Review of Digital Circuits & Logic Design

Number Systems

Common Number Systems

System	Base	Symbols	Used by humans?	Used in computers?
Decimal	10	0, 1, … 9	Yes	No
Binary	2	0, 1	No	Yes
Octal	8	0, 1, … 7	No	No
Hexa-decimal	16	0, 1, … 9, A, B, … F	No	No

Number Systems

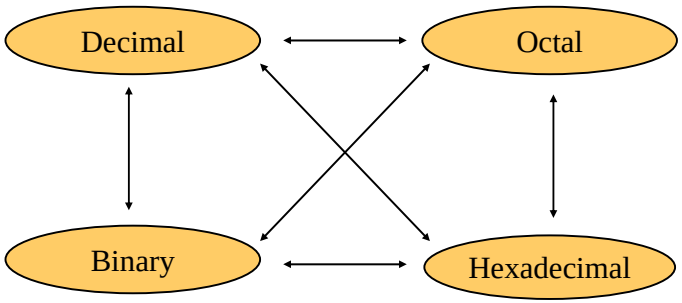
Quantities/Counting

Decimal	Binary	Octal	Hexa-decimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7

Decimal	Binary	Octal	Hexa-decimal
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Number Systems

Conversion Among Bases



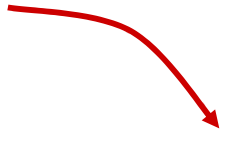
Number Systems

Decimal to Binary

- Technique
  - Divide by two, keep track of the remainder
  - First remainder is bit 0 (LSB, least-significant bit)
  - Second remainder is bit 1, and so on

$125_{10} = ?_2$

2 | 125		
2 | 62	1	
2 | 31	0	
2 | 15	1	
2 | 7	1	
2 | 3	1	
2 | 1	1	
0	1	



$125_{10} = 1111101_2$

Number Systems

Hexadecimal to Binary

- Technique
  - Convert each hexadecimal digit to a 4-bit equivalent binary representation

$10AF_{16} = ?_2$

1 0 A F

0001 0000 1010 1111

$10AF_{16} = 0001000010101111_2$

- Reverse the process when converting binary to hexadecimal

Number Systems

Negative Numbers

- Two ways to represent a negative binary number.
  - a signed bit to represent the positive or negative sign.
  - the complemented form of the positive number to represent a negative value.
- Negative number represented by a *sign* preceding its absolute value, e.g. -65.
- Similar technique can be employed to represent a negative binary number.
- Attach one digit to the binary number to represent the sign.
- This binary digit is the *sign bit*.
- Sign bit is always the MSB. The number is positive if the sign bit is 0 and is negative if the sign bit is 1.
- For example, 00101_2 represents $+0101_2$, while 10101_2 means -0101_2

Number Systems

- The advantage is simple.
- Disadvantages
  - Duplication in representing zero.

    i.e., 00 and 10 represent zero
  - addition to a signed negative number gives an **incorrect** answer.

+	1011101	(+93)
-	1000001	(-65)
	0011100	(+28)

	01011101	(+93)
+	11000001	(-65)
1	00011110	(+30)

↑
Overflow

Number Systems

Complement representation

- The 1's complement and the 2's complement.
- The most significant bit still represents the sign.
- The other digits are not the same.

1's complement

- The negative value of an n-digit binary number, N_2, can be represented by its 1's complement, N'_2
- It is equal to the base number, 2, to the power n minus the number N minus one, or:

  $N'_2 = 2^n - N_2 - 1 = (2^n - 1) - N_2$
- N'_2 can be obtained by subtracting each digit with 1, or by inverting all binary digits.

Number Systems

- 1's complement number of $\mathbf{01001101_2}$:
- n = 8.

  $N'_2 = (2^8 - 1) - N_2$

  $= (100000000 - 1) - 01001101$

  $= 11111111 - 01001101$

  $= 10110010_2$

Number Systems

2's complement

- The 2's complement, N''_2, of an n-digit binary number, N_2, is defined as:

  $N''_2 = 2^n - N_2$

  $= (2^n - 1) - N_2 + 1$

  $= N'_2 + 1$
- where N'_2 is the 1's complement representation of the number.

Number Systems

- No duplicate representation for all numbers.

  $N_2 =$ 00000000

  $N'_2 =$ 11111111

  $N''_2 = N'_2 + 1 = 1\ 00000000$
- Since only 8 bits are used, the base complement of zero is zero, too.
- With 8-bit 2's complement representation, total 256 numbers

  $\mathbf{10000000_2}$ (-128_{10}) to $\mathbf{00000000_2}$ (0_{10}) to $\mathbf{01111111_2}$ ($+127_{10}$).

Number Systems

- 2's complement number of **01001101**.

  1's complement, N' is **10110010**

  N" = N' + 1

  = 10110010 + 1

  = 10110011

- Subtraction of two numbers:

  $X_2 - Y_2 = X_2 + (- Y_2)$

- In 2's complement representation:

  $X_2 + (2^n - Y_2) = X_2 - Y_2 + 2^n$

- If X > Y, the sum will produce an end carry, 2^n, which can be discarded, and the sum is positive.

Number Systems

- If X < Y, the sum is negative and will be

  $2^n - (Y_2 - X_2)$

  which is the 2's complement of $(Y_2 - X_2)$.

- Example: 93 - 65

  +93 = 01011101_2

  -65 = 100000000-01000001 = 10111110 (in1's complement)

  = 10111111 (in 2's complement)

93 - 65

	01011101	(+93)	(X)
+	10111111	(-65)	(2^8 -Y)
1	00011100	(+28)	(X-Y)

↑

Overflow (2^8)

Number Systems

- Example: 65 - 93

  +65 = 01000001_2

  -93 = 100000000-01011101

  = 10100010 (in 1's complement)

  = 10100011 (in 2's complement)

6 5 – 9 3

	0 1 0 0 0 0 0 1	(+ 6 5)	(X)
+	1 0 1 0 0 0 1 1	(- 9 3)	(- Y)
	1 1 1 0 0 1 0 0	(- v e)	

↑

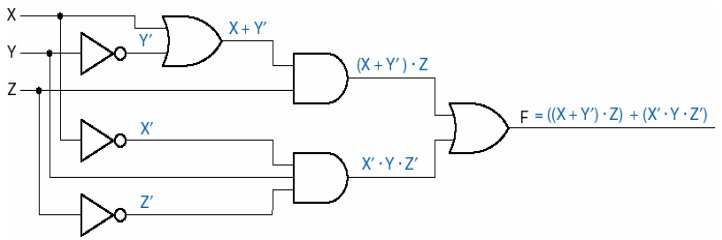
2^8 - (Y - X)

1 0 0 0 0 0 0 0 0		2^8
- 1 1 1 0 0 1 0 0		2^8 - (Y - X)
0 0 0 1 1 1 0 0		- (Y - X)

(X - Y) = - 0 0 0 1 1 1 0 0 $_2$ = - 2 8

Combinational Logic

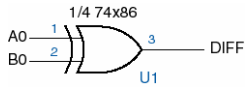
- outputs logical functions of inputs
- outputs appear shortly after changed inputs (propagation delay)
- no feedback loops
- no clock



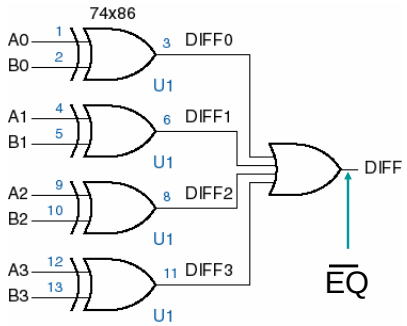
Combinational Logic

Equality Comparators

1-bit comparator



4-bit comparator



Adders

1-bit half-adder: two 1-bit inputs, output "half sum" and carry

1-bit full adder: two 1-bit inputs and carry-in, output sum and carry

Full adder truth table:

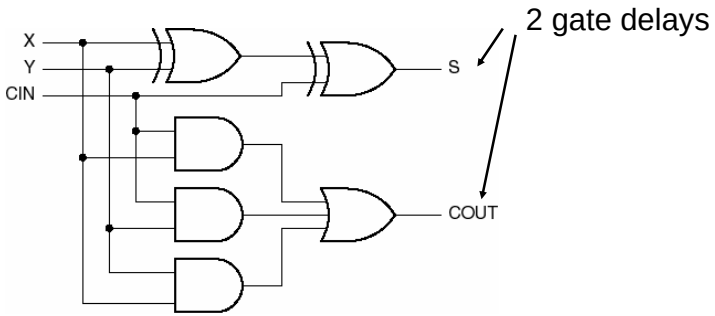
X	Y	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

S:	XY 00	01	11	10
0		1		1
Cin 1	1		1	

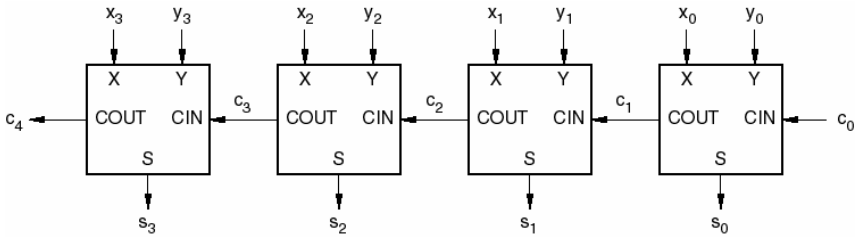
Cout:	XY 00	01	11	10
0			1	
Cin 1		1	1	1

$$C_{out} = XC_{in} + YC_{in} + XY$$

Full-adder circuit

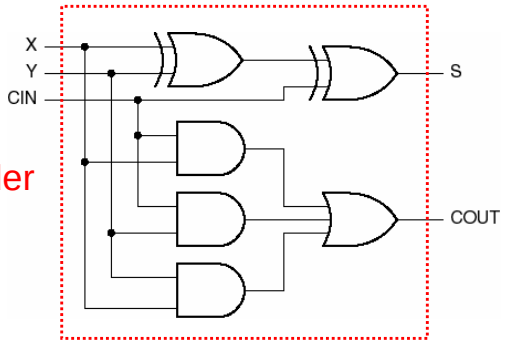


Ripple adder



Any limitation ?

Subtractor



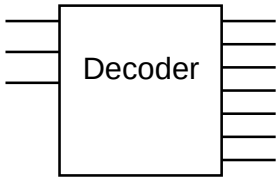
Full adder

- No need for specific hardware
- Use full adder
- How ??? ➔ **HOME WORK #1**

Subtraction

- Subtraction is the same as addition of the two's complement.
- The two's complement is the bit-by-bit complement plus 1.
- Therefore, $X - Y = X + \overline{Y} + 1$
  - Complement Y inputs to adder, set C_{in} to 1
  - For a borrow, set C_{in} to 0.

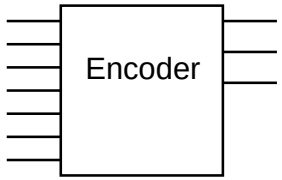
Encoders vs. Decoders



Usually binary in, 1-of-n out

e.g. in out

000 line 0 asserted
001 line 1 asserted
010 line 2 asserted
011 line 3 asserted
100 line 4 asserted
101 line 5 asserted
110 line 6 asserted
111 line 7 asserted

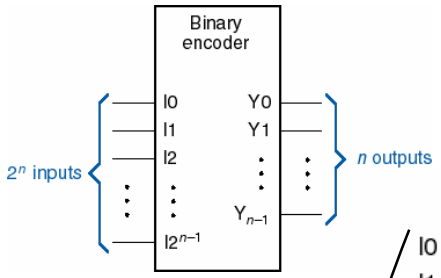


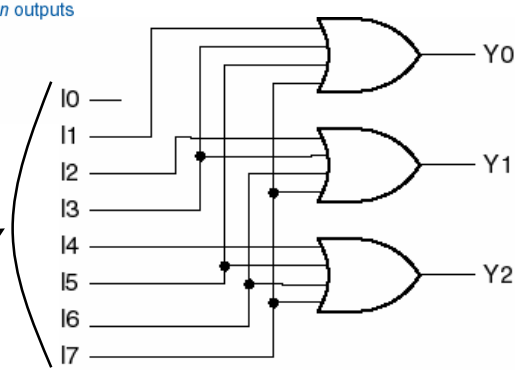
Usually 1-of-n in, binary out

e.g. out in

000 line 0 asserted
001 line 1 asserted
010 line 2 asserted
011 line 3 asserted
100 line 4 asserted
101 line 5 asserted
110 line 6 asserted
111 line 7 asserted

Binary encoders

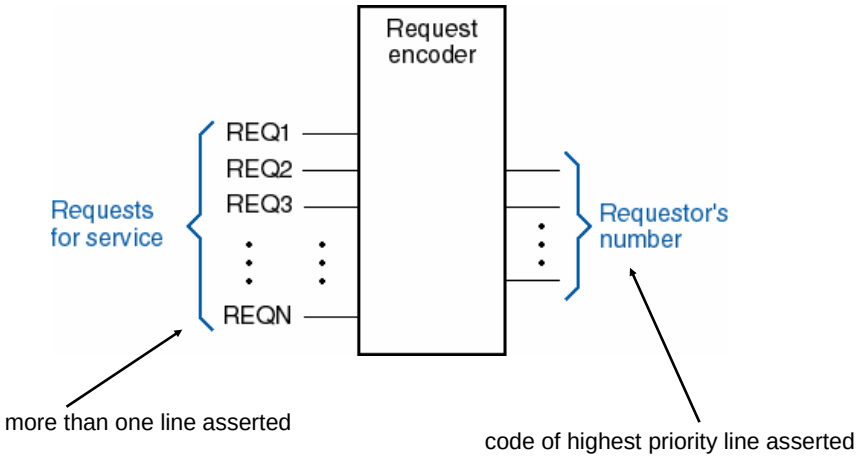




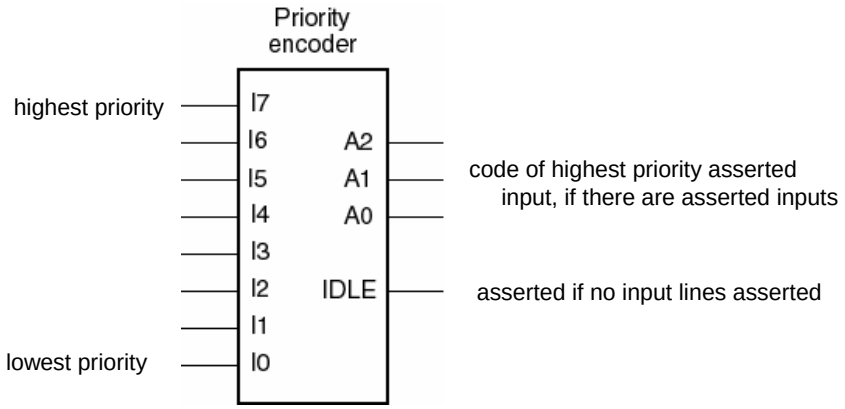
exactly one line asserted

if none of I1 through I7 asserted, assume I0 by default

Need priority encoder in most applications

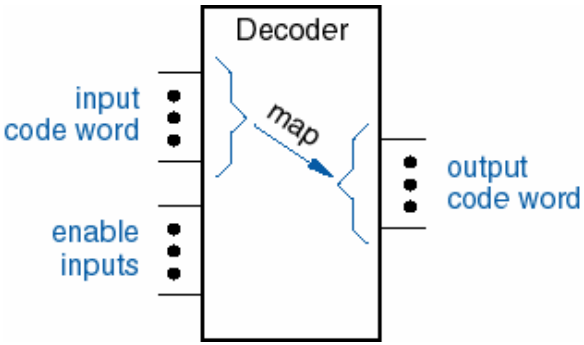


8-input priority encoder



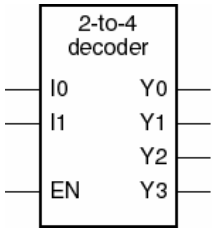
Decoders

- General decoder structure



- Typically *n* inputs, 2^n outputs
  - 2-to-4, 3-to-8, 4-to-16, etc.

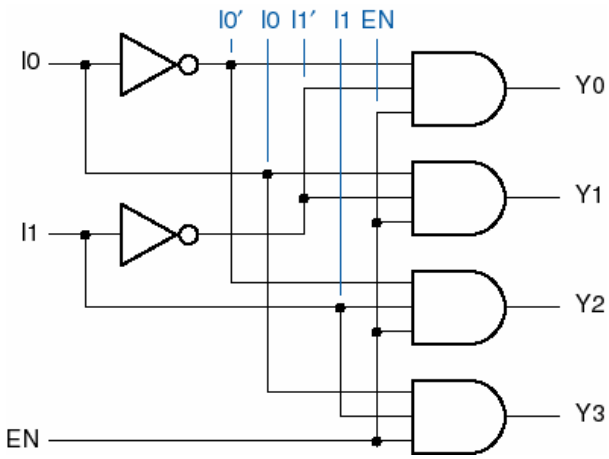
Binary 2-to-4 decoder



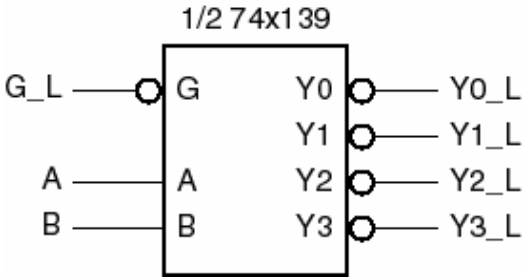
Inputs			*Outputs*			
EN	I1	I0	Y3	Y2	Y1	Y0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Note “x” (don’t care) notation.

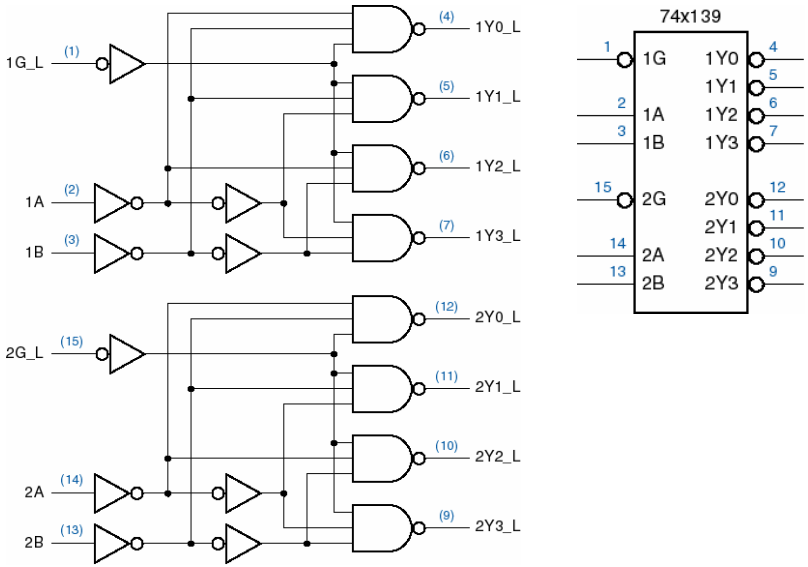
2-to-4-decoder logic diagram



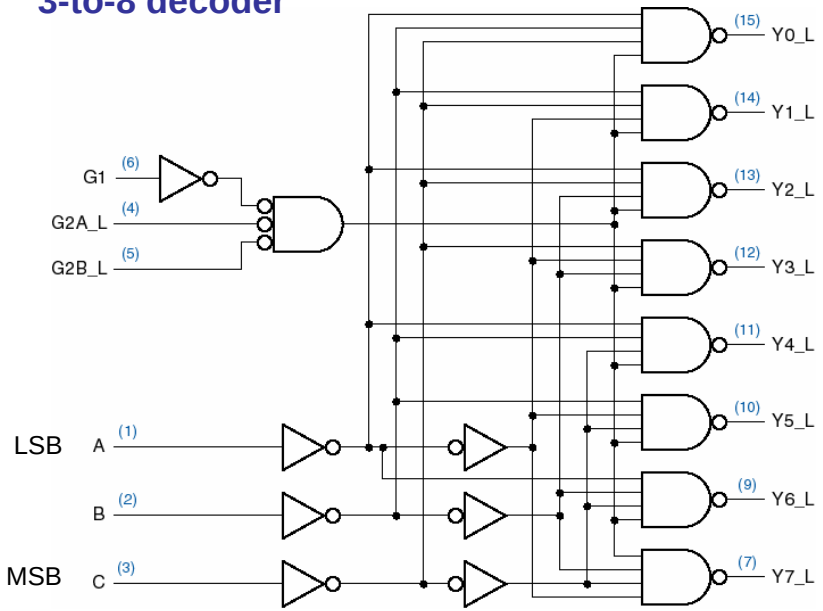
Decoder Symbol



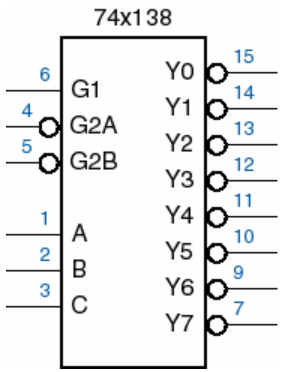
Complete 74x139 Decoder



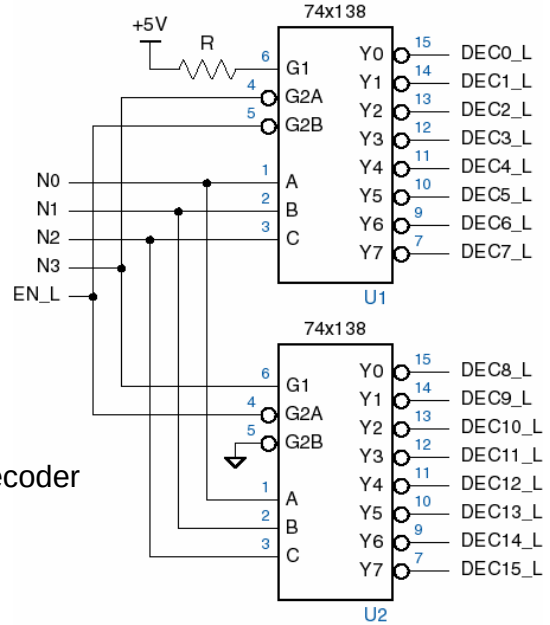
3-to-8 decoder



74x138 3-to-8-decoder symbol



Decoder cascading

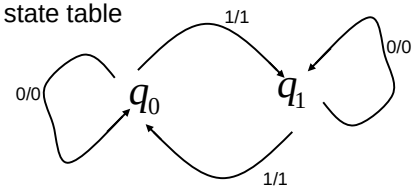


4-to-16 decoder

Sequential Logic

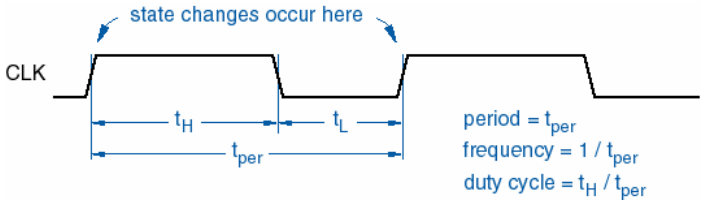
- outputs logical functions of inputs and previous history of circuit (memory)
- after changed inputs, new outputs appear in the next clock cycle
- frequent feedback loops
- sequential circuits can be described using:
  - * state table
    - For each current-state, specify next-states as function of inputs
    - For each current-state, specify outputs as function of inputs
    - Like a separate combinational problem for each state
  - * state diagram

graphical representation of state table



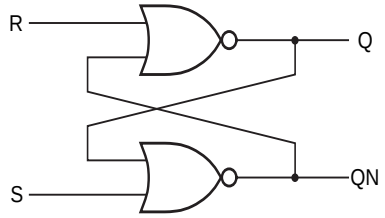
Clock signals

- Very important with most sequential circuits
  - State variables change state at clock edge.



Sequential Circuit Elements

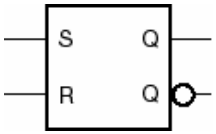
S-R Latch

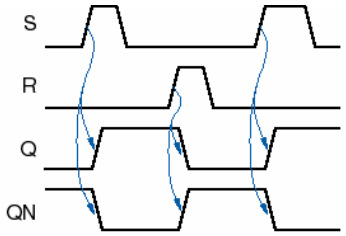


(a)

S	R	Q	QN
0	0	last Q	last QN
0	1	0	1
1	0	1	0
1	1	0	0

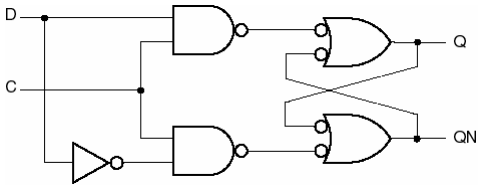
(b)



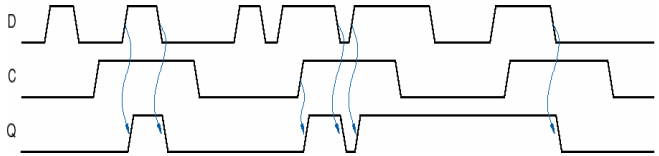


Sequential Circuit Elements

D latch

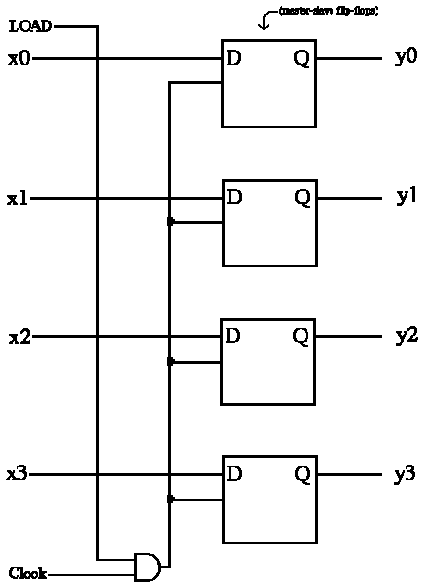


C	D	Q	QN
1	0	0	1
1	1	1	0
0	x	last Q	last QN



Storage Elements

- Single Latch is 1-bit storage
- Multiple latches arranged in linear form is called "**register**"
- 2-D form of register is called "**memory**"



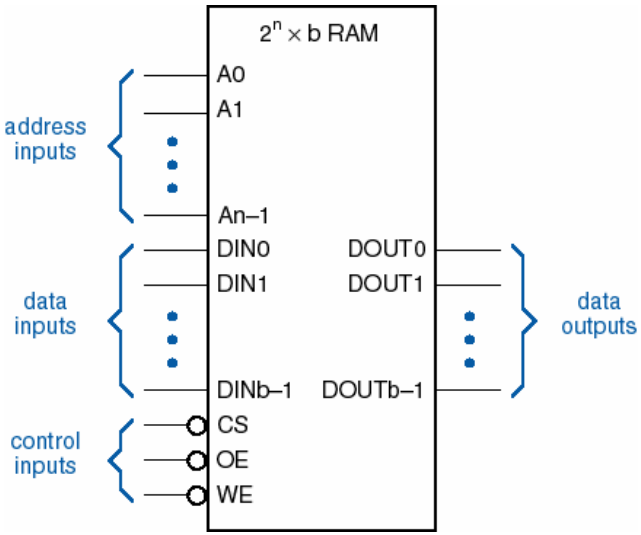
Storage Elements

Read/Write Memories

- a.k.a. "RAM" (Random Access Memory)
- Volatility
  - Most RAMs lose their memory when power is removed
  - NVRAM = RAM + battery
  - Or use EEPROM
- SRAM (Static RAM)
  - Memory behaves like latches or flip-flops
- DRAM (Dynamic Memory)
  - Memory lasts only for a few milliseconds
  - Must "refresh" locations by reading or writing

Storage Elements

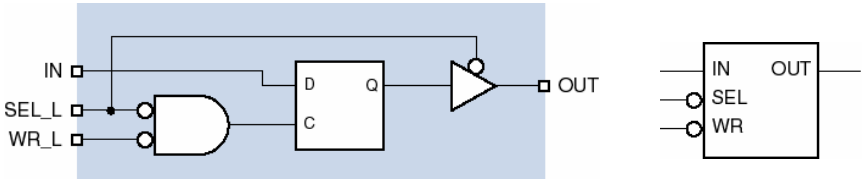
SRAM



Storage Elements

SRAM operation

- Individual bits are D latches, ***not*** edge-triggered D flip-flops.
  - Fewer transistors per cell.
- Implications for write operations:
  - Address must be stable before writing cell.
  - Data must be stable before ending a write.



Storage Elements

SRAM array