Computer Organization
(Logic circuits design and minimization)

KR Chowdhary
Professor & Head
Email: kr.chowdhary@gmail.com
webpage: krchowdhary.com

Department of Computer Science and Engineering
MBM Engineering College, Jodhpur

November 14, 2013

Instructions and Data

- Instructions and data are in binary (1, 0) format
- Binary Levels in logics: TTL (0-0.8) = logic 0, (2.0-5.0 v) = logic 1(true). Other are ECL, DTL, RTL, etc.
- How the CPU identifies a binary string as Data or Instructions?
- What is Minimum circuit to store a bit (0 / 1)?

Single Memory Cell

- Flip-flop(Bipolar Transistor) as single memory cell.
- One transistor is always in saturation and other in cutoff. See figure
- Why?

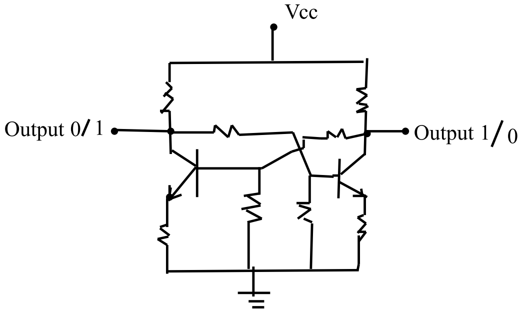
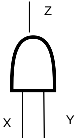


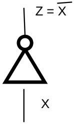
Figure: Single cell to store 1 or 0

- basic gates and universal gates



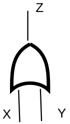
Truth table AND GATE

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1



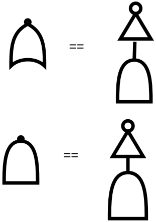
TT not gate

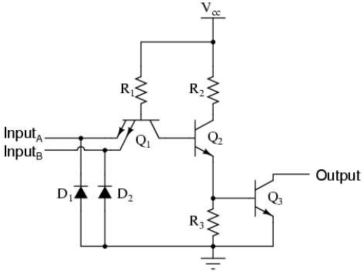
X	Z
0	1
1	0



truth Table OR GATE

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

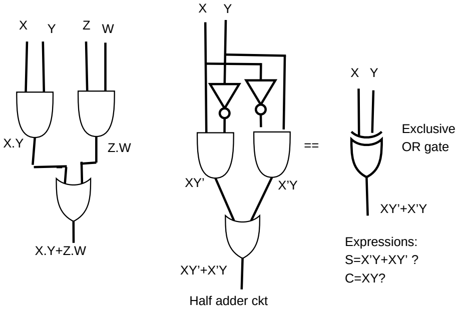




- Fan-In: The maximum number of gates which can be connected input to a gate.
- Fan-Out: The maximum number of gates which can be connected at the output of a gate.

Simple Boolean Logic Circuits

- Building circuits from basic gates
- These are called **Combinational Circuits** (they have no memory element)
- **Sequential circuits:** O/P is function of current I/P and previous I/P.

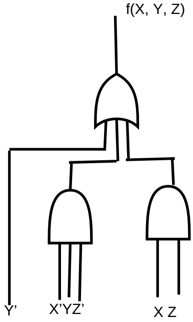


These circuits' output is directly dependent on I/P?

Standard forms of expressions

- We can write expressions in many ways, but some ways are more useful than others
- A sum of products (SOP) expression contains:
  - Only OR (sum) operations at the "outermost" level
  - Each term that is summed must be a product of literals

  $f(x,y,z) = y' + x'yz' + xz$
- The advantage is that any sum of products expression can be implemented using a two-level circuit
  - literals and their complements at the "0th" level
  - AND gates at the first level
  - a single OR gate at the second level



Minterms

- A minterm is a special product of literals, in which each input variable appears exactly once.
- A function with n variables has 2^n minterms (since each variable can appear complemented or not)
- A three-variable function, such as $f(x, y, z)$, has $2^3 = 8$ minterms
  $x'y'z', x'y'z, x'yz', x'yz, xy'z', xy'z, xyz', xyz$
  ($m_0, m_1, m_2, m_3, m_4, m_5, m_6, m_7$ in short)

Product of Sums

- A product of sums (POS) expression contains:
  - Only AND (product) operations at the "outermost" level
  - Each term must be a sum of literals
- Product of sums expressions can be implemented with two-level circuits
  - literals and their complements at the "0th" level
  - OR gates at the first level
  - a single AND gate at the second level

$$f(x,y,z) = y'(x'+y+z')(x+z)$$

Maxterms

- A maxterm is a sum of literals, in which each input variable appears exactly once.
- A function with n variables has 2^n maxterms
- Each maxterm is false for exactly one combination of inputs:
- The maxterms for a three-variable function f(x,y,z): $(x' + y' + z')...(x + y + z)$ ($M_0, ..., M_7$ in short)

  Product of maxterms:
- If you have a truth table for a function, you can write a product of maxterms expression by picking out the rows of the table where the function output is 0 (Be careful if you are writing the actual literals!)

Maxterms

X	Y	Z	$f(X,Y,Z)$	$f'(X,Y,Z)$
0	0	0	1	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	1	0
1	1	1	0	1

$$\begin{aligned} f &= (x' + Y + Z)(X' + Y + Z')(X' + Y' + Z') \\ &= M_4 M_5 M_7 \\ &= \Pi M(4, 5,) \\ f' &= (x + Y + Z)(X + Y + Z')(X + Y' + Z)(X + Y' + Z')(X' + Y' + Z) \\ &= M_0 M_1 M_2 M_3 M_6 \\ &= \Pi M(0, 1, 2, 3, 6) \end{aligned}$$

Conversion between standard forms

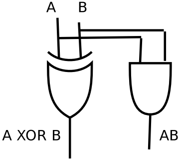
The minterms can be converted into maxterms and vice-versa using De-Morgan's laws. Considering maxterms' expression as

$$\begin{aligned} f' &= \Pi M(0,1,2,3,6) \\ &= M_0 M_1 M_2 M_3 M_6 \\ (f')' &= (M_0 M_1 M_2 M_3 M_6)' \\ f &= (M_0' + M_1' + M_2' + M_3' + M_6') \\ &= (m_4 m_5 m_7) \\ &= \Sigma m(4,5,7) \end{aligned}$$

In the similar way, the minterms can be converted into maxterms.

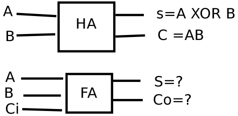
Combinational Circuits

- Adders, Subtractors, Multipliers, Dividers, Multiplexers, Demultiplexers (o/p depends on current input)



truth table

A	B	Sum= A'B+AB' = A XOR B	Carry=AB
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Full Adder (FA) TT:

A	B	Ci	S	CO
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=A XOR B XOR Ci
Co=AB OR ACi OR BCi

- A full adder adds two bits and *carry* of previous addition. How many FAs are required to add two 4-bit binary numbers? Why FA is FA?

Max and Min term expressions

$f(x_1, \ldots, x_n) = \sum_i \dot{x_{i1}} \ldots \dot{x_{in}}$, where $\dot{x_{ij}} = x_{ij} | \overline{x_{ij}}$ (SOP)

$f(x_1, \ldots, x_n) = \prod_i (\dot{x_{i1}} + \cdots + \dot{x_{in}})$. (POS)

The above is called *Quinne-McClusky Method*.

$S = \bar{A}\bar{B}C_i + \bar{A}B\bar{C}_i + A\bar{B}\bar{C}_i + ABC_i$, from truth table

$C_o = \bar{A}BC_i + A\bar{B}C_i + AB\bar{C}_i + ABC_i$, from truth table

(Or, function $C_o(A, B, C_i) = (m_3, m_5, m_6, m_7)$. The *minterm* m_i assumes a value 1 for unique value of variables. *Maxterm* defines the 0s in the truth table).

Alternatively, sum S is: $(A \oplus B) \oplus C_i$, ($\oplus$ is ex-or)

$= (\bar{A}B + A\bar{B}) \oplus C_i$

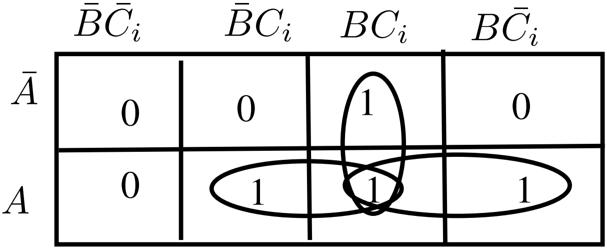
$= ABC_i + \bar{A}\bar{B}C_i + \bar{A}B\bar{C}_i + A\bar{B}\bar{C}_i$

C_o, the carry out of full adder, is $AB + AC_i + BC_i$

Karaugh Map for minimization of gate circuits using minterms

After **Maurice Kaurnaugh** (Bell Labs, 1950)

Karnaugh Map for Co



$$C_o = AC_i + AB + BC_i$$

The function $C_o(A, B, C_i)$ Can be implemented by three *AND* gates plus one *OR* gate. Alternatively by: four gates each having fan-in 2.

K-map for minimization using maxterms

	$\bar{B}\bar{C}_i$	$\bar{B}C_i$	BC_i	$B\bar{C}_i$
$\bar{A}$	0	0	1	0
A	0	1	1	1

$$
\begin{aligned}
C_o(A,B,C_i) &= (A+B)(A+C_i)(B+C_i)\\
&= (A+AC_i+AB+BC_i)(B+C_i)\\
&= \dots\\
&= AB+BC_i+AC_i
\end{aligned}
$$