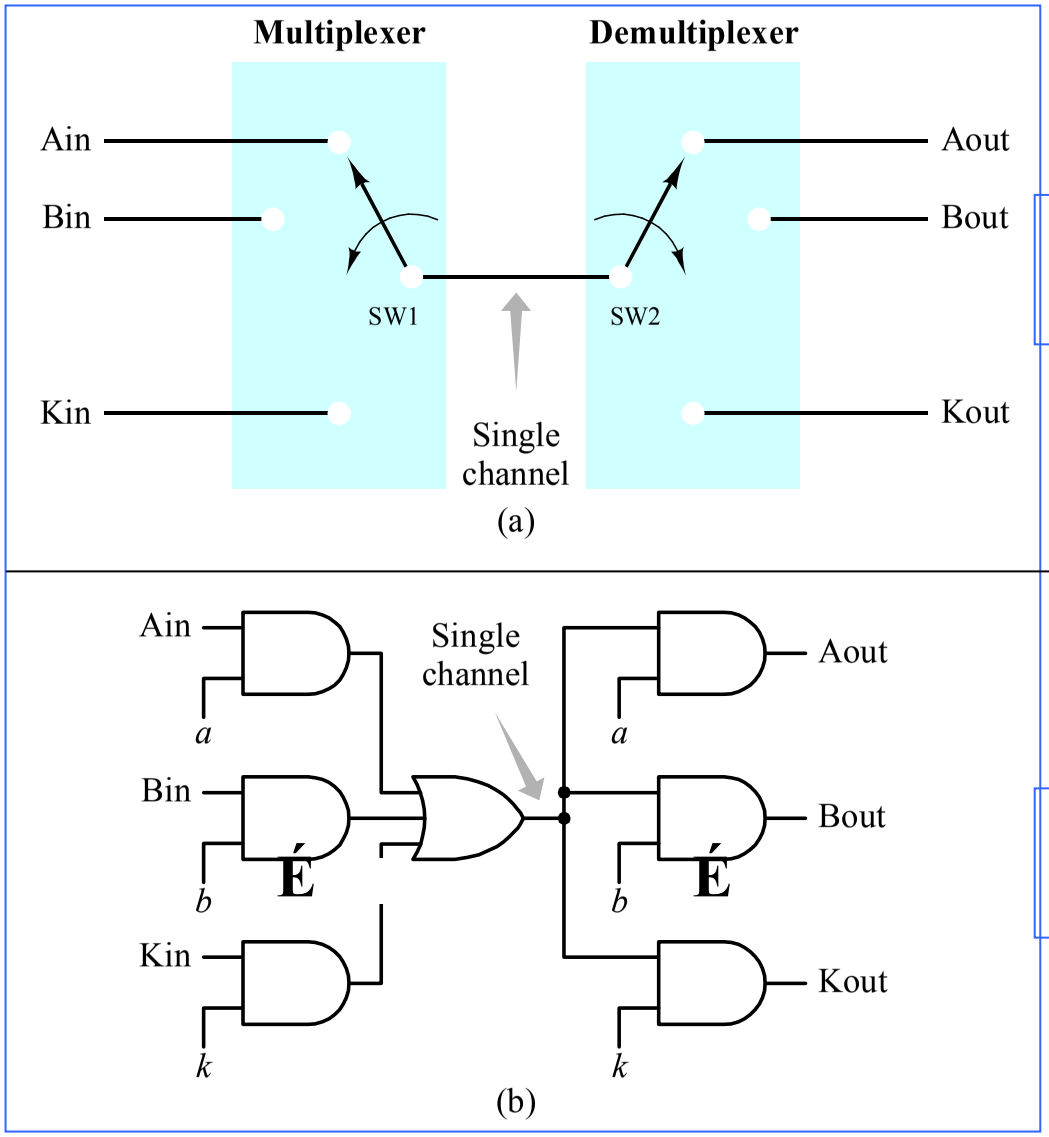
Combinational Logic Design

Instructor: Saraju P Mohanty

(Part 2)

- Multiplexers and Demultiplexers
- Binary Adders
- Binary Subtraction
- Binary Adder-Subtractor
- Binary Multipliers
- Decimal Arithmetic

K-Channel Multiplexing / Demultiplexing



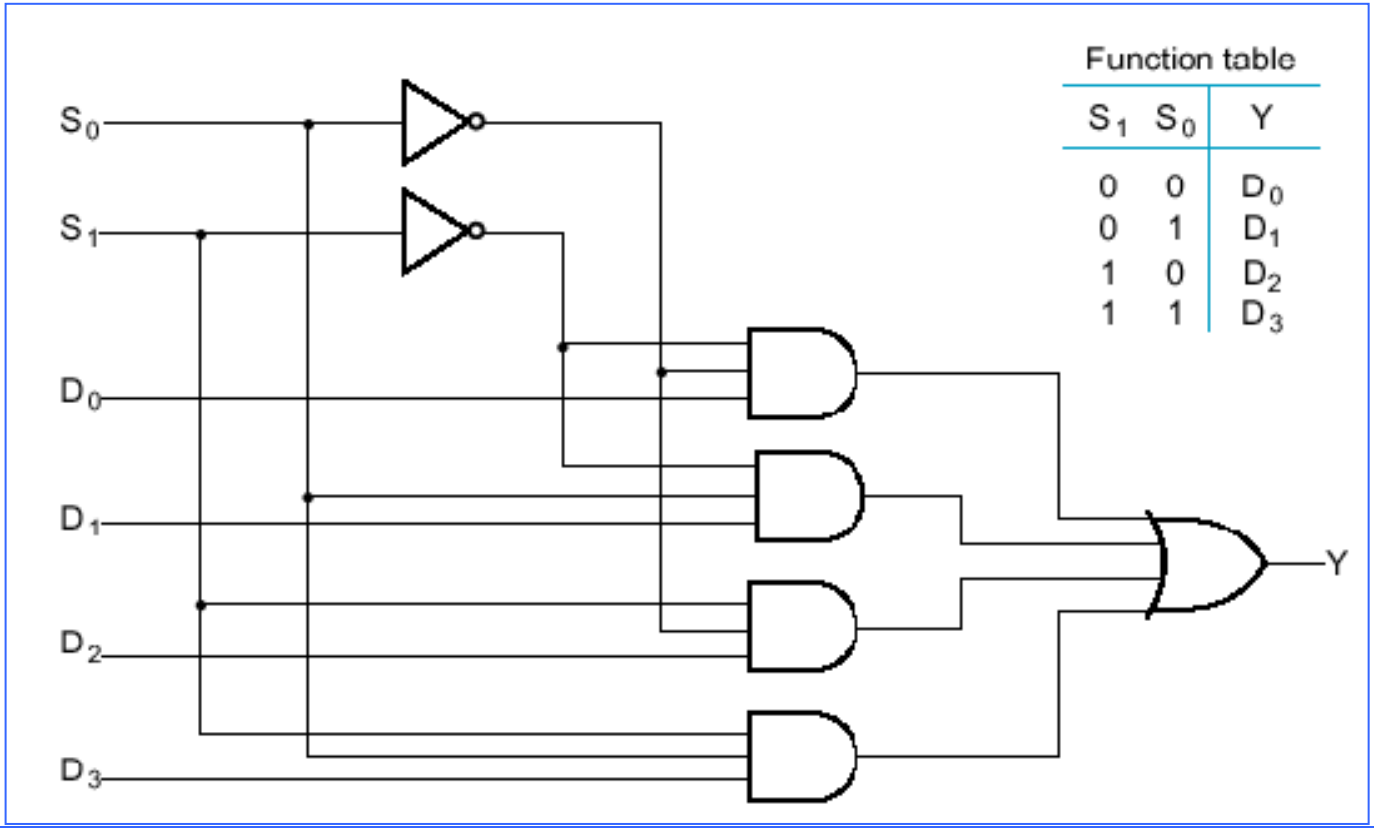
Multiplexing/Demultiplexing Operation

A Simple Logic Configuration

Multiplexers

- Selects binary information from one of many input lines and directs this information to a single output line.
- The selection of a particular input line is controlled by a set of input variables, called selection inputs. Normally, there are 2^n input lines and n selection inputs whose bit combinations determine which input is selected.
- In general a 2^n-to-1-line multiplexer is constructed from an n-to-2^n decoder by adding 2^n input lines to it, one from each data output.
- The size of the multiplexer is specified by the number 2^n of its data input lines and the single output line.
- As in decoders, multiplexers may have an enable input to control the operation of the unit. When the enable input is in the inactive state, the outputs are disabled.
- The enable input is useful for expanding two or more multiplexers into a multiplexer with a larger number of inputs.

Multiplexers : A 4-to-1-line multiplexer



Each input applied to one input of an AND gate. Selection inputs S_0 and S_1 are decoded to select a particular AND gate. AND gate outputs are applied to a single OR gate to provide the 1-line output.

A 4-to-1-line Multiplexer using Trans. Gates

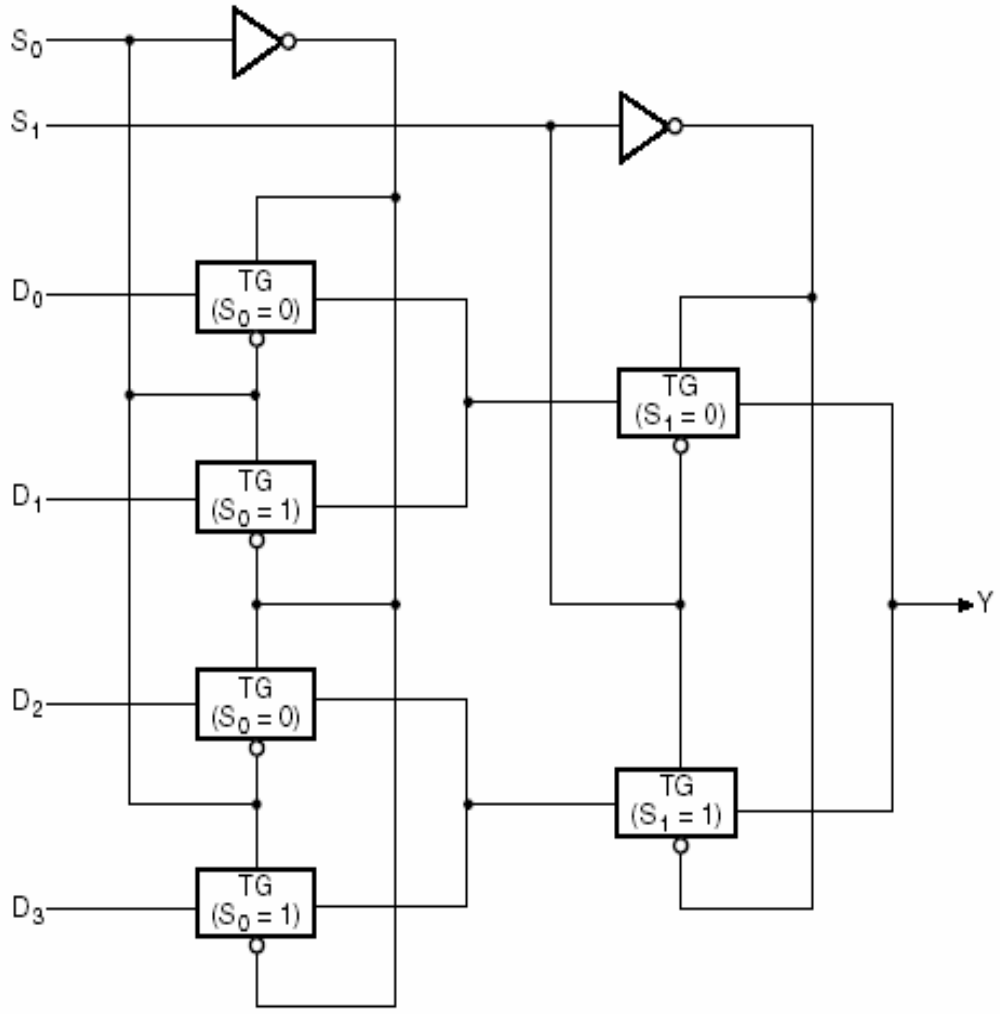


Fig. 3-20 4-to-1-Line Multiplexer with Transmission Gates

Two selection lines S_0 and S_1 control the transmission paths.

For example, S0=0 and S1 =0, then Y = D0.

Quadruple 2-to-1-line Multiplexer

Multiplexer blocks can be combined in parallel with common selection and enable lines to perform selection on multiple-bit quantities.

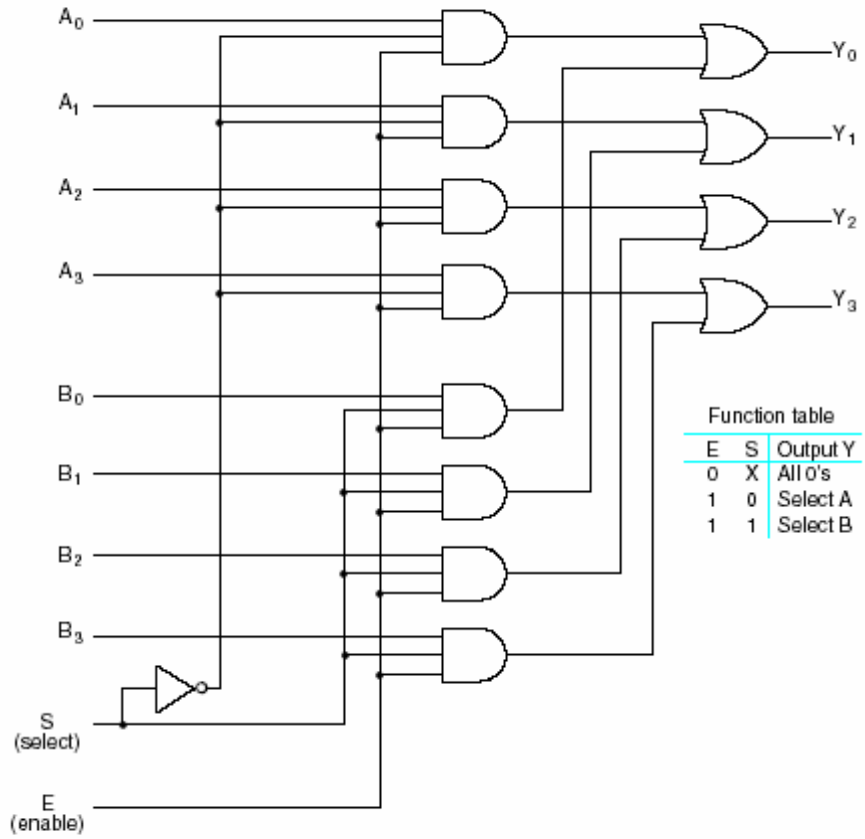


Fig. 3-21 Quadruple 2-to-1-Line Multiplexer

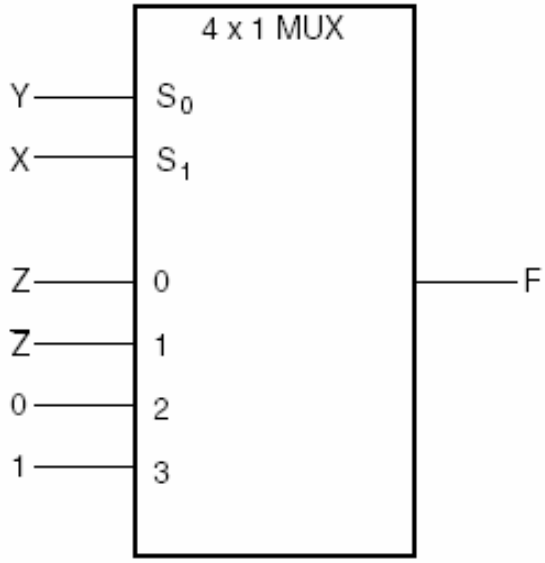
MUX: Combinatorial circuit implementation

- List the Boolean Function in a truth table.
- Apply first n-1 variables as selection inputs.
- For each combination of selection variables evaluate the output as a function of last variable, 0 and 1.

Example: Implementation of the function $F(X,Y,Z) = \Sigma m(1,2,6,7)$

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

Fig. 3-22 Implementing a Boolean Function with a Multiplexer

MUX: Combinatorial circuit implementation

Example:

Implementation of the function F(A,B,C,D) =Σm(1,3,4,11,12,13,14,15)

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	

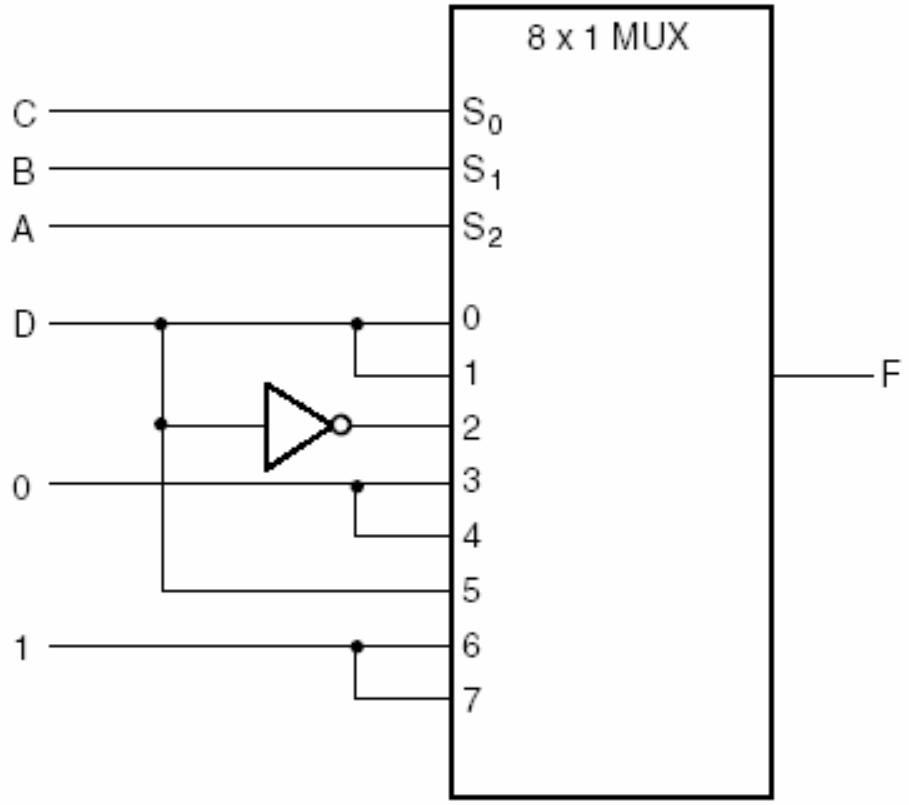
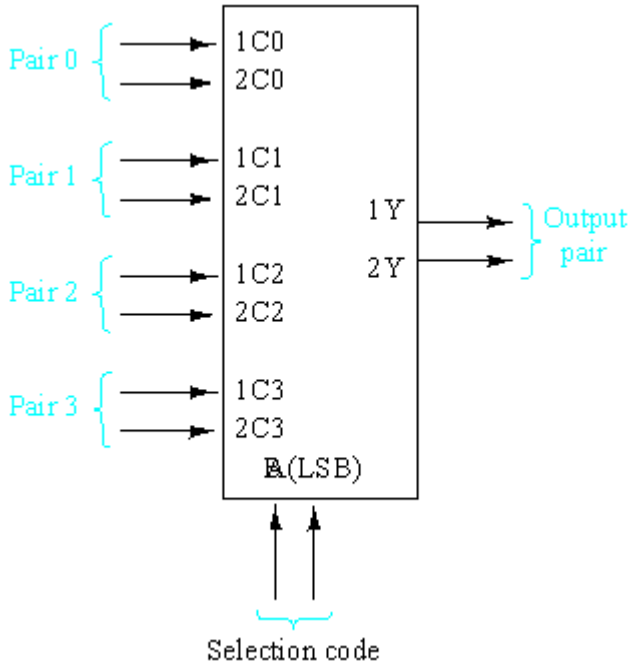


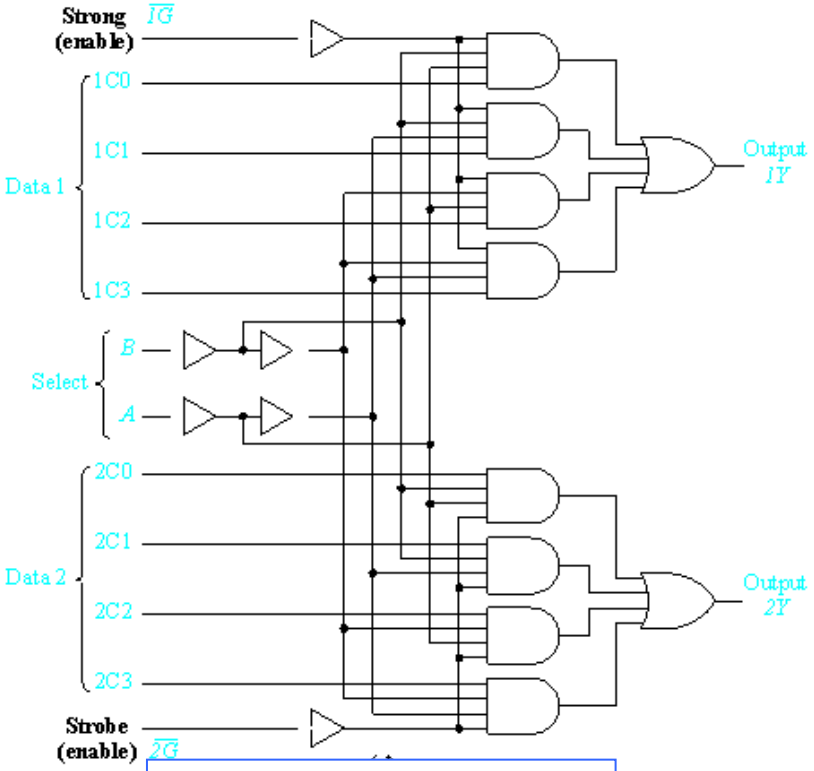
Fig. 3-23 Implementing a Four-Input Function with a Multiplexer

A real multiplexer chip : 74153 Dual 4-to-1

It has two 4-to-1 multiplexers each with four inputs, two select lines, and a single active high output (Y). In addition, there is an active low Enable input called the Strobe (S) for each.



Pin Diagram



Circuit Diagram

Demultiplexer

Performs the opposite of a multiplexer. Receives information from a single input line and transmit it to one of 2^n possible output lines. The selection of the specific output is controlled by the bit combination of n selection lines.

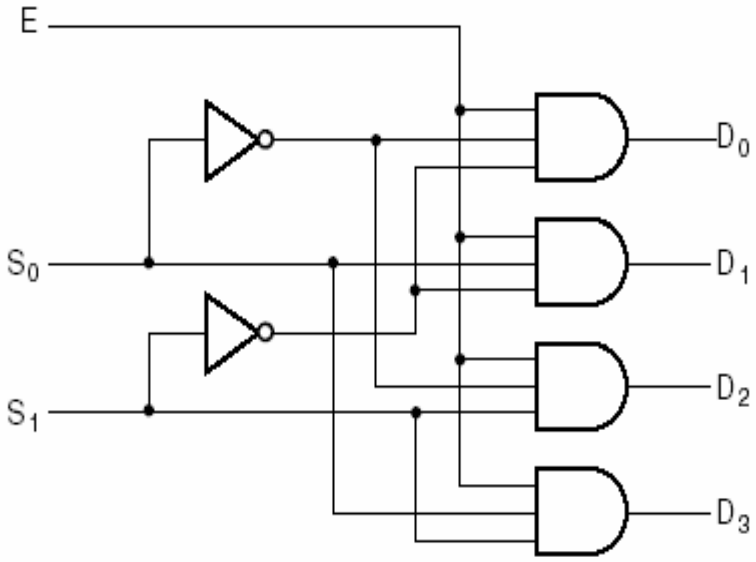


Fig. 3-24 1-to-4-Line Demultiplexer

Example:

When $(S_1,S_0) = 10$, the output D_2 is input E

Decoder Vs Demultiplexer

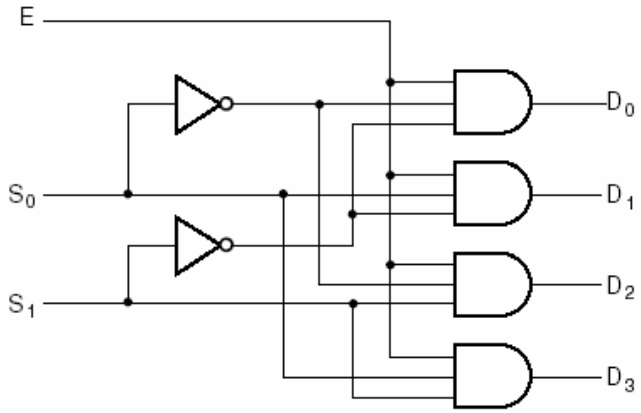


Fig. 3-24 1-to-4-Line Demultiplexer

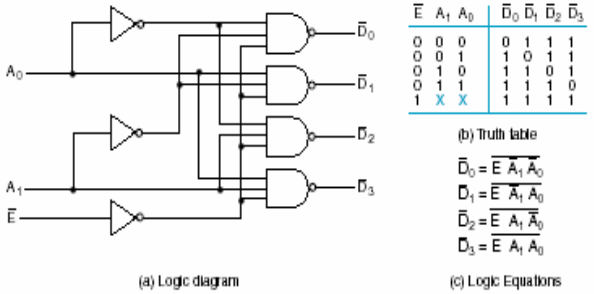


Fig.3-14 A 2–to–4-Line Decoder

- Logic circuits are exactly the same
- Applications are different
- Decoder with enable input is referred to as decoder / demultiplexer

Binary Adders

A combinatorial circuit that performs the addition of two bits is called a half adder. One that performs the addition of three bits (two significant bits and a previous carry is a full adder.

TABLE 3-7
Truth Table of Half Adder

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

Table 3-7 Truth Table of Half Adder

TABLE 3-8
Truth Table of 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

Table 3-8 Truth Table of Full Adder

Binary Adders: Half adder

Boolean Functions :

S=X'Y+XY'=X XORY

C=XY

TABLE 3-7
Truth Table of Half Adder

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

Table 3-7 Truth Table of Half Adder

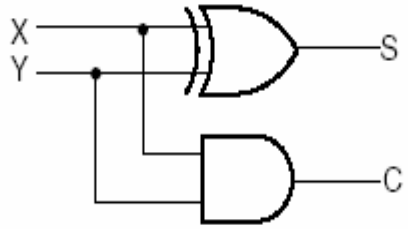


Fig. 3-25 Logic Diagram of Half Adder

Binary Adders : Full adder

Boolean Functions:

$$S = X'Y'Z+X'YZ'+XY'Z'+XYZ = (X \text{ XOR } Y) \text{ XOR } Z$$

$$C = XY+XZ+YZ = XY+Z(X \text{ XOR } Y)$$

TABLE 3-8
Truth Table of 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

Table 3-8 Truth Table of Full Adder

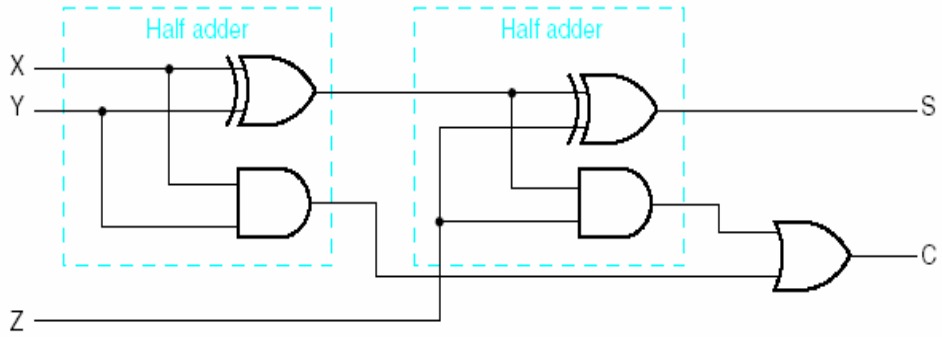


Fig. 3-27 Logic Diagram of Full Adder

Binary Ripple Carry adder

The full adders are connected in cascade, with the carry output from one full adder connected to the carry input of the next full adder.

Since a 1 carry may appear near the LSB of the adder and yet propagate through many full adders to the MSB → the name **ripple carry adder**. An n-bit ripple carry adder requires n full adders.

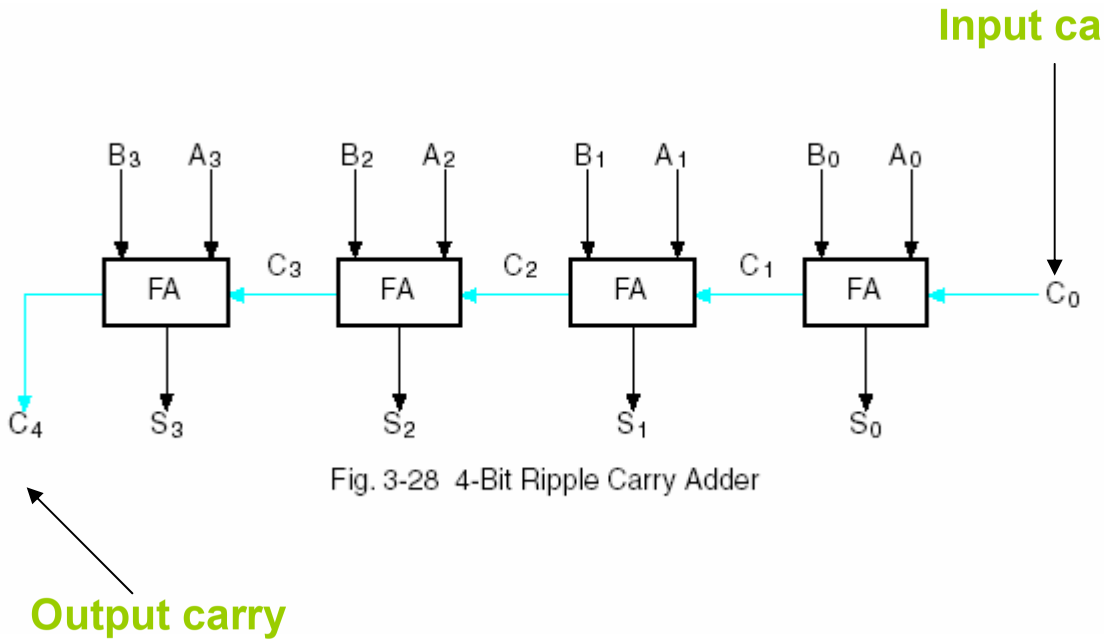


Fig. 3-28 4-Bit Ripple Carry Adder

Input carry:	0110
Augend A:	1011
Addend B:	0011
Sum S:	**1110**
Output carry:	**0011**

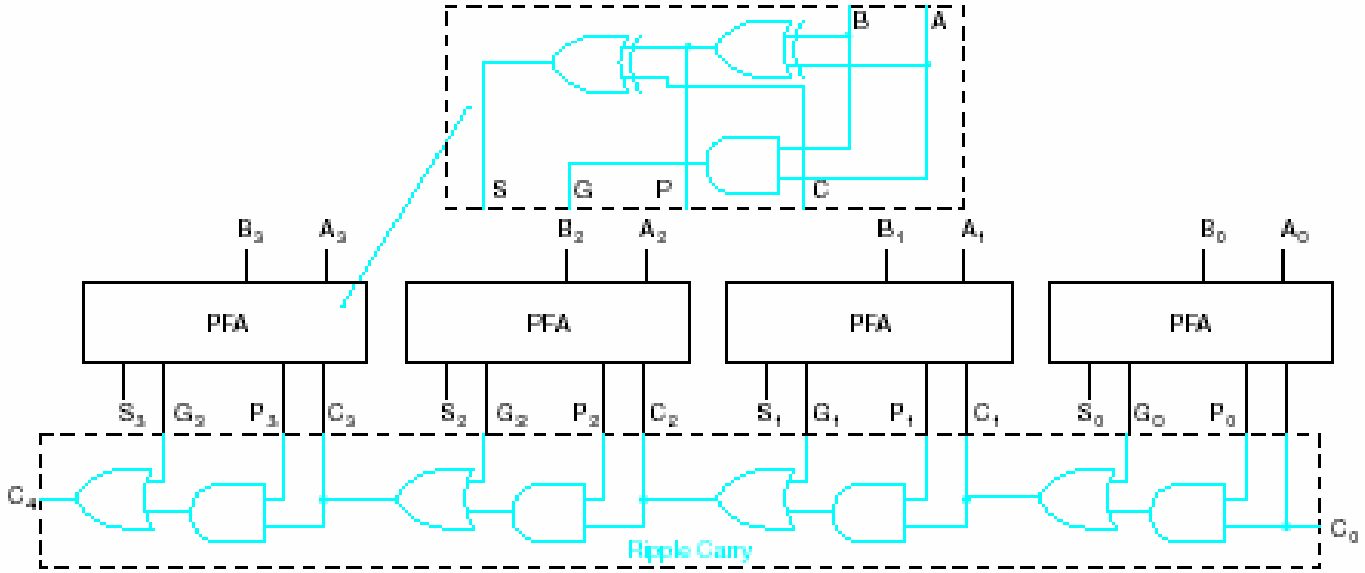
Disadvantage of Ripple Carry Adder

- The design of the 4-bit ripple carry adder with the usual method would require a truth table with 512 entries, since there are nine inputs to the circuit.
- Long circuit delay due to the many gates in the carry path from the LSB to the MSB.
- The longest path delay through an n-bit ripple carry adder is 2n+2 gate delays.
- Carry lookahead adder reduces critical path delay, but there is area penalty involved.

Carry Lookahead Adder

- Reduced delay at the price of more complex hardware.
- The design can be obtained by a transformation of the ripple carry design in which the carry logic over fixed groups of bits of the adder is reduced to two-level logic.
- Construct a new logic hierarchy separating the parts of the full adders not involving the carry propagation path from those containing the path.
- Call the first path of each full adder a **partial full adder** (PFA).

Carry Lookahead Adder: PFA model



- Two outputs: P_i, G_i from each PFA to the ripple carry path
- One input: C_i, the carry input, from the carry path to each PFA.
- Propagate function: $P_i = A_i XOR B_i$. When it is equal to 1 an incoming carry is propagated through the bit position from C_i to C_{i+1}; when equal to zero, carry propagation is blocked.
- Generate function: $G_i = A_i * B_i$. Whenever equal to 1 regardless of the P_i value, the carry output from the position is 1, so a carry has been generated in the position. When equal to zero, no carry is generated, so that C_{i+1} is 0 if the carry propagated through the position from C_i is also zero.

Carry Lookahead Adder: PFA model

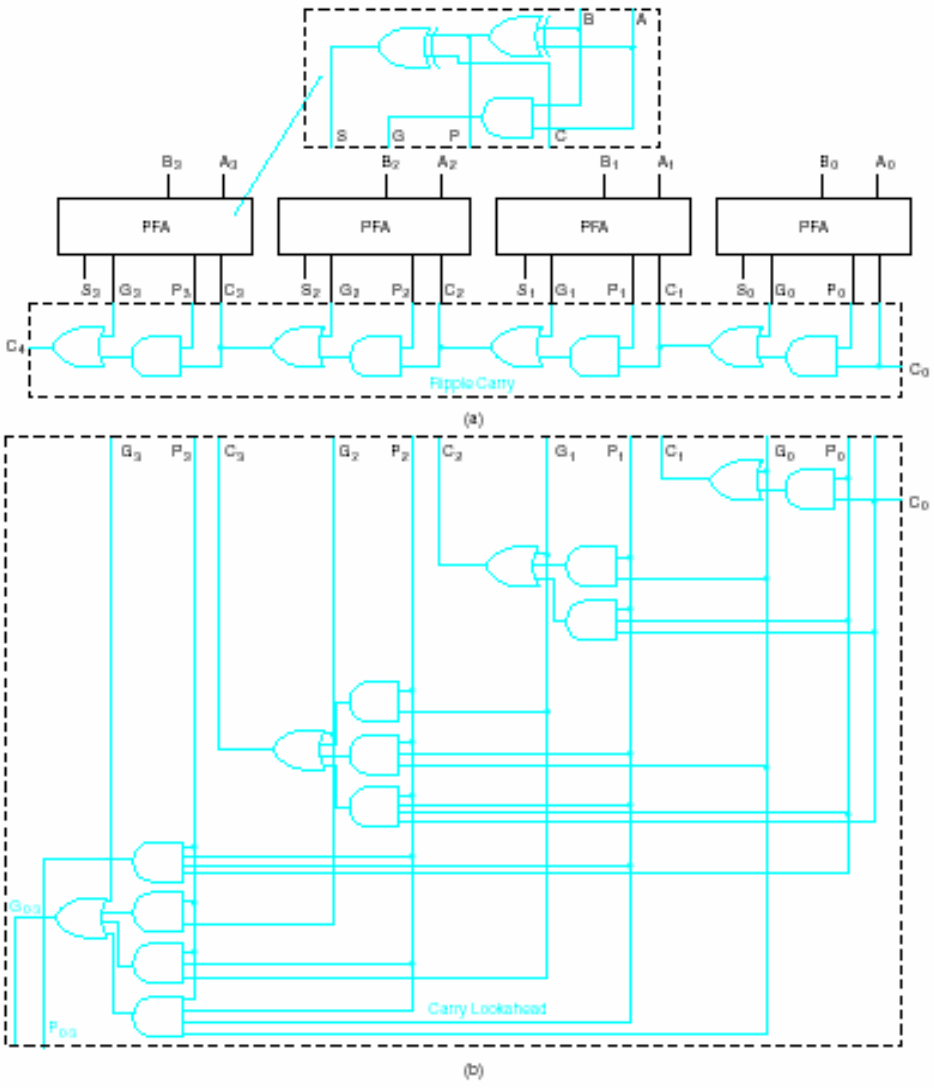


Fig. 3-29 Development of a Carry Lookahead Adder

Ripple Carry Path

Carry Lookahead circuit will replace the ripple carry path above.

4-bit ripple carry adder has delay of 10 gate delays, for carry lookahead it is 6 gate delays. Assume XOR has 2 OR gate delays.

Binary Subtraction

The followings are left for the students as self reading and may appear in the test:

- Complements
- Subtraction by 2's complements subtract
- Binary subtraction by 1's complement addition
- Binary subtraction by 2's complement addition

Adder-Subtractor Circuit

When S = 0 the circuit is an adder, and when S = 1 the circuit is an subtractor.

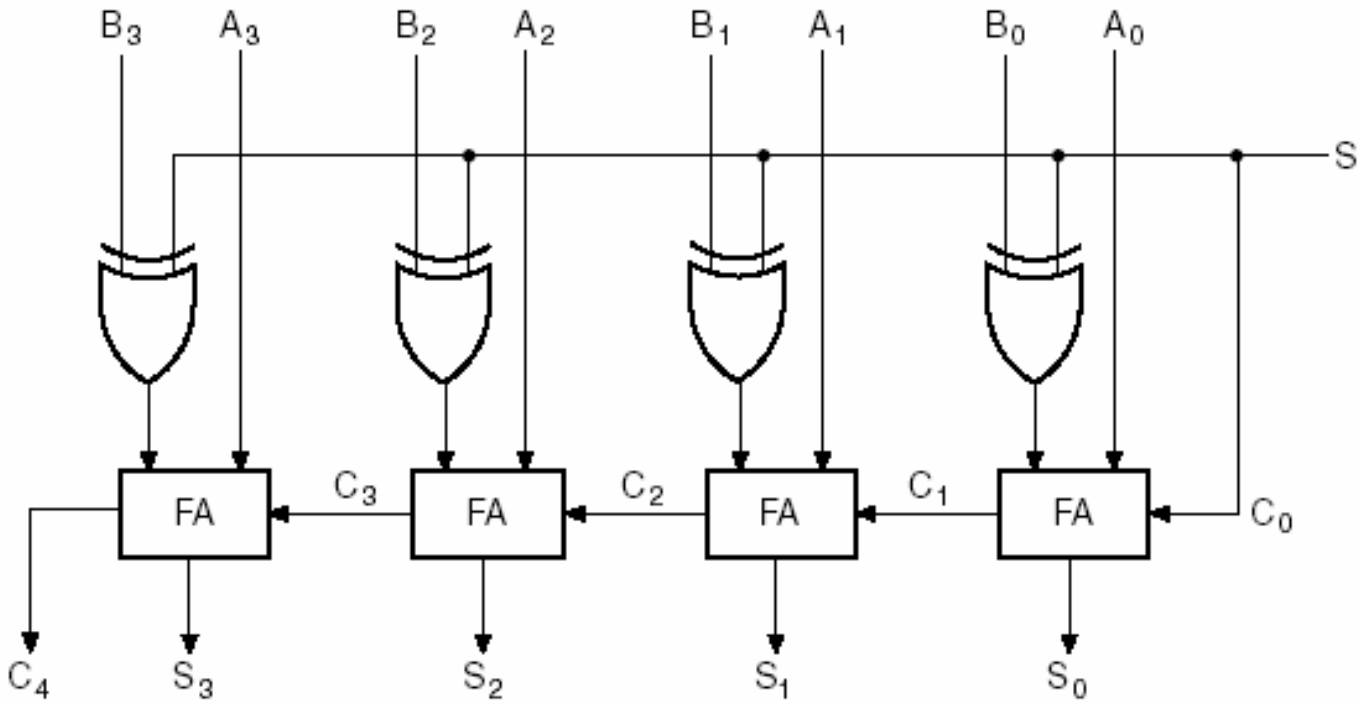


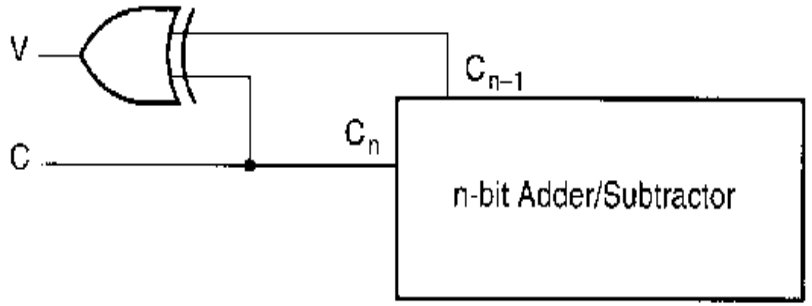
Fig. 3-31 Adder-Subtractor Circuit

Overflow Detection

When adding / subtracting we must ensure that the sum has sufficient number of bits to accommodate the sum. Given two *n-bit* numbers, if the sum occupies *n+1* bits, we say that an *overflow* occurs. Overflow must be detected!

Simple circuit is shown, where if the numbers are unsigned, when *C=1* detects a carry (overflow) for an addition and indicates no correction step for subtraction. When *C= 0* the other way around.

If the numbers are signed, *V* is used to detect an overflow. *V=0* after a signed addition/subtraction indicates no overflow. If *V=1* then result contains *n+1* bits, but only the n rightmost fit in the n-bit result, so an overflow has occurred. The (n+1)th bit is the actual sign but it cannot occupy the sign bit position in the result.



4-bit binary adder-subtractor

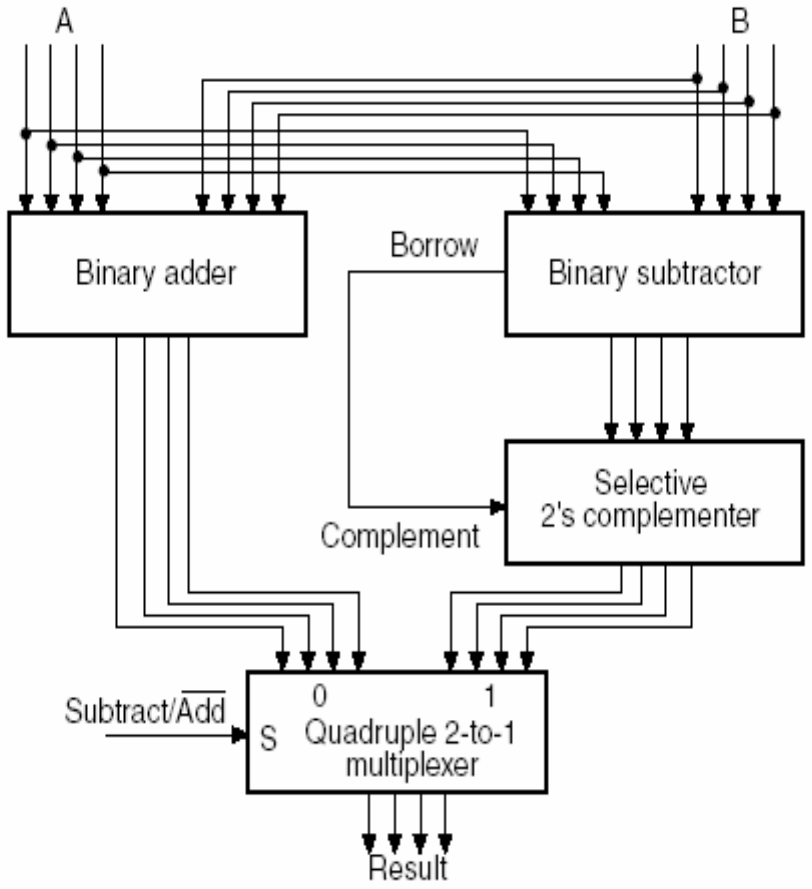
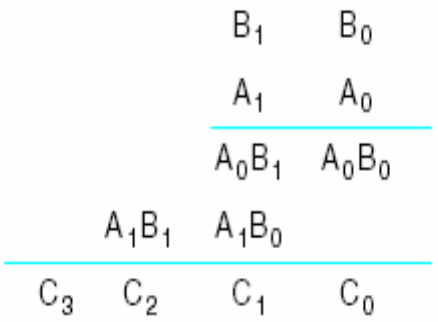


Fig. 3-30 Block Diagram of Binary Adder-Subtractor

Both addition and subtraction can be performed in parallel.

A subtractor can be realized from a ripple carry adder if inverters are placed between each B terminal and corresponding full adder.

Binary Multipliers: A 2-bit example



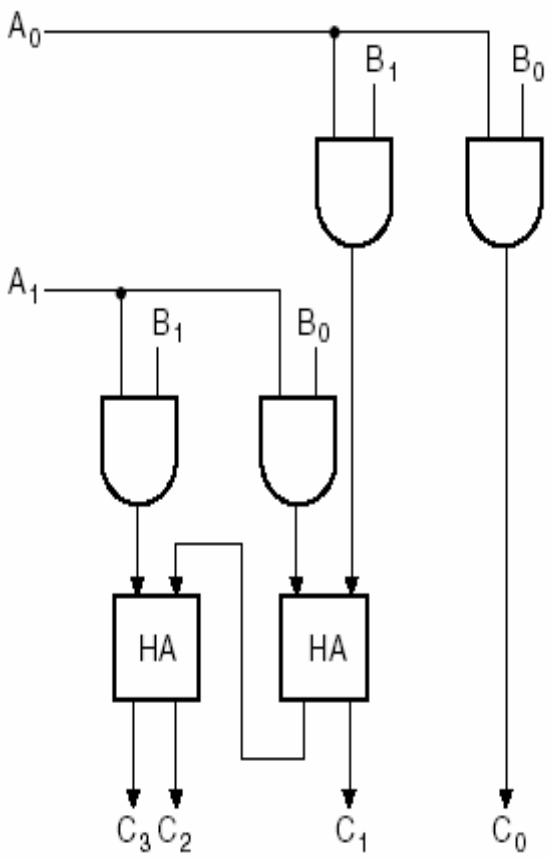


Fig. 3-33 A 2-Bit by 2-Bit Binary Multiplier

Product A_0 and B_0 is 1 if both are 1, else it is 0. Thus, the product is same as AND operation.

Binary Multipliers: A 4-bit by 3-bit example

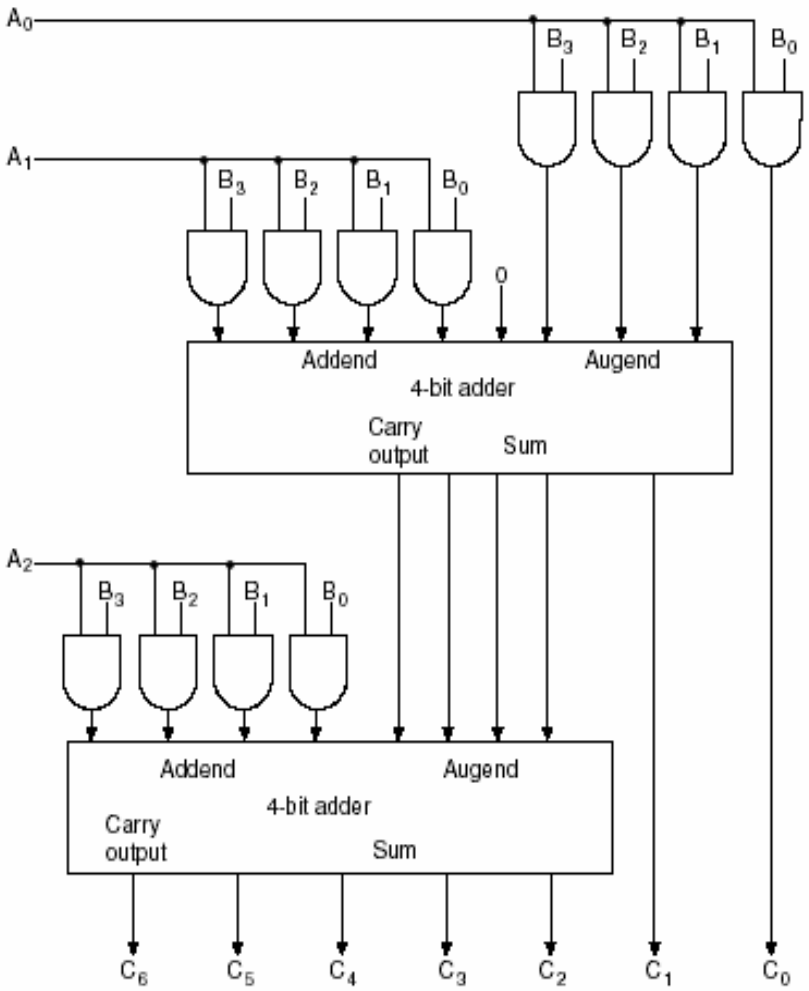


Fig. 3-34 A 4-Bit by 3-Bit Binary Multiplier

For J multiplier bits and K multiplicand bits, we need JxK AND gates and (J-1) K-bit adders to produce a product of J+K bits.

Decimal Arithmetic : BCD Adder

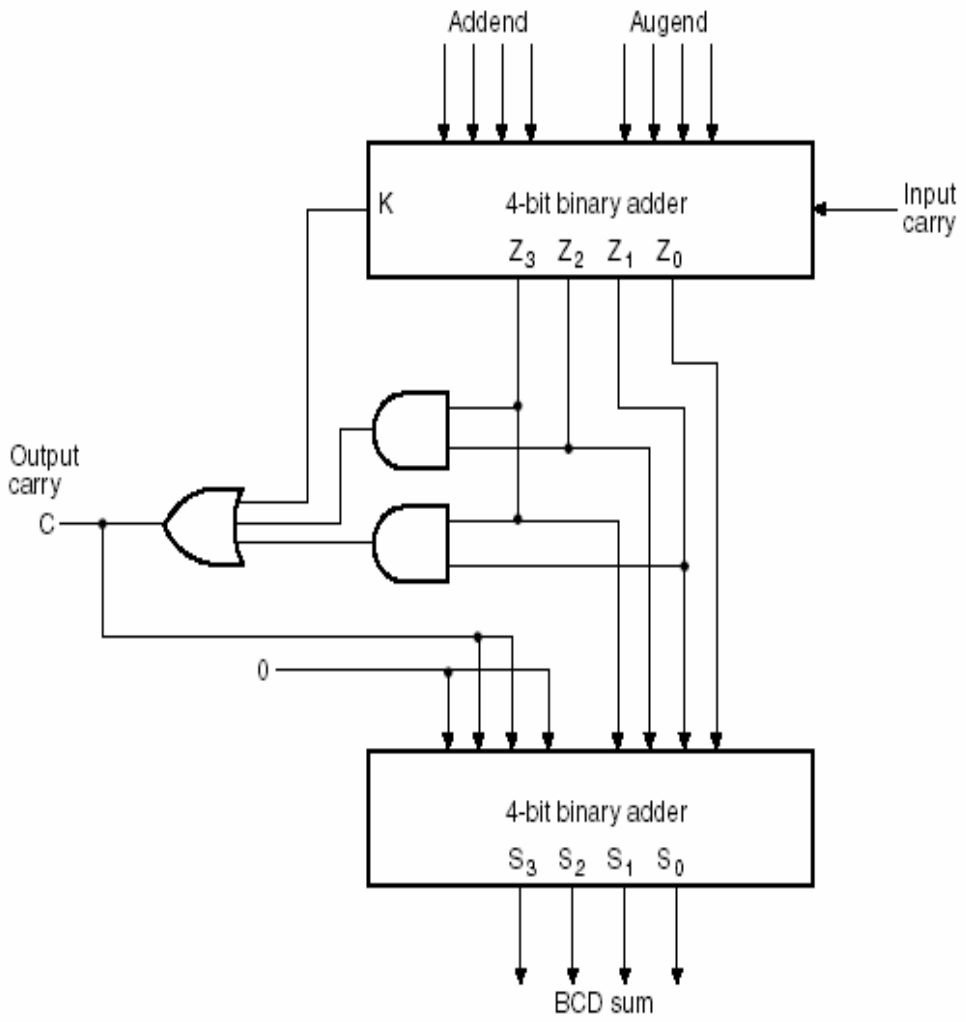


Fig. 3-35 Block Diagram of BCD Adder

Computers or Calculator that perform arithmetic operations directly in the decimal numbers represent numbers in BCD form.

Two BDC numbers are added as if they were two 4-bit numbers. When binary sum is <= 1001 (decimal 9), the BCD sum is correct, else add 0110 (decimal 6) to get appropriate BCD sum.