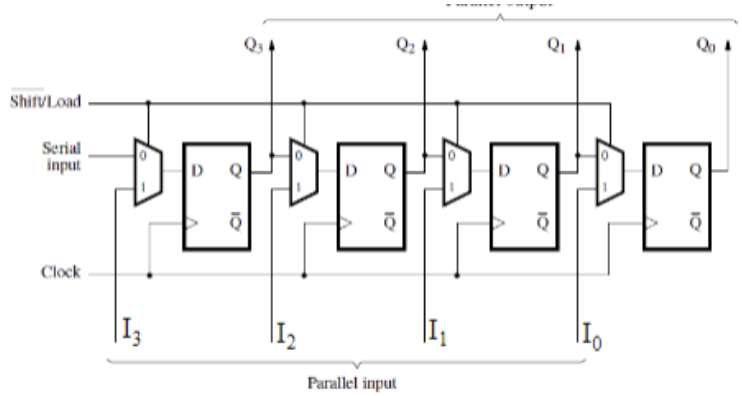
- Parallel-Access Register:



Shift Register:



- Parallel-Access-Shift Register:



COUNTERS: