Internal Assessment Test - 1 Nov '25

CMR

DSDV

Date :- 4/11/25.

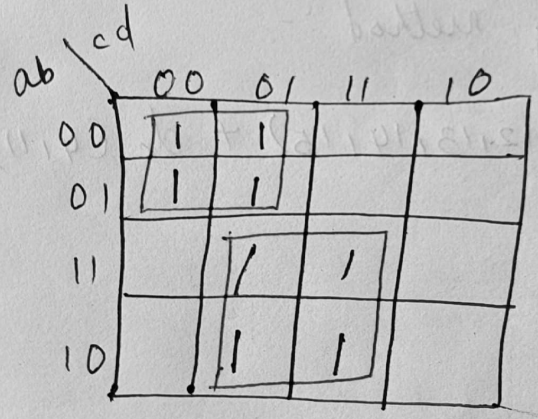
1. Obtain the Minimal expression using K-Map for the following function.

i). $f(a,b,c,d) = \Sigma_m(0,1,4,5,9,11,13,15)$

ii) " $= \Pi_M(0,2,5,7,8,10,13,15)$.

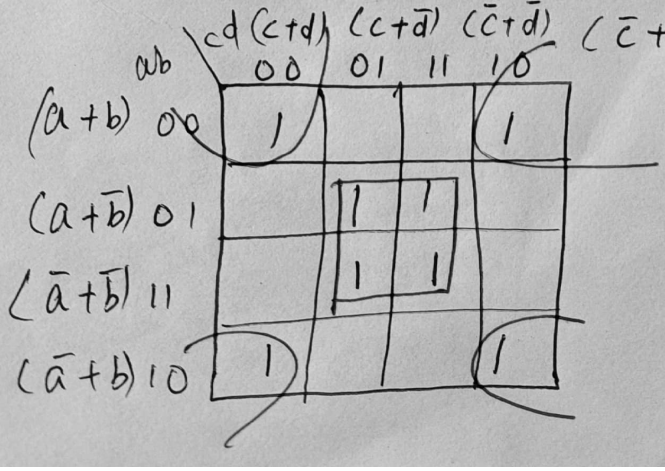
iii) " $= \Sigma_m(0,1,4,6,7,9,15) + \Sigma_d(3,5,11,13)$.

i). $f(a,b,c,d) = \Sigma_m(0,1,4,5,9,11,13,15)$.



$f(a,b,c,d) = a'c' + ad$ //

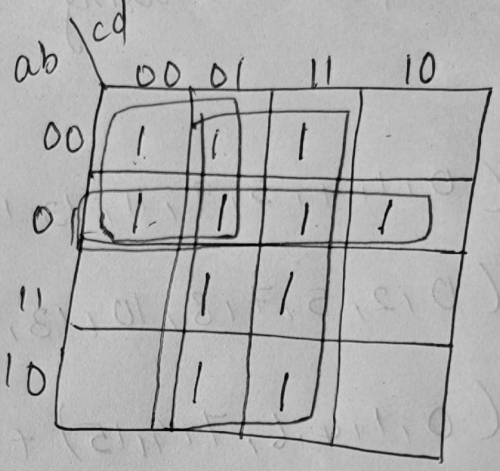
ii) $f(a,b,c,d) = \Pi M(0,2,5,7,8,10,13,15)$.



$f(a,b,c,d) = (\bar{b}+\bar{d}).(b+d)$

//.

iii) $\Sigma m(0,1,4,6,7,9,11,13) + \Sigma d(3,5,11,13)$.



$f(a,b,c,d) = d + a'c' + a'b$ //.

2). Find the minimal sum for the following Boolean fn using Quine - Mc Clusky method :-

$f(a,b,c,d) = \Sigma m(7,9,12,13,14,15) + d_c(4,11)$

Soln :-

7 → 0111
9 → 1001 -
12 → 1100 -
13 → 1101 -
14 → 1110 -
15 → 1111 -

dc

4 → 0100 *
11 → 1011 * -

Step: 1

Group	Minterms	Binary Variable
1	m_4	0100 * ✓
2	m_9	1001 ✓
	m_{12}	1100 ✓
3	m_7	0111 ✓
	m_{11}	1011 * ✓
	m_{13}	1101 ✓
	m_{14}	1110
4	m_{15}	1111 ✓

Step: 2

Group	Matched Pairs	Binary Variable
1	$m_4^* - m_{12}$	-100
2	$m_9 - m_{11}^*$	10-1 ✓
	$m_9 - m_{13}$	1-01 ✓
	$m_{12} - m_{13}$	110- ✓
	$m_{12} - m_{14}$	11-0 ✓
3	$m_7 - m_{15}$	-111
	$m_{11}^* - m_{15}$	1-11 ✓
	$m_{13} - m_{15}$	11-1 ✓
	$m_{14} - m_{15}$	111- ✓

Step: 3

Group	Matched Pairs	Binary Variable a b c d
2	$m_9 - m_{11}^* - m_{13} - m_{15}$	1 - - 1
	$m_9 - m_{13}^* - m_{13} - m_{15}$	1 - - 1
	$m_{12} - m_{13} - m_{14} - m_{15}$	1 1 - -
	$m_{12} - m_{13} - m_{14} - m_{15}$	1 1 - -

a

All the Minterm's are n
covered, ex, m_4^* & m_7 a
Covered, Since m_4^* is do
care term ignore that,
Consider only $m_7 - m_{15}$ grou
for final solution.

Step: 4

$f(a,b,c,d) = ab + ad + bcd$

3. Comparators :-

A simple approach, to compare the magnitudes of two binary numbers for the purpose of establishing whether one is greater than, equal to or less than the other, such a n/w is called a **comparator**.

- A comparator makes use of a cascade connection of identical subnetworks, as like parallel adder.

- Consider two n-bit binary numbers,

$$A = A_{n-1} \text{ ---- } A_i A_{i-1} \text{ ---- } A_1 A_0 +$$

$$B = B_{n-1} \text{ ---- } B_i B_{i-1} \text{ ---- } B_1 B_0 .$$

- For this design, assume that only one bit of corresponding order from each number is entering the subnetwork, say A_i & B_i of the two binary numbers are analyzed from right to left. This subnetwork is called <u>1-bit comparator</u>

- The function of the 1-bit comparator is to establish whether $A_i A_{i-1} \cdots A_1 A_0$ is greater than, equal to or less than $B_i B_{i-1} \cdots B_1 B_0$ given the values of A_i, B_i, & whether $A_{i-1} \cdots A_1 A_0$ is greater than, equal to or less than $B_{i-1} \cdots B_1 B_0$

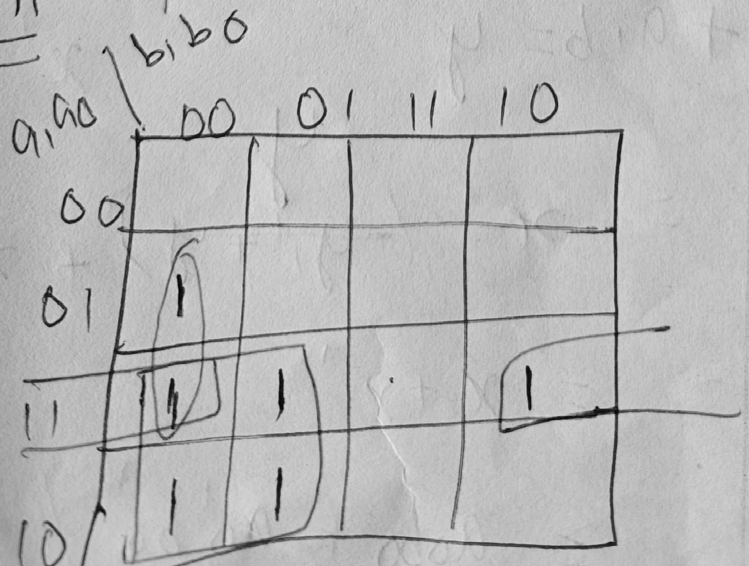
- The three conditions describing the relative magnitudes of $A_{i-1} \cdots A_1 A_0$ and $B_{i-1} \cdots B_1 B_0$ are assigned to three variable G_i, E_i and L_i where $G_i = 1$ denotes $A_{i-1} \cdots A_1 A_0 > B_{i-1} \cdots B_1 B_0$, $E_i = 1$ denotes $A_{i-1} \cdots A_1 A_0 = B_{i-1} \cdots B_1 B_0$, and $L_i = 1$ denotes $A_{i-1} \cdots A_1 A_0 < B_{i-1} \cdots B_1 B_0$

- Thus, the 1-bit comparator is a 5-input, 3-output network.

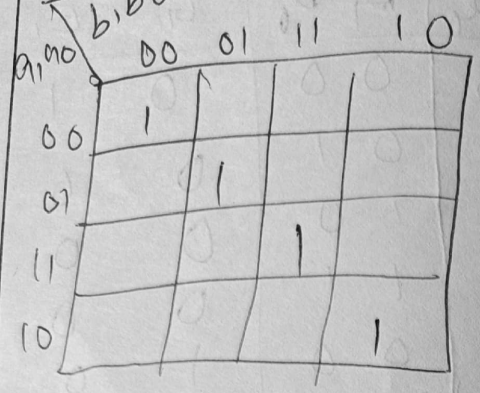
2-bit Comparator

a_1	a_0	b_1	b_0	G_i	E_i	L_i
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

G_i



E_i



$E_i = a_1'a_0'b_1'b_0' + a_1'a_0b_1'b_0 + a_1a_0b_1b_0 + a_1a_0'b_1b_0'$

$= a_0'b_0'(a_1'b_1' + a_1b_1) + a_0b_0(a_1'b_1' + a_1b_1)$

$a_0b_0 = x$

$a_1'b_1' + a_1b_1 = y$

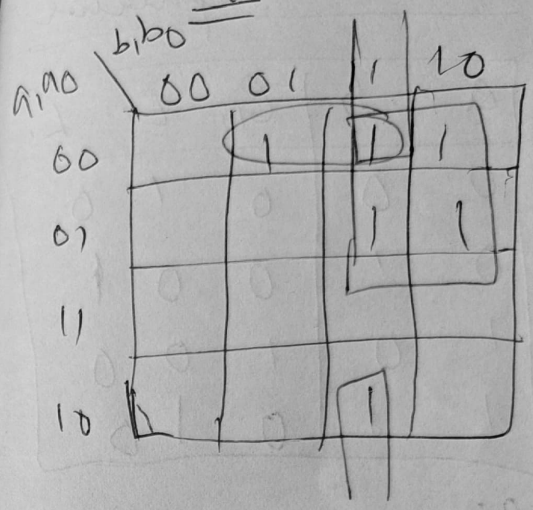
$= x'(\overline{a_1 \oplus b_1}) \quad y$

$= x'(\cancel{-y_1'} + y) + x(\cancel{-y'} + y)$

$= x + x'$

$= a_0b_0 + a_0'b_0'$ //

L_i



$L_i = a_1'b_1 + a_1a_0'b_0 + a_0'b_1b_0$ //.

→ Work out for 3-bit Comparator.

b_n d_{n-1}

Parallel binary Subtractor

4-bit Parallel Adder/Subtractor:-

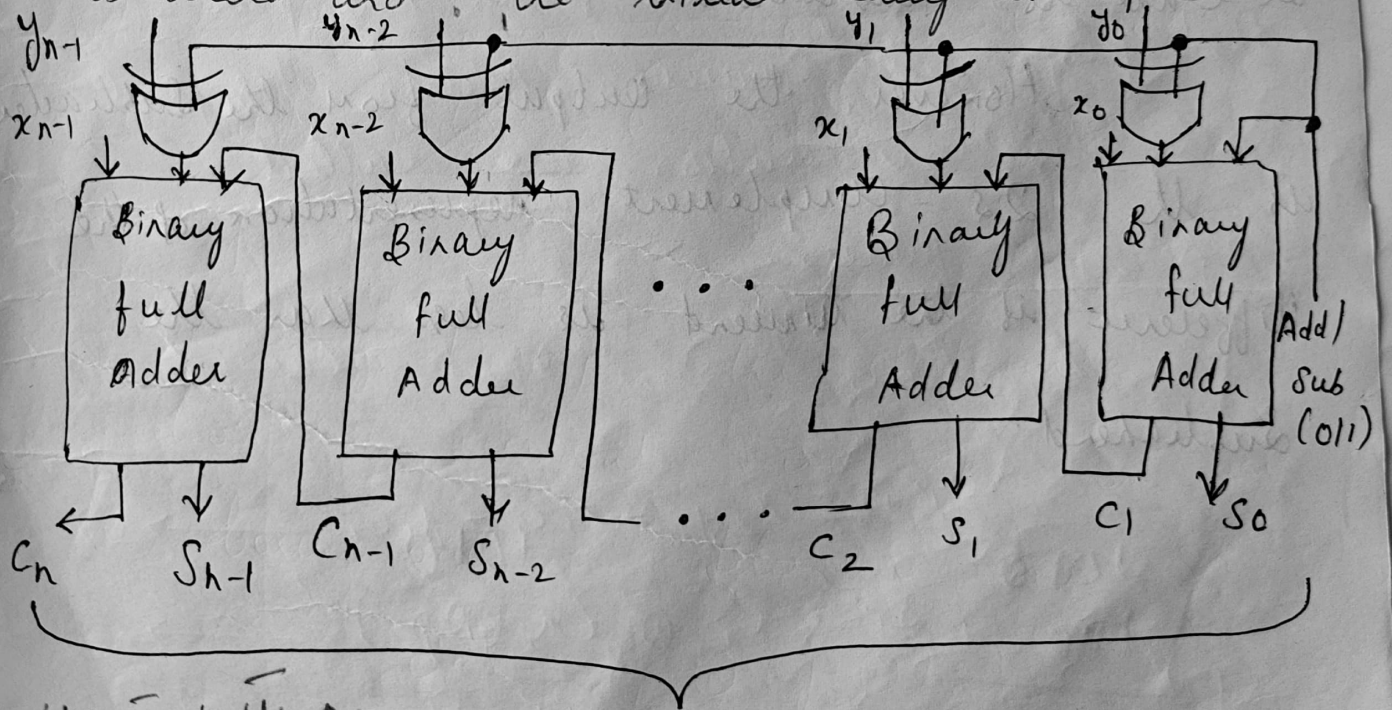
4) The structure of such realization is shown in figure, where $x_{n-1}, x_{n-2} \ldots x_1 x_0$ is the n-bit minuend and $y_{n-1}, y_{n-2} \ldots y_1 y_0$ is the n-bit Subtrahend.

Subtraction can be replaced by addition through the use of complements. For example, adding the 2's complement of the Subtrahend to the minuend results in the difference b/w two numbers.

Furthermore, the 2's complement of a binary number is obtained by adding

One to its 1's complement. The 1's complement of the subtrahend is formed by inverting each of its bits, and a carry-in of 1 in the least significant-bit position provides for the addition of 1 to the 1's complement.

$y_i \oplus 1 = \bar{y}_i$, gives a realization of a parallel adder/subtractor. The behavior of this network is determined by the control signal Add/Sub. The subtraction operation is obtained by letting Add/Sub = 1, in which case 1's are appropriately entered into the exclusive OR-gates to provide the 1's complement of the subtrahend and the initial carry-in of 1.



$y_i \cdot \bar{0} + \bar{y}_i \cdot 0$ — Sum or difference

Fig :- Parallel binary adder/subtractor

The parallel binary adder then produces the difference.

When the two operands are to be added, Add/Sub = 0. The bits of the addend are not modified by the exclusive-OR-gates prior to entering the parallel binary adder and the necessary initial carry-in of 0 is provided.

The operands in subtraction can be either signed or unsigned; the output of a binary subtracter must be interpreted appropriately.

For example, for unsigned operands, the output from the binary subtracter is the true difference if the minuend is greater than or equal to subtrahend.

However, the output from the subtracter is the 2's-complement representation of the difference if the minuend is less than the subtrahend.

5. Binary Subtracters :-

A binary subtracter can be designed using the same approach as that of binary adder. Again here, three bits are involved at each bit order, a Minuend bit x_i, a Subtrahend bit y_i and borrow-in bit from the previous bit-order position b_i. The result of the subtraction is a difference bit d_i, and a borrow-out bit b_{i+1}.

Truth Table for a binary full Subtracter.

x_i	y_i	b_i	b_{i+1}	d_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

A	B	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

The difference bit at each order is obtained by subtracting both the subtrahend and borrow-in bits from the minuend bit. To achieve this, however a borrow-out from the next higher-order bit position may be necessary.

By using k-Map, the difference equation & minimal-Sum expression for the borrow-out is determined as,

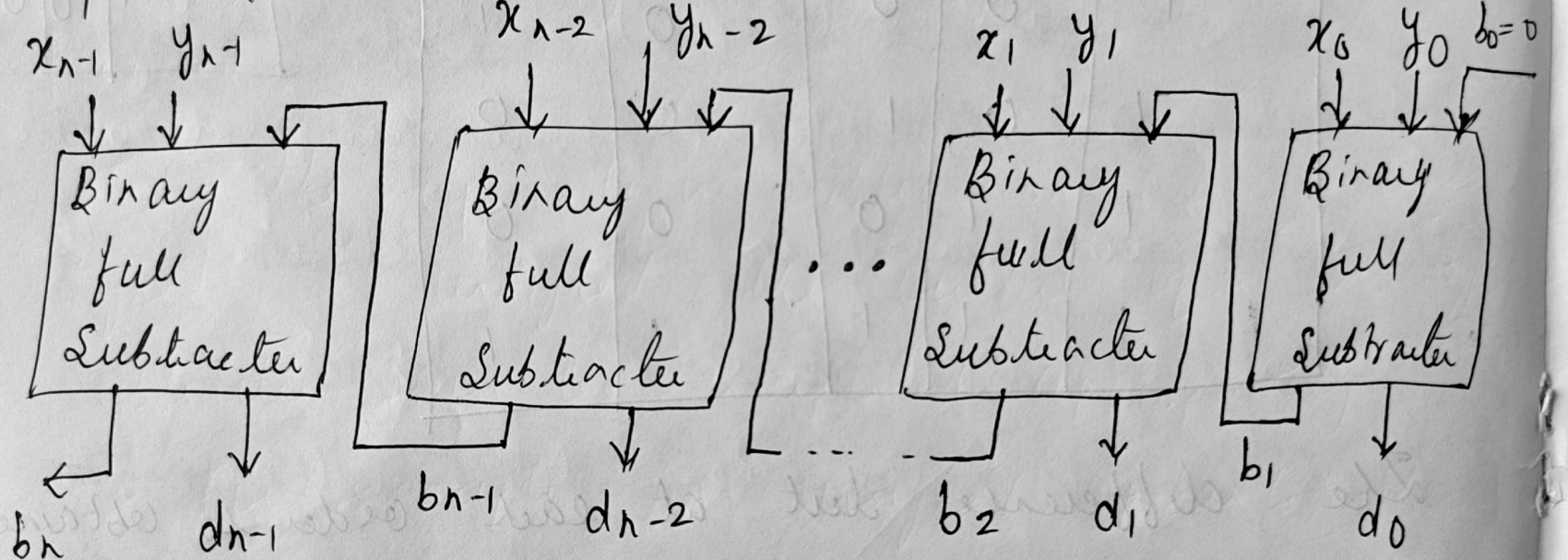
$$d_i = \overset{A}{x_i} \oplus (\overset{B}{y_i} \oplus \overset{C}{b_i})$$

$$b_{i+1} = \bar{x}_i y_i + \bar{x}_i b_i + y_i b_i$$

$$\bar{A}B + \bar{A}C + BC$$

These results can be used to construct a logic diagram for a binary full Subtractor.

As like binary addition, by cascading n binary full Subtractors, a ripple binary Subtractor is realized for handling two n-bit Operands.



Parallel binary Subtractor

6. Carry Lookahead Adder :- / High Speed Adders

All the networks are subject to ripple effect, as the operands are applied in parallel. The ripple effect dictates the overall speed at which the network operates.

By considering the figure from binary, it is possible that a carry is generated in the least-significant-bit-position stage & owing to the operands, this carry must propagate through all the remaining stages to the highest-order-bit position stage.

For example, such a situation occurs when the two n-bit operands are 01...11 and 00..01, leads to a n-bit sum 10...00 is produced.

Two levels of logic are needed to generate the carry at the least-significant-bit position stage, two levels of logic are needed to propagate the carry through each of the next n-2 higher order stages, and two levels of logic are needed to form the sum or carry at highest order-bit position stage.

If each stage is assumed to introduce a unit time of propagation delay, then the Maximum propagation delay of the ripple adder becomes $2n$ units of time.

The disadvantage is that, all signals must complete their propagations through a network, before new inputs are applied. This becomes a limiting factor in the n/w's overall speed of operation.

To decrease the time required to perform addition, an effort must be made to speed up the propagation of the carries. One approach, is to reduce the number of logic levels in the path of the propagated carries. Adders designed with this consideration are called

<u>high-speed Adders</u>.

Equations sum and carry,

$$C_{i+1} = x_i y_i + x_i c_i + y_i c_i \longrightarrow (1)$$

$$S_i = c_i \oplus (x_i \oplus y_i) \longrightarrow (2)$$

are the outputs at the i^{th} stage of a binary adder.

The sum and carry outputs at a given stage are a function of the output carry from the previous stage, which in turn is a function of the output carry from another previous stage. This corresponds to undesirable ripple effect.

If the input carry at a given stage is expressed in terms of the operands variables ie, $x_0, x_1, \ldots x_{n-1}$ and $y_0, y_1, \ldots y_{n-1}$, then the ripple effect is eliminated & the overall speed of the adder is increased.

Let,

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

$$= x_i y_i + (x_i + y_i) c_i$$

The first term in the last equation, $x_i y_i$ is called carry-generate function, since it corresponds to the formation of a carry at the ith stage.

The second term, $(x_i + y_i) c_i$ corresponds to a previously generated carry c_i, that must propagate past the ith stage to the next stage.

The $x_i + y_i$ part of this term is called carry-propagate function.

Letting the carry-generate function be denoted by Boolean variable "g_i" and the carry propagate function by "P_i", i.e,

$$g_i = x_i \cdot y_i \longrightarrow (3)$$

$$P_i = x_i + y_i \longrightarrow (4)$$

The output carry equation for the i^{th} stage is given by

$$C_{i+1} = g_i + P_i C_i \longrightarrow (5)$$

The output carry at each of the stages can be written in terms of carry-generate functions, carry-propagate functions and the initial input carry C_0 as follows,

$$C_1 = g_0 + P_0 C_0 \longrightarrow (6)$$

$$C_2 = g_1 + P_1 C_1$$

$$= g_1 + P_1 (g_0 + P_0 C_0)$$

$$= g_1 + P_1 g_0 + P_0 P_1 C_0 \longrightarrow (7)$$

$$C_3 = g_2 + P_2 C_2$$

$$= g_2 + P_2 (g_1 + P_1 g_0 + P_1 P_0 C_0)$$

$$= g_2 + P_2 g_1 + P_2 P_1 g_0 + P_2 P_1 P_0 C_0 \longrightarrow (8)$$

$$C_4 = g_3 + P_3 C_3$$

$$= g_3 + P_3 (g_2 + P_2 g_1 + P_2 P_1 g_0 + P_2 P_1 P_0 C_0)$$

$$= g_3 + P_3 g_2 + P_3 P_2 g_1 + P_3 P_2 P_1 g_0 + P_3 P_2 P_1 P_0 C_0$$

$$C_{i+1} = g_i + P_i g_{i-1} + P_i P_{i-1} g_{i-2} + \cdots + P_i P_{i-1} \cdots P_1 g_0 + P_i P_{i-1} \cdots P_0 C_0 \longrightarrow (9)$$

Since each carry-generate function and carry-propagate function is itself only a function of the operand variables as indicated by eqn (3) & (4), the output carry and the input carry, at each stage can be expressed as a function of the operand variables & the initial input carry "C_0".

Since the output sum bit at any stage is also a function of the previous stage output carry as indicated by equation (2).

Thus, the parallel adders whose realizations are based on the above carry lookahead adders.

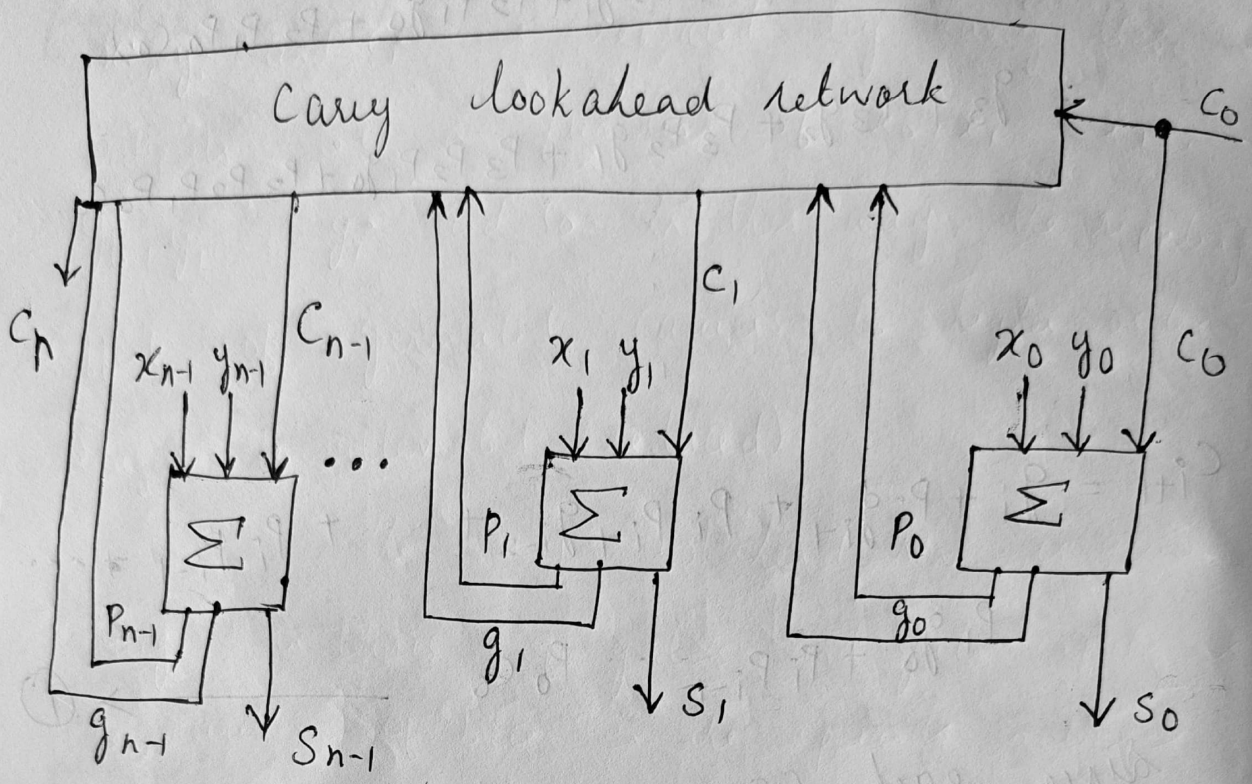


fig:- General Organization of carry lookahead Adder

The general organization of a carry lookahead adder is shown in figure, it corresponds to a logic network based on equation 6 to 9.

The Sigma block corresponds to the logic needed to form the Sum bit, the carry-generate function and the carry-propagate function at each stage.

Generalizing from this figure, the path length from the generation of a carry to its appearance as an input at any higher-order stage, i.e the path length through any stage of the carry lookahead n/w, is two levels of logic.

Thus, with one level of logic to form gi, two levels of logic for the carry to propagate b/w any two stages, and one level of logic to have the carry effect a sum output, the maximum propagation delay for a carry lookahead adder is 4 units of time, under the assumption that each gate introduces a unit time of propagation delay.

140

USN										

CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
CELEBRATING 25 YEARS
ACCREDITED WITH A++ GRADE BY NAAC

Internal Assessment Test 1 – November. 2025

Sub:	Digital System Design using Verilog							Code:	BEC302
Date:	04/ 11 / 2025	Duration:	90 mins	Max Marks:	50	Sem/Section	III: A,B,C,D	Branch:	ECE

	Answer any 5 full questions	Marks	CO	RBT
1	Obtain minimal expression using k-map for the following function i) $f(a,b,c,d) = \Sigma m(0,1,4,5,9,11,13,15)$ ii) $f(a,b,c,d)=\Pi M(0,2,5,7,8,10,13,15)$ iii) $f(a,b,c,d)=\Sigma m(0,1,4,6,7,9,15)+\Sigma d(3,5,11,13)$	10	1	2
2	Find minimal sum for following Boolean function using Quine-McClusky method: $f(a,b,c,d)=\Sigma m(7,9,12,13,14,15)+dc(4,11)$	10	1	1
3	Design a 2-bit magnitude comparator to compare two binary numbers. Follow the design procedure.	10	2	2

140

USN

CELEBRATING 25 YEARS CMRIT

4	Explain in detail 4 bit parallel adder/subtractor with proper example for each.	10	2	2
5	With Truth table, logic diagram explain the binary Full Subtractor.	10	2	2
6	Justify why Carry Lookahead Adder is needed with neat block diagram and expressions.	10	2	2
7	i)Explain the working of a 3-to-**8 decoder** with symbol and truth table. ii)Explain the working of a **8-to-3 encoder** with symbol and truth table.	10	2	2

CI

3/11/25

CCI

M. Pappa

HOD