EED2003 Digital Design

Presentation 3: Combinational Logic Design

Asst. Prof.Dr. Ahmet ÖZKURT
Asst. Prof.Dr Hakkı T. YALAZAN

Based on the Notes by Jaeyoung Choi Fall 2000

3.1 Combinational Circuits

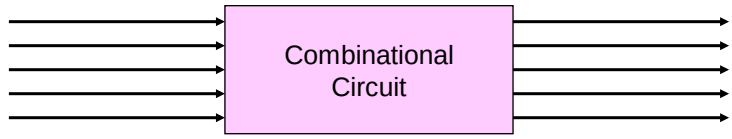
- logic circuits for digital systems: ***combinational*** vs ***sequential***

- Combinational Circuit (Chap 3)
  - outputs are determined by the present applied inputs
  - performs an operation, which can be specified logically by a set of Boolean expressions

- Sequential Circuit (Chap 4)
  - logic gates + storage elements (called flip-flops)
  - outputs are a function of the inputs & bit values in the storage elements (state of storage elements is a function of previous inputs)
  - output depends on the present values of inputs & past inputs

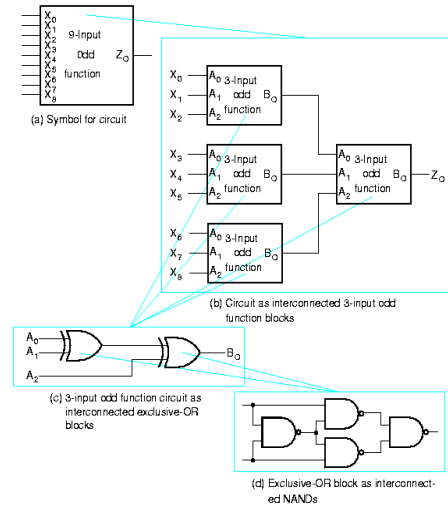
3.1 Combinational Circuits

- Combinational Circuit (Fig 3.1)
  - consist of input/output variables, logic gates, & interconnections
  - n inputs: 2^n possible combinations
  - m outputs: circuit can be described by m Boolean expressions
  - logic gate transforms binary information from input to outputs

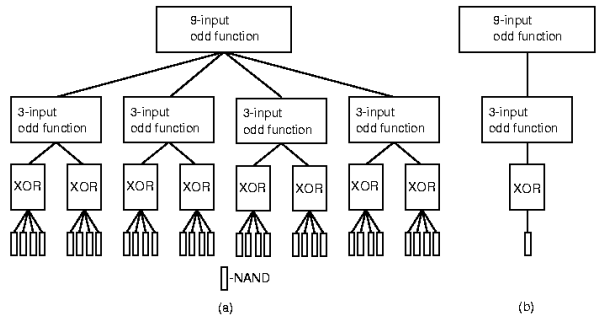


3.2 Design Topics

- Design Hierarchy
  - a circuit may be specified by a symbol showing inputs, outputs, & a description defining exactly how it operates
  - a circuit is composed of logic gates that are interconnected (in terms of implementation)
  - a single VLSI processor contains several million gates
    - ☞ approach divide & conquer
      - ☞ the circuit is broken up into pieces (blocks)



3.2 Design Topics



- Hierarchical Design
  1. reduces the complexity
     required to represent the schematic diagram of a circuit
  2. ends at a set of "leaves" (consisting of AND gates)
  3. reuse of blocks

3.2 Design Topics

- Functional Blocks
  - predefined, reusable blocks
    that typically lie in the lower levels of logic design hierarchies
  - provide functions that are broadly useful in digital design
  - available for decades in MSI circuits
- Computer-Aided Design
  - logic simulator
  - HDL (hardware description language)
- Top-Down Design
  (cf) bottom up

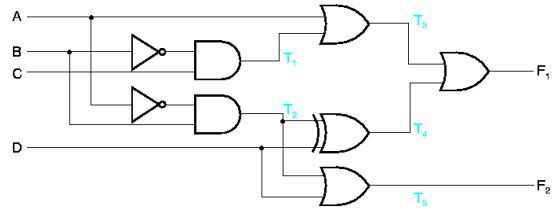
3.3 Analysis Procedure

- Determining the function that the circuit implements
  1) make sure that the circuit is combinational, not sequential
     - no feedback or storage elements
  2) obtain the output Boolean functions or the truth table
     & interpret the operation of the circuit

- Derivation of Boolean Functions
  - to obtain the output Boolean functions from a logic diagram
    1) label all gate outputs &
       determine the Boolean functions for each gate
    2) label the gates & find Boolean functions for each gate
    3) repeat (2) until the outputs of the circuits are obtained

3.3 Analysis Procedure

Ex)



4 inputs, A, B, C, D; 2 outputs, F_1, F_2

$T_1 = B' C$; $T_2 = A' B$

$T_3 = A + T_1 = A + B' C$ $T_4 = T_2 \oplus D = (A' B) \oplus D = A'BD' + AD + B'D$

$T_5 = T_2 + D = A'B + D$

So,

$F_2 = T_5 = T_2 + D = A'B + D$

$F_1 = T_3 + T_4$

$= A + B'C + A'BD' + AD + B'D$

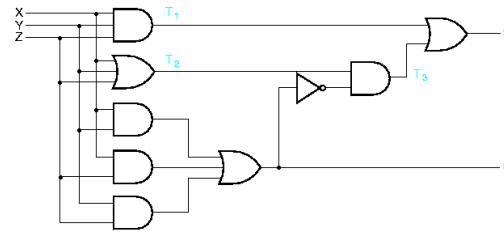
$= A + B'C + BD' + B'D$ (by algebra)

3.3 Analysis Procedure

- Derivation of the Truth Table
  - a straightforward process
    1) Determine no of input variables in the circuit
       For n inputs, list 2^n binary numbers
    2) Label the outputs of selected gates
    3) Obtain the truth table for the outputs of those gates
    4) Proceed to obtain the truth table for the outputs of gates

3.3 Analysis Procedure

Ex) binary adder



- Truth Table for Binary Adder

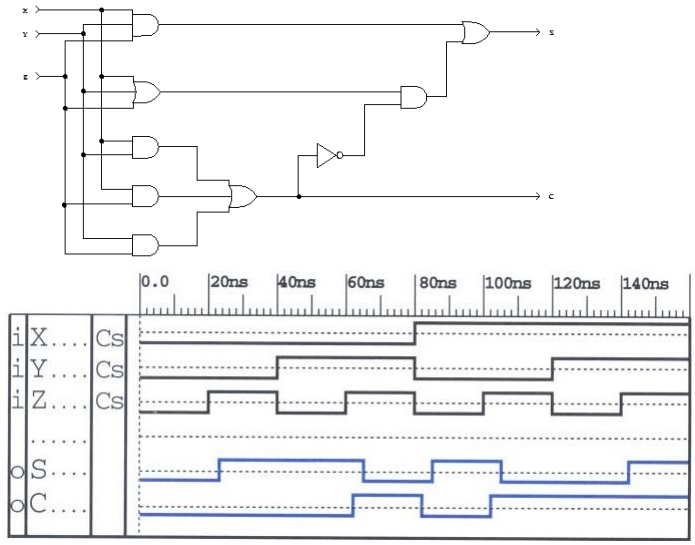
3 inputs, X, Y, Z;
2 outputs C, S (00 ~ 11)

C is 1 if XY, XZ, or YZ = 11;
C is 0 otherwise

X	Y	Z	C	C'	T_1	T_2	T_3	F_1
0	0	0	0	1	0	0	0	0
0	0	1	0	1	0	1	1	1
0	1	0	0	1	0	1	1	1
0	1	1	1	0	0	1	0	0
1	0	0	0	1	0	1	1	1
1	0	1	1	0	0	1	0	0
1	1	0	1	0	0	1	0	0
1	1	1	1	0	1	1	0	1

3.3 Analysis Procedure

- Logic Simulation - ViewDraw & ViewTrace



3.4 Design Procedure

- Procedure to design combinational circuits
  1) Determine required number of inputs & outputs & assign a letter symbol to each
  2) Derive the truth table
  3) Obtain simplified Boolean functions for each output
  4) Draw the logic diagram
  5) Verify the correctness of the design

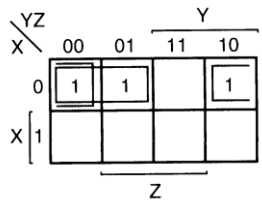
- Truth Table
  - n input variables: 2^n binary numbers
  - output functions give the exact definition of comb circuit simplified by method, such as algebraic manipulation, map method, computer programs

3.4 Design Procedure

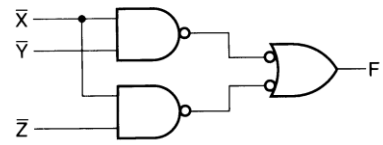
Ex 3.1) Design a combinational circuit with 3 inputs 1 output

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

(a) Truth table



(b) Map $F = \bar{X}\bar{Y} + \bar{X}\bar{Z}$



(c) Logic diagram

3.4 Design Procedure

- Code Converter
  - a circuit that translates info from one binary code to another

Ex 3.2) BCD to Excess-3 Code Converter

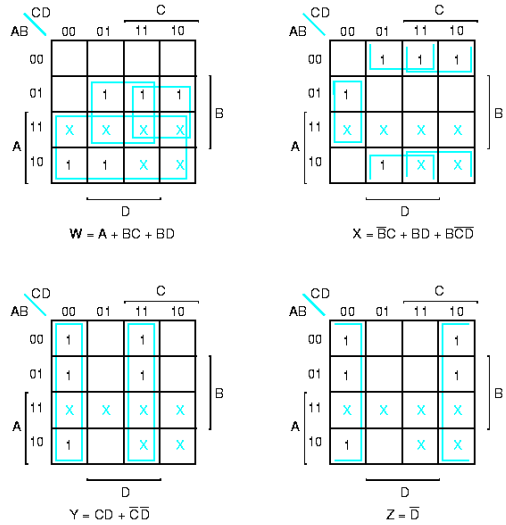
Excess-3 code: decimal digit + 3

desirable to implementing decimal subtraction

(Don't care conditions !!)

Decimal Digit	Input BCD				Output Excess-3			
	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

3.4 Design Procedure



3.4 Design Procedure

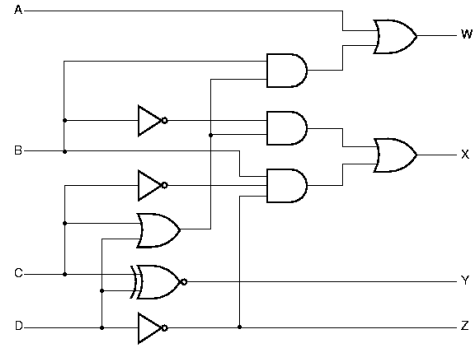
- two-level AND-OR logic diagram

  W = A + BC + BD = A + B(C + D)

  X = B'C + B'D + BC'D' = B'(C + D) + BC'D'

  Y = CD + C'D' = (C D)'

  Z = D'

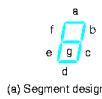


Logic Diagram

3.4 Design Procedure

Ex3.3) BCD to Seven-Segment Decoder

- LED (light emitting diodes), or LCD (liquid crystal display)
- accept a decimal digit in BCD
  & generate the appropriate outputs



(a) Segment designation

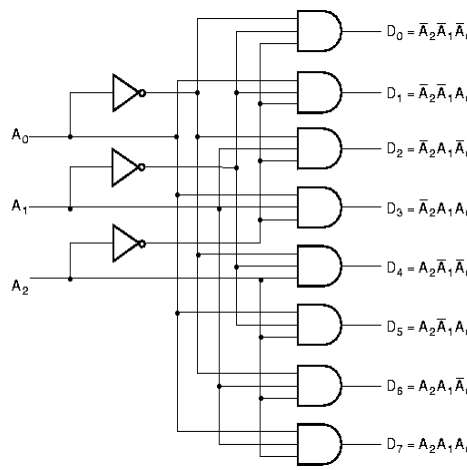

(b) Numeric designation for display

BCD Input				Seven-Segment Decoder						
A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
All other inputs				0	0	0	0	0	0	0

- outputs (a, b, c, d, e, f, g)
- 7 four-variable maps

3.5 Decoders

- Decoders
  - a binary code of n bits represents 2^n distinct elements
  - convert binary info of n coded inputs to 2^n (max) unique outputs
- n-to-m-line decoders ($m \leq 2^n$)
  - generate 2^n (or fewer) minterms of n input variables
  - 3-to-8-line decoder (3 inputs & 8 outputs)



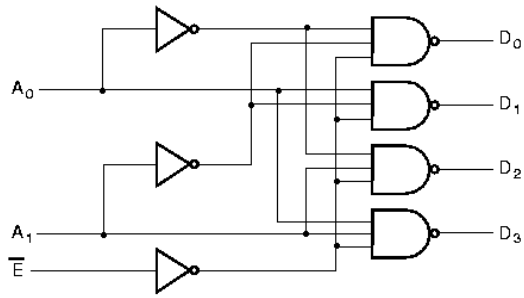
3.5 Decoders

A_2	A_1	A_0	D7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

3.5 Decoders

Ex) binary-to-octal conversion

2-to-4-line decoder with enable input constructed with NAND instead of AND gates operates with comp outputs and comp enable E



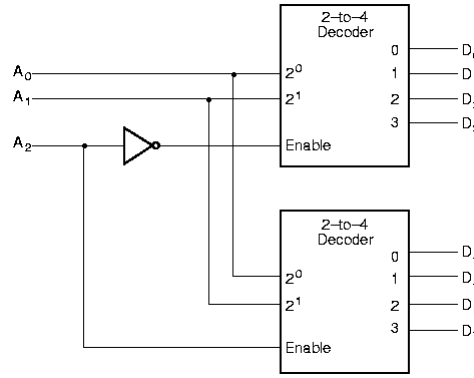
(a) Logic diagram

$\overline{E}$	A_1	A_0	D_0	D_1	D_2	D_3
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	X	X	1	1	1	1

(b) Truth table

3.5 Decoders

- Decoder Expansion
  - combine two or more small decoder w/ enable inputs
  - form a larger decoder
  - 3-to-8-line decoder with two 2-to-4-line decoders



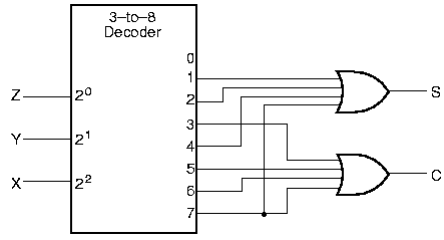
if A_2=0, upper is enabled;
if A_2=1, lower is enabled.

3.5 Decoders

- Combinational Circuit Implementation
  - a decoder provide 2^n minterms of n input variables
  - implementing a comb circuit with a decoder and OR gates requires Boolean functions are expressed as a sum of minterms

Ex 3.4) a binary adder

S(X,Y,Z) = Σ m(1,2,4,7); C(X,Y,Z) = Σ m(3,5,6,7)



3.6 Encoders

- perform the inverse operation of a decoder
- 2^n (or less) input lines and n output lines
- Octal-to-binary encoder

Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

$A_0 = D_1 + D_3 + D_5 + D_7$;

$A_1 = D_2 + D_3 + D_6 + D_7$;

$A_2 = D_4 + D_5 + D_6 + D_7$;

3.6 Encoders

- implemented with 3 OR gates
- only one input can be active at any given time
- more than 2 inputs are active
  - ☞ undefined !
  - ☞ priority for inputs

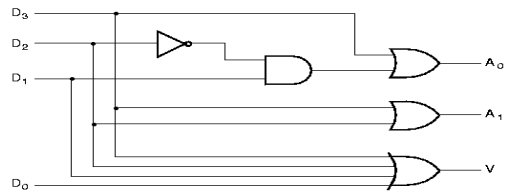
- Priority Encoder
  - 2 or more inputs are equal to 1 at the same time, an input with the highest priority will take precedence

Inputs				Outputs		
D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	X	X	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

3.6 Encoders

D_2D_3 \ D_0D_1	00	01	11	10
00				
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

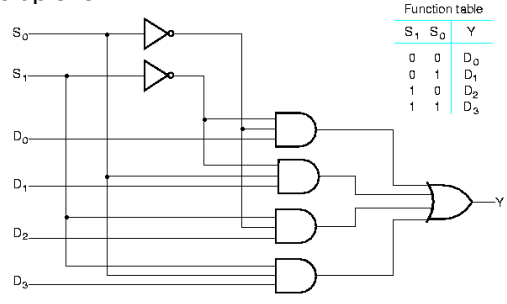
D_2D_3 \ D_0D_1	00	01	11	10
00			1	1
01				
11	1	1	1	1
10	1	1	1	1



$A_1 = D_2 + D_3$ $A_0 = D_3 + D_1D_2'$ $V = D_0 + D_1 + D_2 + D_3$

3.7 Multiplexers

- selects binary information from one of many input lines, and directs it to a single output line
- Normally, 2^n input lines and n selection variables
- 4-to-1-line multiplexer



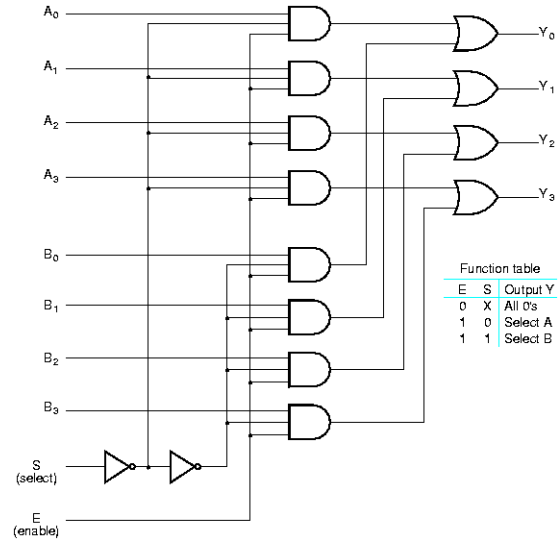
Function table

S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

- called *data selector* or *MUX*
- 2^n-to-1-line multiplexer is constructed from n-to-2^n decoder by adding 2^n input lines

3.7 Multiplexers

- Quadruple 2-to-1-Line Multiplexer
  - common selection & enable lines



3.7 Multiplexers

- Combinational Circuit Implementation
  - a decoder can be used to implement Boolean Functions by employing external OR gates
  - the minterms of a function are generated in a multiplexer by the circuit associated with the selection inputs
  - n variables with a multiplexer with n-1 selection inputs

    first (n-1) variables --> selection inputs

    remaining 1 variable --> data inputs

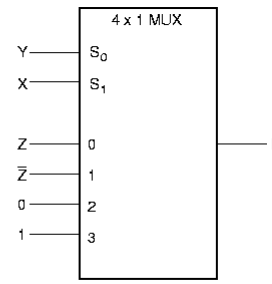
3.7 Multiplexers

Ex) F(X,Y,Z) = m(1,2,6,7)

-- implemented with a 4-to-1-line multiplexer

X	Y	Z	F	
0	0	0	0	F = Z
0	0	1	1	
0	1	0	1	F = $\overline{Z}$
0	1	1	0	
1	0	0	0	F = 0
1	0	1	0	
1	1	0	1	F = 1
1	1	1	1	

(a) Truth table

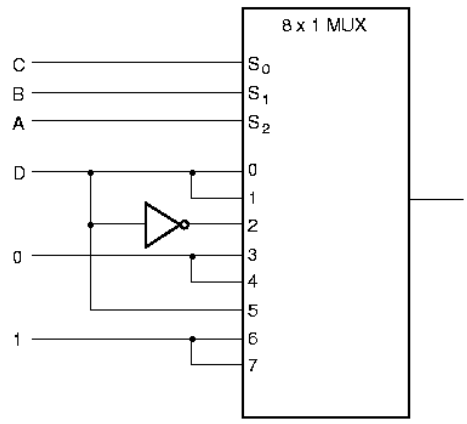


(b) Multiplexer implementation

- any Boolean function of n variables can be implemented with a multiplexer with n-1 selection inputs and 2^n-1 data inputs

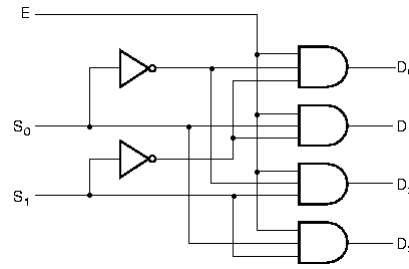
3.7 Multiplexers

A	B	C	D	F	
0	0	0	0	0	F = D
0	0	0	1	1	
0	0	1	0	0	F = D
0	0	1	1	1	
0	1	0	0	1	F = $\overline{D}$
0	1	0	1	0	
0	1	1	0	0	F = 0
0	1	1	1	0	
1	0	0	0	0	F = 0
1	0	0	1	0	
1	0	1	0	0	F = D
1	0	1	1	1	
1	1	0	0	1	F = 1
1	1	0	1	1	
1	1	1	0	1	F = 1
1	1	1	1	1	



3.7 Multiplexers

- Demultiplexer
  - perform the inverse operation of a multiplexer
  - receives information from a single line and transmits it to one of 2^n possible output lines
  - 1-to-4-line demultiplexer
    - the input E has a path to all 4 outputs, selected by two selection lines S1 and S0
  - a decoder with enable input is referred to as decoder /demultiplexer
  - identical to a 2-to-4 line decoder with enable input

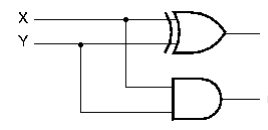


3.8 Binary Adders

- Arithmetic Circuit
  - a combinational circuit for arithmetic operations such as addition, subtraction, multiplication, and division with binary numbers or decimal numbers in a binary code
- Addition of 2 binary inputs, *'Half Adder'*
  - 0+0=0, 0+1=1, 1+0=1, 1+1 = 10

(Inputs)		(Outputs)	
X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

S = X'Y + XY' = X ⊕ Y;
C = XY

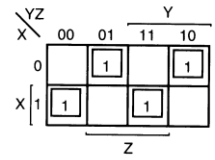


3.8 Binary Adders

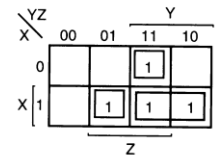
- Addition of 3 binary inputs, *'Full Adder'*

(Inputs)			(Outputs)	
X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Truth Table of Full Adder

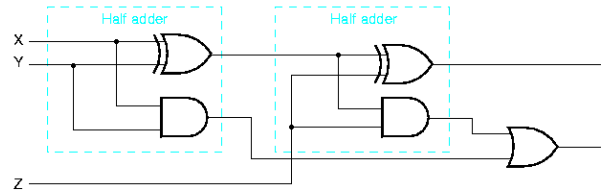


$S = \bar{X}\bar{Y}Z + \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z} + XYZ$
$= X \oplus Y \oplus Z$



$C = XY + XZ + YZ$
$= XY + Z(X\bar{Y} + \bar{X}Y)$
$= XY + Z(X \oplus Y)$

Logic Diagram of Full Adder

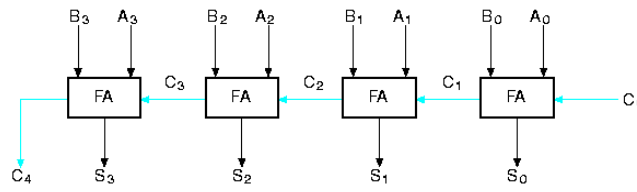


3.8 Binary Adders

- Binary Ripple Carry Adder
  - sum of two n-bit binary numbers in parallel
  - 4-bit parallel adder

  A = 1011, B = 0011

Input carry	0	1	1	0
Augend A	1	0	1	1
Addend B	0	0	1	1
Sum S	1	1	1	0
Output carry	0	0	1	1



3.8 Binary Adders

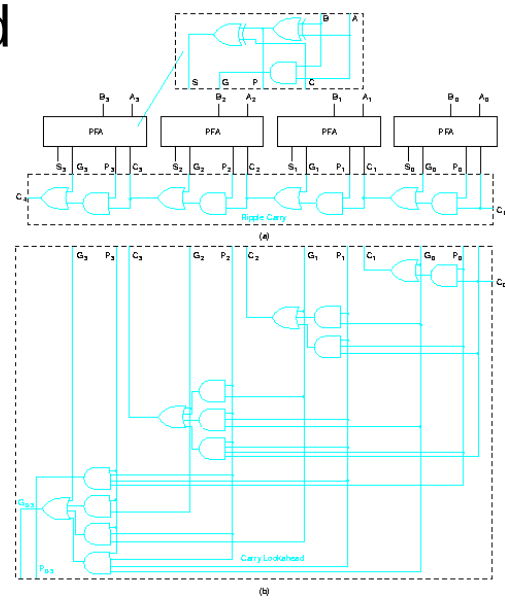
- Carry Lookahead Adder
  - The ripple carry adder has a long circuit delay
    - the longest delay: 2 n + 2 gate delay
      - ☞ *Carry Lookahead Adder*
    - reduced delay at the price of complex hardware
  - a new logic hierarchy

    P_i: propagate function

    G_i: generate function

3.8 Binary Add

Development of Carry Lookahead Adder



3.9 Binary Subtraction

- Subtraction

Borrowed into	1100
Minuend	10011
Subtrahend	-11110
Difference	10101
Correct Difference	-01011

- a borrow occurs into the most significant position

  M - N ==> M - N + 2^n

  ==> 2^n - (M - N + 2^n) = N - M

- 1) Subtract the subtrahend N from the minuend M
  2) If no end borrow occurs, then M $\geq$ N
  3) If an end borrow occurs,
  then N-M is subtracted from 2^n
  & minus sign is appended to the result
- Subtraction of a binary number from 2^n
  - "2's complement form"

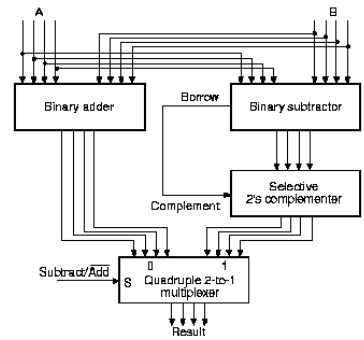
3.9 Binary Subtraction

Borrowed into	10011110
Minuend	01100100
Subtrahend	-10010110
Initial Result	11001110

End borrow of *1* implies correction

2^8	100000000
Subtrahend	- 11001110
Initial Result	- 00110010

- Block Diagram of Binary Adder-Subtractor



3.9 Binary Subtraction

- Complements
  - 2 types:
    - radix complement: r's complement
    - diminished radix complement: (r-1)'s complement
      - 2's & 1's for binary numbers
      - 10's & 9's for decimal numbers
  - 1's complement of N (binary number): $(2^n - 1) - N$
    - 1's comp of 1011001 ==> 0100110
    - 1's comp of 0001111 ==> 1110000

3.9 Binary Subtraction

- 2's complement of N: $2^n - N$ for N != 0, 0 for N = 0
  - add 1 to the 1's complement
  - 2's comp of 101100 ==> 010011 + 1 ==> 010100
  - leaving all least significant 0's and the first unchanged then replacing 1's with 0's, 0's with 1's
  - 2's comp of 1101100 ==> 0010100
  - 2's complement of N is $2^n – N$
    & the complement of the complement is $2^n - (2^n-N) = N$

3.9 Binary Subtraction

- Subtraction with Complements
  - (M - N)
    1) add 2's comp of the subtrahend N to the minuend M
    $M + (2^n - N) = M - N + 2^n$
    2) if $M \geq N$, the end cary is discarded
    3) if $M < N$, the result is $2^n - (N - M)$
    take the 2's complement of the sum & place a minus sign
  - avoid *overflow problem* to accomodate the sum

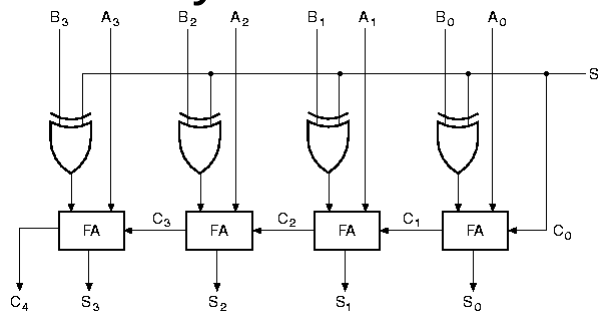
- Ex3.6 X = 1010100, Y = 1000011

X =	1010100
2's comp of Y =	0111101
Sum =	10010001
Discard end carry 2^7 =	- 10000000
Answer X - Y =	0010001

Y =	1000011
2's comp of X =	0101100
Sum =	1101111

Answer Y - X = -0010001

3.10 Binary Adder-Subtractors



- A - B = A + (-B) in 2's complement form
- with exclusive-OR gate (B⊕0=B; B⊕1=B')
  adder if S = 0; subtractor if S = 1

3.10 Binary Adder-Subtractors

- Signed Binary Numbers
  - sign bit: 0 for positive numbers
    1 for negative numbers
  - -9 (=-1001) using 8 bits
    signed-magnitude representation: 10001001
    signed 1's complement representation: 1111 0110
    signed 2's complement representation: 1111 0111
  - positive numbers are identical
    signed-magnitude -7 ~ -1, -0, 0, 1 ~ 7 (2 zeros)
    signed 1's comp -7 ~ -1, -0, 0, 1 ~ 7 (2 zeros)
    signed 2's comp -8 ~ -1, 0, 1 ~ 7
  - signed 1's comp : useful as a logical operation
    signed 2's comp : popular in use

3.10 Binary Adder-Subtractors

Decimal	Signed 2's Complement	Signed 1's Complement	Signed Magnitude
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
−0	—	1111	1000
−1	1111	1110	1001
−2	1110	1101	1010
−3	1101	1100	1011
−4	1100	1011	1100
−5	1011	1010	1101
−6	1010	1001	1110
−7	1001	1000	1111
−8	1000	—	—

Signed Binary Number

3.10 Binary Adder-Subtractors

Ex3.8

+6 00000110	-6 11111010	+6 00000110	-6 11111010
+13 00001101	+13 00001101	-13 11110011	-13 11110011
+19 00010011	+7 00000111	-7 11111001	-19 11101101

- (±A) - (+B) = (±A) + (-B)
  (±A) - (-B) = (±A) + (+B)

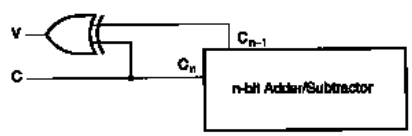
Ex3.9

-6	11111010	11111010	-6	00000110	00000110
-(-13)	-11110011	00001101	-(-13)	11111010	00001101
+7		00000111	19		00010011

3.10 Binary Adder-Subtractors

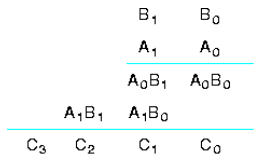
(Ex)

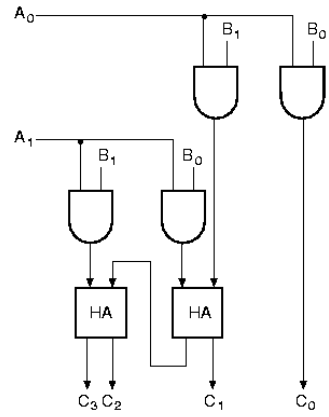
Carries:	0	1	Carries:	1	0
+70	0	1000110	-70	1	0111010
+80	0	1010000	-80	1	0110000
+150	1	0010110	-150	0	1101010



3.11 Binary Multipliers

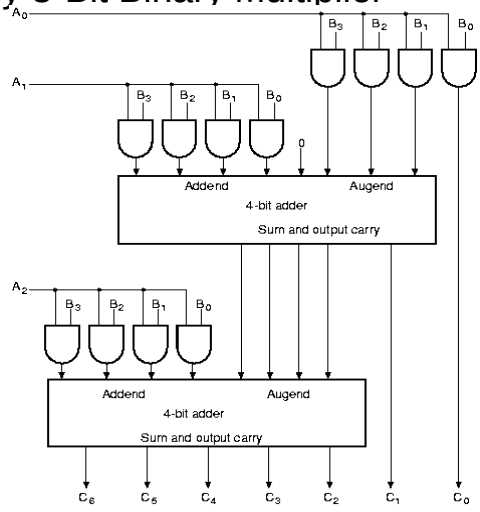
- a 2-Bit by 2-Bit Binary Multiplier





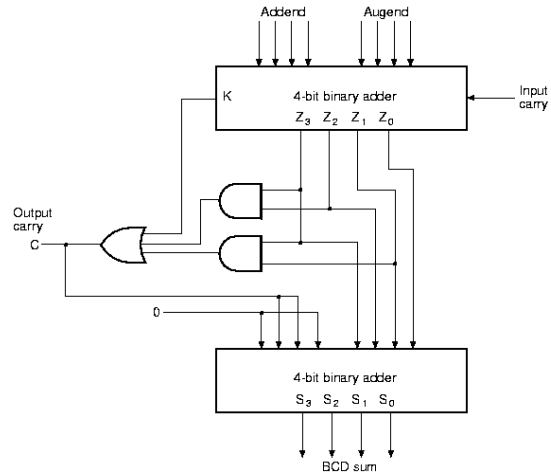
3.11 Binary Multipliers

- a 4-Bit by 3-Bit Binary Multiplier



3.12 Decimal Arithmetic

- BCD Adder
  - binary numbers 1010 ~ 1111 need to correct
  - 1010, 1011, 1100, 1101, 1110, 1111
  - if $C = K + Z_1Z_3 + Z_2Z_3$, add 0110 to the binary sum



  - to add two n decimal digits needs n BCD adders

3.12 Decimal Arithmetic

- Use of Complements in Decimal
  - 9's complement if 546700 is 999999 - 546700 = 453299
  - 10's complement if 546700 is 1000000 - 546700 = 453300
  - 10's complement if 234500 is 765500

Ex3.10 72532 - 3250

M =	72532
10's comp of N =	+ 96750
M =	169282
Discard end carry	- 100000
	69282

Ex3.11 3250 - 72532

M =	03250
10's comp of N =	+ 27468
Sum =	30718