Modern Digital System Design Final Project

Tyler Bowen, Matt Pope, Steven Mathew

Input Select Module

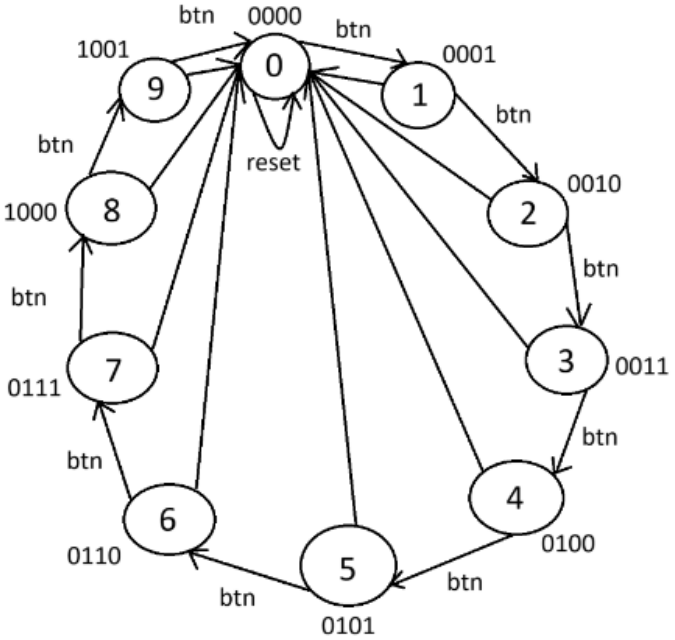
Inputs

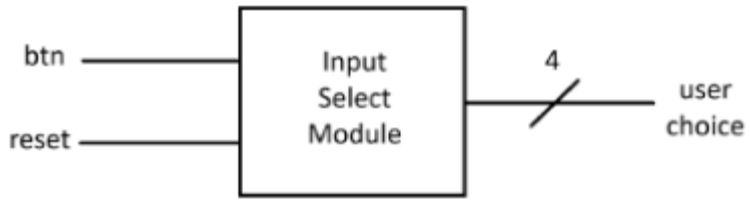
btn -> increment button

reset -> reset button

Outputs

user_choice -> 4 bit user choice





Full Adder Module

Truth Table

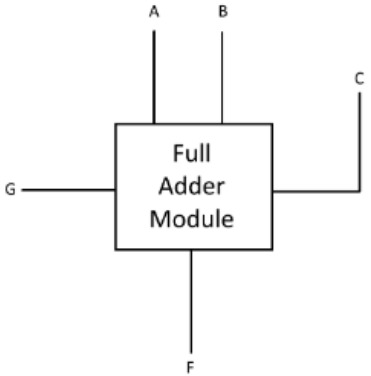
A	B	C	F	G
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

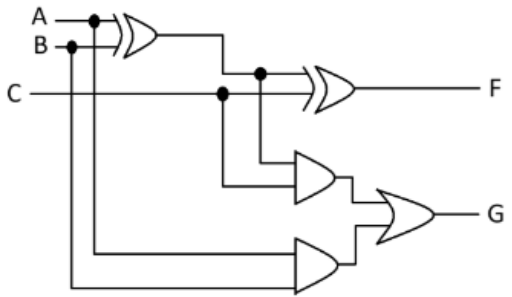
Inputs

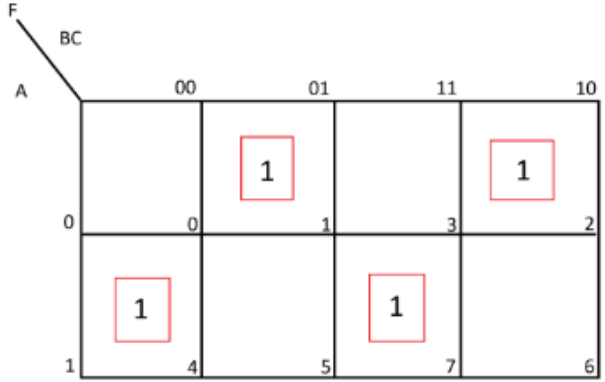
A -> input bit
B -> input bit
C -> carry in

Outputs

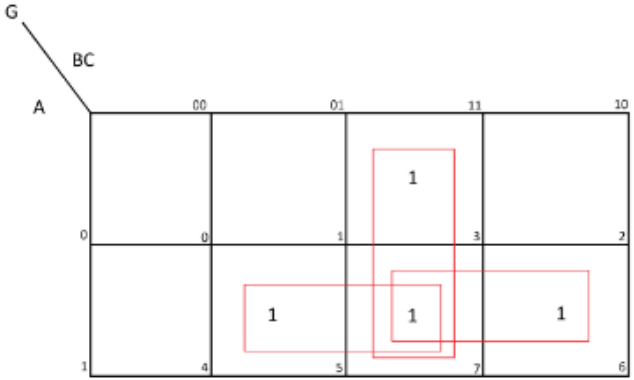
F -> sum
G -> carry out







F = ∑m(1,2,4,7)
F = A'B'C + A'BC' + AB'C' + ABC
F is 1 when an odd number of inputs are 1.
XOR is 1 when an odd number of inputs are 1.
F = (A ^ B) ^ C
F = A ^ B ^ C

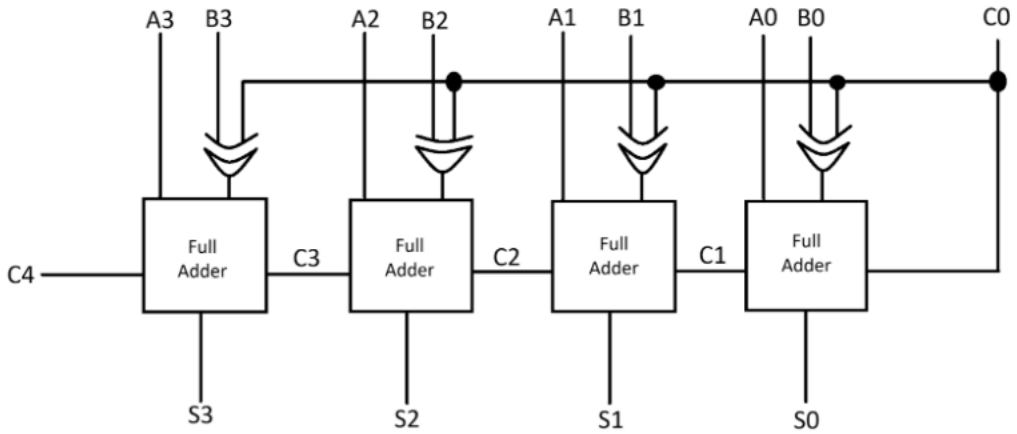


G = ∑m(3, 5, 6, 7)
G = -11 + 1-1 + 11-
G = BC + AC + AB

Ripple Carry Adder Subtractor Module

Inputs
A -> 4 bits
B -> 4 bits
C0 -> carry in (mode select)

Outputs
S -> 4 bit sum or difference
C4 -> carry out

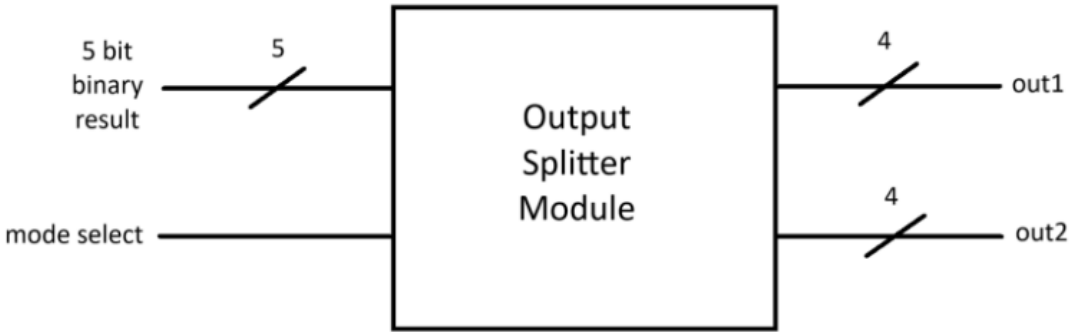


This module takes two 4 bit inputs A and B and a carry in C0 that functions as the mode selector. If the carry in is 0, A and B are added. If the carry in is 1, subtraction is performed by inverting the bits of B and adding 1 from the carry in. The module has a 5 bit output. The sum S represents the lower 4 bits of the output. C4 represents the most significant 5th bit of the output.

Output Splitter Module

Inputs
bin -> 5 bit binary result from adder
mode_select -> mode select

Outputs
out1 -> 4 bits to decoder
out2 -> 4 bits to decoder



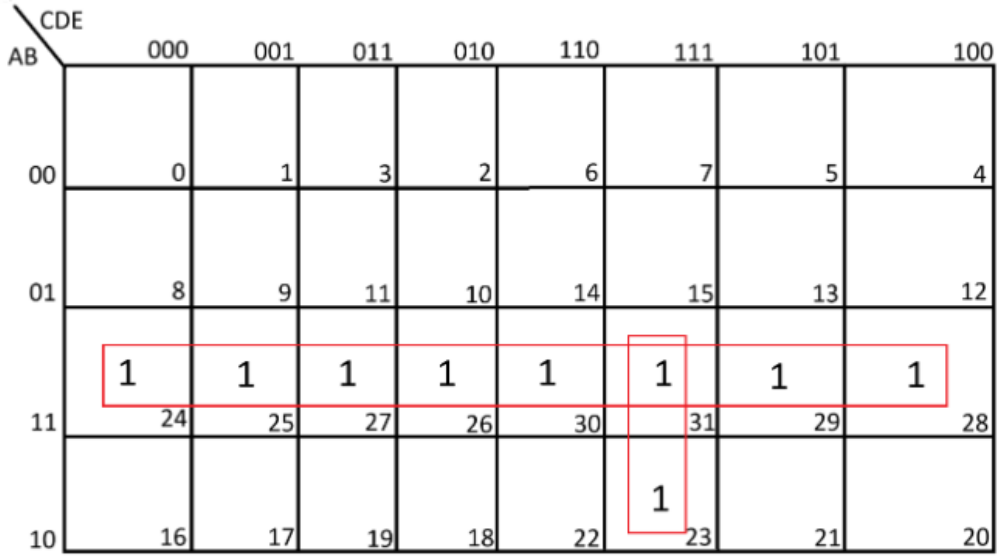
This module takes the 5 bit result from the adder subtractor and turns it into two 4 bit outputs using equations derived from solving K-maps. This module also takes the mode selector as an input. When mode select is 1 (subtraction), the module flips the most significant bit of the 5 bit result before interpreting it. This has to be done to ensure the proper output on the display.

Output Splitter Truth Table

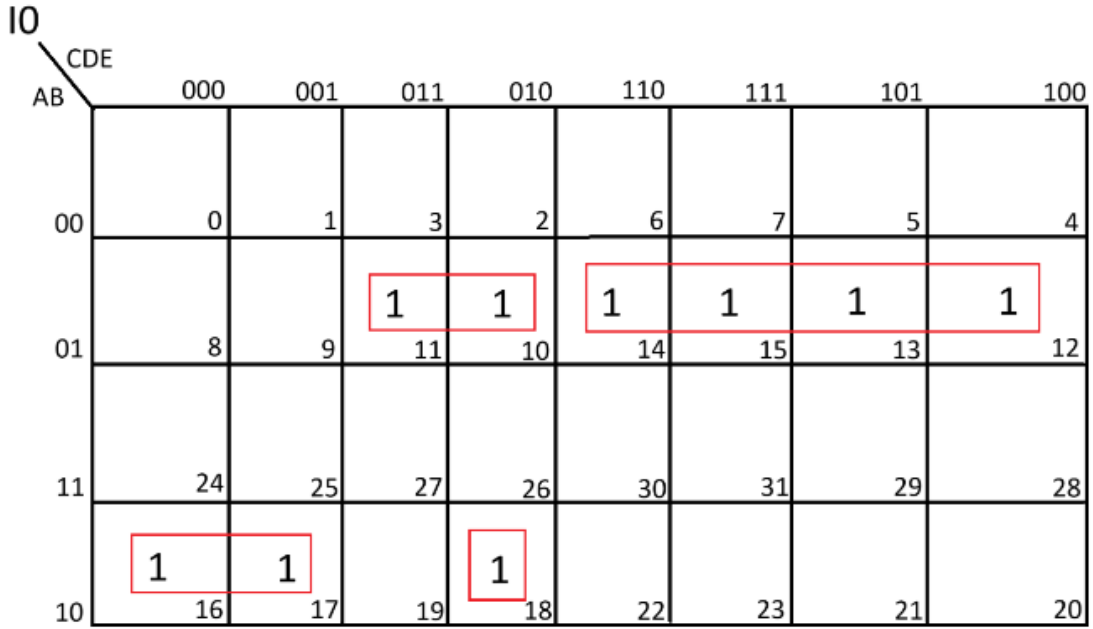
Decimal	A	B	C	D	E	I3	I2	I1	I0	J3	J2	J1	J0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	0	1
2	0	0	0	1	0	0	0	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	0	0	0	1	1
4	0	0	1	0	0	0	0	0	0	0	1	0	0
5	0	0	1	0	1	0	0	0	0	0	1	0	1
6	0	0	1	1	0	0	0	0	0	0	1	1	0
7	0	0	1	1	1	0	0	0	0	0	1	1	1
8	0	1	0	0	0	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	0	0	0	1	0	0	1
10	0	1	0	1	0	0	0	0	1	0	0	0	0
11	0	1	0	1	1	0	0	0	1	0	0	0	1
12	0	1	1	0	0	0	0	0	1	0	0	1	0
13	0	1	1	0	1	0	0	0	1	0	0	1	1
14	0	1	1	1	0	0	0	0	1	0	1	0	0
15	0	1	1	1	1	0	0	0	1	0	1	0	1
16	1	0	0	0	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	0	0	0	1	0	1	1	1
18	1	0	0	1	0	0	0	0	1	1	0	0	0

Decimal	A	B	C	D	E	I3	I2	I1	I0	J3	J2	J1	J0
-9	1	0	1	1	1	1	0	1	0	1	0	0	1
-8	1	1	0	0	0	1	0	1	0	1	0	0	0
-7	1	1	0	0	1	1	0	1	0	0	1	1	1
-6	1	1	0	1	0	1	0	1	0	0	1	1	0
-5	1	1	0	1	1	1	0	1	0	0	1	0	1
-4	1	1	1	0	0	1	0	1	0	0	1	0	0
-3	1	1	1	0	1	1	0	1	0	0	0	1	1
-2	1	1	1	1	0	1	0	1	0	0	0	1	0
-1	1	1	1	1	1	1	0	1	0	0	0	0	1

I3 & I1



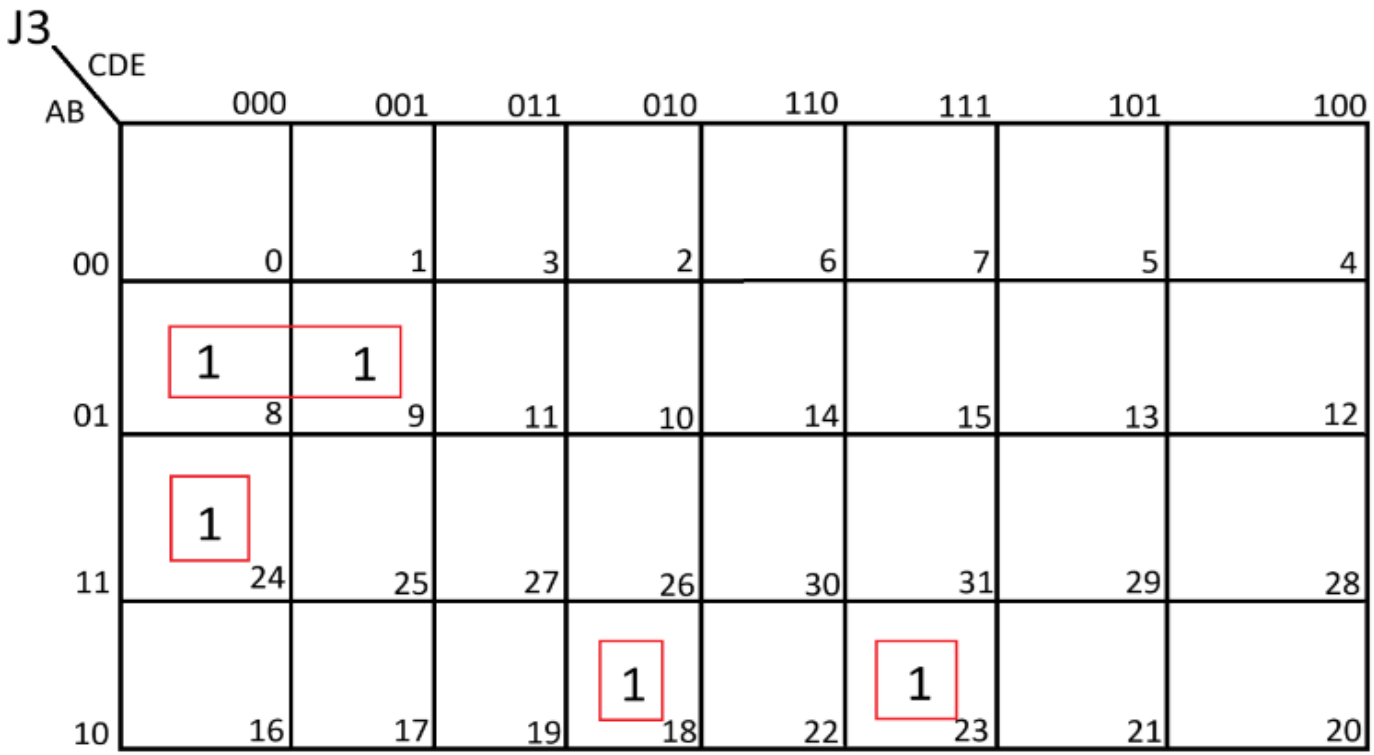
I3 = ∑m(23,24,25,26,27,28,29,30,31)
I3 = 11 - - - + 1 - 111
I3 = AB + ACDE
I1 = AB + ACDE (I3 and I1 are identical)
I2 = 0 (I2 is always equal to 0 so does not require a K map)



I0 = ∑m(10,11,12,13,14,15,16,17,18)

I0 = 0101 - + 011 - - + 10000 - + 10010

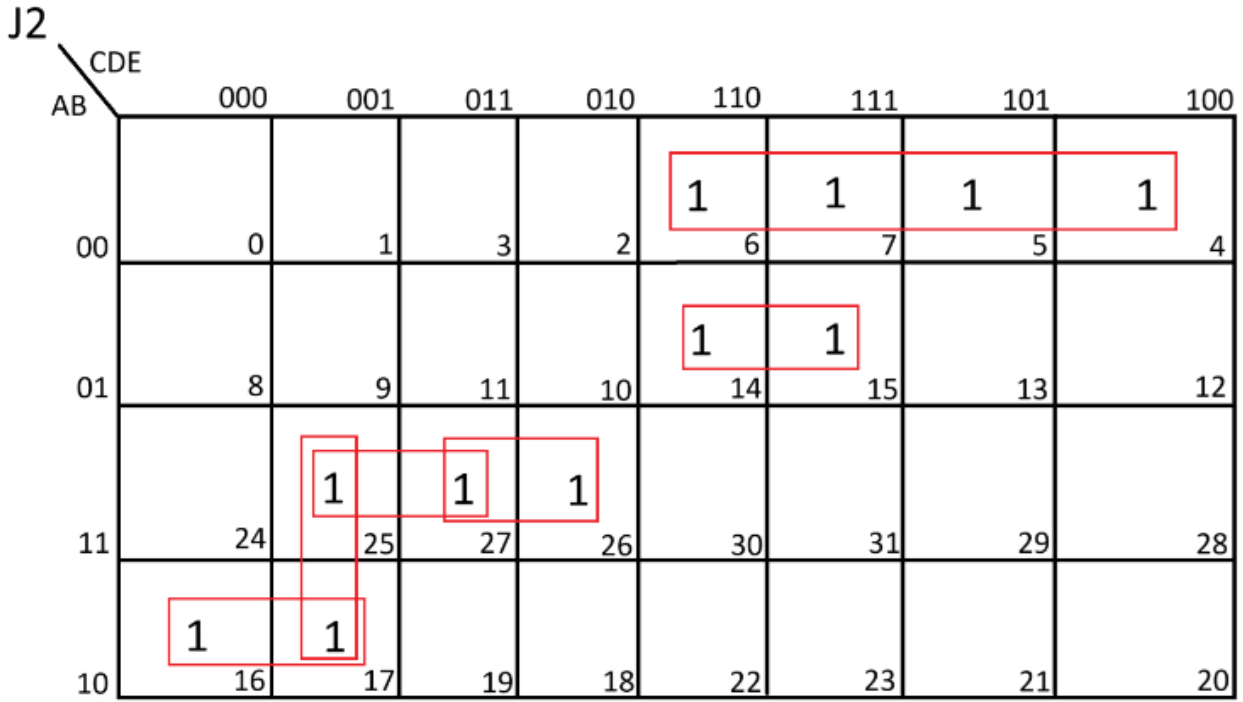
I0 = A'BC'D + A'BC + AB'C'D' + AB'C'DE'



J3 = ∑m(8,9,18,23,24)

J3 = 0100 - + 11000 + 10010 + 10111

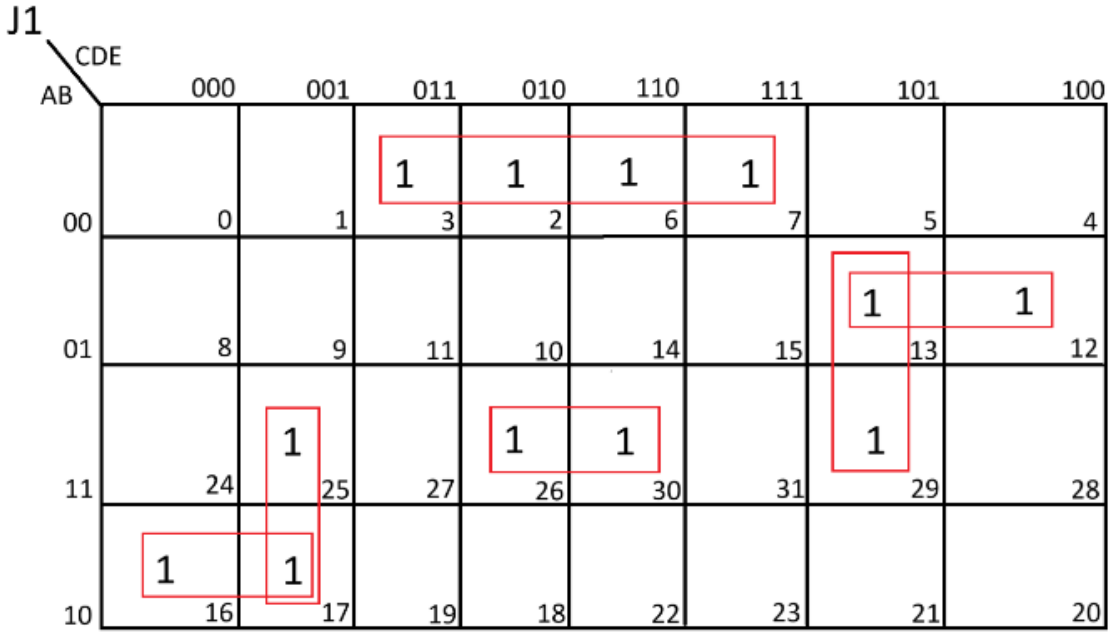
J3 = A'BC'D' + ABC'D'E' + AB'C'DE' + AB'CDE



J2 = ∑m(4,5,6,714,15,16,17,25,26,27,28)

J2 = 1000 - + 1 - 001 + 110 - 1 + 1101 - + 0111 - + 001 - -

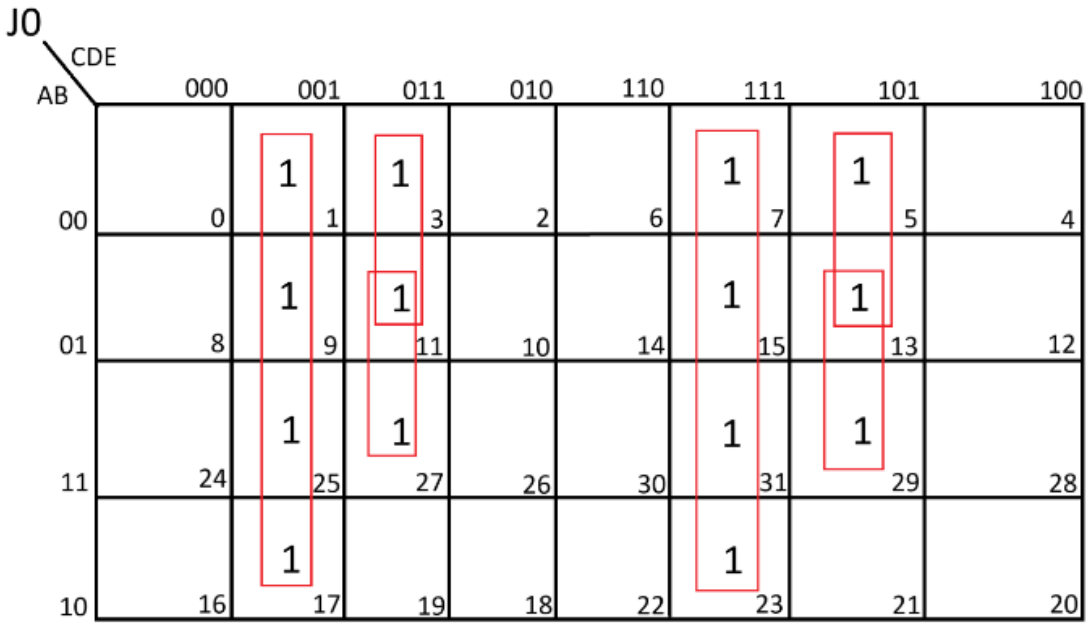
J2 = AB'C'D' + AC'D'E + ABC'E + ABC'D + A'BCD + A'B'C



$J1 = \sum m(2,3,6,7,12,13,16,17,25,26,29,30)$

J1 = 00 - 1 - + 0110 - + - 1101 + 11 - 10 + 1 - 001 + 1000 -

$J1 = A'B'D + A'BCD' + BCD'E + ABDE' + AC'D'E + AB'C'D'$



$J0 = \sum m(1,3,5,7,9,11,13,15,17,23,25,27,29,31)$

$J0 = - - 001 + 0 - 011 + - 1011 + - - 111 + 0 - 101 + - 1101$

$J0 = C'D'E + A'C'DE + BC'DE + CDE + A'CD'E + BCD'E$

2 to 1 Multiplexer Module

Inputs

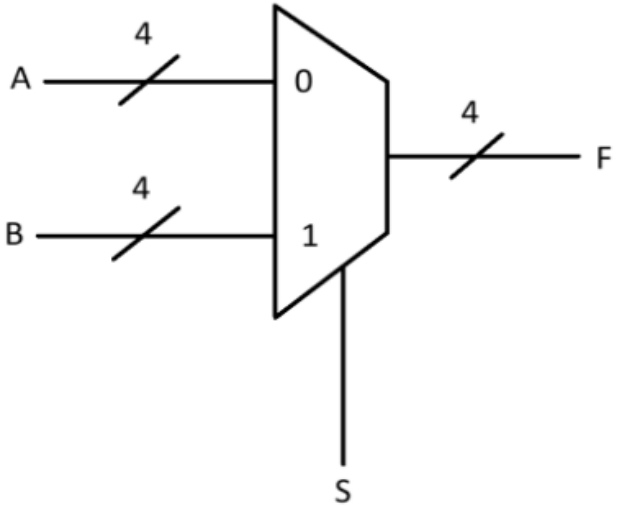
A -> 4 bit binary

B -> 4 bit binary

S -> Selector

Outputs

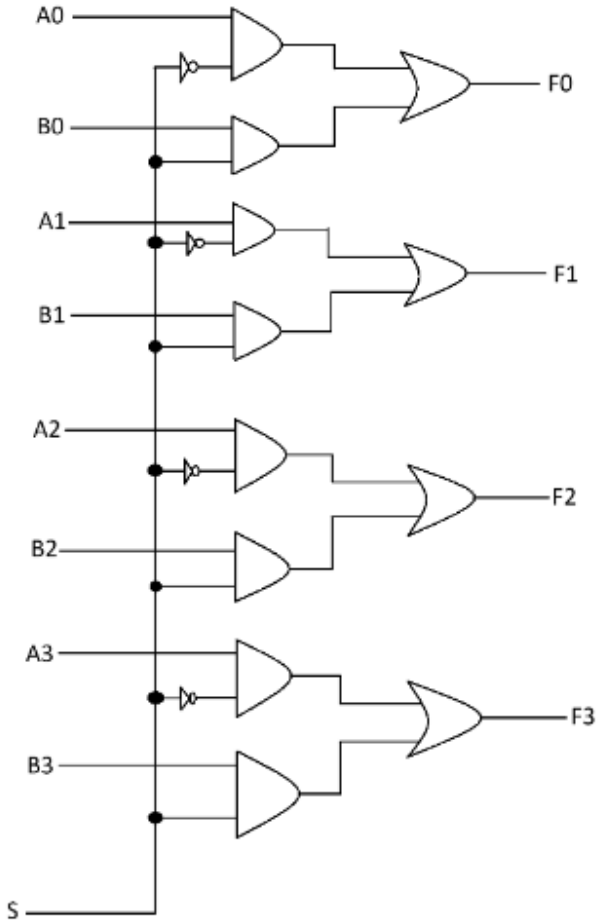
F -> 4 bit binary



F = S ? B : A;

If S is 1, select B. Otherwise, select A.

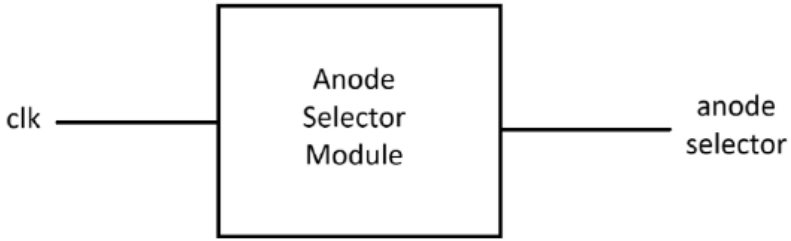
S	F
0	A
1	B


A0
B0
A1
B1
A2
B2
A3
B3
S
F0
F1
F2
F3

Anode Selector Module

Inputs
clock

Outputs
anode selector



Clock (rising edge)	Anode Selector (current)	Anode Selector (next)
0 -> 1	0	1
0 -> 1	1	0

4 Bit 7 Segment Decoder Module

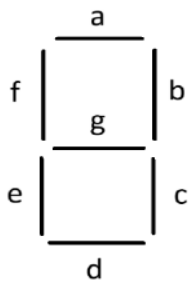
Inputs

bin -> 4 bit binary input

Outputs

seg7 -> 7 bit LED cathode outputs, active high assumes common anode LEDs

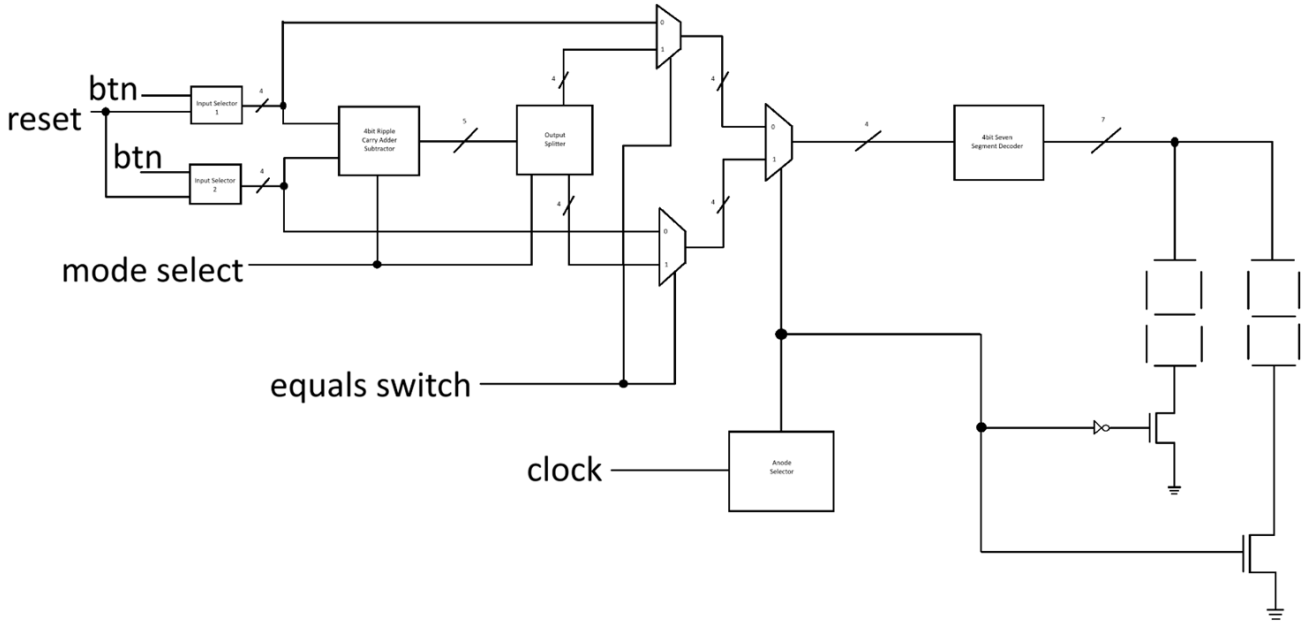
A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	X	X	X	X	X	X	X
1	0	1	1	X	X	X	X	X	X	X
1	1	0	0	X	X	X	X	X	X	X
1	1	0	1	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X

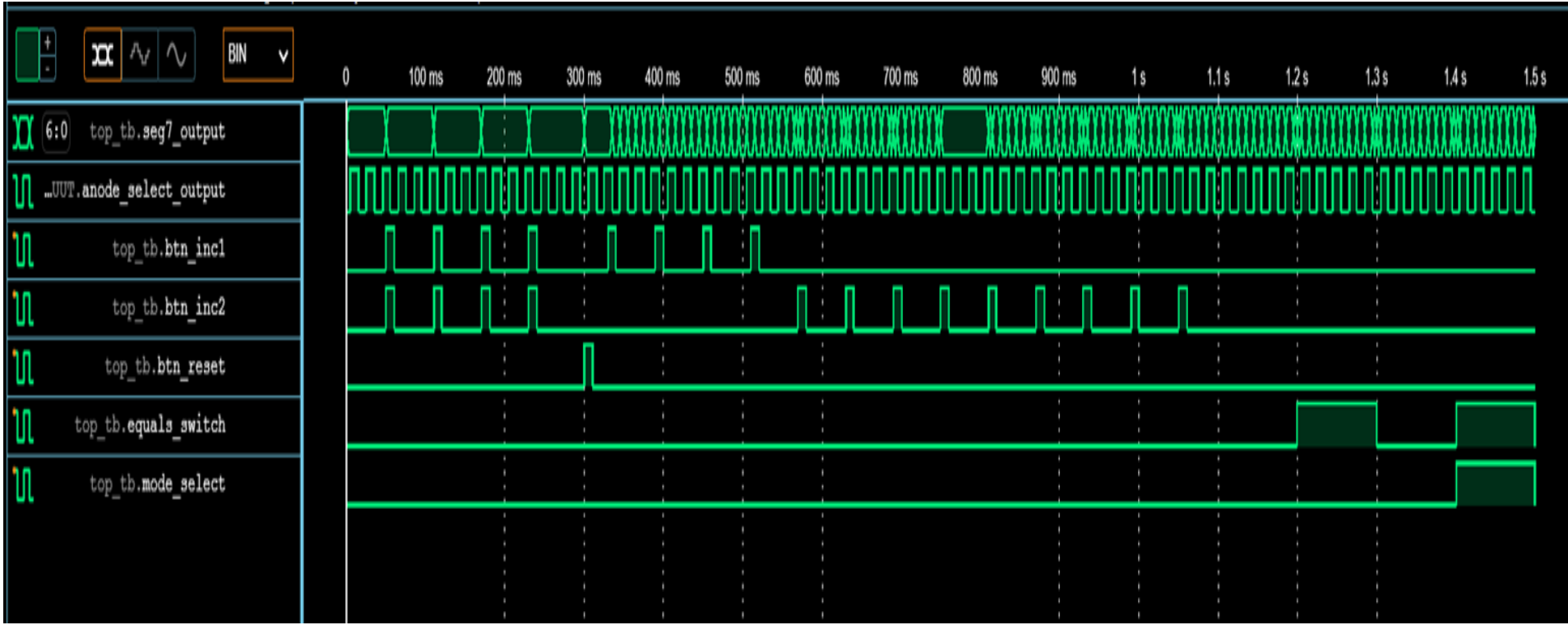


Top Module

Inputs
btn1
btn2
reset
mode selector
equals switch
clock
anode selector

Outputs
seg7




BIN
0
100 ms
200 ms
300 ms
400 ms
500 ms
600 ms
700 ms
800 ms
900 ms
1 s
1.1 s
1.2 s
1.3 s
1.4 s
1.5 s
6:0
top_tb.seg7_output
…UUT.anode_select_output
top_tb.btn_inc1
top_tb.btn_inc2
top_tb.btn_reset
top_tb.equals_switch
top_tb.mode_select