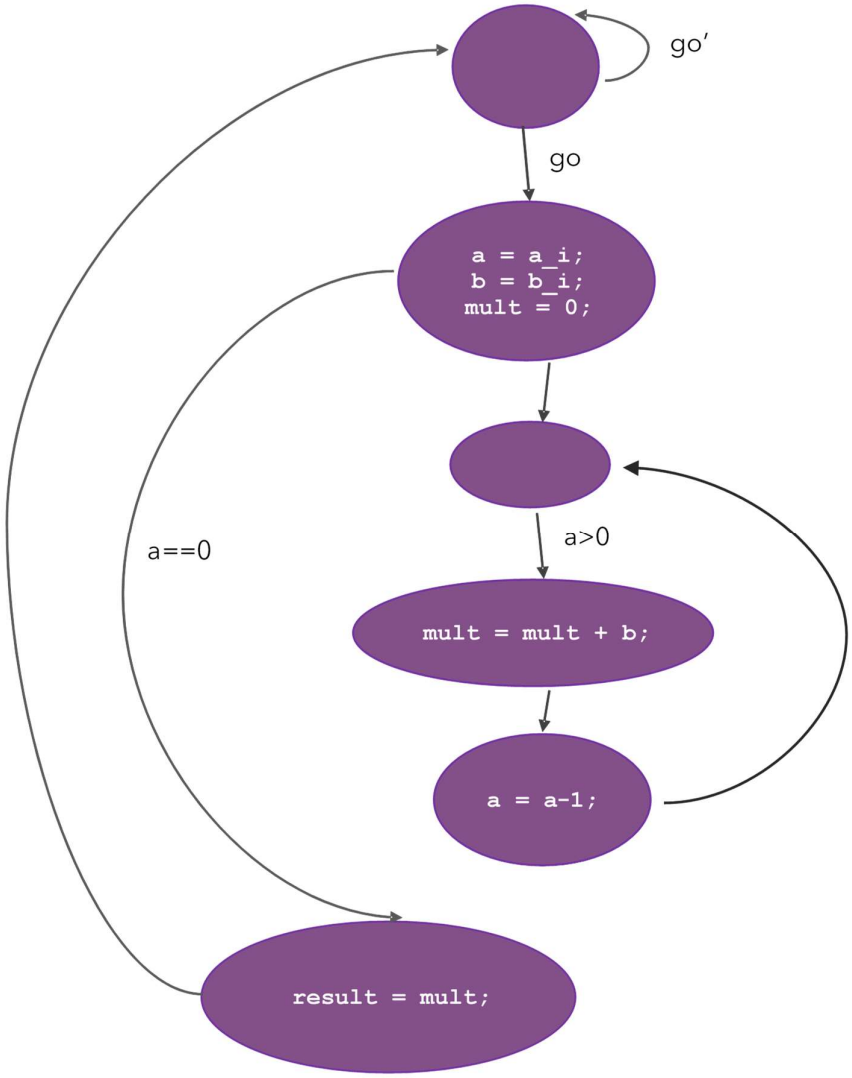
GTU Department of Computer Engineering
CSE 232 - Spring 2020

PROJECT 2 REPORT

ESRA ERYILMAZ
171044046

1. Decide states and draw the state diagram for your FSM controller.

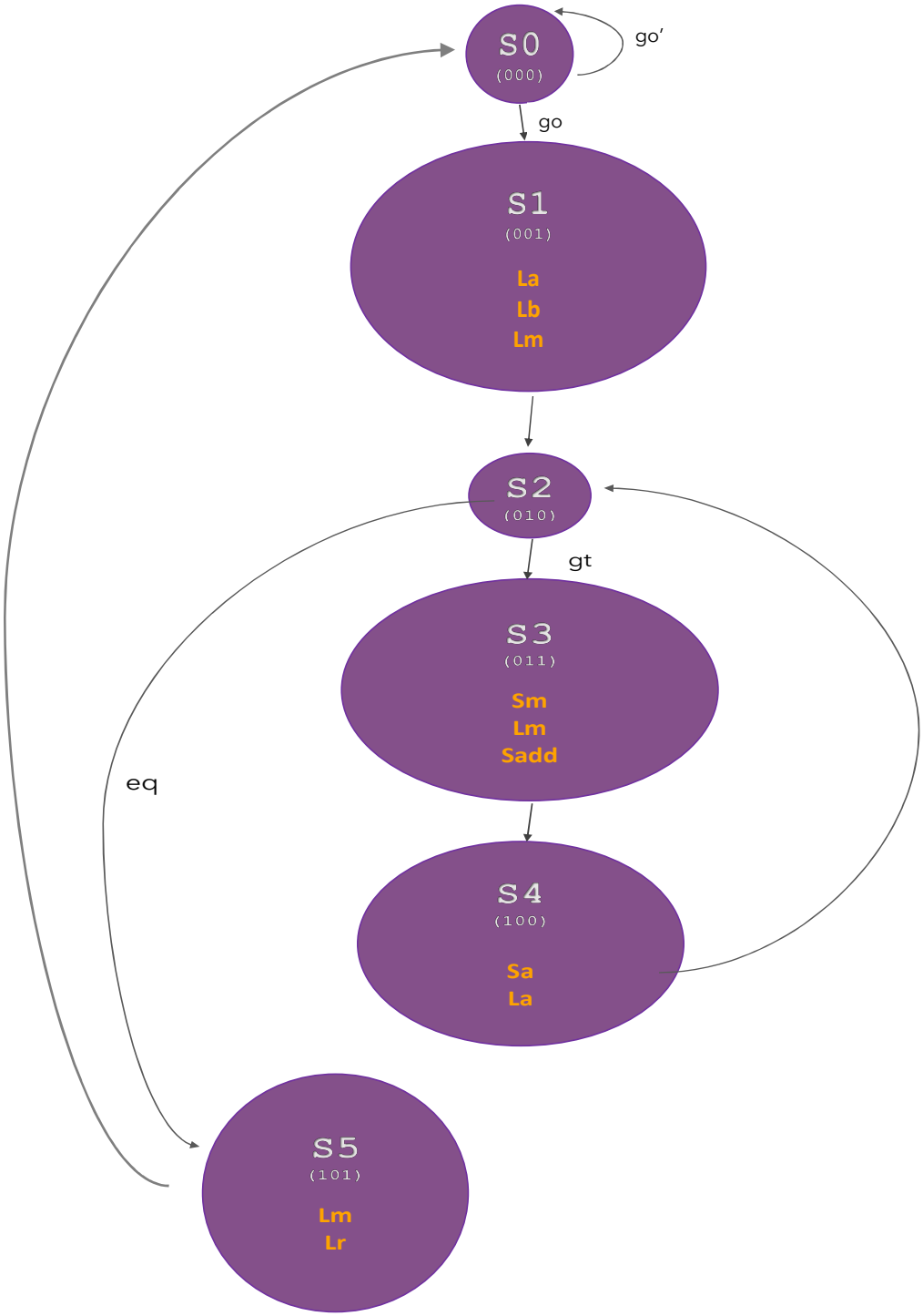


- I get inputs (a_i and b_i) from user and return the multiplication result.
- User should also press "**go**" button once after entering the inputs.

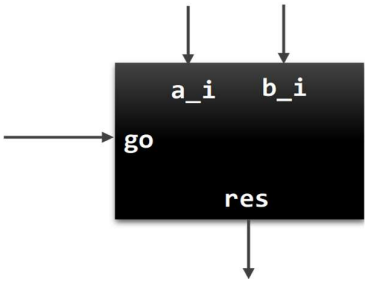
(Since I used a single adder, I didn't put "mult = mult+b" and "a = a-1" with the same state.)
(I didn't need to write state for a < 0 because I made the signs positive at the beginning.)

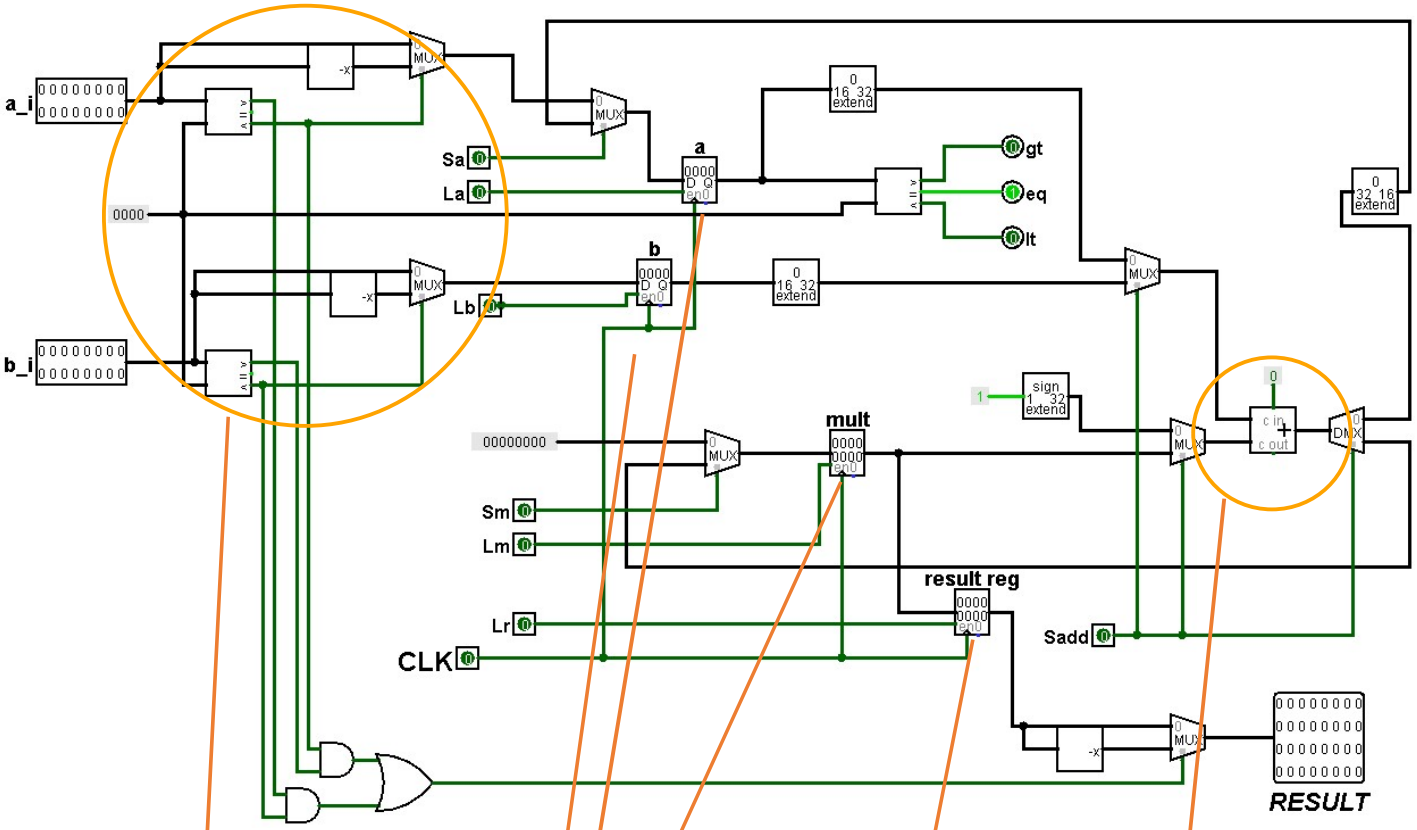
Update FSM

(I accept '1' for signals written in the state , and for all the remaining signals '0'.)



2. Draw datapath.





(a ,b and mult registers)

(register for not to lose the result.)

(This is for handling negative numbers.)

(I used one adder for addition and subtraction.)

3. Draw truth table.

I have 6 states so 3 bit is enough.

Present States (PS) P2 P1 P0		Inputs gt eq go	Next States (NS) N2 N1 N0
S0	0 0 0 0 0 0	- - 0 - - 1	0 0 0 0 0 1
S1	0 0 1	- - -	0 1 0
S2	0 1 0 0 1 0	1 0 - 0 1 -	0 1 1 1 0 1
S3	0 1 1	- - -	1 0 0
S4	1 0 0	- - -	0 1 0
S5	1 0 1	- - -	0 0 0

Outputs are just depends on states, so I showed it in separate truth table.

Present States (PS) P2 P1 P0		La	Lb	Lm	Sa	Sm	Sadd	Lr
S0	0 0 0	0	0	0	0	0	0	0
S1	0 0 1	1	1	1	0	0	0	0
S2	0 1 0	0	0	0	0	0	0	0
S3	0 1 1	0	0	1	0	1	1	0
S4	1 0 0	1	0	0	1	0	0	0
S5	1 0 1	0	0	1	0	0	0	1

4. Derive Boolean expressions from the truth table.

N2 = P2′P1P0′eq + P2′P1P0

N1 = P2′P1′P0 + P2′P1P0′gt + P2P1′P0′

N0 = P2′P1′P0′go + P2′P1P0′gt + P2′P1P0′eq

La = S1 + S4

Lb = S1

Lm = S1 + S3 + S5

Sa = S4

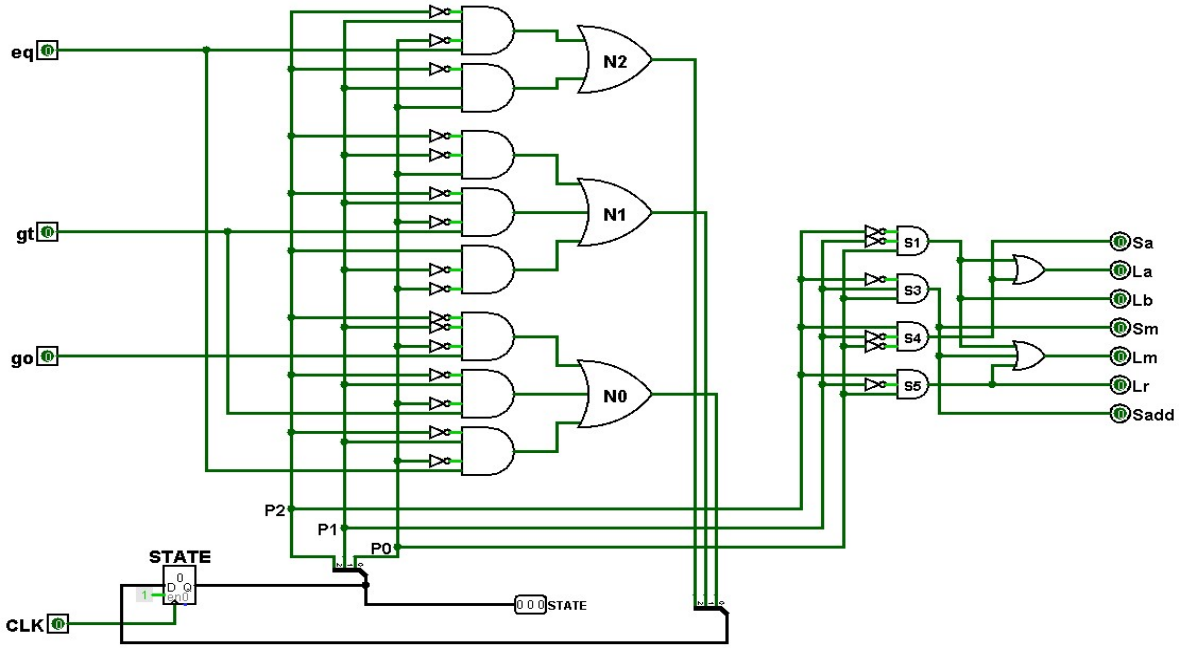
Sm = S3

Sadd = S3

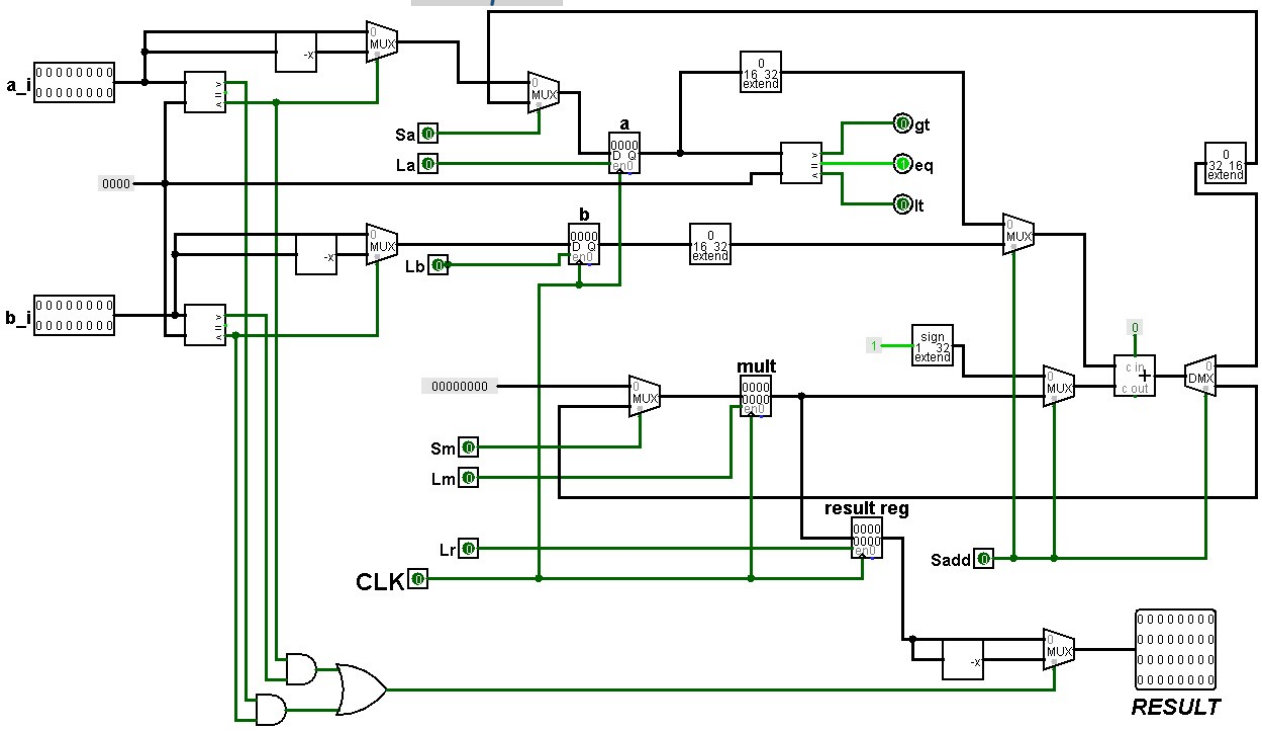
Lr = S5

5. Draw the circuit on Logisim.

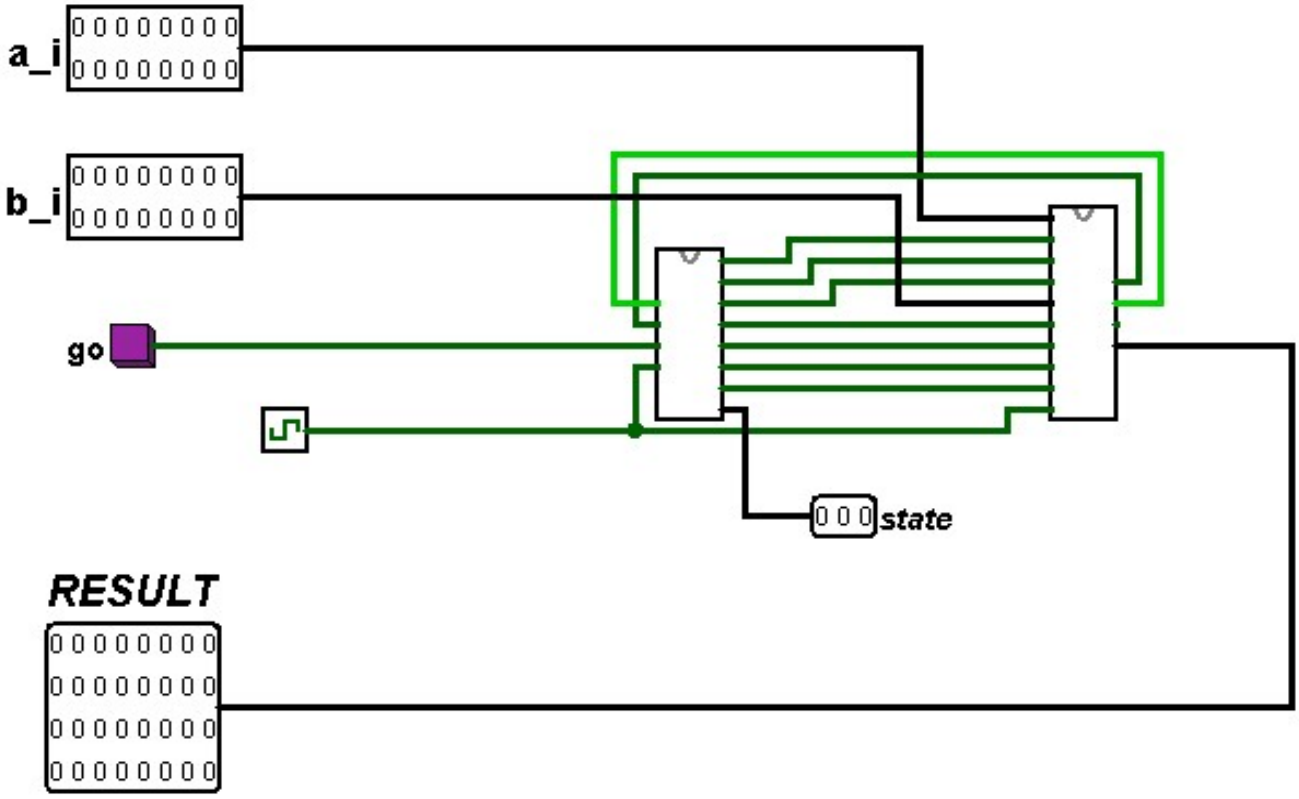
Control Unit



Datapath



Main



Optimizations

- Since I used a single adder, I connected double multiplexer to the input of the adder and I connected one demultiplexer to the output of the adder.
  According to multiplexer select bit (Sadd), state machine decided which action to take.

- After taking the inputs, I check their signs and turn the negatives into positive.
- If the input signs are opposite, I return the result negative.
  ➔ In this way, it works with negatives.

All parts are working.