Carry - look ahead adder

This method uses modified carry algorithm to reduce the time needed in ripple carry. This is because it divides carry out into two groups 'generate' and 'propagate' based on A & B ~~totam~~ [illegible]

When

A = B = 0 Carry out = 0 ← No carry generate
A = B = 1 Carry out = 1 ← carry generat
despite carry in value

When

A = 0 , B = 1 or } ← carry propagate
A = 1 , B = 0

carry out may, may not be 1 and depends on carry in.

$$P_i = A_i \wedge B_i$$

$$G_i = A_i \& B_i$$

$$S_i = P_i \wedge C_{i,in}$$

$$C_{i,out} = G_i + P_i C_{i,in}$$

Let $C_1, C_2, C_3 ..$ represent carry out

$$C_1 = G_0 + P_0 C_{in}$$

$$C_2 = G_1 + P_1 C_1$$

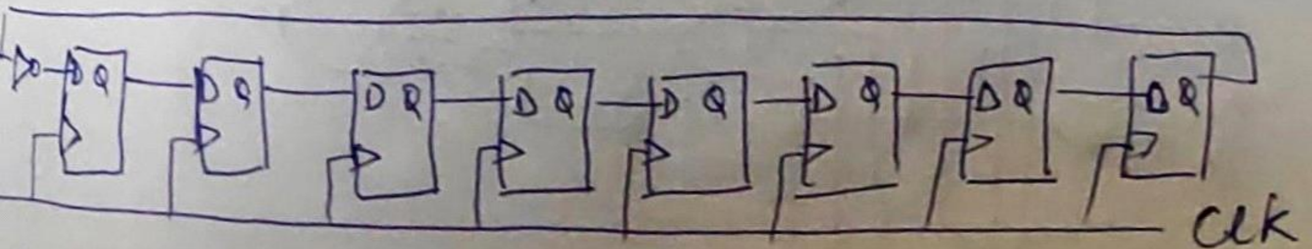
$$C_3 = G_2 + P_2 C_2$$

and so on. Now, C_2 can be written in terms of C_1 and C_1 in terms of C_{in}. So the ~~ripple~~ adder wont have to wait for carry to get calculated

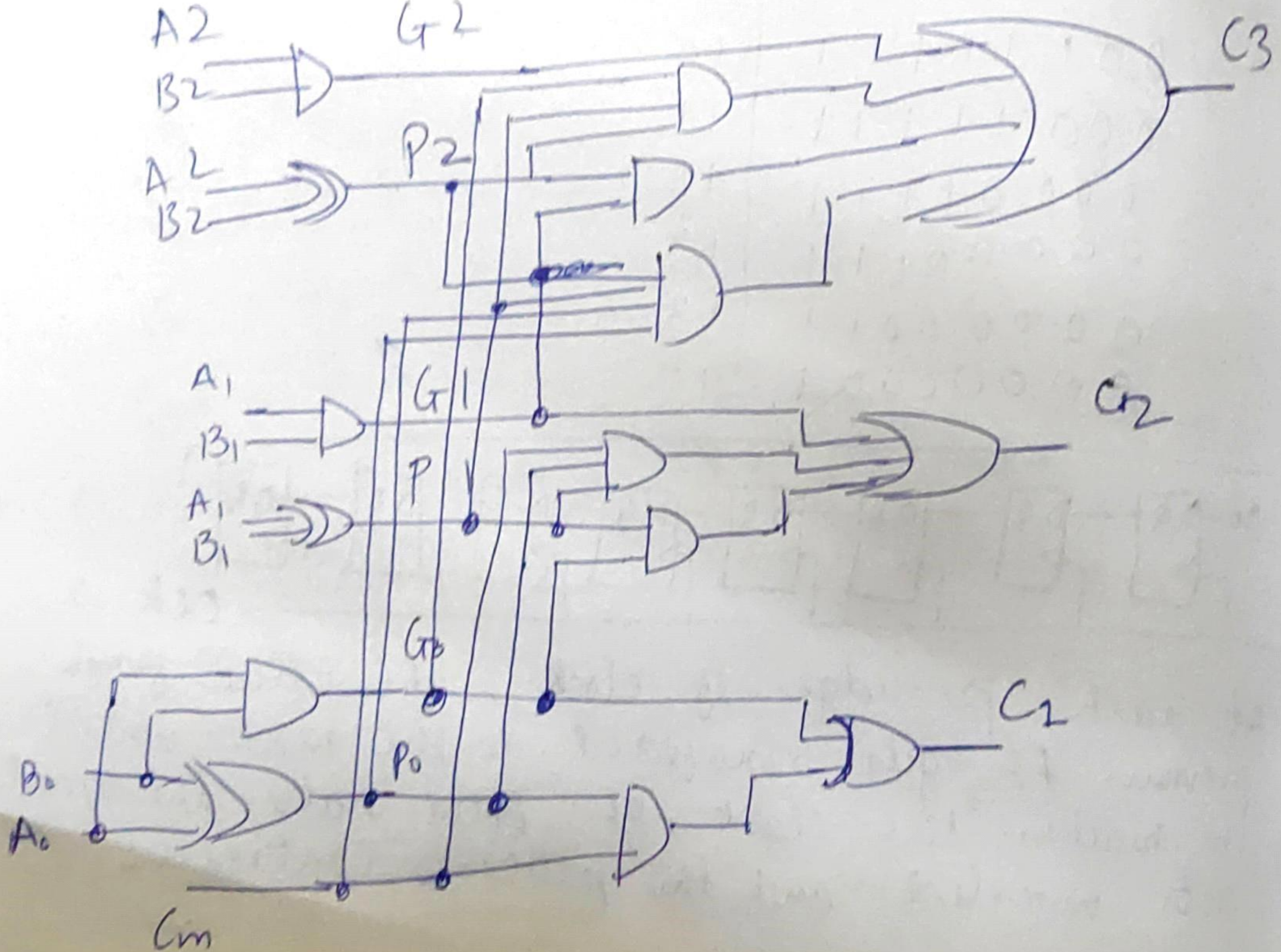
Johnson Counter

In johnson counter, the complemented output of last flipflop is connected to first one. Rest all flip-flops are directly connected

0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1
1	1	0	0	0	0	0	0	2
1	1	1	0	0	0	0	0	3
1	1	1	1	0	0	0	0	4
1	1	1	1	1	0	0	0	5
1	1	1	1	1	1	0	0	6
1	1	1	1	1	1	1	0	7
1	1	1	1	1	1	1	1	8
0	1	1	1	1	1	1	1	9
0	0	1	1	1	1	1	1	10
0	0	0	1	1	1	1	1	11
0	0	0	0	1	1	1	1	12
0	0	0	0	0	1	1	1	13
0	0	0	0	0	0	1	1	14
0	0	0	0	0	0	0	1	15



t each pos-edge of clock, the value from revious FF gets transferred to the next until t reaches the last FF After that, the signal ets inverted and the process continues.


A2
B2
G2
A2
B2
P2
C3
A1
B1
G1
A1
B1
P
C2
G0
B0
A0
P0
C1
Cin