COMPUTER ARCHITECTURE ASSIGNMENT-2

MULTIPLICATION ALGORITHMS

Tirth Patel (18BCE245)
Shrey Viradiya (18BCE259)

Multiplication of two fixed-point binary numbers in signed-magnitude repre sensation is done with paper and pencil by a process of successive shift and add operations. This process is best illustrated with a numerical example.

23	10111	Multiplicand
19	× 10011	Multiplier
	10111	
	10111	
	00000	+
	00000	
	10111	
437	110110101	Product

When multiplication is implemented in a digital computer, it is convenient to change the process slightly.
So mainly there are 2 famous algorithms for multiplication of two numbers . They are as Follow :

1-Hardware Multiplication
2- Booth Multiplication

We had implemented these algorithm for 8 bit numbers.

1 - HARDWARE MULTIPLY ALGORITHM

When multiplication is implemented in a digital computer, it is convenient to change the process slightly. First, instead of providing registers to store and add simultaneously as many binary numbers as there are bits in the multiplier, it is convenient to provide an adder for the summation of only two binary numbers and successively accumulate the partial products in a register. Sec ond, instead of shifting the multiplicand to the left, the partial product is shifted to the right, which results in leaving the partial product and the multiplicand in the required relative positions. Third, when the corresponding bit of the multiplier is 0, there is no need to add all zeros to the partial product since it will not alter its value.

Figure 10-5 Hardware for multiply operation.

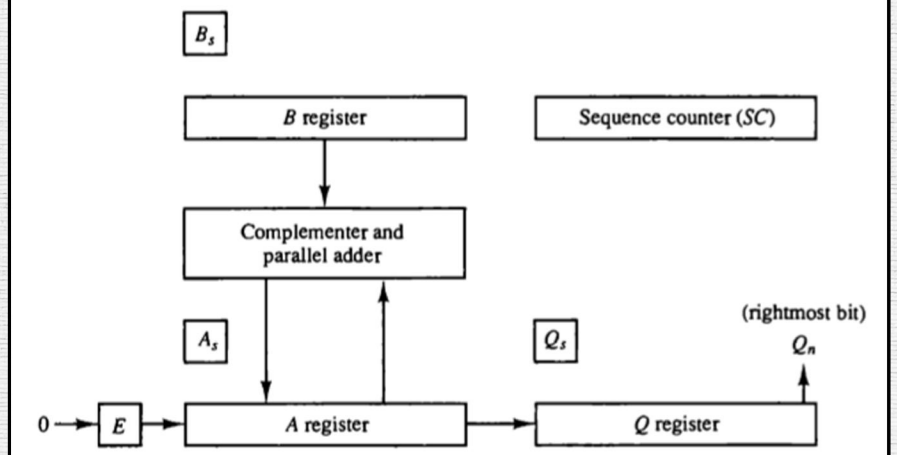
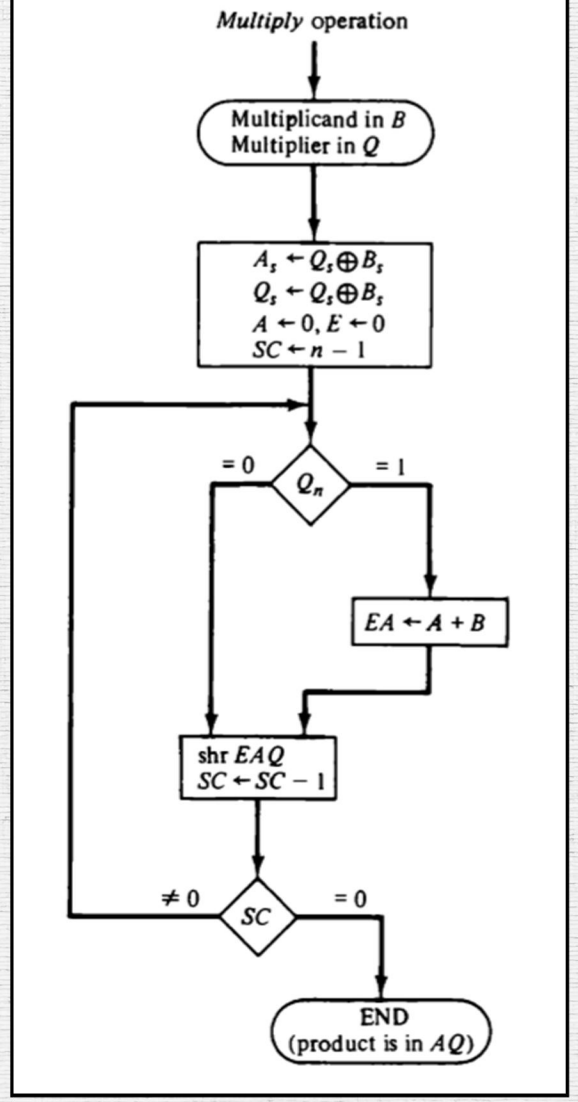


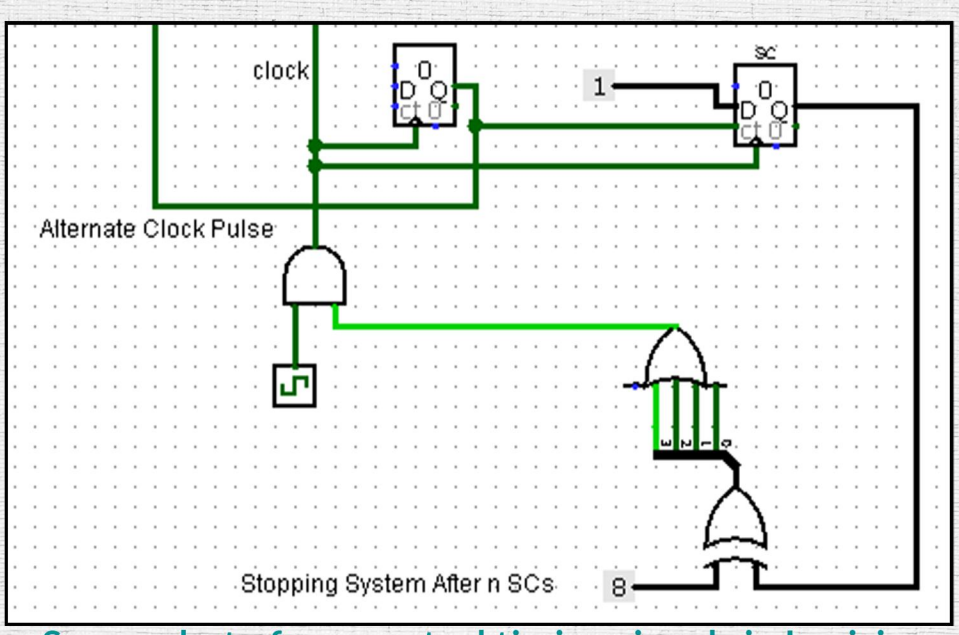
Figure 10-6 Flowchart for multiply operation.



Initially, the multiplicand is in register B and the multiplier in Q. The sum of A and B forms a partial product which is transferred to the EA register. Both partial product and multiplier are shifted to the right. This shift will be denoted by the statement shr EAQ. least significant bit of A is shifted into the most significant position of Q, the bit from E is shifted into the most significant position of A, and 0 is shifted into E. After the shift, one bit of the partial product is shifted into Q, pushing the multiplier bits one position to the right. In this manner, the rightmost

After the initialization, the low-order bit of the multiplier in Q, is tested. If it is a 1, the multiplicand in B is added to the present partial product in A. If it is a 0, nothing is done. Register EAQ is then shifted once to the right t form the new partial product.

IMPLEMENTATION IN LOGISIM

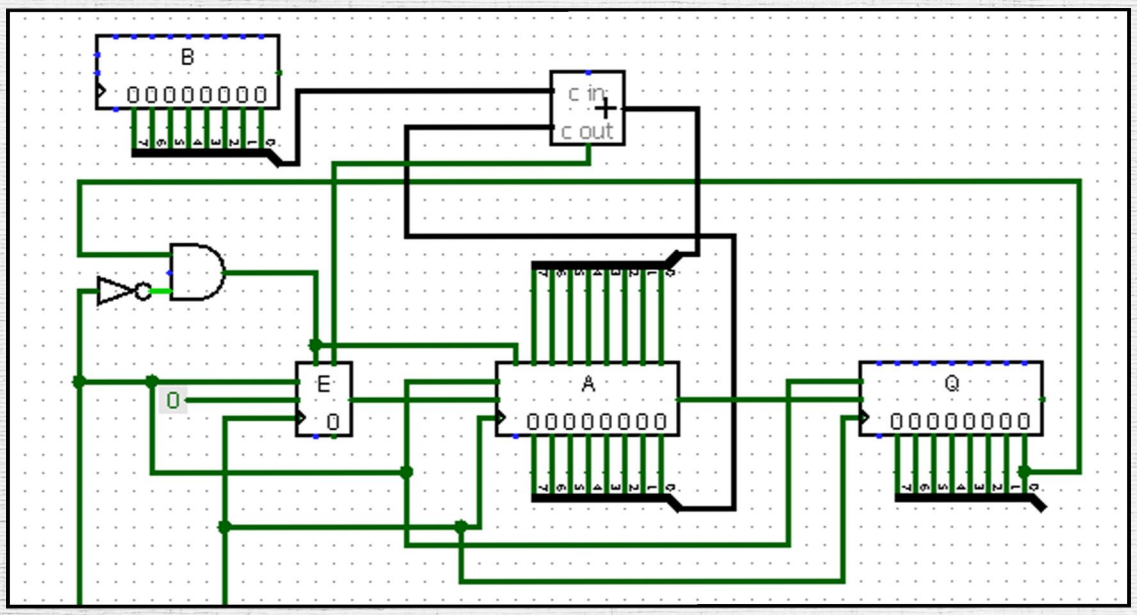


Screenshot of our control timing signals in Logisim

This is a part of our logisim implementation. There is 2 main Parts :

1 - There is sequence counter which will increase the count and when it becomes to 8 our clock pulse will be deactivated because we are performing multiply operation on 8 bit numbers.

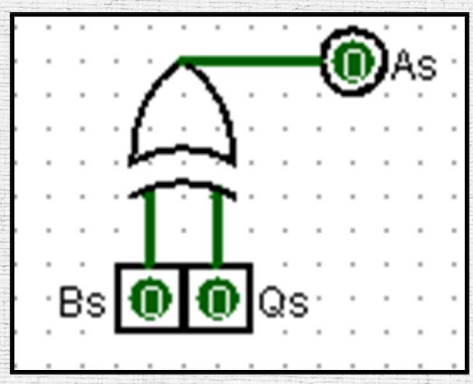
2- We are generating to clock pulse (direct and alternate), out of which alternate clock pulse will trigger the ShR operation and direct clock pulse will trigger load operation one by one.



Screenshot of our main unit for multiplication in Logisim

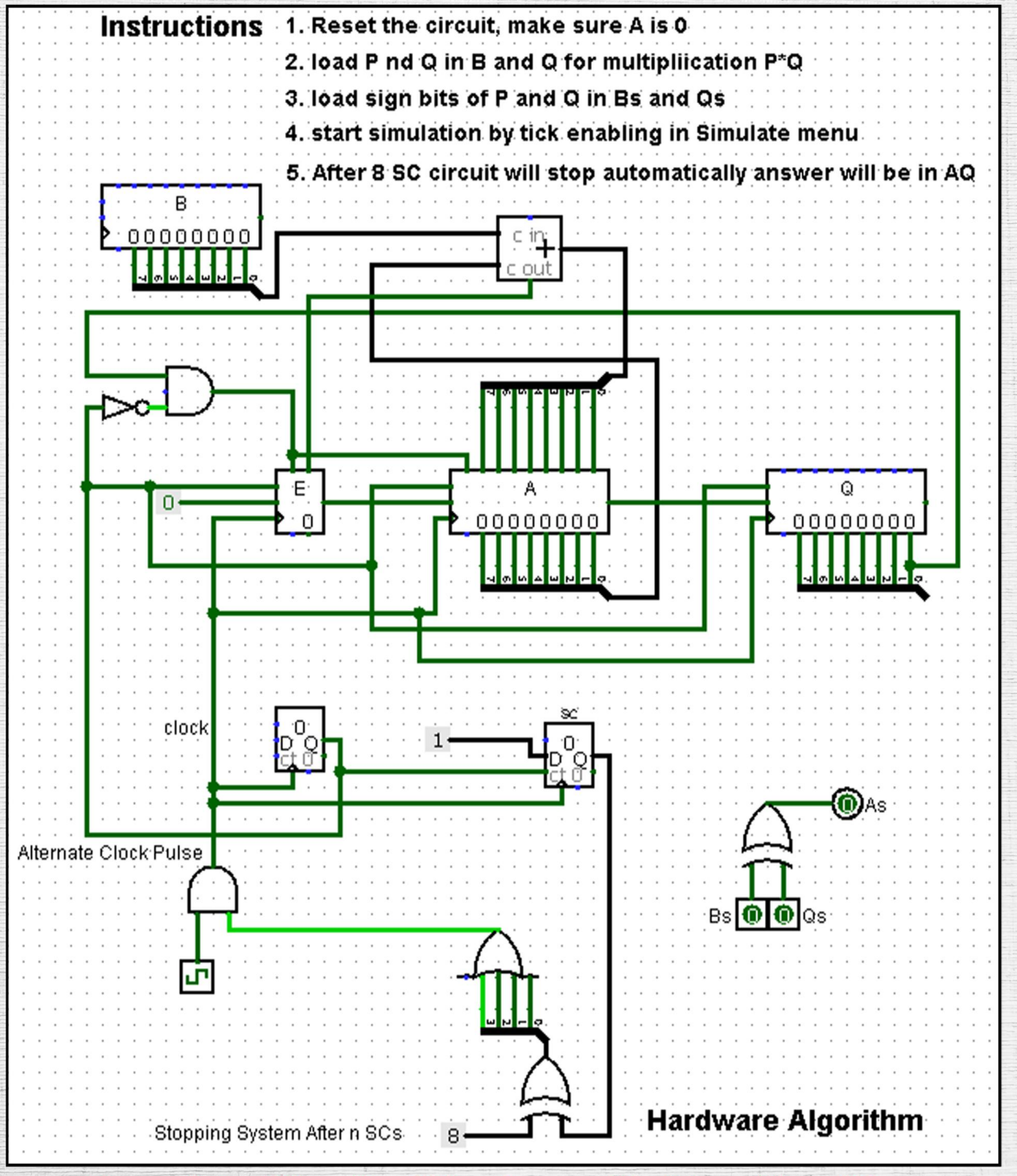
This is our main unit. According to the algorithm if Qn=1, then we will add A & B and will store in EA and if Qn=0 we will do nothing. In the next step with the help of alternate clock pulse we will ShR EAQ. Then the clock pulse will be increased by 1 unit. Our Final Answer will be in AQ.

IMPLEMENTATION IN LOGISIM



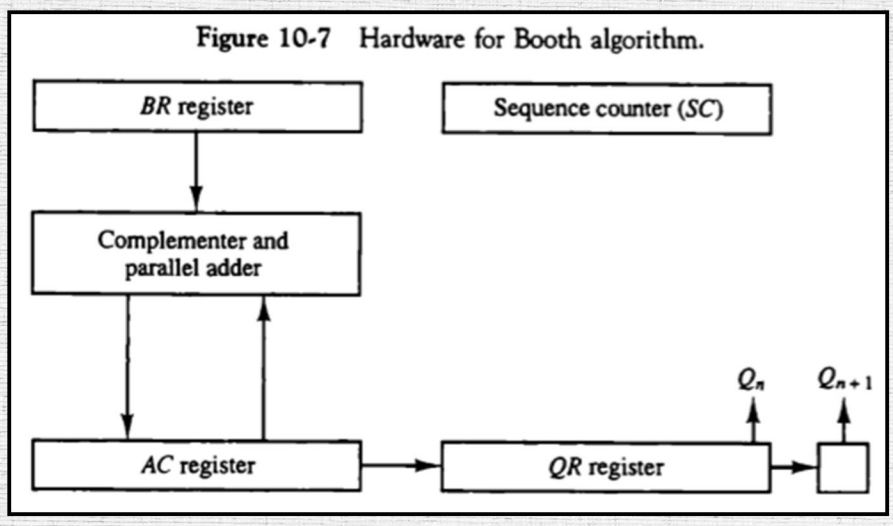
Screenshot of signed bit unit in Logisim

In this unit, we will check the signed bit of both numbers. If XOR of Bs & Qs is equal to 1 then both numbers are of opposite sign and hence our final answer will also be in negative (Signed bit As=1).



Complete Circuit of Hardware Multipy Algorithm

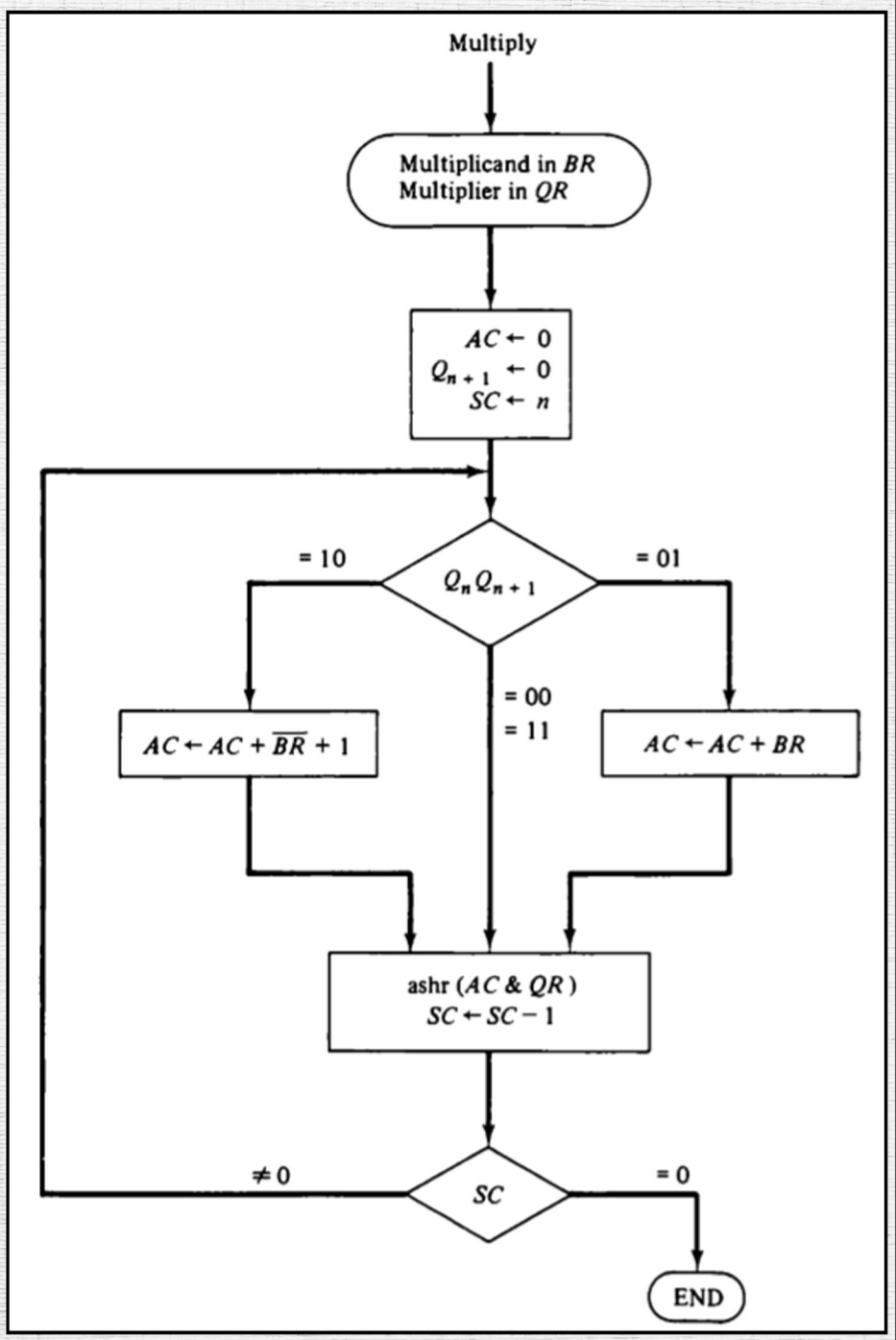
2-BOOTH MULTIPLICATION ALGORITHM



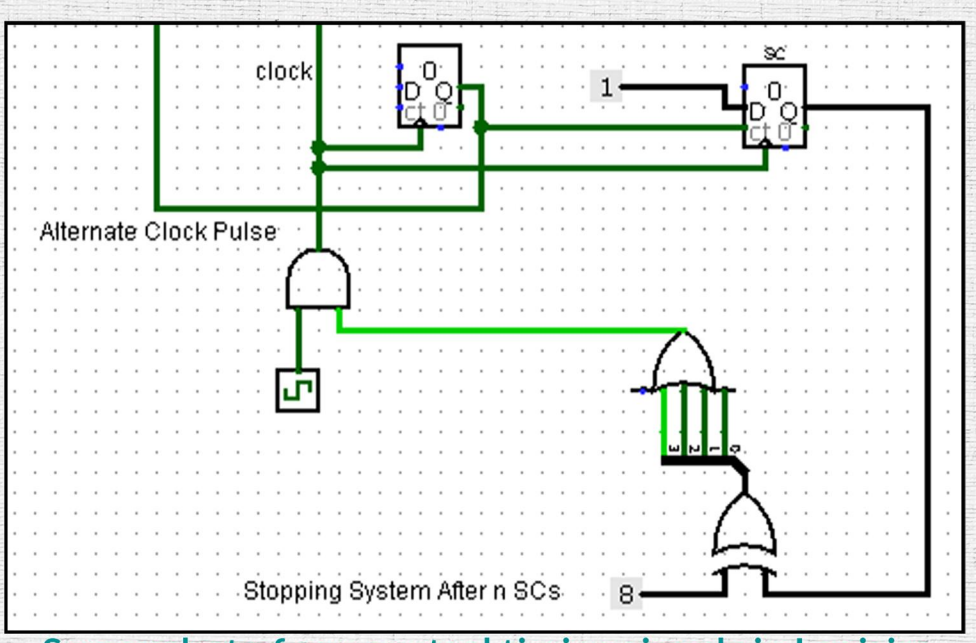
Booth Algorithm gives a procedure for multiplying binary integer in signed-2's complement representation. Hardware implementation is similar to previous one only a new flip flop Qn+1 is added to facilitate a double bit inspection of the multiplier.

Booth algorithm requires examination of the multiplier bits and shifting of the partial product. Prior to the shifting, the multiplicand may be added to the partial product, subtracted from the partial product, or left unchanged according to the following :

1. The multiplicand is subtracted from the partial product upon encountering the first least significant 1 in a string of 1's in the multiplier.
2. The multiplicand is added to the partial product upon encountering the first 0 (provided that there was a previous 1) in a string of 0's in the multiplier.
3. The partial product does not change when the multiplier bit is identical to the previous multiplier bit.



IMPLEMENTATION ON LOGISIM

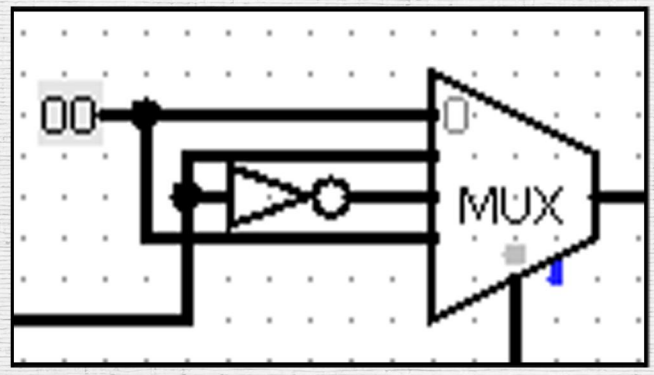


Screenshot of our control timing signals in Logisim

This is a part of our logisim implementation. There is 2 main Parts :

1 - There is sequence counter which will increase the count and when it becomes to 8 our clock pulse will be deactivated because we are performing multiply operation on 8 bit numbers.

2- We are generating to clock pulse (direct and alternate), out of which alternate clock pulse will trigger the ShR operation and direct clock pulse will trigger load operation one by one.

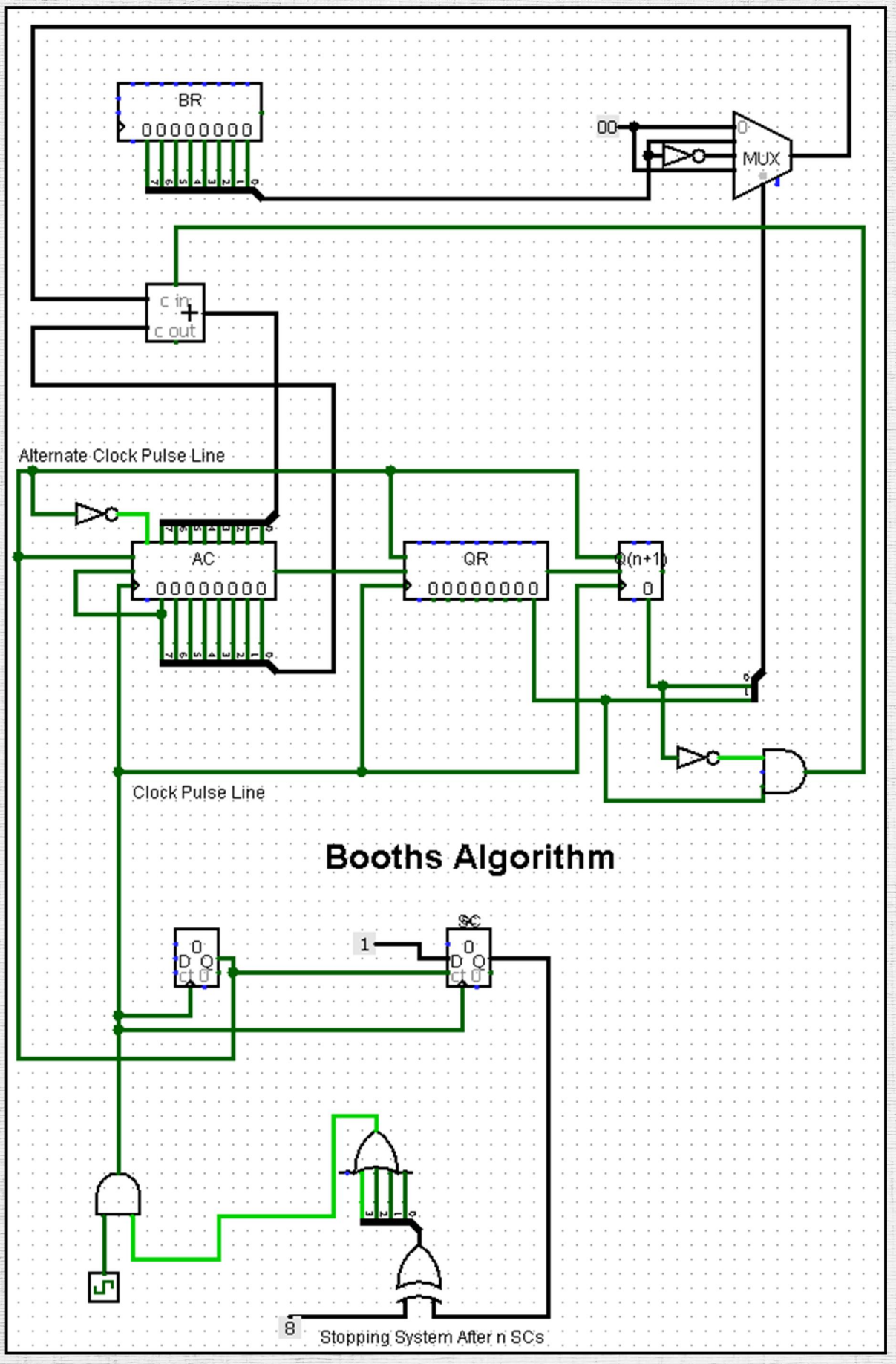


Screenshot of multiplexer Unit

According to algorithm this multiplexer will select one of three operation that need to be performed.

S1 S2	DESCRIPTION
0 0	Nothing will Happen.
0 1	Addition of BR and AC
1 0	Subtraction of AC and BR
1 1	Nothing will Happen

IMPLEMENTATION ON LOGISIM



Complete Circuit for Booth Multiplication Algorithm

EXAMPLES

Here, B = 23 = 10111 , Q = 19 = 10011 , A =0

B x Q = 23 x 19 = 437

Answer = AQ = 0110110101 =437

TABLE 10-2 Numerical Example for Binary Multiplier

Multiplicand $B = 10111$	E	A	Q	SC
Multiplier in Q	0	00000	10011	101
$Q_n = 1$; add B		10111		
First partial product	0	10111		
Shift right EAQ	0	01011	11001	100
$Q_n = 1$; add B		10111		
Second partial product	1	00010		
Shift right EAQ	0	10001	01100	011
$Q_n = 0$; shift right EAQ	0	01000	10110	010
$Q_n = 0$; shift right EAQ	0	00100	01011	001
$Q_n = 1$; add B		10111		
Fifth partial product	0	11011		
Shift right EAQ	0	01101	10101	000
Final product in AQ = 0110110101				

Here, BR = (-9) = 10111 , QR = (-13) = 10011 , AC =0 , Q(n+1)=0

BR x QR = (-9) x (-13) = 117

Answer = (AC)(QR) = 0001110101 =117

TABLE 10-3 Example of Multiplication with Booth Algorithm

Q_n Q_{n+1}	$BR = 10111$ $\overline{BR} + 1 = 01001$	AC	QR	Q_{n+1}	SC
	Initial	00000	10011	0	101
1 0	Subtract BR	01001			
		01001			
	ashr	00100	11001	1	100
1 1	ashr	00010	01100	1	011
0 1	Add BR	10111			
		11001			
	ashr	11100	10110	0	010
0 0	ashr	11110	01011	0	001
1 0	Subtract BR	01001			
		00111			
	ashr	00011	10101	1	000