CSE 331/503

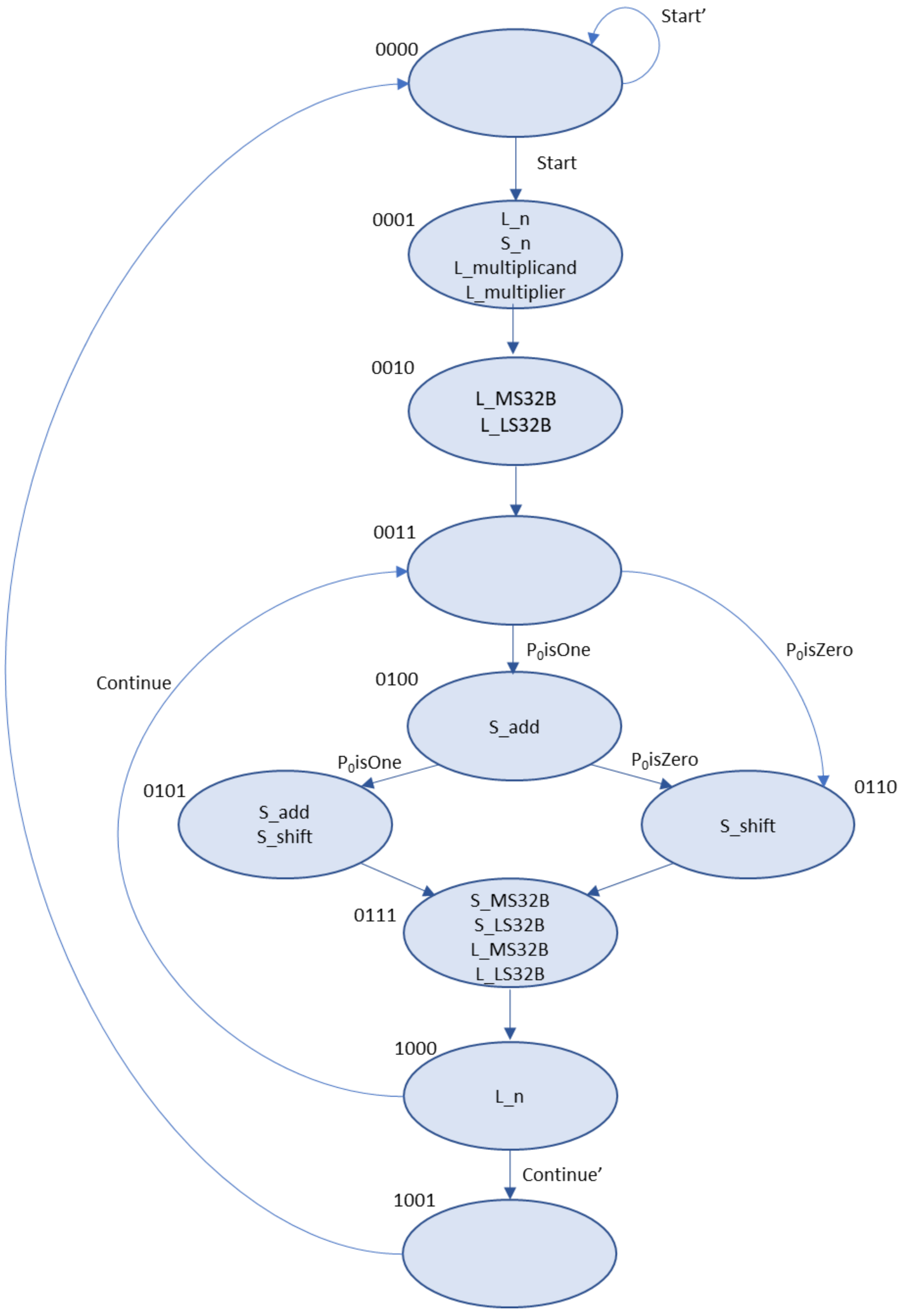
Computer Organization

HW3

Elif Goral

171044003

STATE DIAGRAM



Truth Tables

present state				inputs				next state			
P3	P2	P1	P0	start	P0isZero	P0isOne	continue	N3	N2	N1	N0
0	0	0	0	0	x	x	x	0	0	0	0
0	0	0	0	1	x	x	x	0	0	0	1
0	0	0	1	x	x	x	x	0	0	1	0
0	0	1	0	x	x	x	x	0	0	1	1
0	0	1	1	x	0	1	x	0	1	0	0
0	0	1	1	x	1	0	x	0	1	1	0
0	1	0	0	x	0	1	x	0	1	0	1
0	1	0	0	x	1	0	x	0	1	1	0
0	1	0	1	x	x	x	x	0	1	1	1
0	1	1	0	x	x	x	x	0	1	1	1
0	1	1	1	x	x	x	x	1	0	0	0
1	0	0	0	x	x	x	1	0	0	1	1
1	0	0	0	x	x	x	0	1	0	0	1
1	0	0	1	x	x	x	x	0	0	0	0

present state				outputs									
P3	P2	P1	P0	Lmultiplicand	Lmultiplier	L_MS32B	L_LS32B	S_MS32B	S_LS32B	S_shift	S_add	S_n	Ln
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	1	0	0
0	1	0	1	0	0	0	0	0	0	1	1	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	1	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	0	0	0	0

Signals

L_multiplicand: It indicates whether to write to the multiplicand register. Connecting to the enable input.

L_multiplicand = P3'.P2'.P1'.P0

L_multiplier: It indicates whether to write to the multiplier register. Connecting to the enable input.

L_multiplier = P3'.P2'.P1'.P0

L_MS32B: enable input of MS32B(most significant 32 bit) register.

L_MS32B = P3'.P2'.P1.P0' + P3'.P2.P1.P0

L_MS32B = P3'.P1'.(P2 xnor P0)

L_LS32B: enable input of LS32B(least significant 32 bit) register.

L_LS32B = P3'.P2'.P1.P0' + P3'.P2.P1.P0

L_LS32B = P3'.P1'.(P2 xnor P0)

S_MS32B: select bit of MS32B register. I connect the zero and MS32B register to the mux. If first situation I put zero to the MS32B register, otherwise I put MS32B register's current value.

S_MS32B = P3'.P2.P1.P0

S_LS32B: select bit of LS32B register. I connect the multiplier and LS32B register to the mux. If first situation I put multiplier to the LS32B register, otherwise I put LS32B register's current value.

S_LS32B = P3'.P2.P1.P0

S_shift:

S_shift = P3'.P2.P1'.P0 + P3'.P2.P1.P0'

S_shift = P3'.P2.(P1 xor P0)

S_add: shows whether to add or not

$$S\_add = P3'.P2.P1'.P0' + P3'.P2.P1'.P0$$

$$S\_add = P3'.P2.P1'$$

S_n: I use the signal s_n to initialize the counter register from 0 in the first case. That signal only works in 0001 state.

$$S\_n = P3'.P2'.P1'.P0$$

L_n: I user that signal for increment the counter register's value for finishing the multiplication process.

$$L\_n = P3'.P2'.P1'.P0 + P3.P2'.P1'.P0'$$

P0isZero: That signal controls the product's 0 index value. If P0 is zero, returns 1, otherwise returns 0.

P0isOne: That signal controls the product's 0 index value. If P0 is one, returns 1, otherwise returns 0.

Continue: That signal controls the n index. If it is smaller than 32, returns 1, otherwise return 0.

Start: That signal controls the program start or not.

Outputs

N3 = P3’.P2.P1.P0 + P3. P2’.P1’.P0’.C’

N2 = P3’.P2’.P1.P0.P0isZero’.P0isOne + P3’.P2’.P1.P0.P0isZero.P0isOne’ + P3’.P2.P1’.P0’.P0isZero’.P0isOne + P3’.P2.P1’.P0’.P0isZero.P0isOne’ + P3’.P2.P1’.P0 + P3’.P2.P1’.P0 + P3’.P2.P1.P0’

N2 = P3’.P0isZero’.P0isOne . (P2’.P1.P0 + P2.P1’.P0’) + P3’.P2.(P1 xor P0) + P3’.P0isZero.P0isOne’.(P2’.P1.P0 + P2.P1’.P0’)

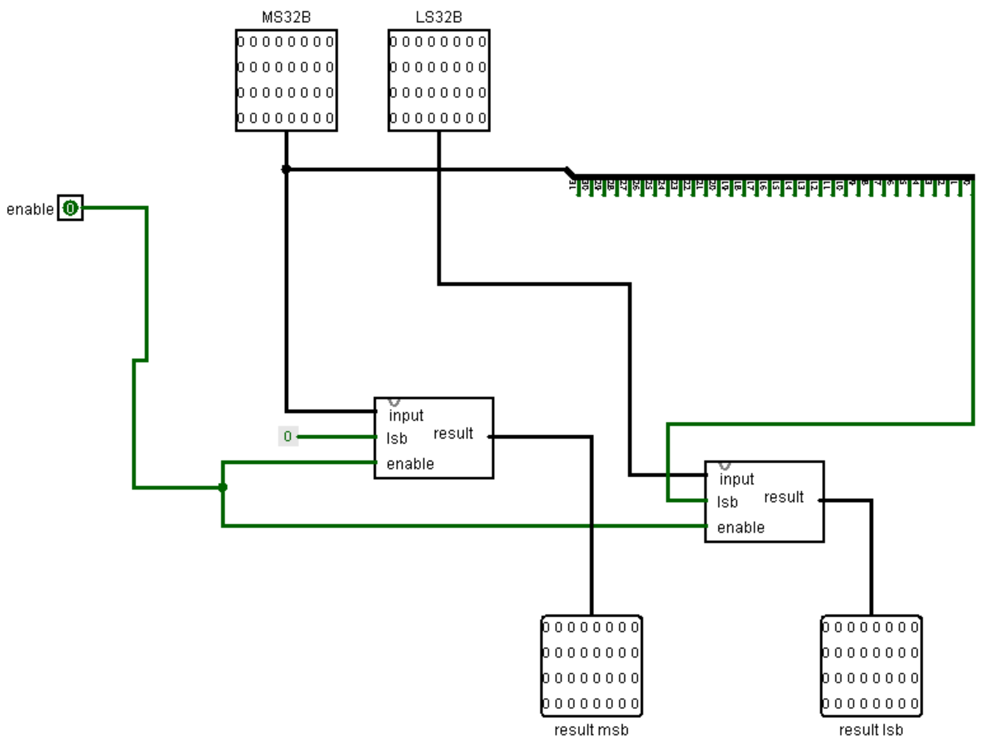
N1 = P3’.P2’.P1’.P0 + P3’.P2’.P1.P0’ + P3’.P2’.P1.P0. P0isZero. P0isOne’+ P3’.P2.P1’.P0’.P0isZero.P0isOne’ + P3’.P2.P1’.P0 + P3’.P2.P1.P0’ + P3.P2’.P1’.P0’.C

N1 = P3’.(P1 xor P0) + P3’.P0isZero.P0isOne’.(P2’.P1.P0 + P2.P1’.P0’) + P3.P2’.P1’.P0’.C

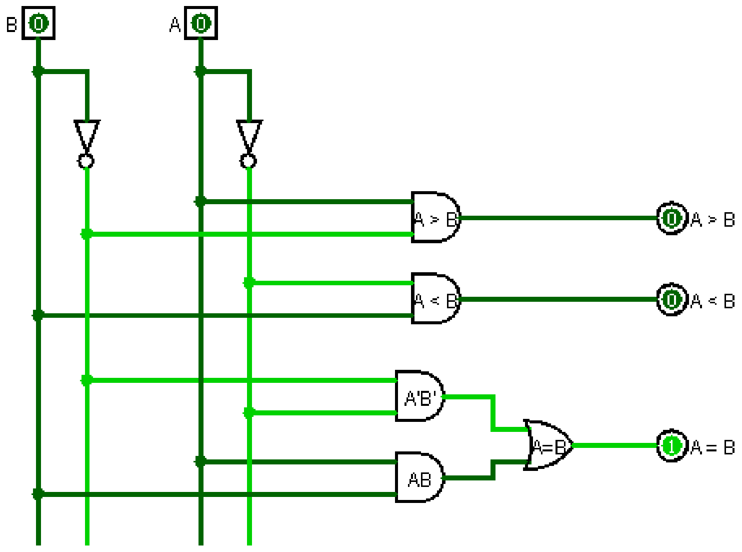
N0 = P3’.P2’.P1’.P0’.S + P3’.P2’.P1.P0’ + P3’.P2.P1’.P0’.P0isZero’.P0isOne + P3’.P2.P1’.P0 + P3’.P2.P1.P0’ + P3.P2’.P1’.P0’.C + P3.P2’.P1’.P0’.C’

N0 = P3’.P2.(P1 xor P0) + P2’.P0’.(P3 xor P1) + P3’.P1’.P0’.(P2’.S + P2.P0isZero’.P0isOne)

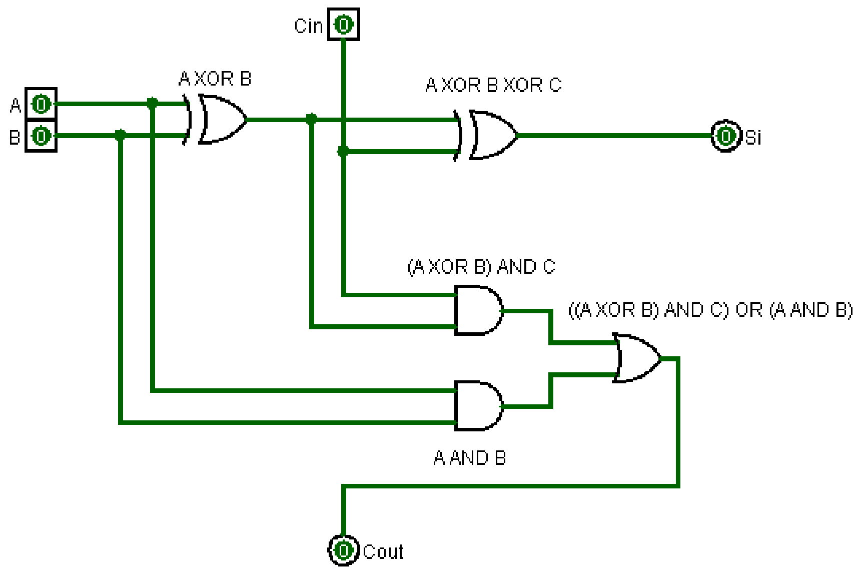
64 bit shifter: I use 32 bit number shifter x2



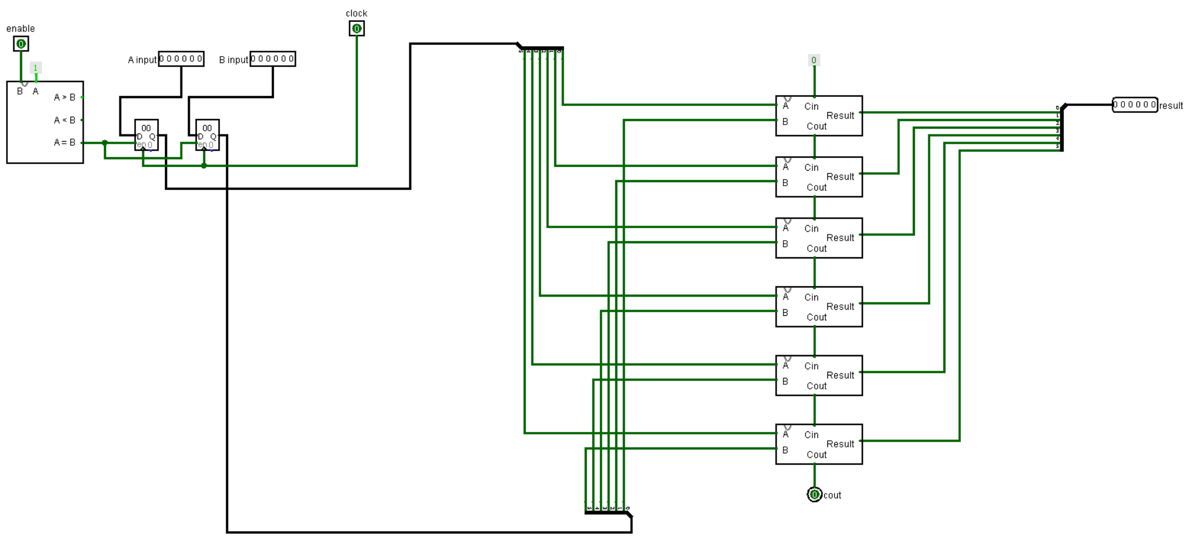
Comparator:



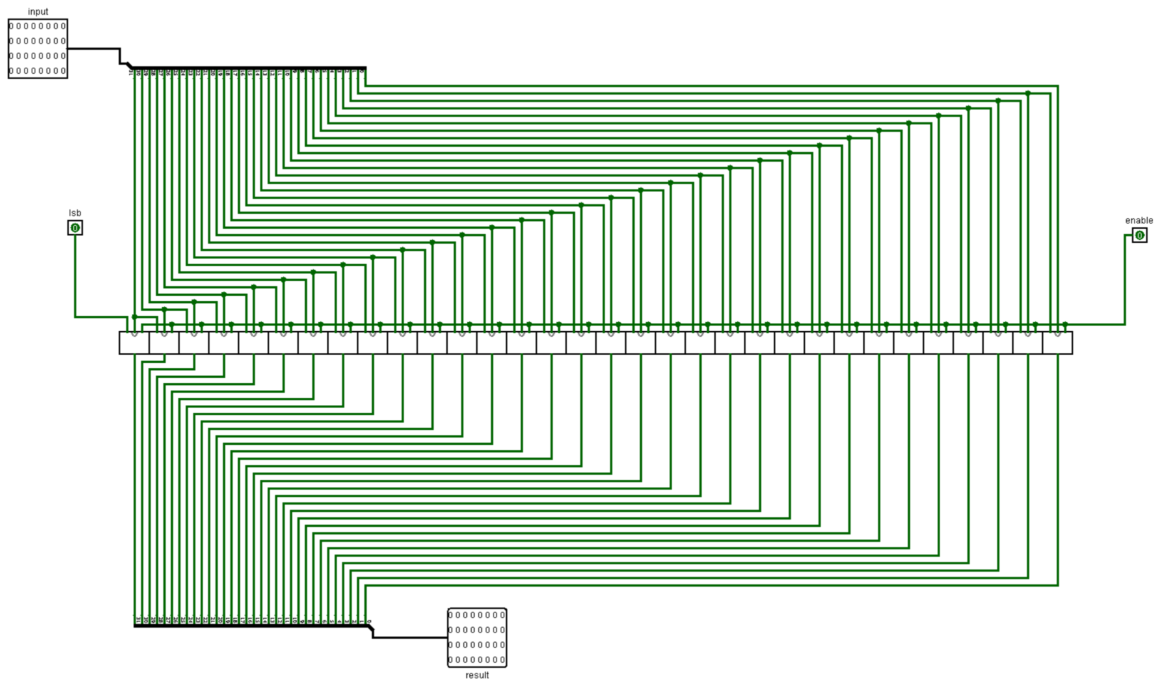
1 bit adder:



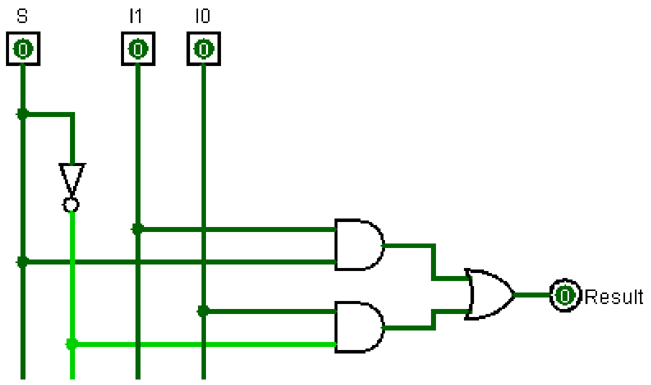
6 bit adder:



32 bit shifter:



2x1 mux:



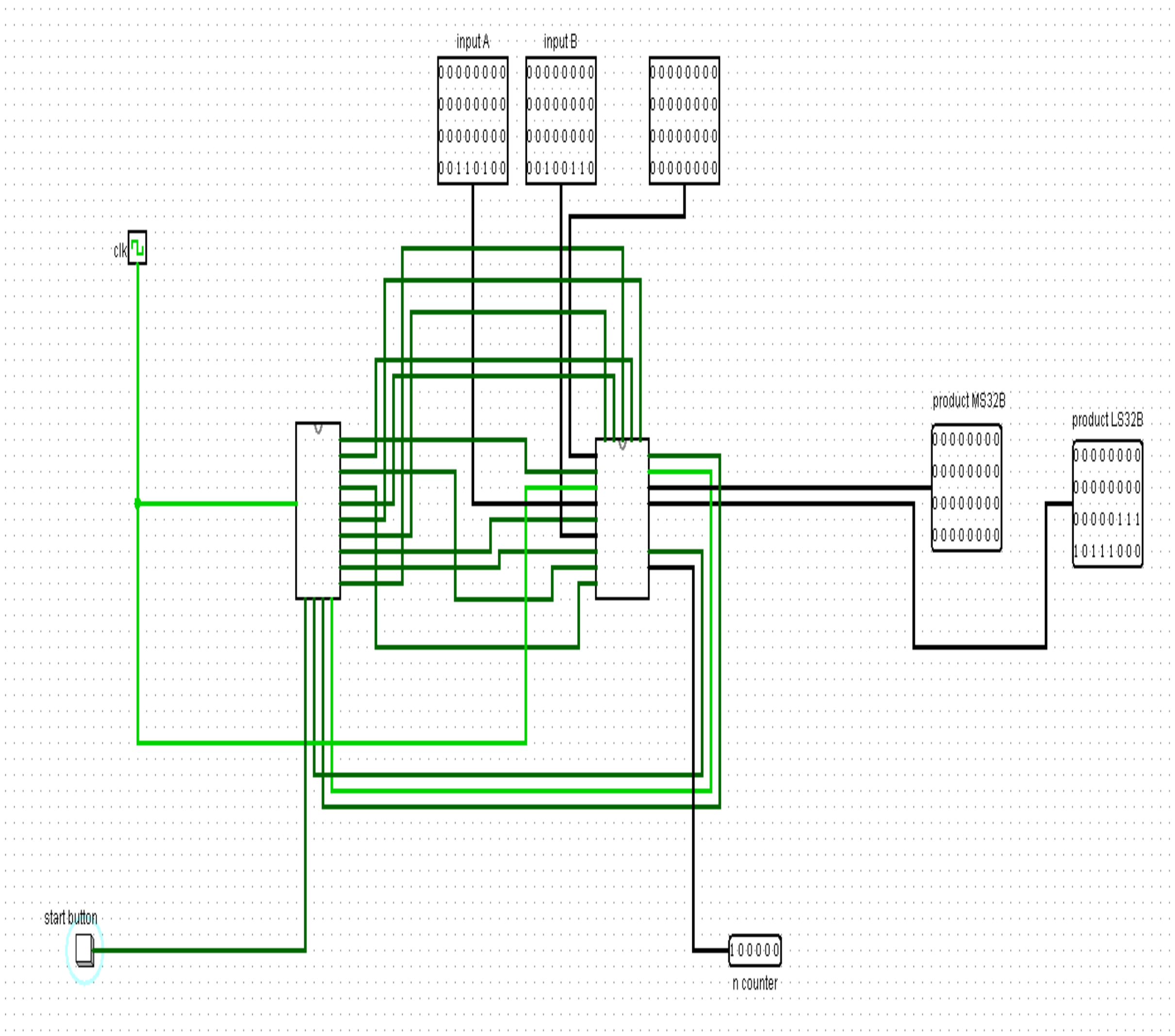
Test Cases:

Test 1:

Input A : 110100

Input B : 100110

Result   : 11110111000

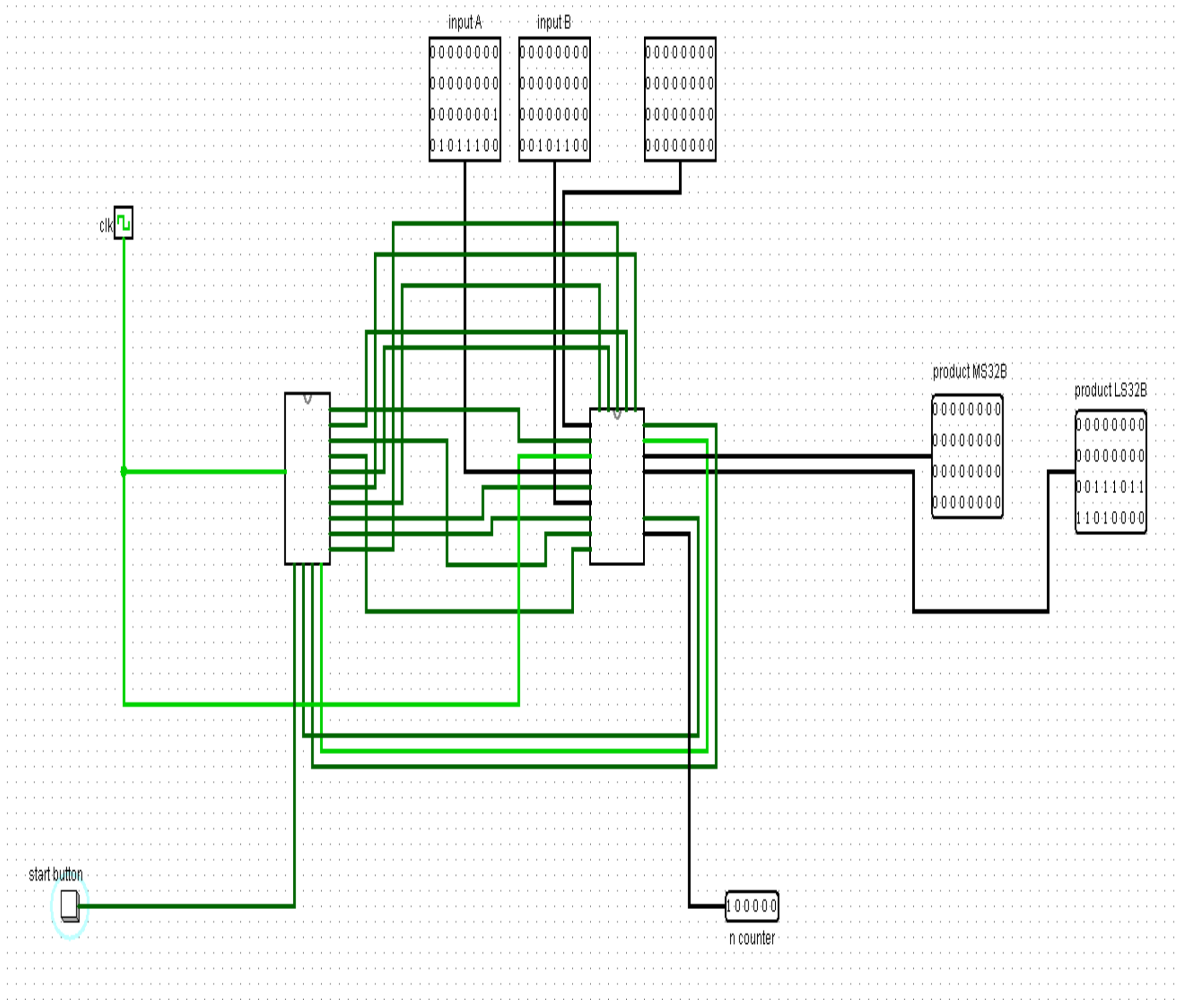


Test 2:

Input A : 101011100

Input B : 101100

Result : 11101111010000

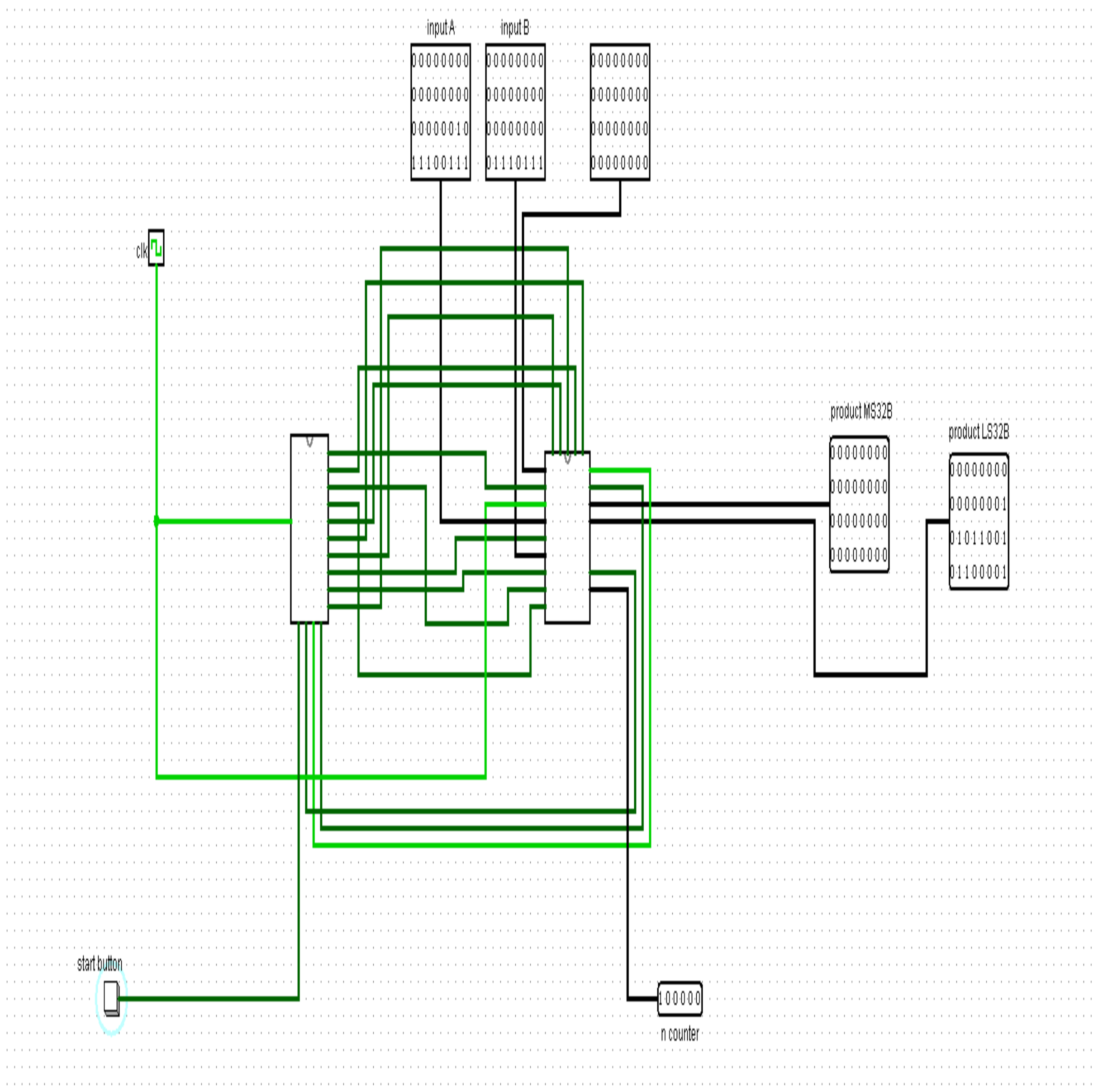


Test 3:

Input A : 1011100111

Input B : 1110111

Result : 10101100101100001

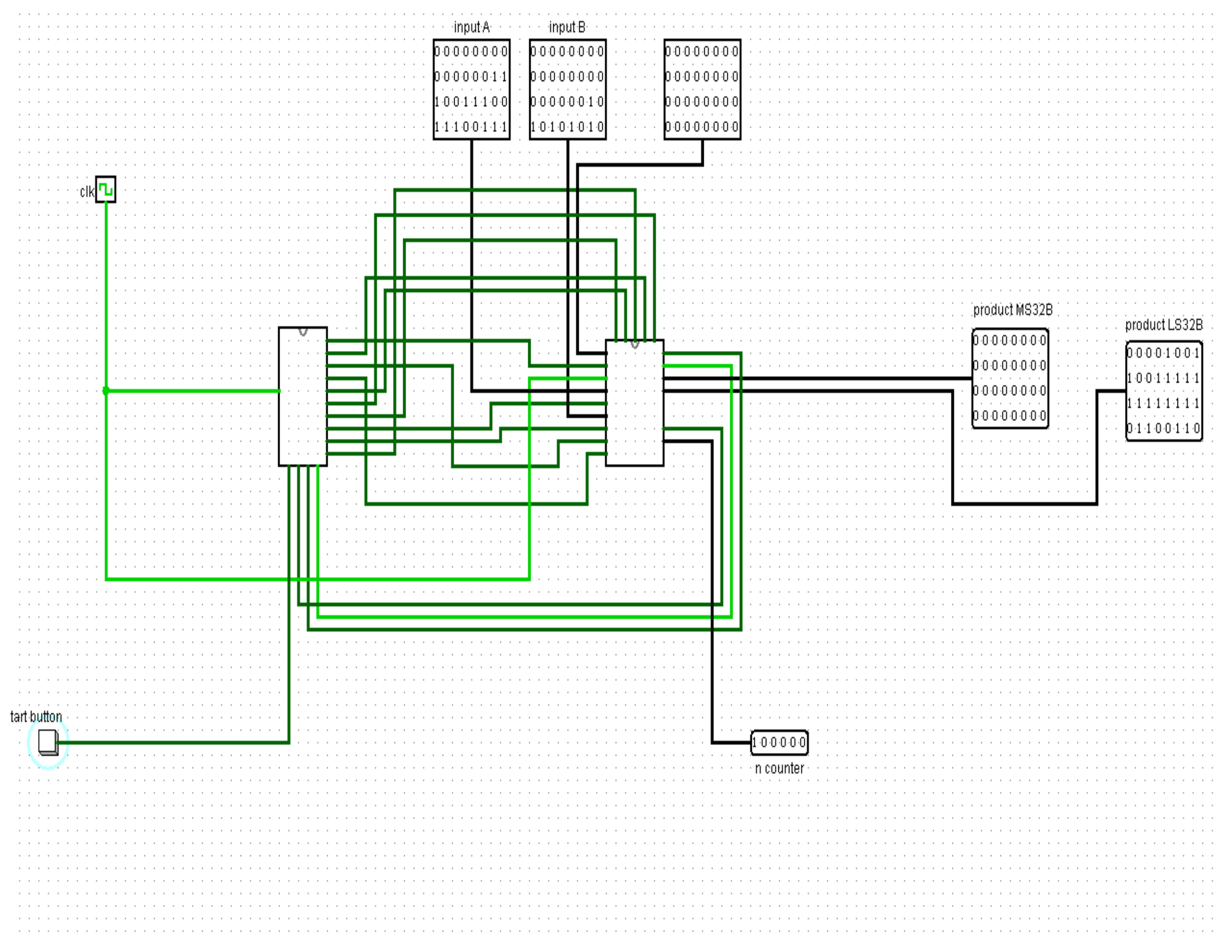


Test 4:

Input A : 111001110011100111

Input B : 1010101010

Result   : 100110011111111111101100110

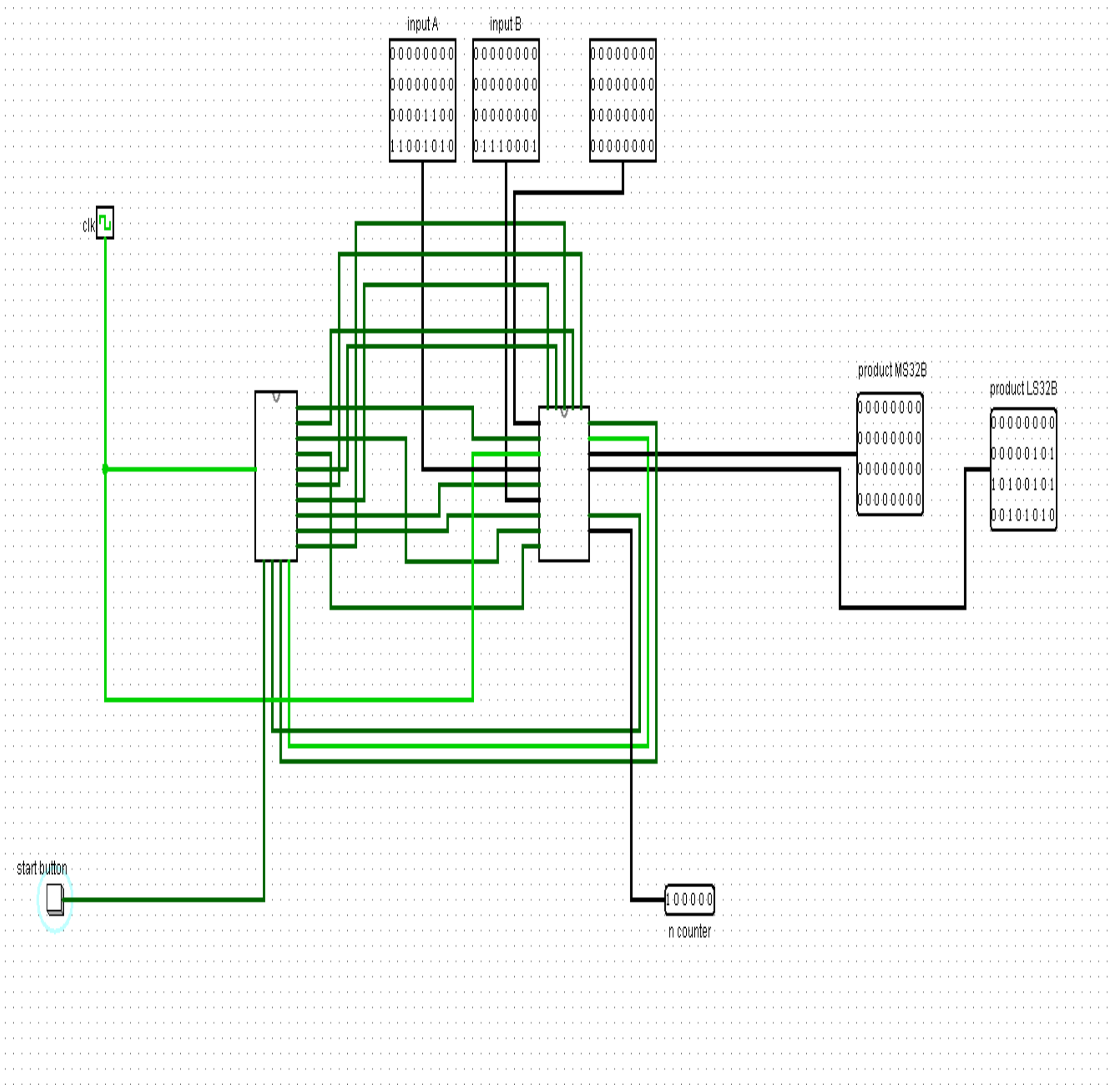


Test 5:

Input A : 110011001010

Input B : 1110001

Result : 1011010010100101010

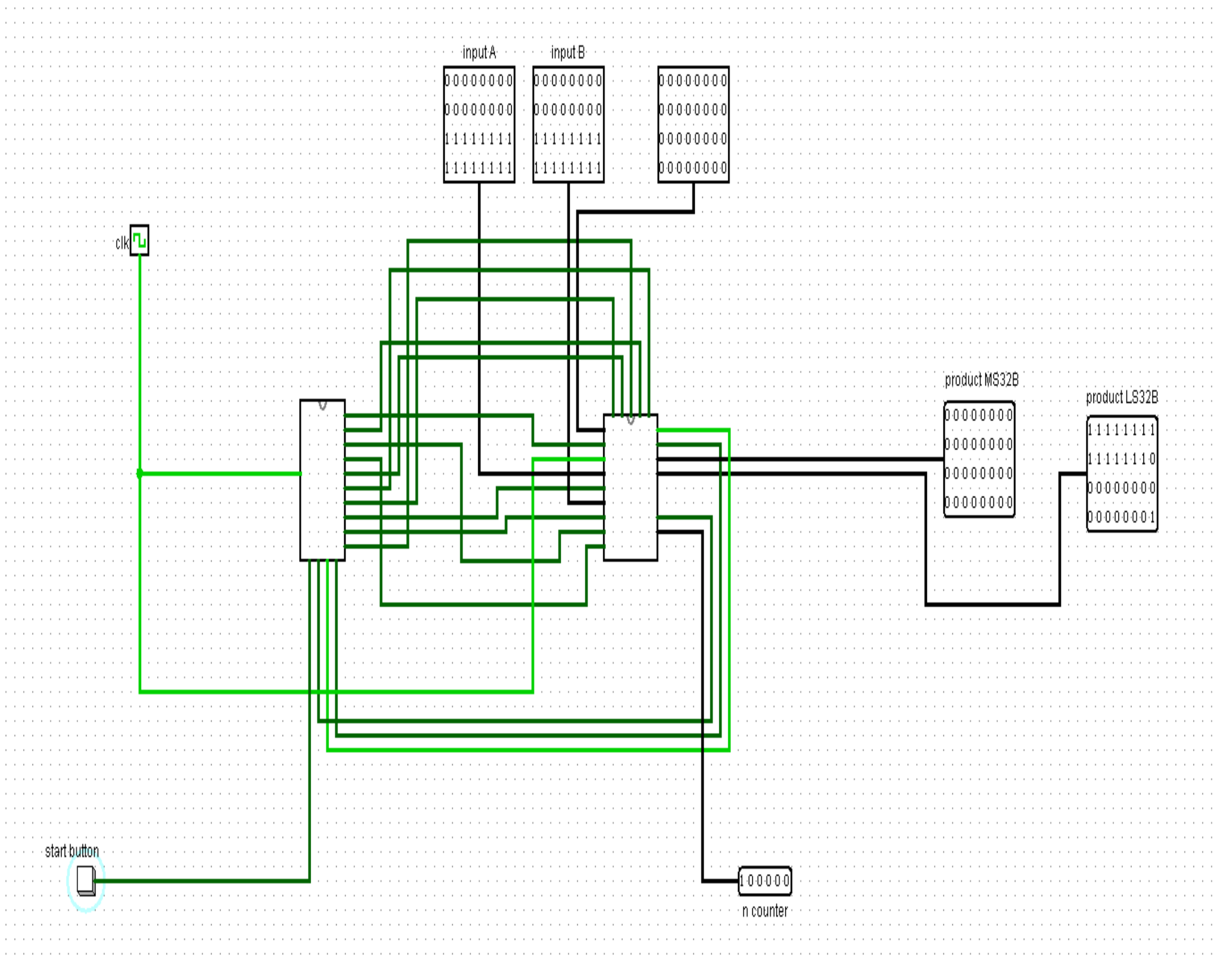


Test 6:

Input A : 1111111111111111

Input B : 1111111111111111

Result : 11111111111111100000000000000001

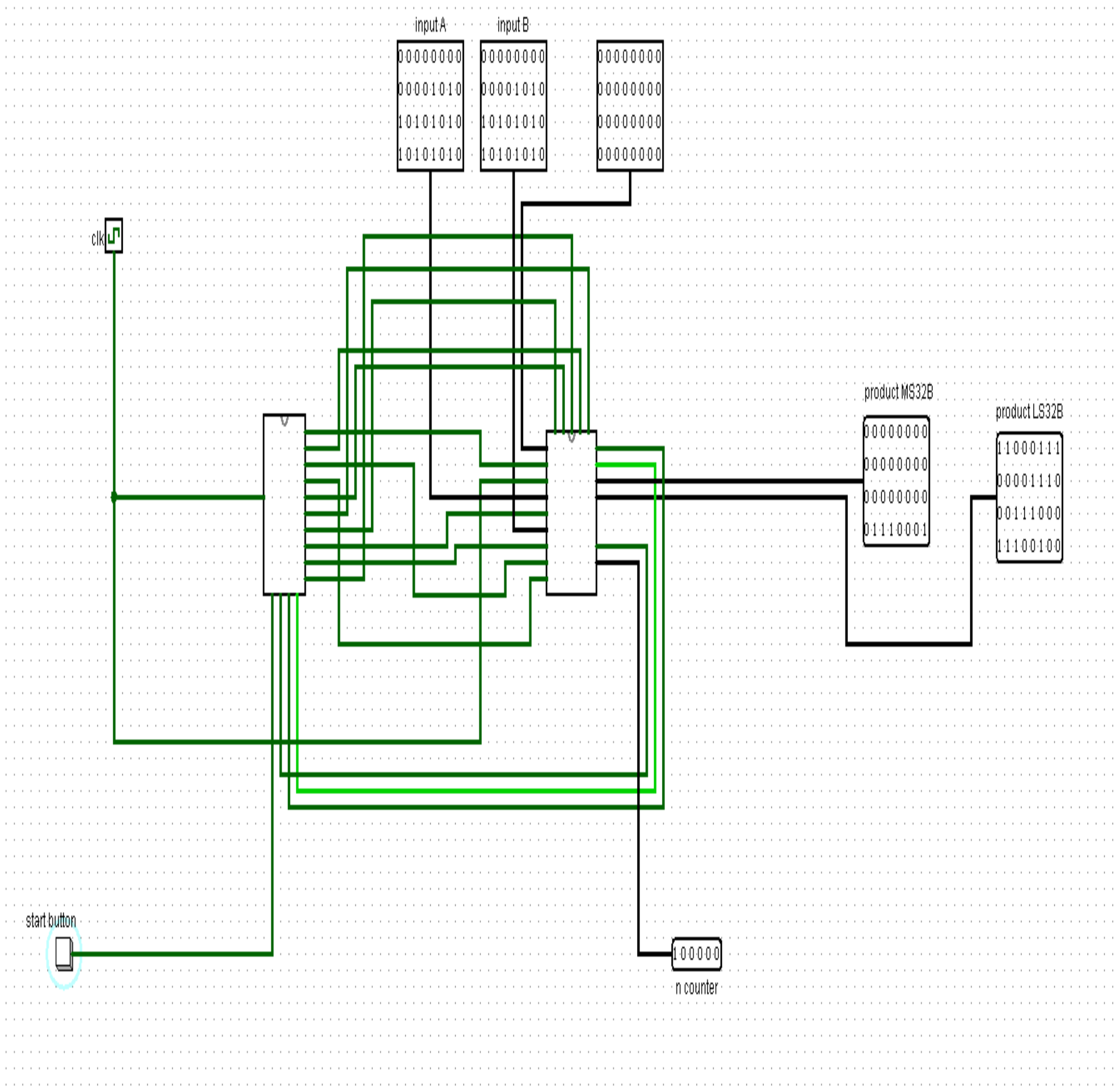


Test 7:

Input A : 1010101010101010101010

Input B : 1010101010101010101010

Result : 1110001110001110000111000111000111000111100100


input A
00000000
00001010
10101010
10101010
input B
00000000
00001010
10101010
10101010
00000000
00000000
00000000
00000000
clk
product MS32B
00000000
00000000
00000000
01110001
product LS32B
11000111
00001110
00111000
11100100
start button
100000
n counter