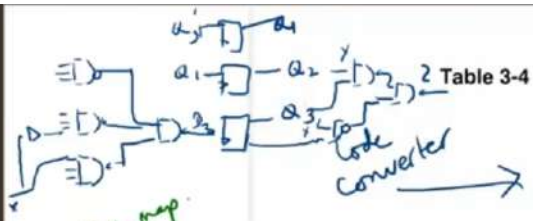
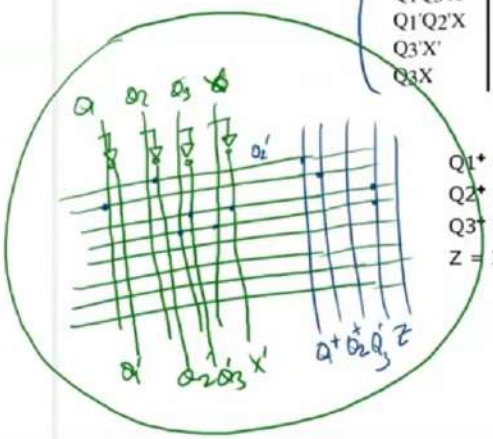


Table 3-4 PLA Table (Based on Figures 1-17 and 1-19)

Product Term	Q1	Q2	Q3	X	$Q1^+$	$Q2^+$	$Q3^+$	Z
Q2'	–	0	–	–	1	0	0	0
Q1	1	–	–	–	0	1	0	0
Q1Q2Q3	1	1	1	–	0	0	1	0
Q1Q3'X'	1	–	0	0	0	0	1	0
Q1'Q2'X	0	0	–	1	0	0	1	0
Q3'X'	–	–	0	0	0	0	0	1
Q3X	–	–	1	1	0	0	0	1



$Q1^+ = Q2'$

$Q2^+ = Q1$

$Q3^+ = Q1Q2\ Q3 + X'Q1Q3' + XQ1'Q2'$

$Z = X'Q3' + XQ3$

PLA

Both And Array

OR Array

programmable

Figure 3-11 PLA Realization of Figure 1-17

```
library ieee;
use ieee.std_logic_1164.all;                    -- IEEE standard logic package
library MVLLIB;                                 -- includes PLAmtrx type and
use MVLLIB.mvl_pack.all;                        -- PLAout function

entity PLA1_2 is
      port(X,CLK: in std_logic;
           Z: out std_logic);
end PLA1_2;

architecture PLA of PLA1_2 is
signal Q, Qplus: std_logic_vector(1 to 3) := "000";
constant FSM_PLA:  PLAmtrx(0 to 6, 7 downto 0) :=
          ("X0XX1000","1XXX0100","111X0010","1X000010",
           "00X10010","XX000001","XX110001");
begin
    process(Q,X)
    variable PLAValue: std_logic_vector(3 downto 0);
    begin
        PLAValue := PLAout(FSM_PLA,Q & X);          -- read PLA output
        Qplus <= PLAValue(3 downto 1);
        Z <= PLAValue(0);
    end process;

    process(CLK)
    begin
        if CLK='1' then Q <= Qplus; end if;         -- update state register
    end process;
end PLA;
```

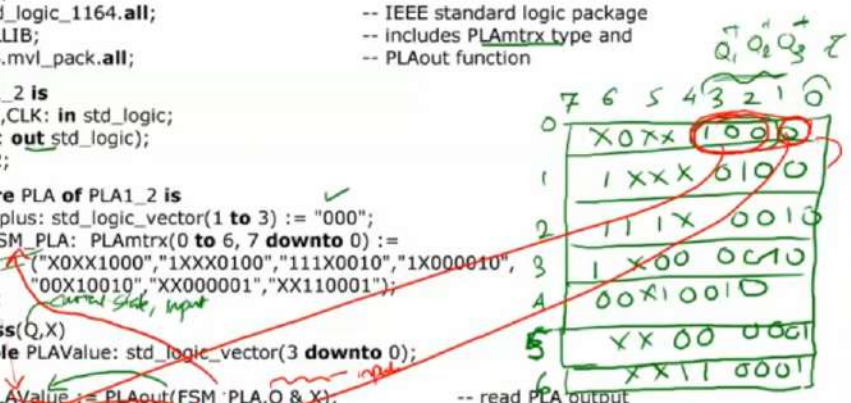
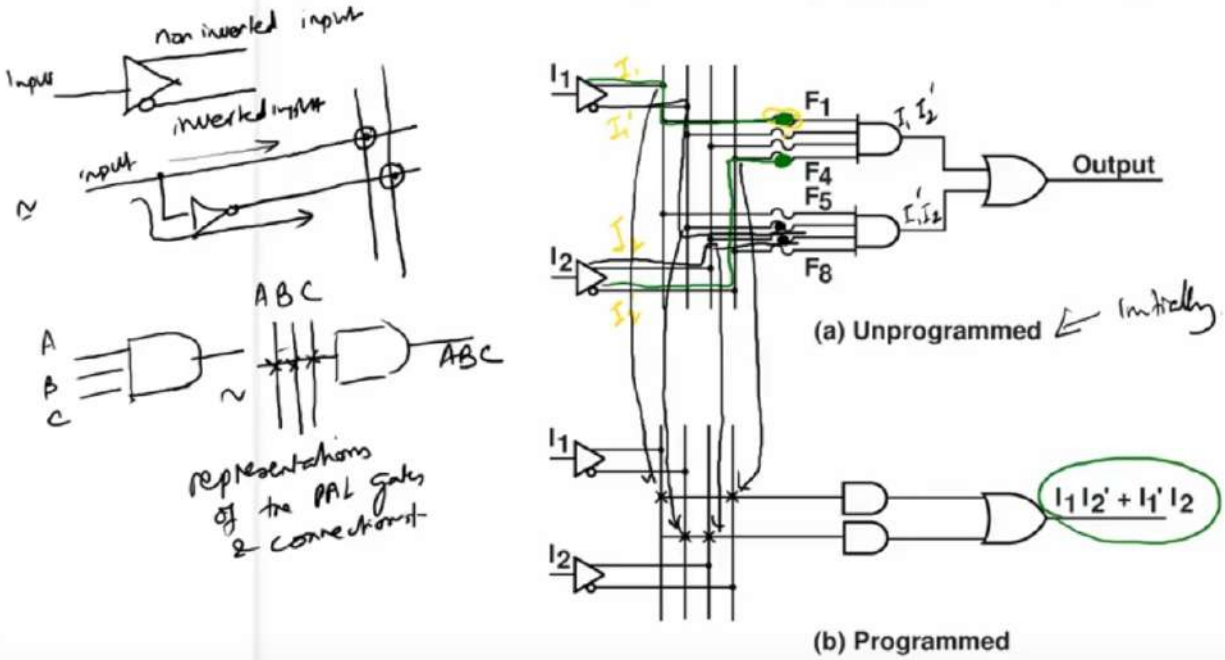


Figure 3-12 Combinational PAL Segment



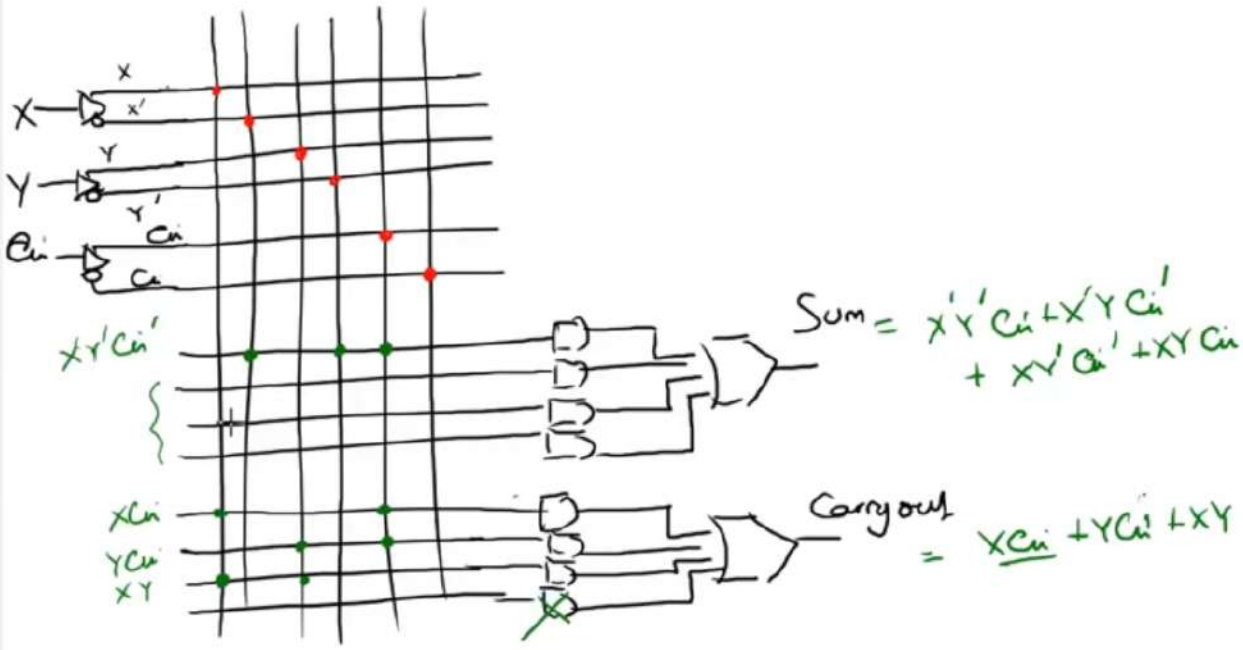

X
Y
Cin
x
x'
y
y'
Cin
Cin
x'y'Cin'
xCin
yCin
xy
Sum = x'y'Cin + x'yCin' + xy'Cin' + xyCin
Carry out = xCin + yCin' + xy

Figure 3-13 Segment of a Sequential PAL

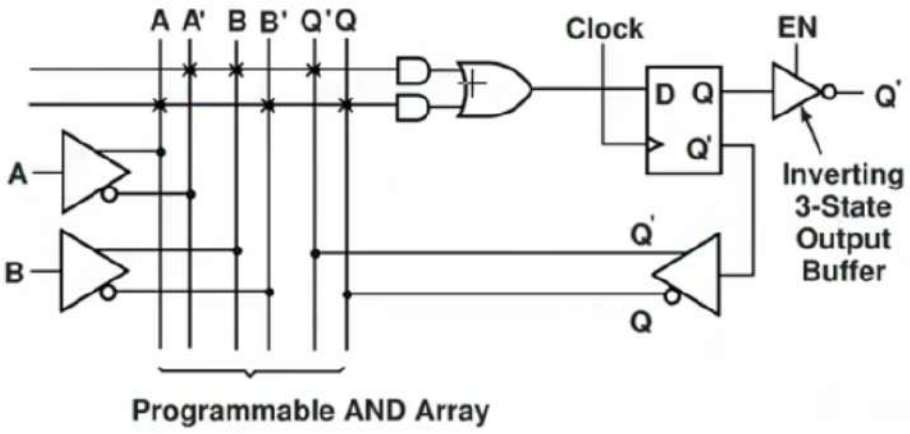
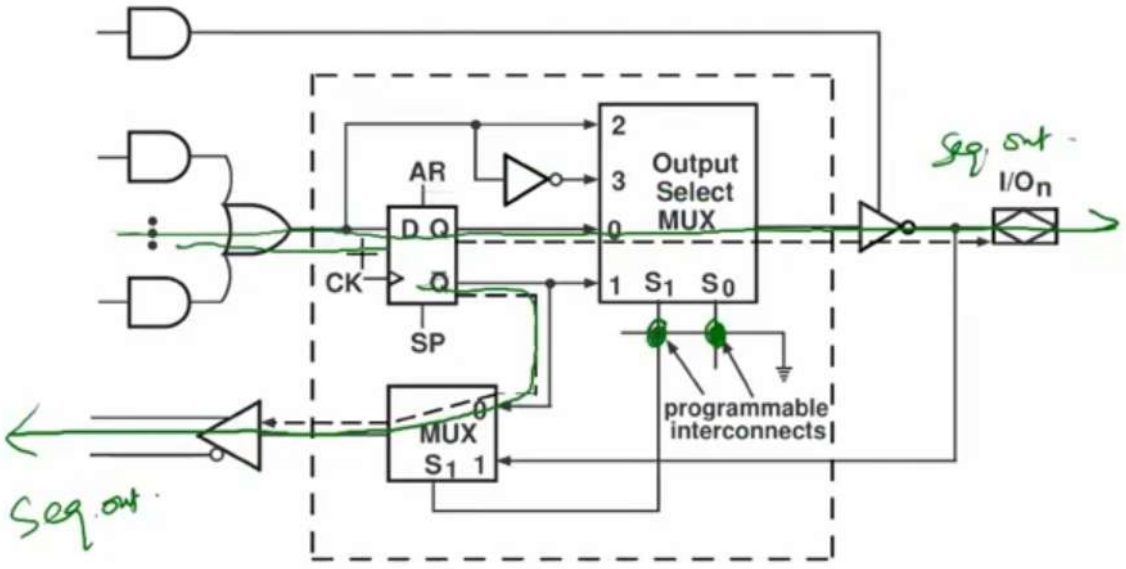


Table 3-5 Characteristics of Simple CMOS PLD's

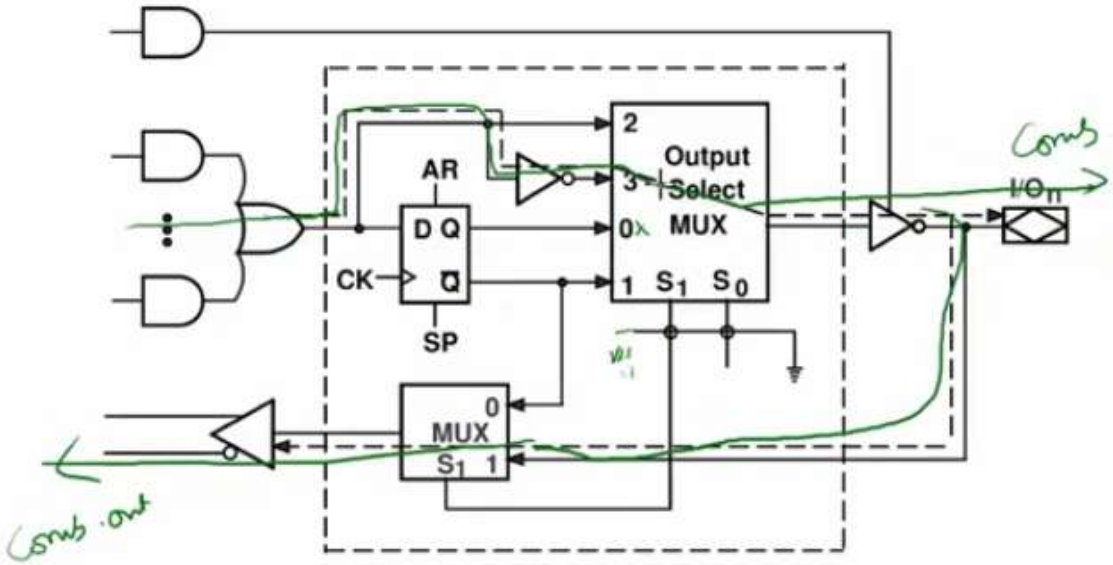
Type No.	No. of inputs	I/O	Macrocells = FFs	AND gates per OR gate
PALCE16V10	8 + $\overline{OE}$ + Clk	8	8	8
PALCE20V8	14	8	8	8
PALCE22V10	12	10	10	8–16
PALCE24V10	14	10	10	8
PALCE29MA16	5 + Clk	16	16	4–12
CY7C335	12 + $\overline{OE}$ + Clk	12	12 in/12 out	9–19

Figure 3-16a Output Macrocell

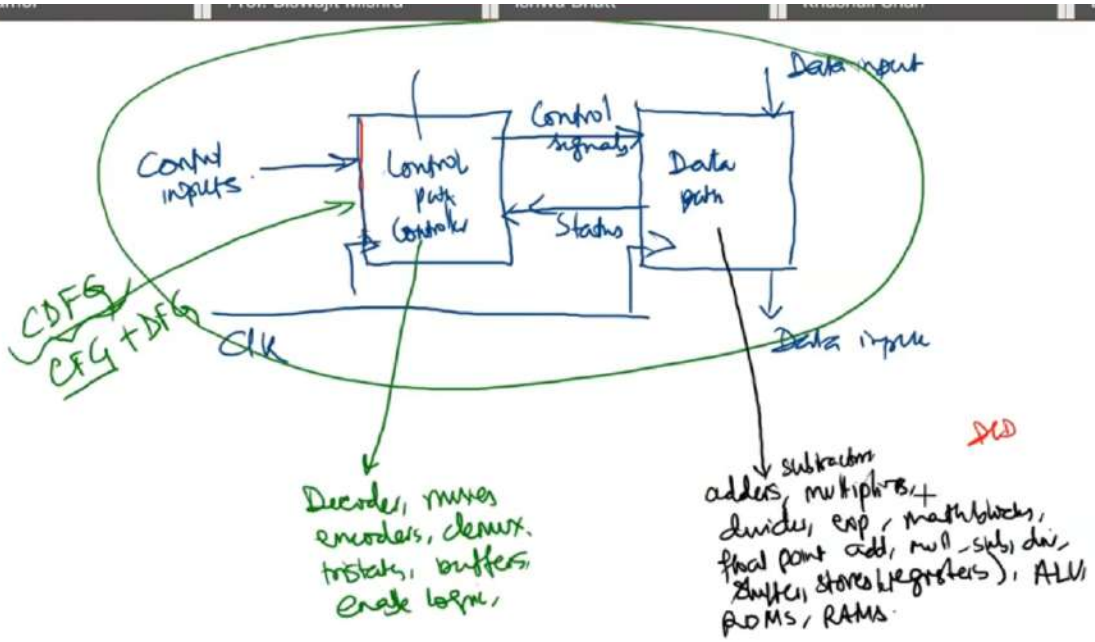


(a) paths with $S_1 = S_0 = 0$

Figure 3-16b Output Macrocell



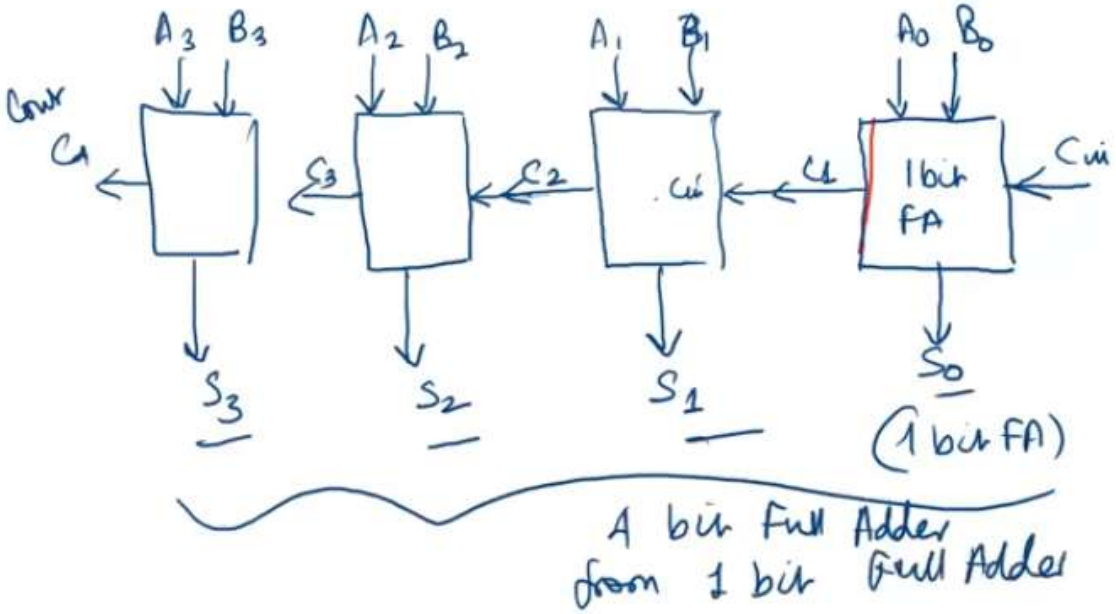
(b) paths with $S_1 = S_0 = 1$

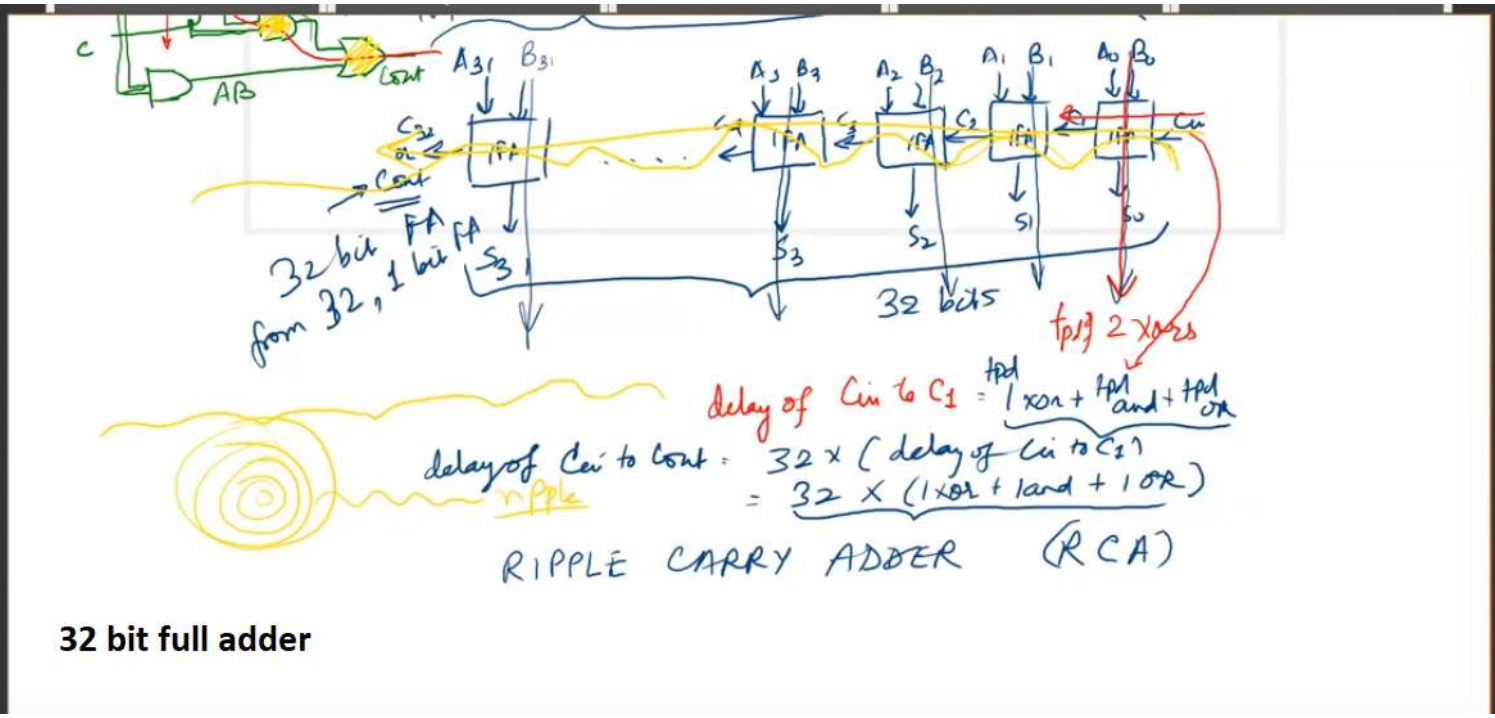


DCD

Decoder, muxes encoders, demux. tristate, buffers, enable logic,

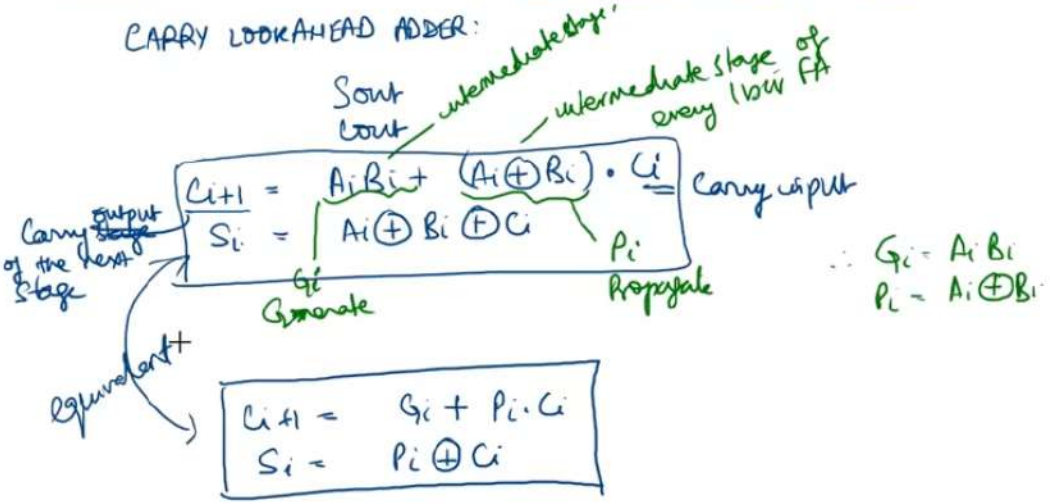
subtractors
adders, multipliers, + dividers, exp, / math blocks, floating point add, mul, sub, div, shifter, storage registers), ALU, ROMS, RAMS.

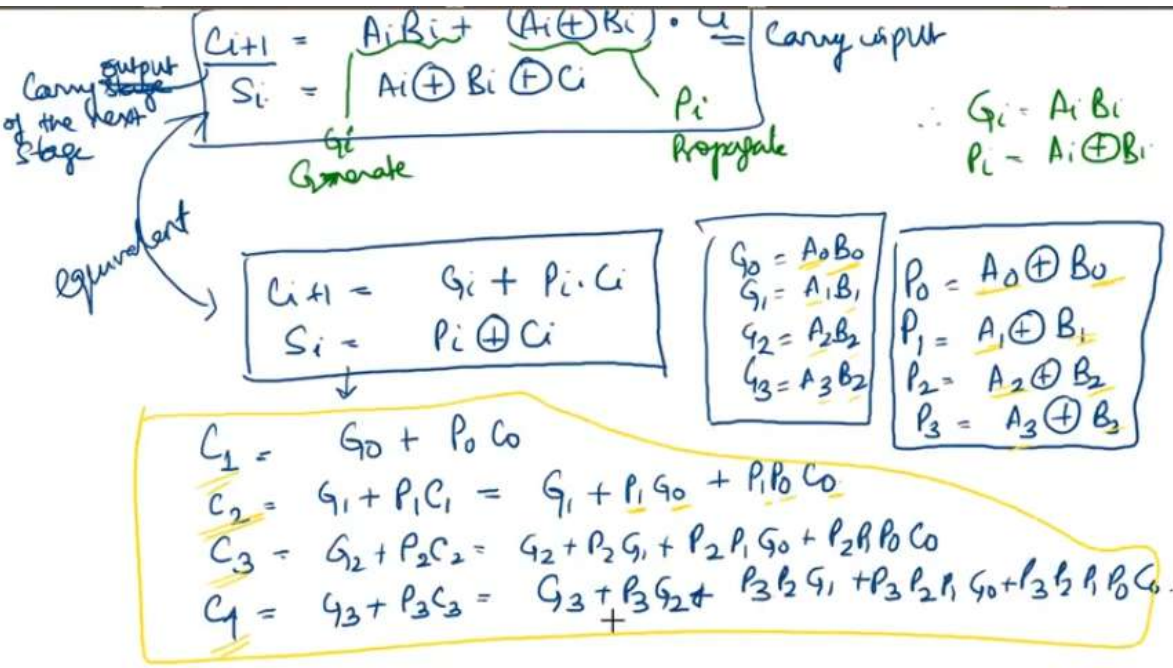

A3 B3
A2 B2
A1 B1
A0 B0
Cout
C4
C3
C2
C1
Cin
1 bit FA
S3
S2
S1
S0
(1 bit FA)
A bit Full Adder from 1 bit Full Adder



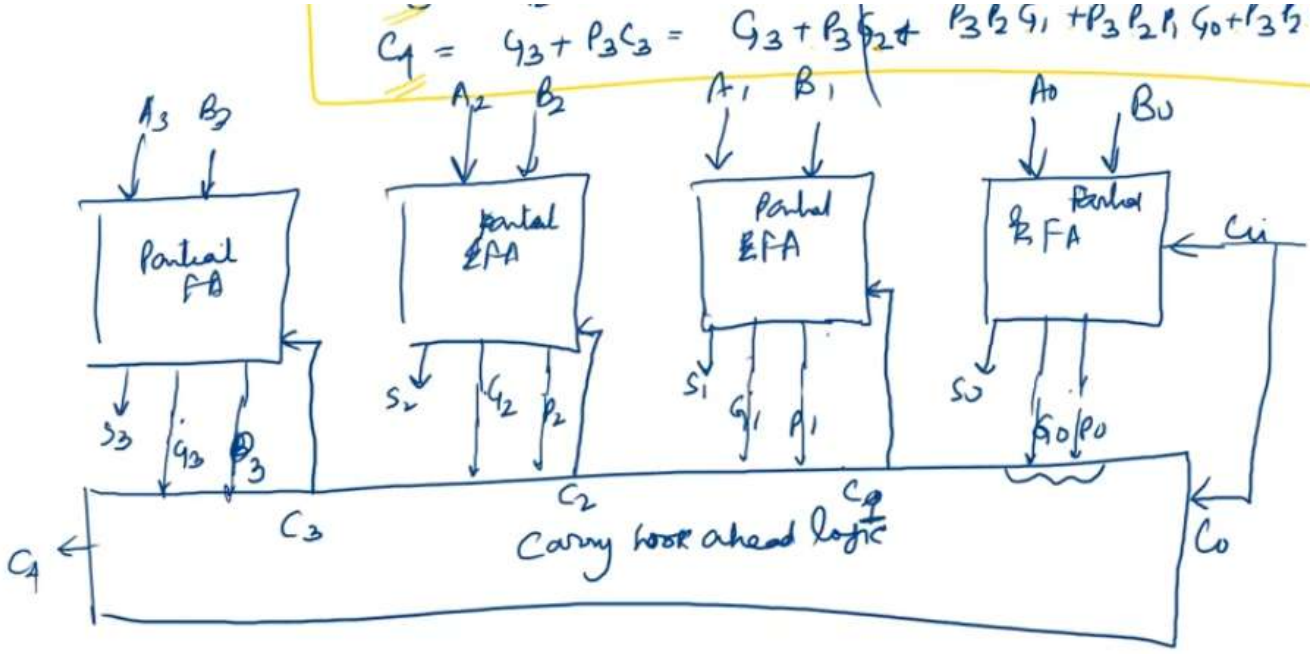
32 bit full adder

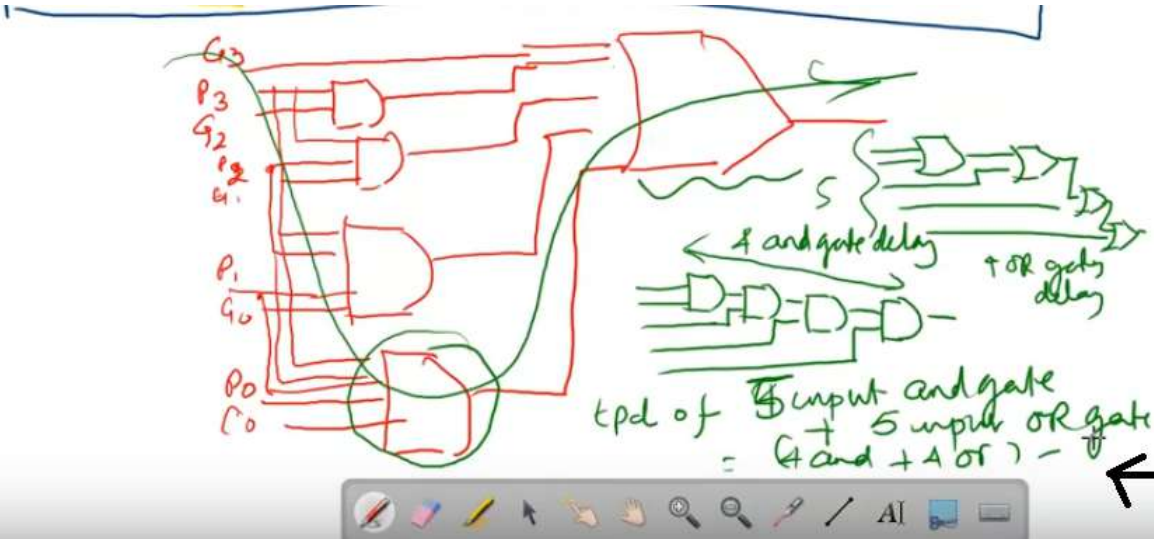
CARRY LOOKAHEAD ADDER:



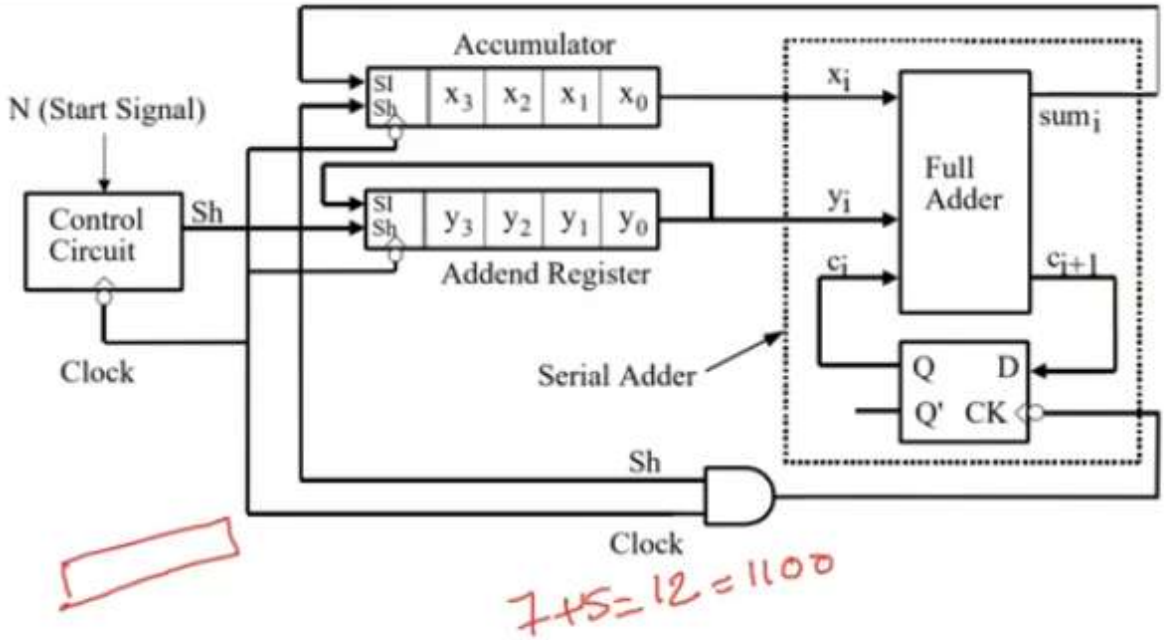

$C_{i+1} = A_i B_i + (A_i \oplus B_i) \cdot C_i$
$S_i = A_i \oplus B_i \oplus C_i$
Carry input
Carry output of the next stage
P_i
Propagate
G_i
Generate
$\therefore G_i = A_i B_i$
$P_i = A_i \oplus B_i$
equivalent
$C_{i+1} = G_i + P_i \cdot C_i$
$S_i = P_i \oplus C_i$
$G_0 = A_0 B_0$
$G_1 = A_1 B_1$
$G_2 = A_2 B_2$
$G_3 = A_3 B_2$
$P_0 = A_0 \oplus B_0$
$P_1 = A_1 \oplus B_1$
$P_2 = A_2 \oplus B_2$
$P_3 = A_3 \oplus B_3$
$C_1 = G_0 + P_0 C_0$
$C_2 = G_1 + P_1 C_1 = G_1 + P_1 G_0 + P_1 P_0 C_0$
$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$
$C_4 = G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$

$C_4 = G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$





New delay for longest path c4

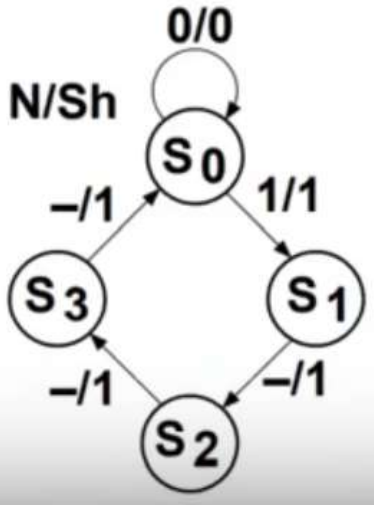


7+5=12=1100

	X	Y	c_i	sum_i	c_{i+1}
t_0	0101 (5)	0111 (7)	0	0	1
t_1	0010	1011	1	0	1
t_2	0001	1101	1	1	1
t_3	1000	1110	1	1	0
t_4	1100	0111	0	(1)	(0)

Serial adder with accumulator

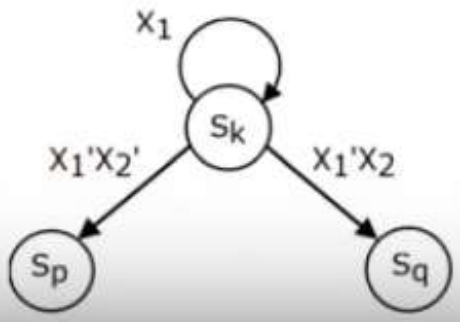
Figure 4-2 Control State Graph and Table for Serial Adder



Present State	Next State N=0	Next State N=1	Present Output (Sh) N=0	Present Output (Sh) N=1
S_0	S_0	S_1	0	1
S_1	S_2	S_2	1	1
S_2	S_3	S_3	1	1
S_3	S_0	S_0	1	1

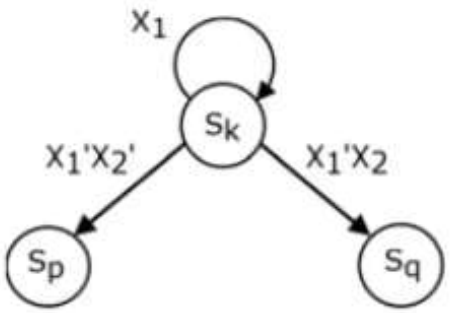
Constraints on Input Labels for Every State S_k *(From Page 123-124)*

1. If I_i and I_j are any pair of input labels on arcs exiting state S_k, then $I_iI_j = 0$ if $i \neq j$.
2. If n arcs exit state S_k and the n arcs have input labels $I_1, I_2, \ldots, I_n$, respectively, then $I_1 + I_2 + \ldots + I_n = 1$.

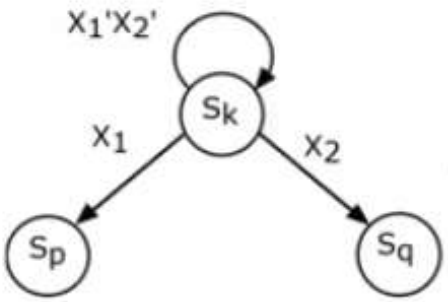


$(X_1)(X_1'X_2') = 0$
$(X_1)(X_1'X_2) = 0$
$(X_1'X_2')(X_1'X_2) = 0$
$X_1 + X_1'X_2' + X_1'X_2 = 1$

2. If n arcs exit state S_k and the n arcs have input labels $I_1, I_2, \ldots, I_n$, respectively, then $I_1 + I_2 + \ldots + I_n = 1$.



$(X_1)(X_1'X_2') = 0$
$(X_1)(X_1'X_2) = 0$
$(X_1'X_2')(X_1'X_2) = 0$
$X_1 + X_1'X_2' + X_1'X_2 = 1$



Inputs are $X_1\ X_2\ X_3$
($X_1 = X_2 = 1$ not allowed)

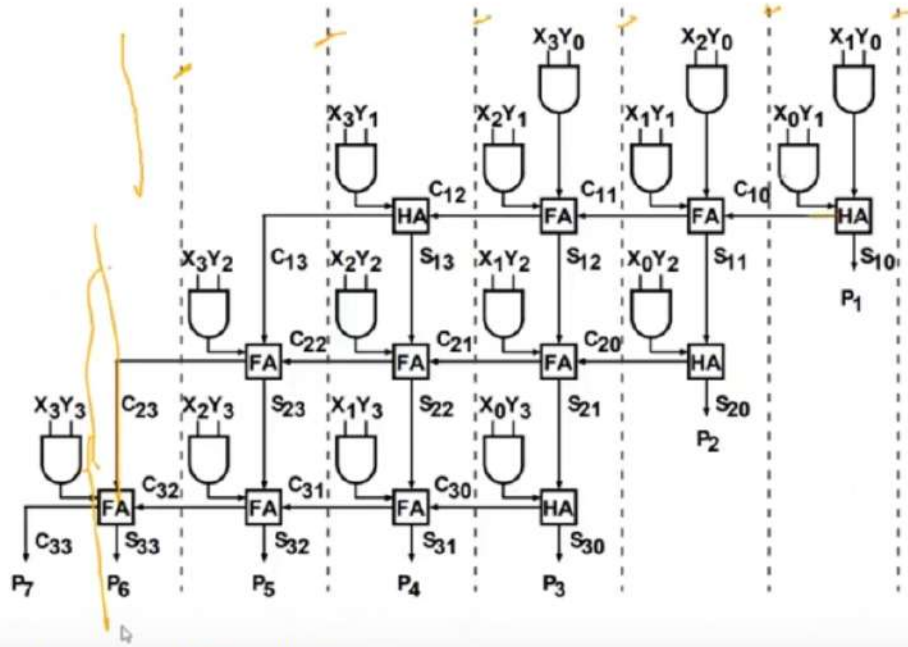
	000	001	010	011	100	101	110	111
S_k	S_k	S_k	S_q	S_q	S_p	S_p	-	-

As an implementation it needs couple of AND gates, few half adder and mostly full adder

Table 4-3 4-bit Multiplier Partial Products

		X_3	X_2	X_1	X_0	Multiplicand
		Y_3	Y_2	Y_1	Y_0	Multiplier
		X_3Y_0	X_2Y_0	X_1Y_0	X_0Y_0	partial product 0
	X_3Y_1	X_2Y_1	X_1Y_1	X_0Y_1		partial product 1
	C_{12}	C_{11}	C_{10}			1st row carries
C_{13}	S_{13}	S_{12}	S_{11}	S_{10}		1st row sums
X_3Y_2	X_2Y_2	X_1Y_2	X_0Y_2			partial product 2
C_{22}	C_{21}	C_{20}				2nd row carries
C_{23}	S_{23}	S_{22}	S_{21}	S_{20}		2nd row sums
X_3Y_3	X_2Y_3	X_1Y_3	X_0Y_3			partial product 3
C_{32}	C_{31}	C_{30}				3rd row carries
C_{33}	S_{33}	S_{32}	S_{31}	S_{30}		3rd row sums
P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0	final product

The longest path in multiplier is ending at P7.



Block diagram of 4x4 multiplier

for 4x4 multiplier
#AND gates needed = 16
#FA(Full adder) needed = 8.
#HA needed = 4.

for 8x8 multiplier
#AND gates = 64
#FA = 48
#HA = 8

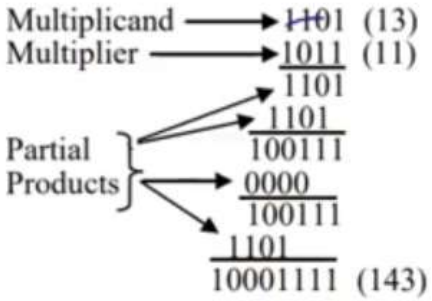
for nxn multiplier
#AND gates = n*n
#FA = n*(n-2)
#HA = n

Glitches happens because of longest path which is from beginning XiYi to P7.

To make this ckt faster we use register for shifting and making all adders together to make it faster.

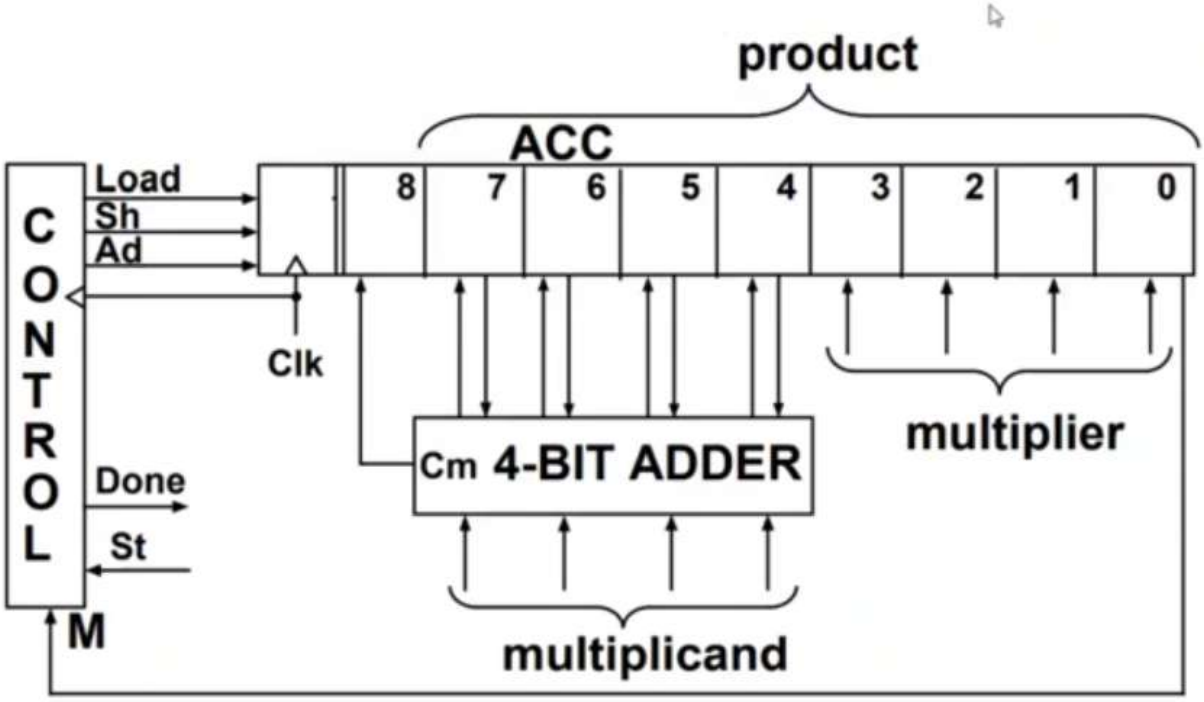
Multiplication of 13_{10} by 11_{10} In Binary – *From Page 124*

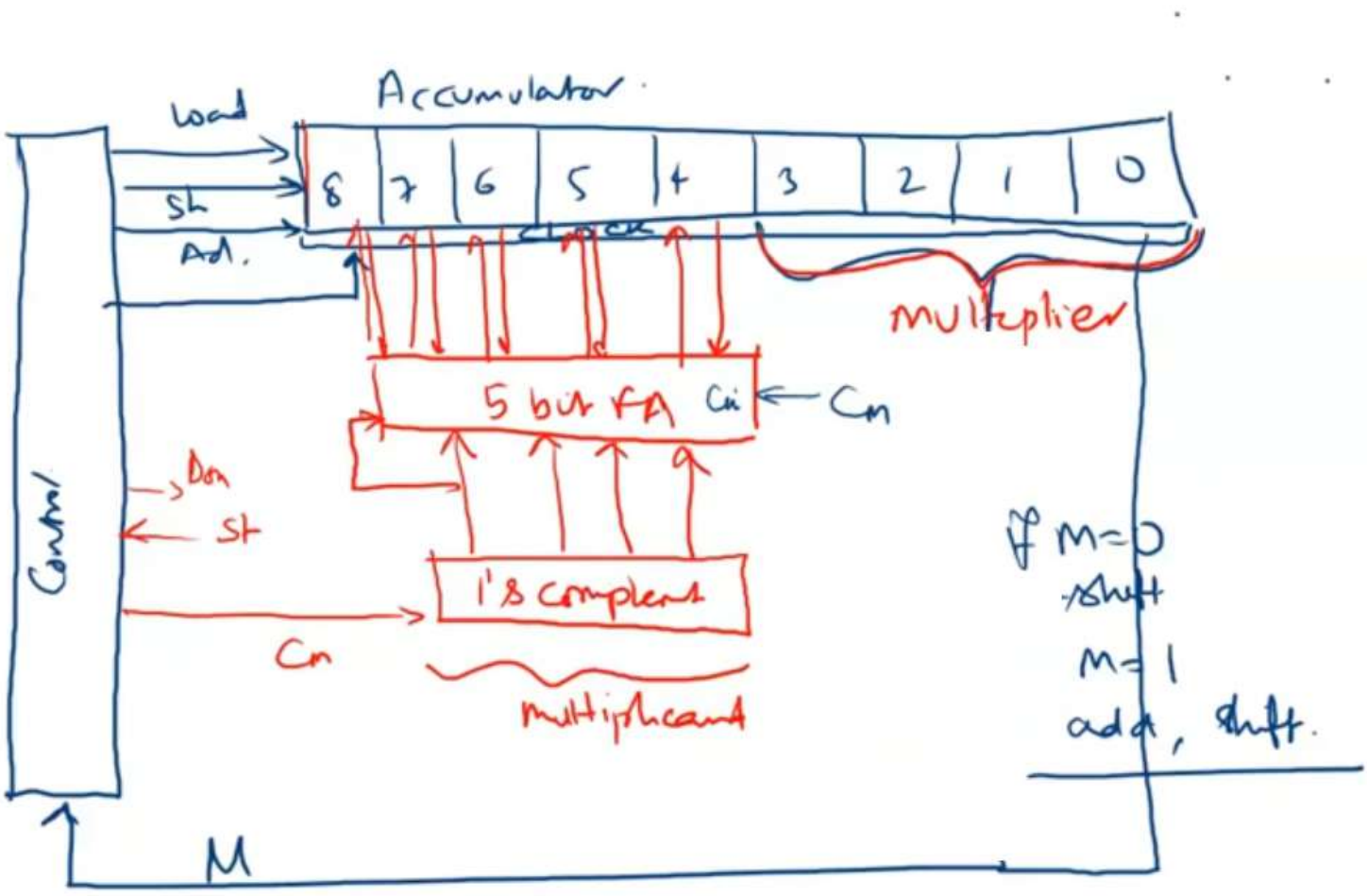
3
13 x 11 = 143



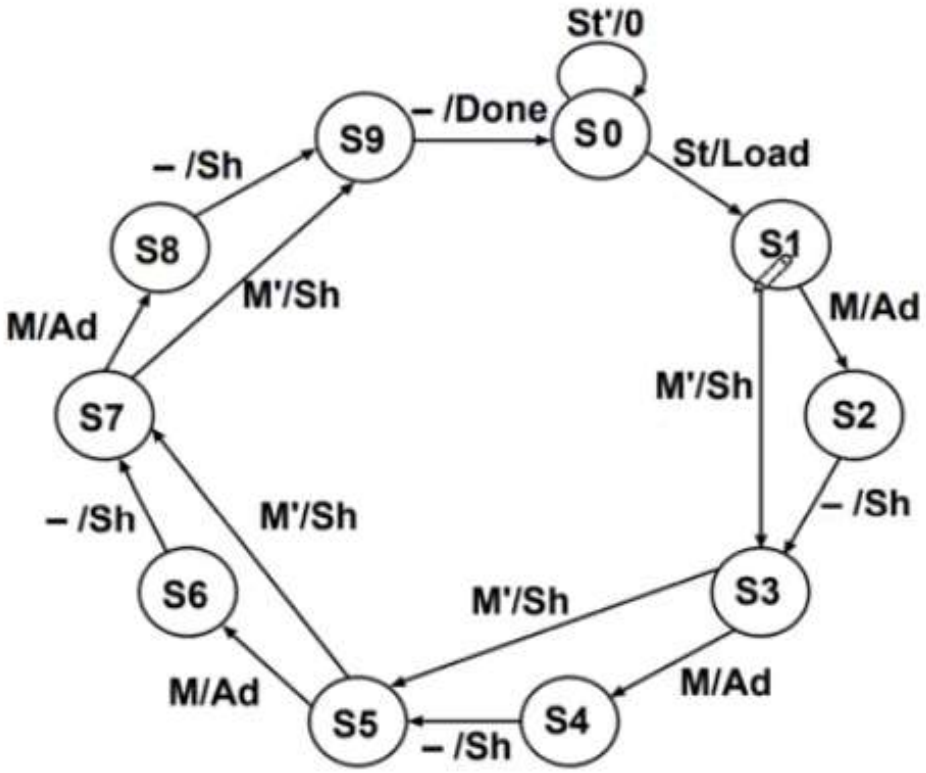
initial contents of product register	0 0 0 0 0 1 0 1 1 ← M	(11)
(add multiplicand since M=1)	1 1 0 1	(13)
after addition	0 1 1 0 1 1 0 1 1	
after shift	0 0 1 1 0 1 1 0 1 ← M	
(add multiplicand since M=1)	1 1 0 1	
after addition	1 0 0 1 1 1 1 0 1	
after shift	0 1 0 0 1 1 1 1 0 ← M	
(skip addition since M=0)		
after shift	0 0 1 0 0 1 1 1 1 ← M	
(add multiplicand since M=1)	1 1 0 1	
after addition	1 0 0 0 1 1 1 1 1	
after shift (final answer)	0 1 0 0 0 1 1 1 1	(143)

Figure 4-3 Block Diagram for Binary Multiplier

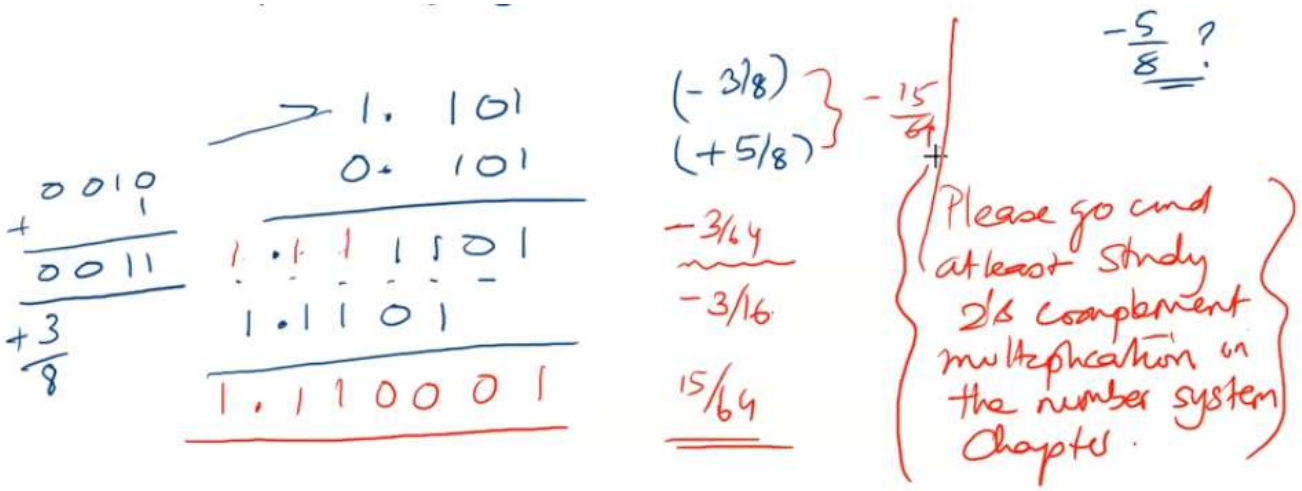


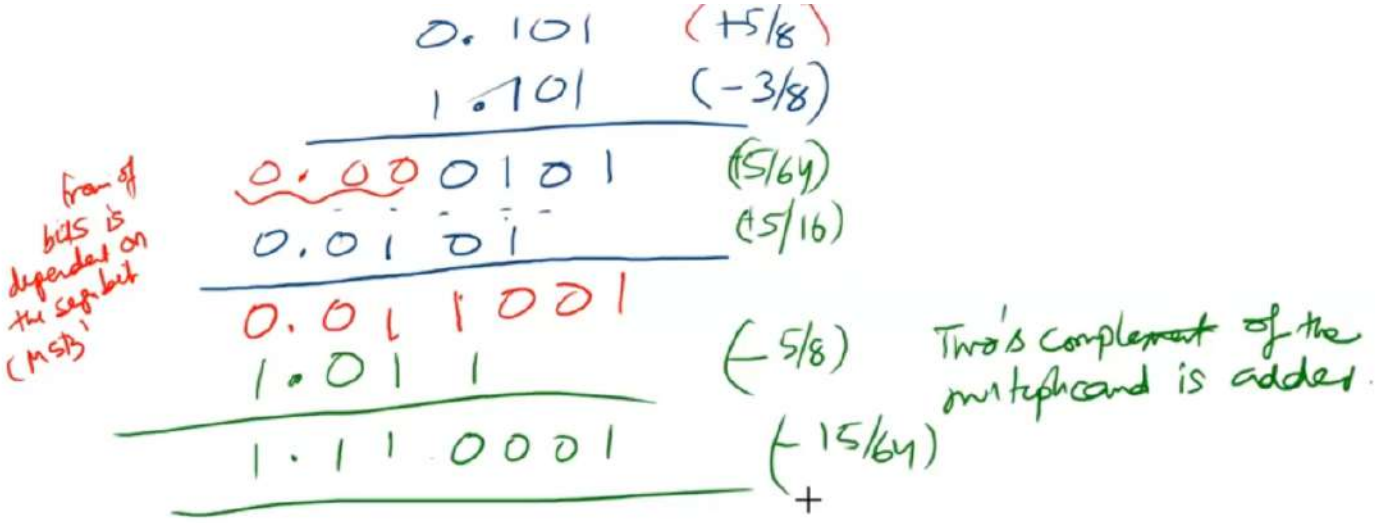


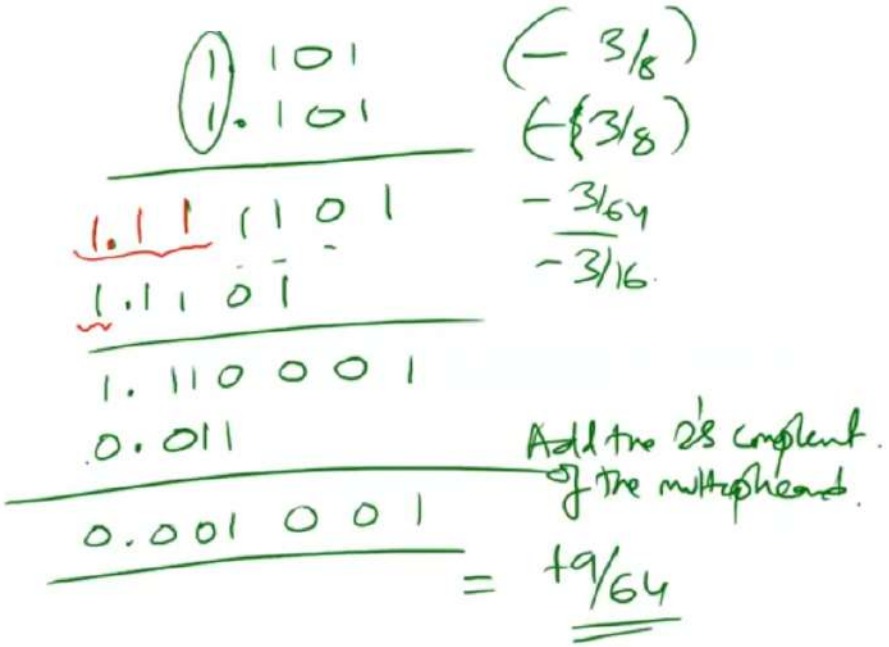
Black Diagram for Decimal Binary Multiplier



State Diagram of Multiplier


0010
+
0011
+3/8
1. 101
0. 101
1.1 1 1 0 1
1.1 1 0 1
1.1 1 0 0 0 1
(-3/8)
(+5/8)
-15/64
-3/64
-3/16
15/64
-5/8 ?
+
Please go and at least Study 2's complement multiplication in the number system Chapter.


0.101 (+5/8)
1.101 (-3/8)
0.000101 (15/64)
0.01 01 (5/16)
0.01 1001
1.011 (-5/8)
1.11 0001 (-15/64)
+
Two's complement of the multiplicand is added.
Num of bits is dependent on the sign bit (MSB)

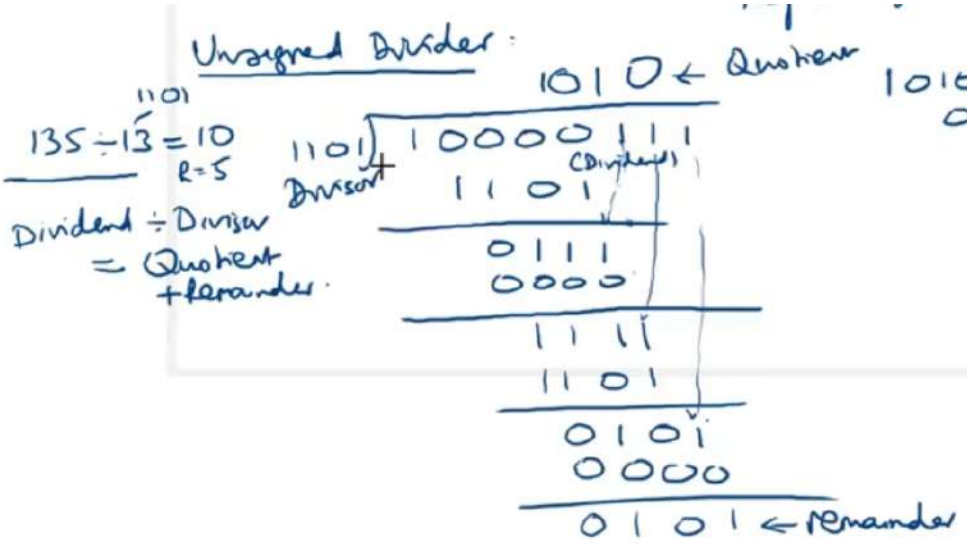


for 3 bit least value = -4 and most value = 3
for 4 bit LV = -8 and MV = 7
for 8 bit LV = -128 and MV = 127

Like that for n bit
Least value = -2^{n-1}
Most value = $2^{n-1}-1$

with 2.

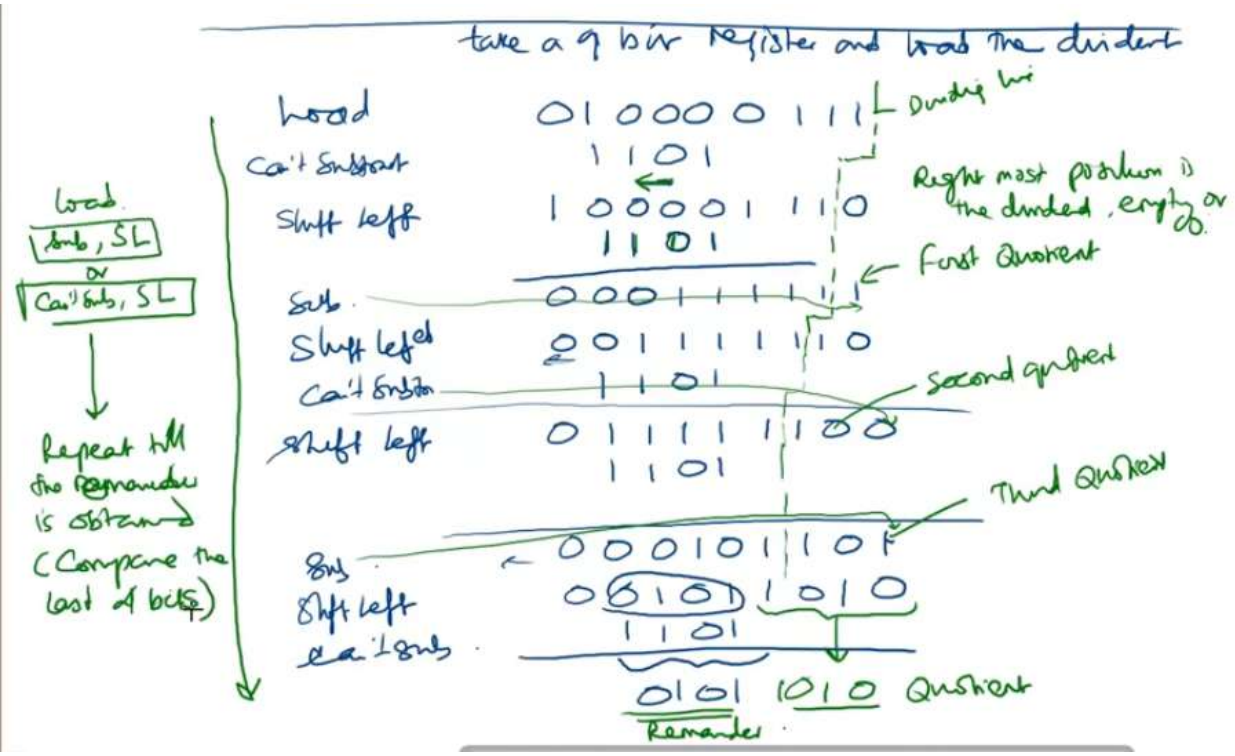
Unsigned Divider:



1101
135 ÷ 13 = 10
R=5

Dividend ÷ Divisor
= Quotient
+ Remainder.

1010
0101


take a 9 bit Register and load the divident
Load
Can't Substract
Shift left
Sub
Shift left
Can't Substract
Shift left
Sub
Shift left
Can't Sub
Dividing bit
Right most position is the divident, empty or
First Quotient
second quotient
Third Quotient
Load
Sub, SL
or
Can't Sub, SL
Repeat till the Reminder is obtained (Compare the last 4 bits)
Quotient
Remainder