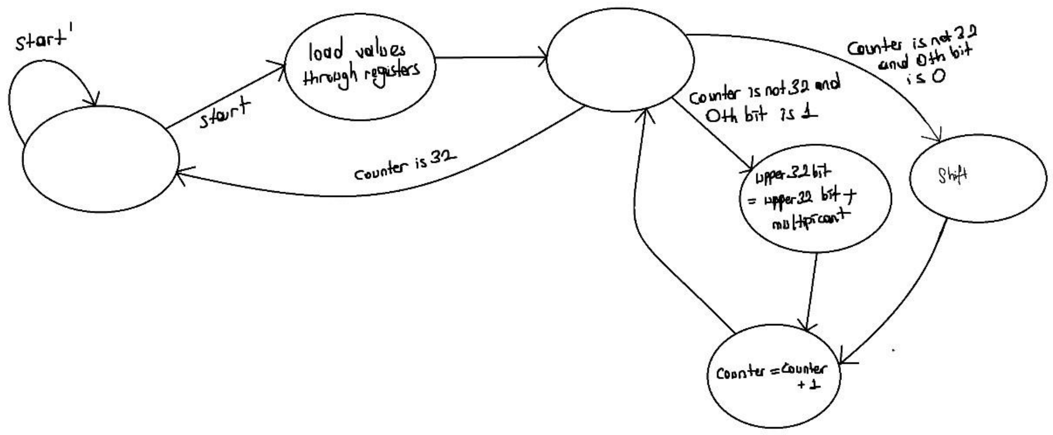
CSE 331 - Computer Organization

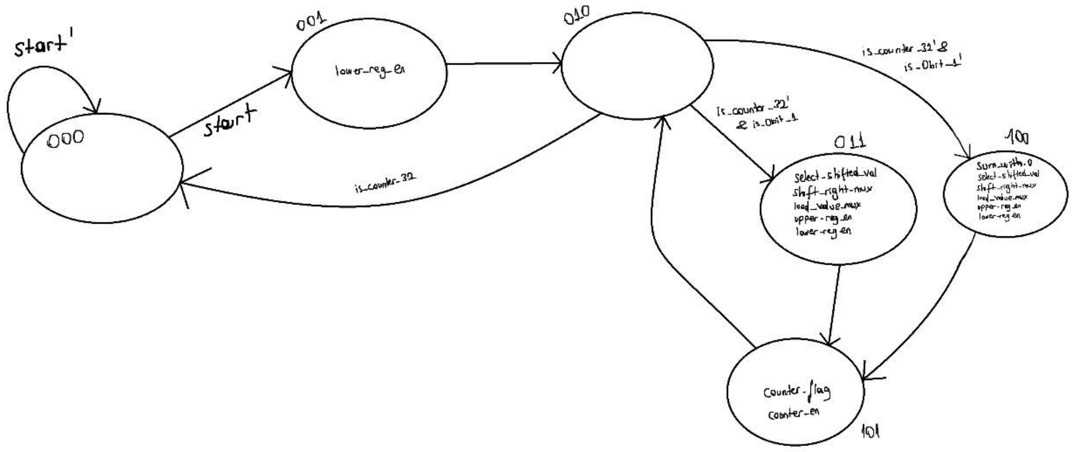
Melihcan Çilek – 1801042092

Assignment 3 Report

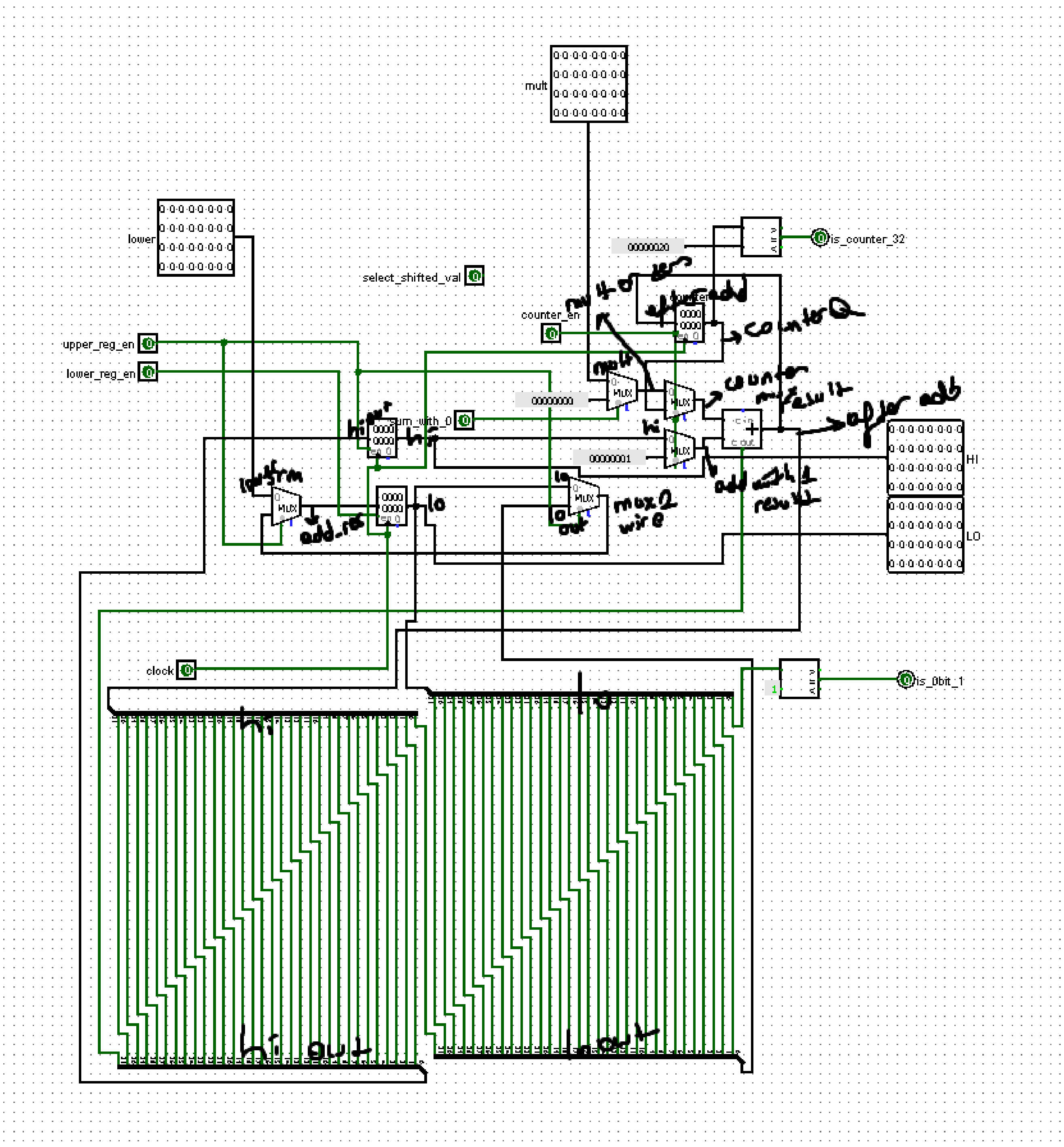
- PART 1:
  - ASM DESIGN OF MULTIPLIER


start'
start
load values through registers
Counter is 32
Counter is not 32 and 0th bit is 0
Counter is not 32 and 0th bit is 1
upper 32 bit = upper 32 bit + multiplicant
Shift
Counter = Counter + 1

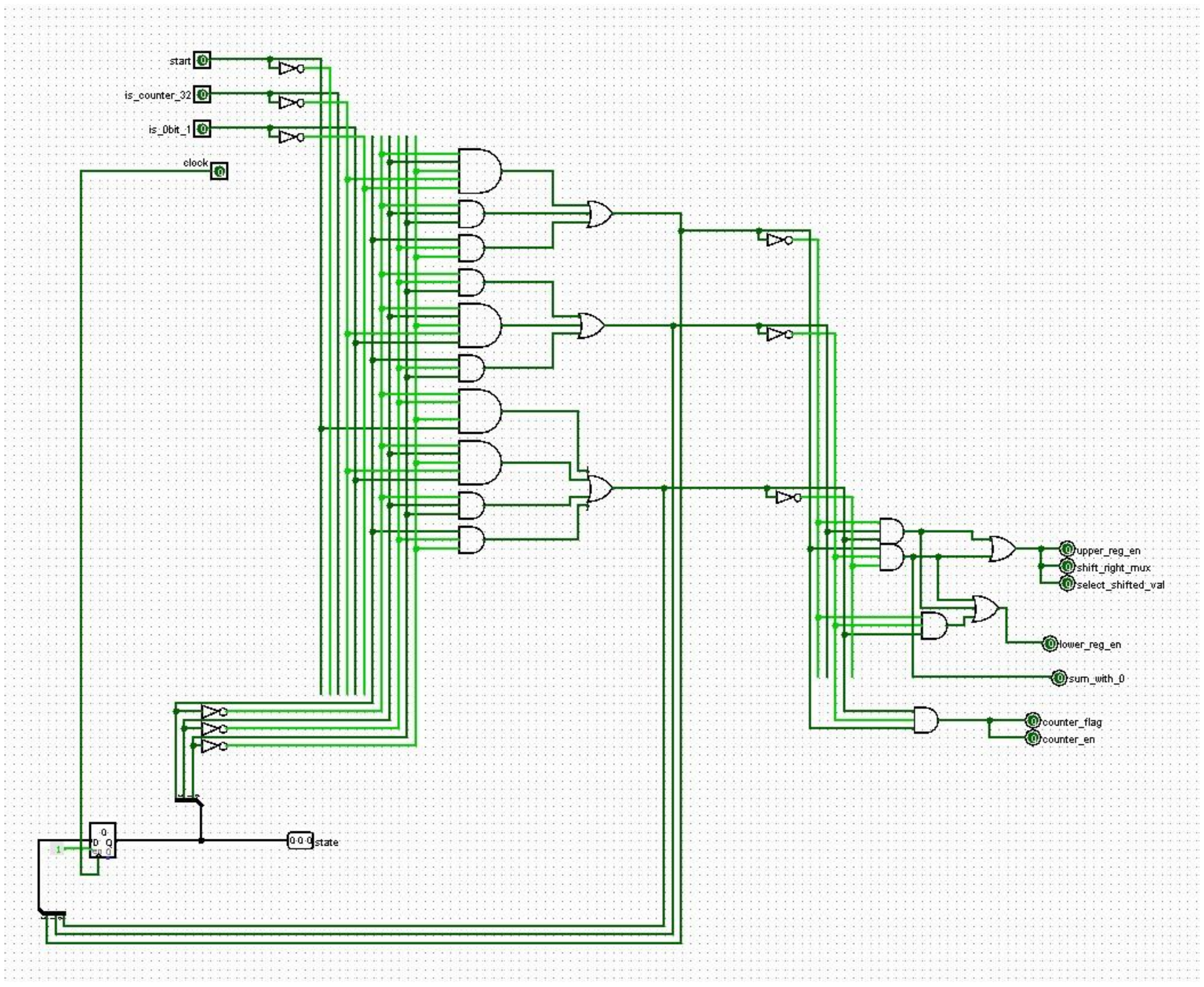
  - FSM DESIGN OF MULTIPLIER



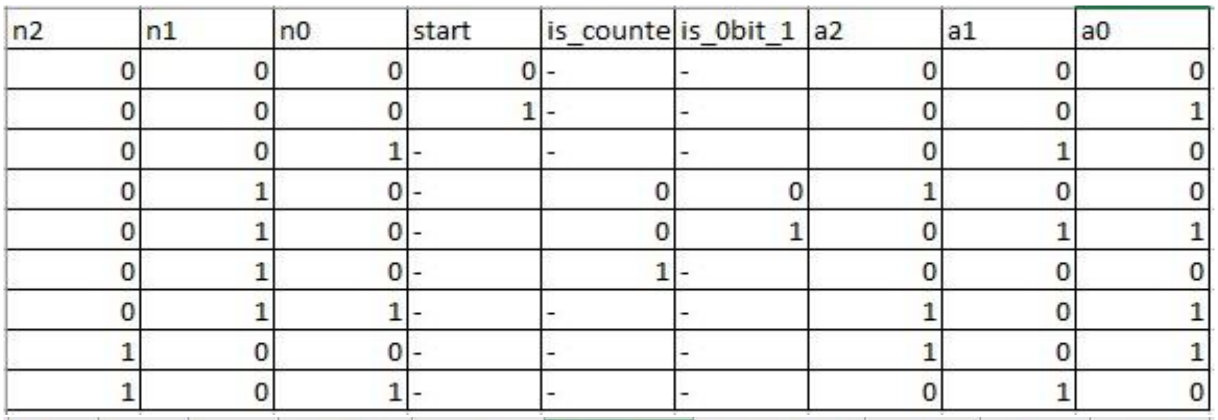
o DATAPATH DESIGN OF MULTIPLIER (LOGISIM)



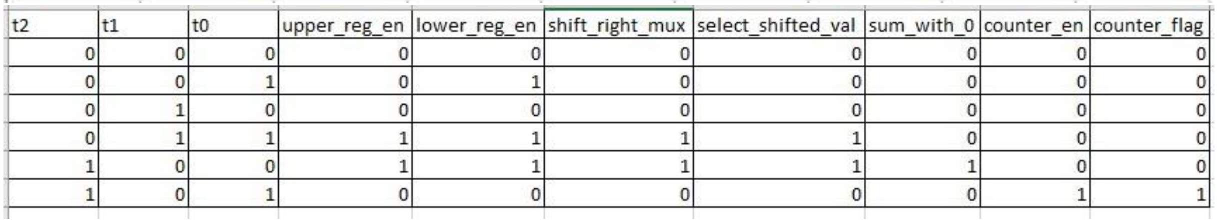
- CONTROL UNIT DESIGN OF MULTIPLIER (LOGISIM)



- TRUTH TABLES OF MULTIPLIER

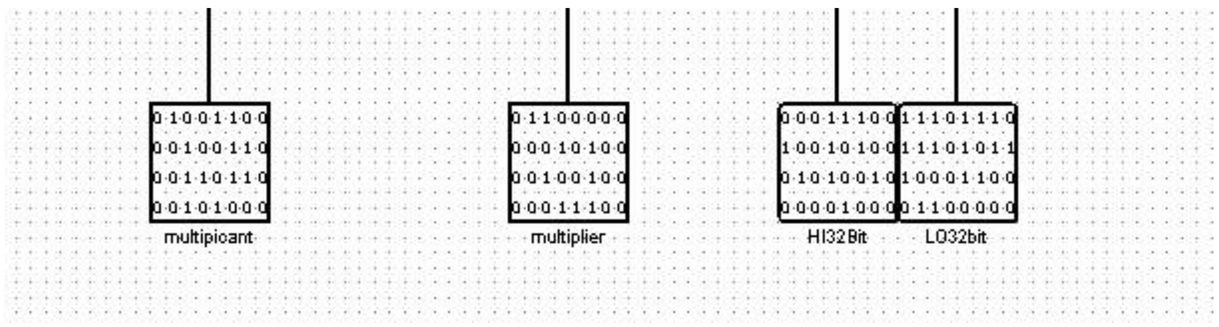


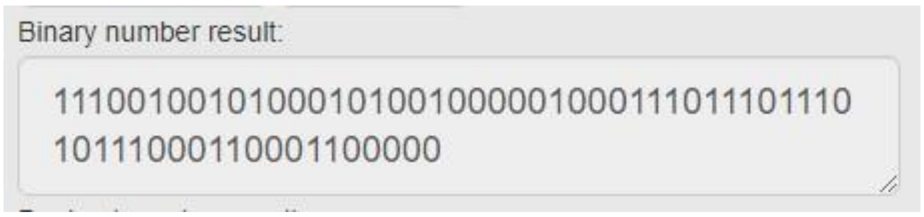
n2	n1	n0	start	is_counte	is_0bit_1	a2	a1	a0
0	0	0	0	-	-	0	0	0
0	0	0	1	-	-	0	0	1
0	0	1	-	-	-	0	1	0
0	1	0	-	0	0	1	0	0
0	1	0	-	0	1	0	1	1
0	1	0	-	1	-	0	0	0
0	1	1	-	-	-	1	0	1
1	0	0	-	-	-	1	0	1
1	0	1	-	-	-	0	1	0



t2	t1	t0	upper_reg_en	lower_reg_en	shift_right_mux	select_shifted_val	sum_with_0	counter_en	counter_flag
0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0	0	0
1	0	0	1	1	1	1	1	0	0
1	0	1	0	0	0	0	0	1	1

TEST1:

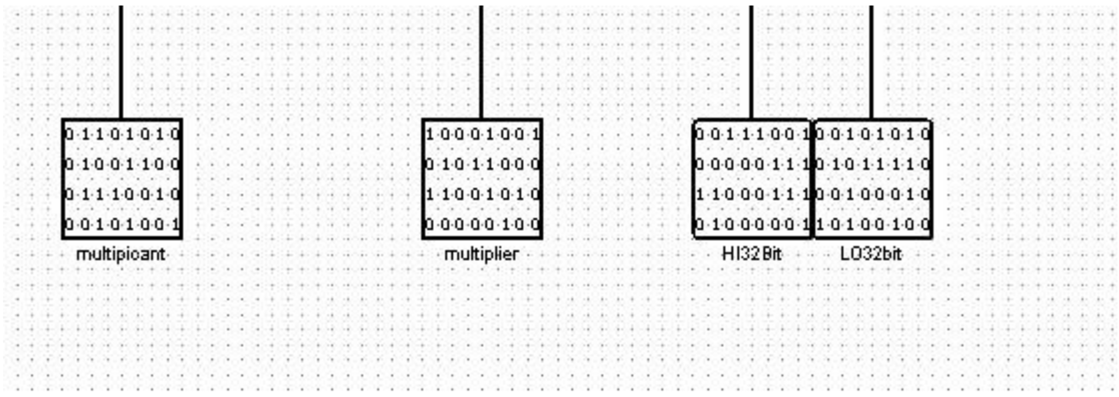


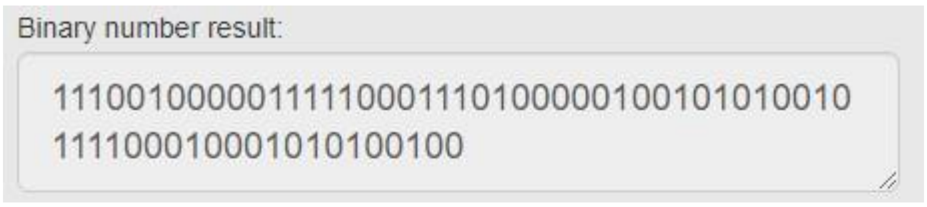


Binary number result:

11100100101000101001000001000111011101110
1011100011000110000

TEST2:





Binary number result:

```
11100100000111110001110100000100101010010
111100010001010100100
```

- module flop32bit (C, D, CE, Q);

32 bit flop for keeping data that is coming from D 32 bit input inside Q register.

- module flop (C, D, CE, Q);

1 bit flop for keeping data that is coming from D 1 bit input inside Q register.

- module mult_datapath(input [31:0] lower, input [31:0] mult,input upper_reg_en,input lower_reg_en,input sum_with_0, input counter_en,input clk,output is_counter_32,output is_0bit_1,output [31:0] hi,output [31:0] lo);

  Module for datapath of multiplier

  Lower and mult : inputs of multiplication operation

  Hi : high 32 bit of 64 bit conclusion

  Lo : low 32 bit of 64 bit conclusion

- module mult_control_unit(input start, input is_counter_32, input is_0bit_1, input clk, output upper_reg_en, output lower_reg_en, output sum_with_0, output counter_en);

  Module for datapath of multiplier

  Start : start flag for conclusion

NOTES FOR MULTIPLIER:

I did this project with logisim as well and I put all the logisim files with this project folder to show that my implementation is true about multipication. I hadn't got time to bring datapath and control unit but you will see that mult_control unit is working, you can test it with mult_cUnit_tb testbench, all steps shown at FSM above will be printed to Modelsim terminal by monitor function and you can see that all the nomenclature at mult_datapath is the same as the datapath photo above and works correctly. Some additional modules defined for datapath part which are

flop32bit : register definition for keep 32 bit values.

shift_right_take_msb : 64 bit shifting without using shift operator defined in Verilog

PART 2 : ALU MODULE

- module alu32(R,cin,cout,A,B,ALUOP);

  R : 32 bit result

  Cin : carry in for addition part

  Cout : carry out for addition part

  A : First 32bit input for ALU

  B : Second 32bit input for ALU

  ALUOP : 3bit ALU operation flag that chooses the operation will be done

- module _32bit_adder (S,C,A,B,C0);

  S : 32 bit result

C : carry out

C0 : carry in

A : First 32bit input for adder

B : Second 32bit input for adder

32bit adder uses full adder, full adder uses half adder and half adder implementation for 1 bit is in

- module half_adder(sum, carry_out, a, b);

- module mux8x1(input [31:0] i0, input [31:0] i1,input [31:0] i2,input [31:0] i3,input [31:0] i4,input [31:0] i5,input [31:0] i6, input [31:0] i7, input s0, input s1, input s2,output [31:0] out);

  8xmux selects one of the result according to selection inputs which are s0, s1 and s2.

- module mux2x1(output [31:0] out, input [31:0] i0, input [31:0] i1, input select);

_32bitadder_tb(); , alu32_testbench_1(); , mux2x1_tb(); , mux8x1_tb(); are testbench modules.