COMBINATIONAL LOGIC

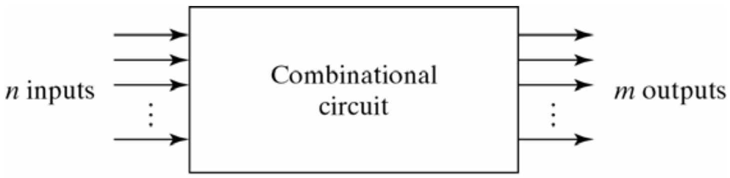


Fig. 4-1 Block Diagram of Combinational Circuit

- Output Changes Based on Input Changes
- Output can Change at Any Time
- Other Type is Sequential – We will study These Later

Two Types of Problems

- Analysis
  - Given a Circuit Describe Behavior
- Design
  - Given Behavior Determine a Circuit
- Both are Really Synthesis Problems
  - Analysis Usually Means Synthesizing from Low-level to Higher-level Model
  - Design Usually Means Transforming High-level Description (like specification) into Low-level Model (like netlist or schematic diagram)

COMB. LOGIC ANALYSIS

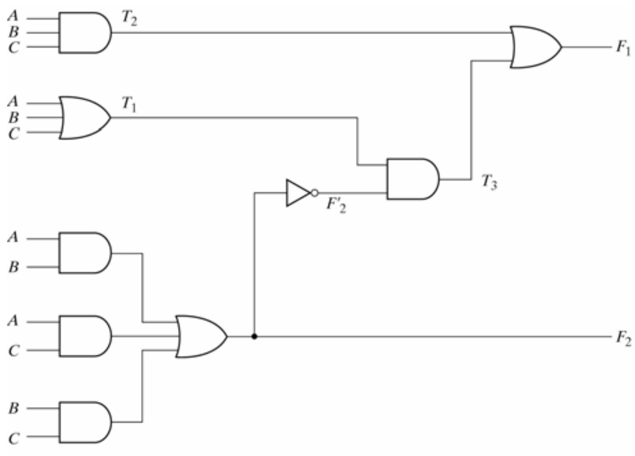


Fig. 4-2 Logic Diagram for Analysis Example

ANALYSIS TECHNIQUE

- Label Intermediate Nets
- Work Back to Inputs
- Use Substitution to Generate Equations
- Equations Evaluated Manually or By HDL
- Could Also Build HDL Netlist and Simulate

$$F_2 = AB + AC + BC$$
$$T_1 = A + B + C$$
$$T_2 = ABC$$
$$T_3 = \overline{F}_2 T_1$$
$$F_1 = T_3 + T_2$$

ANALYSIS TECHNIQUE (cont)

$$F_1 = T_3 + T_2 = \overline{F}_2 T_1 + ABC = (\overline{AB + AC + BC})(A + B + C) + ABC$$
$$= (\overline{A} + \overline{B})(\overline{A} + \overline{C})(\overline{B} + \overline{C})(A + B + C) + ABC$$
$$= (\overline{A} + \overline{B}\overline{C})(A\overline{B} + A\overline{C} + B\overline{C} + \overline{B}C) + ABC$$
$$= \overline{A}B\overline{C} + \overline{A}\overline{B}C + A\overline{B}\overline{C} + ABC$$

A	B	C	F_2	F'_2	T_1	T_2	T_3	F_1
0	0	0	0	1	0	0	0	0
0	0	1	0	1	1	0	1	1
0	1	0	0	1	1	0	1	1
0	1	1	1	0	1	0	0	0
1	0	0	0	1	1	0	1	1
1	0	1	1	0	1	0	0	0
1	1	0	1	0	1	0	0	0
1	1	1	1	0	1	1	0	1

DESIGN PROBLEM

1) From specifications of circuit, determine required number of inputs/outputs and assign a symbol to each.
2) Derive a truth table defining relationship between inputs and outputs.
3) Obtain simplified Boolean functions for each output as a function of input variables.
4) Draw the logic diagram and verify correctness of design.

CODE CONVERSION EXAMPLE

SPECIFICATION

Design a circuit that receives a single BCD digit as input and produces the corresponding excess-3 digit as output.

CODE CONVERSION EXAMPLE

SPECIFICATION

Design a circuit that receives a single BCD digit as input and produces the corresponding excess-3 digit as output.

TRUTH TABLE

Input BCD				Output Excess-3 Code			
A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

CODE CONVERSION EXAMPLE

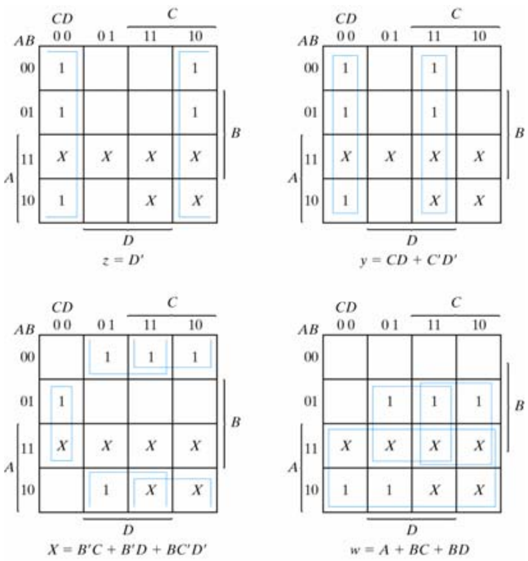


Fig. 4-3 Maps for BCD to Excess-3 Code Converter

CODE CONVERSION EXAMPLE

From the Maps we have:

$$z = \overline{D}$$

$$y = CD + \overline{C}\,\overline{D}$$

$$x = \overline{B}C + \overline{B}D + B\overline{C}\,\overline{D}$$

$$w = A + BC + BD$$

Using Algebra, Manipulate to find Common Gates:

$$z = \overline{D}$$

$$y = CD + \overline{C}\,\overline{D} = CD + (\overline{C + D})$$

$$x = \overline{B}C + \overline{B}D + B\overline{C}\,\overline{D} = \overline{B}(C + D) + B\overline{C}\,\overline{D}$$

$$= \overline{B}(C + D) + B(\overline{C + D})$$

$$w = A + BC + BD = A + B(C + D)$$

CODE CONVERSION EXAMPLE

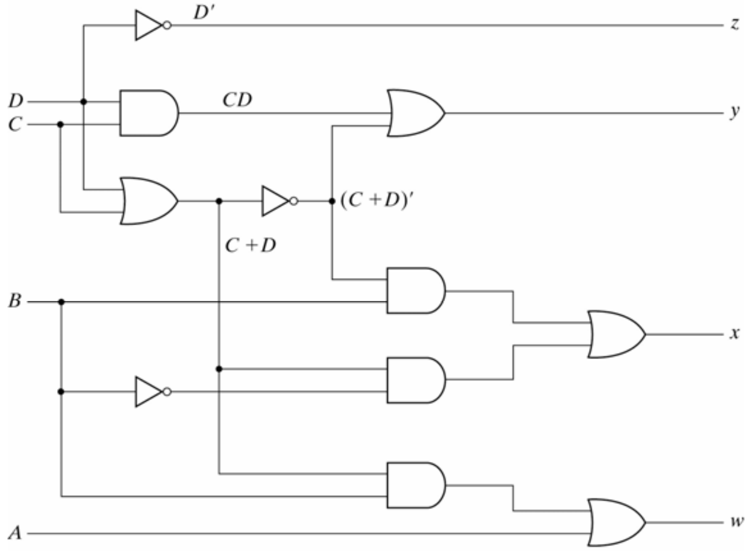


Fig. 4-4 Logic Diagram for BCD to Excess-3 Code Converter

BINARY ADDER-SUBTRACTOR

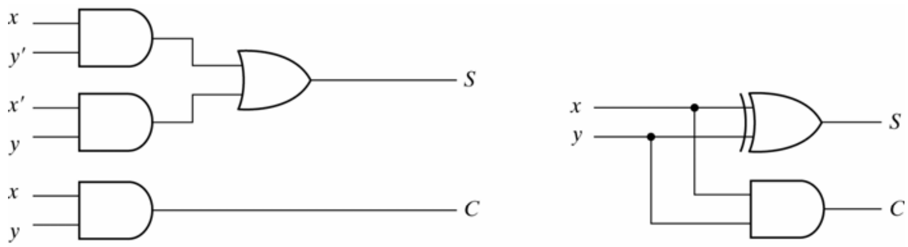
- Half-Adder – Two Input Bits, Sum and Carry Output
- Full-Adder – Two inputs AND Carry-in with Sum and Carry Output
- Subtractor Circuits – Typically Implemented using Adders and a Complement Representation

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$S = \overline{x}y + x\overline{y}$$

$$C = xy$$

HALF ADDER CIRCUITS



(a) $S = xy' + x'y$
$C = xy$

(b) $S = x \oplus y$
$C = xy$

Fig. 4-5 Implementation of Half-Adder

FULL ADDER

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

FULL ADDER

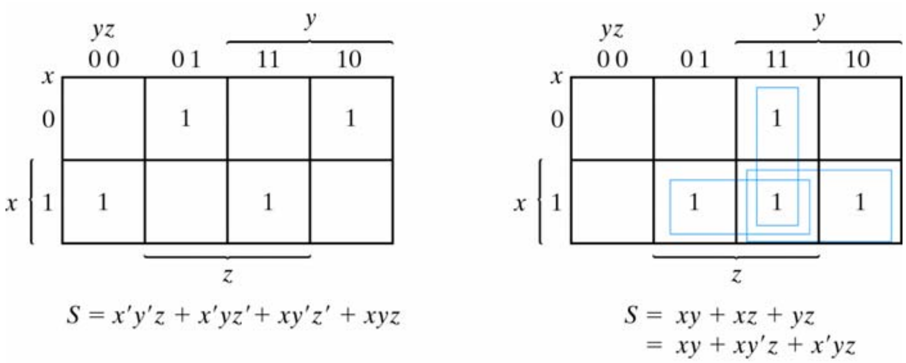


Fig. 4-6 Maps for Full Adder

FULL ADDER CIRCUITS

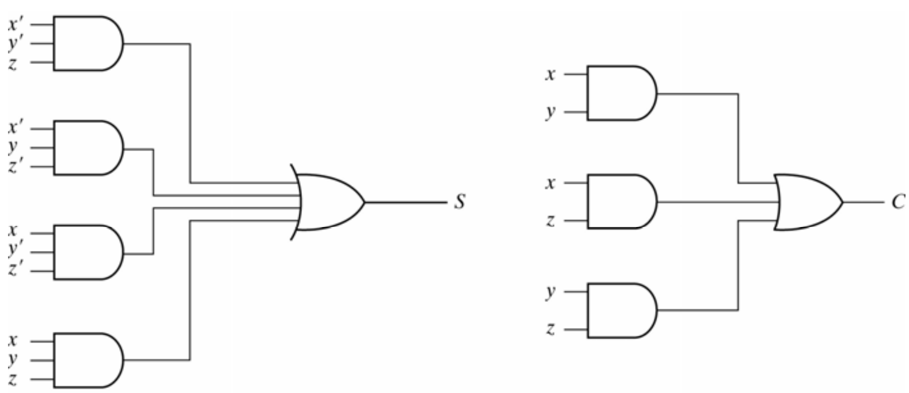


Fig. 4-7 Implementation of Full Adder in Sum of Products

FULL ADDER CIRCUIT

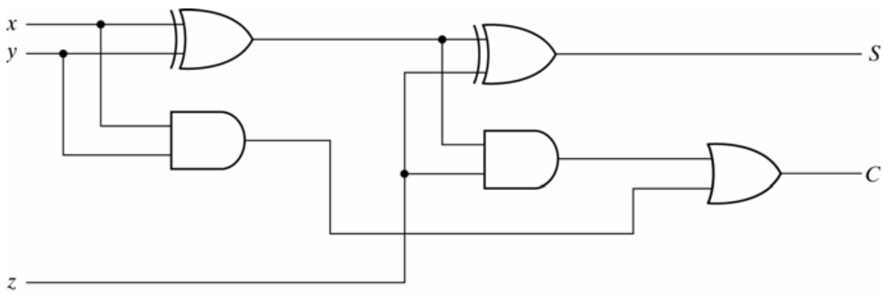


Fig. 4-8 Implementation of Full Adder with Two Half Adders and an OR Gate

RIPPLE-CARRY ADDER

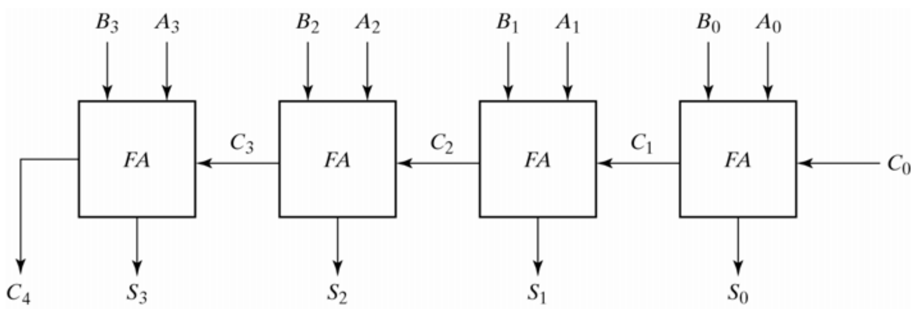


Fig. 4-9 4-Bit Adder

PROPAGATE AND GENERATE SIGNALS

- Function of Two Data Inputs Only
- Propagate Means Carry-in is also Carry-out
- Generate Means Carry-out will be a Logic-1

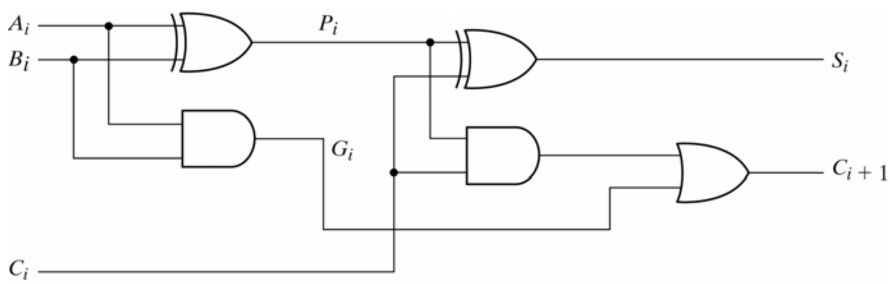


Fig. 4-10 Full Adder with P and G Shown

CARRY LOOKAHEAD EQUATIONS

- Generate Carry-outs Using Data Inputs Only
- Avoids Ripple – Faster Circuit

$$P_i = A_i \oplus B_i$$
$$G_i = A_i B_i$$
$$S_i = P_i \oplus C_i$$
$$C_{i+1} = G_i + P_i C_i$$

- Use Recursive Expression to Expand

$$C_0 = \text{input carry}$$
$$C_1 = G_0 + P_0 C_0$$
$$C_2 = G_1 + P_1 C_1 = G_1 + P_1(G_0 + P_0 C_0) = G_1 + P_1 G_0 + P_1 P_0 C_0$$
$$C_3 = G_2 + P_2 C_2 + G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$

CARRY LOOKAHEAD GENERATOR

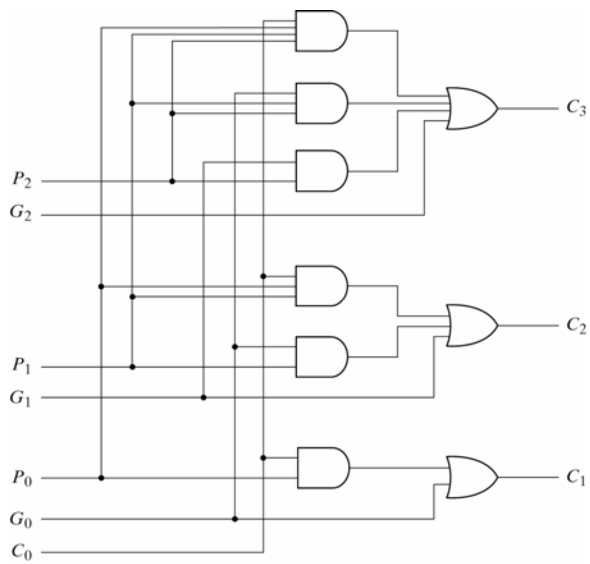


Fig. 4-11 Logic Diagram of Carry Lookahead Generator

CARRY LOOKAHEAD ADDER

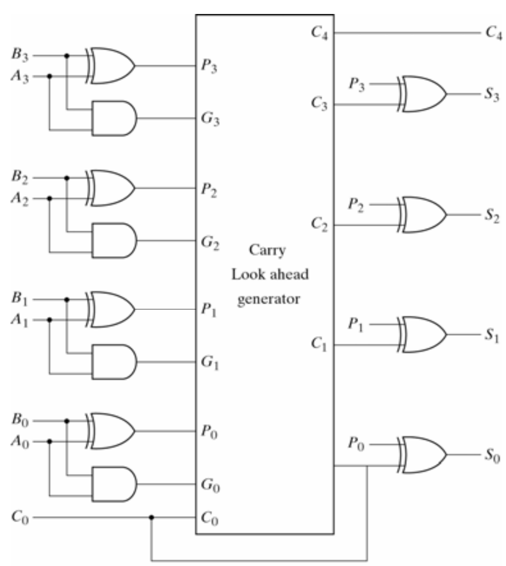


Fig. 4-12 4-Bit Adder with Carry Lookahead

SELECTABLE ADDER/SUBTRACTOR

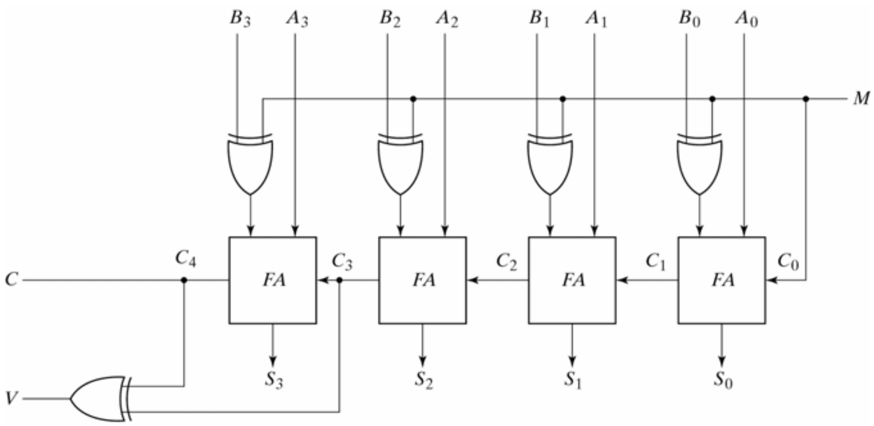


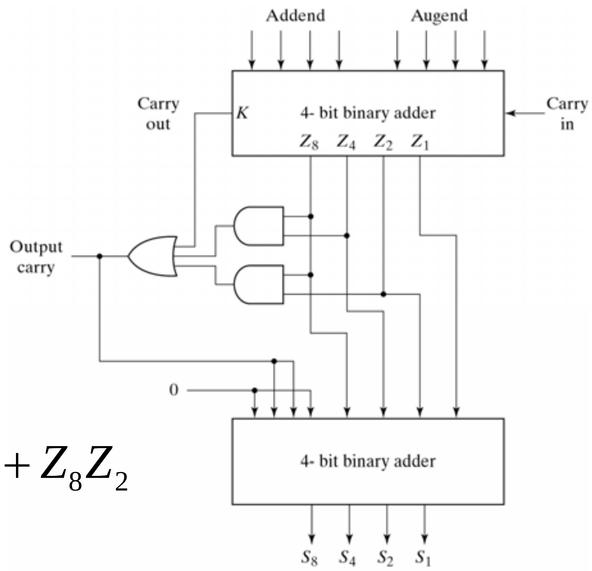
Fig. 4-13 4-Bit Adder Subtractor

DECIMAL (BCD) ADDER

Remember the "Add-6" Rule Discussed Earlier in Class

Binary Sum					BCD Sum					Decimal
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

DECIMAL (BCD) ADDER



$$C = K + Z_8Z_4 + Z_8Z_2$$

Fig. 4-14 Block Diagram of a BCD Adder

BINARY MULTIPLIER

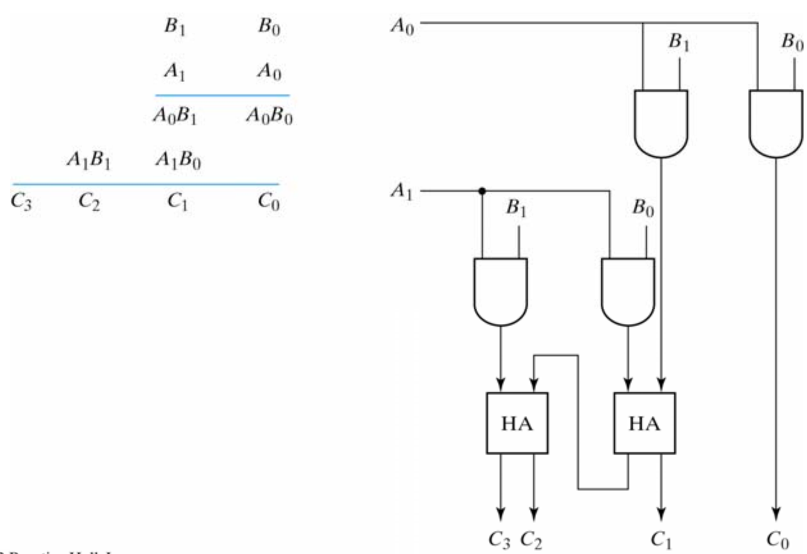


Fig. 4-15 2-Bit by 2-Bit Binary Multiplier

BINARY MULTIPLIER

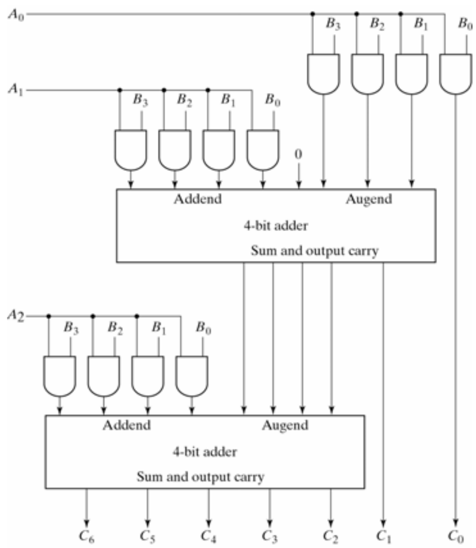


Fig. 4-16 4-Bit by 3-Bit Binary Multiplier