Carry look ahead-Adder

Need For carry look ahead adder:

→ In case of a ripple carry adder, to carry out a sum of a particular bit we need C_{in} which is the carry from previous bit and similarly for the carry-in from previous bit we require its sum, which again require the sum of previous bit.

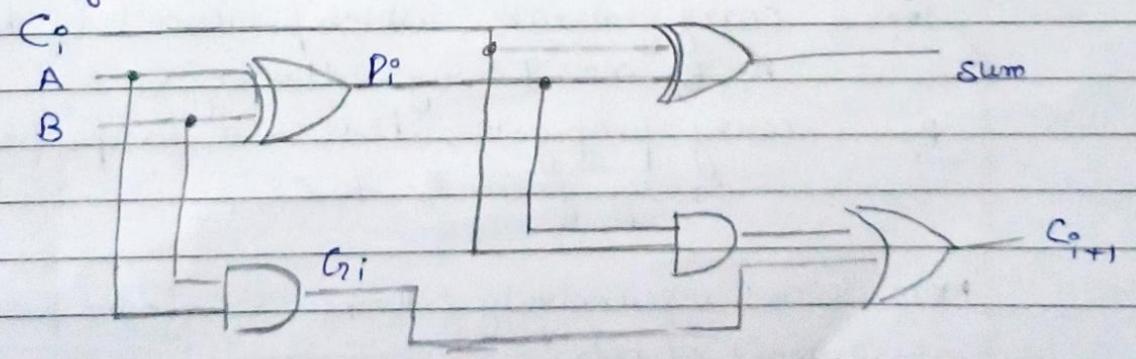
→ So we can't find the sum of any 2 bit until we find the sum of all the previous bits and their carry. So there is a large time delay as all the bits sum is dependent on previous bit.

→ The solution for this problem is:-

a) Either use a very fast gate to carry out the sum of two bit but again it will increase cost and also there is a limit to how much we can reduce time.

b). Or increase the complexity of the circuit to be able to independently find the sum of 2 bit by using some formulas.

Carry look Ahead adder



A	B	C_i	C_{i+1}	Type of carry
0	0	0	0	None
0	0	1	0	None
0	1	0	0	None
0	1	1	1	Propagate.
1	0	0	0	None
1	0	1	1	Propagate
1	1	0	1	Generate
1	1	1	1	Generate / Propagate

We define 2 variable as Carry generate G_i and carry propagate P_i then,

$$P_i = A_i \oplus B_i$$
$$G_i = A_i \cdot B_i$$

The sum and carry output can be expressed as

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

G_i → carry generate, which produces 1 if both A_i, B_i are 1 regardless of C_i

P_i → carry propagate, it is used to propagate carry from C_i to C_{i+1}

Now we can recursively fill the C_i from previous assignment of bits

C_in is carry input
assign

- $C_1 = g_0 \mid (p_0 \& c\_in)$
  $C_2 = g_1 \mid (p_1 \& g_0) \mid (p_1 \& p_0 \& c\_in)$
  $C_3 = g_2 \mid (p_2 \& g_1) \mid (p_2 \& p_1 \& g_0) \mid (p_2 \& p_1 \& p_0 \& c\_in)$
  $C_4 = g_3 \mid (p_3 \& p_2) \mid (p_3 \& p_2 \& g_1) \mid (p_3 \& p_2 \& p_1 \& g_0) \mid (p_3 \& p_2 \& p_1 \& p_0 \& c\_in)$

$C_5 = g_4 \mid (p_4 \& g_3) \mid (p_4 \& p_3 \& g_2) \mid (p_4 \& p_3 \& p_2 \& g_1) \mid (p_4 \& p_3 \& p_2 \& p_1 \& g_0) \mid (p_4 \& p_3 \& p_2 \& p_1 \& p_0 \& c\_in)$

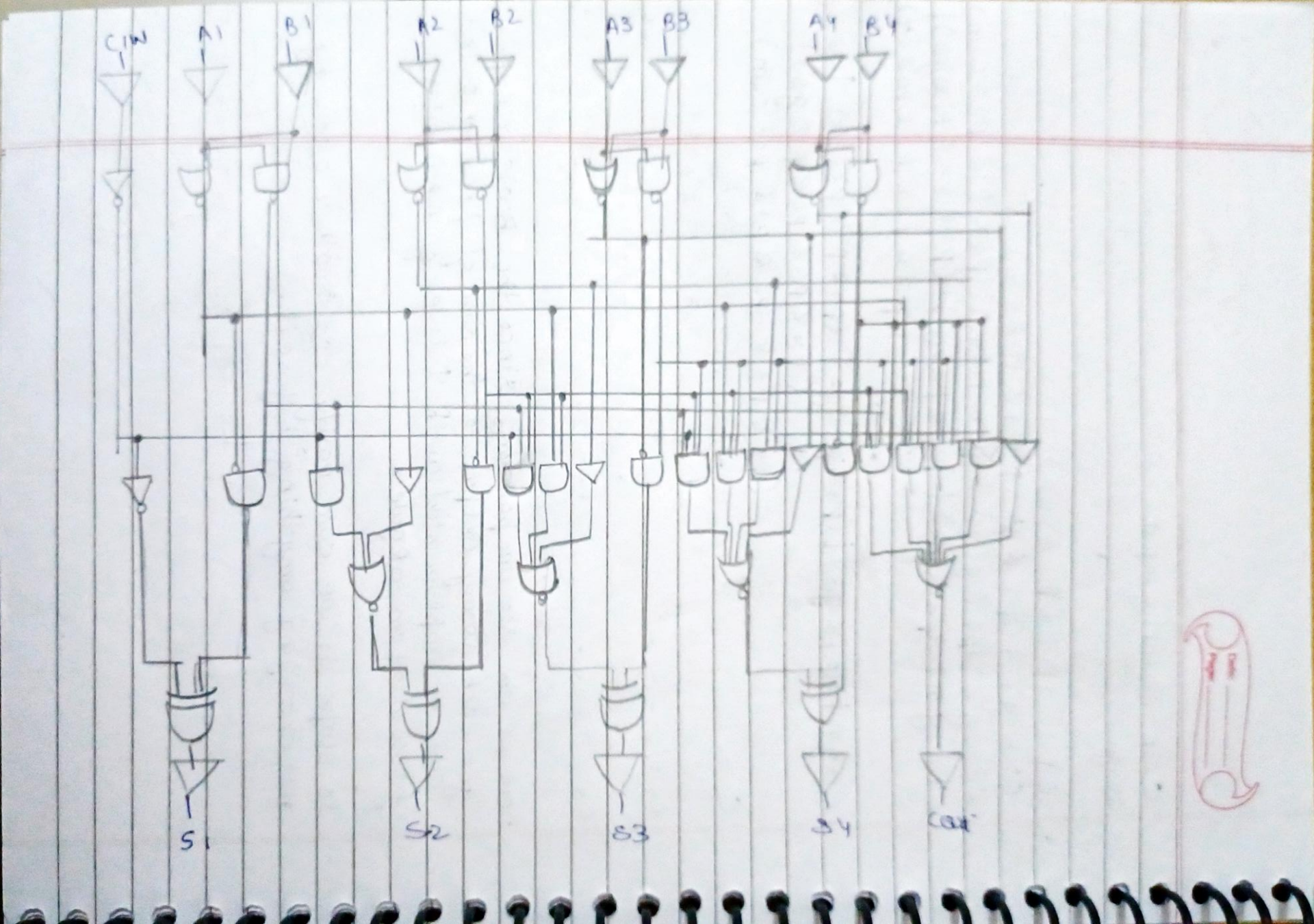
$C_6 = \ldots$

$C_7 = \ldots$

$C_8 = \ldots$

now we can see the pattern, since the Boolean exp. for each carry output as the sum of product so this can be implemented with 1 level of AND gates followed by an OR gate.

→ To scale it we can easily add another adder by not changing anything before it.


CIN
A1
B1
A2
B2
A3
B3
A4
B4
S1
S2
S3
S4
Cout

Explaination

: The 4 bit parelled adder can be implemented with carry look ahead adder. to increase the speed of binary addition. For each sum output 2 XOR gates are required one to generate P_i and the AND gate gives G_i;

: Hence, in 2 gate level we can generate P_i and G_i;
. The carry look ahead generator allowss the P and G signal do propogate after they settle into their steady state values and produces the output carriers at a delay of 2 levels of gates.
∴ we get all the sum at same time and not one after another.