Exercise 9.1

Not yet available.

Exercise 9.2

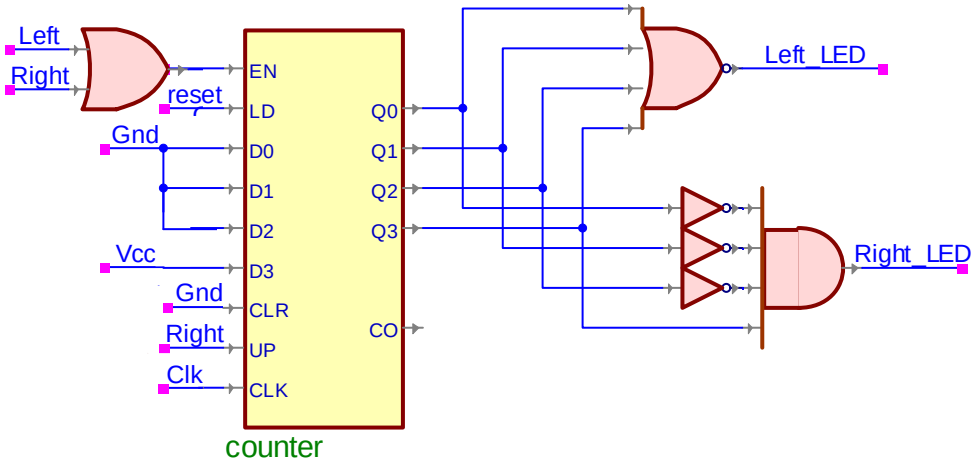
Not yet available.

Exercise 9.3

Not yet available.

Exercise 9.4

Using the counter from Figure 9.5, we can easily construct this left-right LED design. When the system is reset, we need to start out in state nine. This can be accomplished by hard-wiring the load inputs of the counter to 4, which corresponds to the middle of nine states from 0 to 8, and tying the load input to reset. If we assume that either the left or right input will be asserted at once, never at the same time (or if they are, right will take precedence), then we can tie the right signal to the up input of the counter so that when right is asserted it will advance to the next higher state. ORing the left and right inputs and using that as the enable signal of the counter will ensure that the counter only changes when right or left is asserted. Finally, two gates use the outputs of the counter to set the Left_LED and Right_LED as well as disabling the counter from advancing further. The resulting circuit schematic looks as follows:



Exercise 9.5

Not yet available.

Exercise 9.6

Not yet available.

Exercise 9.7

Not yet available.

Exercise 9.8

Not yet available.

Exercise 9.9

Not yet available.

Exercise 9.10

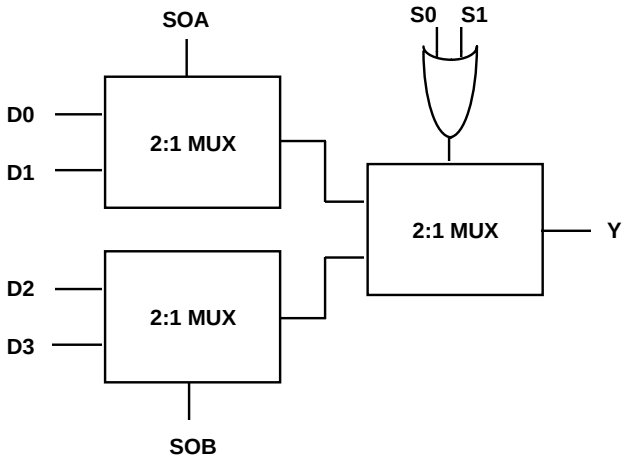
Not yet available.

Exercise 9.11

Not yet available.

Exercise 9.12

Recall that the block diagram of the Actel combinational logic C-cell looks as follows:



(a) Y = AB

D0	D1	SOA	D2	D3	SOB	S0	S1
0	A	B	-	-	-	0	0

(b) Y = (AB)'

D0	D1	SOA	D2	D3	SOB	S0	S1
1	-	0	1	0	B	A	0

(c) Y = (A + B)'

D0	D1	SOA	D2	D3	SOB	S0	S1
1	0	B	0	-	0	A	0

(d) Y = A xor B

D0	D1	SOA	D2	D3	SOB	S0	S1
0	1	B	1	0	B	A	0

(e) Y = AB + C

D0	D1	SOA	D2	D3	SOB	S0	S1
0	A	B	1	-	0	C	0

(f) (A + B) + C

D0	D1	SOA	D2	D3	SOB	S0	S1
0	-	0	B	1	A	C	0

(g) AB + AC + BC

D0	D1	SOA	D2	D3	SOB	S0	S1
0	C	B	C	1	B	A	0

Exercise 9.13

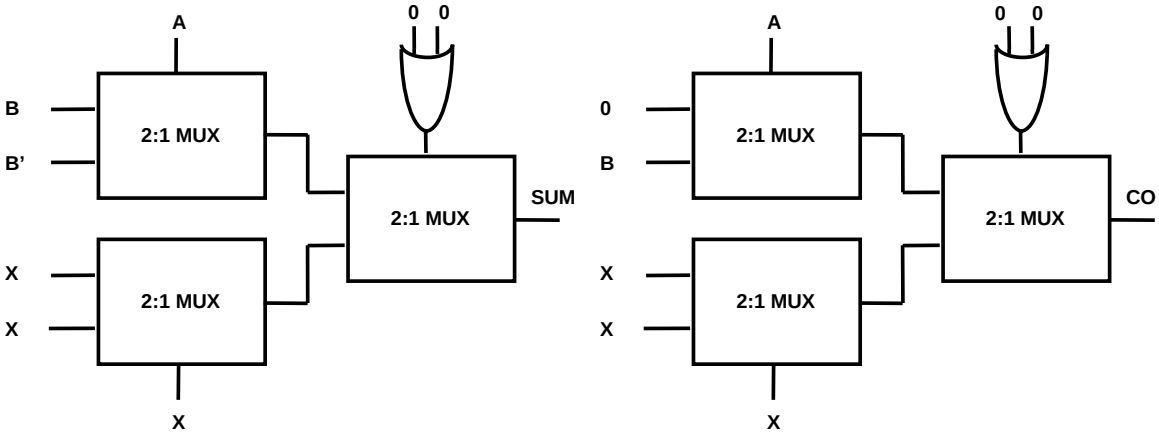
Not yet available.

Exercise 9.14

Half adder Actel Logic Module:

SUM = A'B + AB'

CO = AB

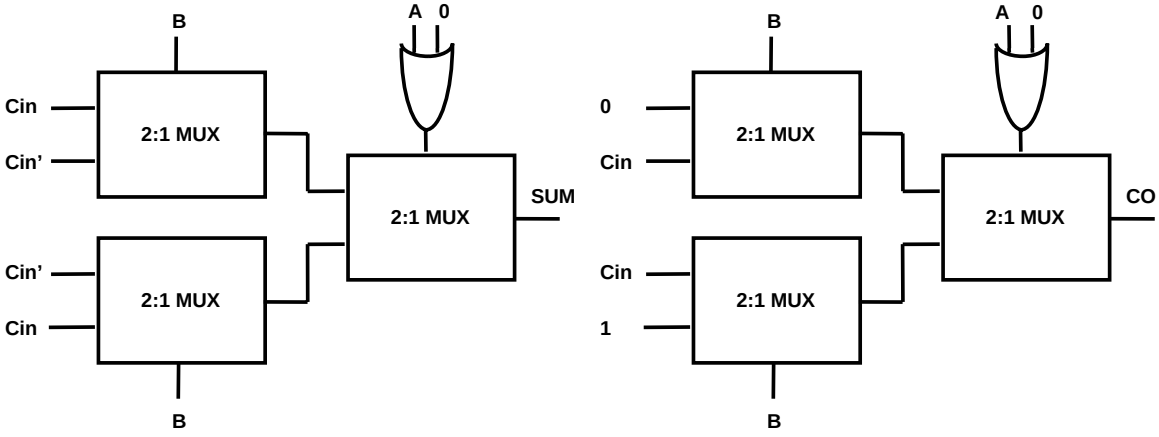


Half Adder

(h) Full adder Actel Logic Module:

SUM = ABCin + A'B'Cin + AB'Cin' + A'BCin'

CO = ABCin + A'BCin + AB'Cin + ABCin' = AB + A'BCin + AB'Cin



Exercise 9.15

Not yet available.

Exercise 9.16

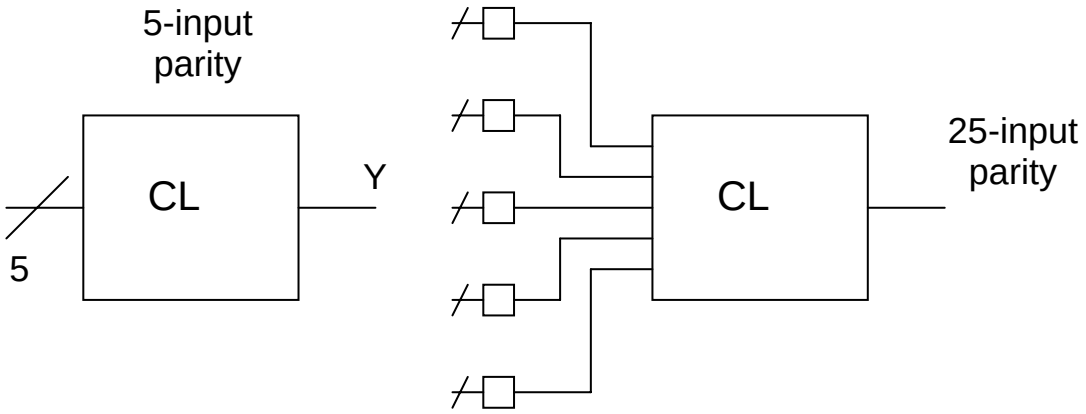
Not yet available.

Exercise 9.17

Not yet available.

Exercise 9.18

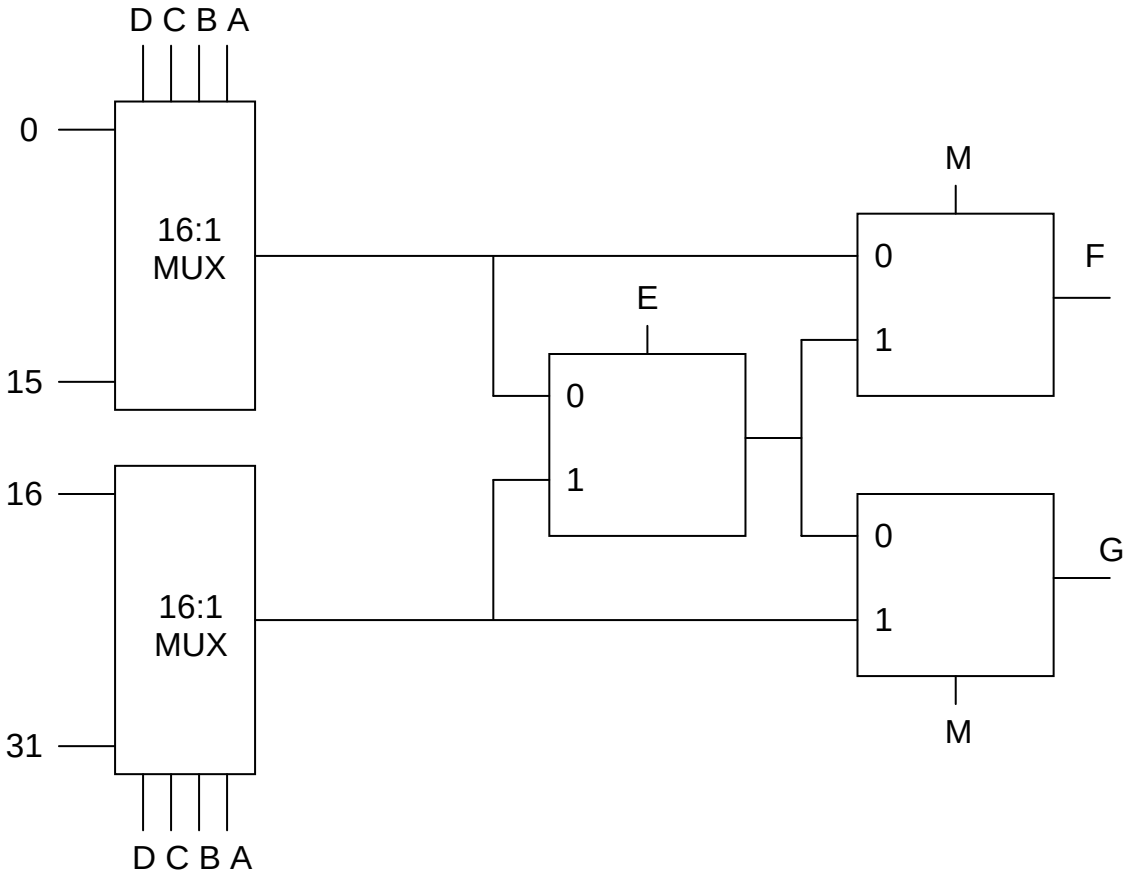
(a) 25-input parity function in 2 level CLB structure:



(b) A 3 level CLB structure can implement a $5^3 = 125$ input parity function.

Exercise 9.19

(a)



(b) F = A xor B xor C xor D xor E

ECDBA	F	ECDBA	F
00000	0	10000	1
00001	1	10001	0
00010	1	10010	0
00011	0	10011	1
00100	1	10100	0
00101	0	10101	1
00110	0	10110	1
00111	1	10111	0
01000	1	11000	0
01001	0	11001	1
01010	0	11010	1
01011	1	11011	0
01100	0	11100	1
01101	1	11101	0
01110	1	11110	0
01111	0	11111	1

(c) F(A,B,C) = A xor B xor C

G(A,B,C) = AB + BC + AC

ABC	AB	BC	AC	F	G
000	0	0	0	0	0
001	0	0	0	1	0
010	0	0	0	1	0
011	0	1	0	0	1
100	0	0	0	1	0
101	0	0	1	0	1
110	1	0	0	0	1
111	1	1	1	1	1

Exercise 9.20

Not yet available.

Exercise 9.21

Not yet available.

Exercise 9.22

Not yet available.

Exercise 9.23

Not yet available.

Exercise 9.24

Not yet available.