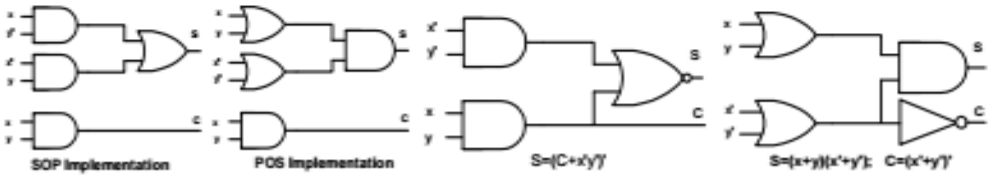
LECTURE 7: DIGITAL LOGIC CIRCUIT DESIGN

Half Adder

Design a logic circuit to add two bits and produce a sum bit and a carry bit. Two inputs and two outputs are needed. Let us call the inputs x and y, and the outputs S and C.

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

$$S = x \oplus y = x'y + xy' = (x'y' + xy)'$$
$$= (x + y)(x' + y')$$
$$C = xy = (x' + y')'$$
$$\therefore S = (C + x'y')'$$



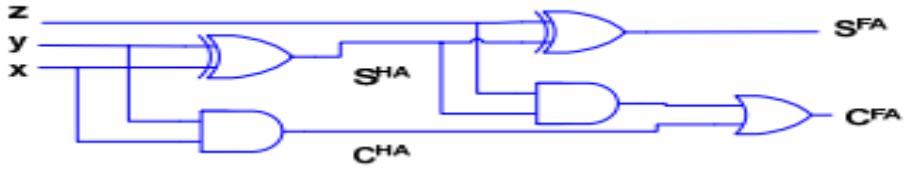
Full-Adder

Design a logic circuit that adds two bits and a carry in bit and produce a sum bit and a carry out bit. Three inputs and two outputs are needed. Let us call the inputs x y and z, and the outputs S and C.

x	y	z	S	C
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

$$S = \sum(1,2,4,7)$$
$$= x'y'z + x'yz' + xy'z' + xyz$$
$$= (x'y' + xy)z + (x;y + xy')z'$$
$$= (x \oplus y)'z + (x \oplus y)z'$$
$$= (x \oplus y) \oplus z$$

$$C = \sum(3,5,6,7)$$
$$= xy + xz + yz$$
$$= x'yz + xy'z + xy$$
$$= (x'y + xy')z + xy$$
$$= (x \oplus y)z + xy$$



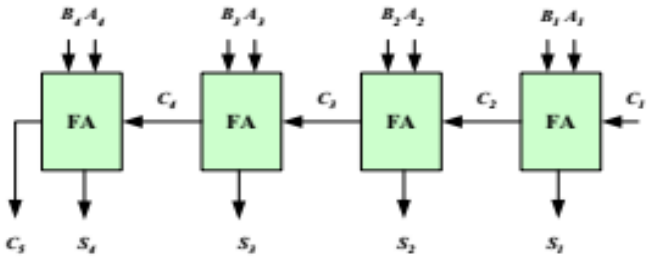
LECTURE 7: DIGITAL LOGIC CIRCUIT DESIGN

4-Bit Binary Adder

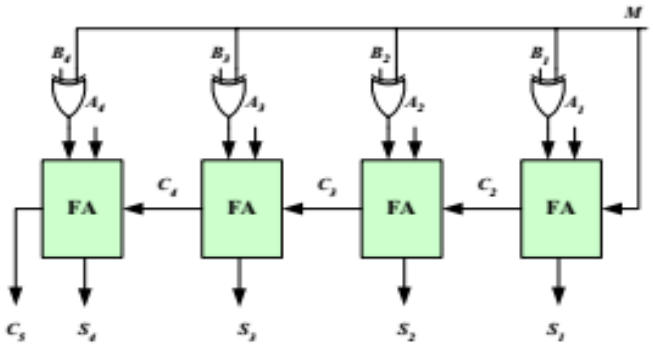
To add two 4-bit binary number, we proceed as shown in the table. The table shows the role of carry-in and carry-out.

0	1	1	0	Ci
1	0	1	1	Ai
0	0	1	1	Bi
1	1	1	0	Si
0	0	1	1	C_{i+1}

4-bit binary parallel adder can be implemented in integrated circuit form by cascading 4 full adders as shown below. The disadvantage of this adder is the possible slowing down of the addition due to the carry propagation time.



The 4-bit parallel adder can be modified to work as 4-bit parallel adder/subtractor b including 4 exclusive-OR gates to provide the 1's complement of B and adding 1 from the M input to make it the 2's complement.



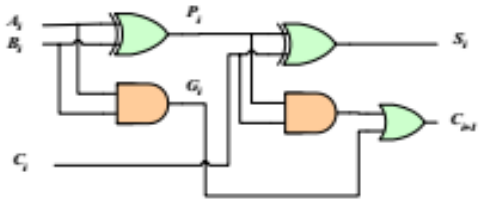
LECTURE 7: DIGITAL LOGIC CIRCUIT DESIGN

Carry Propagation and the look-ahead carry circuit

The carry propagate (Pi) and carry generate (Gi) variables are shown on the full adder logic circuit. The carries C1, C2, and C3 can be expressed in SOP form as functions of C0 and the different (Pi) and (Gi) as follows:

$$P_i = A_i \oplus B_i \quad , \quad G_i = A_i B_i$$

$$S_i = P_i \oplus C_i \quad , \quad C_{i+1} = G_i + P_i C_i$$



Therefore:

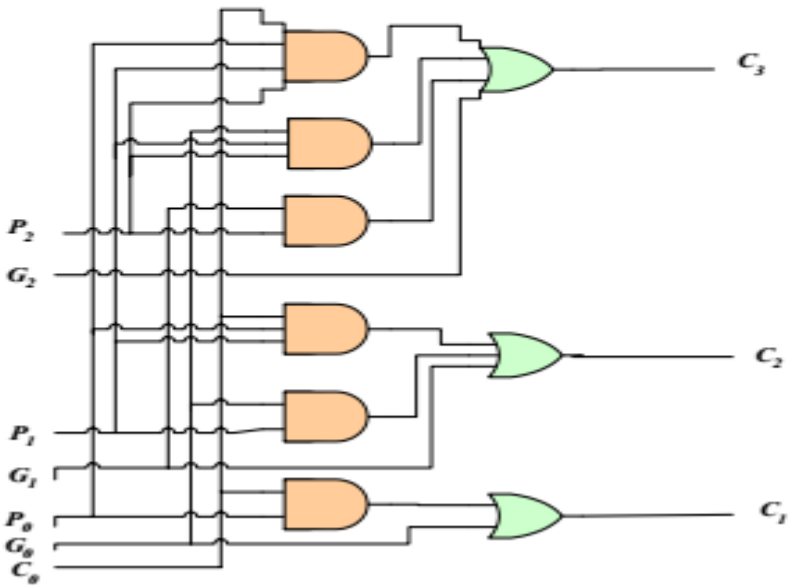
$$C_1 = G_0 + P_0 C_0$$

$$C_2 = G_1 + P_1 C_1 = G_1 + P_1(G_0 + P_0 C_{10}) = G_1 + P_1 G_0 + P_1 P_0 C_0$$

Similarly:

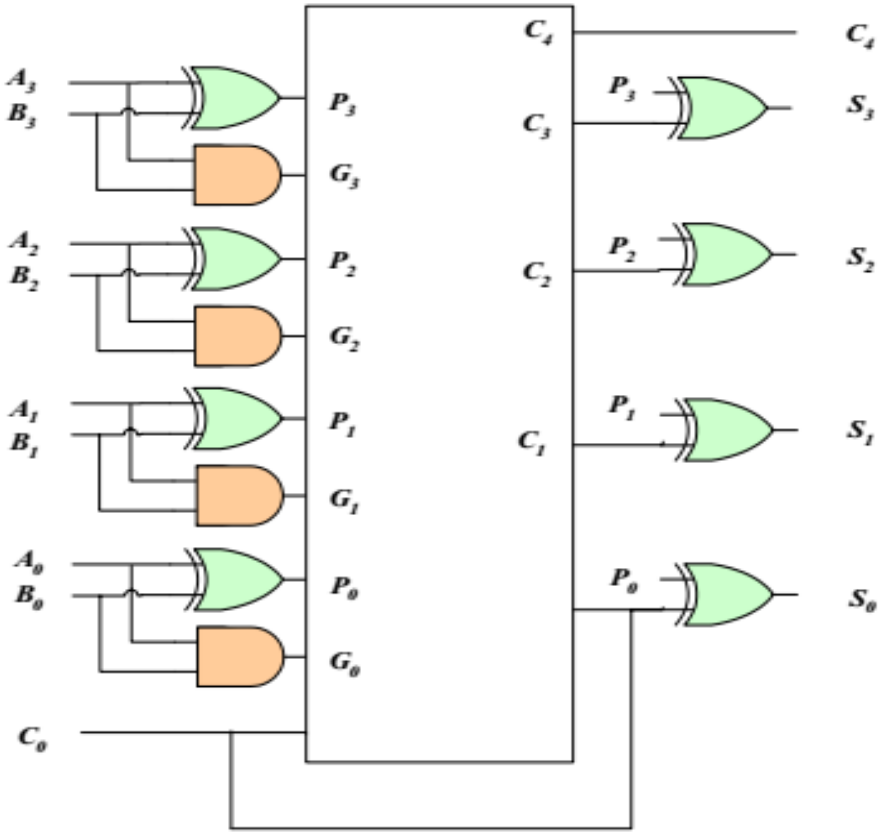
$$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$

The logic diagram of the look-ahead generator is implemented in a two level form as shown in the following logic circuit.



LECTURE 7: DIGITAL LOGIC CIRCUIT DESIGN

The 4-Bit adder with the carry look-ahead circuit is implemented as shown in the following circuit.



LECTURE 7: DIGITAL LOGIC CIRCUIT DESIGN

Overflow

Overflow is defined as the situation when two N-digit numbers are added and the sum occupies (N+1) digits. This situation occurs when adding binary numbers as follows:

1. **An end carry is generated when adding two N-bit unsigned numbers.**
2. **The carry-in and carry-out bits are different when adding two N-bit signed binary numbers.**

Example: Suppose we are adding +70 and +80 using 8-bits in signed 2's complement form.

+70 → 01000110 and +80 → 01010000

The carry 0 1

+70	**0 1 0 0 0 1 1 0**
+80	**0 1 0 1 0 0 0 0**
+150	**1 0 0 1 0 1 1 0**

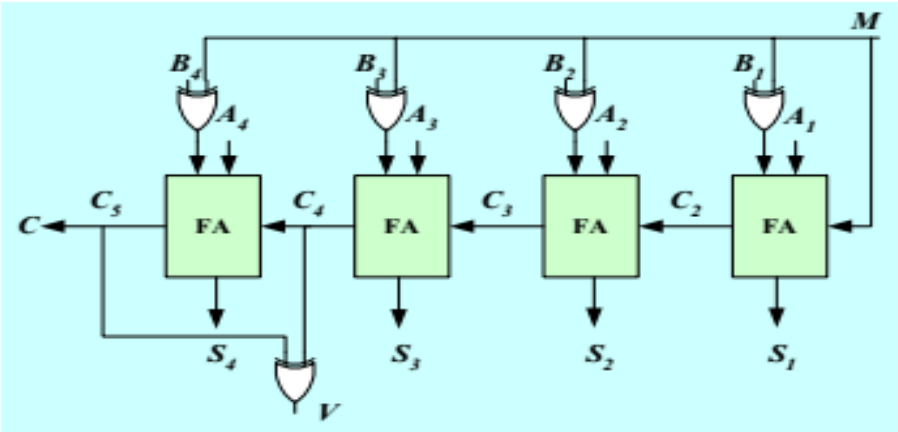
The same overflow occurs when adding (-70) + (-80)

The carry 1 0

-70	**1 0 1 1 1 0 1 0**
-80	**1 0 1 1 0 0 0 0**
-150	**0 1 1 0 1 0 1 0**

An overflow can be detected by observing the carry-in and the carry-out of the sign bit. If we apply them to an exclusive-or gate then the output is one when overflow occurs.

LECTURE 7: DIGITAL LOGIC CIRCUIT DESIGN



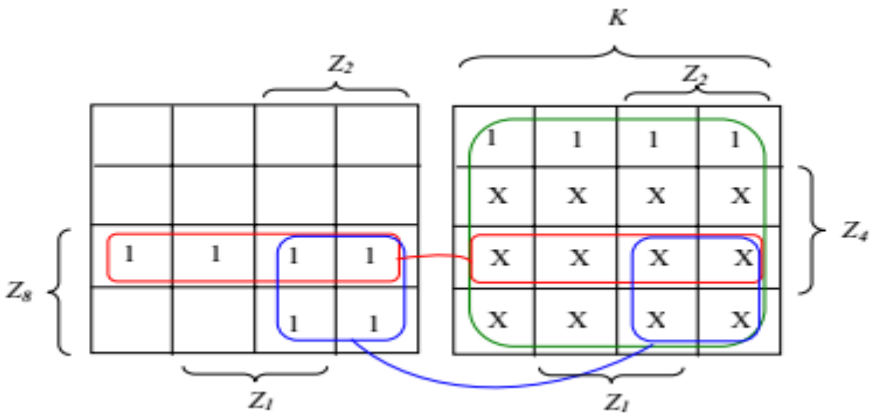
Decimal Adder

This is also called a BCD adder. In order to add two decimal digits with a possible carry in of one, then the maximum sum is 19. The following table shows the sum when performed in binary and compared to the sum when performed in BCD. In both cases five outputs are needed.

Dec.	Binary sum					BCD sum				
	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
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	1	1
4	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	1	0	0	1
10	0	1	0	1	0	1	0	0	0	0
11	0	1	0	1	1	1	0	0	0	1
12	0	1	1	0	0	1	0	0	1	0
13	0	1	1	0	1	1	0	0	1	1
14	0	1	1	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
18	1	0	0	1	0	1	1	0	0	0
19	1	0	0	1	1	1	1	0	0	1

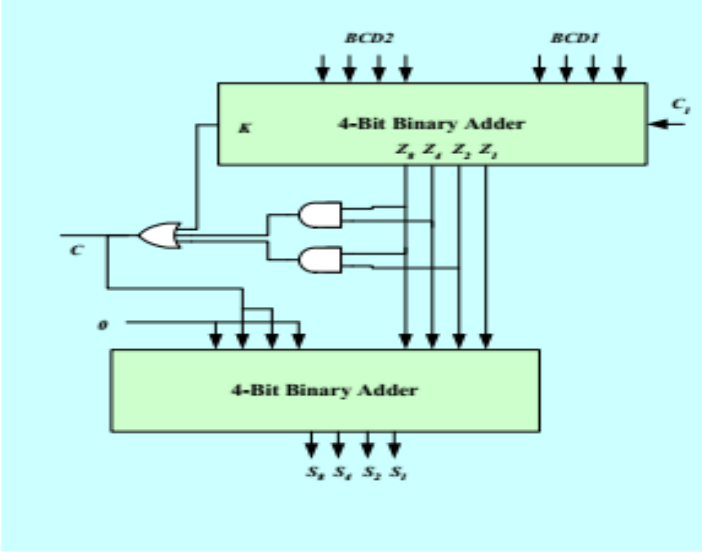
LECTURE 7: DIGITAL LOGIC CIRCUIT DESIGN

Inspecting the table reveals that a correction in the sum is needed when the sum is greater than 9. The correction is adding 6 to the sum. The BCD adder will then consist of the 4-bit binary adder. A second 4-bit binary adder is needed to add 6 to the sum when it is greater than 9. The required logic circuit needed to detect if correction is needed can be obtained by inspecting the table. A second method is to find a simplified expression for the carry out C of the five variables K, Z8, Z4, Z2, and Z1. Minterms m20 to m31 are considered don't care conditions.



$\therefore C = K + Z_8 Z_4 + Z_8 Z_2$.

The logic circuit will be as follows:



Magnitude Comparator

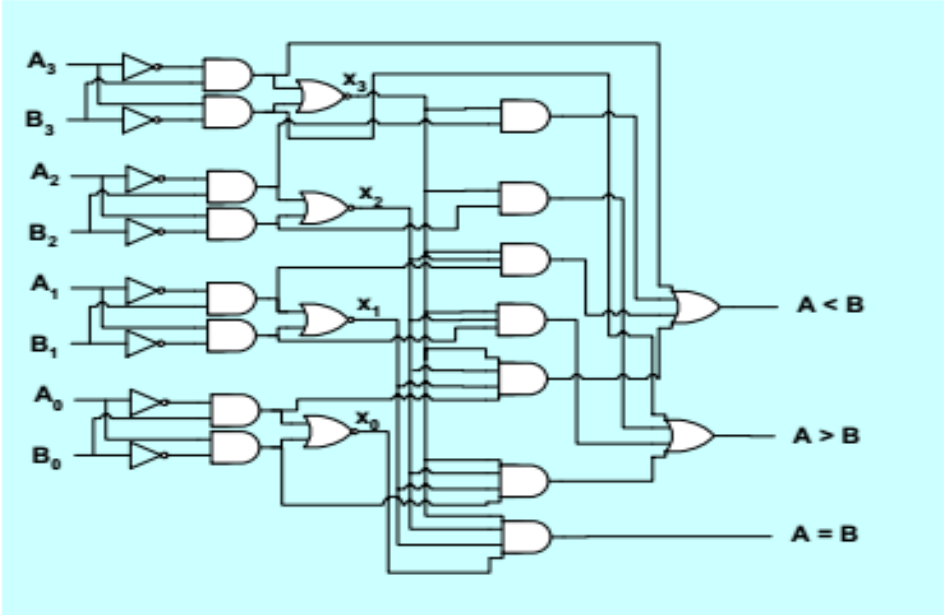
LECTURE 7: DIGITAL LOGIC CIRCUIT DESIGN

To compare two 4-bit numbers ($A_3\ A_2\ A_1\ A_0$) and ($B_3\ B_2\ B_1\ B_0$), we have to design a circuit with eight inputs and three outputs. The outputs are:

- A = B
- A > B
- A < B

A better method to design this circuit is to follow the systematic way of comparison, where we compare each pair of bits starting from the most significant bit. If all pairs are equal then A=B. If we find a difference in the compared bits (i.e. one is 1 and the other is 0), then the number containing the 1 is larger. This leads to the following three Boolean functions

1. $(A=B) = x_3x_2x_1x_0$, where $x_i = A_iB_i + A_i'B_i'$
2. $(A>B) = A_3B_3' + x_3A_2B_2' + x_3x_2A_1B_1' + x_3x_2x_1A_0B_0'$
3. $(A<B) = A'_3B_3 + x_3A_2'B_2 + x_3x_2A_1'B_1 + x_3x_2x_1A_0'B_0$



Binary Multiplier

To multiply two 2-bit binary numbers B1 B0 and A1 A0, we may use half adders and AND gates.

LECTURE 7: DIGITAL LOGIC CIRCUIT DESIGN

$$
\begin{array}{cccc}
 & & B_1 & B_0 \\
 & & A_1 & A_0 \\
\hline
 & & A_0B_1 & A_0B_0 \\
 & A_1B_1 & A_1B_0 & \\
\hline
C_3 & C_2 & C_1 & C_0
\end{array}
$$