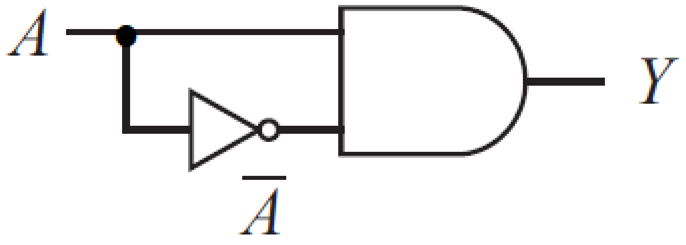
Digital Electronic Circuits
Section 1 (EE, IE)

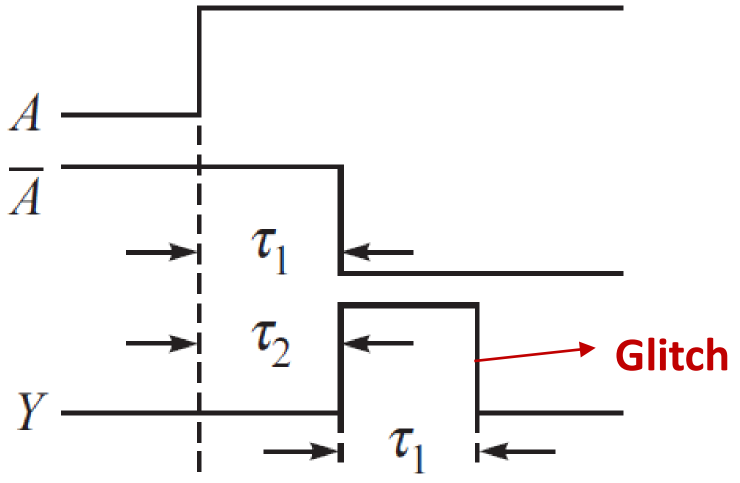
Lecture 10

Static 0 Hazard

***A*.*A′* = 0**

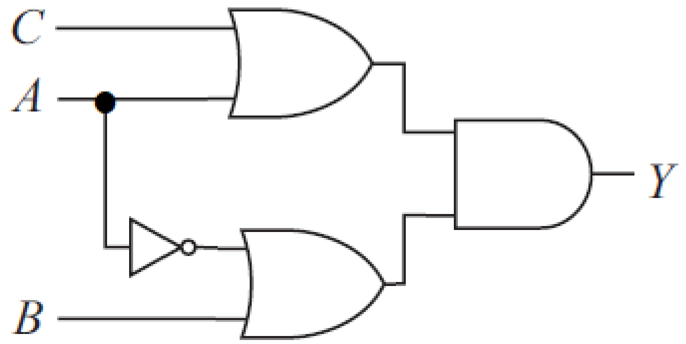


In Static 0 Hazard, output should remain static at 0 according to Boolean Logic but glitch occurs under certain input condition.



τ_1 = NOT gate delay

τ_2 = AND gate delay



***Y* = (*A* + *C*).(*A′* + *B*)**

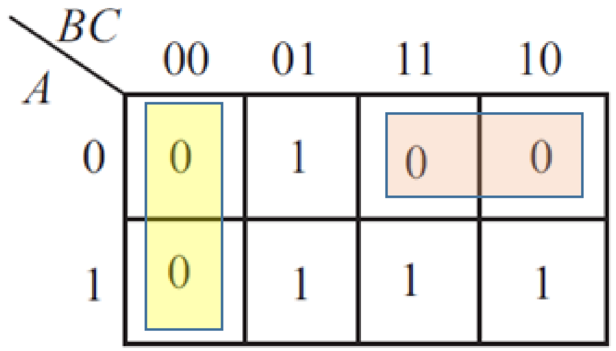
B = 0, *C* = 0

A : 0 → 1

Glitch occurs

Detecting Static 0 Hazard

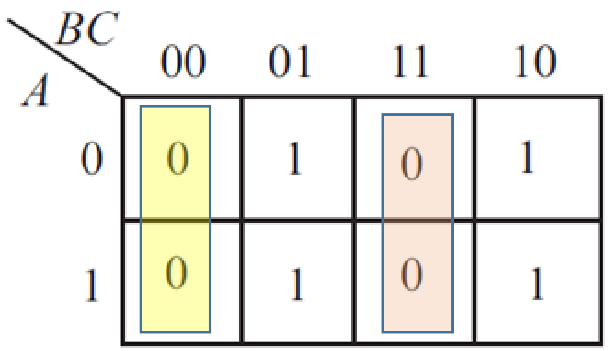
- Two logically adjacent cells with output 0 in K-Map not covered by a common sum term.
- Boolean expression (*A*.*A*′) for certain condition.



Y **= (*A* + *B*′).(*B* + *C*)**

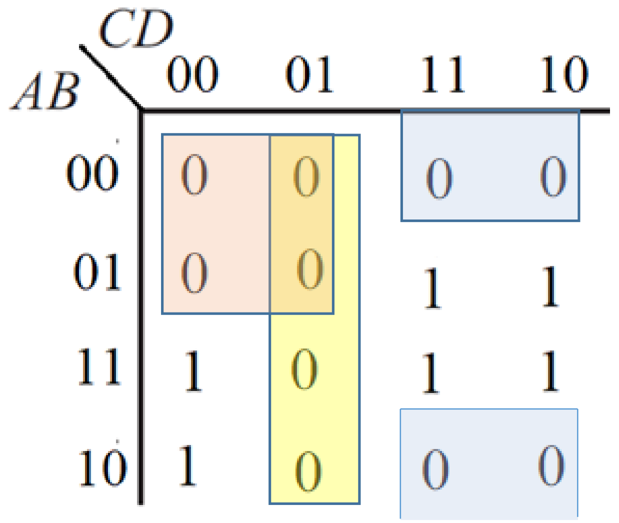
Glitch, *ABC* : 000 → 010

Static 0 Hazard



Y **= (*B* + *C*).(*B*′ + *C*′)**

No Hazard for one variable changing



Y **= (*A* + *C*).(*C* + *D*′).(*B* + *C*′)**

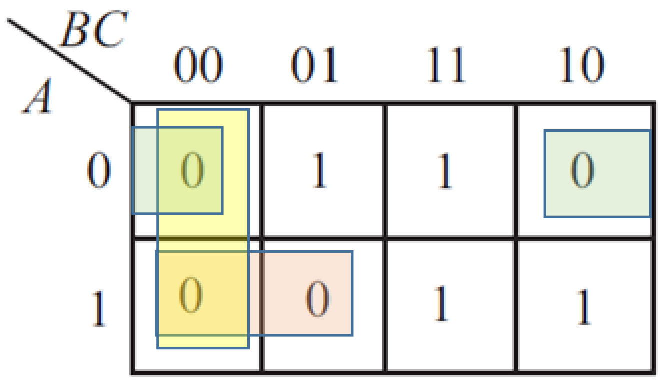
Glitch: Static 0 Hazard

ABCD **: 0001→ 0011**

1001→ 1011

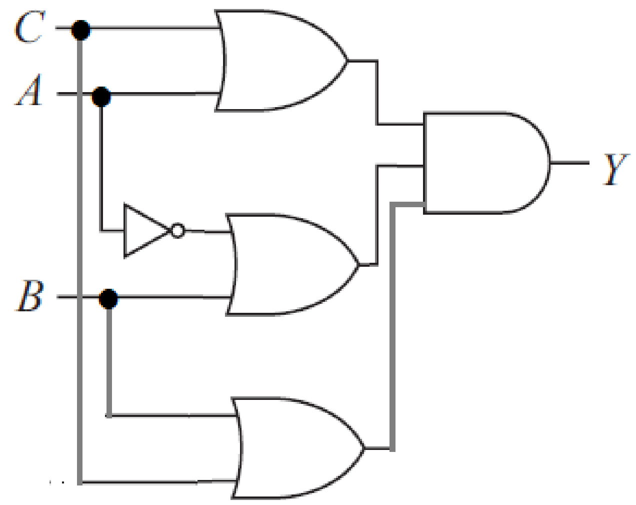
0000→ 0010

Static 0 Hazard and its Cover

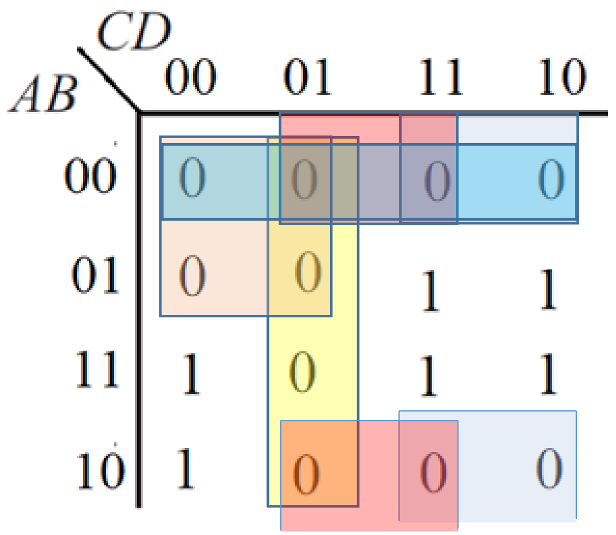


A \ BC	00	01	11	10
0	0	1	1	0
1	0	0	1	1

***Y* = (*A* + *C*).(*A’* + *B*).(*B* + *C*)**



Hazard-free circuit



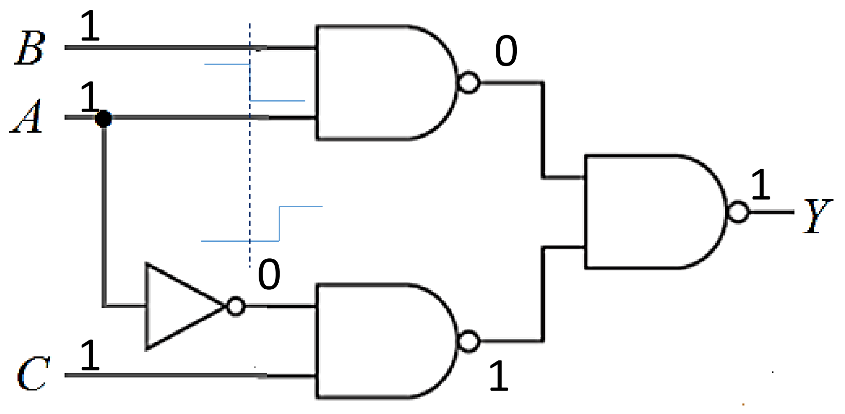
AB \ CD	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	1	0	1	1
10	1	0	0	0

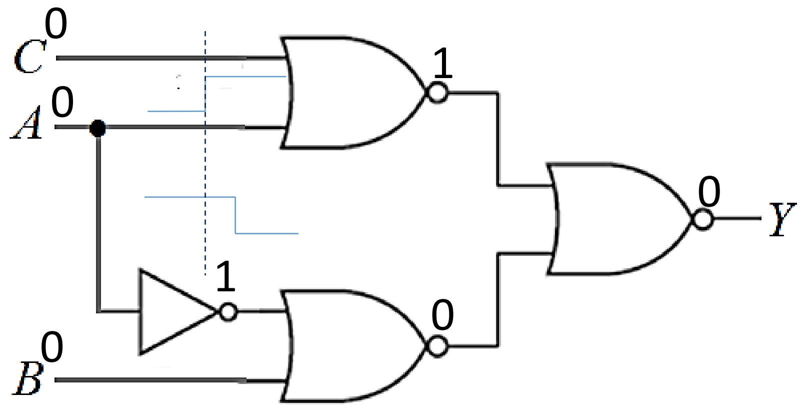
***Y* = (*A* + *C*).(*C* + *D’*).(*B* + *C’*)**
.(*B* + *D’*).(*A* + *B*)

(*B* + *C*) = 0 for *B* = 0, *C* = 0
This OR gate output when fed to AND gate, suppresses glitch.

Hazard-free by covering logically adjacent 0s with common sum term

Hazard in NAND-NAND, NOR-NOR Circuit





Static 1 and Static 0 Hazards can also be avoided by adding delay (controlled) in the transition path.

Y* = *A'*.*C* + *A*.*B

$B = 1$, $C = 1$

$A : 1 \rightarrow 0$

Glitch occurs

Cover: $(B.C)'$ as 3rd input to output NAND

***Y* = (*A* + *C*).(*A'* + *B*)**

$B = 0$, $C = 0$

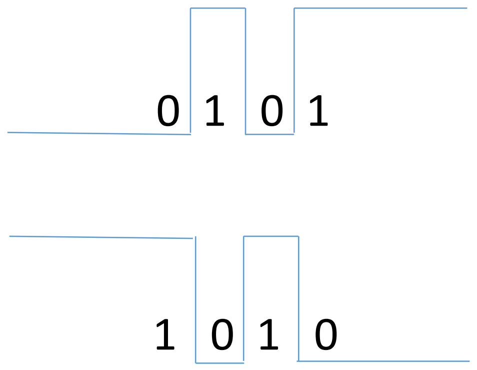
$A : 0 \rightarrow 1$

Glitch occurs

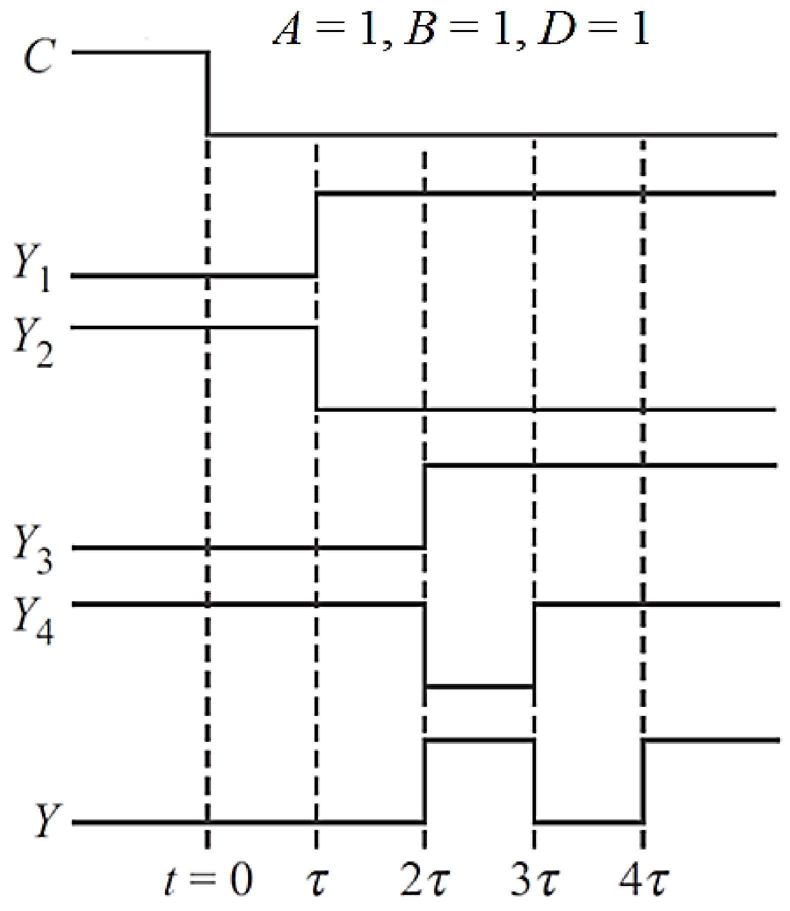
Cover: $(B + C)'$ as 3rd input to output NOR

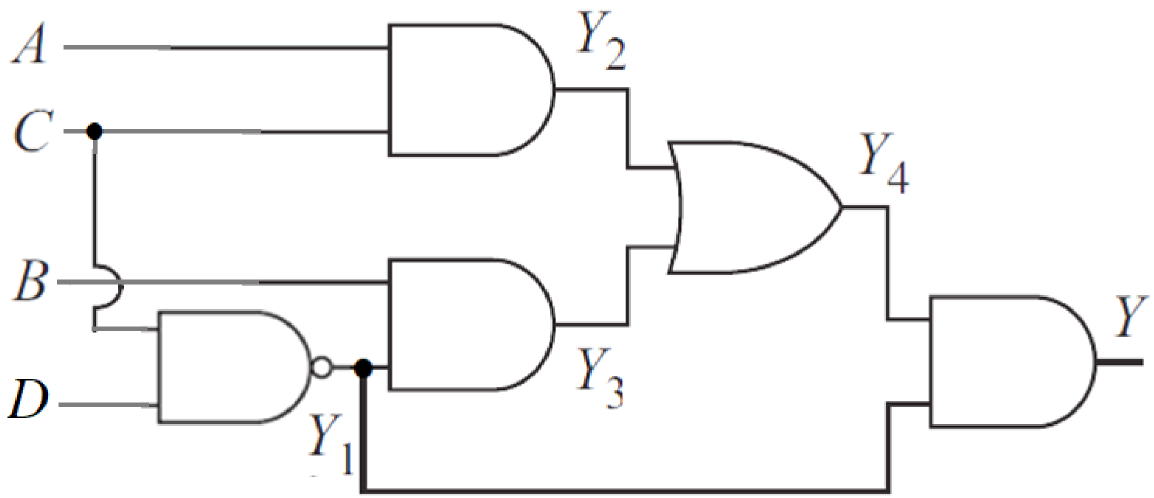
Dynamic Hazard

- Potential for multiple transitions before settling to final value while Boolean logic asks for only one transition.
- One input variable is to have three or more paths to the output.
- No. of levels three or more.
- For specific combination of input variables, Boolean expression reduces to $(A + A').A$ or $A + A'.A$



Dynamic Hazard: Example





Y = (A.C + B.(C.D)’).(C.D)’

Covering Dynamic Hazard

Consider,

$\boldsymbol{Y = (A.C + B.C').(CD)'}$

For, $A = 1$, $B = 1$, $D = 1$

$Y = (C + C').C'$ ⇨ Multiple transition at Y for C: $1 \rightarrow 0$

To prevent, cover implicit Static 1 or 0 hazard

$\boldsymbol{Y = (A.C + B.C' + A.B).(CD)'}$

For, $A = 1$, $B = 1$, $D = 1$

$Y = (C + C' + 1).C' = C'$ ⇨ Single transition

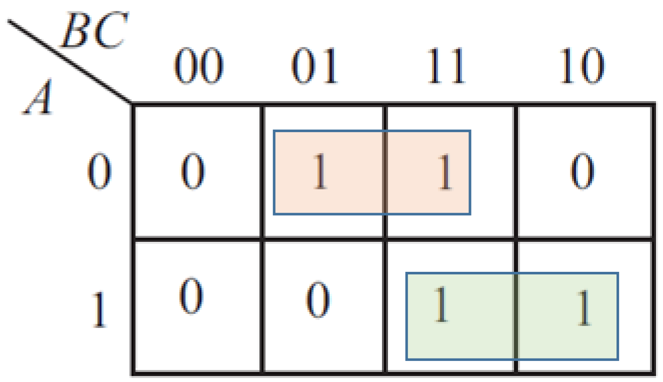
Alternatively, convert multiple-level to 2-level circuit and cover hazard, if any.

Digital Building Blocks:
Beyond Basic Gates

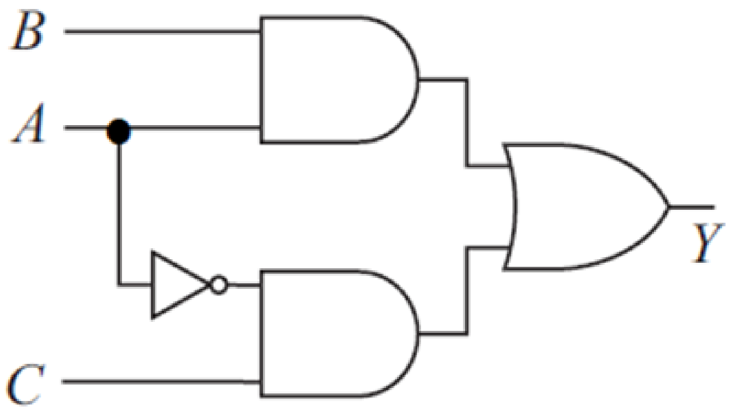
If-Then-Else

A	*B*	*C*	*Y*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Y* = *A'.C* + *A.B



$A \backslash BC$	00	01	11	10
0	0	1	1	0
1	0	0	1	1



If ***A*** (i.e. ***A*** = 1)

Then, ***Y*** = ***B***

Else, ***Y*** = ***C***

A	*Y*
0	***C***
1	***B***

Multiplexer

$$Y = A'.C + A.B$$

S_0	Y
0	D_0
1	D_1

A multiplexer steers one of the many inputs to an output based on control input(s).

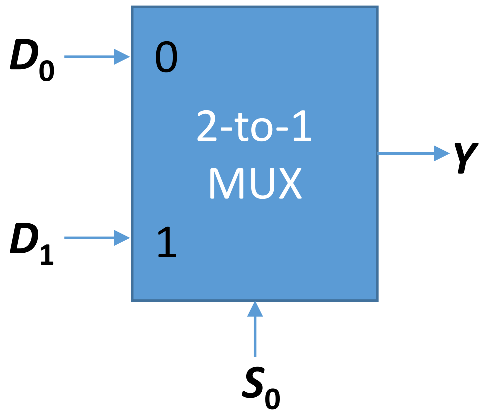
Consider,

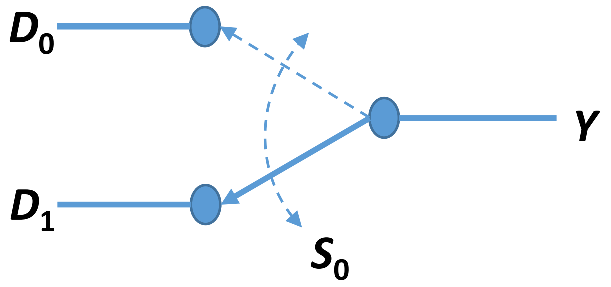
A = Control (Select) input, S_0

B = Data input, D_1

C = Data input, D_0

$$Y = S_0'.D_0 + S_0.D_1$$





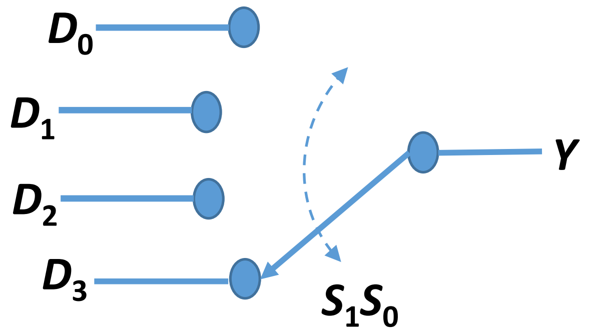
Consider, $S_0 = A$, $D_0 = B$, $D_1 = B'$
then, $Y = A'.B + A.B' = A \text{ XOR } B$

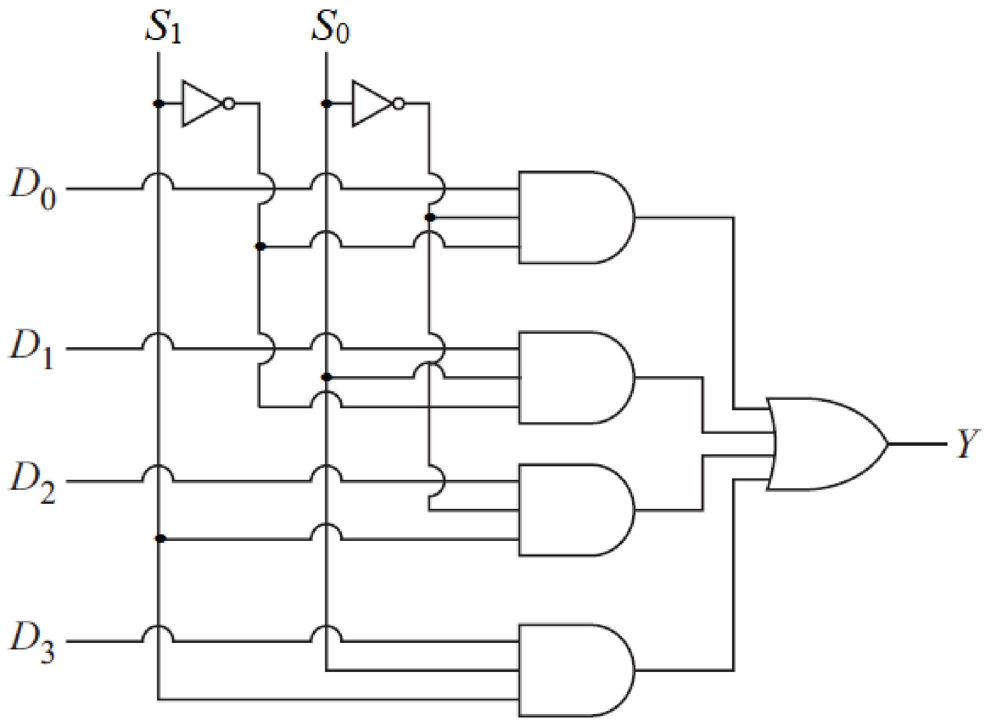
How to get $Y = A.B$?

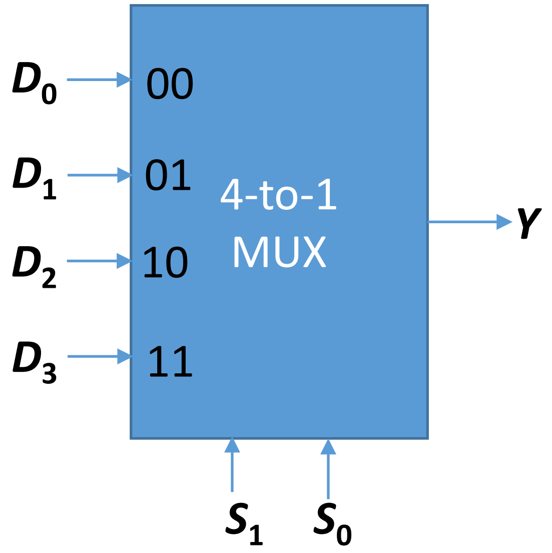
4-to-1 Multiplexer

S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

$$Y = S_1'S_0'D_0 + S_1'S_0D_1 + S_1S_0'D_2 + S_1S_0D_3$$

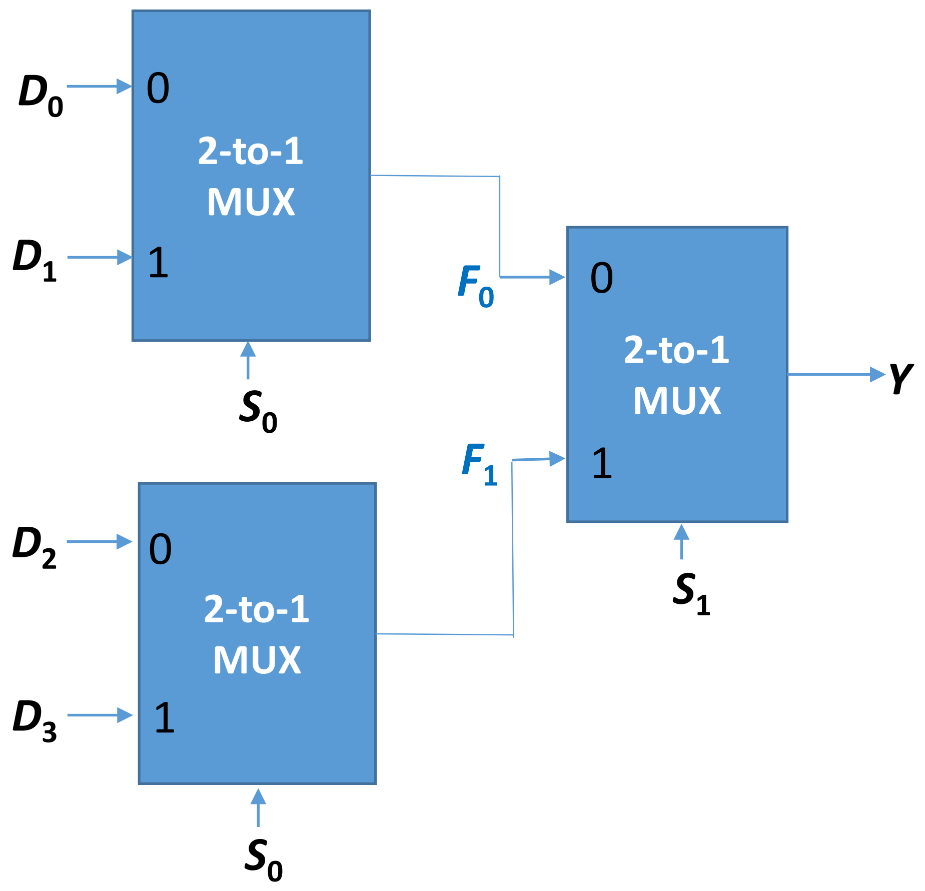




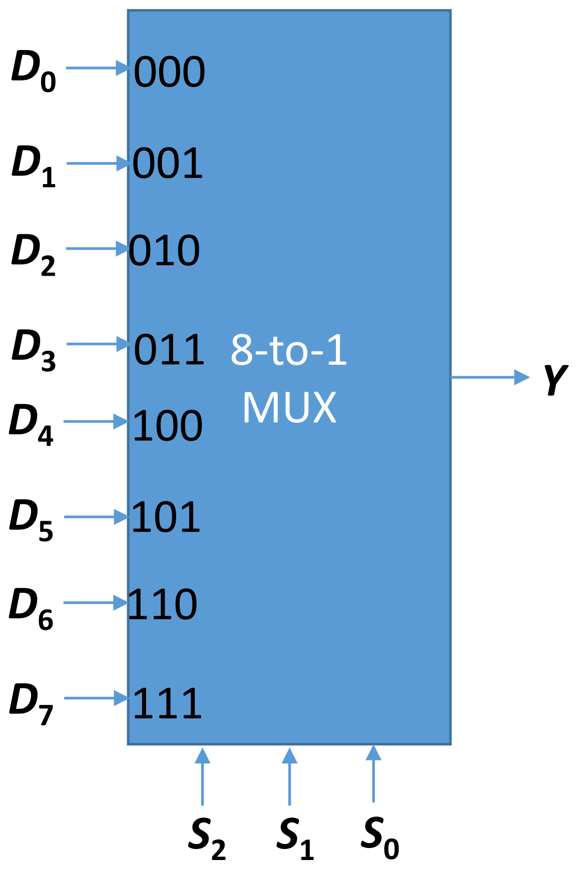


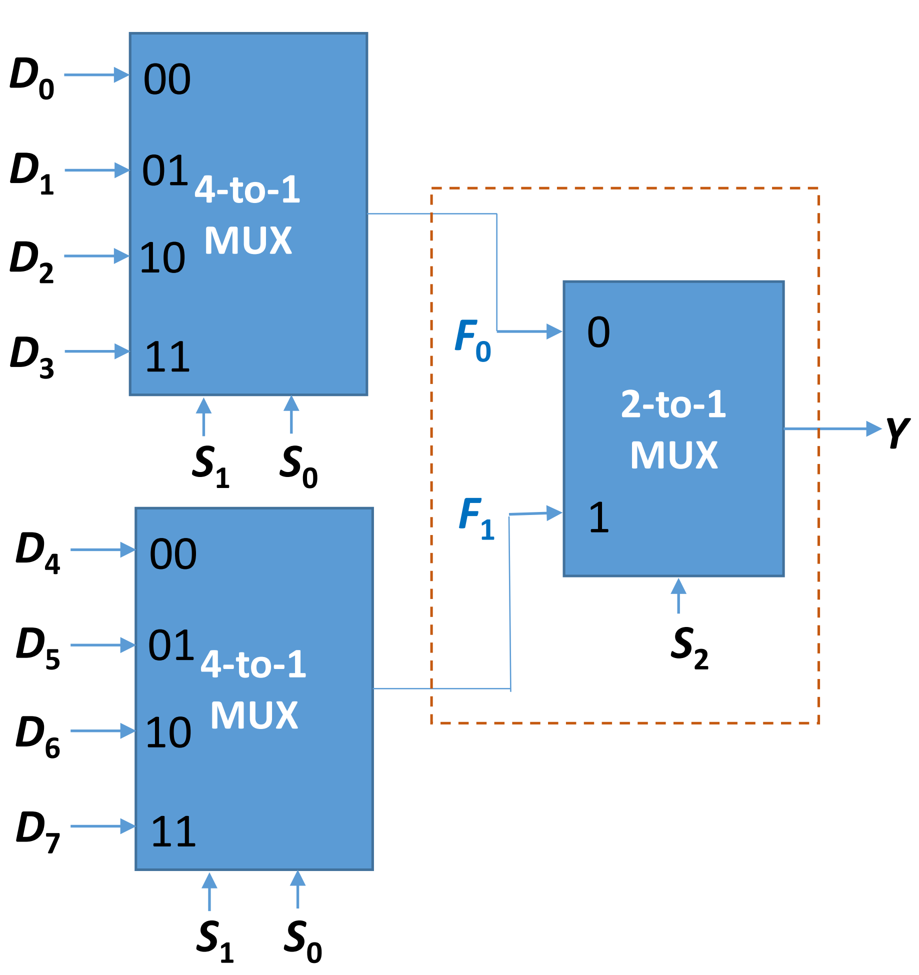
4-to-1 MUX from 2-to-1 MUX

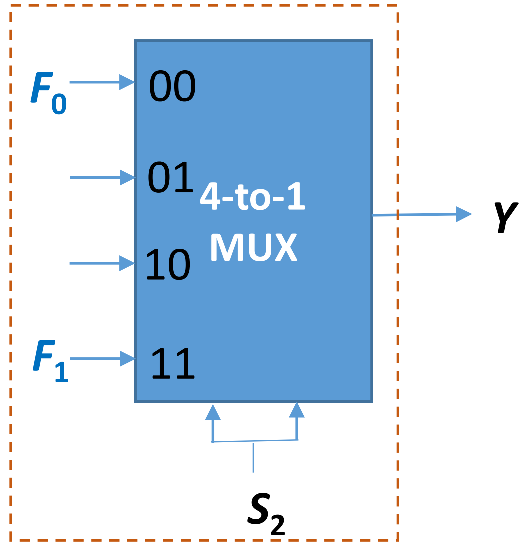
$$
\begin{aligned}
Y &= S_1'S_0'D_0 + S_1'S_0D_1 + S_1S_0'D_2 + S_1S_0D_3 \\
&= S_1'.(S_0'D_0 + S_0D_1) + S_1.(S_0'D_2 + S_0D_3) \\
&= S_1'.F_0 + S_1.F_1
\end{aligned}
$$



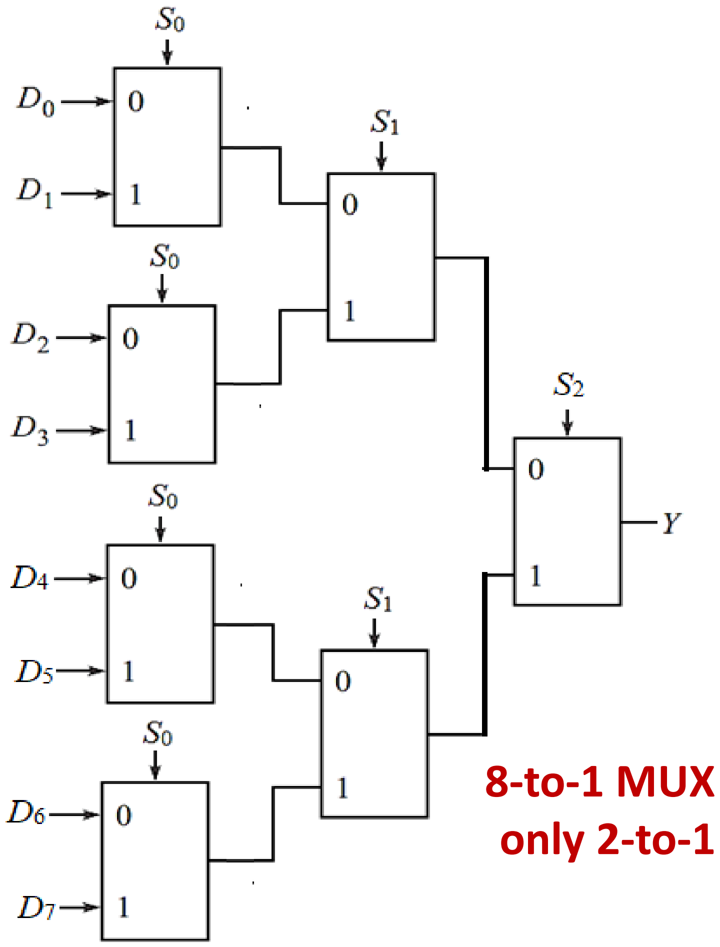
Higher order MUX from lower order



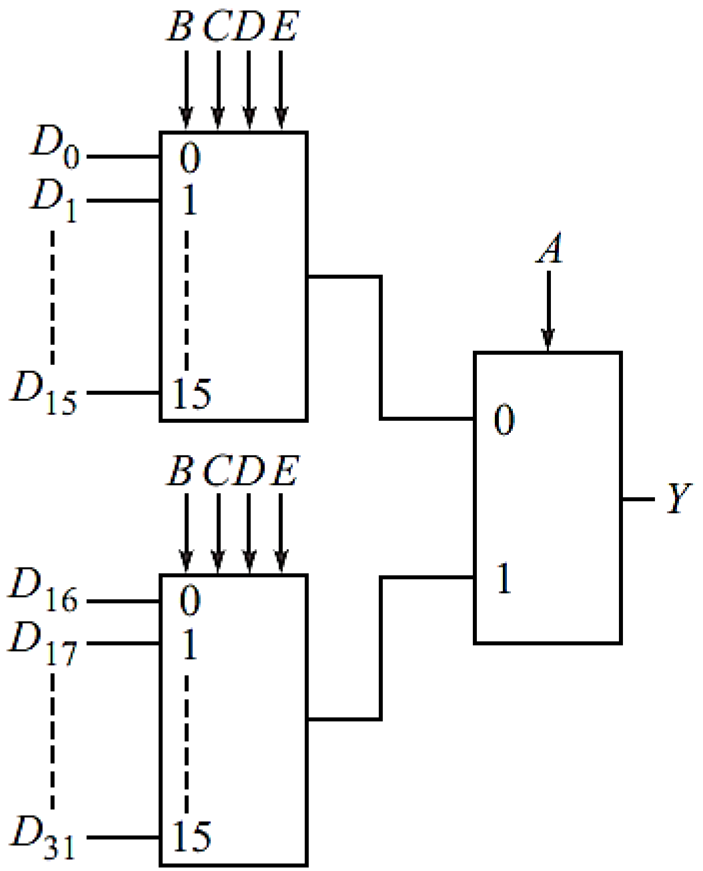




Higher order MUX from lower order



8-to-1 MUX using only 2-to-1 MUX

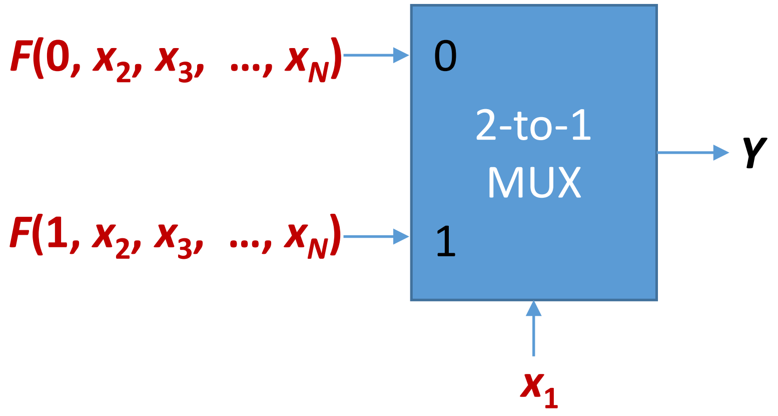


32-to-1 MUX from 16-to-1 MUX and 2-to-1 MUX

Shanon's Expansion Theorem and MUX

Shanon's Expansion Theorem: (inherent If-Then-Else)

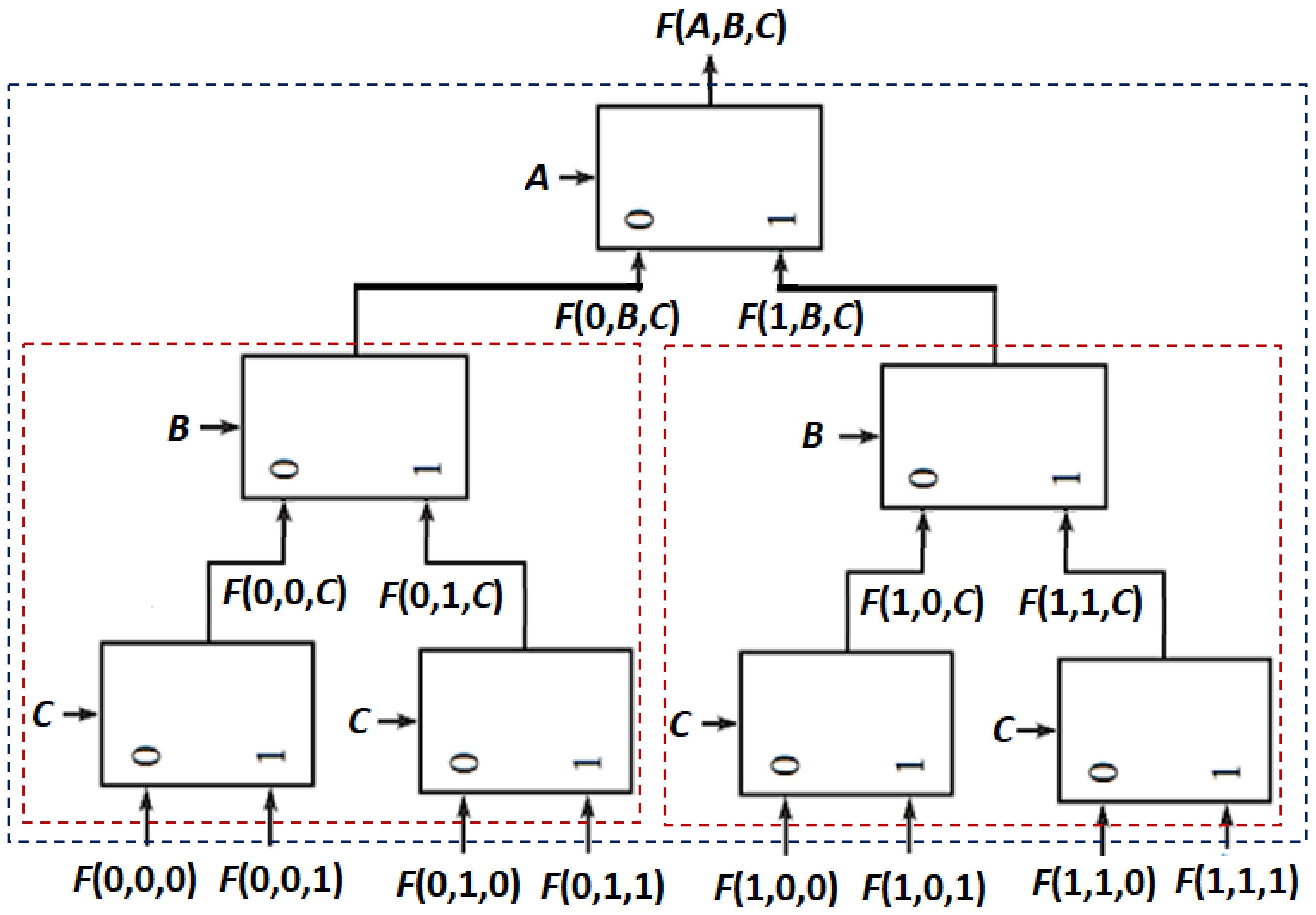
$$F(x_1, x_2, x_3, \ldots, x_N) = x_1'.F(0, x_2, x_3, \ldots, x_N) + x_1.F(1, x_2, x_3, \ldots, x_N)$$



$$\begin{aligned} Y &= S_1'S_0'D_0 + S_1'S_0D_1 + S_1S_0'D_2 + S_1S_0D_3 \\ &= S_1'.[0'.S_0'D_0 + 0'.S_0D_1 + 0.S_0'D_2 + 0.S_0D_3] + S_1.[1'.S_0'D_0 + 1'.S_0D_1 + 1.S_0'D_2 + 1.S_0D_3] \\ &= S_1'.(S_0'D_0 + S_0D_1) + S_1.(S_0'D_2 + S_0D_3) \end{aligned}$$

$$F(x_1, x_2, x_3, \ldots, x_N) = x_1'.[x_2'.F(0, 0, x_3, \ldots, x_N) + x_2.F(0, 1, x_3, \ldots, x_N)] + x_1.[x_2'.F(1, 0, x_3, \ldots, x_N) + x_2.F(1, 1, x_3, \ldots, x_N)]$$

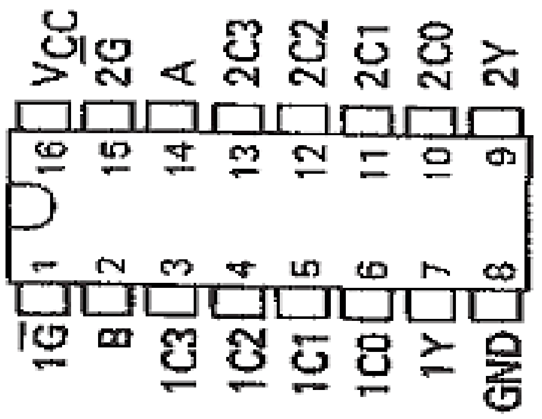
Shanon's Expansion Theorem and MUX



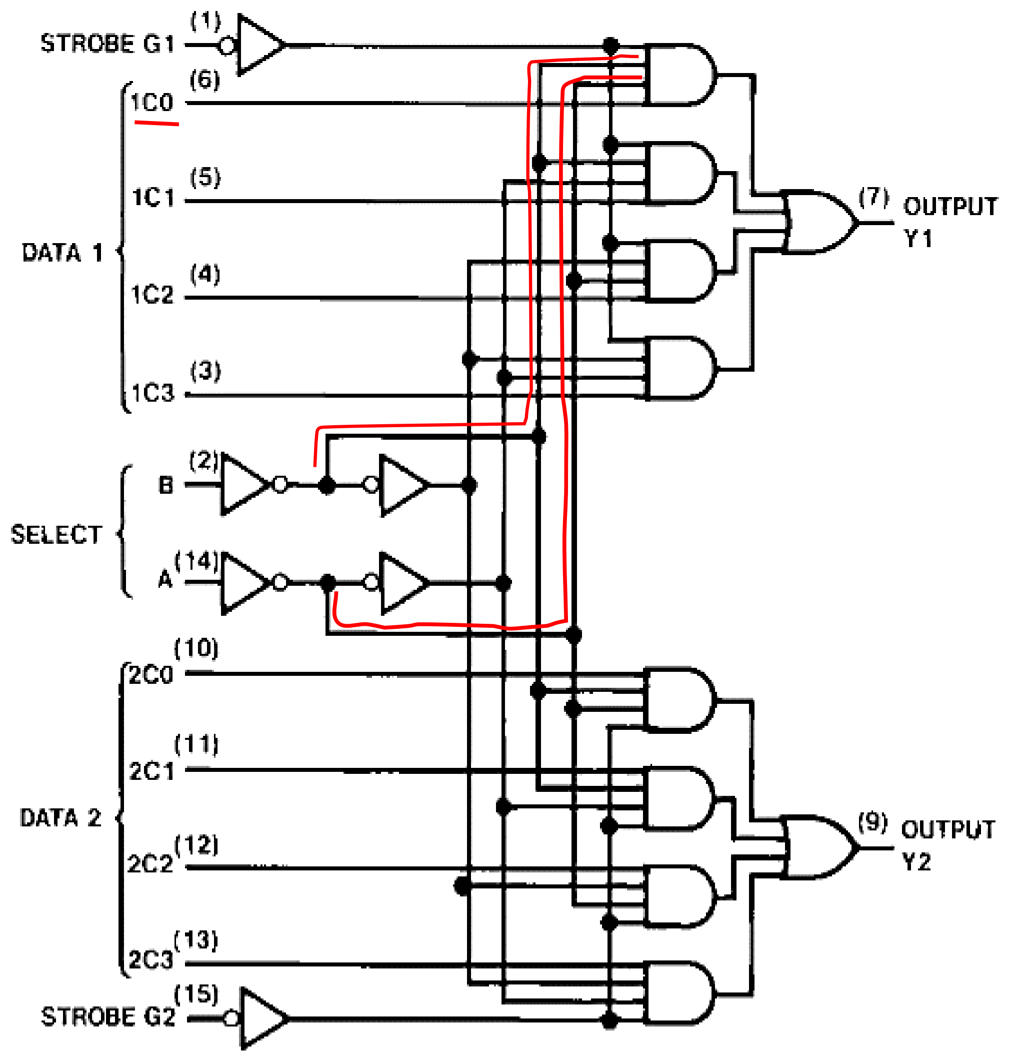
$$F(A,B,C) = A'B'C'.D_0 + A'B'C.D_1 + A'BC'.D_2 + A'BC.D_3 + AB'C'.D_4 + AB'C.D_5 + ABC'.D_6 + ABC.D_7$$

A	B	C	F(A,B,C)	
0	0	0	1	$D_0 = 1$
0	0	1	0	$D_1 = 0$
0	1	0	0	$D_2 = 0$
0	1	1	1	$D_3 = 1$
…	…	…	…	

IC 74153



IC 74153



STROBE (G)	B	A	Y
H	X	X	L
L	L	L	C0
L	L	H	C1
L	H	L	C2
L	H	H	C3

$$Y = E'.[(B'.A').C0 + (B'.A).C1 + (B.A').C2 + (B.A).C3]$$

References:

❑ Donald P. Leach, Albert P. Malvino, and Goutam Saha, Digital Principles & Applications 8e, McGraw Hill