Name: Duc Anh Nguyen

Course: Computer Engineering with lab

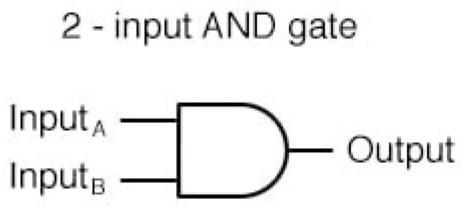
Date: 9.22.2022

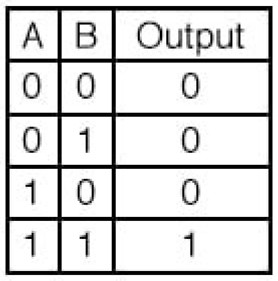
Lab Report 1

Introduction: The goal of this lab is to create basic process components, which include and gate, multiplexer, sign extender, program counter, and shift left gate. Moreover, test benches are needed to verify the function of these components.

Procedure:

1. **A brief description (a sentence or two) of what each component does:**
   a. AND2: It is an and gate with 2 inputs and 1 output with a basic truth table in the picture below.





A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

b. MUX5: It is a logic circuit designed to switch between input lines to the output. In this lab, we will only design a 2 to 1 mux with different bits of inputs/outputs. This multiplexer has 5 bits inputs/outputs.

c. MUX64: It is a logic circuit designed to switch between input lines to the output. This multiplexer has 64 bits inputs/outputs.

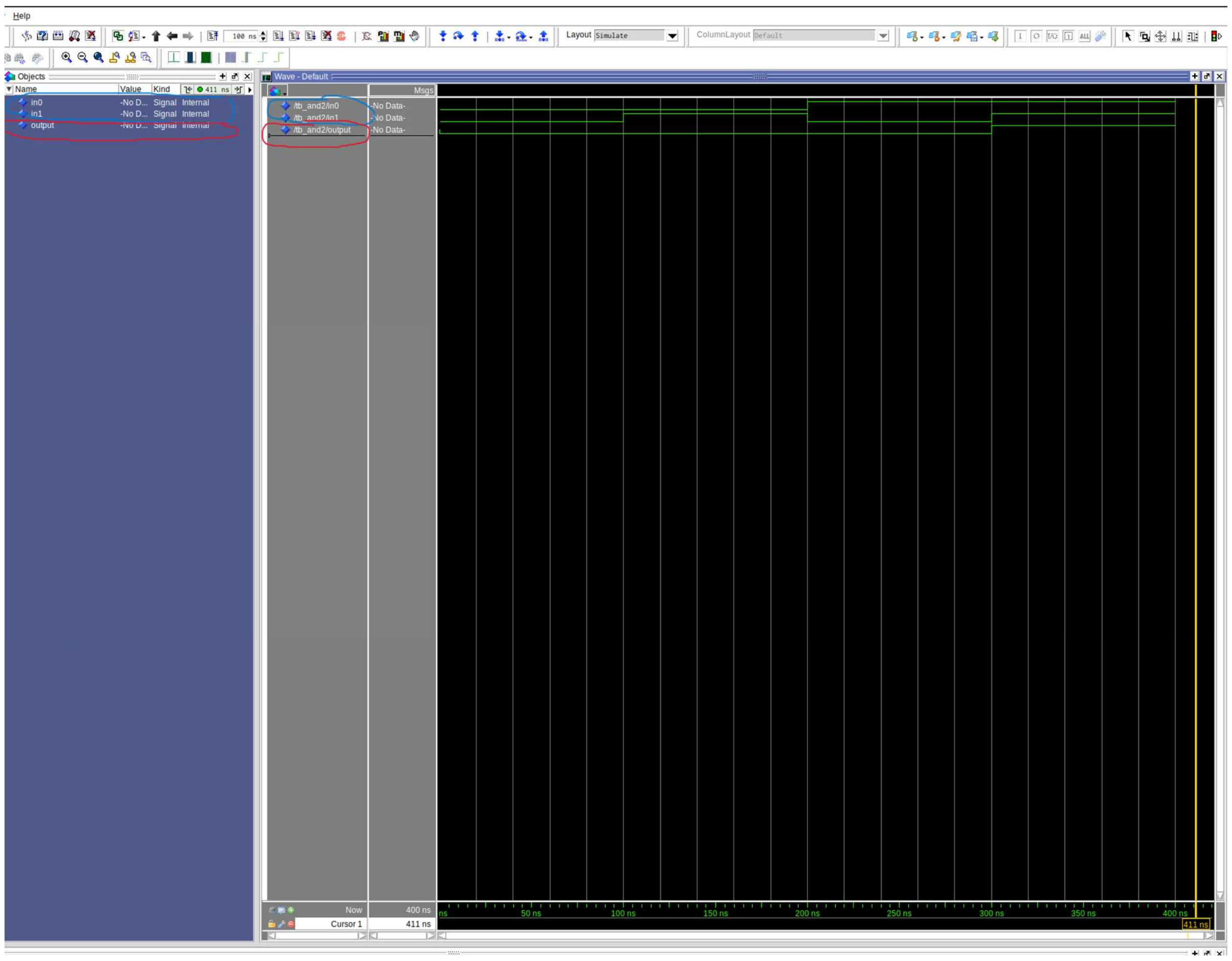
d. SignExtend: It is a component that increase the numbers of bits in a representation of a integers in two's complements. In this case, we are extending 32 bits to 64 bits two's complements.

e. ShiftLeft2: It is a component that shift the numbers of bits in a number by two to the left. It is a 64-bit component.

f. Program Counter: It is a 64-bit program counter that fetch the next address during the rising edge of the clock signal. It also includes a write-enable function, which is enable/disable inputting address, and a asynchronous reset to reset the program counter back to 0.

2. A brief explanation of your choices for modeling type (dataflow, behavioral, structural)
   - I choose behavioral modeling type because it is better to describer these components in sequence, and within a process to design my test benches. All the components are being designed using behavioral modeling.
3. Waveforms obtained by simulating your testbenches with brief descriptions for all the entities.

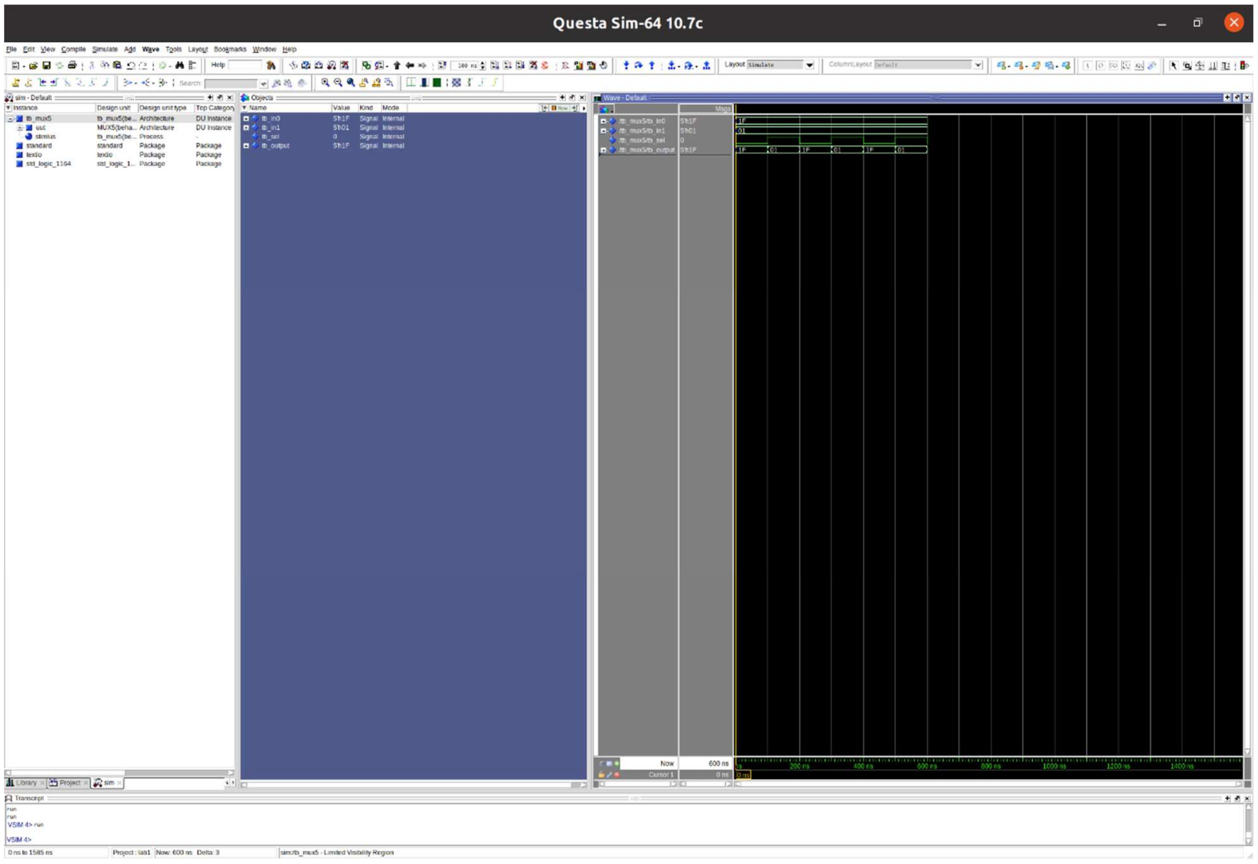
a. AND2: Basic AND gate which follow the truth table in part II.



Name	Value	Kind	
in0	-No D...	Signal	Internal
in1	-No D...	Signal	Internal
output	-No D...	Signal	Internal

	Msgs
/tb_and2/in0	-No Data-
/tb_and2/in1	-No Data-
/tb_and2/output	-No Data-

b. MUX5: This is 5 bit 2 to 1 multiplexer test bench. I assign in0 with “11111” which will be “1F” in hexadecimal, and in1 with “00001” which will be “01” in hexadecimal. In the figure, when the sel line switch to 0, the output will be in0 and vice versa.



Instance	Design unit	Design unit type	Top Category
tb_mux5	tb_mux5(be...	Architecture	DU Instance
uut	MUX5(beha...	Architecture	DU Instance
stimlus	tb_mux5(be...	Process	-
standard	standard	Package	Package
textio	textio	Package	Package
std_logic_1164	std_logic_1...	Package	Package

Name	Value	Kind	Mode
tb_in0	5'h1F	Signal	Internal
tb_in1	5'h01	Signal	Internal
tb_sel	0	Signal	Internal
tb_output	5'h1F	Signal	Internal

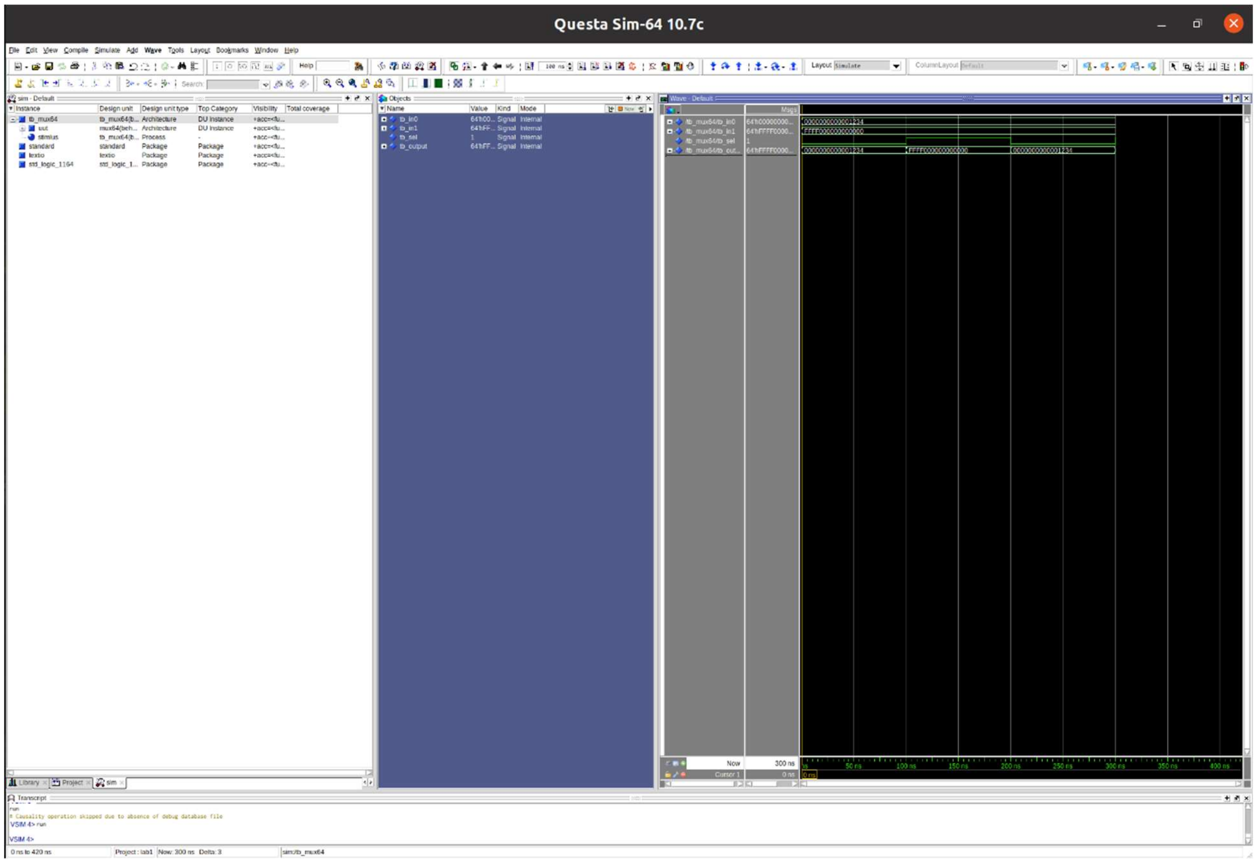
```
/tb_mux5/tb_in0     5'h1F   1F
/tb_mux5/tb_in1     5'h01   01
/tb_mux5/tb_sel     0
/tb_mux5/tb_output  5'h1F   1F  01  1F  01  1F  01
```

```
run
run
VSIM 4> run

VSIM 4>
```

```
0 ns to 1585 ns    Project : lab1   Now: 600 ns  Delta: 3    sim:/tb_mux5 - Limited Visibility Region
```

c. MUX64: This is 5 bit 2 to 1 multiplexer test bench. I assign in0 with "0000000000001234" hexadecimal, and in1 with "FFFF000000000000" in hexadecimal. In the figure, when the sel line switch to 0, the output will be in0 and vice versa.



Instance	Design unit	Design unit type	Top Category	Visibility	Total coverage
tb_mux64	tb_mux64(b...	Architecture	DU Instance	+acc=<fu...	
uut	mux64(beh...	Architecture	DU Instance	+acc=<fu...	
stimulus	tb_mux64(b...	Process	-	+acc=<fu...	
standard	standard	Package	Package	+acc=<fu...	
textio	textio	Package	Package	+acc=<fu...	
std_logic_1164	std_logic_1...	Package	Package	+acc=<fu...	

Name	Value	Kind	Mode
tb_in0	64'h00...	Signal	Internal
tb_in1	64'hFF...	Signal	Internal
tb_sel	1	Signal	Internal
tb_output	64'hFF...	Signal	Internal

	Msgs			
/tb_mux64/tb_in0	64'h00000000...	0000000000001234		
/tb_mux64/tb_in1	64'hFFFF0000...	FFFF000000000000		
/tb_mux64/tb_sel	1			
/tb_mux64/tb_out...	64'hFFFF0000...	0000000000001234	FFFF000000000000	0000000000001234

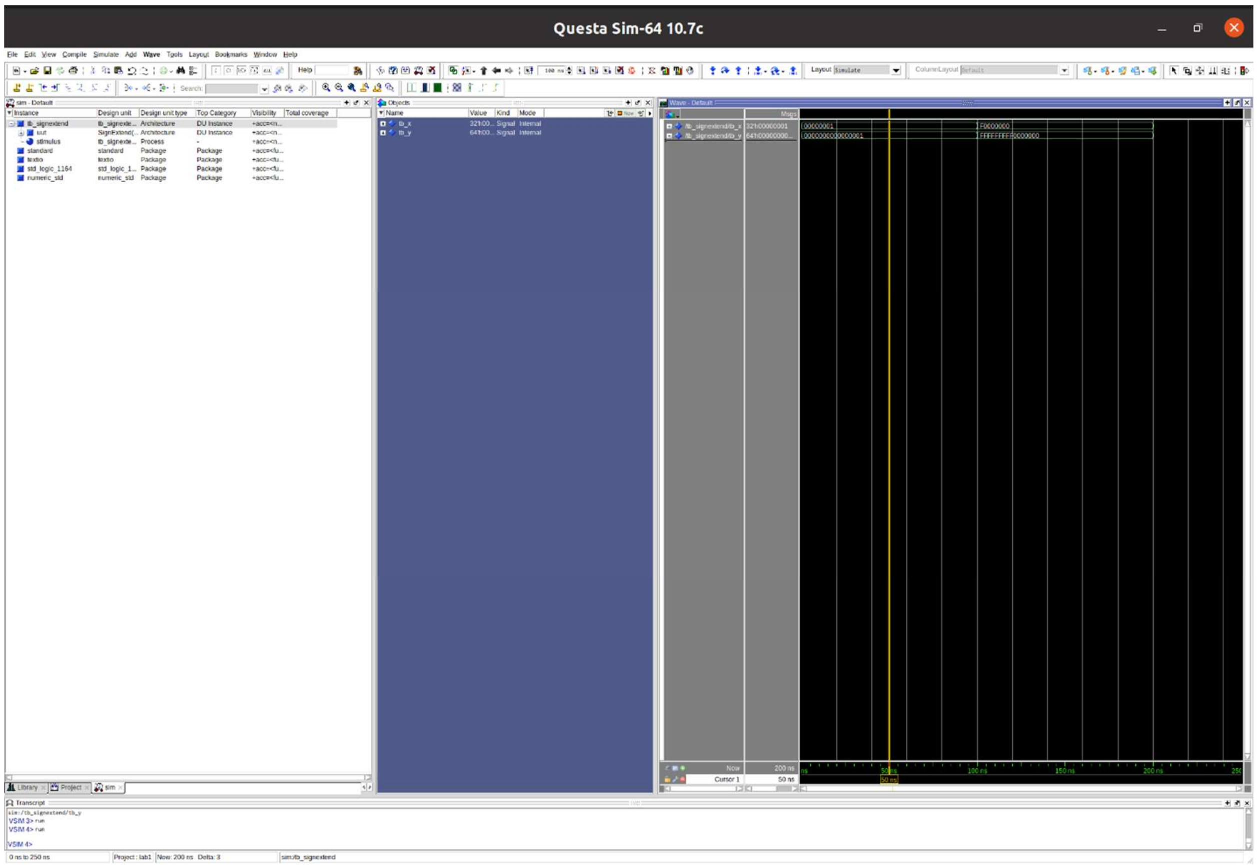
```
Now 300 ns
Cursor 1 0 ns
ns 50 ns 100 ns 150 ns 200 ns 250 ns 300 ns 350 ns 400 ns
```

```
run
# Causality operation skipped due to absence of debug database file
VSIM 4> run

VSIM 4>
```

```
0 ns to 420 ns | Project : lab1 | Now: 300 ns  Delta: 3 | sim:/tb_mux64
```

d. SignExtend: In every 100 ns, I assign a different value for x. For the first 100 ns, I assign “00000001” in hexadecimal, which got turned into “0000000000000001” by the components. For the second 100 ns, I assign “F0000000” in hexadecimal, which got turned into “FFFFFFFFF0000000” by the components.



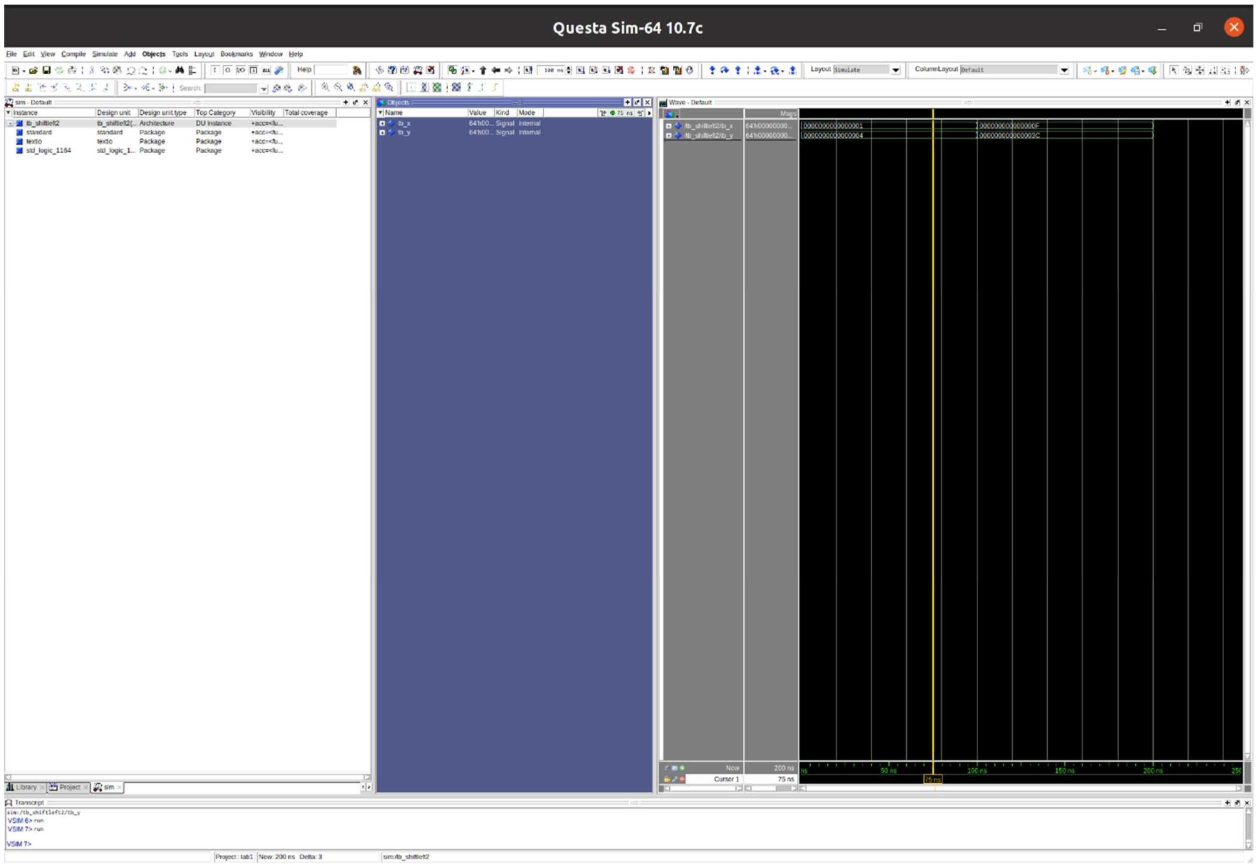
```
/tb_signextend/tb_x  32'h00000001   00000001          F0000000
/tb_signextend/tb_y  64'h00000000...  0000000000000001  FFFFFFFFF0000000
```

```
sim:/tb_signextend/tb_y
VSIM 3> run
VSIM 4> run

VSIM 4>
```

```
0 ns to 250 ns    Project : lab1    Now: 200 ns  Delta: 3    sim:/tb_signextend
```

e. ShiftLeft2: In the first 100 ns, I assign x with “0000000000000001” in hexadecimal, and the number being shifts left into “0000000000000004”. In the second 200 ns, I assign x with “000000000000000F” in hexadecimal, and the number being shifts left into “000000000000003C”.



```
/tb_shiftleft2/tb_x   64'h00000000...   0000000000000001   000000000000000F
/tb_shiftleft2/tb_y   64'h00000000...   0000000000000004   000000000000003C
```

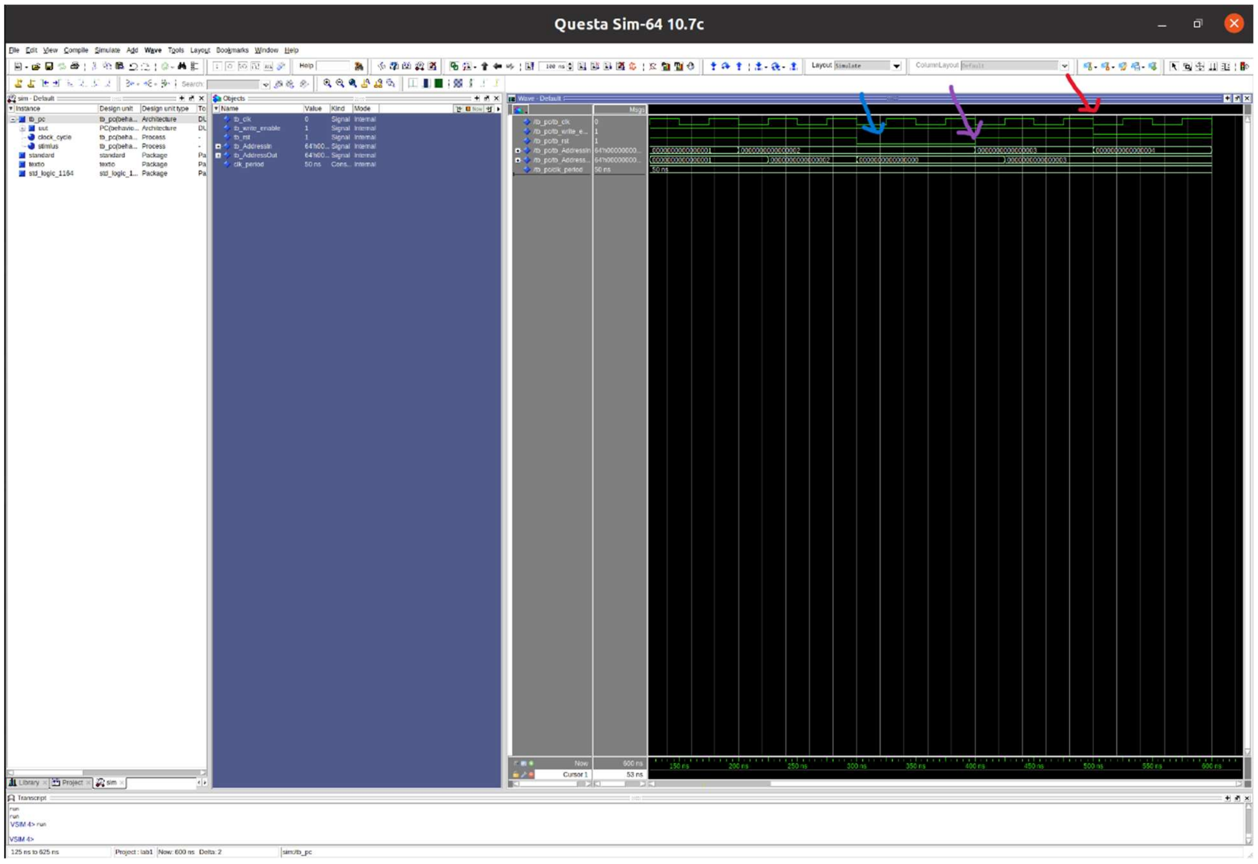
```
sim:/tb_shiftleft2/tb_y
VSIM 6> run
VSIM 7> run

VSIM 7>
```

Project : lab1   Now: 200 ns Delta: 3   sim:/tb_shiftleft2

f. Program Counter: This is a program counter which increment by one every 100 ns. Before the blue markers, I set rst and write_enable to an active-high state, which allow the components to write the address. The initial address is “0000000000000000” in hexadecimal and count up to “0000000000000002”. At the blue marker, when rst is pulled low, AddressOut being set to “0000000000000000”. At the purple marker, rst returns to its active-high state, and AddressOut receives the next address from AddressIn which is “0000000000000003”. At the red marker, when write_enable being pulled low, AddressIn has been incremented to

“0000000000000004”, but the value of AddressOut does not change.



Questa Sim-64 10.7c

sim - Default

Instance	Design unit	Design unit type	To
tb_pc	tb_pc(beha...	Architecture	DU
uut	PC(behavio...	Architecture	DU
clock_cycle	tb_pc(beha...	Process	-
stimulus	tb_pc(beha...	Process	-
standard	standard	Package	Pa
textio	textio	Package	Pa
std_logic_1164	std_logic_1...	Package	Pa

Objects

Name	Value	Kind	Mode
tb_clk	0	Signal	Internal
tb_write_enable	1	Signal	Internal
tb_rst	1	Signal	Internal
tb_AddressIn	64'h00...	Signal	Internal
tb_AddressOut	64'h00...	Signal	Internal
clk_period	50 ns	Cons...	Internal

Wave - Default

	Msgs	
/tb_pc/tb_clk	0	
/tb_pc/tb_write_e...	1	
/tb_pc/tb_rst	1	
/tb_pc/tb_AddressIn	64'h00000000...	0000000000000001 · 0000000000000002 · 0000000000000003 · 0000000000000004
/tb_pc/tb_Address...	64'h00000000...	0000000000000001 · 0000000000000002 · 0000000000000000 · 0000000000000003
/tb_pc/clk_period	50 ns	50 ns

Now 600 ns
Cursor 1 53 ns

Transcript

```
run
run
VSIM 4> run

VSIM 4>
```

125 ns to 625 ns | Project : lab1 | Now: 600 ns Delta: 2 | sim:/tb_pc