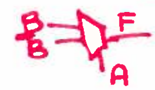
Q1 Dec. 2016

a. Several programming techniques the most used ones are: MUX based, Look-up Table, PAL, EPROM, PLA etc.

1) MUX based programming uses 2→1 Multiplexers to implement the gates as a basic unit and build on the technique to build larger functions

Example an EX-OR is built as $A \oplus B$.

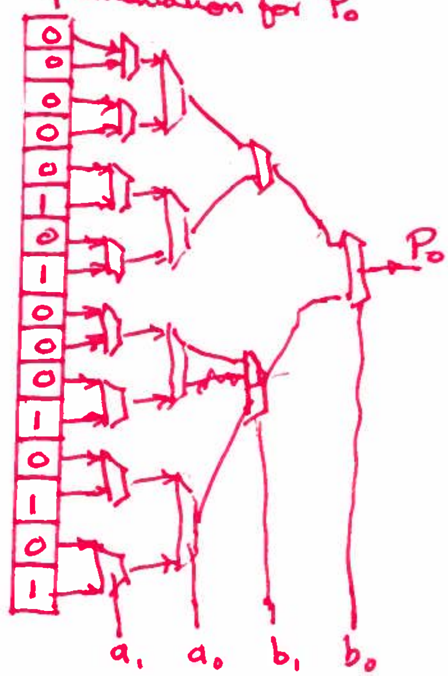


2 - Look Up Table represent a function with K-variables in a memory table with one output. It requires 2^k memory location each location is addressed by k bits with $2^k - 1$ MUXes with access time of $k_{MUX\,access}$ + 1 read of memory.

b.

a_1	a_0	b_1	b_0	P_3	P_2	P_1	P_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	1
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

Implementation for P_0



The same architecture is used for P_1, P_2 and P_3 with different Memory for P_1, P_2 & P_3

Total Area $= 4 * [16 \text{ cells} + 15 \text{ MUX}] = 4 * [16 * 2A + 15A] = 4 * 47 = 188$

Total Delay $= \underset{\text{cell}}{0.5D} + \underset{\text{MUX}}{4D} = 4.5D$

Q2 Dec 2016

$F = (N-1)^2$ . . . N is a 3-bit unsigned number $N = N_2 N_1 N_0$

	N			F							
	N_2	N_1	N_0	F_6	F_5	F_4	F_3	F_2	F_1	F_0	Equivalent Decimal
0	0	0	0	0	0	0	0	0	0	1	1
1	0	0	1	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	1	1
3	0	1	1	0	0	0	0	1	0	0	4
4	1	0	0	0	0	0	1	0	0	1	9
5	1	0	1	0	0	1	0	0	0	0	16
6	1	1	0	0	0	1	1	0	0	1	25
7	1	1	1	0	1	0	0	1	0	0	36
				Sign							

$F_0 = \sum_m (0,2,4,6) = \bar{N_0}$

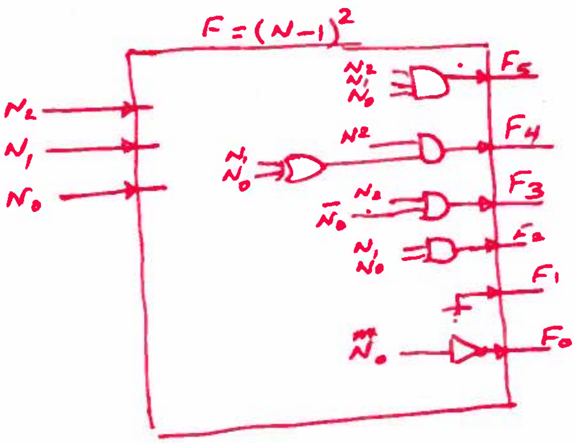
$F_1 = \sum_m 0 = 0$

$F_2 = \sum_m (3,7) = N_1 N_0$

$F_3 = \sum_m (4,6) = N_2 \bar{N_0}$

$F_4 = \sum_m (5,6) = N_2 \cdot (N_1 \bar{N_0} + \bar{N_1} N_0)$

$F_5 = \sum_m (7) = N_2 N_1 N_0$



Delay = 2 * 2 = 4ns

Area = 6 Agate

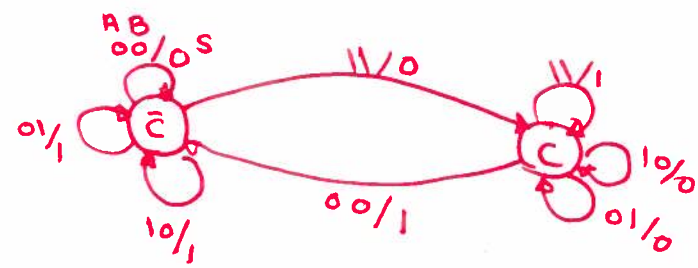
An Alternative to implement the function by a LUT. But that is requires more delay & Area.

Also it can be implemented by a multiplier using carry save adders as $N^2 - 2N + 1$ but the delay & Area is prohibitive.

Q3 Dec 2016

This adder has 2 states to remember C There is a carry

$\bar{C}$ " " no "



Present State	Next state/output 00	01	11	10
$\bar{C}$ = 0	0/0	0/1	1/0	0/1
C = 1	0/1	1/0	1/1	1/0

Transition Table

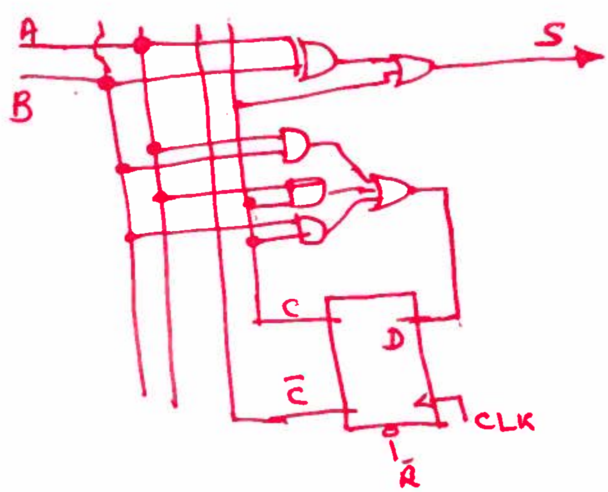
C \ AB	00	01	11	10
0		1		1
1	1		1	

$S = A \oplus B \oplus C$

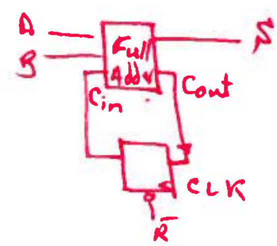
C \ AB	00	01	11	10
0			1	
1		1	1	1

$C = AB + AC + BC$

Using D FlipFlop



or



Q4 Dec 2016

Paths to be Considered

P1 $U_0 \rightarrow U_{10} \rightarrow U_{11}$

P2 $U_0 \rightarrow U_4 \rightarrow U_5 \rightarrow U_7 \rightarrow U_8$

P3 $U_0 \rightarrow U_4 \rightarrow U_1 - U_2$

P4 $U_2 \rightarrow U_{10} - U_{11}$

P5 $U_2 \rightarrow U_3 - U_6 - U_7 - U_8$

P

P6 $U_8 \rightarrow U_9 \rightarrow U_{10} \rightarrow U_{11}$

P7 $U_8 \rightarrow U_6 \rightarrow U_7 \rightarrow U_8$ *

P8 $U_8 \rightarrow U_5 - U_7 \rightarrow U_8$ *

} Critical Path with fan out only

P9 $U_8 \rightarrow U_1 \rightarrow U_2$

We will Consider Fan-Out Only

Combinational Path only

$C_{P_1} = 2 * 0.25 + 0.25 = 0.75\ ns$

$C_{P_2} = 2 * 0.25 + 0.15 + 2 * 0.1 + 0.25 + 0.25 = 1.4\ ns$

$C_{P_3} = 2 * 0.25 + 0.15 + 2 \times 0.1 + 0.25 = 1.15\ ns$

$C_{P_4} = 2 * 0.25 + 0.25 = 0.75\ ns$

$C_{P_5} = 2 * 0.25 + 0.15 + 0.25 + 0.25 = 1.15\ ns$

$C_{P_6} = 5 * 0.25 + 0.15 + 0.25 = 1.65\ ns$

$C_{P_7} = 5 * 0.25 + 0.25 + 0.25 = 1.75$

$C_{P_8} = 5 * 0.25 + 0.25 + 0.25 = 1.75$ } Critical Path

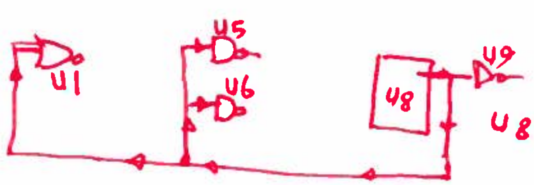
$C_{P_9} = 5 * 0.25 + 0.25 = 1.5$

$$\text{Path Delay} = \overset{C\rightarrow Q}{1.5} + \overset{C_t}{1.75} + \overset{Su}{1} = 4.25\ ns$$

$$\text{Frequency of operation} \le \frac{10^9}{4.25} \le 235\ MHz$$

$$K_T = \frac{273 + (40 + 2 * 30)}{273 + 40} = \left(\frac{373}{313}\right)^{1.5} = 1.3$$

$$\text{New frequency of operation} \simeq \frac{235}{1.3} \simeq 180\ MHz$$



U8 has the highest fan-out

Q5 Dec 2016

There are 3 paths in this circuit:

Path 1 $D_0 \to XOR \to D_0$ = 3.65 ns ✓ Critical Path

Path 2 $D_1 \to XOR \to D_0$ = 2.75 ns

Path 3 $D1 \to D1$ = 2.2 ns

a) Arrival at Point A $= 0.7 + 0.3 * 2 + 0.1(1.5+2) = \underline{1.65n}$

b) Time period $= \frac{10^9}{200 \times 10^6} = 5\,ns$

Time required for both D_0 & D_1 $= T - t_{su} = 5 - 0.5 = 4.5\,ns$

D_0

Arrival at point $D_0 = 1.65 + 2 = 3.65\,ns$

$T_{set\text{-}up}$ slack $= 4.5 - 3.65 = \underline{0.85\,ns}$

$T_h \leq t_{cQ} + t_{\text{Q}} - t_{cs} = t_{cQ} + t_{Q} = 1.65 + 0.3 + 0.2 * 3 + 0.2 = 2.75\,ns$

Hold time slack $= 2.75 - 0.2 = \underline{2.55}$

D_1

Arrival at point $D_1 = 0.7 + \overbrace{3 * 0.3}^{f_o} + \overbrace{(2 * 1.5 * 3)0.1}^{load} = 0.7 + 0.9 + 0.6 = 2.2\,ns$

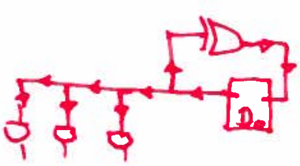
T_{setup} slack $= 4.5 - 2.2 = 2.3\,ns$

$T_h \leq t_{cQ_{min}} + t_{Q_{min}} - t_{cs} = 0.7 + (3 * 0.3) + (2 * 1.5 * 3)0.1 = 2.2\,ns$

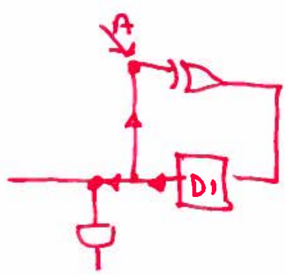
T_{hold} slack $= 2.2 - 0.2 = \underline{2\,ns}$

Path 1 $= 0.7 + 4 * 0.3 + (4.5+2)0.1 + 0.3 + 0.2 + 3 * 0.2 = 365$

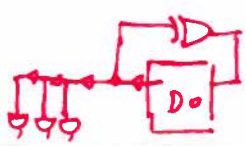
Maximum frequency $= \frac{1}{(3.65 + 0.5)ns} \approx 240\,MHz$



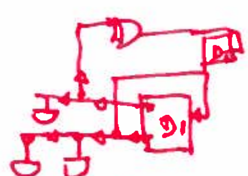
Critical path



Arrival @ A



Slack for hold & setup time D_0



Slack for hold & setup time D_1

Q6 Dec 2016

" This is only sample of possible answer "

refer to Alans notes for more detail

MUX

```
library ieee ;
use ieee . std_logic. 1164 . all ;
entity MUX is
 Port (D1, D2, C : in std_logic;
       Z : out std logic);
 end MUX;

 Architecture RTL of MUX is
   Component AND2 is
    Port (A, B : in std_logic ; C : out std_logic); end component AND2;
   Component OR2 is
    Port (A, B : in std_logic; C : out std_logic); end Component OR2;
   Component INV is
    Port ( A : in std_logic; B : out std logic); end Component INV;
    Signal S1, S2, S3 : std_logic;
 → For statements ⟶ work . . .      Configuration Statements ←
   Port map statement →
 end RTL;
```

Generator

```
library ieee;
use ieee . std_logic 1164 . all;
entity Generator is
Port (A, B, C : out std logic);  end Generator;
Architecture of Behavioral of Generator is
Begin
P1 : Process
Begin
  A <= '0';  B <= '0';  C <= '0';  wait for T1 ns;
  → other values of A, B, C with wait statements ←
end Process P1;  end Behavioral;
Use ieee . Std.logic . 1164 . all
entity Test Bench is        end Test Bench;
Architecture of TB is
Component Generator is  Port (A, B, C : out std.logic);  end Compond;
Component MUX is  Port (D1, D2, C : in std_logic; Z : out std logic; end Compont;
Signal S1, S2, S3 : Std logic
Begin  Generator  Port map (S1, S2, S3);  Portmap (S1, S2, S3 Z);
end TB;
```