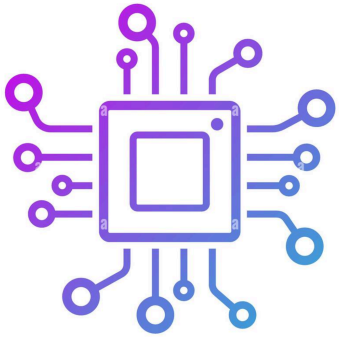


CPU Design Progression Log

Objective:

Create a 4 bits adder/subtractor with your assigned number base

Your number base = Last TWO digits of your => student number mod 7 + 3

E.g student #: 12345678

Input Display

7 mod 7 + 3 = 3

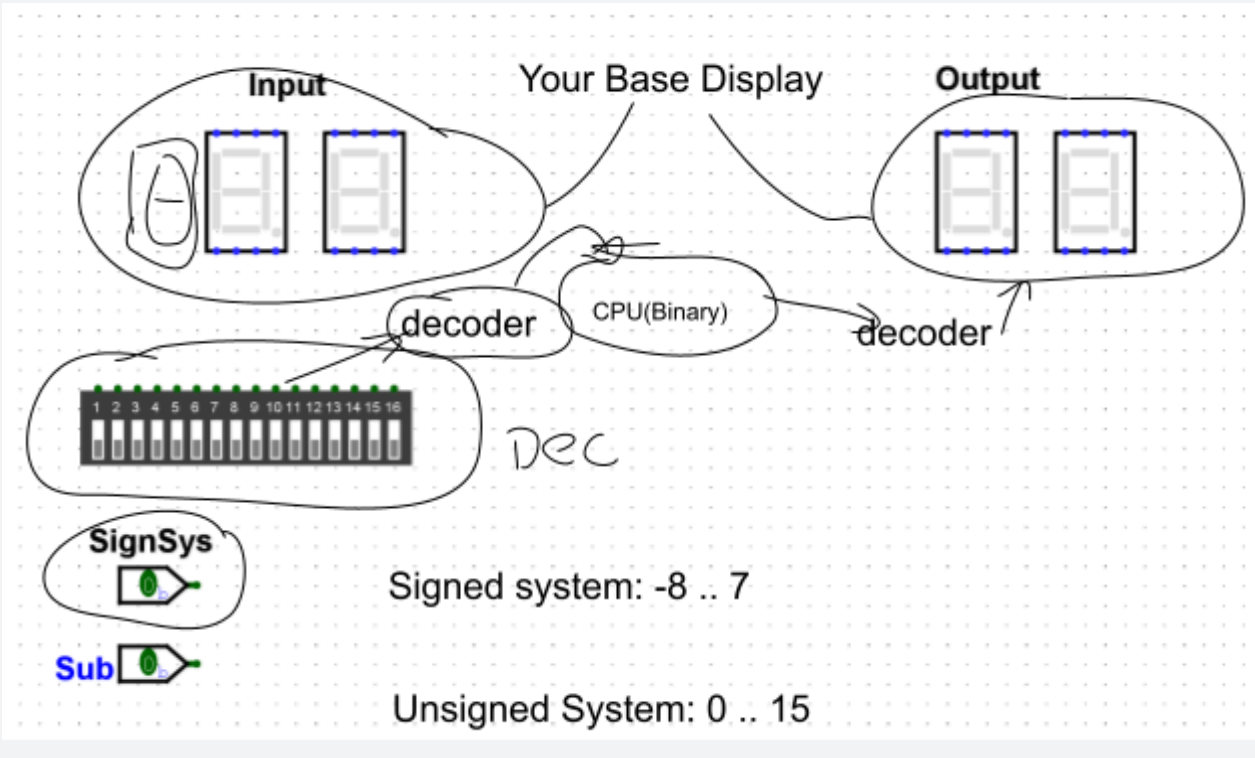
Output Display

8 mod 7 +3 = 1 +3 = 4

Student will have their number system between 3..9

Checklist:

- **MY INPUT NUMBER SYSTEM:** 1 + 3 = 4
- **MY OUTPUT NUMBER SYSTEM:** 1 + 3 = 4
- Both input and output will be display on 7 Segment display devices with respect to their base system
- Design would have a switch to enter into a signed or unsigned system.
    - Signed system: A system which includes both (+) and (-) numbers, thus, in a 4 bits system, the number would be from -8..7.
    - Unsigned System: A system which includes only (+), thus, in a 4 bits system, the number would be from 0..15.



*** Student must submit work evidence (e.g. K-Map, Boolean Simplifications, Truth Table) for designing the MUX, and Decoder under this document**

Put all your evidence here and organize them properly

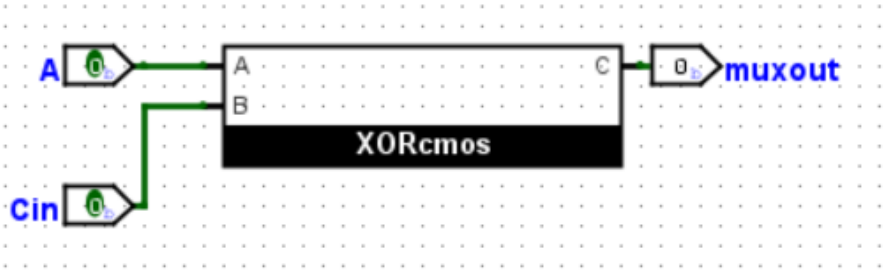
MUX Design

1. Adder and Subtractor MUX (insert its working log here(truthtable, kmap and etc) and logisim design)

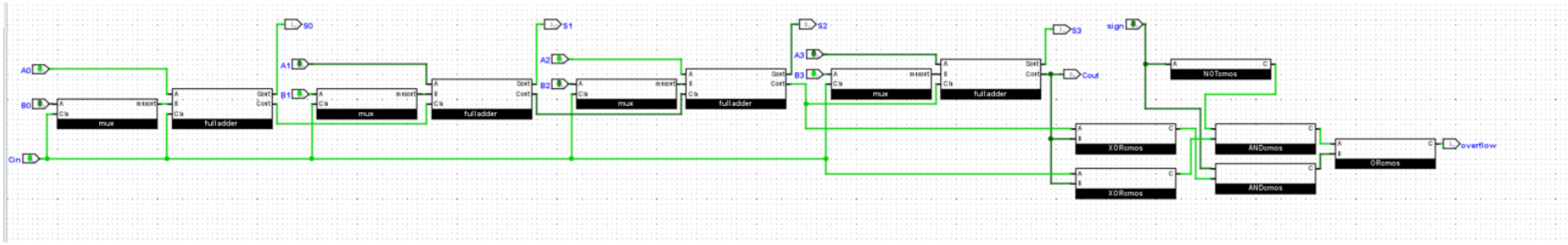
   Truth Table:

A	not A	Cin (0 = add, 1 = sub)	MUX Output
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0

   Reduce 3 inputs into 2 input; MUX Output = A XOR Cin



   Four bit adder/subtractor with signed/unsigned system and overflow:



Decoder Design

1. Input Decoder

SIGNED:

Convert decimal to binary (mapping): 16 inputs (-8 to 7) map to 4 outputs (4 bits of binary number)

Decimal	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
Input Letter	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Binary (4 bit 2's comp)	1000	1001	1010	1011	1100	1101	1110	1111	0000	0001	0010	0011	0100	0101	0110	0111

Boolean equations for each bit of binary number:

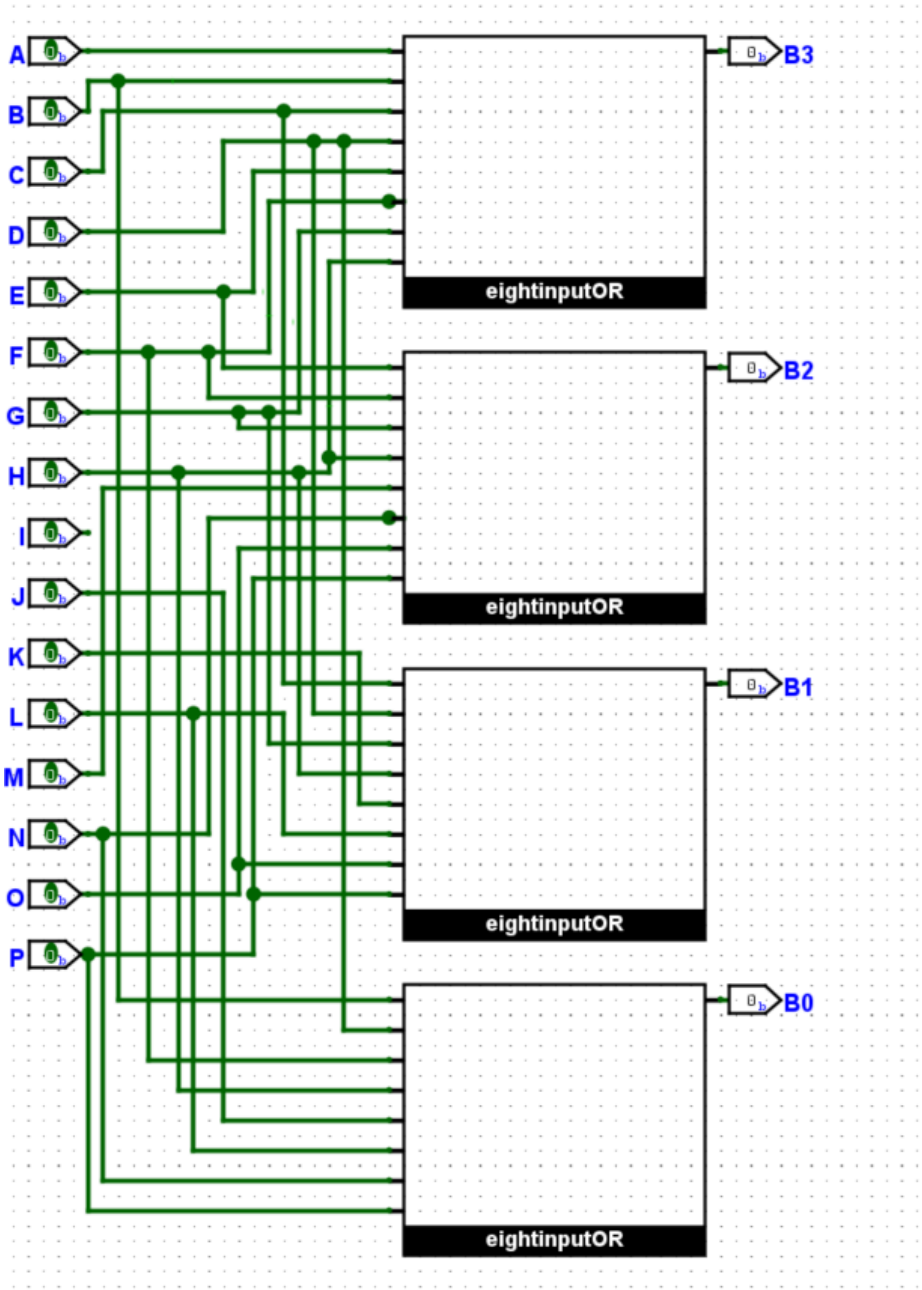
B3 = MSB, B0 = LSB

$B_3 = A + B + C + D + E + F + G + H$

$B_2 = E + F + G + H + M + N + O + P$

$B_1 = C + D + G + H + K + L + O + P$

$B_0 = B + D + F + H + J + L + N + P$


A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
B3
B2
B1
B0
eightinputOR
eightinputOR
eightinputOR
eightinputOR

UNSIGNED:

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Input Letter	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Binary (4 bit 2's comp)	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

Boolean equations for each bit of binary number:

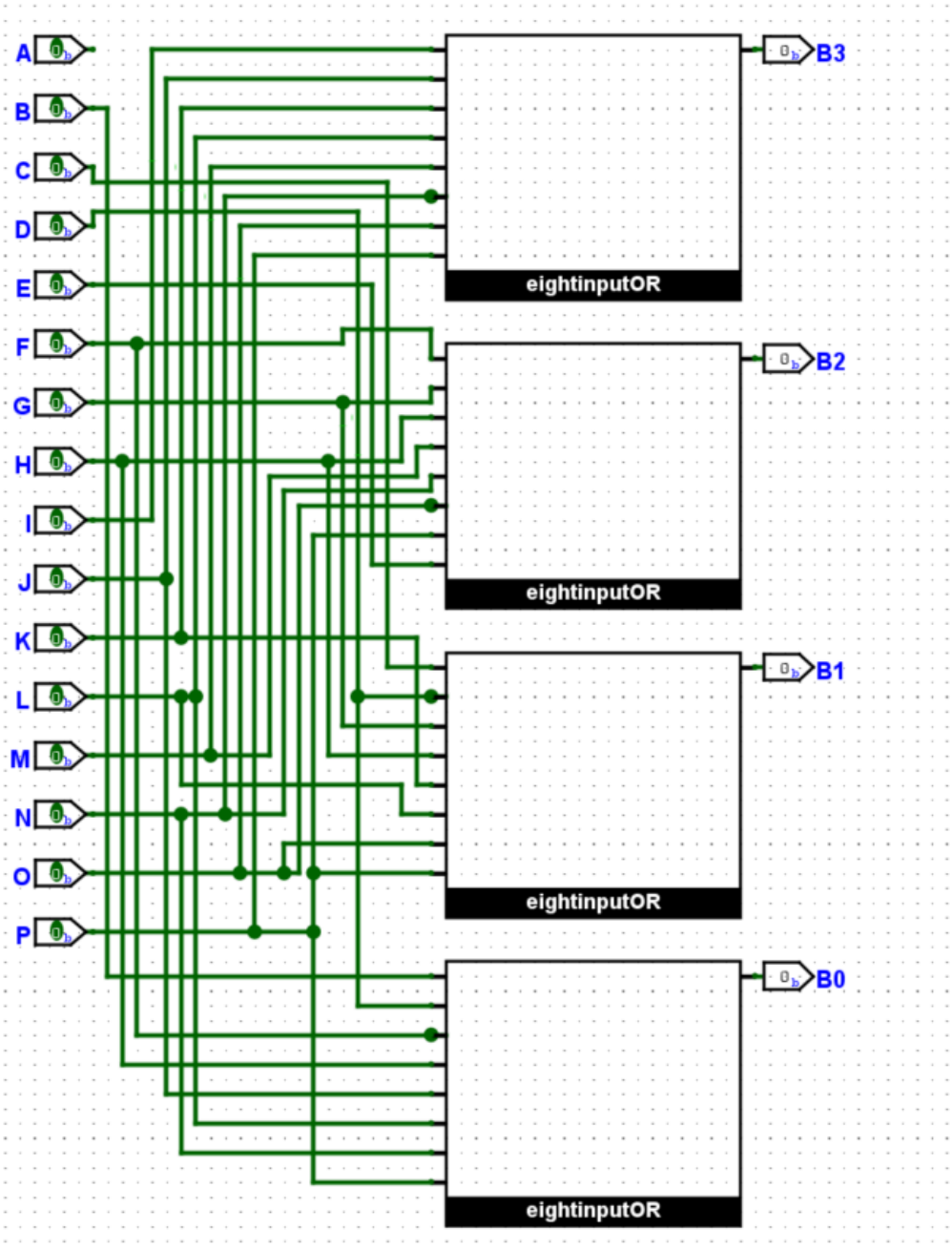
B3 = MSB, B0 = LSB

$B_3 = I + J + K + L + M + N + O + P$

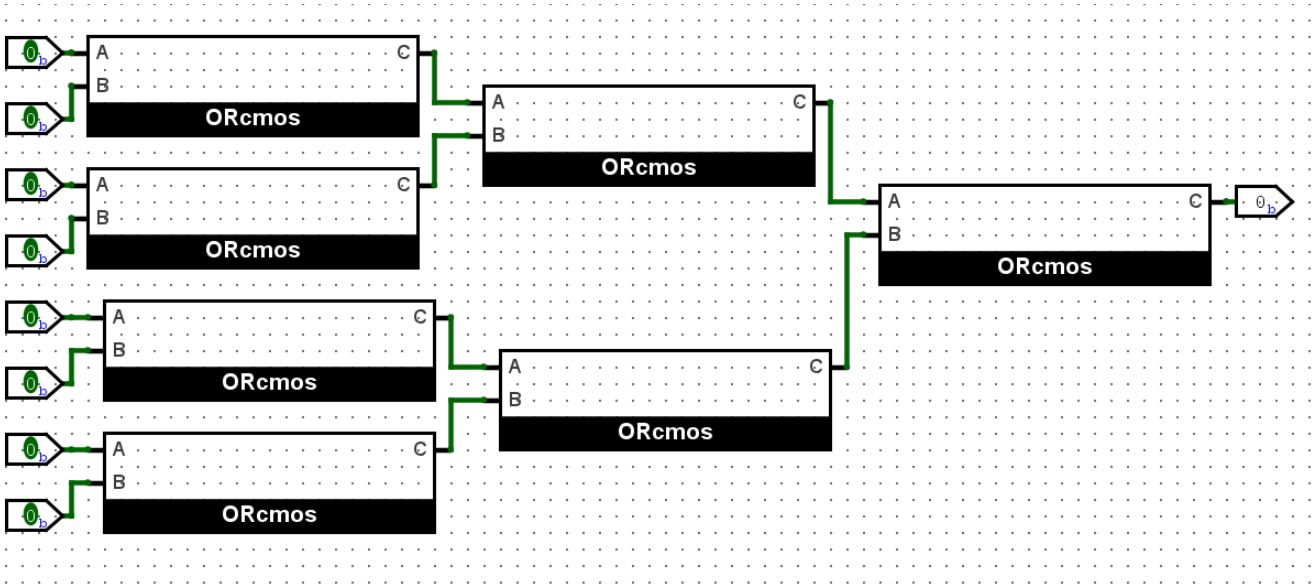
$B_2 = E + F + G + H + M + N + O + P$

$B_1 = C + D + G + H + K + L + O + P$

$B_0 = B + D + F + H + J + L + N + P$


A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
eightinputOR
B3
eightinputOR
B2
eightinputOR
B1
eightinputOR
B0

Custom 8-input OR Gate (because every boolean equation is 8 ORs):



4-bit MUX

Toggle between the signed vs unsigned value based on the sign value

Inputs: A3, A2, A1, A0 (unsigned values), B3, B2, B1, B0 (signed values), sign=0 for unsigned, sign=1 for signed

Output: C3, C2, C1, C0 → correct value

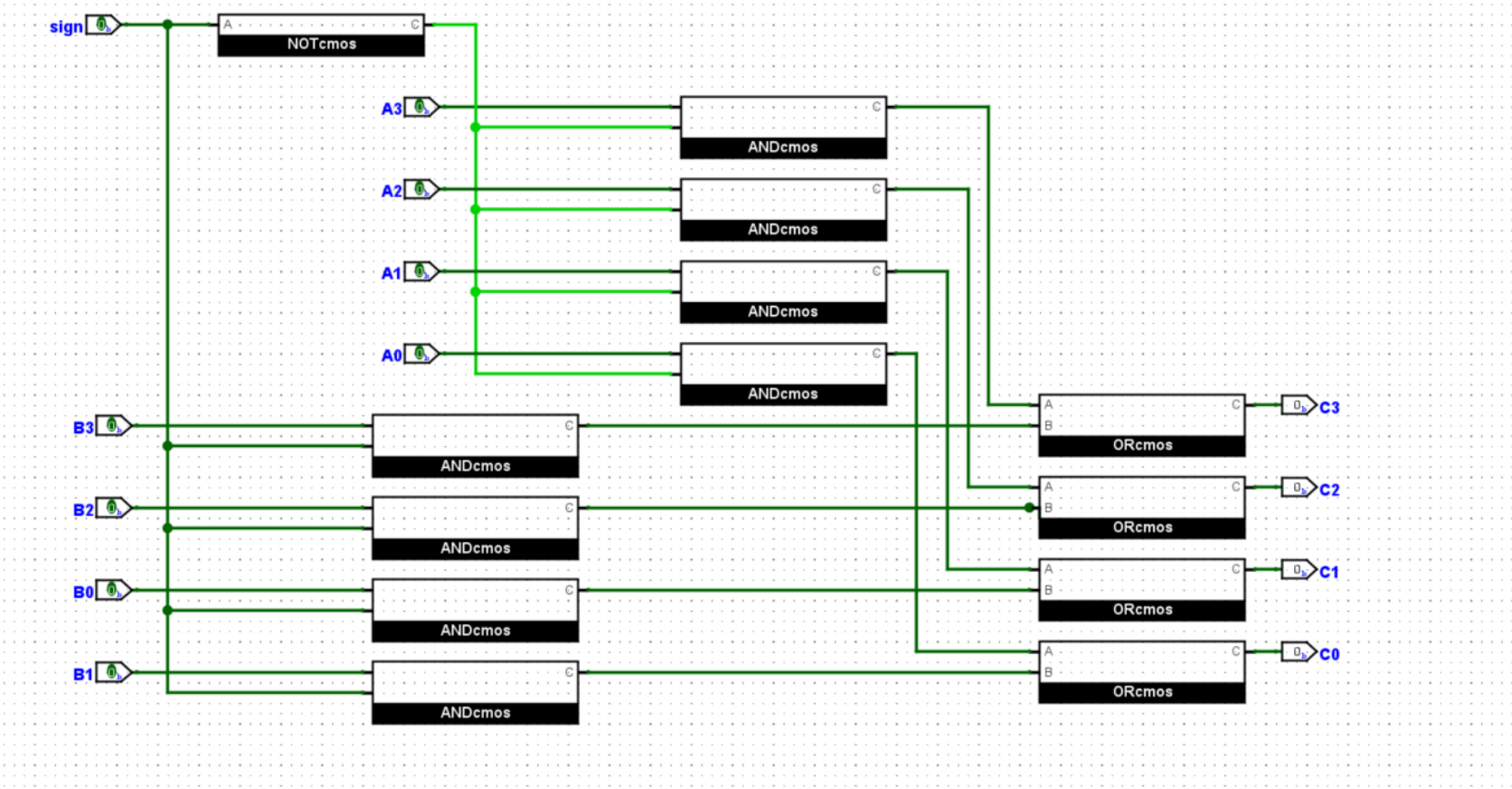
Boolean equations:

$$C3 = (A3 * \overline{sign}) + (B3 * sign)$$

$$C2 = (A2 * \overline{sign}) + (B2 * sign)$$

$$C1 = (A1 * \overline{sign}) + (B1 * sign)$$

$$C0 = (A0 * \overline{sign}) + (B0 * sign)$$



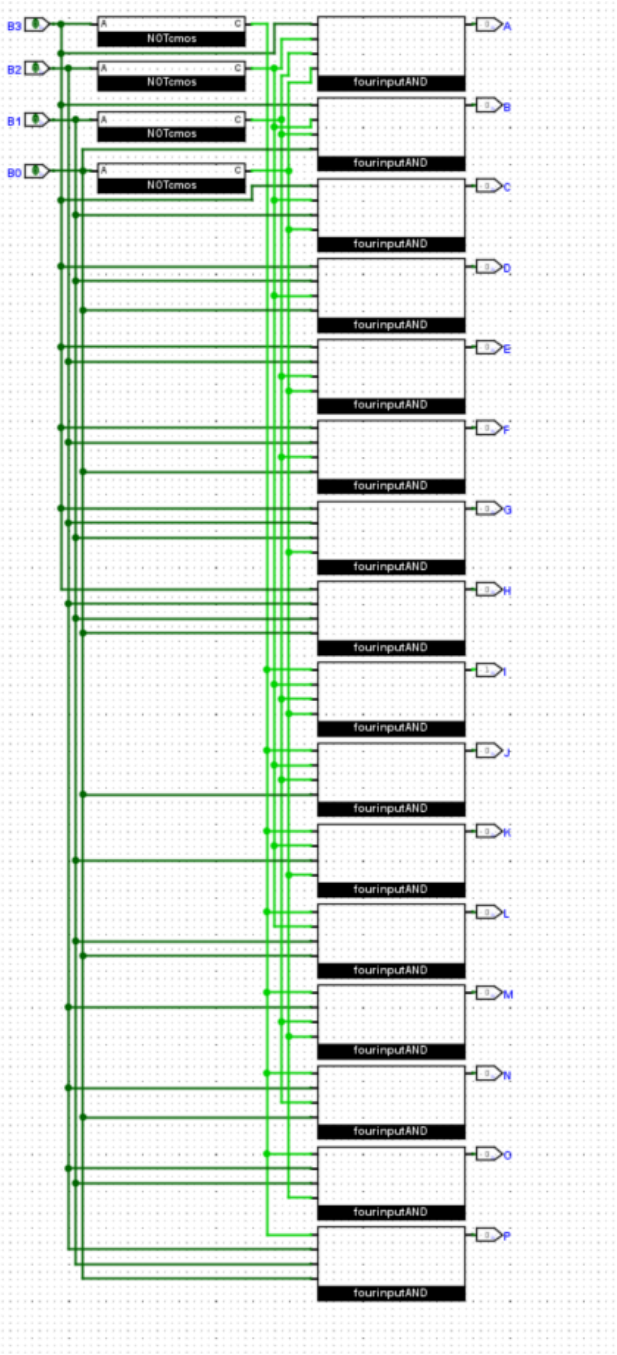
2. Output Decoder

4 inputs: 4 bits of binary number B0, B1, B2, B3, B3 MSB, B0 LSB

SIGNED:

Truth Table:

B3	B2	B1	B0	Decimal	Output Letter	Boolean Equation
1	0	0	0	-8	A	$\overline{B0} * \overline{B1} * \overline{B2} * B3$
1	0	0	1	-7	B	$B0 * \overline{B1} * \overline{B2} * B3$
1	0	1	0	-6	C	$\overline{B0} * B1 * \overline{B2} * B3$
1	0	1	1	-5	D	$B0 * B1 * \overline{B2} * B3$
1	1	0	0	-4	E	$\overline{B0} * \overline{B1} * B2 * B3$
1	1	0	1	-3	F	$B0 * \overline{B1} * B2 * B3$
1	1	1	0	-2	G	$\overline{B0} * B1 * B2 * B3$
1	1	1	1	-1	H	$B0 * B1 * B2 * B3$
0	0	0	0	0	I	$\overline{B0} * \overline{B1} * \overline{B2} * \overline{B3}$
0	0	0	1	1	J	$B0 * \overline{B1} * \overline{B2} * \overline{B3}$
0	0	1	0	2	K	$\overline{B0} * B1 * \overline{B2} * \overline{B3}$
0	0	1	1	3	L	$B0 * B1 * \overline{B2} * \overline{B3}$
0	1	0	0	4	M	$\overline{B0} * \overline{B1} * B2 * \overline{B3}$
0	1	0	1	5	N	$B0 * \overline{B1} * B2 * \overline{B3}$
0	1	1	0	6	O	$\overline{B0} * B1 * B2 * \overline{B3}$
0	1	1	1	7	P	$B0 * B1 * B2 * \overline{B3}$

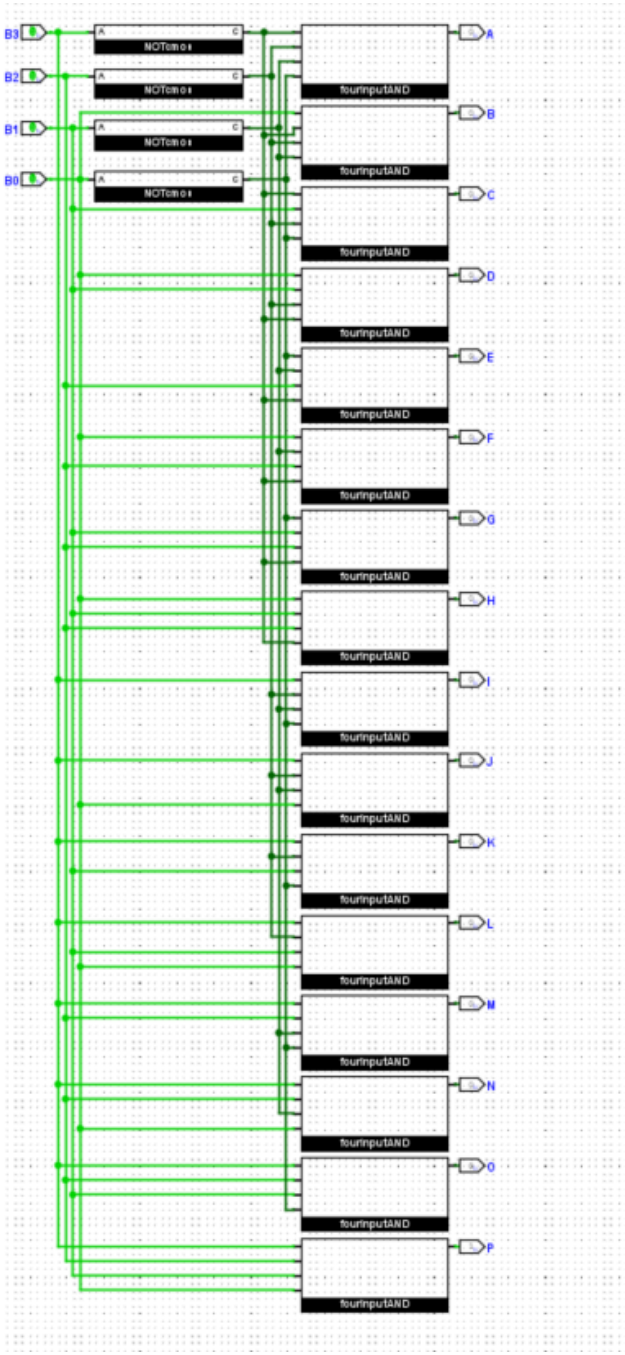

B3
B2
B1
B0
NOTcmos
fourinputAND
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P

UNSIGNED:

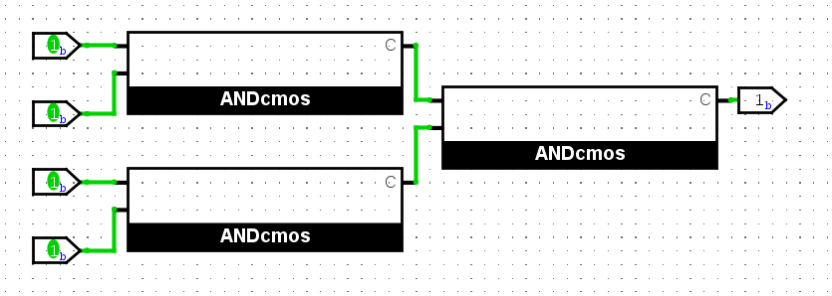
Truth Table:

B3	B2	B1	B0	Decimal	Output Letter	Boolean Equation
0	0	0	0	0	A	$\overline{B0} * \overline{B1} * \overline{B2} * \overline{B3}$
0	0	0	1	1	B	$B0 * \overline{B1} * \overline{B2} * \overline{B3}$
0	0	1	0	2	C	$\overline{B0} * B1 * \overline{B2} * \overline{B3}$
0	0	1	1	3	D	$B0 * B1 * \overline{B2} * \overline{B3}$
0	1	0	0	4	E	$\overline{B0} * \overline{B1} * B2 * \overline{B3}$
0	1	0	1	5	F	$B0 * \overline{B1} * B2 * \overline{B3}$
0	1	1	0	6	G	$\overline{B0} * B1 * B2 * \overline{B3}$
0	1	1	1	7	H	$B0 * B1 * B2 * \overline{B3}$
1	0	0	0	8	I	$\overline{B0} * \overline{B1} * \overline{B2} * B3$
1	0	0	1	9	J	$B0 * \overline{B1} * \overline{B2} * B3$
1	0	1	0	10	K	$\overline{B0} * B1 * \overline{B2} * B3$
1	0	1	1	11	L	$B0 * B1 * \overline{B2} * B3$
1	1	0	0	12	M	$\overline{B0} * \overline{B1} * B2 * B3$
1	1	0	1	13	N	$B0 * \overline{B1} * B2 * B3$

1	1	1	0	14	O	$\overline{B0} * B1 * B2 * B3$
1	1	1	1	15	P	$B0 * B1 * B2 * B3$


B3
B2
B1
B0
NOTgate
fourinputAND
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P

Custom 4-input AND gate (every boolean equation is 4 ANDs):



16-bit MUX

Toggle between the signed vs unsigned value based on the sign value

Inputs: A0, B0, C0, D0, E0, F0, … P0 (unsigned values), A1, B1, C1, D1, E1, F1, … P1 (signed values), sign=0 for unsigned, sign=1 for signed

Output: A, B, C, D, E, … P → correct value

Boolean equations:

$A = (A0 * \overline{sign}) + (A1 * sign)$

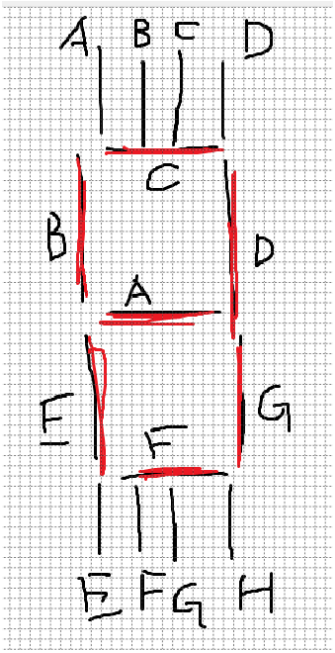
$B = (B0 * \overline{sign}) + (B1 * sign)$

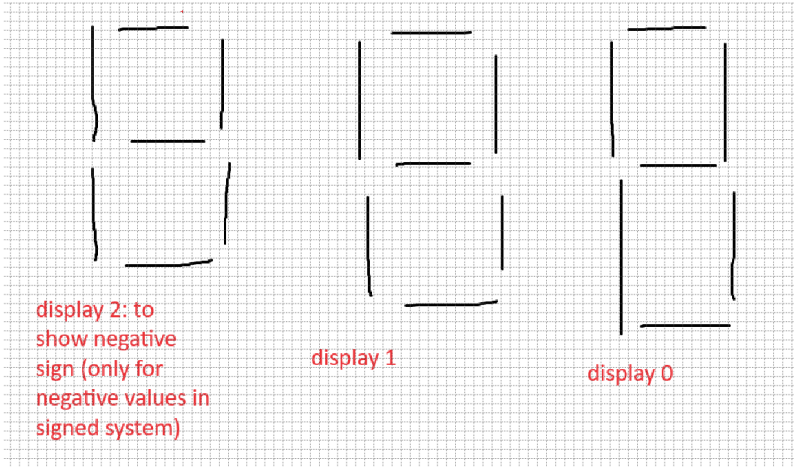
$B = (B0 * \overline{sign}) + (B1 * sign)$

…

$P = (P0 * \overline{sign}) + (P1 * sign)$

7 Segment





Input (map decimal to base 4)

Digit	A	B	C	D	E	F	G
0	0	1	1	1	1	1	1
1	0	0	0	1	0	0	1
2	1	0	1	1	1	1	0
3	1	0	1	1	0	1	1

SIGNED:

	Base 10	Base 4		Display 1							Display 0						
			A2	A1	B1	C1	D1	E1	F1	G1	A0	B0	C0	D0	E0	F0	G0
A	-8	-20	1	1	0	1	1	1	1	0	0	1	1	1	1	1	1
B	-7	-13	1	0	0	0	1	0	0	1	1	0	1	1	0	1	1
C	-6	-12	1	0	0	0	1	0	0	1	1	0	1	1	1	1	0
D	-5	-11	1	0	0	0	1	0	0	1	0	0	0	1	0	0	1
E	-4	-10	1	0	0	0	1	0	0	1	0	1	1	1	1	1	1
F	-3	-03	1	0	1	1	1	1	1	1	1	0	1	1	0	1	1
G	-2	-02	1	0	1	1	1	1	1	1	1	0	1	1	1	1	0
H	-1	-01	1	0	1	1	1	1	1	1	0	0	0	1	0	0	1
I	0	00	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1
J	1	01	0	0	1	1	1	1	1	1	0	0	0	1	0	0	1
K	2	02	0	0	1	1	1	1	1	1	1	0	1	1	1	1	0
L	3	03	0	0	1	1	1	1	1	1	1	0	1	1	0	1	1
M	4	10	0	0	0	0	1	0	0	1	0	1	1	1	1	1	1
N	5	11	0	0	0	0	1	0	0	1	0	0	0	1	0	0	1
O	6	12	0	0	0	0	1	0	0	1	1	0	1	1	1	1	0

P	7	13	0	0	0	0	1	0	0	1	1	0	1	1	0	1	1

UNSIGNED:

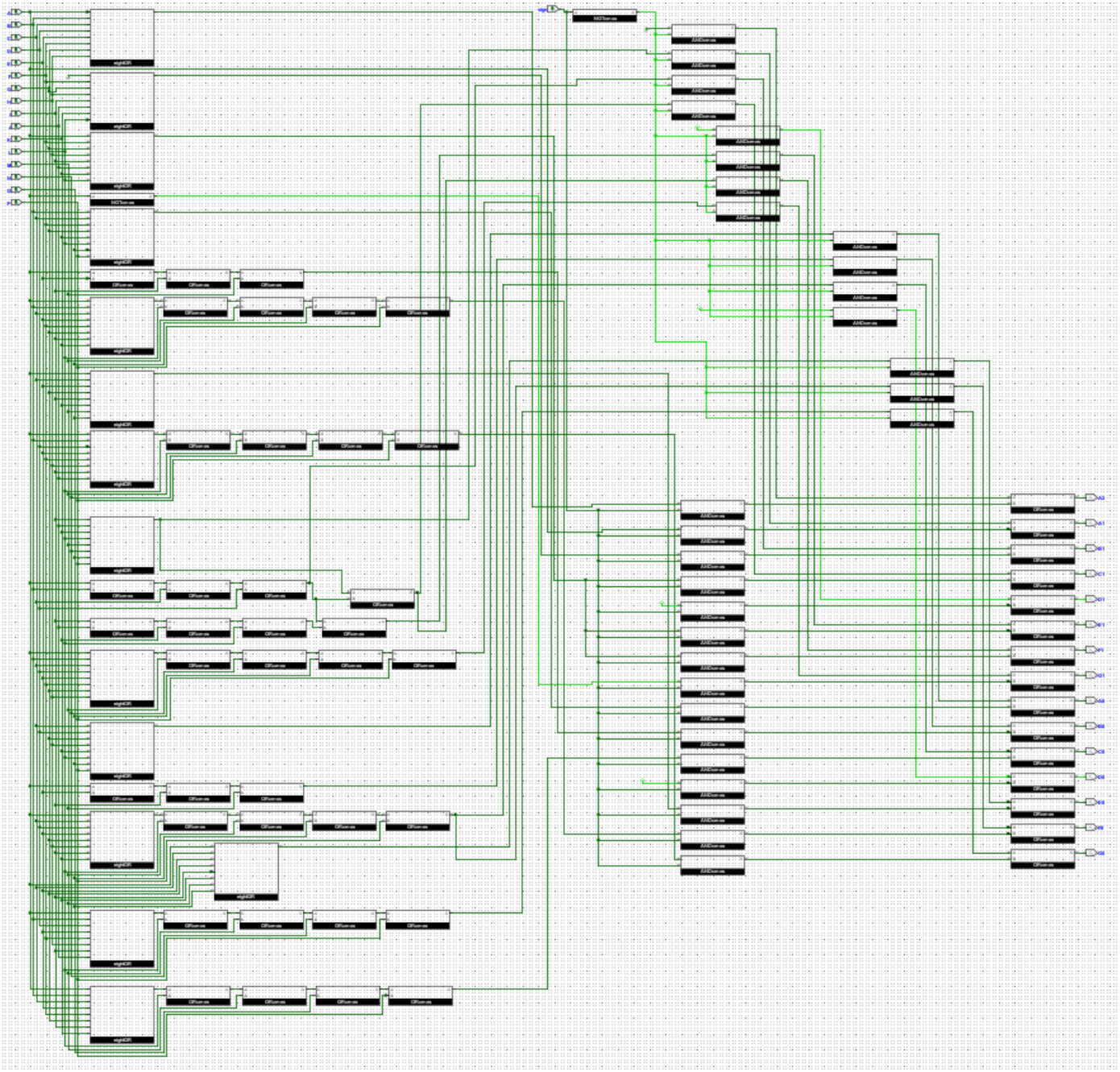
Input	Base 10	Base 4		Display 1							Display 0						
			A2	A1	B1	C1	D1	E1	F1	G1	A0	B0	C0	D0	E0	F0	G0
A	0	00	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1
B	1	01	0	0	1	1	1	1	1	1	0	0	0	1	0	0	1
C	2	02	0	0	1	1	1	1	1	1	1	0	1	1	1	1	0
D	3	03	0	0	1	1	1	1	1	1	1	0	1	1	0	1	1
E	4	10	0	0	0	0	1	0	0	1	0	1	1	1	1	1	1
F	5	11	0	0	0	0	1	0	0	1	0	0	0	1	0	0	1
G	6	12	0	0	0	0	1	0	0	1	1	0	1	1	1	1	0
H	7	13	0	0	0	0	1	0	0	1	1	0	1	1	0	1	1
I	8	20	0	1	0	1	1	1	1	0	0	1	1	1	1	1	1
J	9	21	0	1	0	1	1	1	1	0	0	0	0	1	0	0	1
K	10	22	0	1	0	1	1	1	1	0	1	0	1	1	1	1	0
L	11	23	0	1	0	1	1	1	1	0	1	0	1	1	0	1	1
M	12	30	0	1	0	1	1	0	1	1	0	1	1	1	1	1	1
N	13	31	0	1	0	1	1	0	1	1	0	0	0	1	0	0	1

O	14	32	0	1	0	1	1	0	1	1	1	0	1	1	1	1	0
P	15	33	0	1	0	1	1	0	1	1	1	0	1	1	0	1	1

Using truth table, map using boolean equation by OR the letter inputs that turn each segment on

MUX: segOn = (signedMap() * sign) + (unsignedMap() * (not sign))

Full circuit (signed + unsigned + MUX):



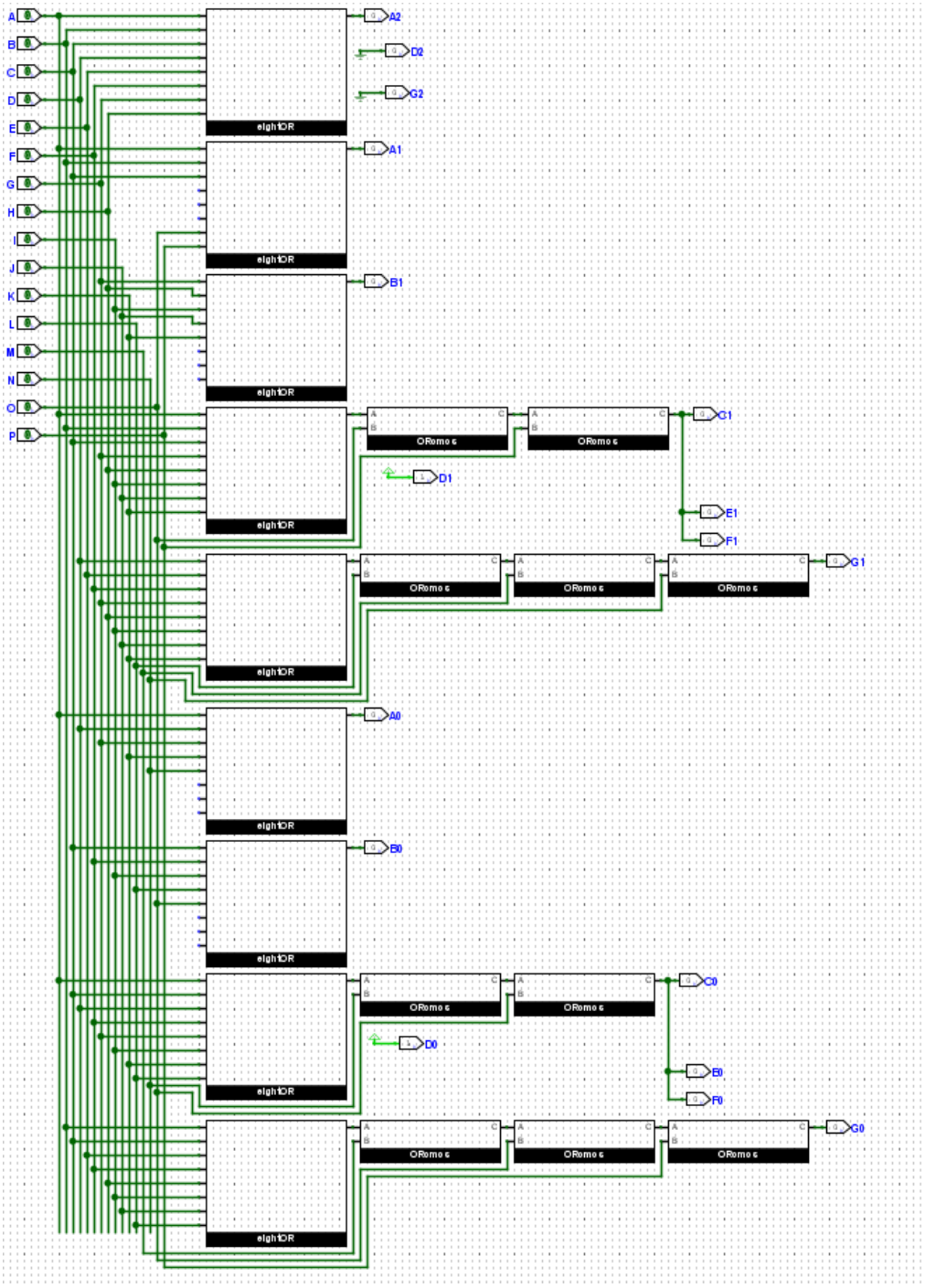
Output (map decimal to base 3)

Digit	A	B	C	D	E	F	G
0	0	1	1	1	1	1	1
1	0	0	0	1	0	0	1
2	1	0	1	1	1	1	0

SIGNED:

Input	Base 10	Base 3	Display 2			Display 1							Display 0						
			A2	D2	G2	A1	B1	C1	D1	E1	F1	G1	A0	B0	C0	D0	E0	F0	G0
A	-8	-22	1			1	0	1	1	1	1	0	1	0	1	1	1	1	0
B	-7	-21	1			1	0	1	1	1	1	0	0	0	0	1	0	0	1
C	-6	-20	1			1	0	1	1	1	1	0	0	1	1	1	1	1	1
D	-5	-12	1			0	0	0	1	0	0	1	1	0	1	1	1	1	0
E	-4	-11	1			0	0	0	1	0	0	1	0	0	0	1	0	0	1
F	-3	-10	1			0	0	0	1	0	0	1	0	1	1	1	1	1	1
G	-2	-02	1			0	1	1	1	1	1	1	1	0	1	1	1	1	0
H	-1	-01	1			0	1	1	1	1	1	1	0	0	0	1	0	0	1
I	0	00				0	1	1	1	1	1	1	0	1	1	1	1	1	1

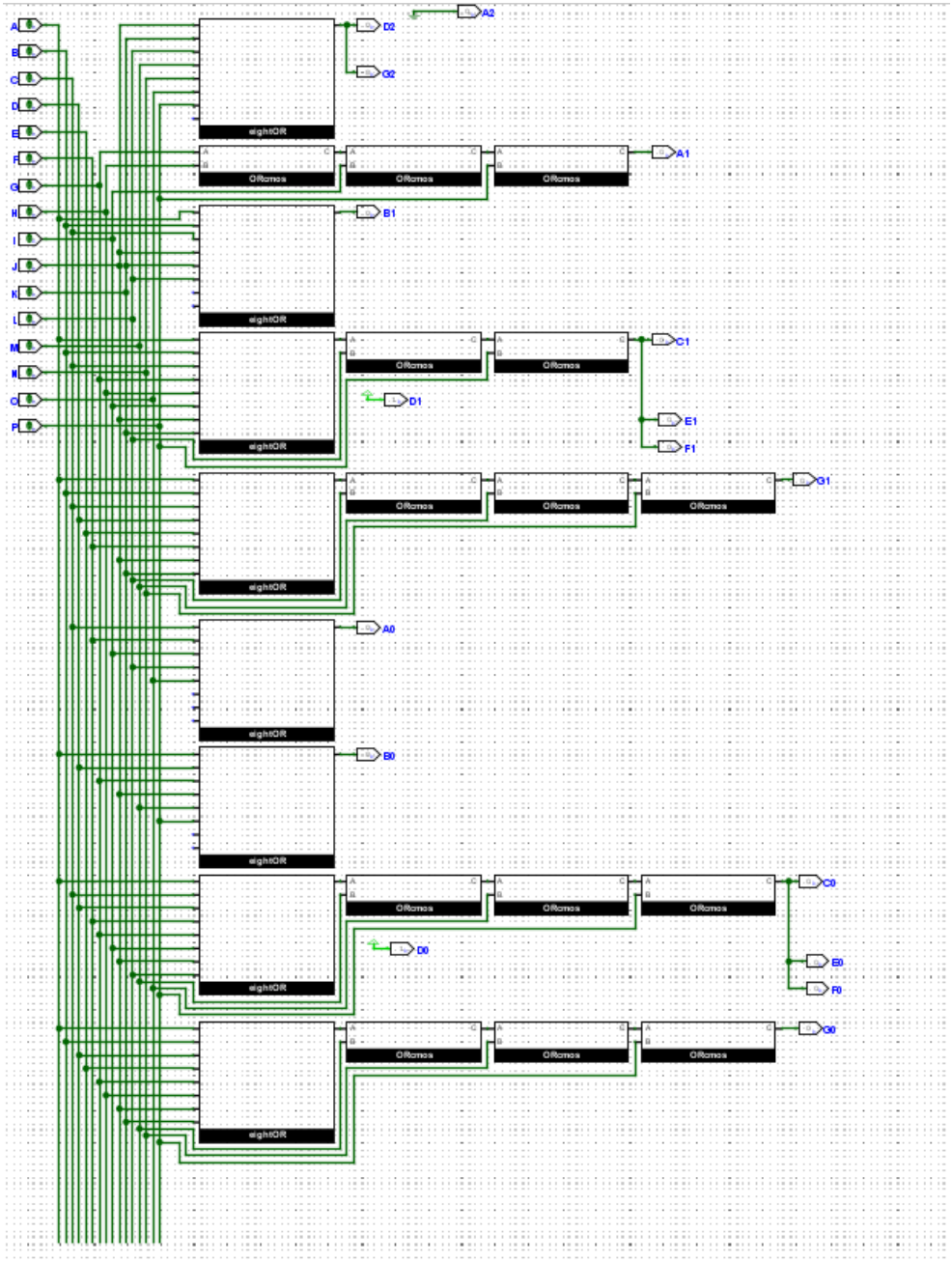
J	1	01				0	1	1	1	1	1	1	0	0	0	1	0	0	1
K	2	02				0	1	1	1	1	1	1	1	0	1	1	1	1	0
L	3	10				0	0	0	1	0	0	1	0	1	1	1	1	1	1
M	4	11				0	0	0	1	0	0	1	0	0	0	1	0	0	1
N	5	12				0	0	0	1	0	0	1	1	0	1	1	1	1	0
O	6	20				1	0	1	1	1	1	0	0	1	1	1	1	1	1
P	7	21				1	0	1	1	1	1	0	0	0	0	1	0	0	1


A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
eight0R
A2
D2
G2
A1
B1
C1
D1
E1
F1
G1
A0
B0
C0
D0
E0
F0
G0
ORomega

UNSIGNED:

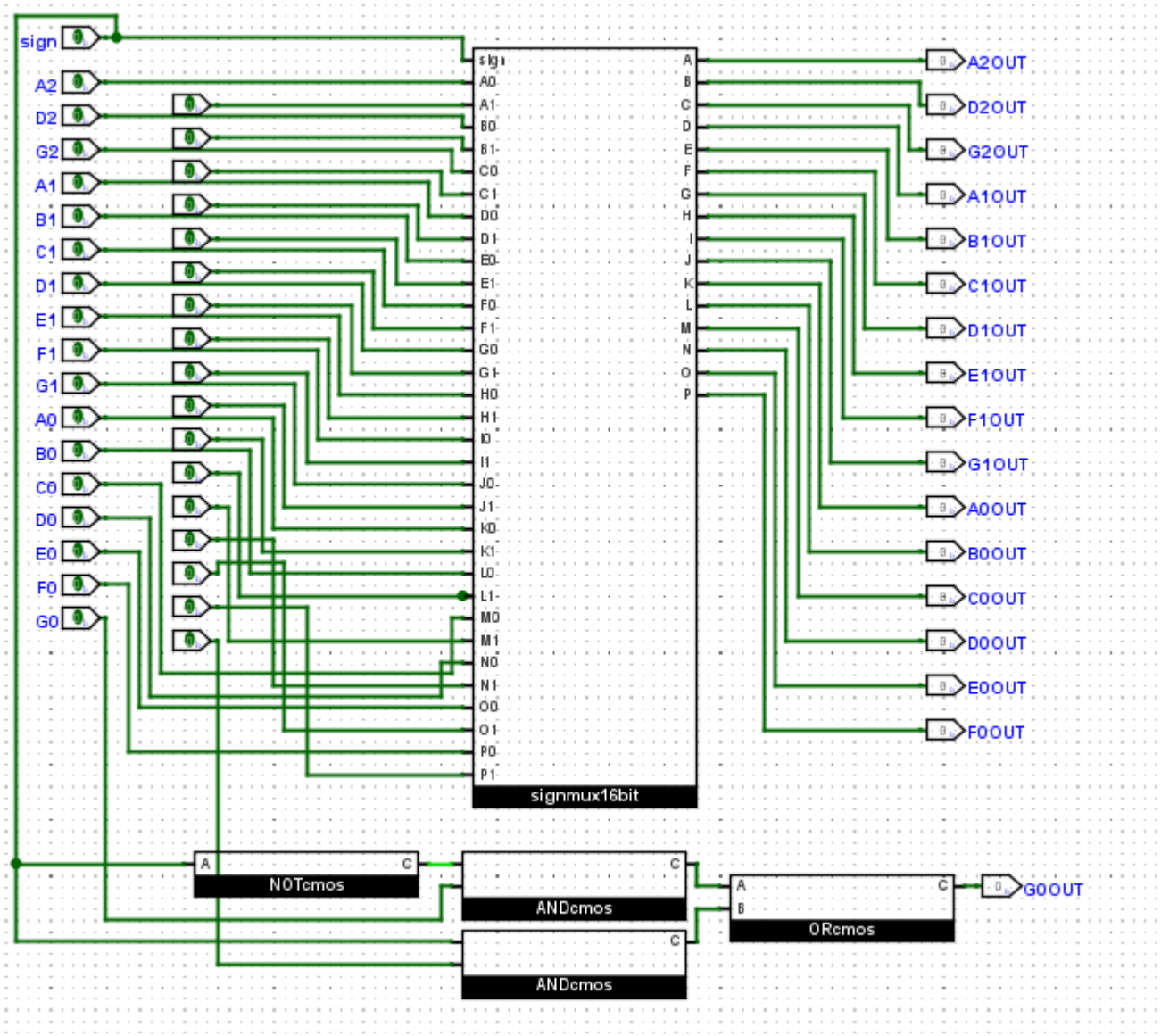
Input	Base 10	Base 3	Display 2			Display 1							Display 0						
			A2	D2	G2	A1	B1	C1	D1	E1	F1	G1	A0	B0	C0	D0	E0	F0	G0
A	0	00				0	1	1	1	1	1	1	0	1	1	1	1	1	1
B	1	01				0	1	1	1	1	1	1	0	0	0	1	0	0	1
C	2	02				0	1	1	1	1	1	1	1	0	1	1	1	1	0
D	3	10				0	0	0	1	0	0	1	0	1	1	1	1	1	1
E	4	11				0	0	0	1	0	0	1	0	0	0	1	0	0	1
F	5	12				0	0	0	1	0	0	1	1	0	1	1	1	1	0
G	6	20				1	0	1	1	1	1	0	0	1	1	1	1	1	1
H	7	21				1	0	1	1	1	1	0	0	0	0	1	0	0	1
I	8	22				1	0	1	1	1	1	0	1	0	1	1	1	1	0
J	9	100		1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1
K	10	101		1	1	0	1	1	1	1	1	1	0	0	0	1	0	0	1
L	11	102		1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	0
M	12	110		1	1	0	0	0	1	0	0	1	0	1	1	1	1	1	1
N	13	111		1	1	0	0	0	1	0	0	1	0	0	0	1	0	0	1
O	14	112		1	1	0	0	0	1	0	0	1	1	0	1	1	1	1	0

P	15	120		1	1	1	0	1	1	1	1	0	0	1	1	1	1	1	1

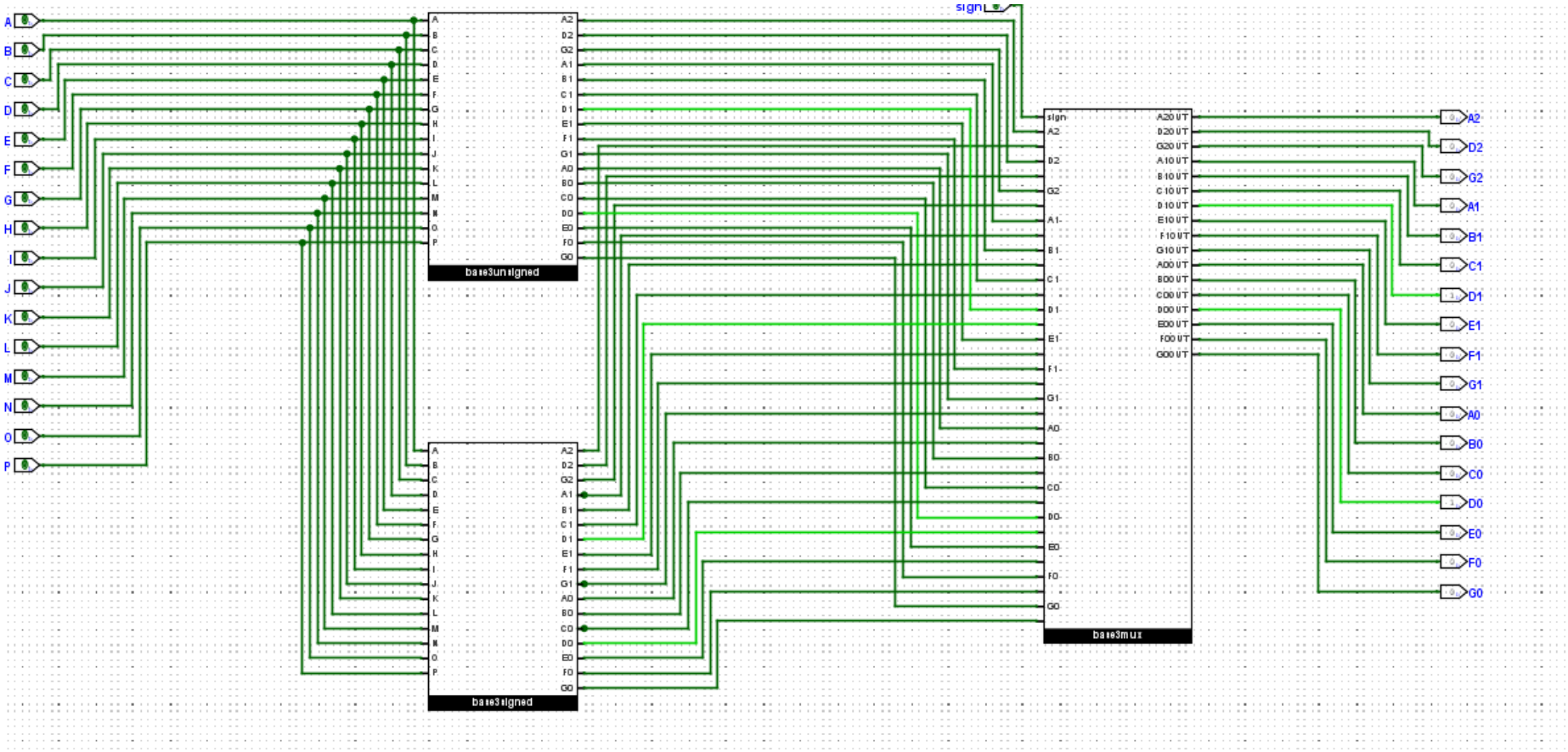

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
eightOR
ORomos
A2
D2
C2
A1
B1
C1
D1
E1
F1
G1
A0
B0
C0
D0
E0
F0
G0

MUX: segOn = (signedMap() * sign) + (unsignedMap() * (not sign))

Suffix 0 = unsigned, suffix 1 = signed


sign
A2
D2
G2
A1
B1
C1
D1
E1
F1
G1
A0
B0
C0
D0
E0
F0
G0
signmux16bit
A2OUT
D2OUT
G2OUT
A1OUT
B1OUT
C1OUT
D1OUT
E1OUT
F1OUT
G1OUT
A0OUT
B0OUT
C0OUT
D0OUT
E0OUT
F0OUT
NOTcmos
ANDcmos
ANDcmos
ORcmos
G0OUT

Put all 3 circuits above together:



Output Overflow:

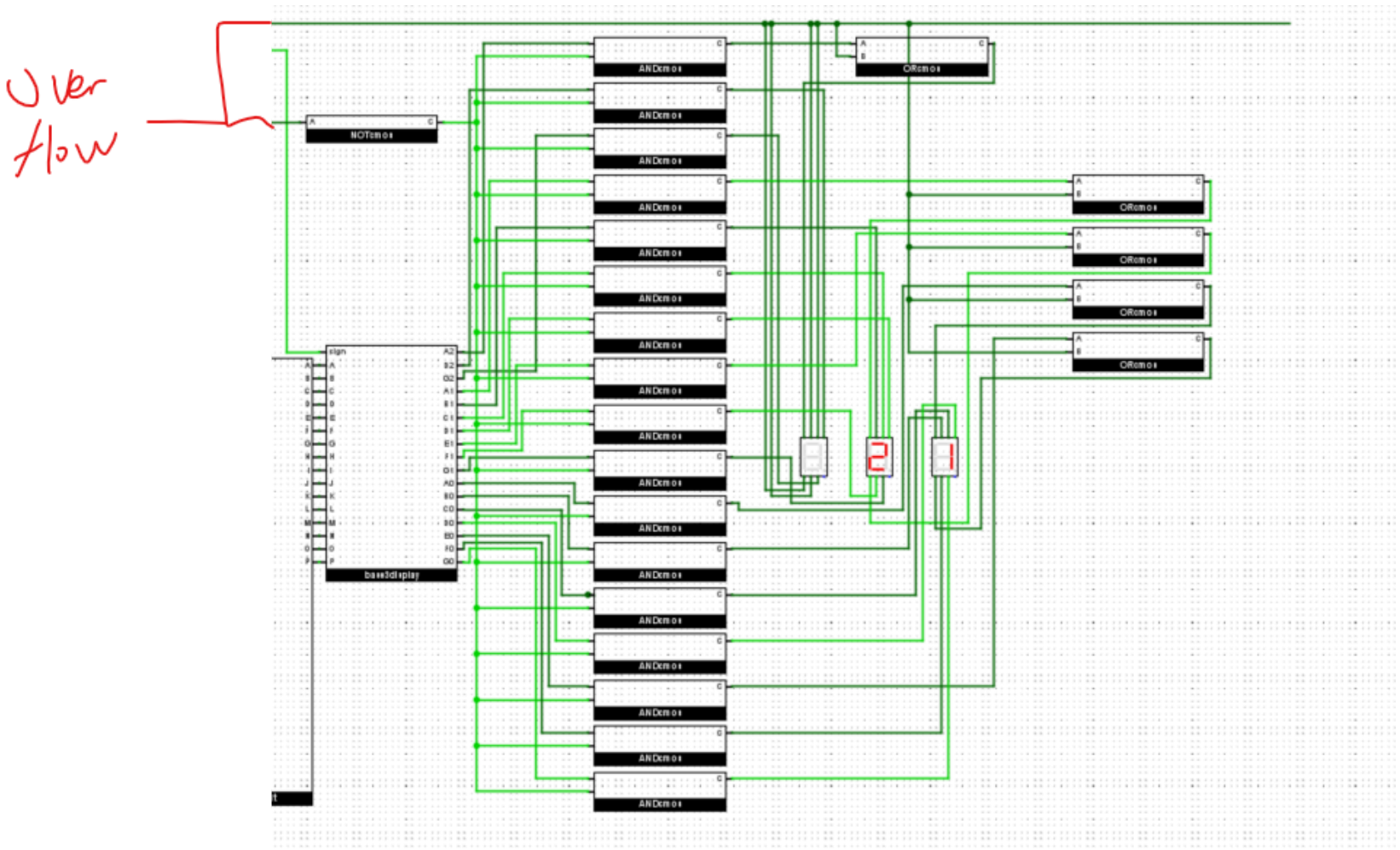
If overflow=1 then do not output anything

Boolean equation:

segOn = sevensegmap() * (not overflow)

For segments that are turned on to display Err:

segOn = sevensegmap() * (not overflow) + overflow



Main Circuit: