CSE 331
HW2

Burak Kocausta

1901042605

General

Part 1:

Firstly, I draw the finite state machine diagram for the controller. Controller gets 2 signals which are “write” and “less32”, generates 3 signals which are “add”, “shr”, “incr”. This controller uses less32 to reach final state. write signal means, least significant bit of product is 1, so it should generate add signal. shr signal is generated when product must be shifted. incr is for incrementing the counter. With these signals only 4 state is needed. 00, 01, 10, 11. I draw the state diagram, truth table, and equations in Designs part in this homework. With these schematic I implemented the control.v module. Then I decide to do datapath.v module, again I designed the schematic firstly, 2 register is needed for the datapath. One of them is for counter, other one is for product. Datapath must use controller’s signals and generate write, less32 signals. For the write signal, I only check the lsb of product, and send it as result. Wrote a simple addition for incrementing the counter, and send the 5^{th} bit of result if it is equal to 32 or not, whole comparator is not needed. For adding multiplicand to products first 32 bit, I used 32 bit multiplexer which gets add as select signal, then choses 0 or multiplicand. It adds this multiplexers result, and adds with the product’s first 32 bit. Then the result is made 64 bit with products other 32 bit. Then I don’t need to write shifter, because there is only 1 bit shifting is needed, therefore I decide to make this shifting with using buses. This shifted result goes to multiplexer which has select bit as shr. Multiplexer’s result can be either shifted or not. So the result is written to product register in each positive edge of clock. This result’s lsb is output as write signal. This datapath’s schematic is also shown in the Design part. For the mult32 module I only combine these modules. There is no additionally logic needed for mult32 module. I tested each of the module, and results are in the test part.

Part 2:

Initially I designed the part 2 for the homework. Every submodule is designed before ALU. Then I eliminate some of the submodules like slt, and sub. Because the adder in the ALU can handle these operations. I do this with using aluop signals, and multiplexers. To make addition to subtraction, second operand must be two's complemented. I make this with, getting not of second operand then putting it to a 2x1 multiplexer with its unchanged signal. Select bit decides it will be subtraction or not. This select with comes from aluop logic. I made an equation which is only true when subtraction is necessary. This logic with aluop signals generates 1 when subtraction is needed. After the multiplexer, result is an operand of adder. Also, adder's carry in signal is aluop logics select bit. For the set less than I used, the overflow bit of addition result, then anded it with most significant bit of the sum. Then, alu part needed a 32 bit 8x1 multiplexer. I designed 1 bit 2x1 multiplexer. Then 8 bit is designed using 1 bit, then 32 bit. After designing 32 bit 2x1 multiplexer, 4x1 is designed with 2x1 multiplexers, after that 8x1 is designed with 2x1 and 4x1 multiplexers. After doing part 1 I added the multiply operation to the ALU. Gates are shown in the ALU design part. Tests of the each submodule is also on the test part.

Modules

full_adder.v: 1 bit full adder

adder_8b.v: 8 bit adder. It is designed using 1 bit full adders.

adder_32.v: 32 bit adder. It is designed using 8 bit adders

or_32b.v: 32 bit or

and_32b.v: 32 bit and

not_32b.v: 32 bit not

mux_2x1.v: 1 bit mux with 1 select bit.

mux_2x1_8b.v: 8 bit mux with 1 select bit.

mux_2x1_32b.v: 32 bit mux with 1 select bit.

mux_4x1_32b.v: 32 bit mux with 2 select, It is designed using 2x1 32 bit multiplexers.

mux_8x1_32b.v: 32 bit mux with 3 select, designed using 2x1 and 4x1 32 bit multiplexers.

slt_32b.v:32 bit set less than. **This is not in the ALU**, it is only used for testing slt logic that I thought. **ALU operates slt without any additional adders.**

sub_32b.v: **This is not in the ALU**, it is only used for testing substractor logic. **ALU makes substraction without any additional adders.**

alu32.v: 32 bit alu which can operate add, sub, mult, xor, and, or, slt, nor operations.

control.v: controller module for finite state machine.

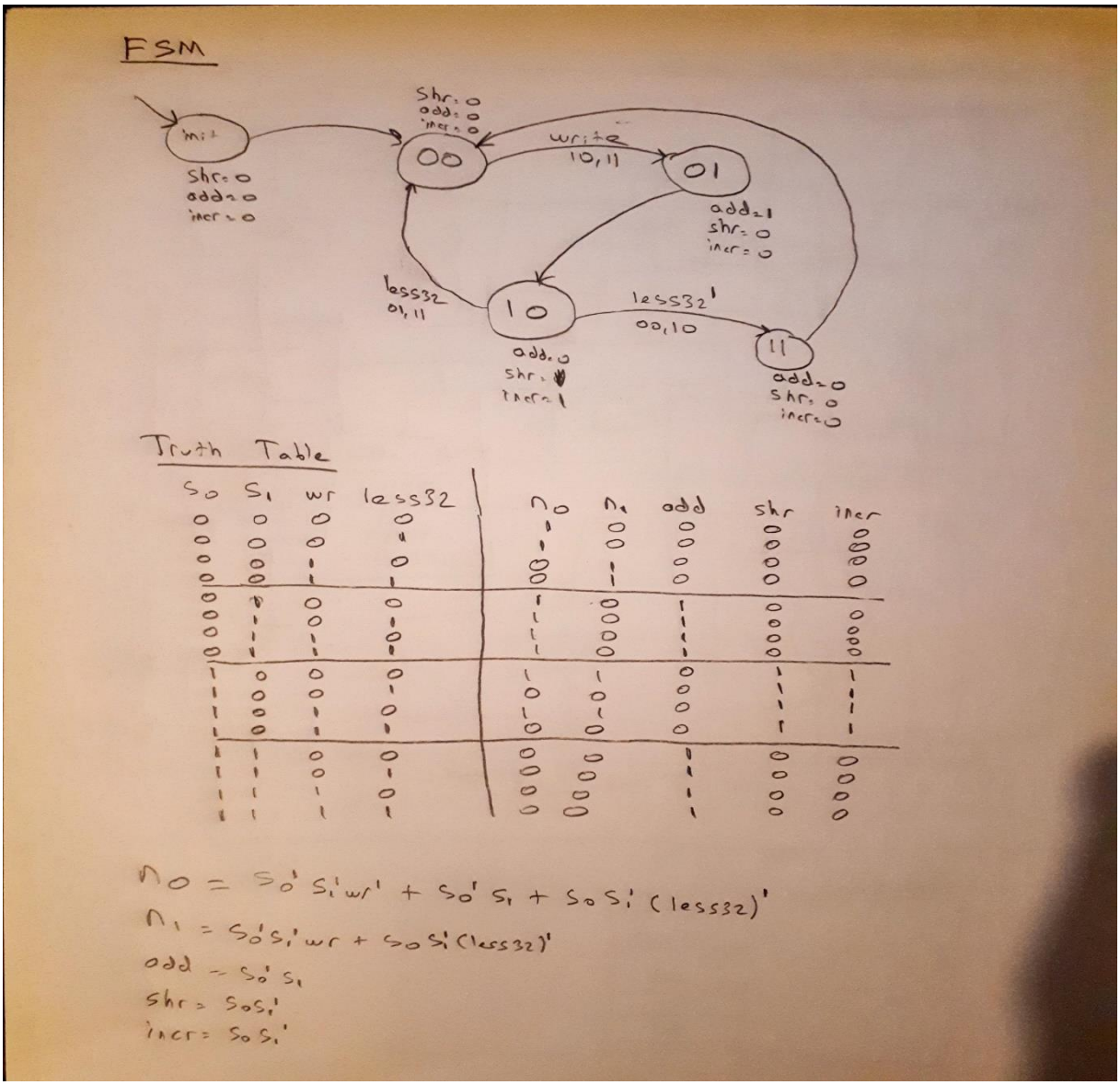
datapath.v: datapath of multiply operation.

mult32.v: 32 bit multiplication module, output is 64 bit product. It uses datapath, and control modules.

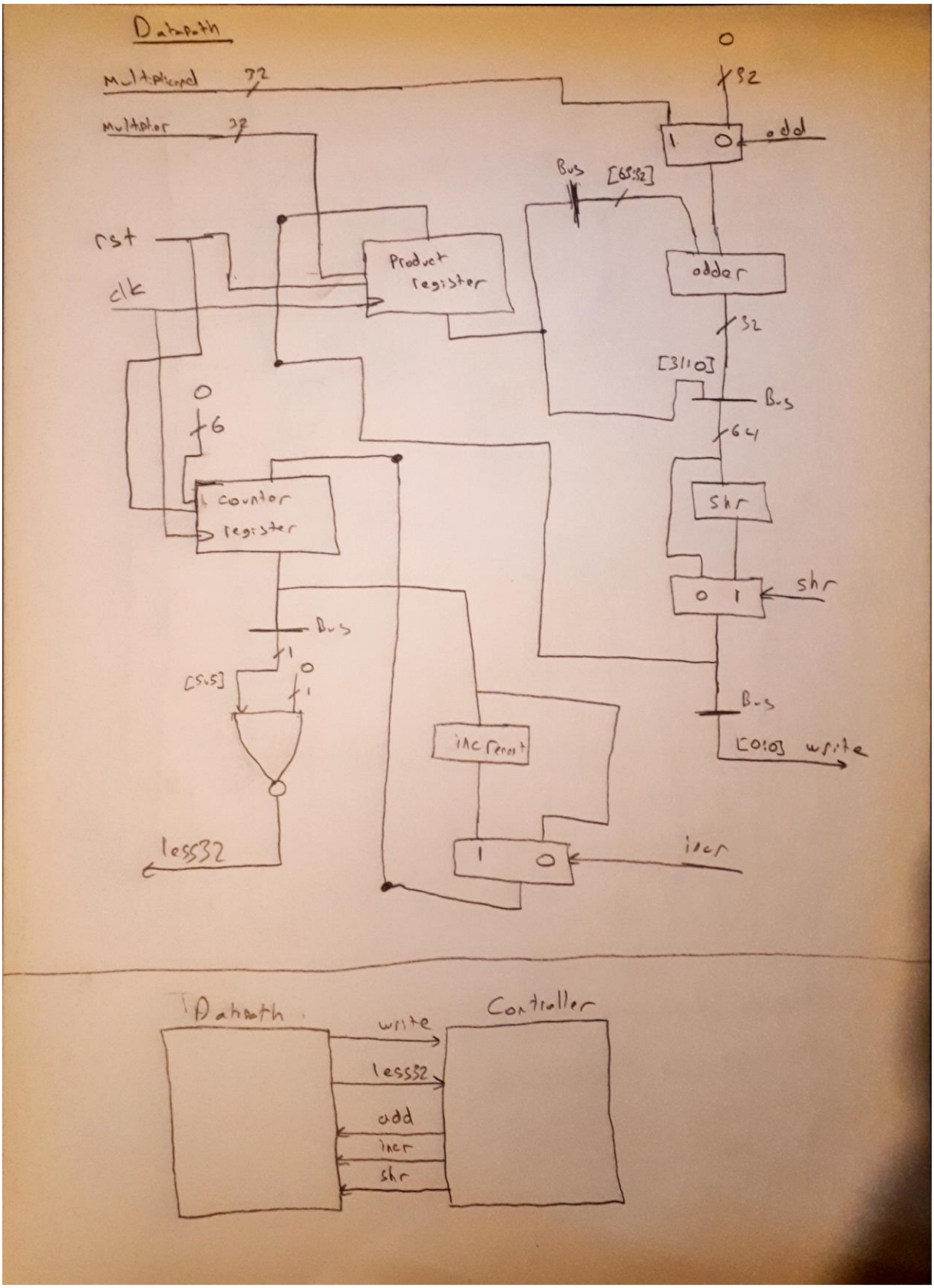
Designs

Part 1:

Finite State Machine & Truth Table for Control Unit:

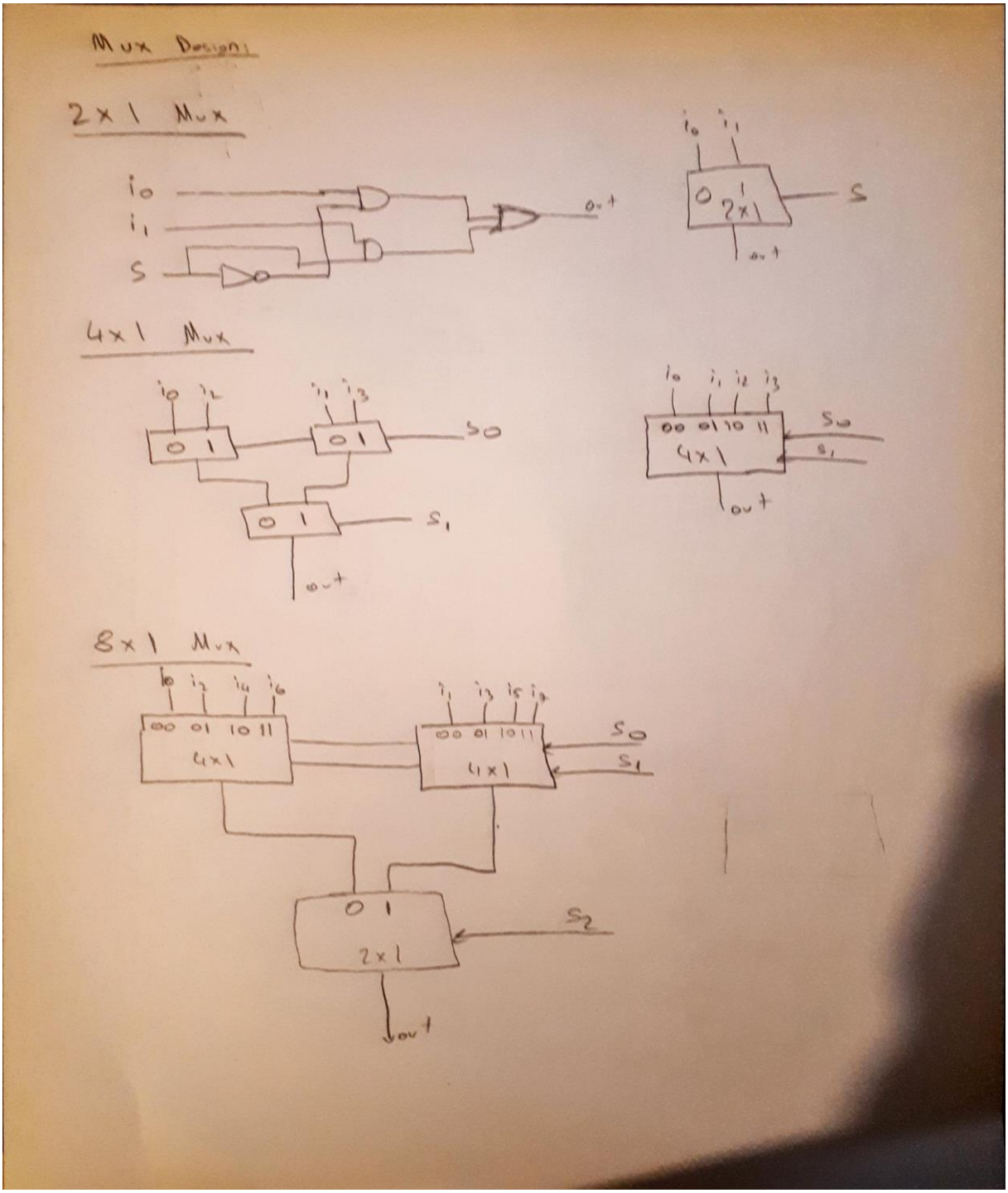


Datapath:

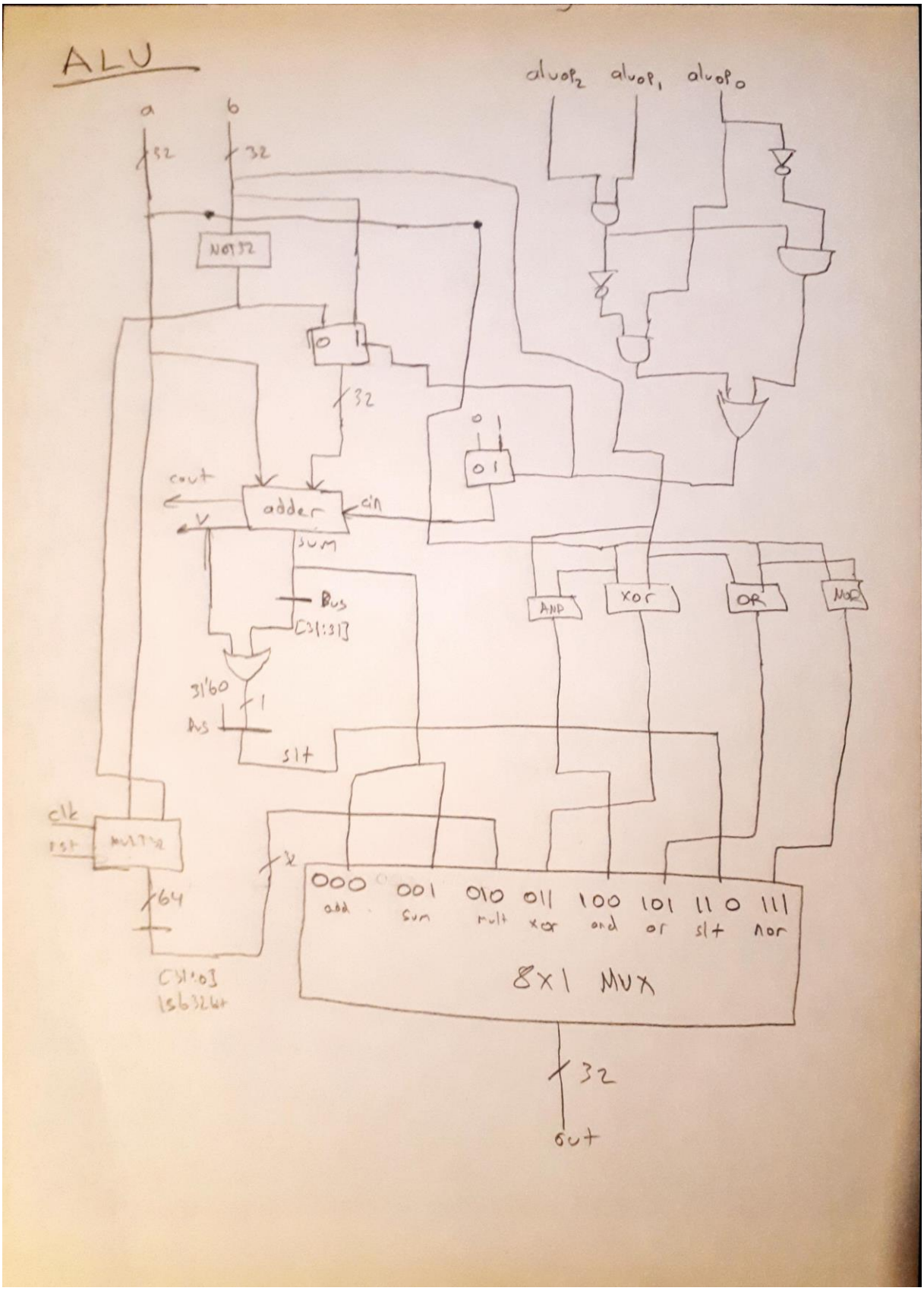

Datapath
Multiplicand
Multiplier
32
32
0
32
odd
1
0
Bus
[63:32]
rst
clk
Product register
adder
32
[31:0]
Bus
64
shr
shr
1
0
Bus
[5:5]
1
0
1
counter register
6
0
increment
Bus
[0:0]
write
incr
1
0
less32
Datapath
Controller
write
less32
odd
incr
shr

Part 2:

Multiplexer Designs:

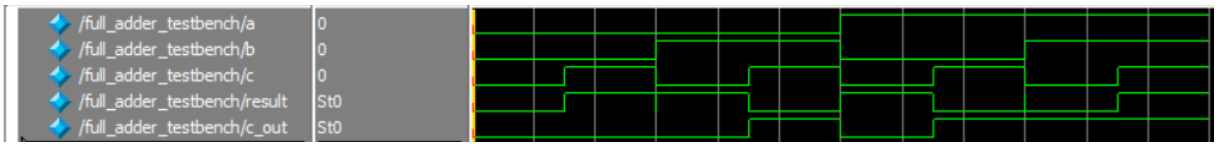


ALU Design:


ALU
aluop2 aluop1 aluop0
a
b
32
32
NOT32
0 1
32
0 1
cout
adder
cin
V
sum
Bus
[31:31]
31'b0
1
Ass
slt
AND
xor
OR
NOR
clk
rst
MULT32
64
[31:0]
lsb32bit
32
000 add
001 sum
010 mult
011 xor
100 and
101 or
110 slt
111 nor
8x1 MUX
32
out

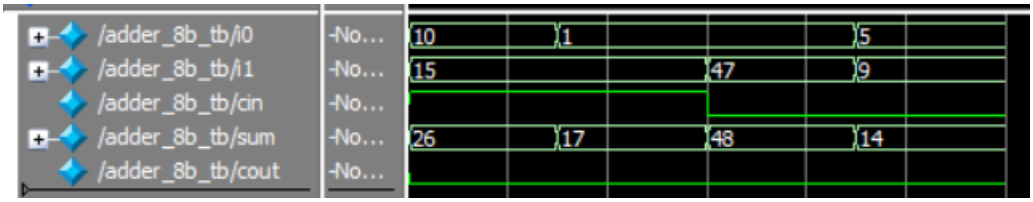
Test Benches

full_adder:




```
VSIM(paused)> step -current
# time =  0, i0 = 0, i1 = 0, cin = 0, sum = 0, carry_out = 0
# time = 30, i0 = 0, i1 = 0, cin = 1, sum = 1, carry_out = 0
# time = 60, i0 = 0, i1 = 1, cin = 0, sum = 1, carry_out = 0
# time = 90, i0 = 0, i1 = 1, cin = 1, sum = 0, carry_out = 1
# time = 120, i0 = 1, i1 = 0, cin = 0, sum = 1, carry_out = 0
# time = 150, i0 = 1, i1 = 0, cin = 1, sum = 0, carry_out = 1
# time = 180, i0 = 1, i1 = 1, cin = 0, sum = 0, carry_out = 1
# time = 210, i0 = 1, i1 = 1, cin = 1, sum = 1, carry_out = 1
```

adder_8b:

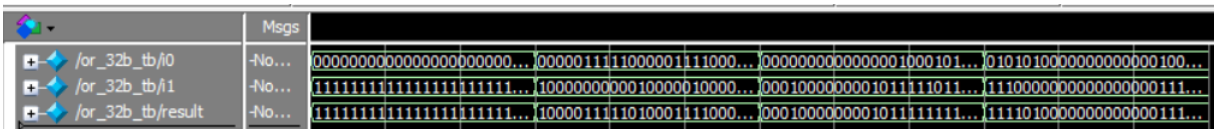



```
VSIM(paused)> step -current
#
# time = 0
# i0 = 00001010 (decimal= 10)
# i1 = 00001111 (decimal= 15)
# cin = 1
# sum = 00011010 (decimal= 26)
# cout = 0
#
# time = 30
# i0 = 00000001 (decimal= 1)
# i1 = 00001111 (decimal= 15)
# cin = 1
# sum = 00010001 (decimal= 17)
# cout = 0
#
# time = 60
# i0 = 00000001 (decimal= 1)
# i1 = 00101111 (decimal= 47)
# cin = 0
# sum = 00110000 (decimal= 48)
# cout = 0
#
# time = 90
# i0 = 00000101 (decimal= 5)
# i1 = 00001001 (decimal= 9)
# cin = 0
# sum = 00001110 (decimal= 14)
# cout = 0
```

adder_32b:

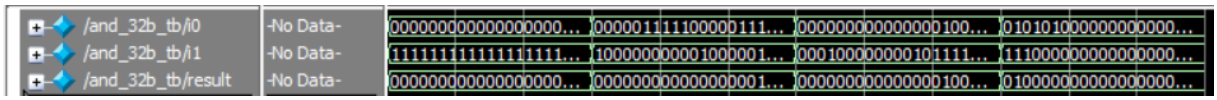

```
# time = 0
# i0 = 00000000000000000000000000001010 (decimal= 10)
# i1 = 00000000000000000000000000001111 (decimal= 15)
# cin = 1
# sum = 00000000000000000000000000011010 (decimal= 26)
# cout = 0 overflow = 0
# time = 30
# i0 = 00000000000000000000000000000001 (decimal= 1)
# i1 = 11111111111111111111111111111111 (decimal= -1)
# cin = 1
# sum = 00000000000000000000000000000001 (decimal= 1)
# cout = 1 overflow = 0
# time = 60
# i0 = 00000000000000010001010001000001 (decimal= 70721)
# i1 = 00010000000010111110111111101111 (decimal= 269217775)
# cin = 0
# sum = 00010000000011010000010000110000 (decimal= 269288496)
# cout = 0 overflow = 0
# time = 90
# i0 = 01111111111111111111111111111111 (decimal= 2147483647)
# i1 = 01111111111111111111111111111111 (decimal= 2147483647)
# cin = 1
# sum = 11111111111111111111111111111111 (decimal= -1)
# cout = 0 overflow = 1
# time = 120
# i0 = 00000000000000000000000001110010 (decimal= 114)
# i1 = 00000000000000000000000011100101 (decimal= 229)
# cin = 0
# sum = 00000000000000000000000101010111 (decimal= 343)
# cout = 0 overflow = 0
```

or_32b:



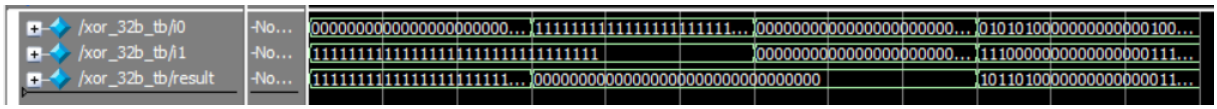

```
VSIM(paused)> step -current
# time = 0
# i0 = 00000000000000000000000000000000
# i1 = 11111111111111111111111111111111
# result = 11111111111111111111111111111111
#
# time = 30
# i0 = 00000111110000011110000011100111
# i1 = 10000000000100000100000100001000
# result = 10000111111010001111000111101111
#
# time = 60
# i0 = 00000000000000010001010001000001
# i1 = 00010000000010111110111111101111
# result = 00010000000010111111111111101111
#
# time = 90
# i0 = 01010100000000000001000000000000
# i1 = 11100000000000000001111111111111
# result = 11110100000000000001111111111111
#
```

and_32b_tb:



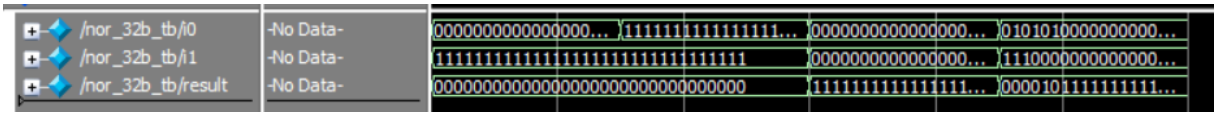

```
VSIM(paused)> step -current
# time = 0
# i0 = 00000000000000000000000000000000
# i1 = 11111111111111111111111111111111
# result = 00000000000000000000000000000000
#
# time = 30
# i0 = 00000111110000011110000011100111
# i1 = 10000000000100000100001100001000
# result = 00000000000000000100000000000000
#
# time = 60
# i0 = 00000000000000001000101000100001
# i1 = 00010000000010111111011111101111
# result = 00000000000000001000001000100001
#
# time = 90
# i0 = 01010100000000000001000000000000
# i1 = 11100000000000000001111111111111
# result = 01000000000000000001000000000000
#
```

xor_32b_tb:



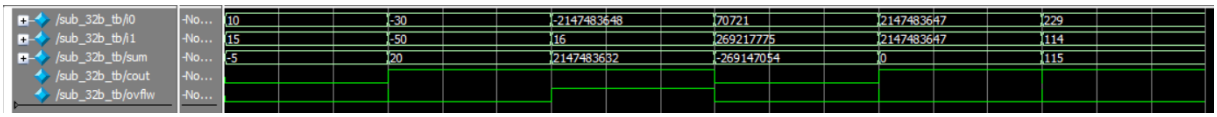

```
# time = 0
# i0 = 00000000000000000000000000000000
# i1 = 11111111111111111111111111111111
# result = 11111111111111111111111111111111
#
# time = 30
# i0 = 11111111111111111111111111111111
# i1 = 11111111111111111111111111111111
# result = 00000000000000000000000000000000
#
# time = 60
# i0 = 00000000000000000000000000000000
# i1 = 00000000000000000000000000000000
# result = 00000000000000000000000000000000
#
# time = 90
# i0 = 01010100000000000001000000000000
# i1 = 11100000000000000001111111111111
# result = 10110100000000000001111111111111
#
```

nor_32b_tb:



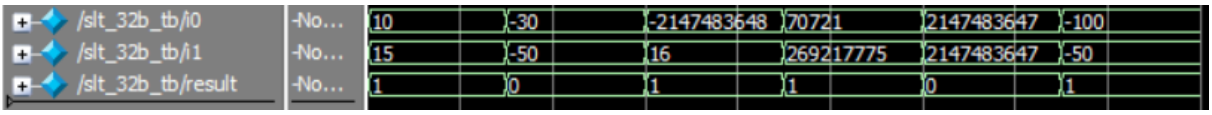
```
# time = 0
# i0 = 00000000000000000000000000000000
# i1 = 11111111111111111111111111111111
# result = 00000000000000000000000000000000
#
# time = 30
# i0 = 11111111111111111111111111111111
# i1 = 11111111111111111111111111111111
# result = 00000000000000000000000000000000
#
# time = 60
# i0 = 00000000000000000000000000000000
# i1 = 00000000000000000000000000000000
# result = 11111111111111111111111111111111
#
# time = 90
# i0 = 01010100000000000001000000000000
# i1 = 11100000000000000001111111111111
# result = 00001011111111111110000000000000
#
```

sub_32b_tb:



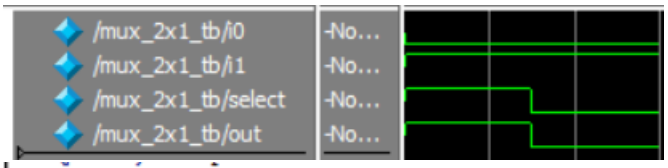

```
# time = 0
# i0 = 00000000000000000000000000001010 (decimal= 10)
# i1 = 00000000000000000000000000001111 (decimal= 15)
# sum = 11111111111111111111111111111011 (decimal= -5)
# cout = 0 overflow = 0
# time = 30
# i0 = 11111111111111111111111111100010 (decimal= -30)
# i1 = 11111111111111111111111111001110 (decimal= -50)
# sum = 00000000000000000000000000010100 (decimal= 20)
# cout = 1 overflow = 0
# time = 60
# i0 = 10000000000000000000000000000000 (decimal= -2147483648)
# i1 = 00000000000000000000000000010000 (decimal= 16)
# sum = 01111111111111111111111111110000 (decimal= 2147483632)
# cout = 1 overflow = 1
# time = 90
# i0 = 00000000000000010001010001000001 (decimal= 70721)
# i1 = 00010000000010111111011111101111 (decimal= 269217775)
# sum = 11101111111101010010010001010010 (decimal= -269147054)
# cout = 0 overflow = 0
# time = 120
# i0 = 01111111111111111111111111111111 (decimal= 2147483647)
# i1 = 01111111111111111111111111111111 (decimal= 2147483647)
# sum = 00000000000000000000000000000000 (decimal= 0)
# cout = 1 overflow = 0
# time = 150
# i0 = 00000000000000000000000011100101 (decimal= 229)
# i1 = 00000000000000000000000001110010 (decimal= 114)
# sum = 00000000000000000000000001110011 (decimal= 115)
# cout = 1 overflow = 0
```

slt_32b_tb:



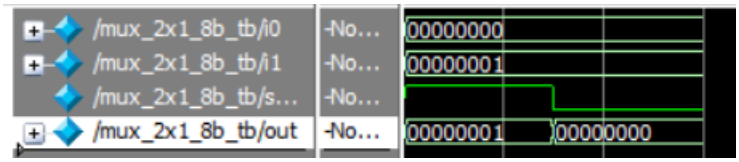

```
# time = 0
# i0 = 00000000000000000000000000001010 (decimal= 10)
# i1 = 00000000000000000000000000001111 (decimal= 15)
# result = 00000000000000000000000000000001 (decimal= 1)
# time = 30
# i0 = 11111111111111111111111111100010 (decimal= -30)
# i1 = 11111111111111111111111111001110 (decimal= -50)
# result = 00000000000000000000000000000000 (decimal= 0)
# time = 60
# i0 = 10000000000000000000000000000000 (decimal= -2147483648)
# i1 = 00000000000000000000000000010000 (decimal= 16)
# result = 00000000000000000000000000000001 (decimal= 1)
# time = 90
# i0 = 00000000000000010001010001000001 (decimal= 70721)
# i1 = 00010000000101111101111111101111 (decimal= 269217775)
# result = 00000000000000000000000000000001 (decimal= 1)
# time = 120
# i0 = 01111111111111111111111111111111 (decimal= 2147483647)
# i1 = 01111111111111111111111111111111 (decimal= 2147483647)
# result = 00000000000000000000000000000000 (decimal= 0)
# time = 150
# i0 = 11111111111111111111111110011100 (decimal= -100)
# i1 = 11111111111111111111111111001110 (decimal= -50)
# result = 00000000000000000000000000000001 (decimal= 1)
```

mux_2x1_tb:



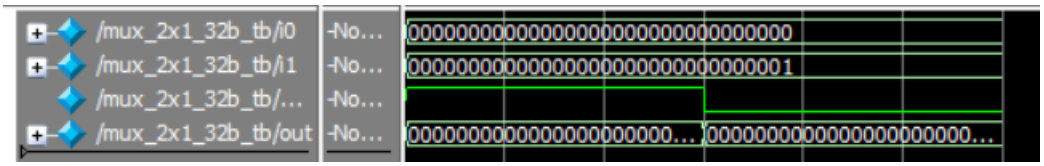

```
#
# time = 0, i0 = 0, i1 = 1, select = 1 --> out = 1
#
# time = 30, i0 = 0, i1 = 1, select = 0 --> out = 0
```

mux_2x1_8b_tb:



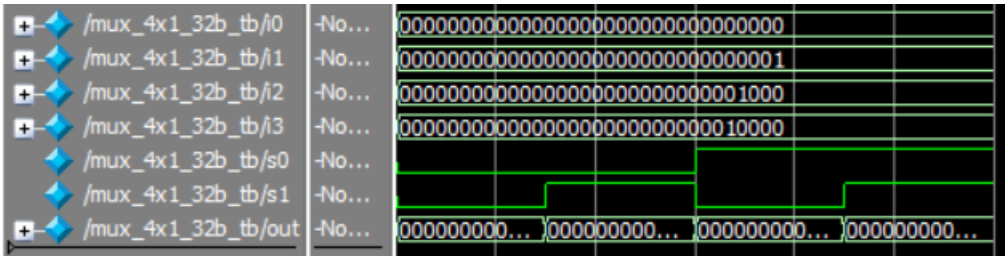

```
#
# time = 0, i0 = 00000000, i1 = 00000001, select = 1 --> out = 00000001
#
# time = 30, i0 = 00000000, i1 = 00000001, select = 0 --> out = 00000000
```

mux_2x1_32b_tb:



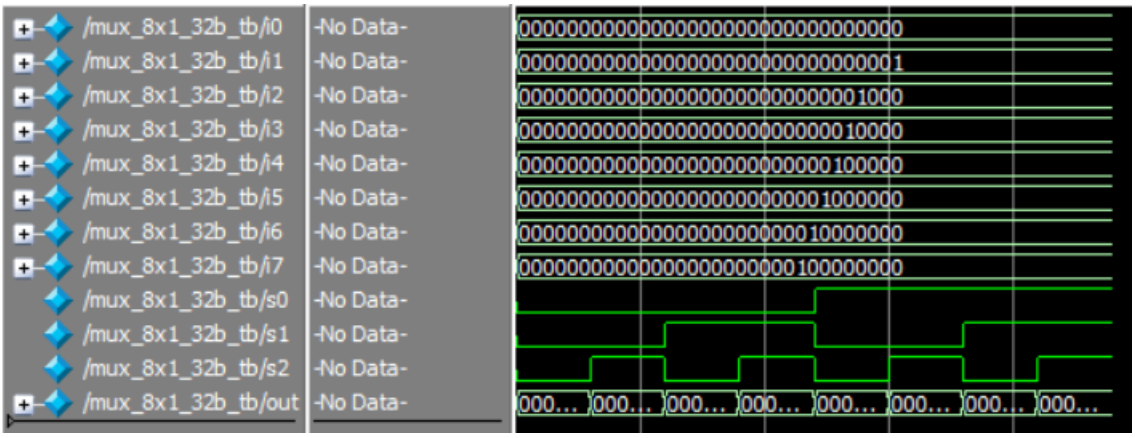

```
#
# time = 0, i0 = 00000000000000000000000000000000, i1 = 00000000000000000000000000000001, select = 1 --> out = 00000000000000000000000000000001
#
# time = 30, i0 = 00000000000000000000000000000000, i1 = 00000000000000000000000000000001, select = 0 --> out = 00000000000000000000000000000000
```

mux_4x1_32b_tb:



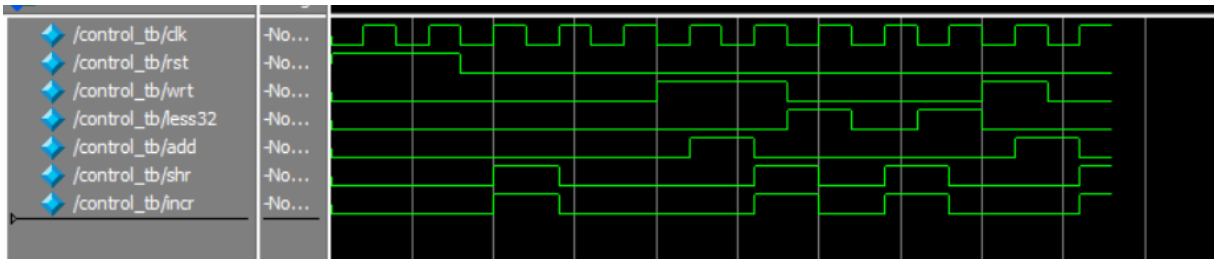

```
#
# time = 0, i0 = 0, i1 = 1, i2 = 8, i3 = 16
# s0 = 0, s1 = 0 --> out = 0
#
# time = 30, i0 = 0, i1 = 1, i2 = 8, i3 = 16
# s0 = 0, s1 = 1 --> out = 1
#
# time = 60, i0 = 0, i1 = 1, i2 = 8, i3 = 16
# s0 = 1, s1 = 0 --> out = 8
#
# time = 90, i0 = 0, i1 = 1, i2 = 8, i3 = 16
# s0 = 1, s1 = 1 --> out = 16
```

mux_8x1_32b_tb:



```
#
# time = 0, i0 = 0, i1 = 1, i2 = 8, i3 = 16, i4 = 32, i5 = 64, i6 = 128, i7 = 256
# s0 = 0, s1 = 0, s2 = 0 --> out = 0
#
# time = 30, i0 = 0, i1 = 1, i2 = 8, i3 = 16, i4 = 32, i5 = 64, i6 = 128, i7 = 256
# s0 = 0, s1 = 0, s2 = 1 --> out = 1
#
# time = 60, i0 = 0, i1 = 1, i2 = 8, i3 = 16, i4 = 32, i5 = 64, i6 = 128, i7 = 256
# s0 = 0, s1 = 1, s2 = 0 --> out = 8
#
# time = 90, i0 = 0, i1 = 1, i2 = 8, i3 = 16, i4 = 32, i5 = 64, i6 = 128, i7 = 256
# s0 = 0, s1 = 1, s2 = 1 --> out = 16
#
# time = 120, i0 = 0, i1 = 1, i2 = 8, i3 = 16, i4 = 32, i5 = 64, i6 = 128, i7 = 256
# s0 = 1, s1 = 0, s2 = 0 --> out = 32
#
# time = 150, i0 = 0, i1 = 1, i2 = 8, i3 = 16, i4 = 32, i5 = 64, i6 = 128, i7 = 256
# s0 = 1, s1 = 0, s2 = 1 --> out = 64
#
# time = 180, i0 = 0, i1 = 1, i2 = 8, i3 = 16, i4 = 32, i5 = 64, i6 = 128, i7 = 256
# s0 = 1, s1 = 1, s2 = 0 --> out = 128
#
# time = 210, i0 = 0, i1 = 1, i2 = 8, i3 = 16, i4 = 32, i5 = 64, i6 = 128, i7 = 256
# s0 = 1, s1 = 1, s2 = 1 --> out = 256
```

control_tb: (states are changing when rst is 0, and clk is 1)



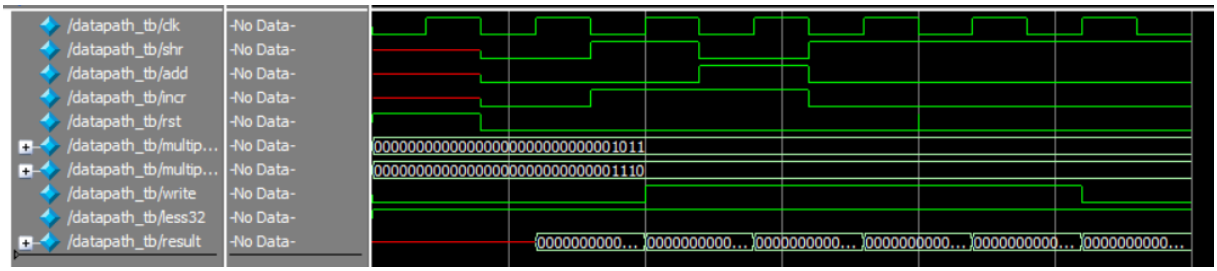

```
#
# time =  0, clk = 0 p_state = 00, n_state = 10
# rst = 1, wrt = 0, less32 = 0 --> add = 0, shr = 0, incr = 0
#
# time =  2, clk = 1 p_state = 00, n_state = 10
# rst = 1, wrt = 0, less32 = 0 --> add = 0, shr = 0, incr = 0
#
# time =  4, clk = 0 p_state = 00, n_state = 10
# rst = 1, wrt = 0, less32 = 0 --> add = 0, shr = 0, incr = 0
#
# time =  6, clk = 1 p_state = 00, n_state = 10
# rst = 1, wrt = 0, less32 = 0 --> add = 0, shr = 0, incr = 0
#
# time =  8, clk = 0 p_state = 00, n_state = 10
# rst = 0, wrt = 0, less32 = 0 --> add = 0, shr = 0, incr = 0
#
# time = 10, clk = 1 p_state = 10, n_state = 11
# rst = 0, wrt = 0, less32 = 0 --> add = 0, shr = 1, incr = 1
#
# time = 12, clk = 0 p_state = 10, n_state = 11
# rst = 0, wrt = 0, less32 = 0 --> add = 0, shr = 1, incr = 1
#
# time = 14, clk = 1 p_state = 11, n_state = 00
# rst = 0, wrt = 0, less32 = 0 --> add = 0, shr = 0, incr = 0
#
# time = 16, clk = 0 p_state = 11, n_state = 00
# rst = 0, wrt = 0, less32 = 0 --> add = 0, shr = 0, incr = 0
```

```
#
# time = 18, clk = 1 p_state = 00, n_state = 10
# rst = 0, wrt = 0, less32 = 0 --> add = 0, shr = 0, incr = 0
#
# time = 20, clk = 0 p_state = 00, n_state = 01
# rst = 0, wrt = 1, less32 = 0 --> add = 0, shr = 0, incr = 0
#
# time = 22, clk = 1 p_state = 01, n_state = 10
# rst = 0, wrt = 1, less32 = 0 --> add = 1, shr = 0, incr = 0
#
# time = 24, clk = 0 p_state = 01, n_state = 10
# rst = 0, wrt = 1, less32 = 0 --> add = 1, shr = 0, incr = 0
#
# time = 26, clk = 1 p_state = 10, n_state = 11
# rst = 0, wrt = 1, less32 = 0 --> add = 0, shr = 1, incr = 1
#
# time = 28, clk = 0 p_state = 10, n_state = 00
# rst = 0, wrt = 0, less32 = 1 --> add = 0, shr = 1, incr = 1
#
# time = 30, clk = 1 p_state = 00, n_state = 10
# rst = 0, wrt = 0, less32 = 1 --> add = 0, shr = 0, incr = 0
#
# time = 32, clk = 0 p_state = 00, n_state = 10
# rst = 0, wrt = 0, less32 = 0 --> add = 0, shr = 0, incr = 0
#
# time = 34, clk = 1 p_state = 10, n_state = 11
# rst = 0, wrt = 0, less32 = 0 --> add = 0, shr = 1, incr = 1
```

```
#
# time = 36, clk = 0 p_state = 10, n_state = 00
# rst = 0, wrt = 0, less32 = 1 --> add = 0, shr = 1, incr = 1
#
# time = 38, clk = 1 p_state = 00, n_state = 10
# rst = 0, wrt = 0, less32 = 1 --> add = 0, shr = 0, incr = 0
#
# time = 40, clk = 0 p_state = 00, n_state = 01
# rst = 0, wrt = 1, less32 = 0 --> add = 0, shr = 0, incr = 0
#
# time = 42, clk = 1 p_state = 01, n_state = 10
# rst = 0, wrt = 1, less32 = 0 --> add = 1, shr = 0, incr = 0
#
# time = 44, clk = 0 p_state = 01, n_state = 10
# rst = 0, wrt = 0, less32 = 0 --> add = 1, shr = 0, incr = 0
#
# time = 46, clk = 1 p_state = 10, n_state = 11
# rst = 0, wrt = 0, less32 = 0 --> add = 0, shr = 1, incr = 1
```

datapath_tb: (signals, product, and result changing when rst is 0, clk is 1)

it is red at first because rst is 1.



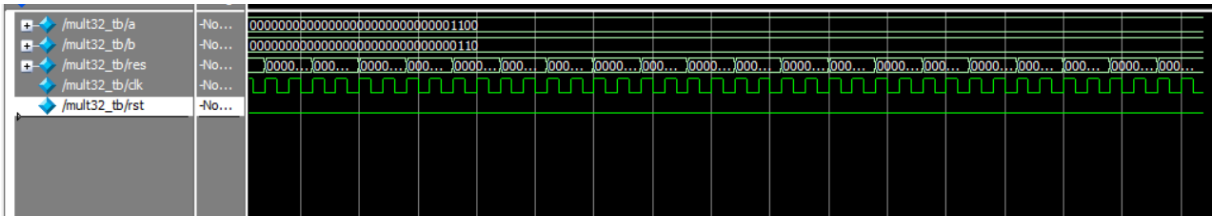
```
# time = 4, clk = 0, rst = 0
# multiplicand = 00000000000000000000000000001011
# multiplier = 00000000000000000000000000001110
# count = 000000
# shr = 0, add = 0, incr = 0
# results:
# product = 00000000000000000000000000000000000000000000000000000000001110
# result = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
# write = 0, less32 = 1
#
# time = 6, clk = 1, rst = 0
# multiplicand = 00000000000000000000000000001011
# multiplier = 00000000000000000000000000001110
# count = 000000
# shr = 0, add = 0, incr = 0
# results:
# product = 00000000000000000000000000000000000000000000000000000000001110
# result = 00000000000000000000000000000000000000000000000000000000001110
# write = 0, less32 = 1
#

# time = 8, clk = 0, rst = 0
# multiplicand = 00000000000000000000000000001011
# multiplier = 00000000000000000000000000001110
# count = 000000
# shr = 1, add = 0, incr = 1
# results:
# product = 00000000000000000000000000000000000000000000000000000000001110
# result = 00000000000000000000000000000000000000000000000000000000001110
# write = 0, less32 = 1
#
# time = 10, clk = 1, rst = 0
# multiplicand = 00000000000000000000000000001011
# multiplier = 00000000000000000000000000001110
# count = 000001
# shr = 1, add = 0, incr = 1
# results:
# product = 00000000000000000000000000000000000000000000000000000000000111
# result = 00000000000000000000000000000000000000000000000000000000000111
# write = 1, less32 = 1

# time = 12, clk = 0, rst = 0
# multiplicand = 00000000000000000000000000001011
# multiplier = 00000000000000000000000000001110
# count = 000001
# shr = 0, add = 1, incr = 1
# results:
# product = 00000000000000000000000000000000000000000000000000000000000111
# result = 00000000000000000000000000000000000000000000000000000000000111
# write = 1, less32 = 1
#
# time = 14, clk = 1, rst = 0
# multiplicand = 00000000000000000000000000001011
# multiplier = 00000000000000000000000000001110
# count = 000010
# shr = 0, add = 1, incr = 1
# results:
# product = 00000000000000000000000000001011000000000000000000000000000000111
# result = 00000000000000000000000000001011000000000000000000000000000000111
# write = 1, less32 = 1
```

```
# time = 16, clk = 0, rst = 0
# multiplicand = 00000000000000000000000000001011
# multiplier = 00000000000000000000000000001110
# count = 000010
# shr = 1, add = 0, incr = 0
# results:
# product = 00000000000000000000000000010110000000000000000000000000000111
# result = 00000000000000000000000000001011000000000000000000000000000111
# write = 1, less32 = 1
#
# time = 18, clk = 1, rst = 0
# multiplicand = 00000000000000000000000000001011
# multiplier = 00000000000000000000000000001110
# count = 000010
# shr = 1, add = 0, incr = 0
# results:
# product = 00000000000000000000000000010110000000000000000000000000000011
# result = 00000000000000000000000000001011000000000000000000000000000011
# write = 1, less32 = 1
#
```

```
# time = 20, clk = 0, rst = 0
# multiplicand = 00000000000000000000000000001011
# multiplier = 00000000000000000000000000001110
# count = 000010
# shr = 1, add = 0, incr = 0
# results:
# product = 00000000000000000000000000010110000000000000000000000000000011
# result = 00000000000000000000000000001011000000000000000000000000000011
# write = 1, less32 = 1
#
# time = 22, clk = 1, rst = 0
# multiplicand = 00000000000000000000000000001011
# multiplier = 00000000000000000000000000001110
# count = 000010
# shr = 1, add = 0, incr = 0
# results:
# product = 00000000000000000000000000010110000000000000000000000000000001
# result = 00000000000000000000000000001011000000000000000000000000000001
# write = 1, less32 = 1
#
```

```
# time = 24, clk = 0, rst = 0
# multiplicand = 00000000000000000000000000001011
# multiplier = 00000000000000000000000000001110
# count = 000010
# shr = 1, add = 0, incr = 0
# results:
# product = 00000000000000000000000000001011000000000000000000000000000000001
# result = 00000000000000000000000000001011000000000000000000000000000000001
# write = 1, less32 = 1
#
# time = 26, clk = 1, rst = 0
# multiplicand = 00000000000000000000000000001011
# multiplier = 00000000000000000000000000001110
# count = 000010
# shr = 1, add = 0, incr = 0
# results:
# product = 00000000000000000000000000001011000000000000000000000000000000000
# result = 00000000000000000000000000001011000000000000000000000000000000000
# write = 0, less32 = 1
#
# time = 28, clk = 0, rst = 0
# multiplicand = 00000000000000000000000000001011
# multiplier = 00000000000000000000000000001110
# count = 000010
# shr = 1, add = 0, incr = 0
# results:
# product = 00000000000000000000000000001011000000000000000000000000000000000
# result = 00000000000000000000000000001011000000000000000000000000000000000
# write = 0, less32 = 1
```

mult32_tb: Every step, and signals are shown.



```
#
# time =  0, a =12, b =6, product = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
# result =  x
# less32 = 1, shr = 0, add = 0, write = 0
#
# time =  6, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000000000000110
# result =  6
# less32 = 1, shr = 1, add = 0, write = 0
#
# time = 10, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000000000000011
# result =  3
# less32 = 1, shr = 0, add = 0, write = 1
#
# time = 14, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000000000000011
# result =  3
# less32 = 1, shr = 0, add = 1, write = 1
#
# time = 18, a =12, b =6, product = 00000000000000000000000000011000000000000000000000000000000000011
# result =  3
# less32 = 1, shr = 1, add = 0, write = 1
#
# time = 22, a =12, b =6, product = 00000000000000000000000000011000000000000000000000000000000000001
# result =  1
# less32 = 1, shr = 0, add = 0, write = 1
#
# time = 26, a =12, b =6, product = 00000000000000000000000000011000000000000000000000000000000000001
# result =  1
# less32 = 1, shr = 0, add = 1, write = 1
#
# time = 30, a =12, b =6, product = 00000000000000000000000001001000000000000000000000000000000000001
# result =  1
# less32 = 1, shr = 1, add = 0, write = 1
#

time = 34, a =12, b =6, product = 00000000000000000000000001001000000000000000000000000000000000000
result =  0
less32 = 1, shr = 0, add = 0, write = 0

time = 38, a =12, b =6, product = 00000000000000000000000001001000000000000000000000000000000000000
result =  0
less32 = 1, shr = 1, add = 0, write = 0

time = 42, a =12, b =6, product = 00000000000000000000000001001000000000000000000000000000000000000
result = 2147483648
less32 = 1, shr = 0, add = 0, write = 0

time = 46, a =12, b =6, product = 00000000000000000000000001001000000000000000000000000000000000000
result = 2147483648
less32 = 1, shr = 1, add = 0, write = 0

time = 50, a =12, b =6, product = 00000000000000000000000001001000000000000000000000000000000000000
result = 1073741824
less32 = 1, shr = 0, add = 0, write = 0

time = 54, a =12, b =6, product = 00000000000000000000000001001000000000000000000000000000000000000
result = 1073741824
less32 = 1, shr = 1, add = 0, write = 0

time = 58, a =12, b =6, product = 00000000000000000000000001001000000000000000000000000000000000000
result = 536870912
less32 = 1, shr = 0, add = 0, write = 0
```

```
time = 58, a =12, b =6, product = 0000000000000000000000000000000100100000000000000000000000000000
result = 536870912
less32 = 1, shr = 0, add = 0, write = 0

time = 62, a =12, b =6, product = 0000000000000000000000000000000100100000000000000000000000000000
result = 536870912
less32 = 1, shr = 1, add = 0, write = 0

time = 66, a =12, b =6, product = 0000000000000000000000000000000100100000000000000000000000000000
result = 2415919104
less32 = 1, shr = 0, add = 0, write = 0

time = 70, a =12, b =6, product = 0000000000000000000000000000000100100000000000000000000000000000
result = 2415919104
less32 = 1, shr = 1, add = 0, write = 0

time = 74, a =12, b =6, product = 0000000000000000000000000000000100100000000000000000000000000000
result = 1207959552
less32 = 1, shr = 0, add = 0, write = 0

time = 78, a =12, b =6, product = 0000000000000000000000000000000100100000000000000000000000000000
result = 1207959552
less32 = 1, shr = 1, add = 0, write = 0

time = 82, a =12, b =6, product = 0000000000000000000000000000000100100000000000000000000000000000
result = 603979776
less32 = 1, shr = 0, add = 0, write = 0

time = 86, a =12, b =6, product = 0000000000000000000000000000000100100000000000000000000000000000
result = 603979776
less32 = 1, shr = 1, add = 0, write = 0

# time = 90, a =12, b =6, product = 0000000000000000000000000000000100100000000000000000000000000000
# result = 301989888
# less32 = 1, shr = 0, add = 0, write = 0
#
# time = 94, a =12, b =6, product = 0000000000000000000000000000000100100000000000000000000000000000
# result = 301989888
# less32 = 1, shr = 1, add = 0, write = 0
#
# time = 98, a =12, b =6, product = 0000000000000000000000000000000100100000000000000000000000000000
# result = 150994944
# less32 = 1, shr = 0, add = 0, write = 0
#
# time = 102, a =12, b =6, product = 0000000000000000000000000000000100100000000000000000000000000000
# result = 150994944
# less32 = 1, shr = 1, add = 0, write = 0
#
# time = 106, a =12, b =6, product = 0000000000000000000000000000000100100000000000000000000000000000
# result = 75497472
# less32 = 1, shr = 0, add = 0, write = 0
#
# time = 110, a =12, b =6, product = 0000000000000000000000000000000100100000000000000000000000000000
# result = 75497472
# less32 = 1, shr = 1, add = 0, write = 0
#
# time = 114, a =12, b =6, product = 0000000000000000000000000000000100100000000000000000000000000000
# result = 37748736
# less32 = 1, shr = 0, add = 0, write = 0
#
# time = 118, a =12, b =6, product = 0000000000000000000000000000000100100000000000000000000000000000
# result = 37748736
# less32 = 1, shr = 1, add = 0, write = 0
```

```
# time = 122, a =12, b =6, product = 00000000000000000000000000000000000000000100100000000000000000000000
# result = 18874368
# less32 = 1, shr = 0, add = 0, write = 0
#
# time = 126, a =12, b =6, product = 00000000000000000000000000000000000000000100100000000000000000000000
# result = 18874368
# less32 = 1, shr = 1, add = 0, write = 0
#
# time = 130, a =12, b =6, product = 00000000000000000000000000000000000000000100100000000000000000000000
# result = 9437184
# less32 = 1, shr = 0, add = 0, write = 0
#
# time = 134, a =12, b =6, product = 00000000000000000000000000000000000000000100100000000000000000000000
# result = 9437184
# less32 = 1, shr = 1, add = 0, write = 0
#
# time = 138, a =12, b =6, product = 00000000000000000000000000000000000000000100100000000000000000000000
# result = 4718592
# less32 = 1, shr = 0, add = 0, write = 0
#
# time = 142, a =12, b =6, product = 00000000000000000000000000000000000000000100100000000000000000000000
# result = 4718592
# less32 = 1, shr = 1, add = 0, write = 0
#
# time = 146, a =12, b =6, product = 00000000000000000000000000000000000000000100100000000000000000000000
# result = 2359296
# less32 = 1, shr = 0, add = 0, write = 0
#
# time = 150, a =12, b =6, product = 00000000000000000000000000000000000000000100100000000000000000000000
# result = 2359296
# less32 = 1, shr = 1, add = 0, write = 0
#

# time = 154, a =12, b =6, product = 00000000000000000000000000000000000000000100100000000000000000000000
# result = 1179648
# less32 = 1, shr = 0, add = 0, write = 0
#
# time = 158, a =12, b =6, product = 00000000000000000000000000000000000000000100100000000000000000000000
# result = 1179648
# less32 = 1, shr = 1, add = 0, write = 0
#
# time = 162, a =12, b =6, product = 00000000000000000000000000000000000000000100100000000000000000000000
# result = 589824
# less32 = 1, shr = 0, add = 0, write = 0
#
# time = 166, a =12, b =6, product = 00000000000000000000000000000000000000000100100000000000000000000000
# result = 589824
# less32 = 1, shr = 1, add = 0, write = 0
#
# time = 170, a =12, b =6, product = 00000000000000000000000000000000000000000100100000000000000000000000
# result = 294912
# less32 = 1, shr = 0, add = 0, write = 0
#
# time = 174, a =12, b =6, product = 00000000000000000000000000000000000000000100100000000000000000000000
# result = 294912
# less32 = 1, shr = 1, add = 0, write = 0
#
# time = 178, a =12, b =6, product = 00000000000000000000000000000000000000000100100000000000000000000000
# result = 147456
# less32 = 1, shr = 0, add = 0, write = 0
#
# time = 182, a =12, b =6, product = 00000000000000000000000000000000000000000100100000000000000000000000
# result = 147456
# less32 = 1, shr = 1, add = 0, write = 0
#
```

```
# time = 186, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000100100000000000000
# result = 73728
# less32 = 1, shr = 0, add = 0, write = 0
#
# time = 190, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000100100000000000000
# result = 73728
# less32 = 1, shr = 1, add = 0, write = 0
#
# time = 194, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000100100000000000000
# result = 36864
# less32 = 1, shr = 0, add = 0, write = 0
#
# time = 198, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000100100000000000000
# result = 36864
# less32 = 1, shr = 1, add = 0, write = 0
#
# time = 202, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000100100000000000000
# result = 18432
# less32 = 1, shr = 0, add = 0, write = 0
#
# time = 206, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000100100000000000000
# result = 18432
# less32 = 1, shr = 1, add = 0, write = 0
#
# time = 210, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000100100000000000000
# result = 9216
# less32 = 1, shr = 0, add = 0, write = 0
#
# time = 214, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000100100000000000000
# result = 9216
# less32 = 1, shr = 1, add = 0, write = 0
#
# time = 218, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000100100000000000000
# result = 4608
# less32 = 1, shr = 0, add = 0, write = 0
#
# time = 222, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000100100000000000000
# result = 4608
# less32 = 1, shr = 1, add = 0, write = 0
#
# time = 226, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000100100000000000000
# result = 2304
# less32 = 1, shr = 0, add = 0, write = 0
#
# time = 230, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000100100000000000000
# result = 2304
# less32 = 1, shr = 1, add = 0, write = 0
#
# time = 234, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000100100000000000000
# result = 1152
# less32 = 1, shr = 0, add = 0, write = 0
#
# time = 238, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000100100000000000000
# result = 1152
# less32 = 1, shr = 1, add = 0, write = 0
#
# time = 242, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000100100000000000000
# result = 576
# less32 = 1, shr = 0, add = 0, write = 0
#
# time = 246, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000100100000000000000
# result = 576
# less32 = 1, shr = 1, add = 0, write = 0
```

Result is found

```
# time = 246, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000001001000000
# result = 576
# less32 = 1, shr = 1, add = 0, write = 0
#
# time = 250, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000000100100000
# result = 288
# less32 = 1, shr = 0, add = 0, write = 0
#
# time = 254, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000000100100000
# result = 288
# less32 = 1, shr = 1, add = 0, write = 0
#
# time = 258, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000000010010000
# result = 144
# less32 = 1, shr = 0, add = 0, write = 0
#
# time = 262, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000000010010000
# result = 144
# less32 = 1, shr = 1, add = 0, write = 0
#
# time = 266, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000000001001000
# result = 72
# less32 = 0, shr = 0, add = 0, write = 0
#
# time = 270, a =12, b =6, product = 00000000000000000000000000000000000000000000000000000000001001000
# result = 72
# less32 = 0, shr = 1, add = 0, write = 0
# Break in Module mult32_tb at C:/Users/burak kocausta/Desktop/cse331/homework/hw2/verilog/mult32_tb.v line 15
```

mult32_tb2: (only results here)

```
# time = 282, a =35, b =190, product = 00000000000000000000000000000000000000000000000000001100111111010
# result = 6650
# less32 = 0, shr = 0, add = 0, write = 0

# time = 570, a =57, b =813, product = 00000000000000000000000000000000000000000000000001011010100000101
# result = 46341
# less32 = 0, shr = 0, add = 1, write = 1

# time = 838, a =111, b =20, product = 00000000000000000000000000000000000000000000000001000101011100
# result = 2220
# less32 = 0, shr = 0, add = 0, write = 0
```

alu32_tb1:

```
#
# time = 0
# ALUOP = 000 (decimal= 0)
# a = 00000000000000000000000000001100 (decimal= 12)
# b = 00000000000000000000000000001000 (decimal= 8)
# result = 00000000000000000000000000010100 (decimal = 20)
#
# time = 30
# ALUOP = 001 (decimal= 1)
# a = 00000000000000000000000000001100 (decimal= 12)
# b = 00000000000000000000000000001000 (decimal= 8)
# result = 00000000000000000000000000000100 (decimal = 4)
```

```
time = 320
ALUOP = 010 (decimal= 2)
a = 00000000000000000000000000001100 (decimal= 12)
b = 00000000000000000000000000001000 (decimal= 8)
result = 00000000000000000000000001100000 (decimal = 96)

time = 324
ALUOP = 011 (decimal= 3)
a = 00000000000000000000000000001100 (decimal= 12)
b = 00000000000000000000000000001000 (decimal= 8)
result = 00000000000000000000000000000100 (decimal = 4)

time = 354
ALUOP = 100 (decimal= 4)
a = 00000000000000000000000000001100 (decimal= 12)
b = 00000000000000000000000000001000 (decimal= 8)
result = 00000000000000000000000000001000 (decimal = 8)

time = 384
ALUOP = 101 (decimal= 5)
a = 00000000000000000000000000001100 (decimal= 12)
b = 00000000000000000000000000001000 (decimal= 8)
result = 00000000000000000000000000001100 (decimal = 12)

# time = 414
# ALUOP = 110 (decimal= 6)
# a = 00000000000000000000000000001100 (decimal= 12)
# b = 00000000000000000000000000001000 (decimal= 8)
# result = 00000000000000000000000000000000 (decimal = 0)
#
# time = 444
# ALUOP = 111 (decimal= 7)
# a = 00000000000000000000000000001100 (decimal= 12)
# b = 00000000000000000000000000001000 (decimal= 8)
# result = 11111111111111111111111111110011 (decimal = -13)
```

alu32_tb2: (multiply is not tested here because test is for signed numbers)

```
# time = 0
# ALUOP = 000 (decimal= 0)
# a = 11111111111111111111111111101100 (decimal= -20)
# b = 11111111111111111111111111100111 (decimal= -25)
# result = 11111111111111111111111111010011 (decimal = -45)
#
# time = 30
# ALUOP = 001 (decimal= 1)
# a = 11111111111111111111111111101100 (decimal= -20)
# b = 11111111111111111111111111100111 (decimal= -25)
# result = 00000000000000000000000000000101 (decimal = 5)
#
# time = 60
# ALUOP = 011 (decimal= 3)
# a = 11111111111111111111111111101100 (decimal= -20)
# b = 11111111111111111111111111100111 (decimal= -25)
# result = 00000000000000000000000000001011 (decimal = 11)
#
# time = 90
# ALUOP = 100 (decimal= 4)
# a = 11111111111111111111111111101100 (decimal= -20)
# b = 11111111111111111111111111100111 (decimal= -25)
# result = 11111111111111111111111111100100 (decimal = -28)
#
# time = 120
# ALUOP = 101 (decimal= 5)
# a = 11111111111111111111111111101100 (decimal= -20)
# b = 11111111111111111111111111100111 (decimal= -25)
# result = 11111111111111111111111111101111 (decimal = -17)
```

```
# time = 150
# ALUOP = 110 (decimal= 6)
# a = 11111111111111111111111111101100 (decimal= -20)
# b = 11111111111111111111111111100111 (decimal= -25)
# result = 00000000000000000000000000000000 (decimal = 0)
#
# time = 180
# ALUOP = 111 (decimal= 7)
# a = 11111111111111111111111111101100 (decimal= -20)
# b = 11111111111111111111111111100111 (decimal= -25)
# result = 00000000000000000000000000010000 (decimal = 16)
```