GTU Computer Science Engineering
CSE 331/ Fall 2020
Homework #3 Report

Elif Keleş
161044033

In this assignment, I was expected to make an Unsigned Number Multiplier program on Logisim, that multiplies two 32 Bit unsigned numbers as described in the flow chart in the homework PDF.

As requested, I created a file called mult32 which uses my control unit and the datapath.

First I created an FSM. I have eight states and related signals. (As shown at the end of this document)

With the present states and signals, I formed my next states expressions with AND and OR gates in control unit. And lastly, I formed the signals that will be going to datapath according to the states.

All of my multiplexers, registers, adder, shifter, inputs and outputs etc are in the datapath. As stated, I used only one adder and one shifter for my circuit. I used the logisim's adder and shifter. (I did not design my own)

First I initialized my multiplier and multiplicand values with input values. To keep the values, I used registers. For the product and multiplier, I used a Mux to select input value, considering the values changes with shifting.

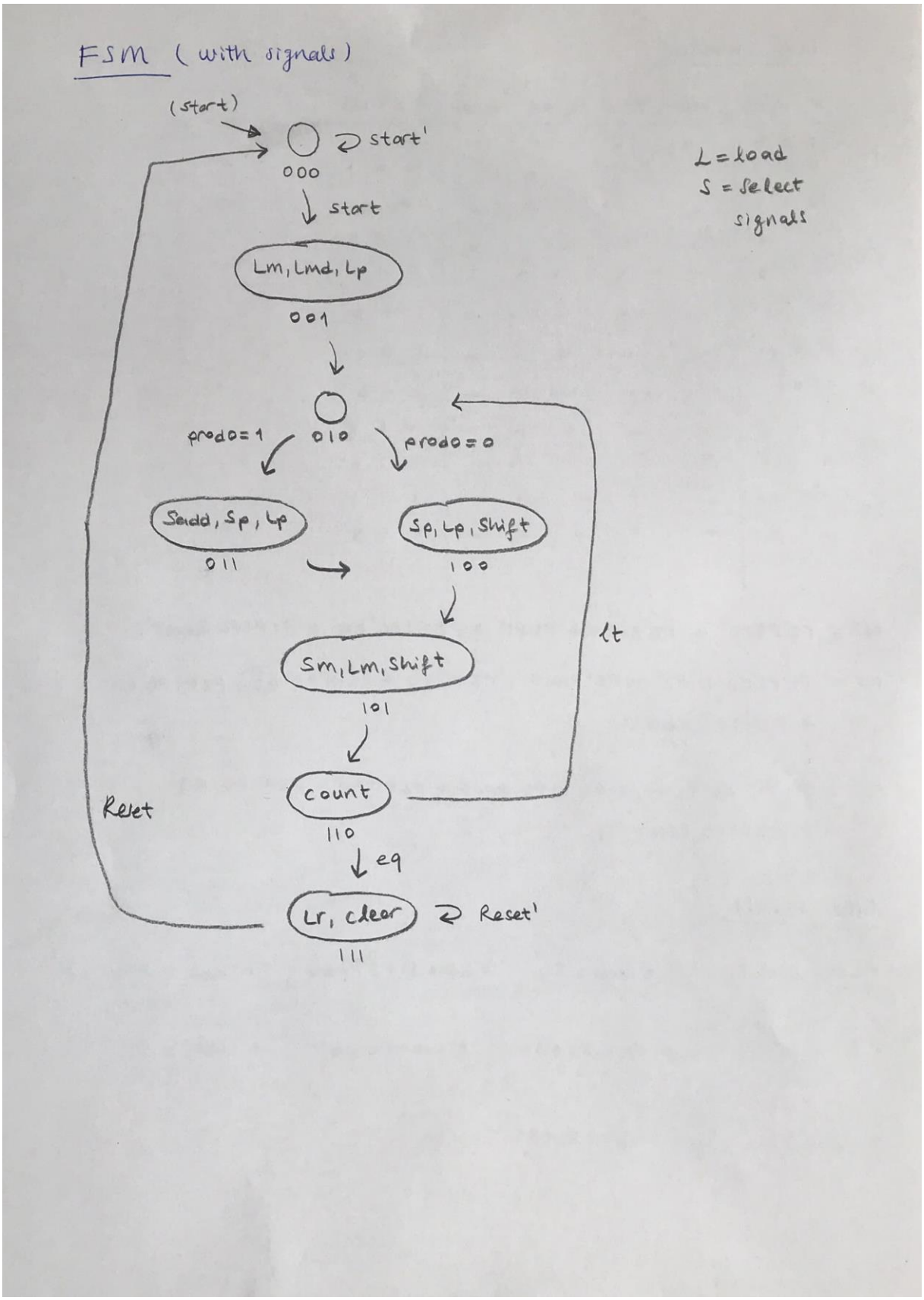
The compelling part of the homework for me, was designing the 64 Bit shifting. For that, I used separators, and kept the product's shifted bit (lsb in the [64-32] bits) in a register. Than made sure that the shifted bit is located in multiplier's msb.

Also, I had two 32 Bit numbers for 64 Bit product. So, to be able to shift the number correctly with one shifter, I kept the states for shifting, apart in two steps.

I used an unsigned comparator and a counter to keep the number of repetition.

To control the circuit easily, I added a start and a reset button.

My FSM and signal designs



State Changes

	P2 P1 P0	start	Reset	lt	eq	prod0	N2 N1 N0
S0	0 0 0	0	-	-	-	-	0 0 0
	0 0 0	1	-	-	-	-	0 0 1
S1	0 0 1	-	-	-	-	-	0 1 0
S2	0 1 0	-	-	-	-	0	1 0 0
	0 1 0	-	-	-	-	1	0 1 1
S3	0 1 1	-	-	-	-	-	1 0 0
S4	1 0 0	-	-	-	-	-	1 0 1
S5	1 0 1	-	-	-	-	-	1 1 0
S6	1 1 0	-	-	1	-	-	0 1 0
	1 1 0	-	-	-	1	-	1 1 1
S7	1 1 1	-	0	-	-	-	1 1 1
	1 1 1	-	1	-	-	-	0 0 0

$N2 = P2'P1P0' + P2'P1P0 + P2P1' + P2P1P0'eq + P2P1P0Reset'$

$N1 = P2'P1'P0 + P2'P1P0'prod0 + P2P1'P0 + P2P1P0'lt + P2P1P0'eq + P2P1P0Reset'$

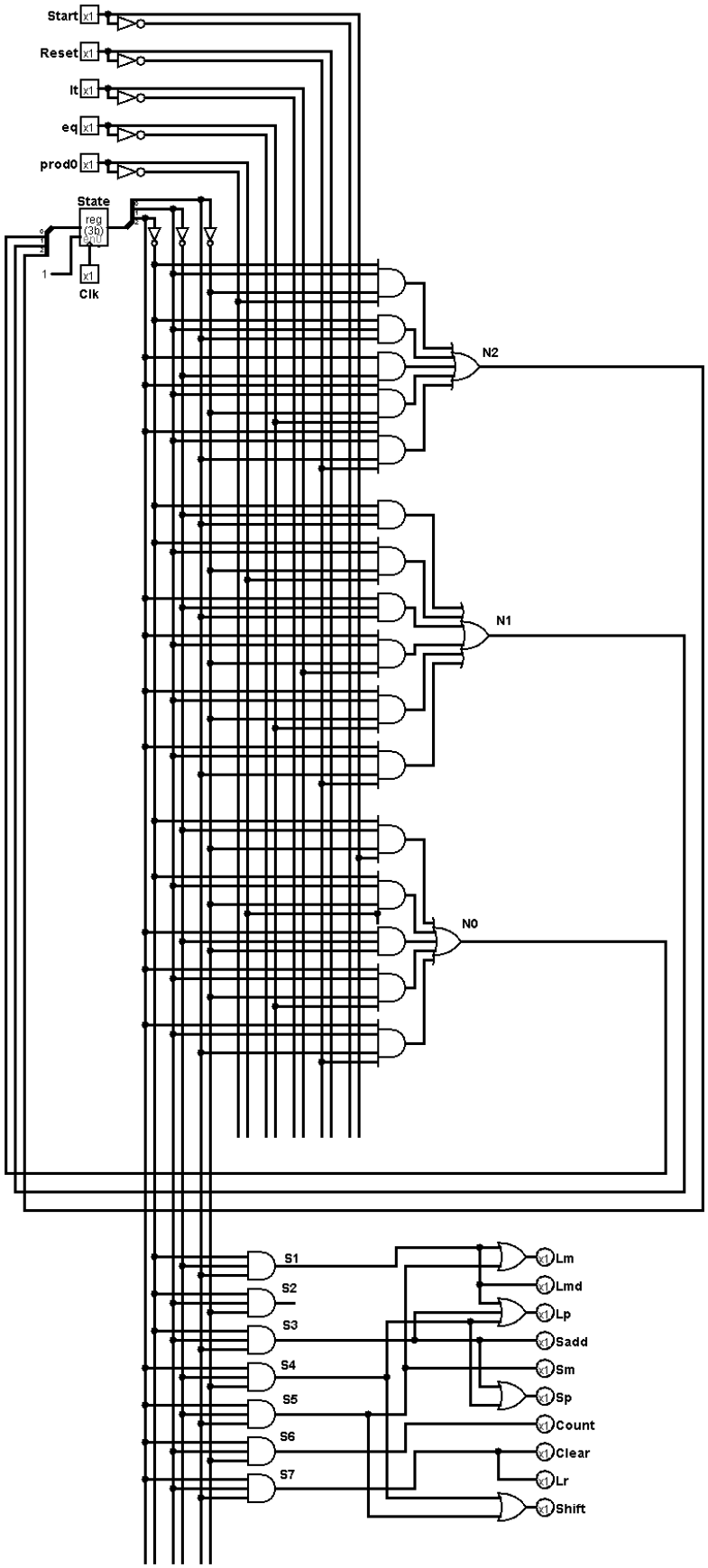
$N0 = P2'P1'P0'start + P2'P1P0'prod0 + P2P1'P0' + P2P1P0'eq + P2P1P0Reset'$

Other signals

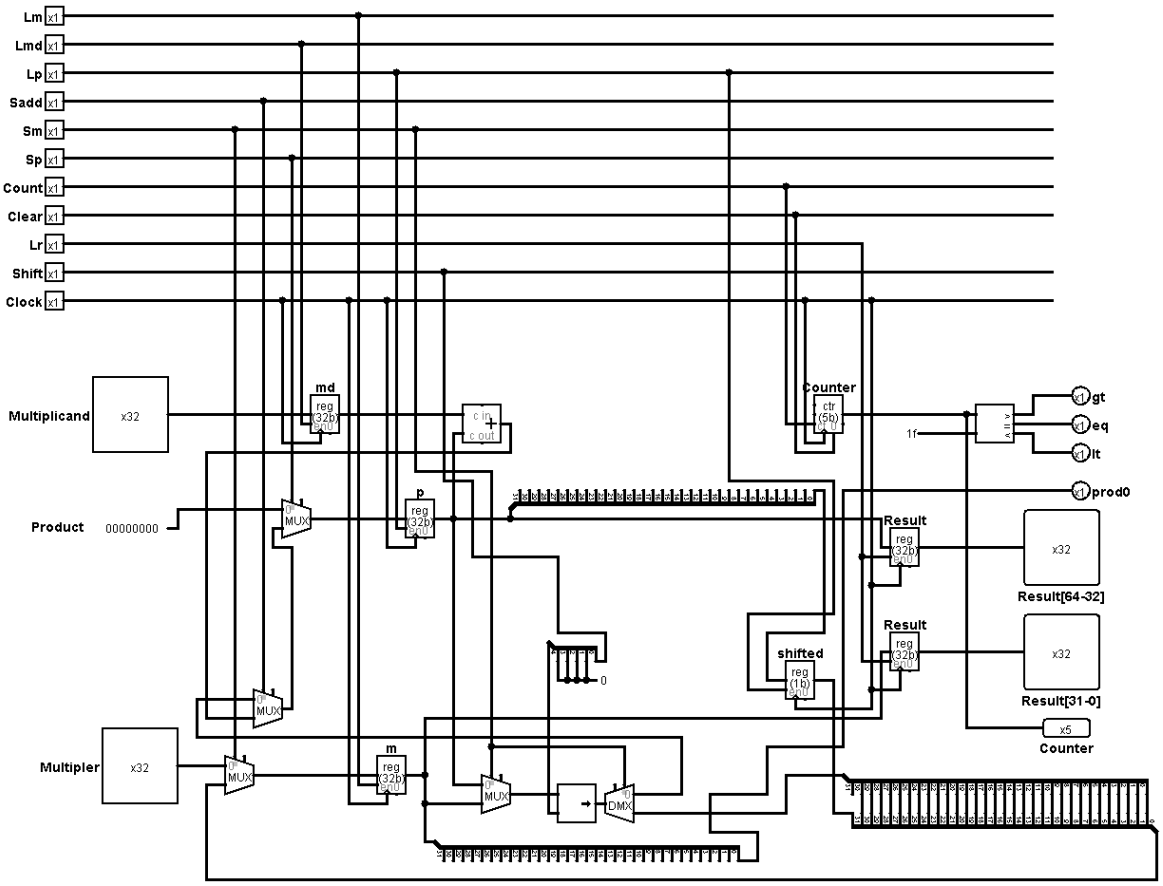
- $Lm = S1 + S5$
- $Lmd = S1$
- $Lp = S1 + S3 + S4$
- $Sadd = S3$
- $Sm = S5$
- $Sp = S3 + S4$
- $Count = S6$
- $Clear = S7$
- $Lr = S7$
- $Shift = S4 + S5$

Screenshots of the Circuit

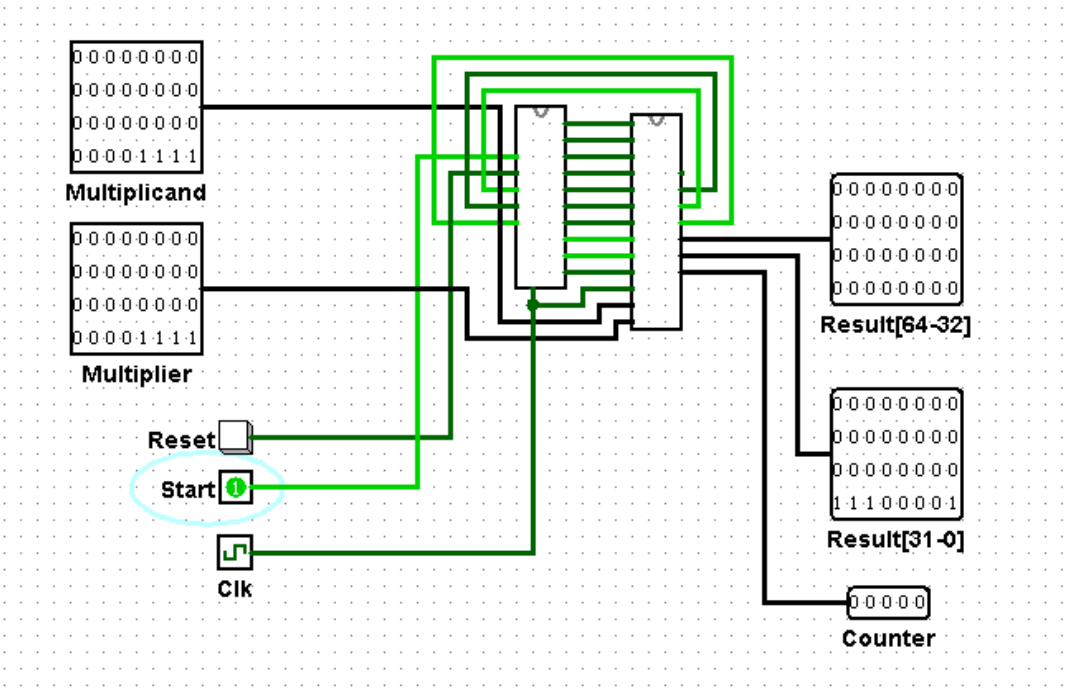
Control Unit

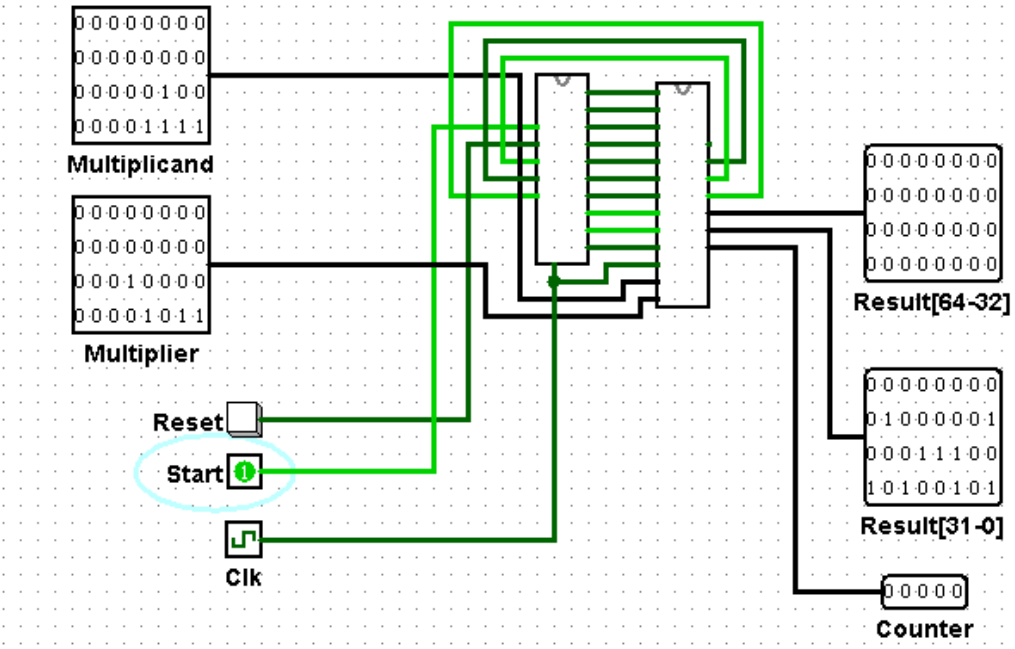


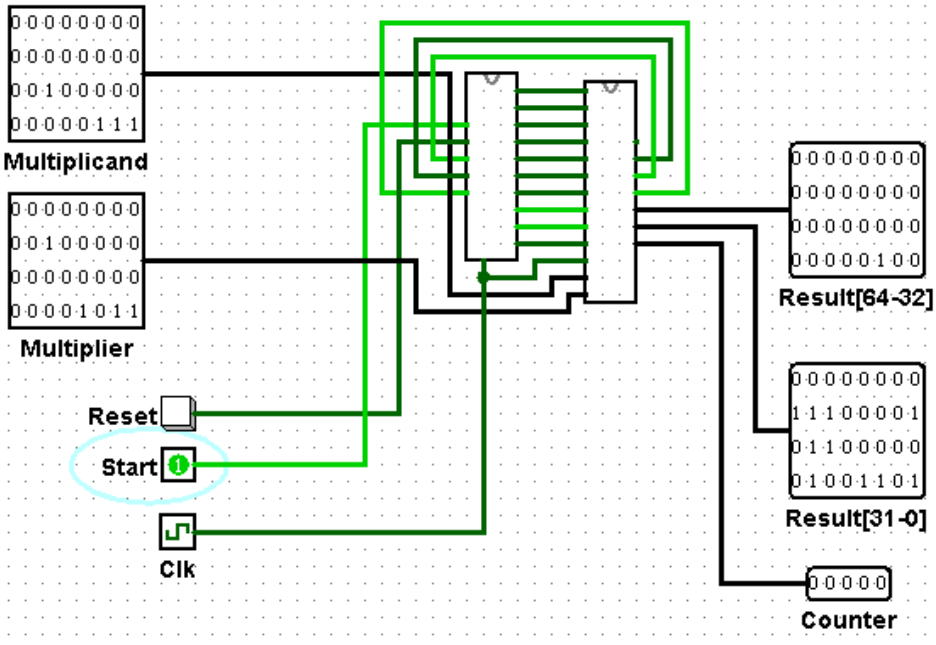
Datapath

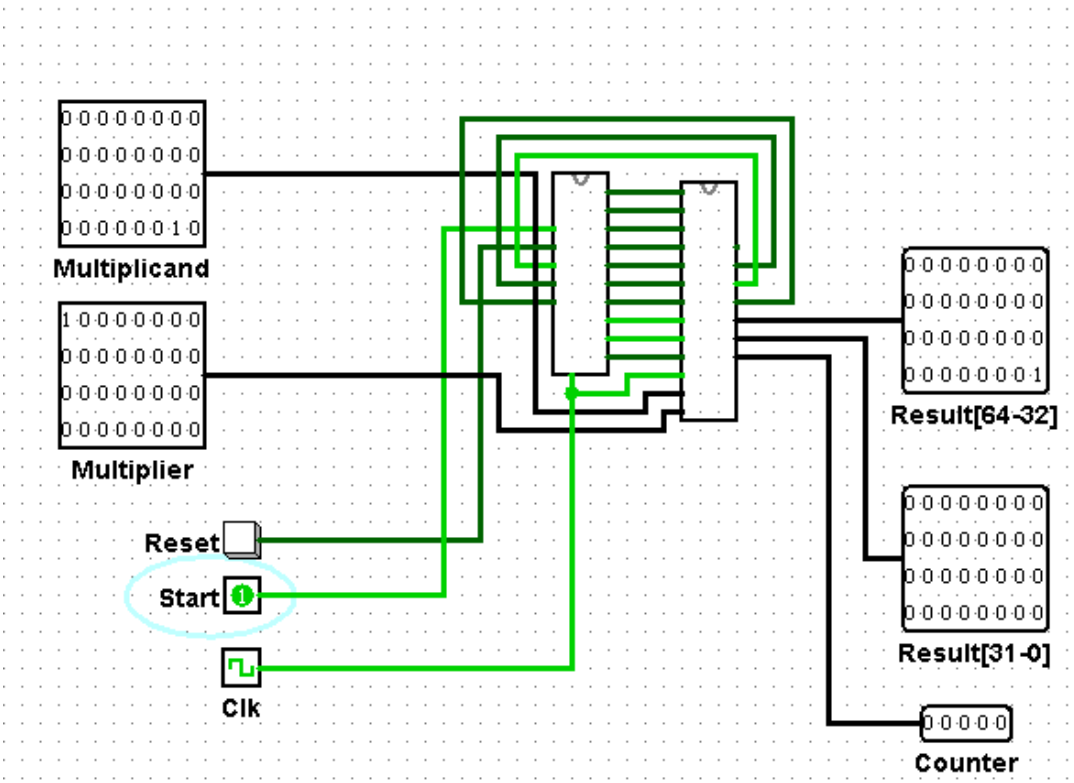


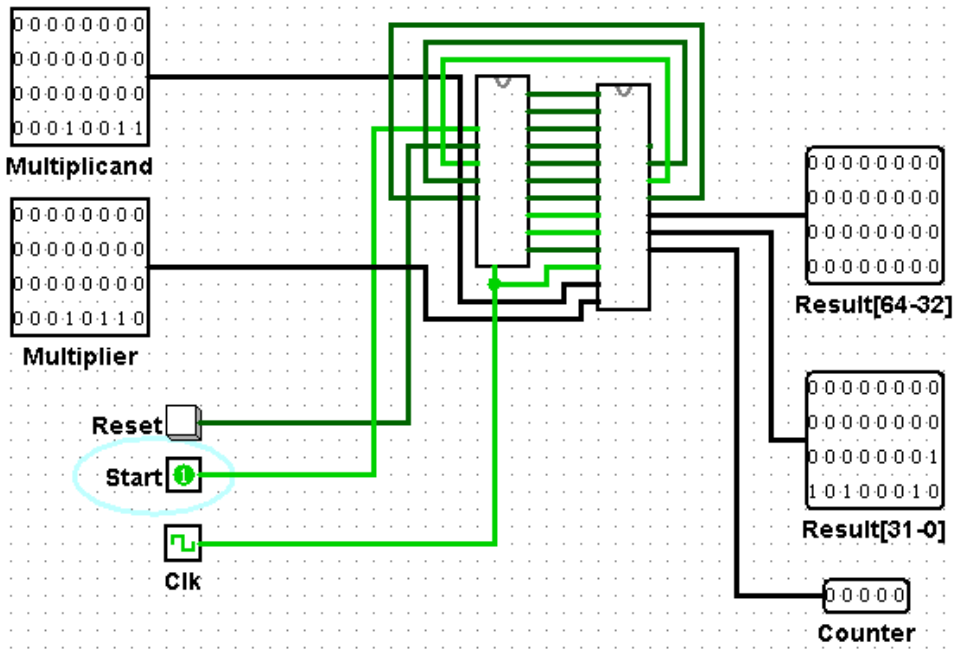
Tests for some cases

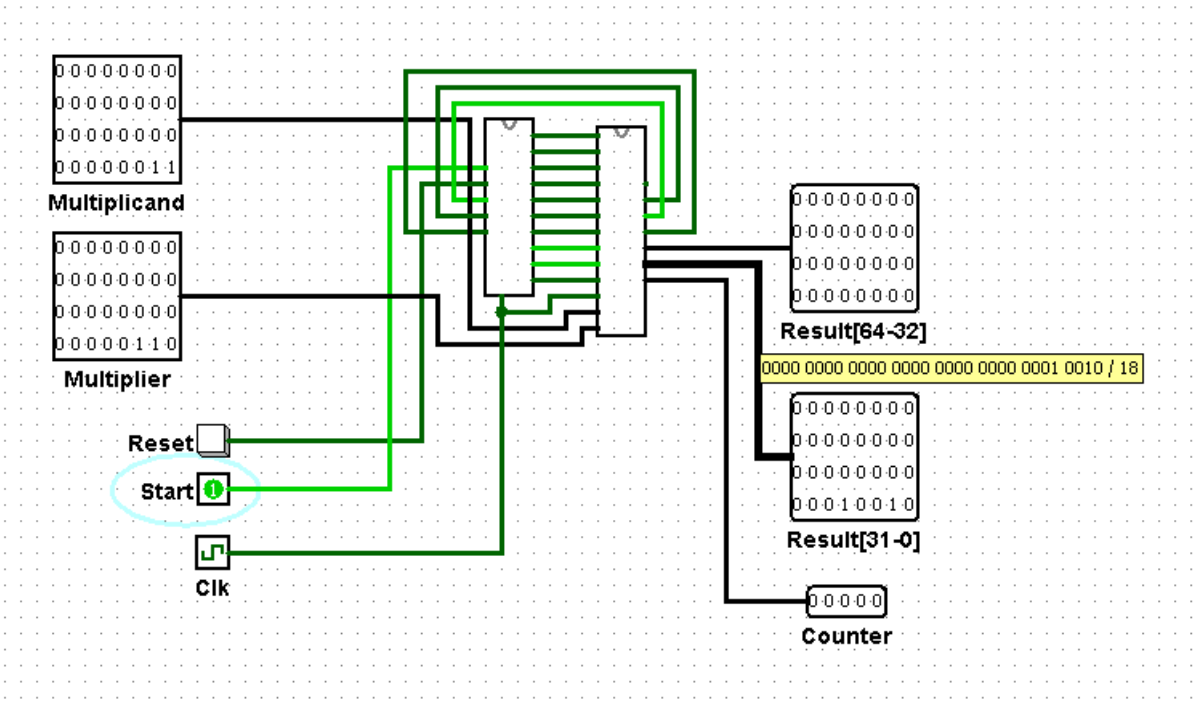



0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 1 1 1 1
Multiplicand
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 1 0 1 1
Multiplier
Reset
Start
Clk
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
Result[64-32]
0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 1
0 0 0 1 1 1 0 0
1 0 1 0 0 1 0 1
Result[31-0]
0 0 0 0 0
Counter


0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 0 0 0 1 1 1
Multiplicand
0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 1 0 1 1
Multiplier
Reset
Start
Clk
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0
Result[64-32]
0 0 0 0 0 0 0 0
1 1 1 0 0 0 0 1
0 1 1 0 0 0 0 0
0 1 0 0 1 1 0 1
Result[31-0]
0 0 0 0 0
Counter


Multiplicand
Multiplier
Reset
Start
Clk
Result[64-32]
Result[31-0]
Counter


Multiplicand
Multiplier
Reset
Start
Clk
Result[64-32]
Result[31-0]
Counter


0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1
Multiplicand
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 0
Multiplier
Reset
Start
Clk
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
Result[64-32]
0000 0000 0000 0000 0000 0000 0001 0010 / 18
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 1 0 0 1 0
Result[31-0]
0 0 0 0 0
Counter