Digital Electronics
ES-207A

Rajeev Kumar

AP ECE Department (PIET)

Research Domain: Soft Computing, Li-Fi

AI, ML & DL

Course Outcome

CO1	**To introduce basic postulates of Boolean algebra and shows the correlation between Boolean expressions**
CO2	To introduce the methods for simplifying Boolean expressions
CO3	To outline the formal procedures for the analysis and design of combinational circuits and sequential circuits
CO4	To introduce the concept of memories and programmable logic devices.

Syllabus

UNIT 1 MINIMIZATION TECHNIQUES AND LOGIC GATES

- Binary Digits, Logic Levels, and Digital Waveforms, Logic Systems-Positive and negative, Logic Operations, Logical Operators, Logic Gates-AND, OR, NOT, NAND, NOR, Exclusive-OR and Exclusive-NOR, Active high and Active low concepts, Universal Gates and realization of other gates using universal gates, Gate Performance Characteristics and Parameters. Boolean Algebra: Rules and laws of Boolean algebra, Demorgan's Theorems, Boolean Expressions and Truth Tables, Standard SOP and POS forms; Minterm and Maxterms, Canaonical representation of Boolean expressions, Duality Theorem, Simplification of Boolean Expressions, Minimization Techniques for Boolean Expressions using Karnaugh Map and Quine McCluskey Tabular method. Introduction of TTL and CMOS Logic and their characteristics, Tristate gates.

UNIT 2 COMBIMATIONAL CIRCUITS

- Introduction to combinational Circuits, Adders-Half-Adder and Full-Adder, Subtractors- Half and Full Subtractor; Parallel adder and Subtractor; Look-Ahead Carry Adders. BCD adder, BCD subtractor, Parity Checker/Generator, Multiplexer, Demultiplexer, Encoder, Priority Encoder; Decoder ,BCD to Seven segment Display Decoder/Driver, LCD Display, and Comparators.

Cont…

UNIT 3 SEQUENTIAL CIRCUITS

- **Introduction to Sequential Circuits**, Flip-Flops: Types of Flip Flops -RS, T, D, JK; Edge triggering, Level Triggering; Flip Flop conversions; Master-Salve JK.
- Introduction to shift registers, Basic Shift Register Operations, types of shift registers, Bidirectional Shift Registers, Shift Register Counters. Introduction to counters, Types of Counters-Asynchronous and synchronous counters, Up/Down Synchronous Counters, Modulo-n Counter , State table, excitation table concepts, Design of asynchronous and synchronous counters, Ring Counter, Applications of counters.

UNIT 4 CONVERTER and MEMORY DEVICES

- **Digital to Analog Converter**, Weighed Register: R-2R Ladder Network: **Analog to Digital Conversion**, Successive Approximation Type, Dual Slope Type.
- **Classification of memories** - ROM: ROM organization, PROM, EPROM, EEPROM, EAPROM, RAM: - RAM organization - Write operation, Read operation, Memory cycle, Timing wave forms, memory expansion, Static RAM Cell, MOSFET RAM cell structure, Dynamic RAM cell structure, Programmable Logic Devices - Programmable Logic Array (PLA), Programmable Array Logic (PAL), Implementation of PLA, PAL using ROM.

Books

Suggested Books

- R. P. Jain , “Modem Digital Electronics (Edition III)”; TMH
- Anand Kumar , “Fundamentals of digital circuits” ; PHI
- Malvino & Leach, “Digital Principles and Applications”, McGraw Hill.
- Thomas L. Floyd, “Digital Fundamentals”, Pearson Education Inc,
-
- **Note: The Examiner will be given the question paper template and will have to set the question paper according to the template provided along with the syllabus.**

DE Lab

ES- 209AL

LIST OF EXPERIMENTS

- Familiarization with Digital Trainer Kit and associated equipment.
- Study of TTL gates AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.
- Design and realize a given function using K-Maps and verify its performance.
- To verify the operation of Multiplexer and De-multiplexer.
- Universal Gate
- HA and FA
- To verify the operation of Comparator.
- To verify the truth table of S-R, J-K, T, D Flip-flops.
- To verify the operation of Bi-directional shift register.
- To design and verify the operation of 3-bit asynchronous counter.
- To design and verify the operation of asynchronous Up/down counter using J-K FFs.
- To design and verify the operation of asynchronous Decade counter.
- Study of TTL logic family characteristics.
- Study of Encoder and Decoder.
- Study of BCD to 7 segment Decoder.

Digital Circuits

Combinational Circuits

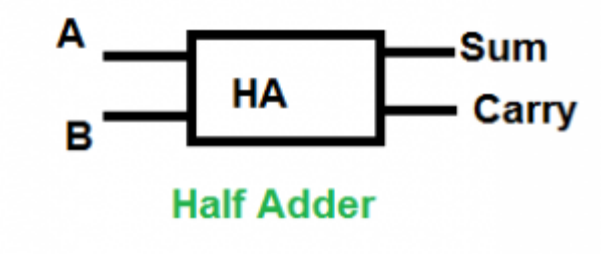
1. In this output depends only upon present input.
2. Speed is fast.
3. It is designed easy.
4. There is no feedback between input and output.
5. This is time independent.
6. Elementary building blocks: Logic gates
7. Used for arithmetic as well as Boolean operations.
8. Combinational circuits don't have capability to store any state.
9. As combinational circuits don't have clock, they don't require triggering.
10. These circuits do not have any memory element.
11. It is easy to use and handle.
12. **Example –** Encoder, Decoder, Multiplexer, De-multiplexer

Sequential Circuits

1. In this output depends upon present as well as past input.
2. Speed is slow.
3. It is designed tough as compared to combinational circuits.
4. There exists a feedback path between input and output.
5. This is time dependent.
6. Elementary building blocks: Flip-flops
7. Mainly used for storing data.
8. Sequential circuits have capability to store any state or to retain earlier state.
9. As sequential circuits are clock dependent they need triggering.
10. These circuits have memory element.
11. It is not easy to use and handle.
12. **Examples –** Flip-flops, Counters

Half Adder

- The addition of 2 bits is done using a combination circuit called a Half adder. The input variables are augend and addend bits and output variables are sum & carry bits. A and B are the two input bits



Half Adder

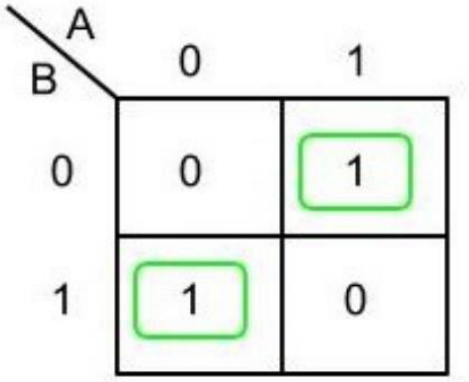
- **Truth Table:**

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Half Adder

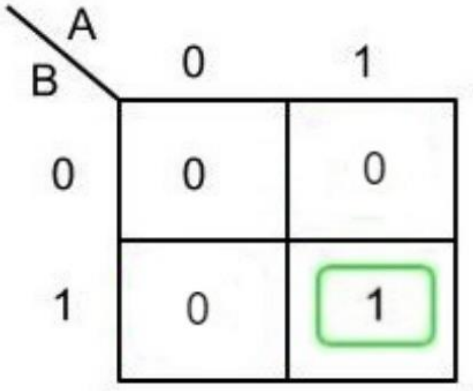
- **Logical Expression:**

For Sum



Sum = A XOR B

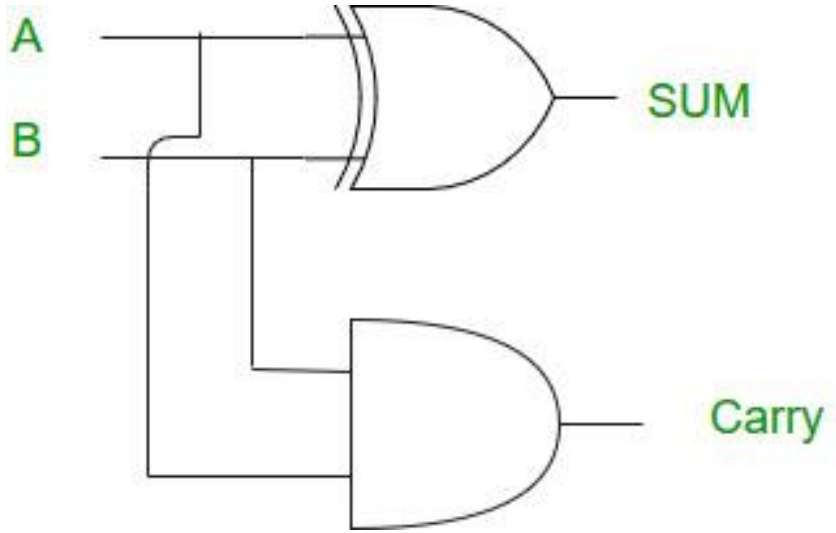
For Carry



Carry = A AND B

Half Adder

- **Implementation**

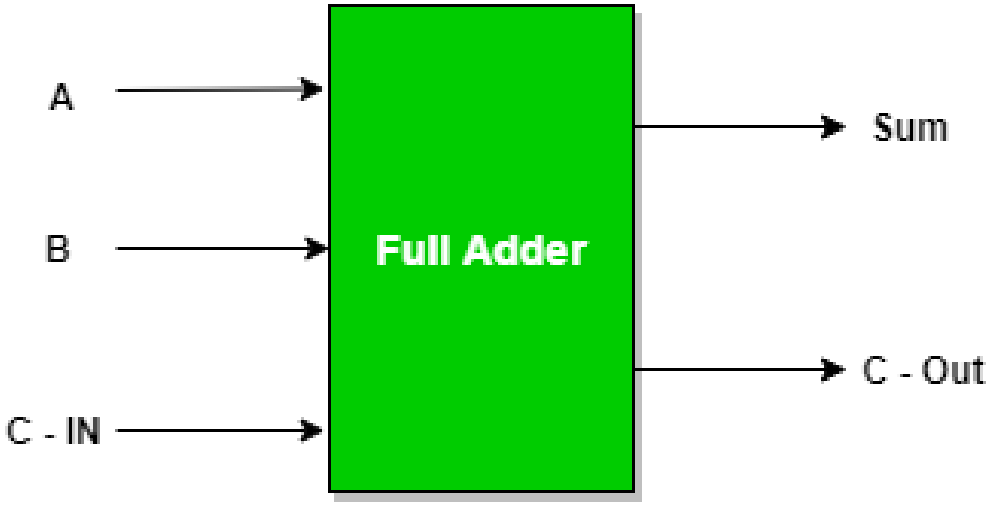


- **Note:** Half adder has only two inputs and there is no provision to add a carry coming from the lower order bits when multi addition is performed.

Full Adder

- Full Adder is the adder which adds three inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry as C-IN. The output carry is designated as C-OUT and the normal output is designated as S which is SUM.

- A full adder logic is designed in such a manner that can take eight inputs together to create a byte-wide adder and cascade the carry bit from one adder to the another.



Cont…

- **Full Adder Truth Table:**

Inputs			Outputs	
A	B	C – IN	Sum	C - Out
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

Cont…

- **Logical Expression for SUM:**

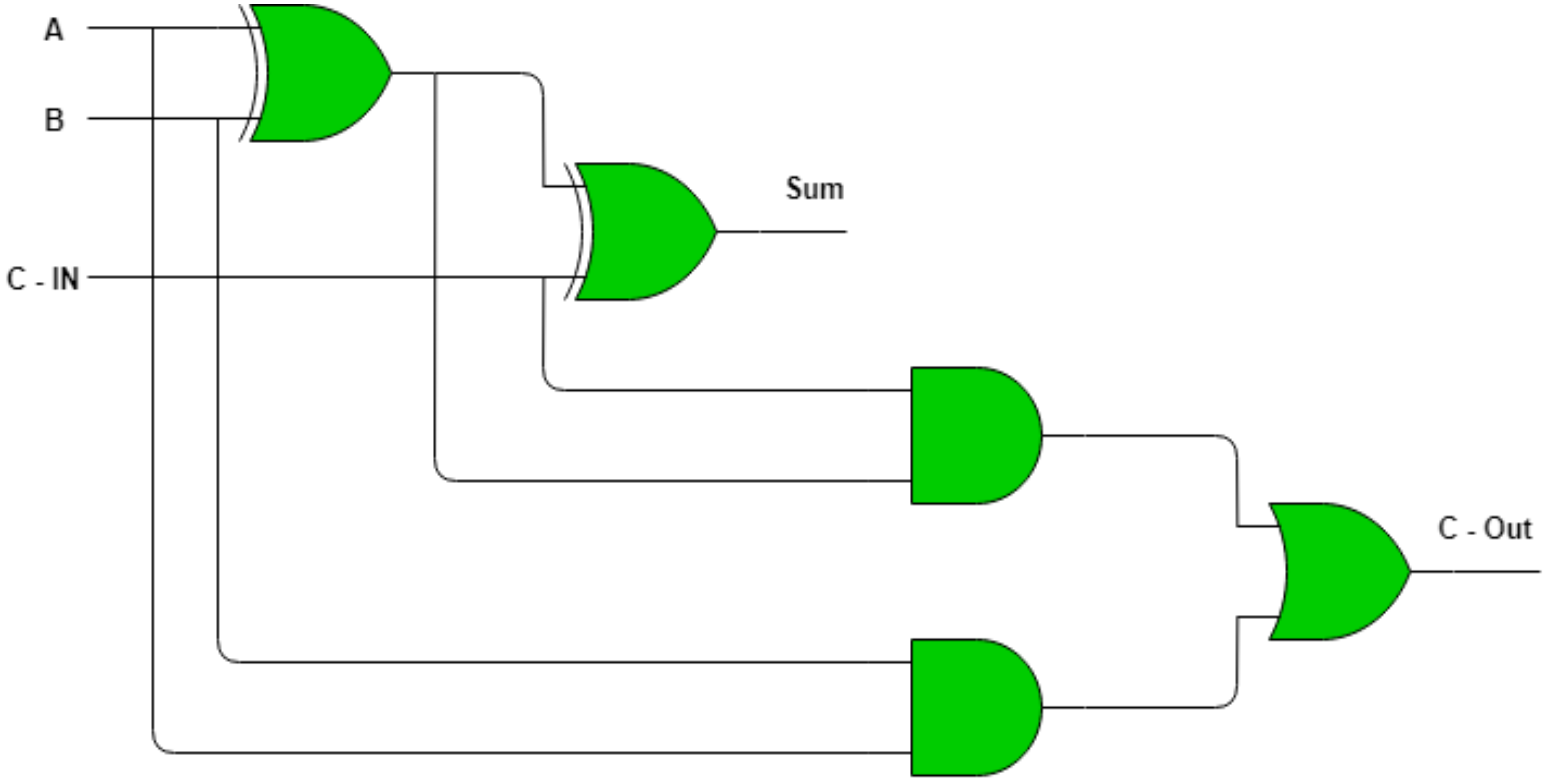
$$S = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in}$$
$$= (A\bar{B} + \bar{A}B)\bar{C}_{in} + (AB + \bar{A}\bar{B})C_{in}$$
$$= (A \oplus B)\bar{C}_{in} + (\overline{A \oplus B})C_{in} = A \oplus B \oplus C_{in}$$

- **Logical Expression for C-OUT:**

$$C_{out} = \bar{A}BC_{in} + A\bar{B}C_{in} + AB\bar{C}_{in} + ABC_{in}$$
$$= AB + (A \oplus B)C_{in}$$

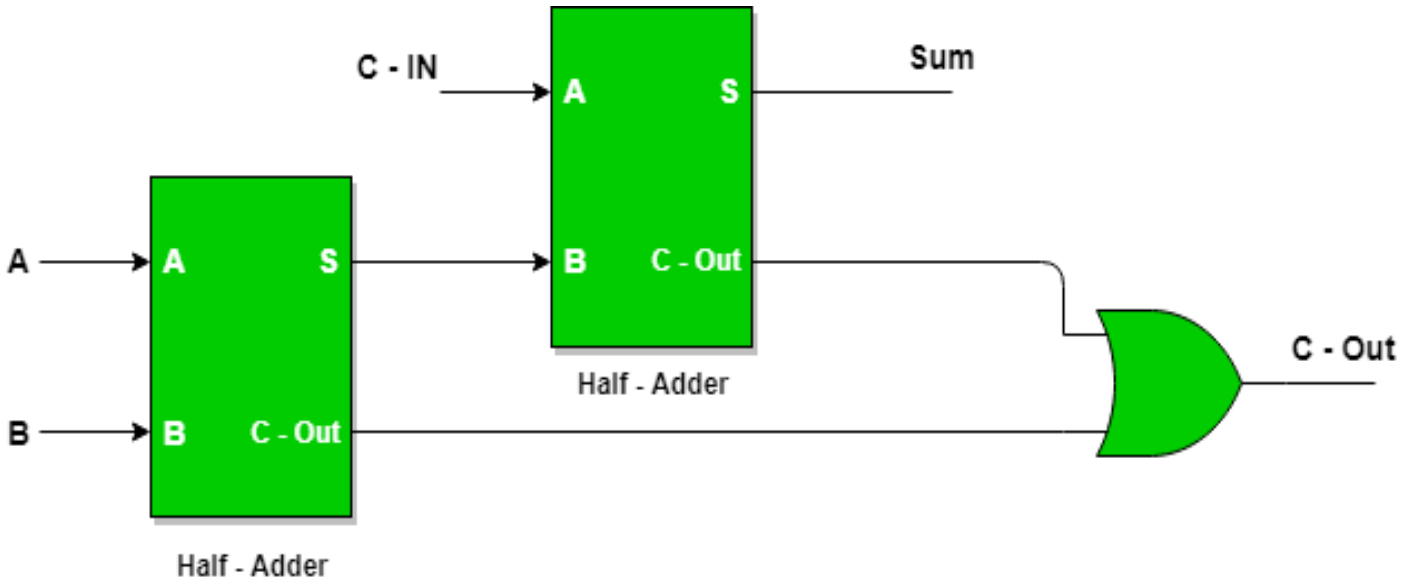
Cont…

- **Full Adder Logic Circuit**



Implementation of Full Adder using Half Adders

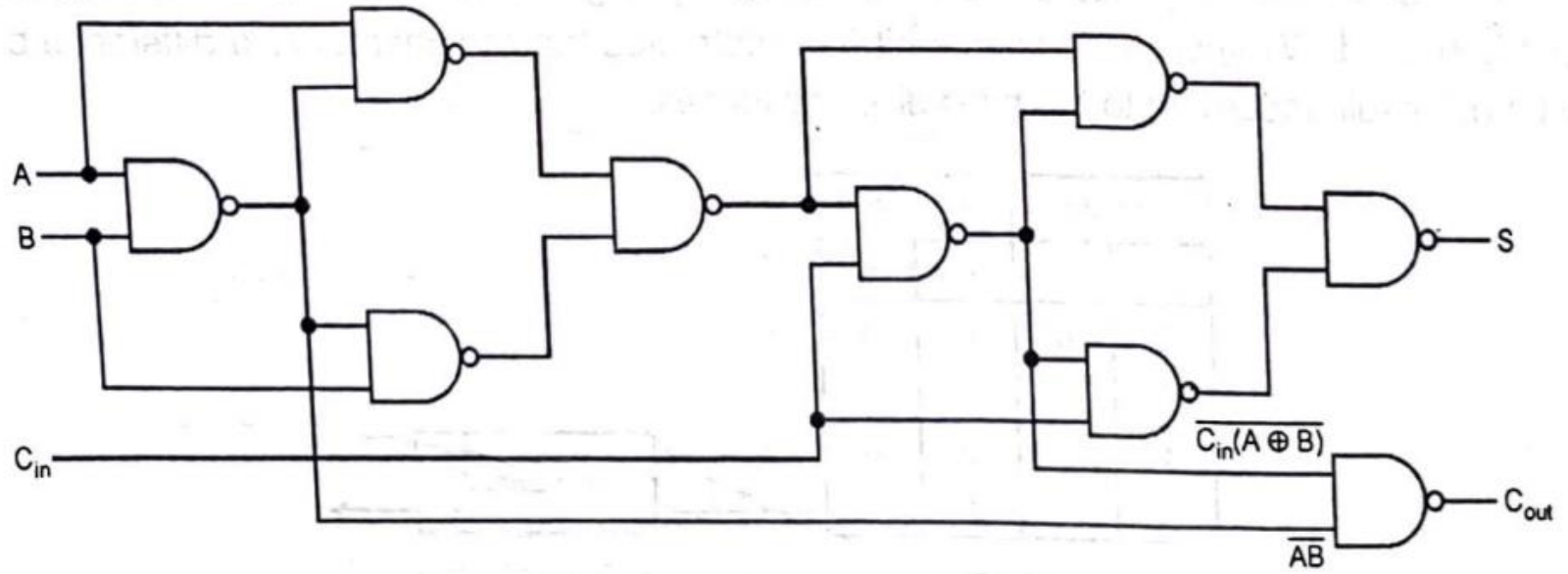
- 2 Half Adders and a OR gate is required to implement a Full Adder.



- With this logic circuit, two bits can be added together, taking a carry from the next lower order of magnitude, and sending a carry to the next higher order of magnitude.

Implementation of Full Adder using NAND gates

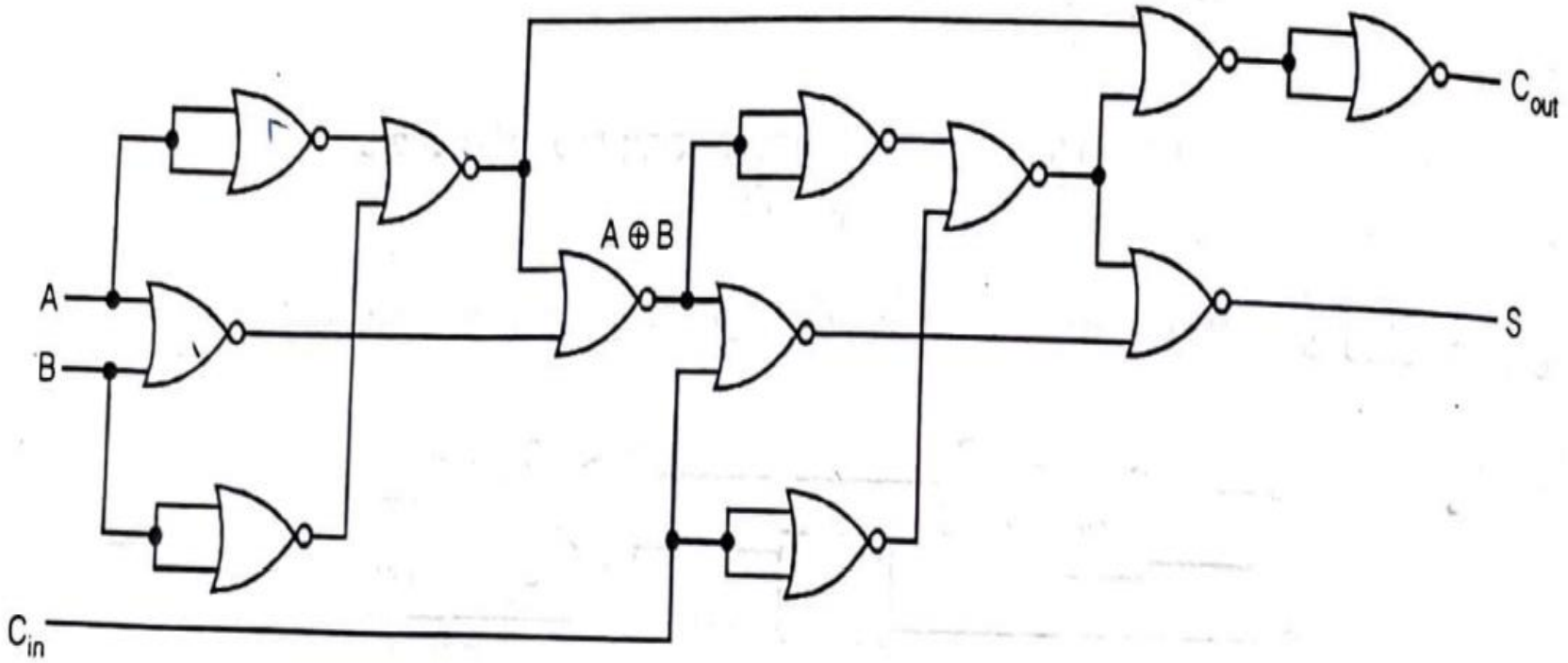
Total 9 NAND gates are required to implement a Full Adder



(Logic diagram of a full-adder using only 2-input NAND gates)

Implementation of Full Adder using NOR gates

Total 9 NOR gates are required to implement a Full Adder



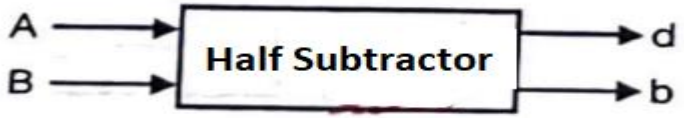
(Logic diagram of a full-adder using only 2-input NOR gates)

Half Subtractor

- The subtraction of two binary numbers may be accomplished by taking the compliment of the subtrahend and adding it to the minuend. By this method, the subtraction operation becomes an addition operation and instead of having a separate circuit for subtraction, the adder itself can be used to perform subtraction. This results in reduction of hardware.
- A half subtractor shown in figure is a combinational circuit with two inputs A and B and two outputs **d** and **b**.
- **d** indicates the difference and **b** is the output signal generated that informs the next stage that a 1 has been borrowed

Inputs		Outputs	
A	B	d	b
0	0	0	0
1	0	1	0
1	1	0	0
0	1	1	1

(a) Truth table



(b) Block diagram

(Half-subtractor)

Fig. (4.11)

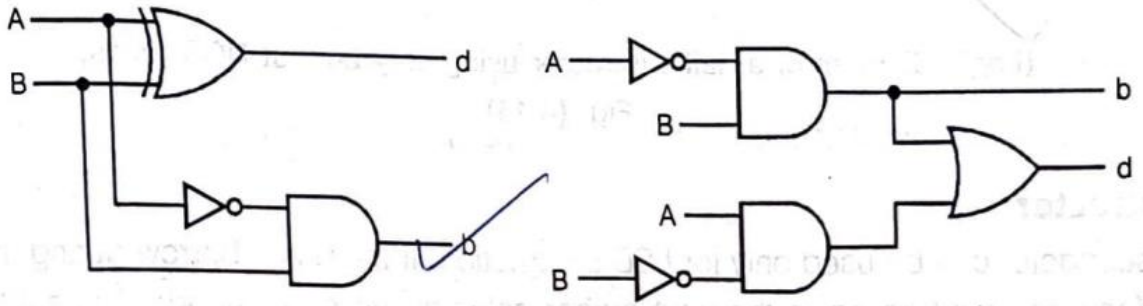
Cont…

The output borrow b is a 0 as long as A ≥ B. It is a 1 for A = 0 and B = 1. The d output is the result of the arithmetic opeation A – B.

A circuit that produces the correct difference and borrow bits in response to every possible combination of the two 1-bit numbers is, therefore, described by

$$d = A\bar{B} + \bar{A}B = A \oplus B \quad \text{and} \quad b = \bar{A}B$$

Figure (4.12) shows two logic diagrams of a half-subtractor – one using an Ex-OR gate together with one each NOT gate and AND gate and the other using the AOI gates. Note that the logic for d is exactly the same as the logic for output S in the half-adder.



(Logic diagrams of a half-subtractor)

Fig. (4.12)

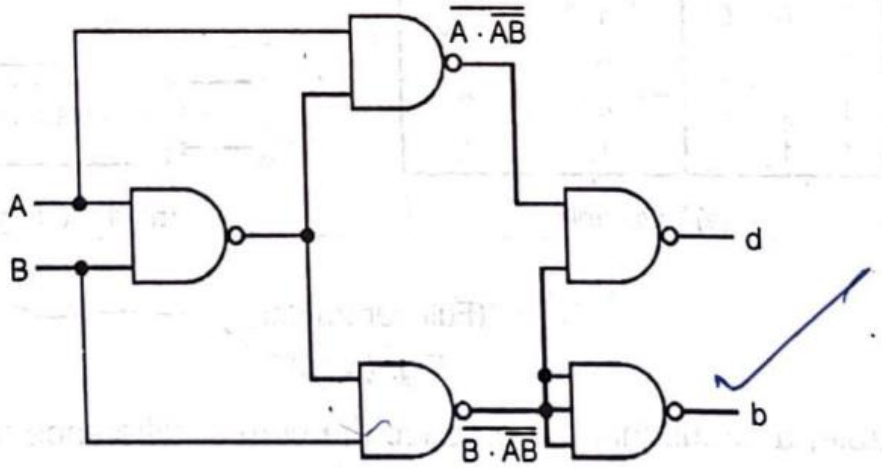
Cont…

- A half –subtractor can also be realized using universal logic gates as shown in figure

NAND Logic

$$d = A \oplus B = \overline{A \cdot \overline{AB}} \cdot \overline{B \cdot \overline{AB}}$$

$$b = \overline{A}B = B(\overline{A} + \overline{B}) = B(\overline{AB}) = \overline{B \cdot \overline{AB}}$$



(Logic diagram of a half-subtractor using only 2-input NAND gates)

Fig. (4.13)

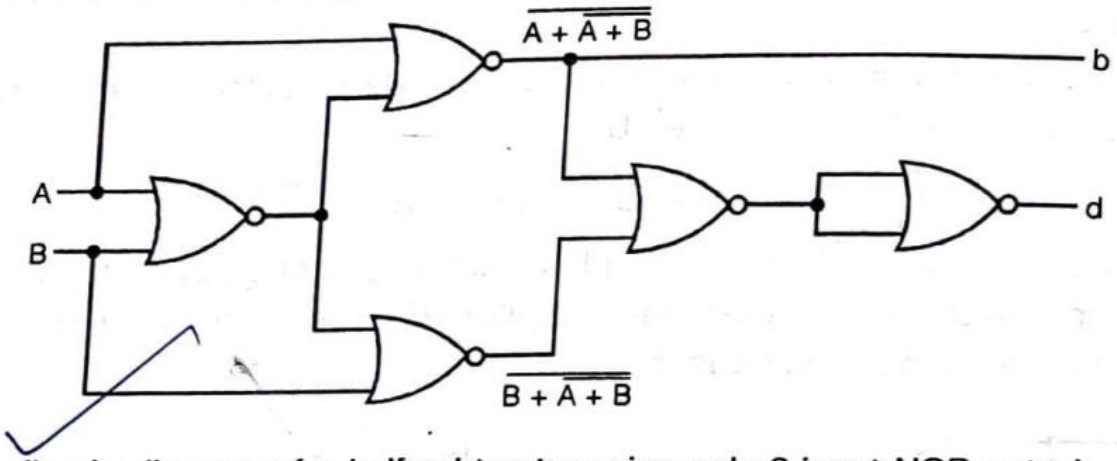
Cont…

- **NOR Logic**

$$d = A \oplus B = A\bar{B} + \bar{A}B = A\bar{B} + B\bar{B} + \bar{A}B + \bar{A}\bar{A}$$

$$= \bar{B}(A+B) + \bar{A}(A+B) = \overline{\overline{B + \overline{A+B}} + \overline{A + \overline{A+B}}}$$

$$b = \bar{A}B = \bar{A}(A+B) = \overline{\overline{\bar{A}(A+B)}} = \overline{A + \overline{(A+B)}}$$

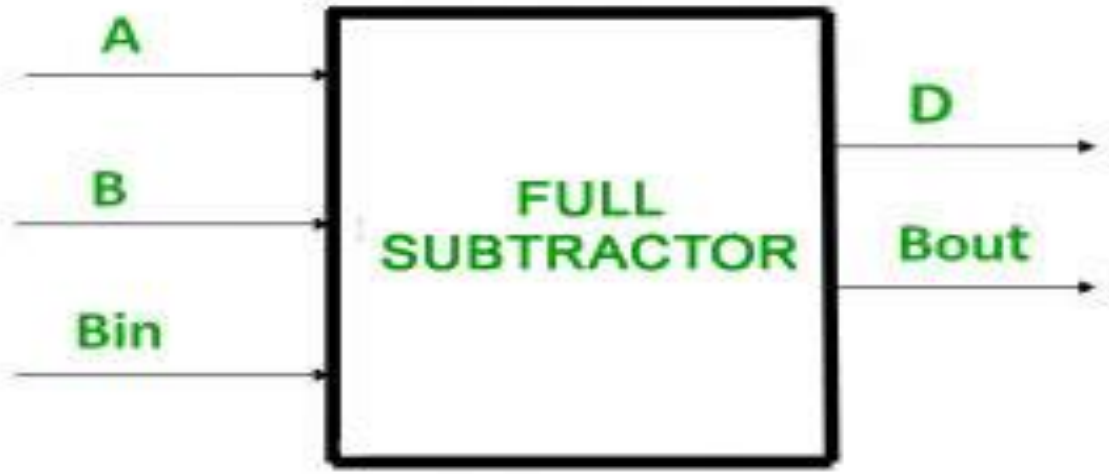


(Logic diagram of a half-subtractor using only 2-input NOR gates)

Fig. (4.14)

Full Subtractor

- A full subtractor is a **combinational circuit** that performs subtraction of two bits, one is minuend and other is subtrahend, taking into account borrow of the previous adjacent lower minuend bit. This circuit **has three inputs and two outputs**.
- The three inputs A, B and Bin, denote the minuend, subtrahend, and previous borrow, respectively. The two outputs, D and Bout represent the difference and output borrow, respectively.



Full Subtractor

- **Truth Table**

INPUT			OUTPUT	
A	B	Bin	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Full Subtractor

- **Logical expression for difference –**

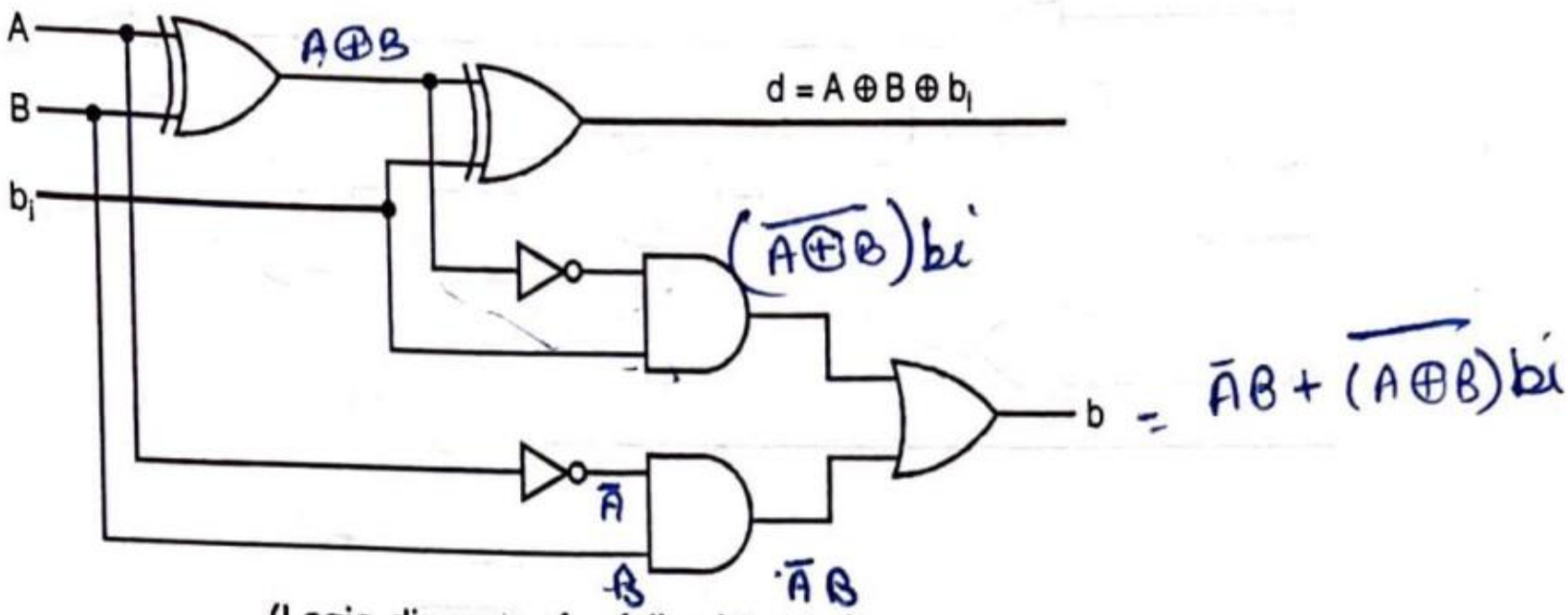
$$
\begin{aligned}
d &= \bar{A}\bar{B}b_i + \bar{A}B\bar{b}_i + A\bar{B}\,\bar{b}_i + ABb_i \\
&= b_i(AB + \bar{A}\bar{B}) + \bar{b}_i(A\bar{B} + \bar{A}B) \\
&= b_i(\overline{A \oplus B}) + b_i(A \oplus B) = A \oplus B \oplus b_i
\end{aligned}
$$

- **Logical expression for Borrow –**

$$
\begin{aligned}
b &= \bar{A}\bar{B}b_i + \bar{A}B\bar{b}_i + \bar{A}Bb_i + ABb_i \\
&= \bar{A}B(b_i + \bar{b}_i) + (AB + \bar{A}\bar{B})b_i \\
&= \bar{A}B + (\overline{A \oplus B})b_i
\end{aligned}
$$

Full Subtractor

- A full subtractor can, therefore be realized using Ex-OR gates and AOI gates as shown in figure



(Logic diagram of a full-subtractor)

Implementation of Full Subtractor using NAND gates

Total 9 NAND gates are required to implement a Full Subtractor

NAND Logic

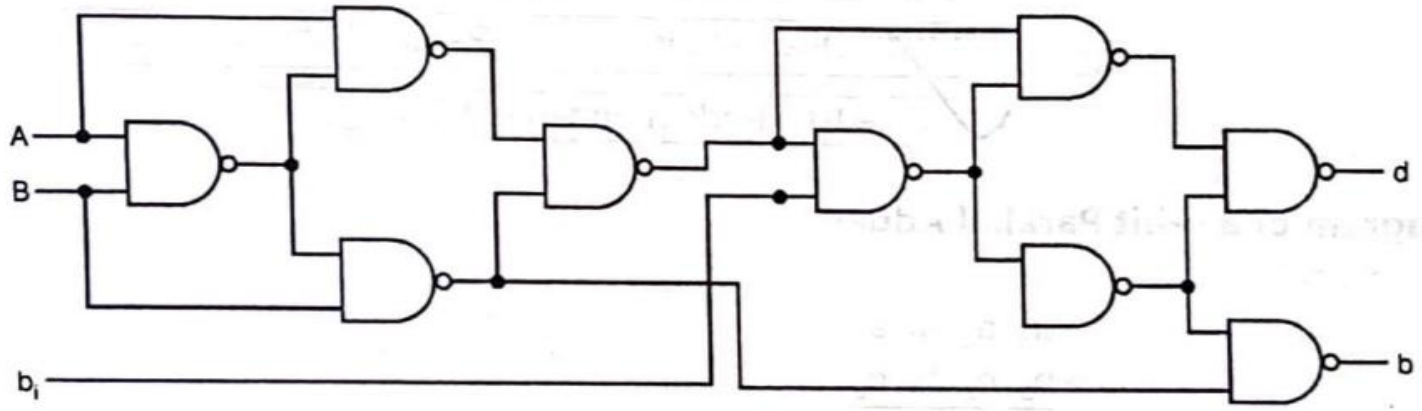
$$d = A \oplus B \oplus b_i = \overline{\overline{(A \oplus B) \oplus b_i}}$$

$$= \overline{\overline{(A \oplus B)\,\overline{(A \oplus B) b_i} \cdot b_i \overline{(A \oplus B) b_i}}}$$

$$b = \bar{A}B + b_i \overline{(A \oplus B)} = \overline{\overline{\bar{A}B + b_i \overline{(A \oplus B)}}}$$

$$= \overline{\overline{\bar{A}B \cdot b_i \overline{(A \oplus B)}}} = \overline{\overline{B(\bar{A} + \bar{B}) \cdot b_i (\bar{b}_i + \overline{(A \oplus B)})}}$$

$$= \overline{\overline{B \cdot \overline{AB} \cdot b_i [\bar{b}_i \cdot (A \oplus B)]}}$$



Logic diagram of a full-subtractor using 2-input NAND gates

Fig. (4.17)

Implementation of Full Subtractor using NOR gates

Total 12 NOR gates are required to implement a Full Subtractor

NOR Logic

$$d = A \oplus B \oplus b_i = \overline{\overline{(A \oplus B) \oplus b_i}} = \overline{(A \oplus B)b_i + \overline{(A \oplus B)}\,\bar{b}_i}$$

$$= \overline{\left[(A \oplus B) + \overline{(A \oplus B)}\,\bar{b}_i\right]\left[b_i + \overline{(A \oplus B)}\,\bar{b}_i\right]}$$

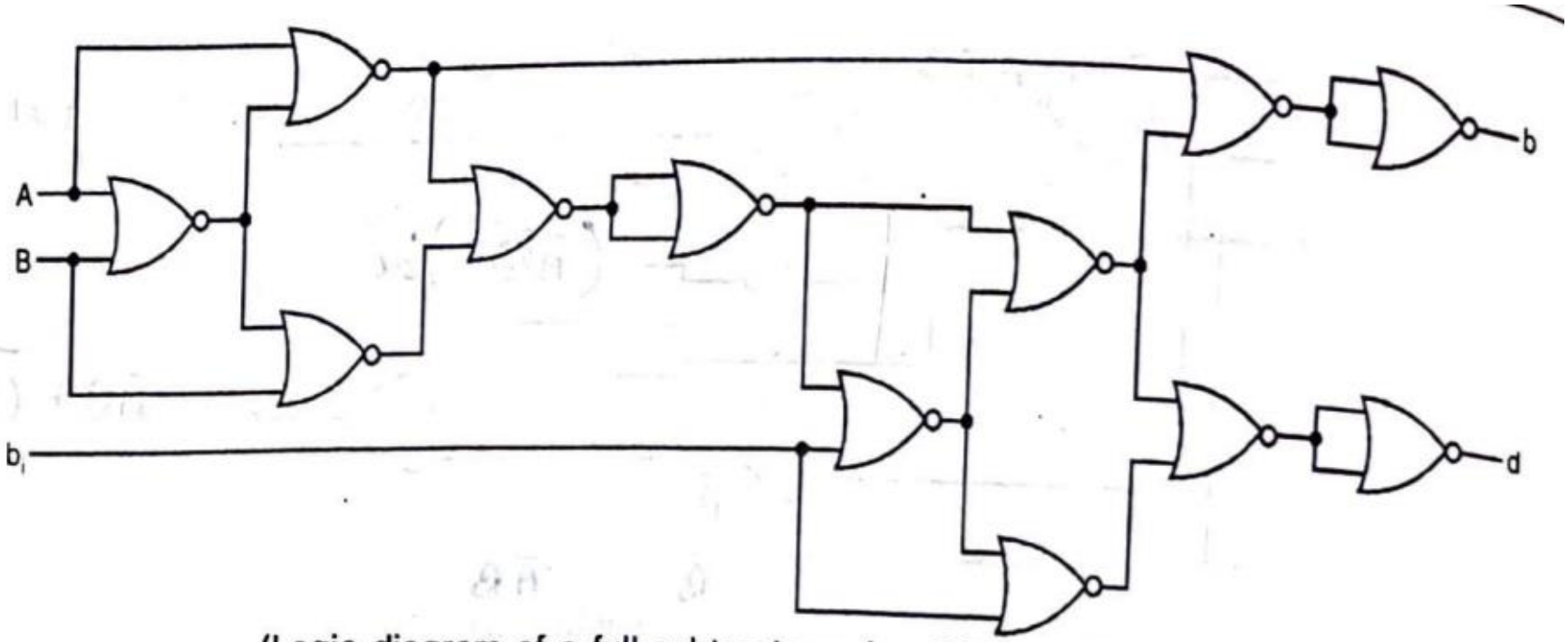
$$= \overline{(A \oplus B) + \overline{(A \oplus B) + b_i} + b_i + \overline{(A \oplus B) + b_i}}$$

$$= \overline{\overline{(A \oplus B) + \overline{(A \oplus B) + b_i} + b_i + \overline{(A \oplus B) + b_i}}}$$

$$b = \bar{A}B + b_i\,\overline{(A \oplus B)} = \bar{A}(A + B) + \overline{(A \oplus B)}\left[(A \oplus B) + b_i\right]$$

$$= \overline{\overline{A + \overline{(A + B)} + (A \oplus B) + \overline{(A \oplus B) + b_i}}}$$

Full Subtractor



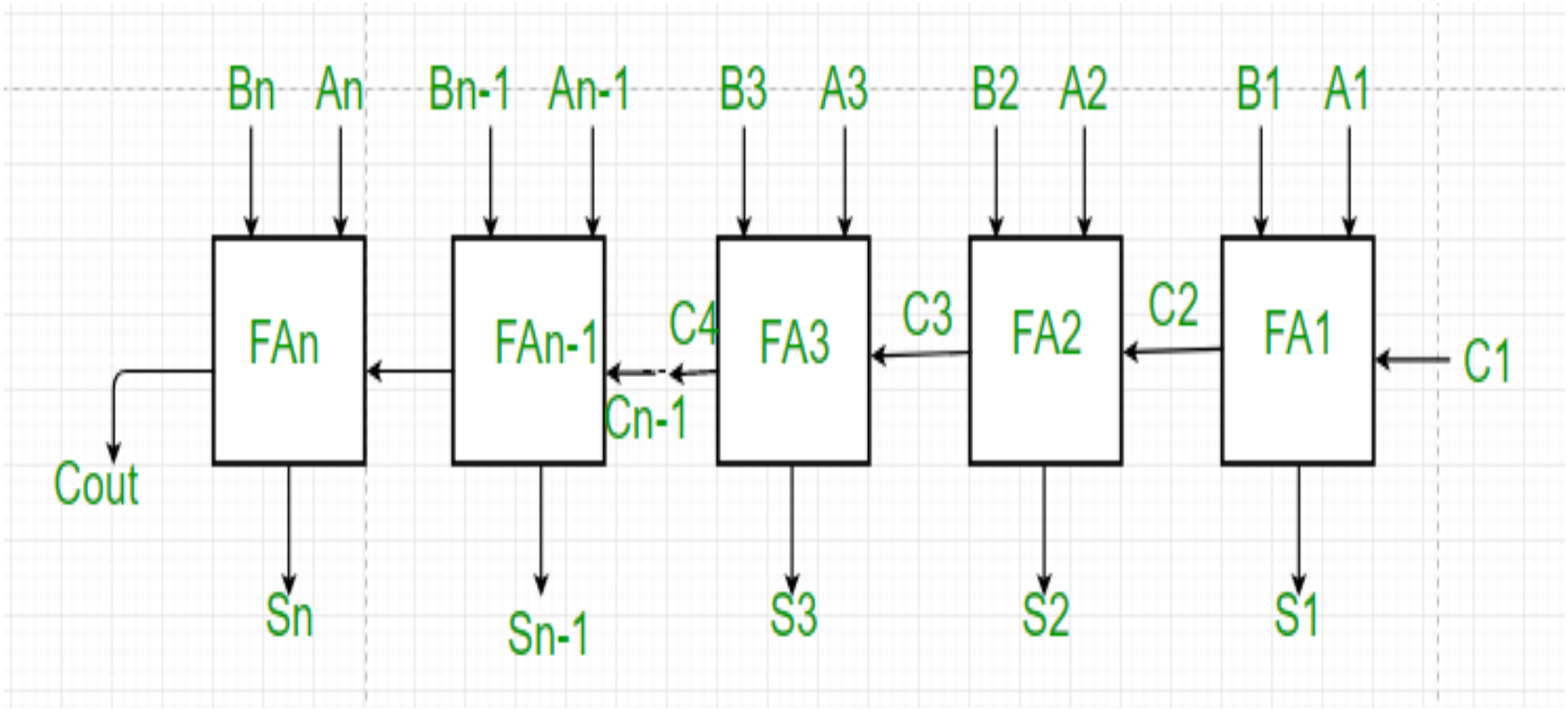
(Logic diagram of a full-subtractor using 2-input NOR gates)

Fig. (4.18)

Parallel Adder

- A single full adder performs the addition of two one bit numbers and an input carry.
- But a **Parallel Adder** is a digital circuit capable of finding the arithmetic **sum** of two binary numbers that is **greater than one bit** in length by operating on corresponding pairs of bits in parallel.
- It consists of **full adders connected in a chain** where the output carry from each full adder is connected to the carry input of the next higher order full adder in the chain.
- **A n bit parallel adder requires n full adders to perform the operation.** So for the two-bit number, two adders are needed while for four bit number, four adders are needed and so on.
- Parallel adders normally incorporate carry lookahead logic to ensure that carry propagation between subsequent stages of addition does not limit addition speed.

Parallel Adder



Parallel Adder

- **Working of parallel Adder –**

1. As shown in the figure, firstly the full adder FA1 adds A1 and B1 along with the carry C1 to generate the sum S1 (the first bit of the output sum) and the carry C2 which is connected to the next adder in chain.

2. Next, the full adder FA2 uses this carry bit C2 to add with the input bits A2 and B2 to generate the sum S2(the second bit of the output sum) and the carry C3 which is again further connected to the next adder in chain and so on.

3. The process continues till the last full adder FAn uses the carry bit Cn to add with its input An and Bn to generate the last bit of the output along last carry bit Cout.

Parallel Adder

- In parallel adder each full adder provide 2 logic gate delay hence n **bit parallel** adder to provide result 2n logic gate or **$2nt_{pd}$**
- In parallel adder as **number of bit increasing speed of operation decreasing** due to carry propagation delay from input to output.
- In parallel adder if each full adder is specified **t_{pdFA}** then to provide **n bit** addition minimum delay is **n* t_{pdFA}**
- If sum and carry delay is given different then final delay will be

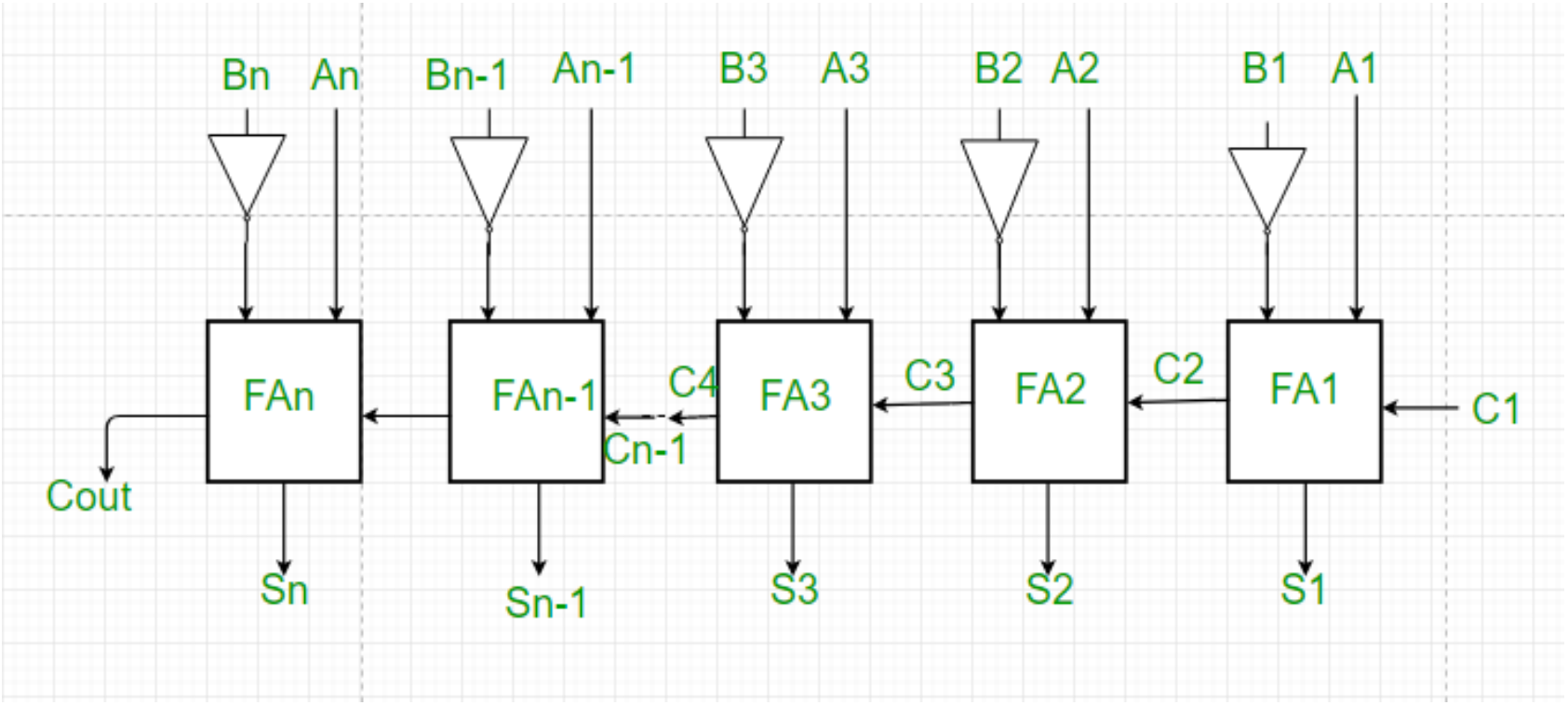
(n-1) carry delay +1* sum delay

Parallel Subtractor

- A Parallel Subtractor is a digital circuit capable of finding the arithmetic difference of two binary numbers that is greater than one bit in length by operating on corresponding pairs of bits in parallel.

- The parallel subtractor can be designed in several ways including combination of half and full subtractors, all full subtractors or all full adders with subtrahend complement input.

Parallel adder and Subtractor



Parallel Subtractor

- **Working of Parallel Subtractor –**

1. As shown in the figure, the parallel binary subtractor is formed by combination of all full adders with subtrahend complement input.
2. This operation considers that the addition of minuend along with the 2's complement of the subtrahend is equal to their subtraction.
3. Firstly the 1's complement of B is obtained by the NOT gate and 1 can be added through the carry to find out the 2's complement of B. This is further added to A to carry out the arithmetic subtraction.
4. The process continues till the last full adder FAn uses the carry bit Cn to add with its input An and 2's complement of Bn to generate the last bit of the output along last carry bit Cout.

4 Bit Parallel Adder & Subtractor

2's Complemented Adder

If we perform (A – B) operation, then we take it as A +(–B) i.e. A(as it is) + (2's complement of B) For this purpose we use a **control circuit** by using **EX-OR gate**. Consider a **4-bit 2's complement parallel adder** as shown in figure (4.21) below:

Let $A = A_3\ A_2\ A_1\ A_0$

$B = B_3\ B_2\ B_1\ B_0$

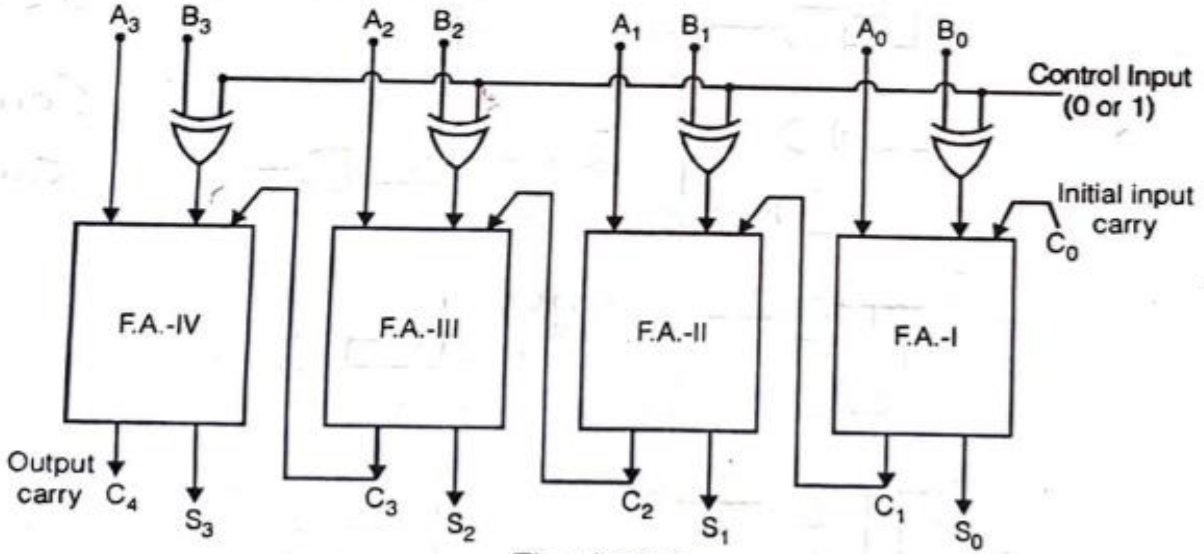


Fig. (4.21)

- When control input = "0" ⇒ circuit acts as an "ADDER".
- When control input = "1" ⇒ circuit acts as a "SUBTRACTOR".

Parallel Adder & Subtractor

- **Advantages of parallel Adder/Subtractor –**

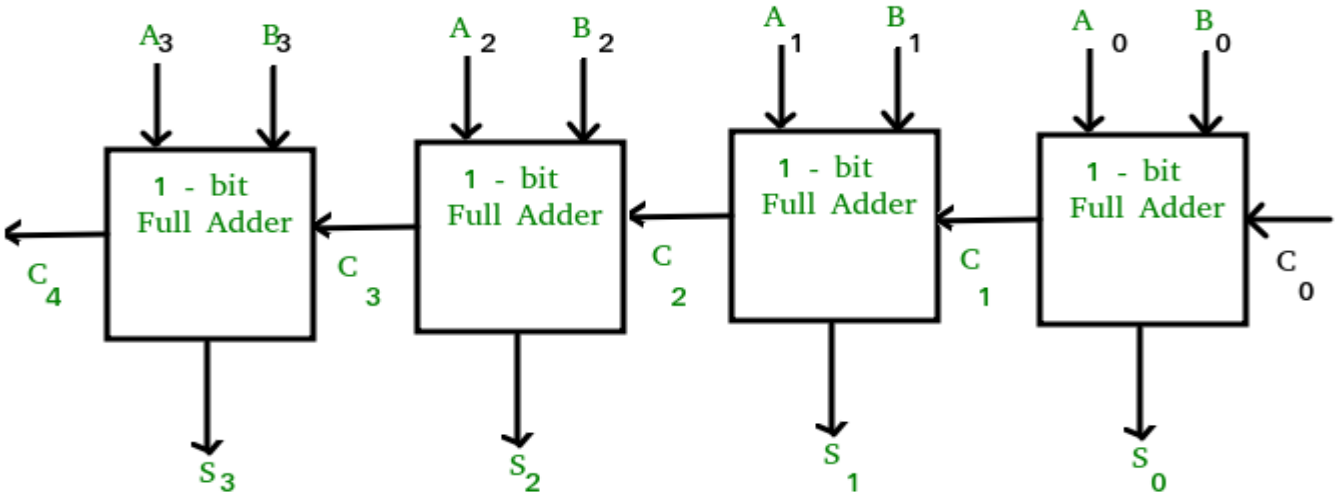
1. The parallel adder/subtractor performs the addition operation faster as compared to serial adder/subtractor.
2. Time required for addition does not depend on the number of bits.
3. The output is in parallel form i.e all the bits are added/subtracted at the same time.
4. It is less costly.

- **Disadvantages of parallel Adder/Subtractor –**

1. Each adder has to wait for the carry which is to be generated from the previous adder in chain.
2. The propagation delay(delay associated with the travelling of carry bit) is found to increase with the increase in the number of bits to be added.

Look-Ahead Carry Adders

- **Motivation behind Carry Look-Ahead Adder:** In ripple carry adders, for each adder block, the two bits that are to be added are available instantly. However, each adder block waits for the carry to arrive from its previous block.
- So, it is not possible to generate the sum and carry of any block until the input carry is known. The **i^{th}** block waits for the **$i\text{-}1^{th}$** block to produce its carry. So, there will be a considerable time delay which is carry propagation delay.



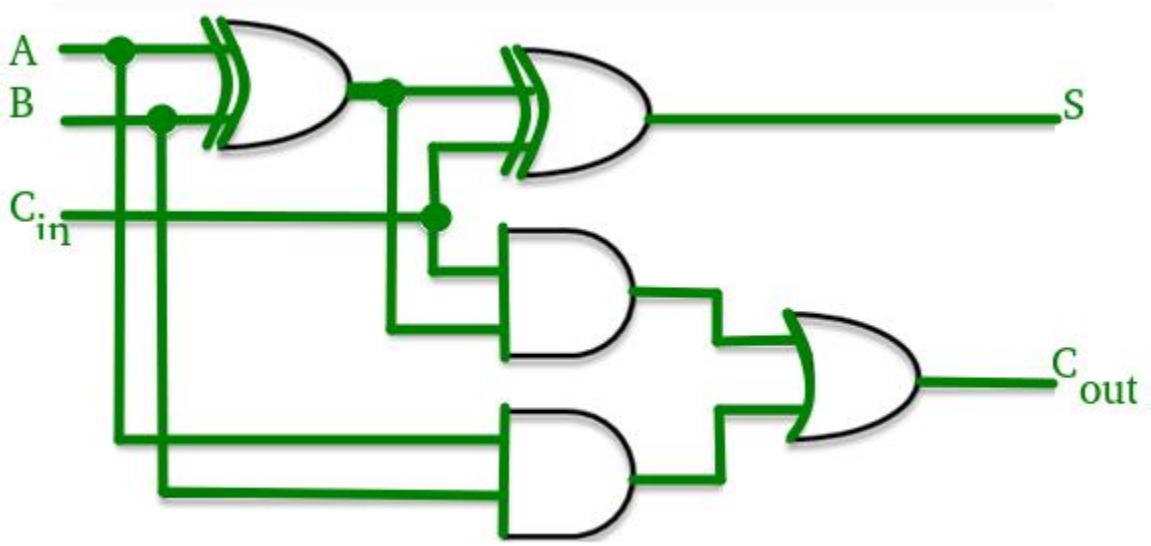
Look-Ahead Carry Adders

- Consider the above 4-bit ripple carry adder. The sum **S3** is produced by the corresponding full adder as soon as the input signals are applied to it. But the carry input C4is not available on its final steady state value until carry C3 is available at its steady state value.
- Similarly, C3 depends on C2 and C2 on C1. Therefore, though the carry must propagate to all the stages in order that output S3 and carry C4 settle their final steady-state value.

- The propagation time is equal to the propagation delay of each adder block, multiplied by the number of adder blocks in the circuit. For example, if each full adder stage has a propagation delay of 20 nanoseconds, then will reach its final correct value after 60 (20 × 3) nanoseconds. The situation gets worse, if we extend the number of stages for adding more number of bits.

Look-Ahead Carry Adders

- A carry look-ahead adder reduces the propagation delay by introducing more complex hardware. In this design, the ripple carry design is suitably transformed such that the carry logic over fixed groups of bits of the adder is reduced to two-level logic. Let us discuss the design in detail.



Look-Ahead Carry Adders

- Consider the full adder circuit shown above with corresponding truth table. We define two variables as **'carry generate'** Gi and **'carry propagate'** Pi then,
- $P_i = A_i \oplus B_i$
- $G_i = A_i B_i$

A	B	C	C +1	Condition
0	0	0	0	
0	0	1	0	No Carry
0	1	0	0	Generate
0	1	1	1	
1	0	0	0	No Carry
1	0	1	1	Propogate
1	1	0	1	Carry
1	1	1	1	Generate

Look-Ahead Carry Adders

- Consider the full adder circuit shown above with corresponding truth table. We define two variables as **'carry generate'** Gi and **'carry propagate'** Pi then,

$$\mathbf{P_i = A_i \oplus B_i}$$
$$\mathbf{G_i = A_i B_i}$$

- The sum output and carry output can be expressed in terms of carry generate G_i and carry propagate P_i as

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

- where G_i produces the carry when both A_i, B_i are 1 regardless of the input carry. P_i is associated with the propagation of carry from C_i to C_{i+1}.

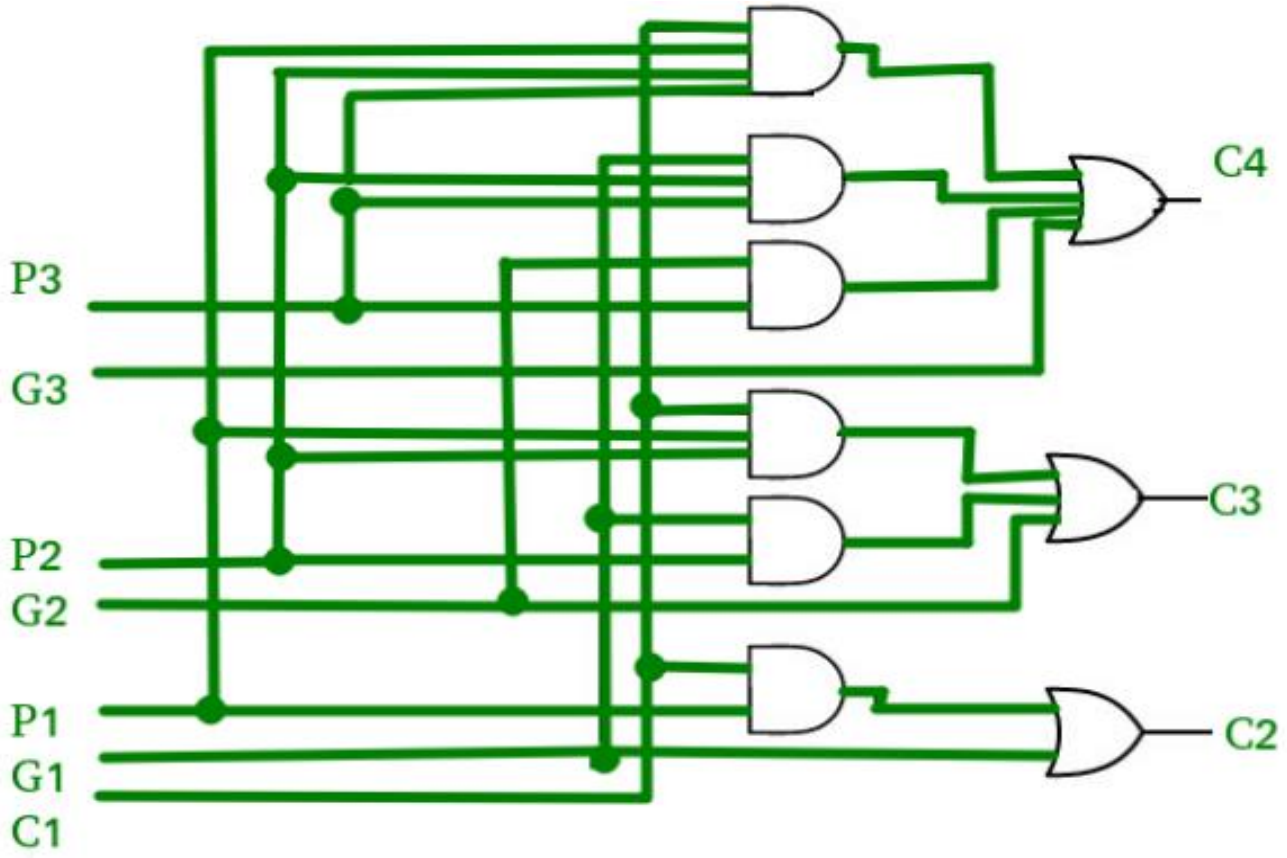
Look-Ahead Carry Adders

- **The carry output Boolean function of each stage in a 4 stage carry look-ahead adder can be expressed as**

$$
\begin{aligned}
C_1 &= G_0 + P_0 C_{in} \\
C_2 &= G_1 + P_1 C_1 = G_1 + P_1 G_0 + P_1 P_0 C_{in} \\
C_3 &= G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{in} \\
C_4 &= G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{in}
\end{aligned}
$$

- From the above Boolean equations, we can observe that C_4 does not have to wait for C_3 and C_2 to propagate but actually C_4 is propagated at the same time as C_3 and C_2. Since the Boolean expression for each carry output is the sum of products so these can be implemented with one level of AND gates followed by an OR gate.
- The implementation of three Boolean functions for each carry output (C_2, C_3 and C_4) for a carry look-ahead carry generator shown in below figure.

Look-Ahead Carry Adders



Look-Ahead Carry Adders

Advantages and Disadvantages of Carry Look-Ahead Adder

- **Advantages**
- The propagation delay is reduced.
- It provides the fastest addition logic.

- **Disadvantages**
- The Carry Look-ahead adder circuit gets complicated as the number of variables increase.
- The circuit is costlier as it involves more number of hardware.

Multiplexer

Demultiplexer

BCD Adder and BCD subtractor

4.11 BCD Adder

The digital systems handles the decimal number in the form of binary coded decimal numbers (BCD). A BCD adder is a circuit that adds two BCD digits and produces a sum digit also in BCD. BCD numbers use 10 digits, 0 to 9 which are represented in the binary form 0 0 0 0 to 1 0 0 1, i.e. each BCD digit is represented as a 4-bit binary number. When we write BCD number say 526, it can be represented as

5	2	6
↓	↓	↓
0 1 0 1	0 0 1 0	0 1 1 0

Here, we should note that BCD cannot be greater than 9.

The addition of two BCD numbers can be best understood by considering the three cases that occur when two BCD digits are added.

Sum equals 9 or less with carry 0

Let us consider additions of 3 and 6 in BCD.

6	0 1 1 0	← BCD for 6
+ 3	0 0 1 1	← BCD for 3
9	1 0 0 1	← BCD for 9

The addition is carried out as in normal binary addition and the sum is 1 0 0 1, which is BCD code for 9.

Cont…

Sum greater than 9 with carry 0

Let us consider addition of 6 and 8 in BCD.

6	0 1 1 0	← BCD for 6
+ 8	1 0 0 0	← BCD for 8
14	1 1 1 0	← Invalid BCD number

The sum 1 1 1 0 is an invalid BCD number. This has occurred because the sum of the two digits exceeds 9. Whenever this occurs the sum has to be corrected by the addition of six (0110) in the invalid BCD number, as shown below.

6	0 1 1 0	← BCD for 6
+ 8	1 0 0 0	← BCD for 8
14	1 1 1 0	← Invalid BCD number
	+ 0 1 1 0	← Add 6 for correction
0 0 0 1 (1)	0 1 0 0 (4)	← BCD for 14

Cont…

After addition of 6 carry is produced into the second decimal position.

Sum equals 9 or less with carry 1

Let us consider addition of 8 and 9 in BCD.

	8				1	0	0	0	← BCD for 8
+	9				1	0	0	1	← BCD for 9
	17	0	0	0	1	0	0	0	1 ← Incorrect BCD result

In this case, result (0001 0001) is valid BCD number, but it is incorrect. To get the correct BCD result correction factor of 6 has to be added to the least significant digit sum, as shown below.

	8				1	0	0	0	← BCD for 8
+	9				1	0	0	1	← BCD for 9
	17	0	0	0	1	0	0	0	1 ← Incorrect BCD result
	+	0	0	0	0	0	1	1	0 ← Add 6 for correction
		0	0	0	1	0	1	1	1 ← BCD for 17

Going through these three cases of BCD addition we can summarize the BCD addition procedure as follows.

Cont…

0 0 0 1 0 1 1 1 ← BCD for 17

Going through these three cases of BCD addition we can summarize the BCD addition procedure as follows.

1. Add two BCD numbers using ordinary binary addition.
2. If four-bit sum is equal to or less than 9, no correction is needed. The sum is in proper BCD form.
3. If the four-bit sum is greater than 9 or if a carry is generated from the four-bit sum, the sum is invalid.
4. To correct the invalid sum, add 0110_2 to the four-bit sum. If a carry results from this addition, add it to the next higher-order BCD digit.

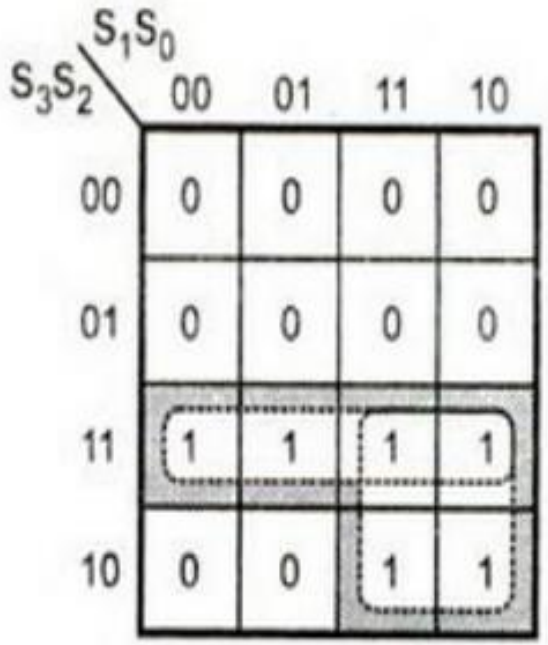
Thus to implement BCD adder we require :

- 4-bit binary adder for initial addition.
- Logic circuit to detect sum greater than 9 and
- One more 4-bit adder to add 0110_2 in the sum if sum is greater than 9 or carry is 1.

The logic circuit to detect sum greater than 9 can be determined by simplifying the Boolean expression of given truth table.

Cont…

Inputs				Output
S_3	S_2	S_1	S_0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1



$$Y = S_3S_2 + S_3S_1$$

Y = 1 indicates sum is greater than 9. We can put one more term, C_{out} in the above expression to check whether carry is one. If any one condition is satisfied we add 6 (0110) in the sum.

Cont…

With this design information we can draw the block diagram of BCD adder, as shown in the Fig. 4.102.

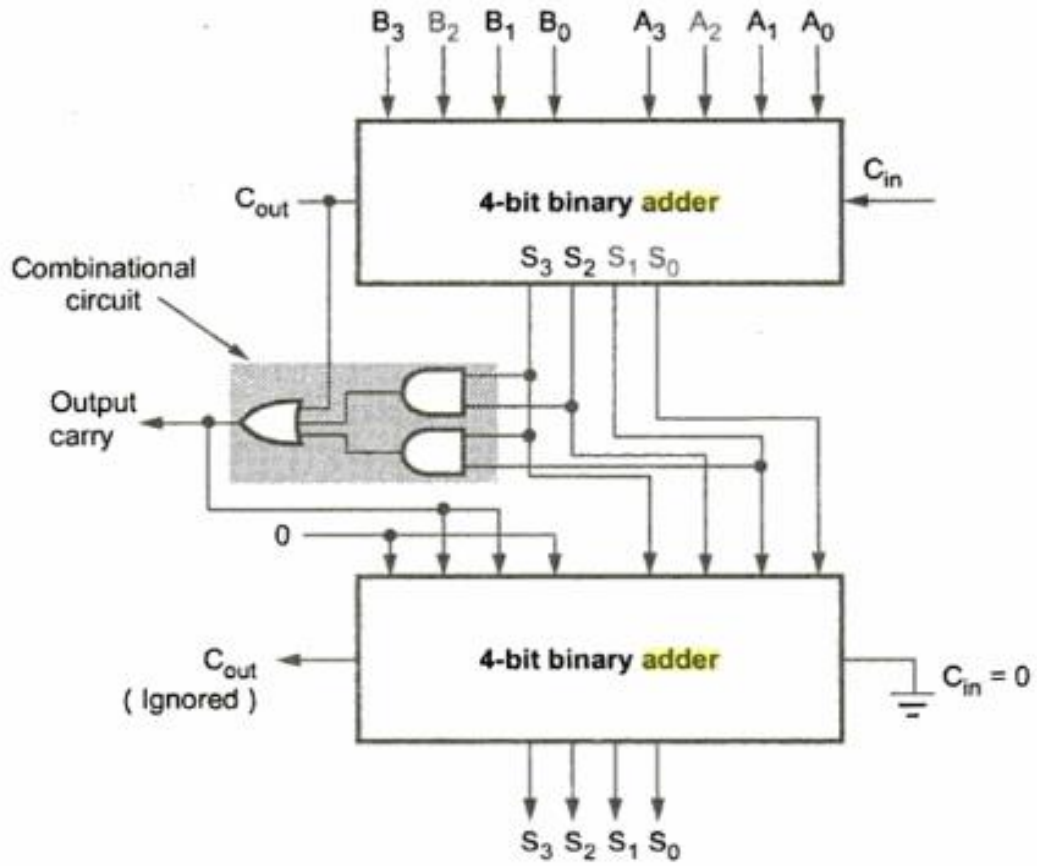


Fig. 4.102 Block diagram of BCD adder

Cont…

As shown in the Fig. 4.102, the two BCD numbers, together with input carry, are first added in the top 4-bit binary adder to produce a binary sum. When the output carry is equal to zero (i.e. when sum ≤ 9 and C_{out} = 0) nothing (zero) is added to the binary sum. When it is equal to one (i.e. when sum > 9 or C_{out} = 1), binary 0110 is added to the binary sum through the bottom 4-bit binary adder. The output carry generated from the bottom binary adder can be ignored, since it supplies information already available at the output-carry terminal.

Parity Checker/Generator

- **EVEN PARITY GENERATOR**

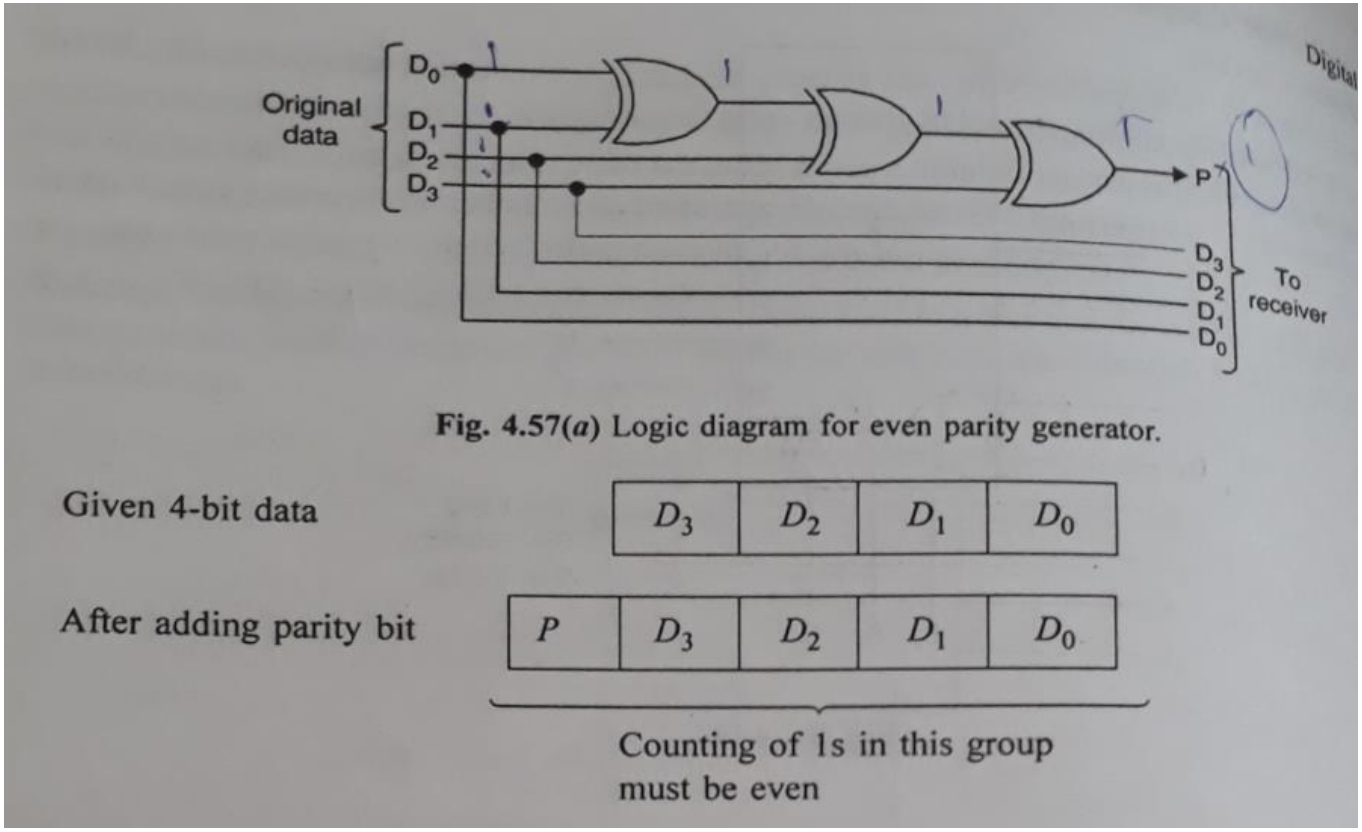


Fig. 4.57(*a*) Logic diagram for even parity generator.

Odd Parity Generator

Odd parity generator

To generate an odd parity bit four data bits are added using three XOR gates and the sum bit is inverted. Fig. 4.57(*b*) shows the logic diagram of an odd parity generator.

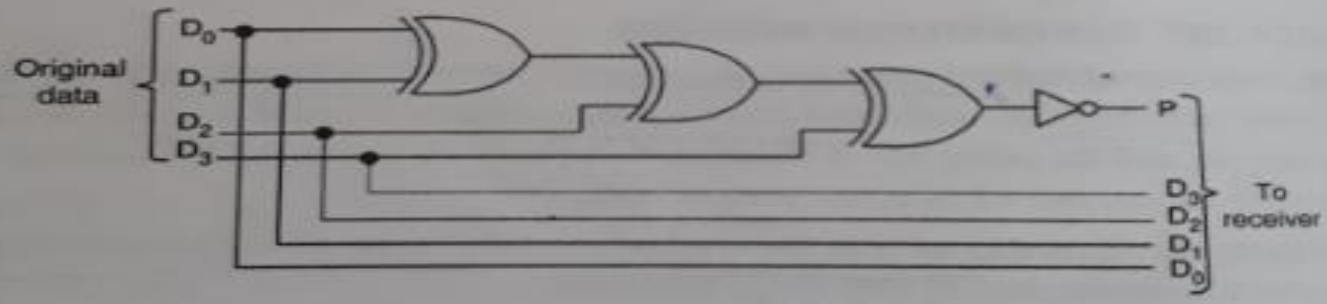
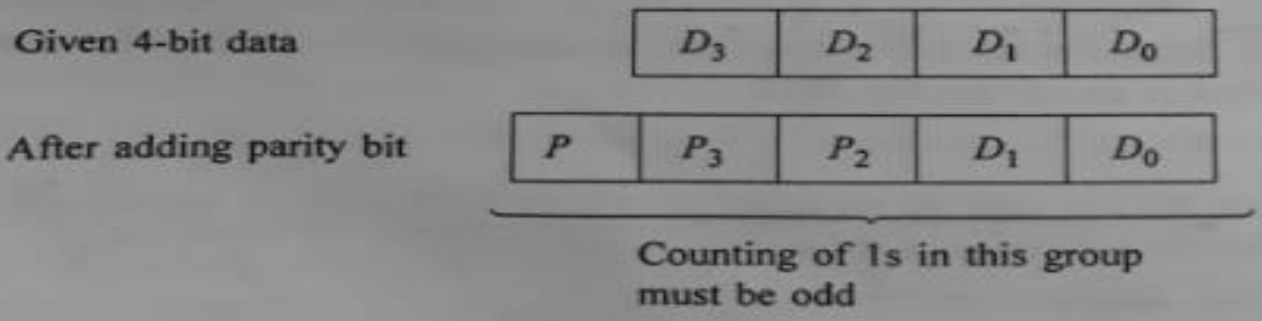


Fig. 4.57(*b*) Logic diagram for odd parity generator.



3 bit Even Parity Generator

4.12.3 3-bit Even Parity Generator

In even parity the added bit (parity bit) will make the number of 1's an even. Table 4.26 shows the truth table of a 3 bit even parity generator.

Table 4.26 Truth Table for 3-bit even parity generator

3-bit message			*Even parity bit*
A	*B*	*C*	*P*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

First, get the simplified boolean expression for above mentioned 3 bit even parity generator by using three variable K-maps.

3 bit Even Parity Generator

A \ BC	00	01	11	10
0	0	1	3	2
1	4	5	7	6

$$
\begin{aligned}
P &= \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC \\
&= \bar{A}(\bar{B}C + B\bar{C}) + A(\bar{B}\bar{C} + BC) \\
&= \bar{A}(B \oplus C) + A(B \odot C) \\
&= \bar{A}(B \oplus C) + A\overline{(B \oplus C)} \\
&= A \oplus (B \oplus C)
\end{aligned}
$$

The 3 bit even parity generator logic circuit is constructed by using EXOR gates, It is shown in Fig. 4.59

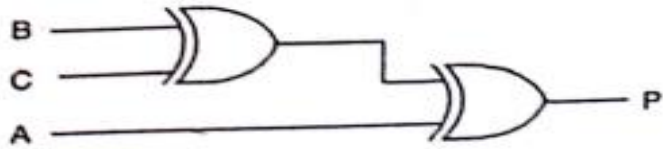


Fig. 4.59 Logic circuit for 3 bit even parity generator.

3 bit Even Parity Generator

4.12.4 3 bit even Parity Checker

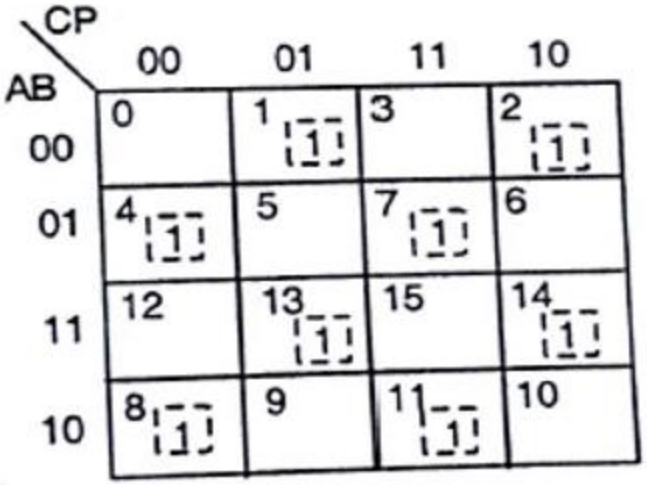
The binary message together with the parity bit are transmitted to the destination. The received information is given to input of parity checker. The parity checker circuit checks for possible errors in the transmission. Since the information was transmitted with even parity, the four bits (three bit for binary message and one bit for parity bit) must have an even numbers of 1's. An error has occured during transmission if the four bits received have an odd number of 1's, indicating that one bit has changed during transmission. The output of parity checker is represented P_C (parity checker) and table 4.27 shows the truth table of even parity checker

Table 4.27 Truth table of even parity checker

Received bits				*Parity checker output*
A	*B*	*C*	*P*	P_C
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

K-map simplification

Expression for P_C



3 bit Even Parity Generator

$$
\begin{aligned}
P_C = {} & \overline{A}\,\overline{B}(\overline{C}P + C\overline{P}) + \overline{A}B(\overline{C}\,\overline{P} + CP) \\
& + AB(\overline{C}P + C\overline{P}) + A\overline{B}(\overline{C}\,\overline{P} + CP) \\
= {} & \overline{A}\,\overline{B}(C \oplus P) + \overline{A}B(C \odot P) + AB(C \oplus P) + A\overline{B}(C \odot P) \\
= {} & (C \oplus P)(\overline{A}\,\overline{B} + AB) + (C \odot P)(\overline{A}B + A\overline{B}) \\
= {} & (C \oplus P)(A \odot B) + (C \odot P)(A \oplus B) \\
= {} & (C \oplus P)\overline{(A \oplus B)} + \overline{(C \oplus P)}(A \odot B) \\
= {} & (A \oplus B) \oplus (C \oplus P)
\end{aligned}
$$

The even parity checker for the above expression is constructed by using EXOR gate, it is sh in Fig. 4.60. The odd parity checker logic circuit is also obtained when we use last gate as EX gate instead of EXOR gate (Replace gate 3 by EXNOR gate for odd parity checker)

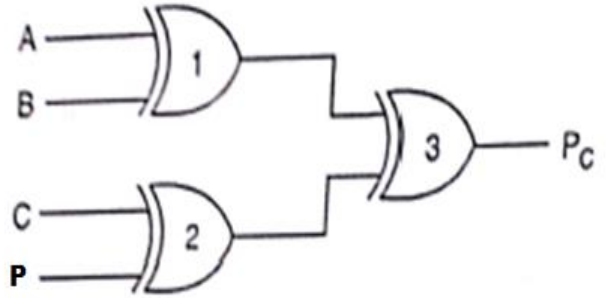


Fig. 4.60 Logic circuit for even parity generator.

Encoder

- Binary code of N digits can be used to store 2^N distinct elements of coded information. This is what encoders and decoders are used for. **Encoders** convert 2^N lines of input into a code of N bits and **Decoders** decode the N bits into 2^N lines.

Encoder

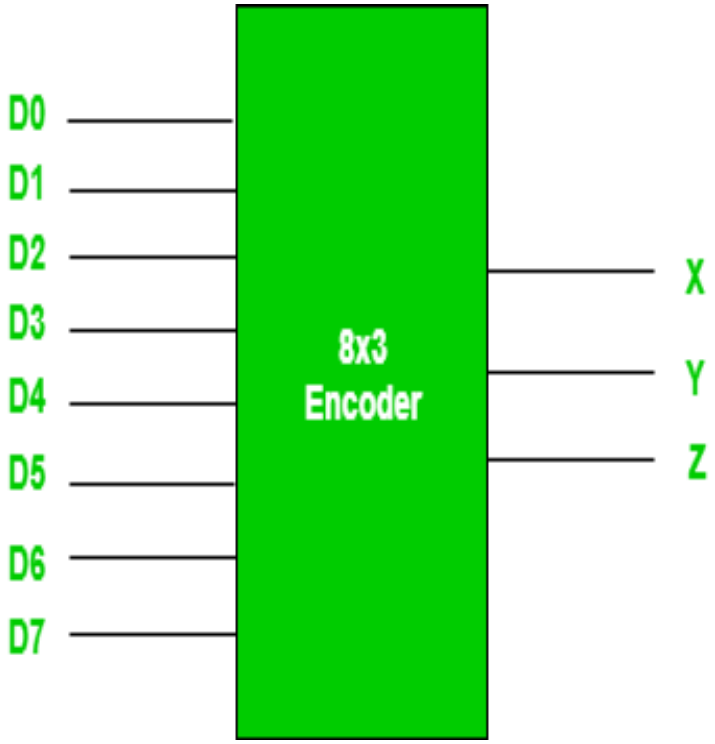
- An encoder is a combinational circuit that converts binary information in the form of a 2^N input lines into N output lines, which represent N bit code for the input. For simple encoders, it is assumed that only one input line is active at a time.

- As an example, let's consider **Octal to Binary** encoder. As shown in the following figure, an octal-to-binary encoder takes 8 input lines and generates 3 output lines.

Encoder

Truth Table –

D7	D6	D5	D4	D3	D2	D1	D0	X	Y	Z
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



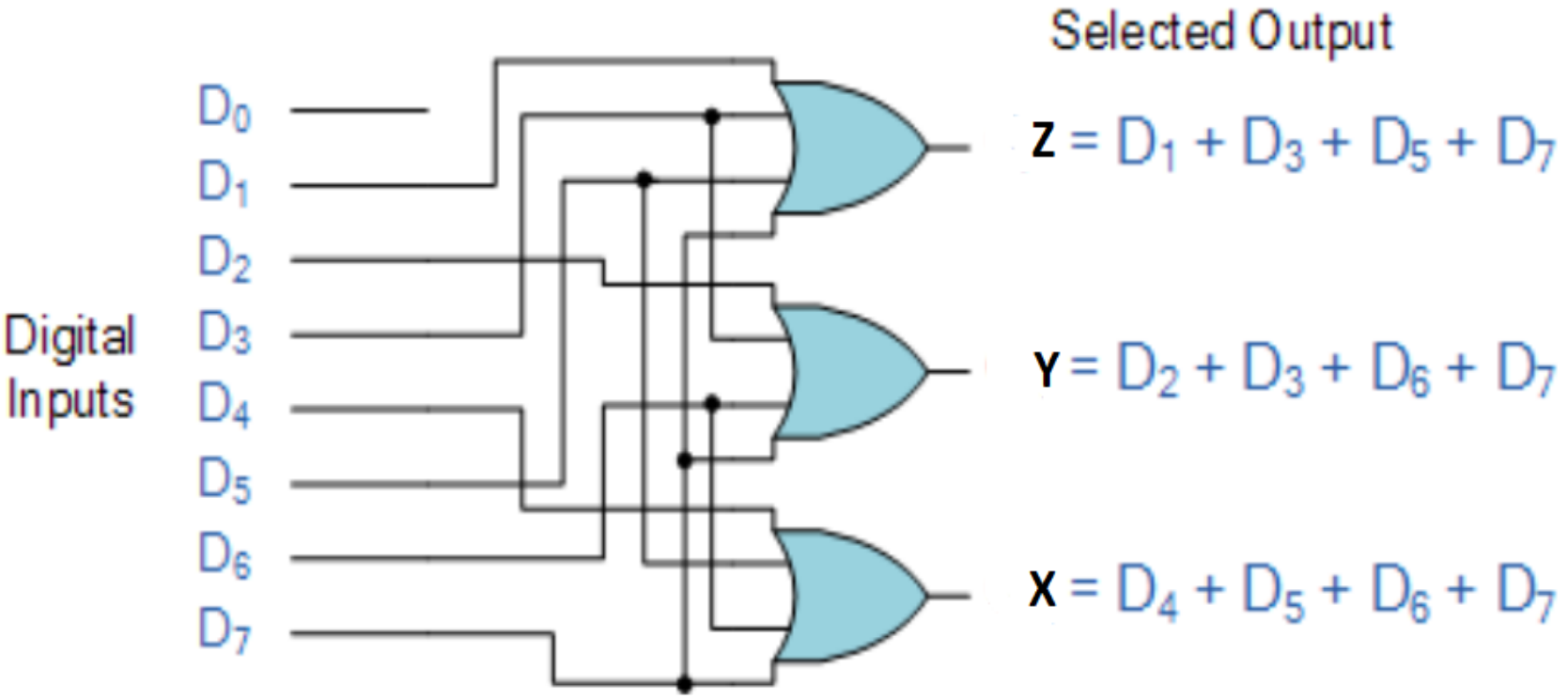
Encoder

- As seen from the truth table, the output is 000 when D0 is active; 001 when D1 is active; 010 when D2 is active and so on.

- **Implementation** – From the truth table, the output line Z is active when the input octal digit is 1, 3, 5 or 7. Similarly, Y is 1 when input octal digit is 2, 3, 6 or 7 and X is 1 for input octal digits 4, 5, 6 or 7. Hence, the Boolean functions would be:
- X = D4 + D5 + D6 + D7
- Y = D2 +D3 + D6 + D7
- Z = D1 + D3 + D5 + D7

Encoder

- Hence, the encoder can be realized with OR gates as follows:



Encoder

- **Limitations:**

- One limitation of this encoder is that only one input can be active at any given time. If more than one inputs are active, then the output is undefined. For example, if D6 and D3 are both active, then, our output would be 111 which is the output for D7. To overcome this, we use Priority Encoders.

- Another ambiguity arises when all inputs are 0. In this case, encoder outputs 000 which actually is the output for D0 active. In order to avoid this, an extra bit can be added to the output, called the valid bit which is 0 when all inputs are 0 and 1 otherwise.

Priority Encoder

Priority Encoder:

- A priority encoder is an encoder circuit in which inputs are given priorities. When more than one inputs are active at the same time, the input with higher priority takes precedence and the output corresponding to that is generated.
- Let us consider the 4 to 2 priority encoder as an example. From the truth table, we see that when all inputs are 0, our V bit or the valid bit is zero and outputs are not used. The x's in the table show the don't care condition, i.e, it may either be 0 or 1. Here, D3 has highest priority, therefore, whatever be the other inputs, when D3 is high, output has to be 11. And D0 has the lowest priority, therefore the output would be 00 only when D0 is high and the other input lines are low. Similarly, D2 has higher priority over D1 and D0 but lower than D3 therefore the output would be 010 only when D2 is high and D3 are low (D0 & D1 are don't care).

Priority Encoder

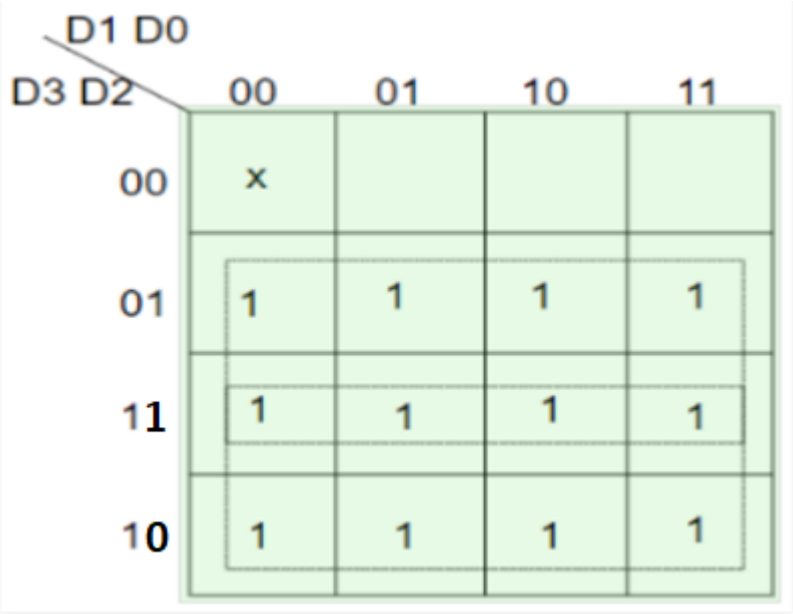
- **Truth Table –**

D3	D2	D1	D0	X	Y	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

Priority Encoder

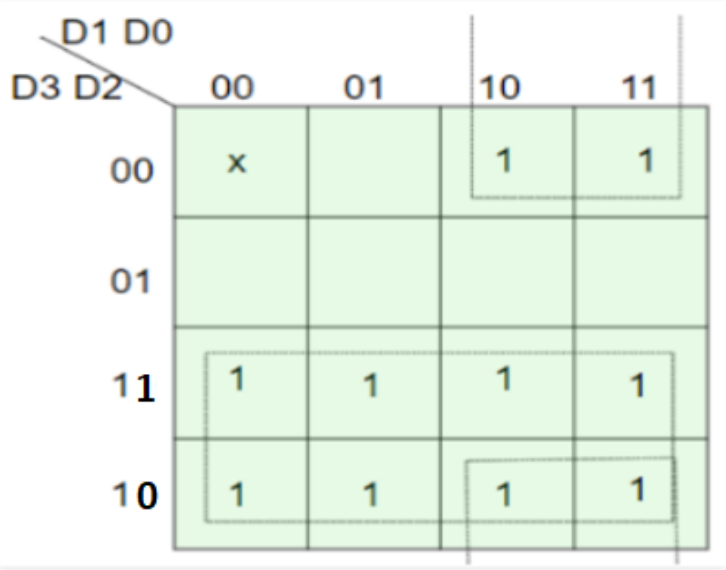
- **Implementation –**
  It can clearly be seen that the condition for valid bit to be 1 is that at least any one of the inputs should be high. Hence,
- `V = D0 + D1 + D2 + D3`

For X:



X = D2 + D3

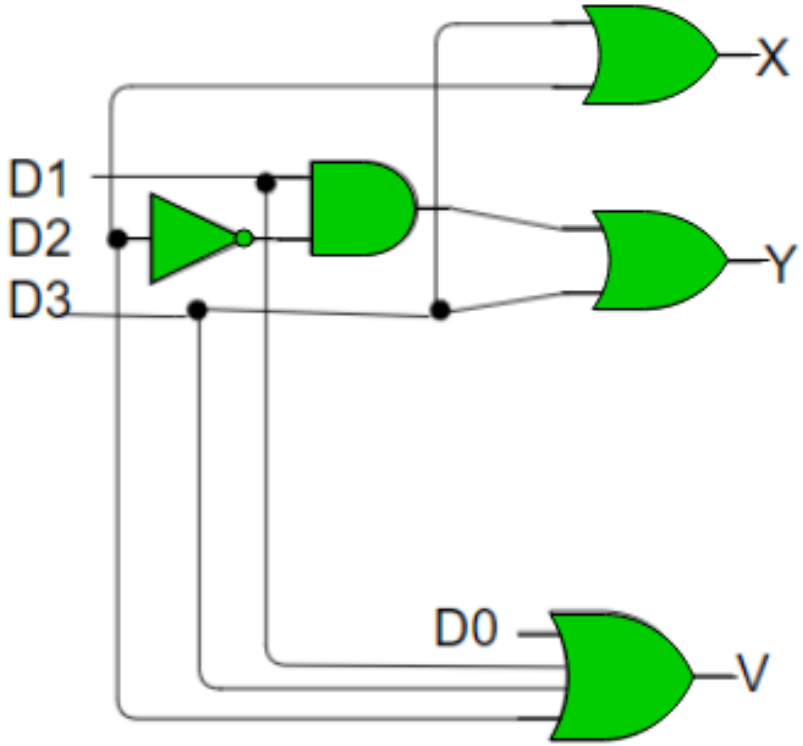
For Y:



Y = D1 D2′ + D3

Priority Encoder

- Hence, the priority 4-to-2 encoder can be realized as follows:



Decoder

- A decoder does the opposite job of an encoder. It is a combinational circuit that converts n lines of input into 2^n lines of output.
- Let's take an example of 3-to-8 line decoder.
- **Truth Table –**

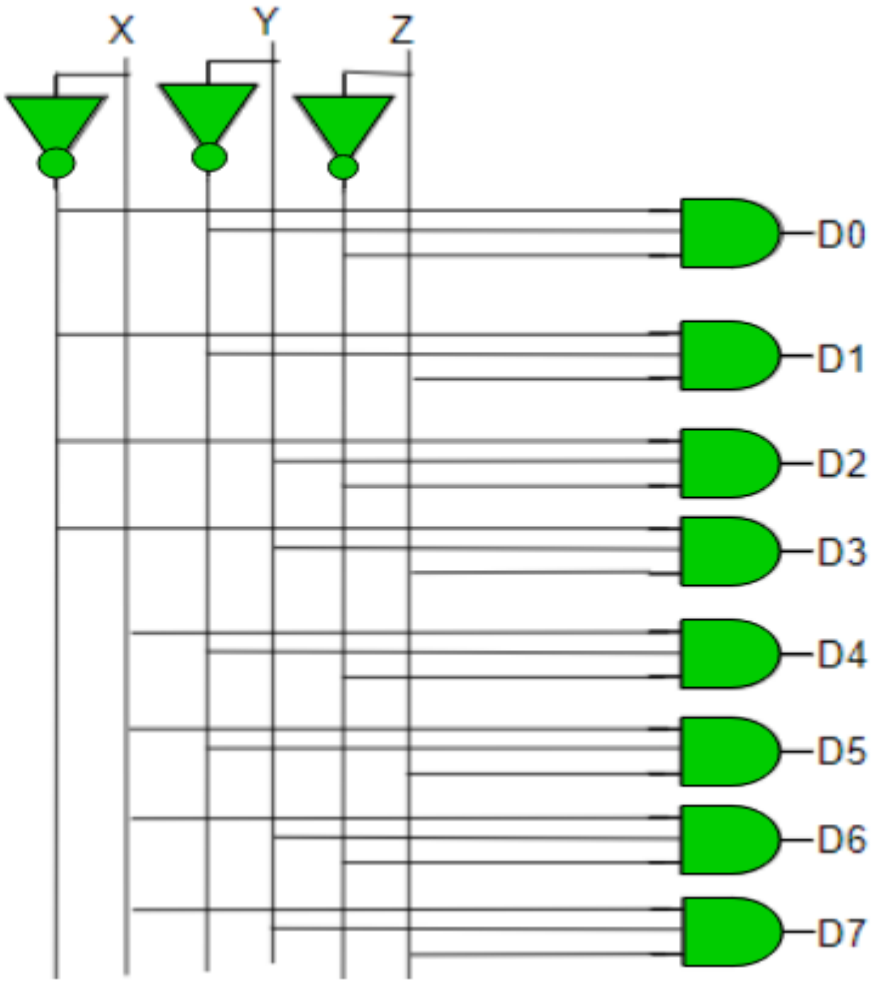
X	Y	Z	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Decoder

- **Implementation –**
  D0 is high when X = 0, Y = 0 and Z = 0. Hence,
- D0 = X’ Y’ Z’
- **Similarly**
- D1 = X’ Y’ Z
- D2 = X’ Y Z’
- D3 = X’ Y Z
- D4 = X Y’ Z’
- D5 = X Y’ Z
- D6 = X Y Z’
- D7 = X Y Z

Decoder

- **Logic Circuit**



BCD to Seven segment Display Decoder/Driver

- In **Binary Coded Decimal (BCD)** encoding scheme each of the decimal numbers(0-9) is represented by its equivalent binary pattern(which is generally of 4-bits).

- Whereas, **Seven segment** display is an electronic device which consists of seven Light Emitting Diodes (LEDs) arranged in a some definite pattern (common cathode or common anode type), which is used to display Hexadecimal numerals(in this case decimal numbers,as input is BCD i.e., 0-9).

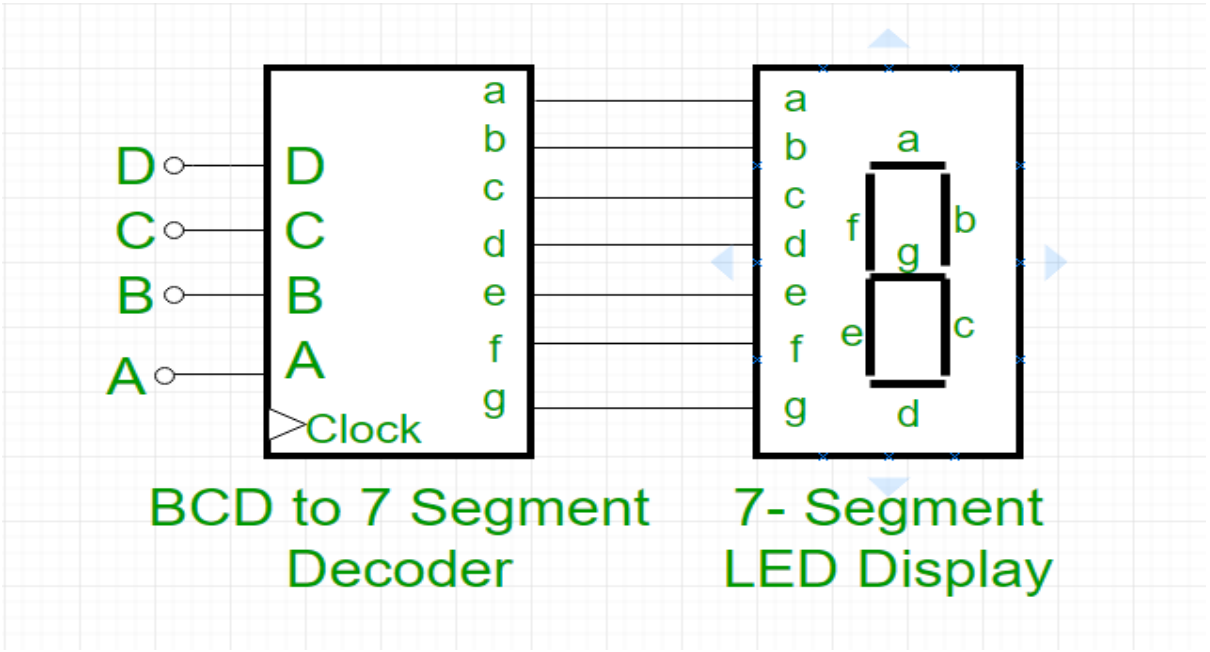
Two types of seven segment LED display:

1. **Common Cathode Type:** In this type of display all cathodes of the seven LEDs are connected together to the ground or -Vcc(hence,common cathode) and LED displays digits when some 'HIGH' signal is supplied to the individual anodes.

2. **Common Anode Type:** In this type of display all the anodes of the seven LEDs are connected to battery or +Vcc and LED displays digits when some 'LOW' signal is supplied to the individual cathodes.

Cont…

- But, seven segment display does not work by directly supplying voltage to different segments of LEDs. First, our decimal number is changed to its BCD equivalent signal then BCD to seven segment decoder converts that signals to the form which is fed to seven segment display.
- This BCD to seven segment decoder has four input lines (A, B, C and D) and 7 output lines (a, b, c, d, e, f and g), this output is given to seven segment LED display which displays the decimal number depending upon inputs.



Cont...

- **Truth Table –** For common cathode type BCD to seven segment decoder:

Decimal Digit	Input lines				Output lines							Display pattern
	A	B	C	D	a	b	c	d	e	f	g	
0	0	0	0	0	1	1	1	1	1	1	0	
1	0	0	0	1	0	1	1	0	0	0	0	
2	0	0	1	0	1	1	0	1	1	0	1	
3	0	0	1	1	1	1	1	1	0	0	1	
4	0	1	0	0	0	1	1	0	0	1	1	
5	0	1	0	1	1	0	1	1	0	1	1	
6	0	1	1	0	1	0	1	1	1	1	1	
7	0	1	1	1	1	1	1	0	0	0	0	
8	1	0	0	0	1	1	1	1	1	1	1	
9	1	0	0	1	1	1	1	1	0	1	1	

Cont...

- **Note:** For Common Anode type seven segment LED display, we only have to interchange all ‘0s’ and ‘1s’ in the output side i.e., (for a, b, c, d, e, f, and g replace all ‘1’ by ‘0’ and vice versa) and solve using K-map.
- Output for first combination of inputs (A, B, C and D) in Truth Table corresponds to ‘0’ and last combination corresponds to ‘9’. Similarly rest corresponds from 2 to 8 from top to bottom.
- BCD numbers only range from 0 to 9,thus rest inputs from 10-F are invalid inputs.
- **Explanation** –
  For combination where all the inputs (A, B, C and D) are zero (see Truth Table), our output lines are a = 1, b = 1, c = 1, d = 1, e = 1, f = 1 and g = 0. So 7 segment display shows ‘zero’ as output.
- Similarly, for combination where one of the input is one (D = 1) and rest are zero, our output lines are a = 0, b = 1, c = 1, d = 0, e = 0, f = 0 and g = 0. So only LEDs ‘b’ and ‘c’ (see diagram above) will glow and 7 segment display shows ‘one’ as output.

Cont...

- **K-Maps:**

For a

AB\CD	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

For b

AB\CD	00	01	11	10
00	1	1	1	1
01	1	0	1	0
11	X	X	X	X
10	1	1	X	X

For c

AB\CD	00	01	11	10
00	1	1	1	0
01	1	1	1	1
11	X	X	X	X
10	1	1	X	X

$a = \bar{B}\bar{D} + BD + CD + A$

$b = \bar{B} + \bar{C}\bar{D} + CD$

$c = B + \bar{C} + D = \overline{\bar{B}\,C\,\bar{D}}$

$d = \bar{B}\bar{D} + C\bar{D} + \bar{B}C + B\bar{C}D$

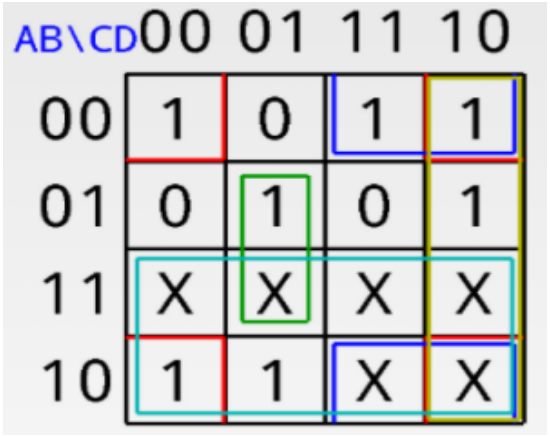
$e = \bar{B}\bar{D} + C\bar{D}$

$f = A + \bar{C}\bar{D} + B\bar{C} + B\bar{D}$

$g = A + B\bar{C} + \bar{B}C + C\bar{D}$

Cont...

- **For d**

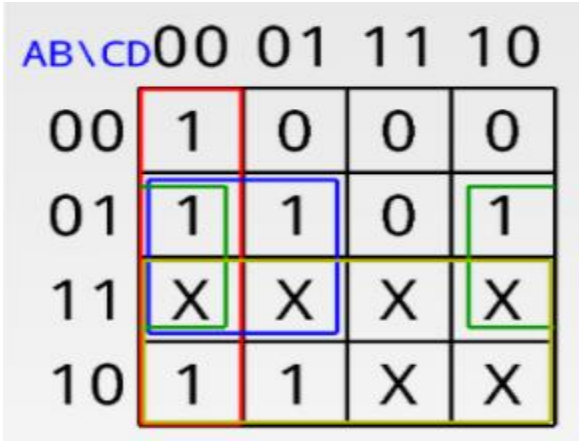


AB\CD	00	01	11	10
00	1	0	1	1
01	0	1	0	1
11	X	X	X	X
10	1	1	X	X

- **For e**

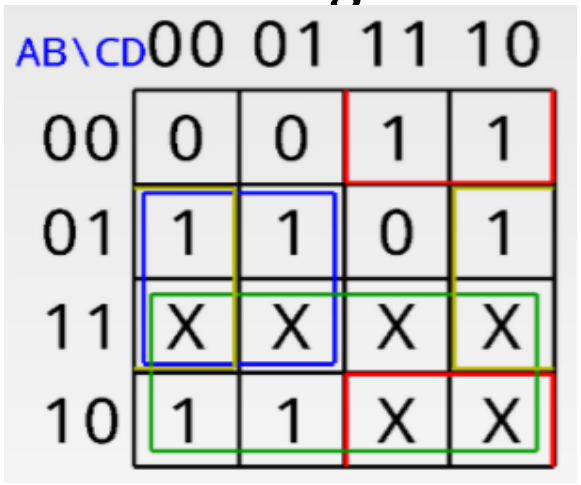
AB\CD	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	X	X	X	X
10	1	0	X	X

- **For f**



AB\CD	00	01	11	10
00	1	0	0	0
01	1	1	0	1
11	X	X	X	X
10	1	1	X	X

- **For g**



AB\CD	00	01	11	10
00	0	0	1	1
01	1	1	0	1
11	X	X	X	X
10	1	1	X	X

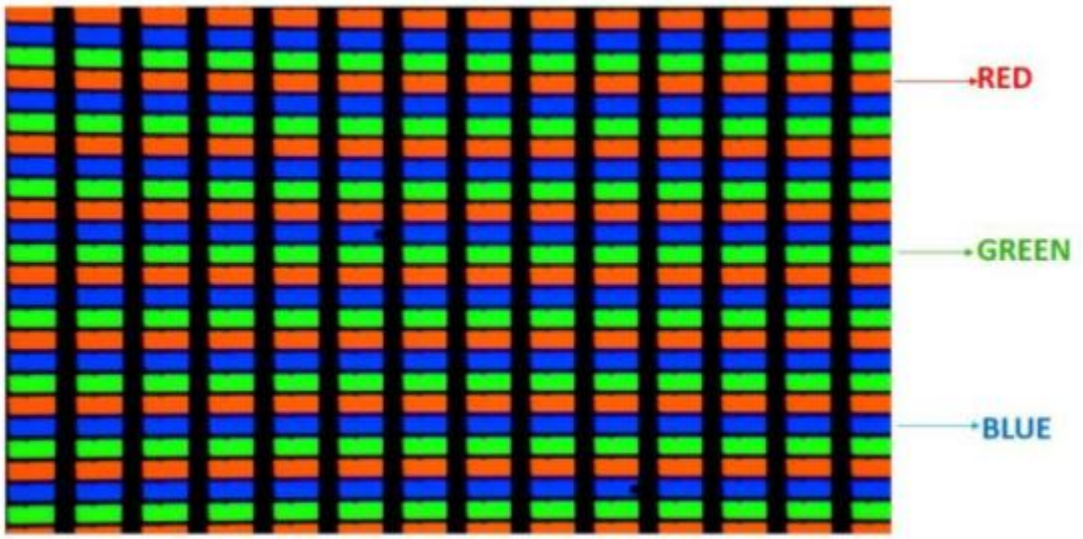
Cont...

- **Applications** –
  Seven-segment displays are used to display the digits in calculators, clocks, various measuring instruments, digital watches and digital counters.

LCD Display

- **LCD** stands for **Liquid Crystal Display**. It is a flat panel display technology, mainly used in TVs and computer monitors, nowadays it is used for mobile phones also. These LCDs are completely different from that old CRT displays, it uses **liquid crystals** instead of cathode ray in its primary form of operation.

- In LCD display, it consists of millions of pixels made of crystal and arranged in a rectangular grid. In LCD it has backlights that provide light to each pixel. Each pixel has a red, green, and blue (RGB) sub-pixel that can be turned on or off. When all of the sub-pixels are turned off, then it’s black and when all the sub-pixels are turned on 100%, then it’s white.



LCD Display

- LCD is a combination of two states of matter, the solid and the liquid. The solid part is crystal and this liquid and crystal together make the visible image. LCD consists of two layers which are two polarized panels- filters and electrodes.

- LCD screen works by blocking the light rather than emitting the light. There are two types of pixel grids in LCD:

- **Active Matrix Grid**– It is a newer technology. In smartphone with LCD display uses this technology.
- **Passive Matrix Grid**– It is an older technology. Some older devices used this technology.

LCD Characteristics

- **Main characteristics of LCD are:**

- **Voltage:** 3V to 12V
- **Operating temperature:** Normally it ranges from 0^0C to $+60^0C$, but for extreme cases it varies from -40^0C to $+85^0C$.
- **Frequency:** 30Hz to 60 Hz
- **Average Current consumption:** 1.2μA to 6μA
- **Opening Time:** 100 ms

LCD has other Characteristics

- **Resolution:** LCD is made up of liquid crystal, which is neither liquid nor solid, and this thing reflects the light in a well-formed way, lights enter into the crystal and reflect very clearly. So the image made with this liquid crystal is very accurate. LCD is a Digital display, which addresses each individual pixel using a fixed matrix of horizontal and vertical dots. LCD scales the image according to the resolution the device provided. So the quality of the image is not degraded.

- **Brightness:** Brightness means the light provided by the LCD, which is nothing but the intensity of visible light, it is measured using *nits*. **Nits** is defined as one candela per square meter. In LCD brightness is very much accurate for the good resolutions and pixels.

- **Contrast Ratio:** It is the ratio of the brightest color and the darkest color for a particular position of the screen provided by the display. To calculate contrast ratio(CR) see the below formula.

$$Contrast Ratio(CR) = \frac{brightness of the screen when pixels are white}{brightness of the screen when pixels are white}$$

LCD has other Characteristics

Typically the ratios of modern monitors are 1000:1 and TVs are 4000:1.

- **Response Rate:** Response Rate is high in LCD, it means the time required for changing colours of the pixels is very much less, so that the refresh rate is very high in LCD than CRT. There is no lagging between the pixels when the image is changed.

Advantages of LCD

- The main advantage of LCD is, it has low in cost and energy efficient and very less power consumption.
- LCD is thinner and lighter and very flexible.
- LCD provides excellent contrast, brightness and resolution, so the picture quality is very clear like a crystal.
- Radiation of LCD monitors are much less than CRT monitors
- LCDs can be suitable with CMOS integrated circuits so that making of LCD is very easy.
- It gives perfect sharpness at the native resolution
- Zero geometric distortion at the native resolution of the pane
- It provides various conveniences like portability as compared to previous technology based screens.

Disadvantages of LCD

- LCD require additional light sources for lighting the pixels, so if the light source is destroyed then the LCD is not providing any image on the display.
- LCD is less reliable display.
- The image visibility depends on light intensity
- The aspect ratio and resolution are fixed for LCD.
- LCD has an irregular intensity scale and it produce lower than 256 discrete intensity levels.
- In LCD color saturation is reduced at the low intensity level due to poor black-level.
- LCD provide limited viewing angle, it effects the brightness. if we are watching the screen by an angle then the color of the image is changed in our eyes.

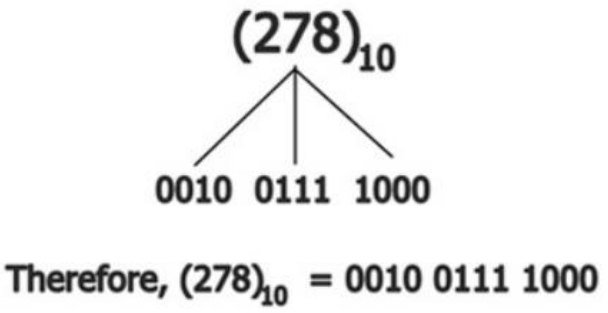
Code Converter

- A code converter is a logic circuit which changes data in one type of binary code to another type of binary code. A conversion circuit must be inserted between two systems if each uses different codes for the same information.

BCD code Excess-3 code

2421 code Gray code

- **Binary Coded Decimal (BCD) Code**
- Binary Coded Decimal (BCD) code is the simplest binary code to represent a decimal number.
- In BCD code, a decimal number is represent by 4-binary bit. If a decimal number consist of two or more than two digit, then each decimal digit is individually represented by 4-bit binary equivalent.
- For example, $(278)_{10}$ in BCD is represented as given in the following photo:
- Therefore we can say that the number 278 in binary is represented as 0010 0111 1000 or **$(278)_{10}$ = 0010 0111 1000**.



Excess-3 Code and 2421 Code

- BCD code is a weighed code, that is the weight of four binary bits which represent an individual digits are 8, 4, 2, 1 modern computer perform subtraction using complements and there is a difficulty in forming complements when number are represented by BCD codes.

- **Excess-3 Code**
- In Excess-3 code, 3 is added to the individual digit of a decimal number then these binary equivalent are written. For example, $(278)_{10}$ in excess-3 code is represented by 0101101011. This code is not a weighted code.

- **2421 Code**
- 2421 code is another BCD code. It is weighed code. For example, 6 is 1100 and 3 is 0011.

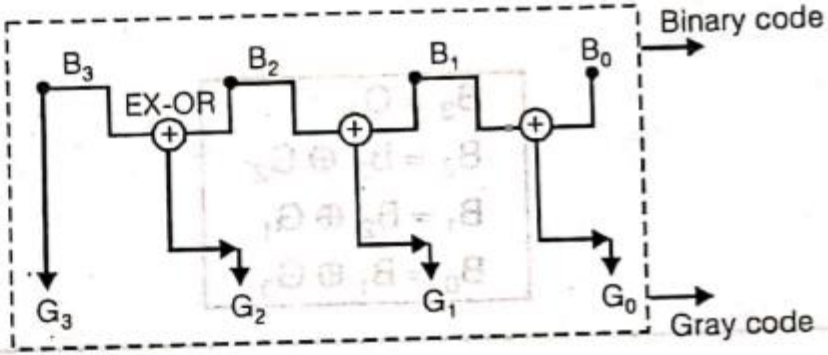
GRAY Code

- The gray code is a binary code. The binary bits are arranged in such a way that only one binary bit changes at a time when we make a change from any number to the next.
- Gray code is reflected code. This code is used in shaft encodes which indicates the angular position of a shaft in digital form. Shaft position encoder disks are used as sensors. It is not a weighted code. Gray code can be constructed using the following properties:
- A 1 bit gray code has two code words 0 and 1 to represent decimal number 0 and 1 respectively.
- A n bit gray code will have first 2^{n-1} gray codes for n-1 bits written in order with a leading 0 appended.
- The last 2^{n-1} gray code will be equal to the gray code words of an (n-1) bits gray code, written in reverse order (assuming a mirror placed between first 2^{n-1} and last 2^{n-1} gray codes) with a leading 1 appended.

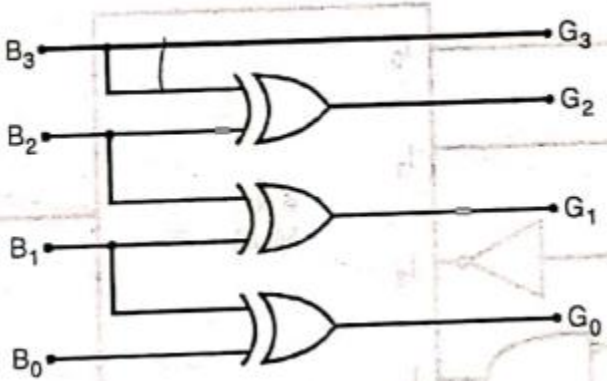
Binary to Gray Converter

Binary-to-Gray code converter

The basic conversion concept is shown below:



Equivalent logical gate diagram

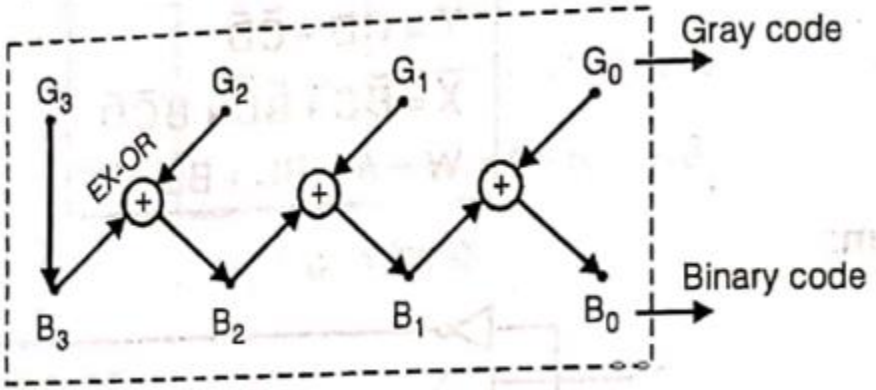


$$G_3 = B_3$$
$$G_2 = B_3 \oplus B_2$$
$$G_1 = B_2 \oplus B_1$$
$$G_0 = B_1 \oplus B_0$$

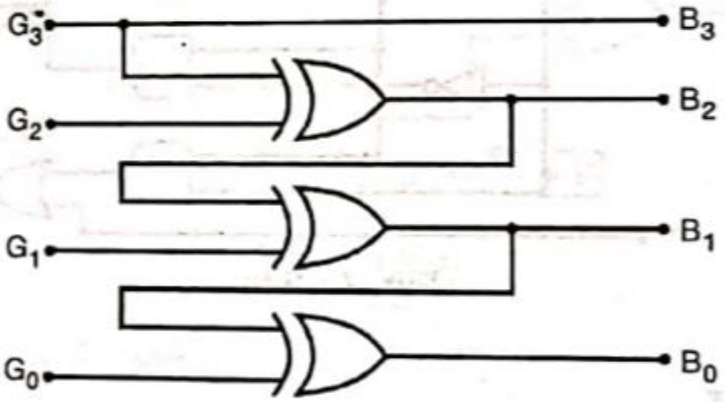
Gray to Binary Converter

Gray to Binary Converter

The basic conversion concept is shown below:



Equivalent logical gate diagram



$$B_3 = G_3$$
$$B_2 = B_3 \oplus G_2$$
$$B_1 = B_2 \oplus G_1$$
$$B_0 = B_1 \oplus G_0$$

BCD to Excess 3

BCD to Excess-3 code

Let input variables of BCD code is A, B, C, D and output variables of excess-3 is W, X, Y, Z then required truth table (BCD + 0011 = excess-3-code) is given below:

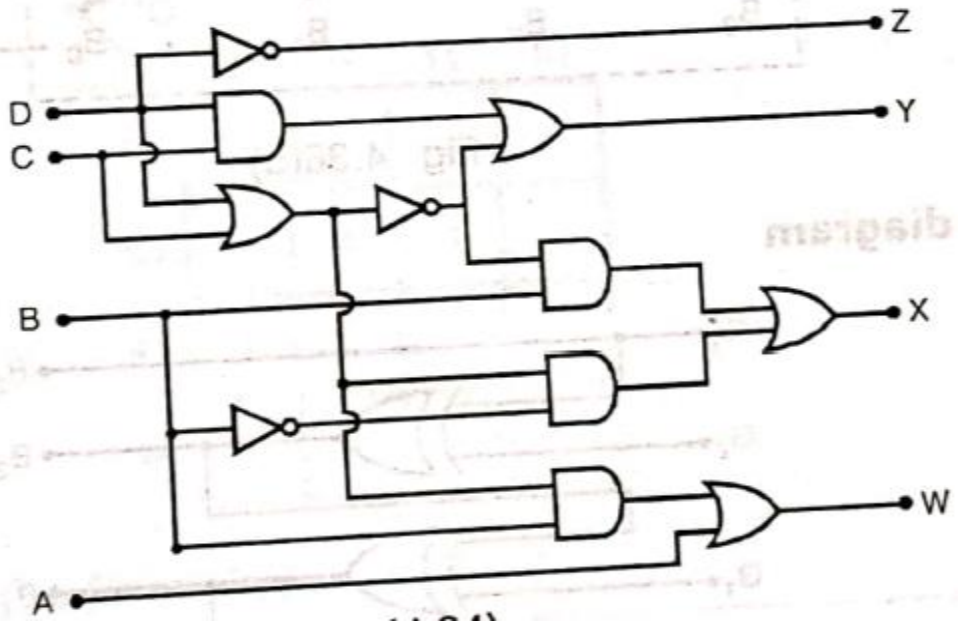
Inputs (BCD)				Output (Excess-3 code)			
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

BCD to Excess 3

From the above truth table, the minimised Boolean function is

$$Z = \bar{D}$$
$$Y = CD + \bar{C}\bar{D}$$
$$X = \bar{B}C + \bar{B}D + B\bar{C}\bar{D}$$
$$W = A + BC + BD$$

Logic circuit diagram:



Code Table

- The table given below shows an example of all the four type of code converted into each other.

Decimal No.	BCD (8421)	Excess-3 Code	2421 Code	Gray Code
0	0000	0011	0000	0000
1	0001	0100	0001	0001
2	0010	0101	0010	0011
3	0011	0110	0011	0010
4	0100	0111	0100	0110
5	0101	1000	0101	0111
6	0110	1001	1100	0101
7	0111	1010	1101	0100
8	1000	1011	1110	1100
9	1001	1100	1111	1101
10	0001 0000			1111
11	0001 0001			1110
12	0001 0010			1010
13	0001 0011			1011
14	0001 0100			1001
15	0001 0101			1000

Comparators

Magnitude Comparators

It is a combinational logic circuit that compares two input binary quantities and generates outputs to indicate which one has the greater magnitude. Consider an **"one-bit comparator"** whose inputs are A and B and the truth table is given as:

Inputs		Outputs		
A	B	A < B	A = B	A > B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

From truth table have,

⇒ $A < B$; output $= \bar{A}B$

⇒ $A = B$; output $= \bar{A}\bar{B} + AB = A \odot B$

⇒ $A > B$; output $= A\bar{B}$

Comparators

IEEE/ANSI diagram

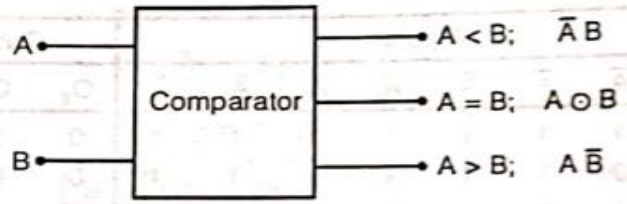


Fig. 4.33(a)

Equivalent logic gate diagram is;

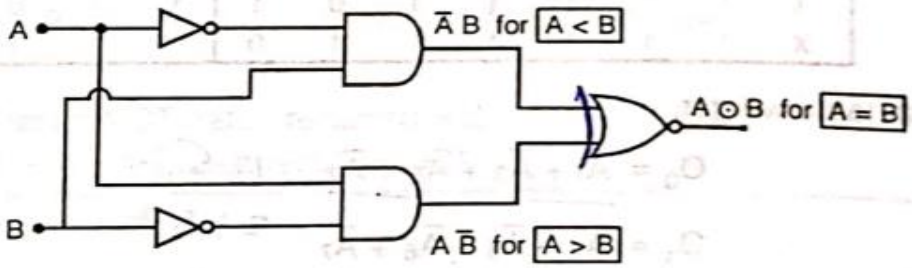


Fig. 4.33(b)

Applications of Magnitude Comparators

These are often used as part of the address decoding circuitry used in a computer to select a specific "input-output device" or area of memory for the storage or retrieval of data. It is also useful in control applications where a binary number representing the physical variable being controlled (e.g. position, speed) is compared with a reference value.