Lab Report #1(Resubmit)

Name: Ruochen Duan
Student ID: 1405106

The modified parts are SignExtend andLeftShift2, and the entity name of MUX32 is changed to MUX64.

Explanation and Description

① AND2

I choose the behavioral for modeling type. And the truth table of the AND2 gate is

In0	In1	Output
0	0	0
0	1	0
1	0	0
1	1	1

My testbench is that during the simulation of 200ns, I first set

In0 = 0 In1 = 0 for 50ns

Then set

In0 = 0 In1 = 1 for 50ns

Then set

In0 = 1 In1 = 0 for 50ns

Then set

In0 = 1 In1 = 1

The result is showed in Figure 1

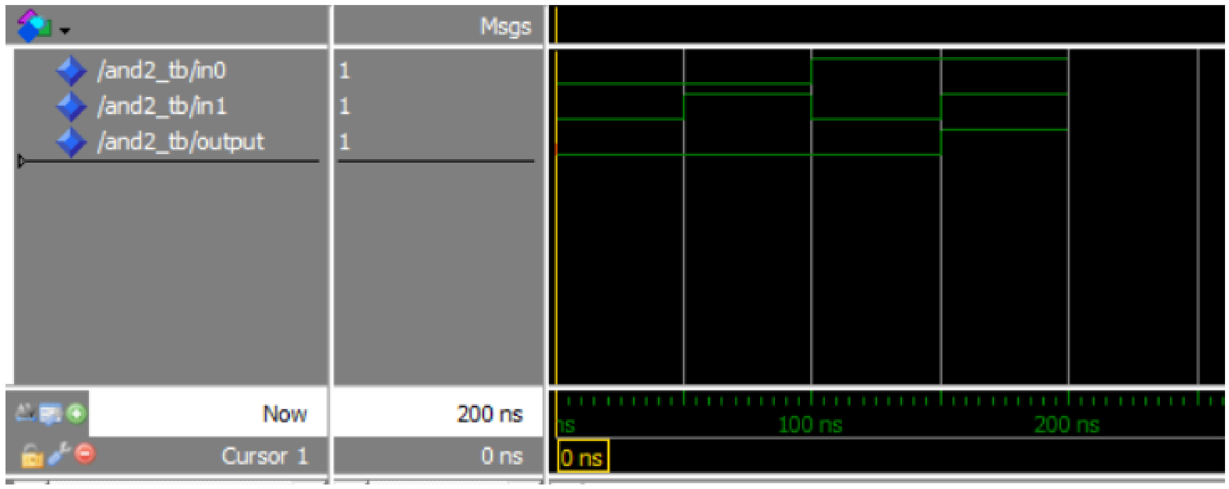


Figure 1

We can wee from the Figure 1 that the output is '1' during t = 150 ns to t =200 ns. The simulation results are the same as the truth table.

② MUX5

I choose the behavioral for modeling type. MUX5 is a two by one mux with 5-bit inputs/outputs. When the sel =0, the ouput = in0. When the sel =1, the ouput = in1.

My testbench is that during the simulation of 200ns, I first set

Sel = 1 In0 = "00000" In1 = "11110" for 50 ns

Then set

Sel = 1 In0 = "00000" In1 = "10100" for 50 ns

Then set

Sel = 0 In0 = "01100" In1 = "10100" for 50 ns

Then set

Sel = 0 In0 = "10010" In1 = "01010"

The simulation result is showed in Figure 2

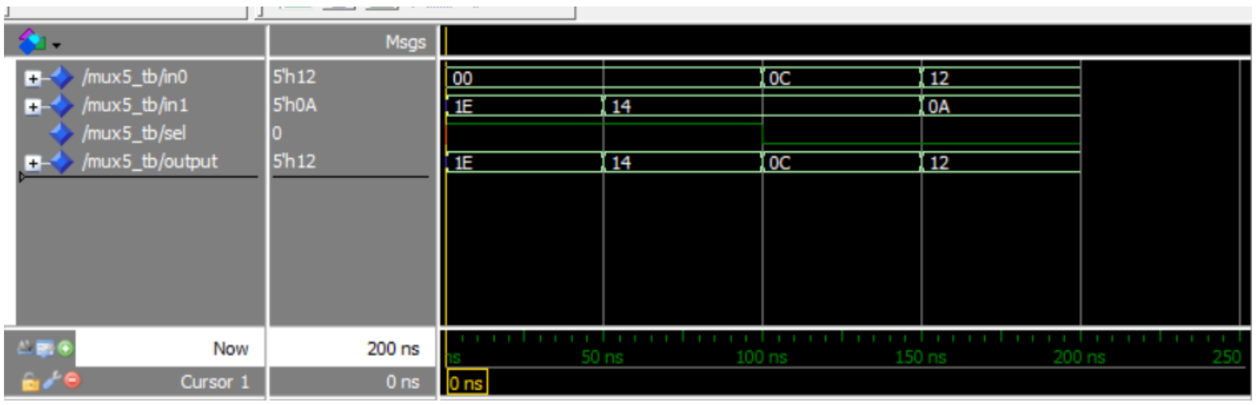


Figure 2

We can see from the simulation result that
during t = 0 ns to t =50 ns,
the output = 0x1e which is the input of In1, when sel = 1.
during t = 50 ns to t =100 ns,
the output = 0x14 which is the input of In1, when sel = 1.
during t = 100 ns to t =150 ns,
the output = 0x0c which is the input of In0, when sel = 0.
during t = 150 ns to t =200 ns,
the output = 0x12 which is the input of In0, when sel = 0. Even the input of the In1is also changed.

③ MUX64

I choose the behavioral for modeling type. The function of the MUX64 is same as the function of MUX5 but change the input to 64-bit.

My testbench is that during the simulation of 200ns, I first set
Sel = 1
In0=
"0000000000000000000000000000000000000000000000000000000000000000"
In1=
"1111000000000000000000000000000000000000000000000000000000000000"
for 50 ns

Then set
Sel = 1
In0=
"0000000000000000000000000000000000000000000000000000000000000000"
In1=
"1010000000000000000000000000000000000000000000000000000000000000"

for 50 ns

Then set

Sel = 0

In0=
“0110000000000000000000000000000000000000000000000000000000000000"

In1=
"1010000000000000000000000000000000000000000000000000000000000000"

for 50 ns

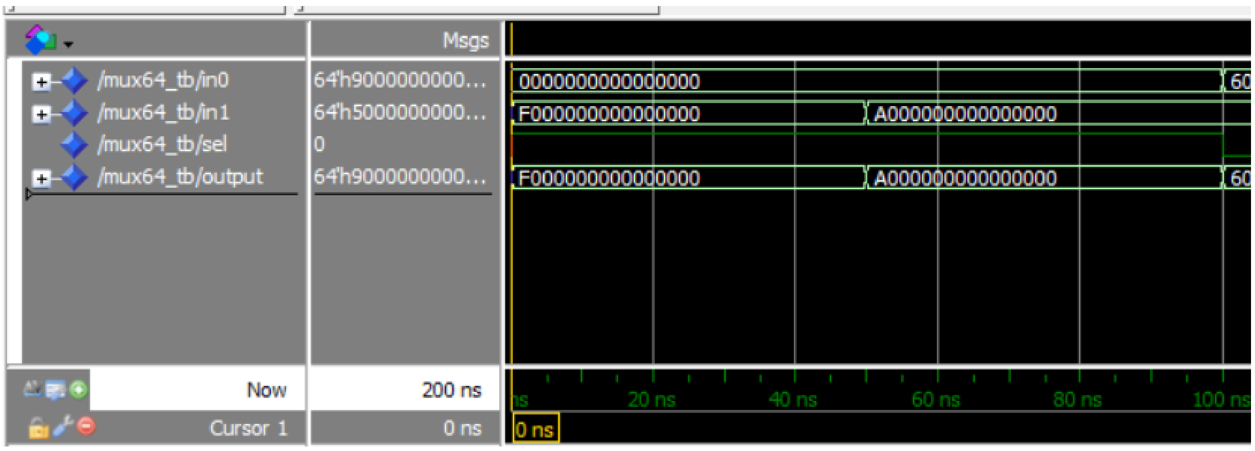
Then set

Sel = 0

In0=
"1001000000000000000000000000000000000000000000000000000000000000"

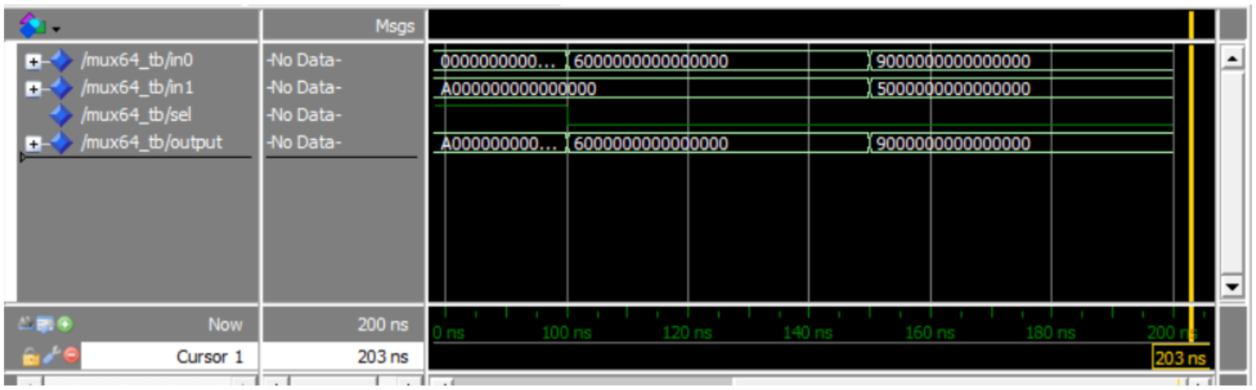
In1=
"0101000000000000000000000000000000000000000000000000000000000000"

The simulation result is showed in Figure 3 and Figure 4



```
                   Msgs
/mux64_tb/in0      64'h9000000000...   0000000000000000                              60
/mux64_tb/in1      64'h5000000000...   F000000000000000       A000000000000000
/mux64_tb/sel      0
/mux64_tb/output   64'h9000000000...   F000000000000000       A000000000000000       60

Now                200 ns              ns   20 ns   40 ns   60 ns   80 ns   100 ns
Cursor 1           0 ns                0 ns
```

Figure 3(t=0ns to t=100ns)



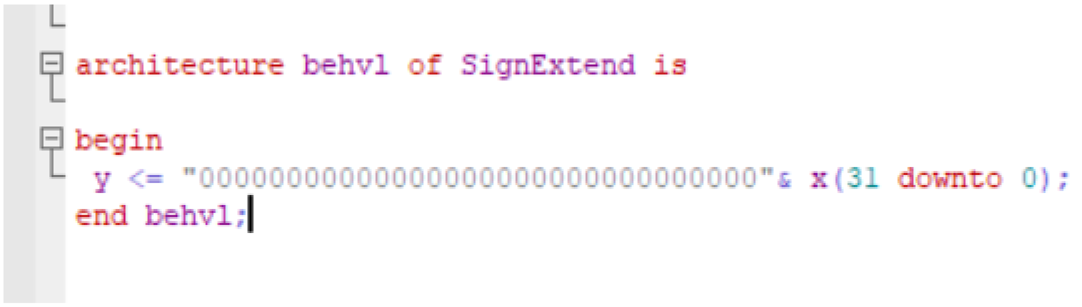
```
                    Msgs
/mux64_tb/in0      -No Data-   0000000000...  6000000000000000  9000000000000000
/mux64_tb/in1      -No Data-   A000000000000000                 5000000000000000
/mux64_tb/sel      -No Data-
/mux64_tb/output   -No Data-   A000000000...  6000000000000000  9000000000000000

Now       200 ns   0 ns  100 ns  120 ns  140 ns  160 ns  180 ns  200 ns
Cursor 1  203 ns                                                 203 ns
```

Figure 4(t=100ns to t=200ns)

We can see from the simulation result that
during t = 0 ns to t =50 ns,
the output is equal to the input of In1, when sel = 1.
during t = 50 ns to t =100 ns,
the output is equal to the input of In1, when sel = 1.
during t = 100 ns to t =150 ns,
the output is equal to the input of In0, when sel = 0.
during t = 150 ns to t =200 ns,
the output is equal to the input of In0, when sel = 0. Even the input of In1 is changed.

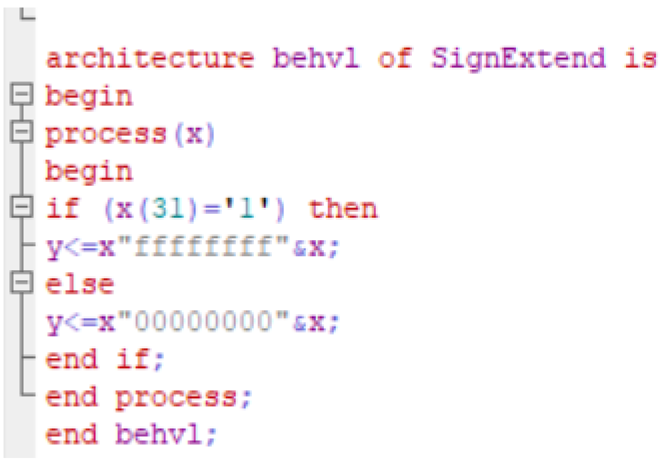
④ SignExtend(code and testbench are modified)

Before



```
architecture behvl of SignExtend is

begin
  y <= "00000000000000000000000000000000"& x(31 downto 0);
end behvl;
```

After



```vhdl
architecture behvl of SignExtend is
begin
process(x)
begin
if (x(31)='1') then
y<=x"ffffffff"&x;
else
y<=x"00000000"&x;
end if;
end process;
end behvl;
```

I choose the behavioral for modeling type. The function of SignExtend is to extend a 32-bit input to a 64-bit output by extending the most significant bit.

My testbench is that during the simulation of 200ns, I first set

x = x"11110000" for 50 ns

then set

x = x"01010101" for 50 ns

the set

x = x "f0001111" for 50 ns (modified)

the set

x = x "10110100"

The simulation result is showed in Figure 5 and Figure 6

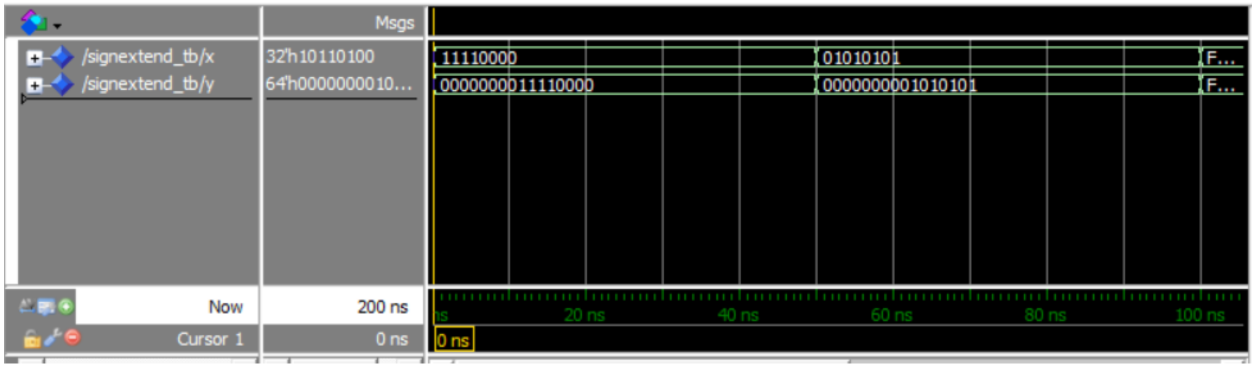


Figure 5(t=0ns to t=100ns)

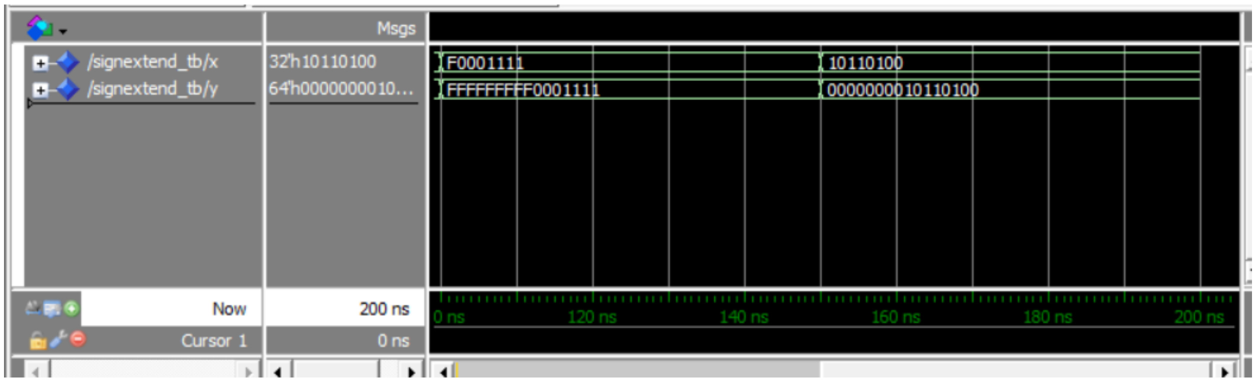
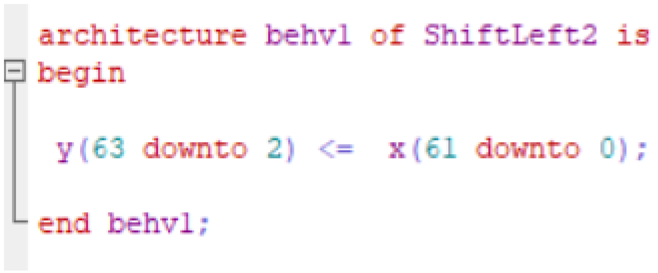

```
/signextend_tb/x   32'h10110100         F0001111              10110100
/signextend_tb/y   64'h0000000010...    FFFFFFFFF0001111      0000000010110100
Now      200 ns
Cursor 1   0 ns
```

Figure 6(t=100ns to t=200ns)

We can see from the simulation result that the value of the output y is same as the input x, but the length of the signal is changed.

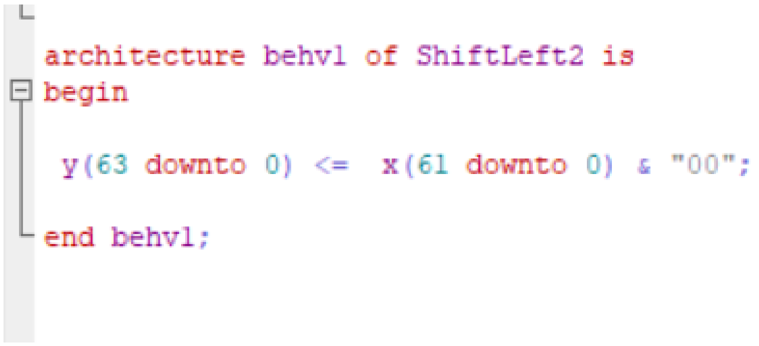
⑤ Shift Left 2(code is modified)

Before



```
architecture behvl of ShiftLeft2 is
begin

 y(63 downto 2) <=  x(61 downto 0);

end behvl;
```

After



```
architecture behvl of ShiftLeft2 is
begin

 y(63 downto 0) <=  x(61 downto 0) & "00";

end behvl;
```

I choose the behavioral for modeling type. The function of ShiftLeft2 is to Shifts the input by 2 bits. The input bits higher than 32 will be discarded and 0s will be added on the lowest two bits of the input.

My testbench is that during the simulation of 200ns, I first set

x = x "0000000000000010"for 50 ns

then set

x = x" 0101010101010101" for 50 ns

the set

x = x " FFFFFFFFFFFFFFFF" for 50 ns

the set

x = x " 0000000000000001"

The simulation result is showed in Figure 7 and Figure 8

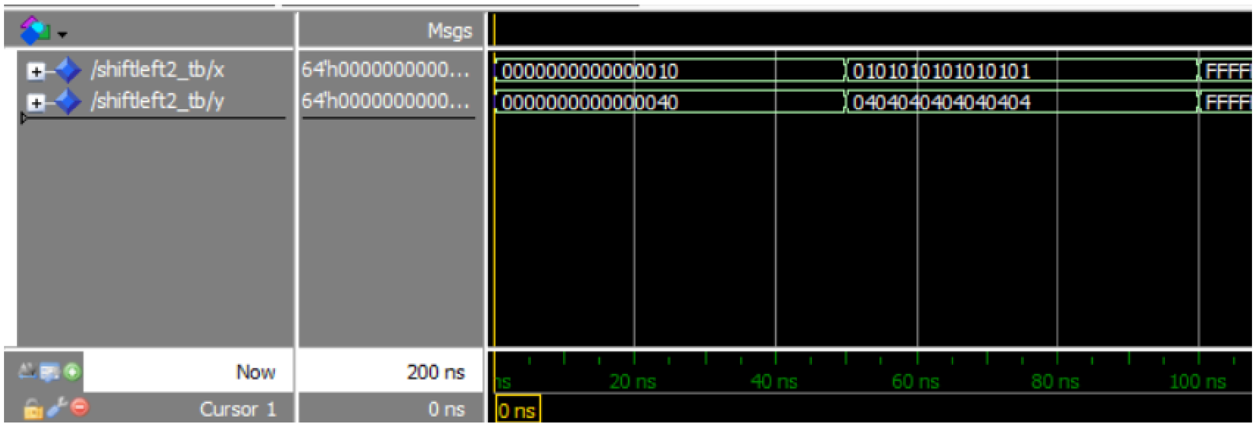


Figure 7(t=0ns to t=100ns)

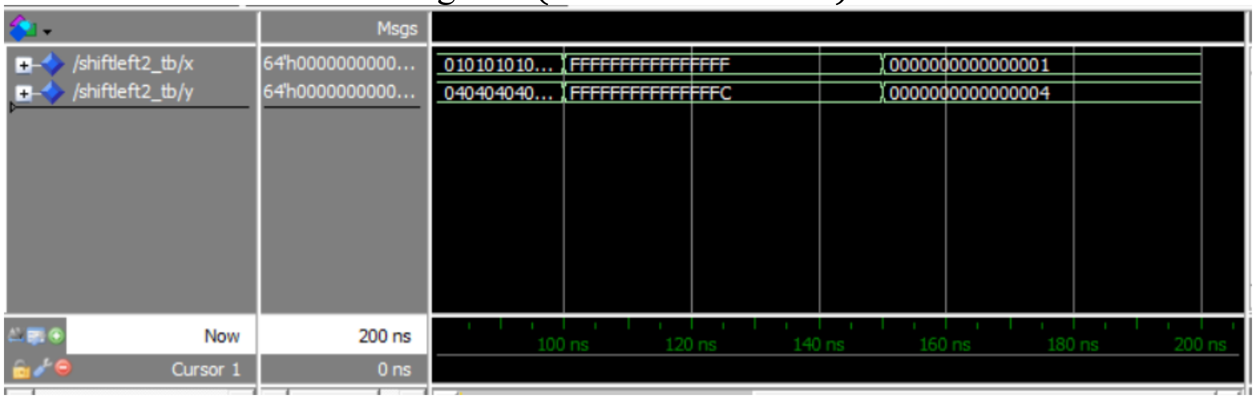


Figure 8(t=100ns to t=200ns)

We can see from the simulation result that

During the t = 0 ns to 50 ns

y =x”0000000000000040” which is input x shifting 2 bits toward left.

During the t = 50 ns to 100 ns

y =x”0404040404040404” which is input x shifting 2 bits toward left.

During the t = 100 ns to 150 ns

y =x”FFFFFFFFFFFFFFFC” which is input x shifting 2 bits toward left.

During the t = 100 ns to 200 ns

y =x”0000000000000004” which is input x shifting 2 bits toward left.

⑥ PC

I choose the behavioral for modeling type. PC is program counter. The PC in the Lab has 4 input which are clk, write_enable, rst and AddressIn and has one output which is AddressOut. The main function of this PC is controlled by the clk and write_enable. When the PC receive the rising-edge of clk(which means the moment clk is changed to ‘1’ form ‘0’) and write_enable =’1’ at that time, the input of AddressIn will be assigned to output AddressOut (AddressOut = AddressIn). But at any situation ,when the rst = 1, the output AddressOut will be changed to x”0000000000000000’.

My testbench is that during the simulation of 600ns, I first set

clk = 0
write_enable=1
rst = 0
AddressIn = x"aaaaaaaaaaaaaaaa" for 50 ns

then set

clk = 1
write_enable=1
rst = 0
AddressIn keeps its value for 50 ns

then set

clk = 0
write_enable=1
rst = 1
AddressIn = x"1234123412341234" for 50 ns

then set

clk = 1
write_enable=1
rst = 1
AddressIn keeps its value for 50 ns

then set

clk = 0
write_enable=1
rst = 0
AddressIn = x"cccccccccccccccc" for 50 ns

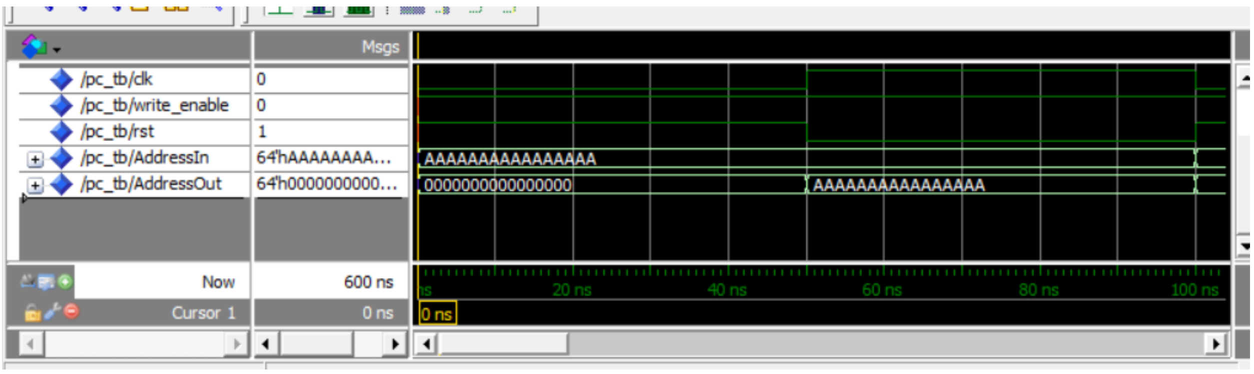
then set
  clk = 1
  write_enable=1
  rst = 0
  AddressIn keeps its value for 50 ns

then set
 clk = 0
  write_enable=1
  rst = 0
  AddressIn = x"4567456745674567"   for 50 ns
then set
  clk = 0
  write_enable=1
  rst = 0
  AddressIn keeps its value for 50 ns

then set
 clk = 0
  write_enable=1
  rst = 0
  AddressIn = x"FFFFFFFFFFFFFFFF"   for 50 ns
then set
  clk = 1
  write_enable=1
  rst = 0
  AddressIn keeps its value for 50 ns

then set
 clk = 1
  write_enable=1
  rst = 0
  AddressIn = x"aaaaaaaaaaaaaaaa"   for 50 ns
then set
  clk = 0
  write_enable=0
  rst = 1
  AddressIn keeps its value for 50 ns

The simulation result is showed in Figure 9 and Figure 14. And I will analyze the result under each Figure.



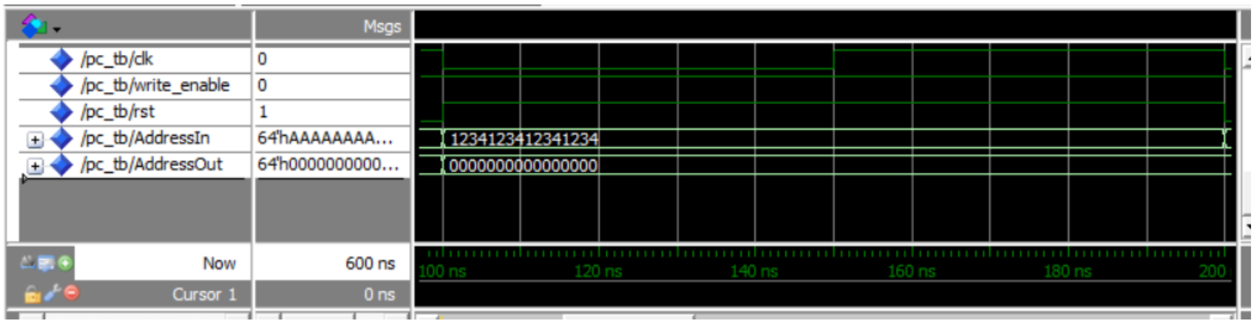
```
/pc_tb/clk            0
/pc_tb/write_enable   0
/pc_tb/rst            1
/pc_tb/AddressIn      64'hAAAAAAAA...   AAAAAAAAAAAAAAAA
/pc_tb/AddressOut     64'h0000000000... 0000000000000000  AAAAAAAAAAAAAAAA
Now      600 ns
Cursor 1   0 ns
```

Figure 9(t=0ns to t=100ns)

We can see from the Figure 9 that

During t = 0 ns to t=50 ns, the output AddressOut =0, because rst = ‘0’.

At t=50 ns, the output AddressOut changed its value to x”AAAAAAAAAAAAAAAA”, because at that time the PC received a rising edge of clk and write_enable =1, so the value of AddressIn was assigned to AddressOut.



```
/pc_tb/clk            0
/pc_tb/write_enable   0
/pc_tb/rst            1
/pc_tb/AddressIn      64'hAAAAAAAA...   1234123412341234
/pc_tb/AddressOut     64'h0000000000... 0000000000000000
Now      600 ns
Cursor 1   0 ns
```

Figure 10(t=100ns to t=200ns)

We can see from the Figure 9 that

At t = 100 ns, the output AddressOut was changed to 0 from x”AAAAAAAAAAAAAAAA”, because at that time the signal rst is set to 1.

At t= 150 ns , the output AddressOut wat still 0 even at that moment the PC received a rising edge of clk and write_enable= 0. Because rst was set to 1 during t = 100ns to t =200s.

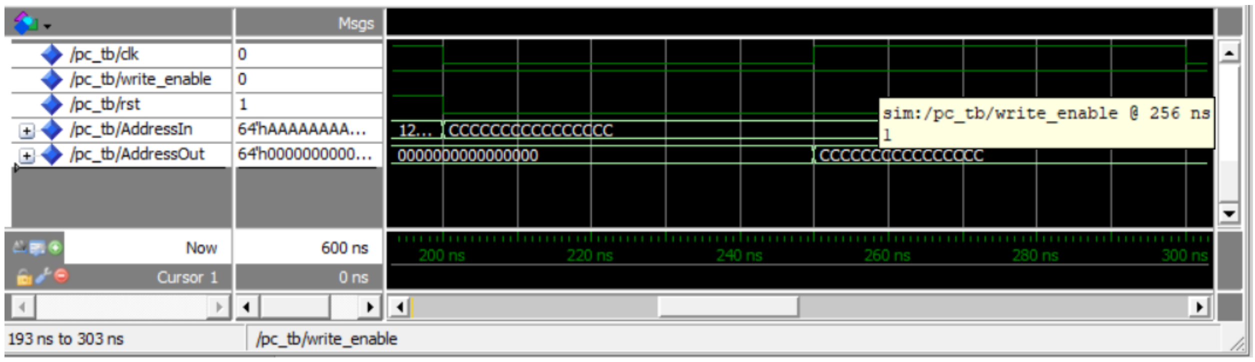


Figure 11(t=200ns to t=300ns)

At t=250 ns, the output AddressOut changed its value to x"CCCCCCCCCCCCCCCC", because at that time the PC received a rising edge of clk and write_enable =1, so the value of AddressIn was assigned to AddressOut.

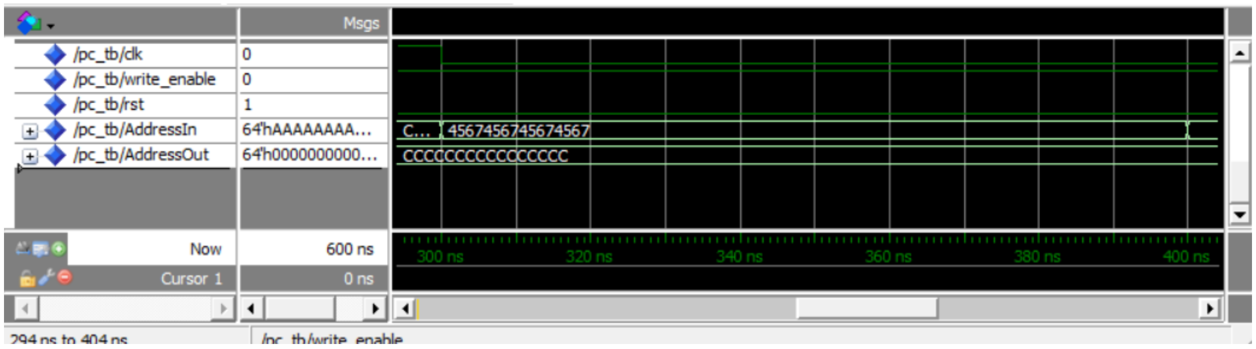


Figure 12(t=300ns to t=400ns)

During t =300ns to t = 400ns, I set clk = 0. We can see that the value of AddressOut was not changed at time t = 300 ns, even write_enable =1 and AddressIn was changed tox"4567456745674567".

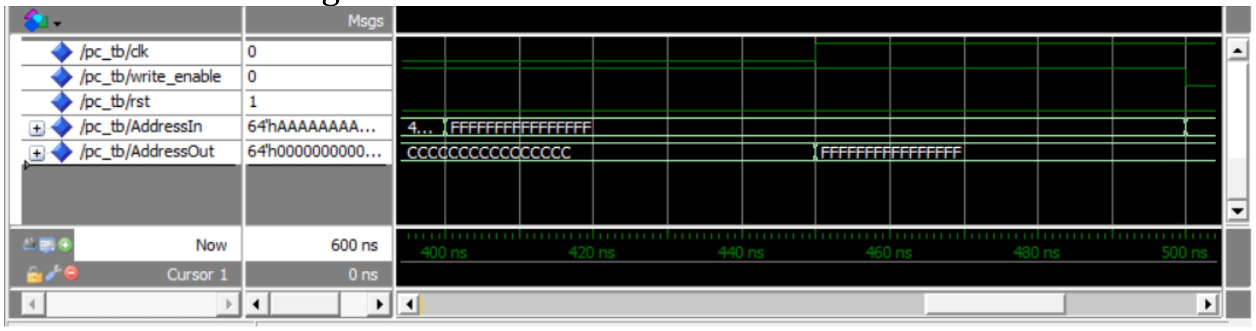
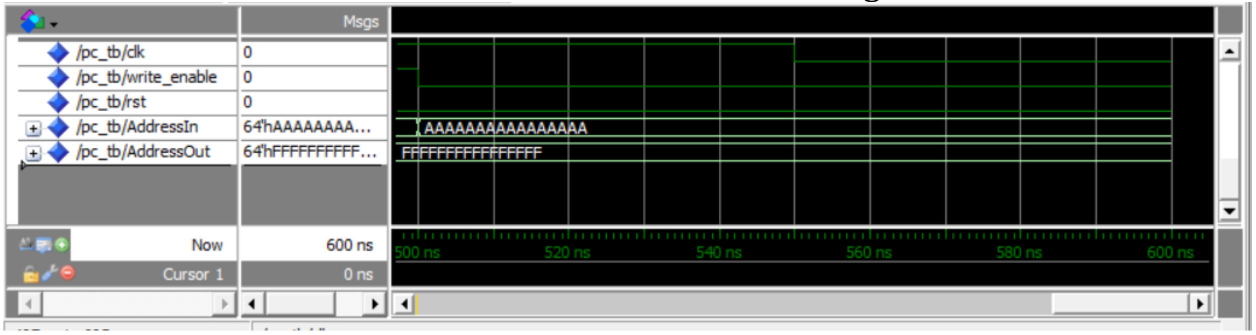


Figure 13(t=400ns to t=500ns)

At t=450 ns, the output AddressOut changed its value to x"FFFFFFFFFFFFFFFF", because at that time the PC received a rising edge of clk and write_enable =1, so the value of AddressIn was assigned to AddressOut.



```
/pc_tb/clk            0
/pc_tb/write_enable   0
/pc_tb/rst            0
/pc_tb/AddressIn      64'hAAAAAAAA...    AAAAAAAAAAAAAAAA
/pc_tb/AddressOut     64'hFFFFFFFFF...   FFFFFFFFFFFFFFFF
Now                   600 ns
Cursor 1              0 ns
```

Figure 14(t=500ns to t=600ns)

At t=550 ns, the output AddressOut didn't change its value to x"AAAAAAAAAAAAAAAA", even write_enable = 1 and rst = 0,because at that time the PC received a falling edge of clk , so the value of AddressIn was not assigned to AddressOut.