Report for lab1 assignment

1. A brief description (a sentence or two) of what each component does.
   AND2: Perform AND logic operation on two input signals.
   MUX5: Multiplexer for two 5-bit input signals.
   MUX64: Multiplexer for two 64-bit input signals
   SignExtend: Spread the input signal, for example, from 32 bits to 64 bits.
   ShiftLeft2: Perform logical left shift operation on 64 bit input signal, and make up 0 after left shift.
   Program Counter: A register that holds the address of the current instruction.

2. A brief explanation of your choices for modeling type.
   Behavioral:
   We describe the behavior of an entity using sequential statements, this makes it very similar to high-level programming languages in syntax and semantics. Behavioral modeling describes how the output is derived from the inputs using structured statements. This is what we want to use in lab1.

3. Waveform with brief description for all entities.

(1) AND2： From figure 3.1, we can see four parts with different input signals executing AND logic operation.

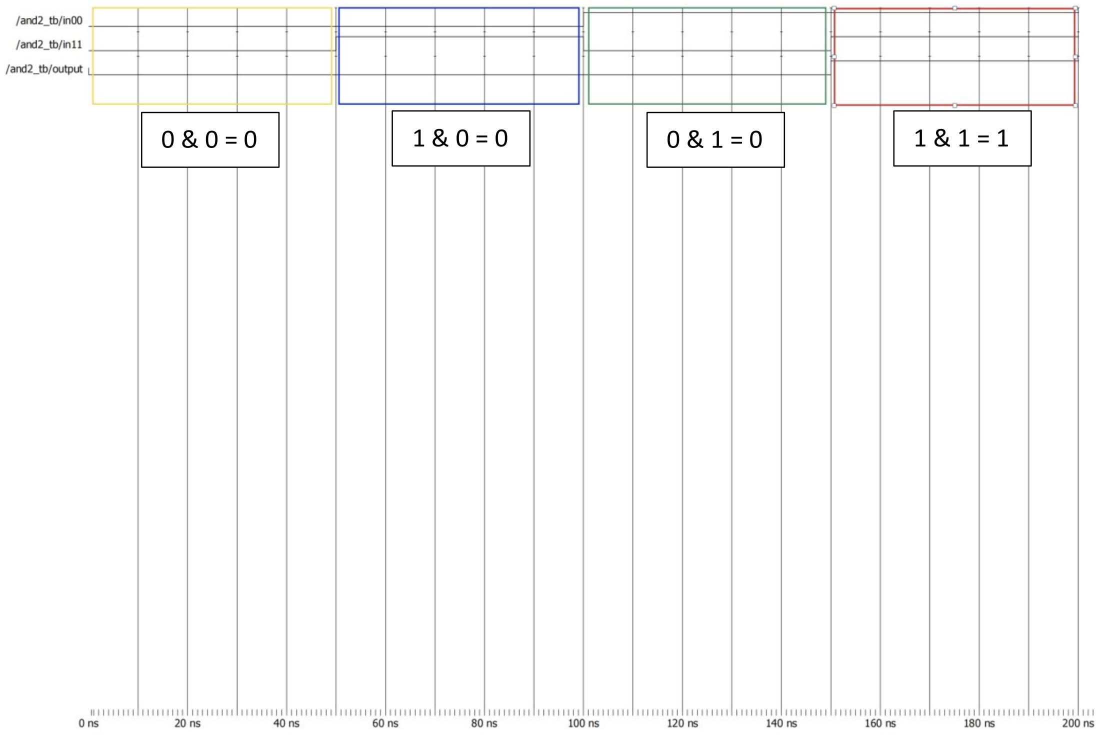

Figure 3.1 Wave of AND2 testbench

(2) MUX5: This is a 5-bit multiplexer, the entity contains two 5-bit input signals, a selection signal, and a 5-bit output signal. When sel is 0, select in0; when sel is 1, select in1. From figure 3.2, For a pair of input signals that are different each time, it is obvious that each selection is correct.

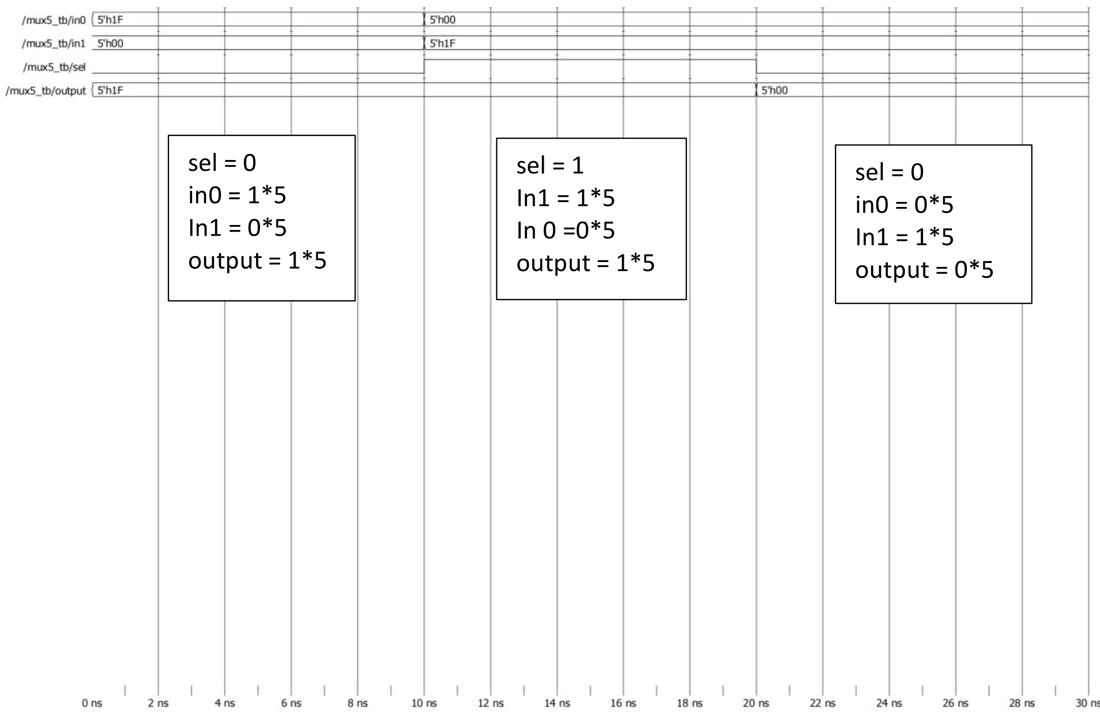

Figure 3.2 Wave of MUX5 testbench

(3) MUX64: A 64-bit multiplexer, the entity contains two 64-bit input signals, a selection signal, and a 64-bit output signal. When sel is 0, select in0; when sel is 1, select in1. Through three pairs of signals, we can clearly see the realization of the selection function.

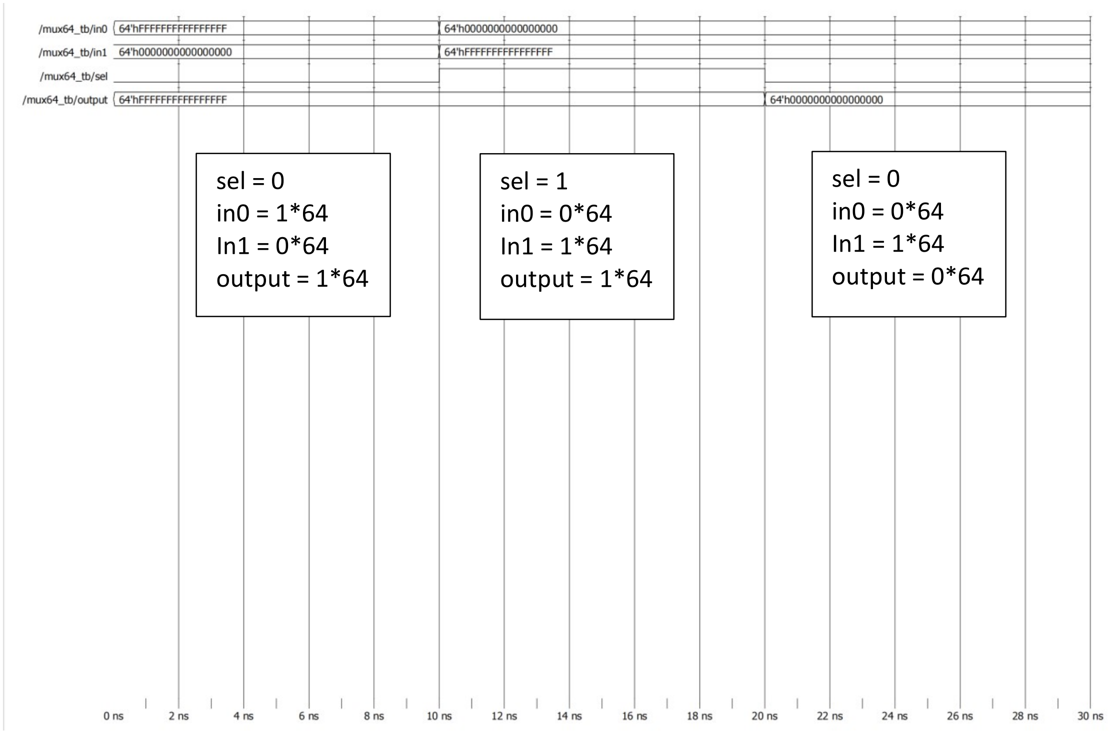

Figure 3.3 Wave of MUX64 testbench

(4) ShiftLeft2: A signal logic shifter. Move the input 64 bit signal two bits to the left, and make up 0 in the following empty bits. The entity includes a 64 bit input signal and an output signal shifted two bits to the left. In the architecture part, I used SLL, shift left logical operator for shift operation. We can see the correct results after shift left 2 bit from figure 3.4. The last 4 bit 0001(1)->0100(4), 0010(2)->1000(8)

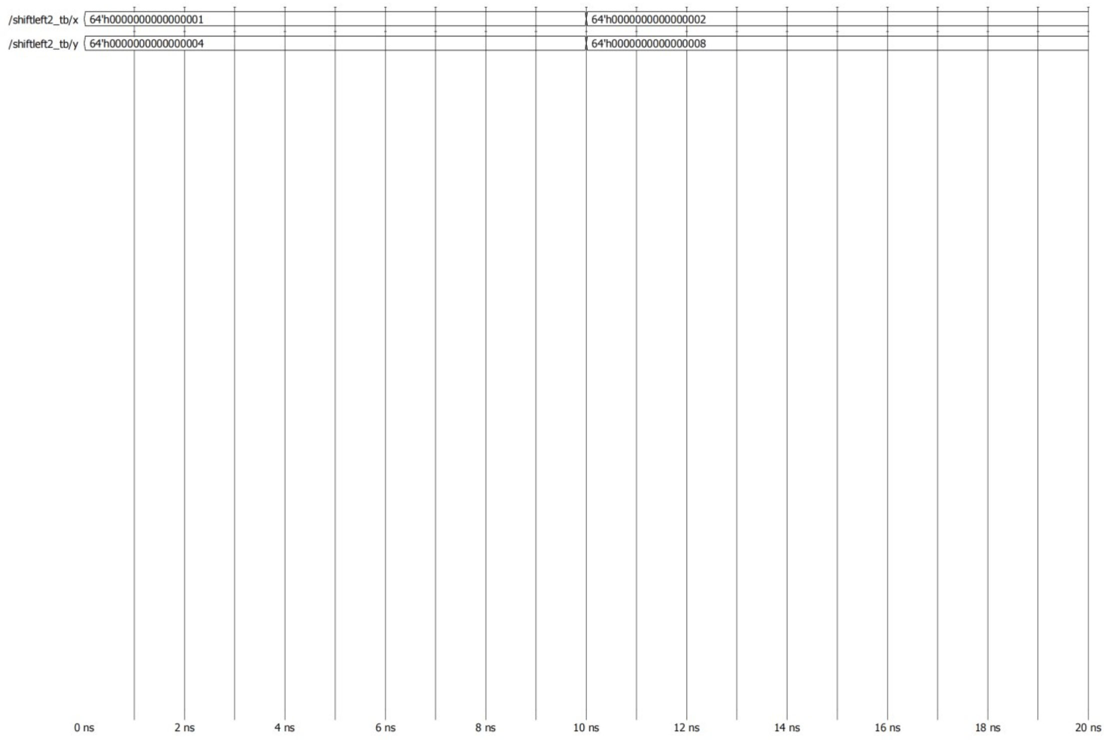

Figure 3.4 Wave of ShiftLeft2 testbench

(5) SignExtend: The entity contains a 32-bit input signal and expands it to a 64 bit signal output. Use the resize function to directly expand the input signal to a 64 bit signal. When the first bit is 0, the first 32 bits will be 0, when the first bit is 1, the first 32 bits will be 1.

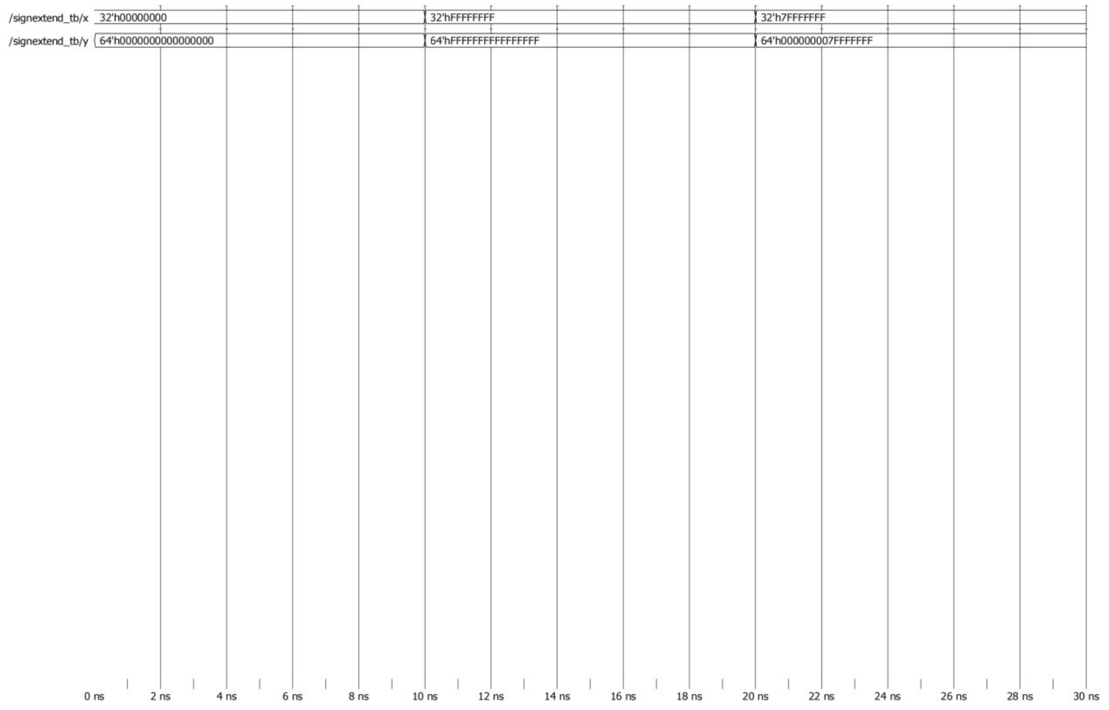

Figure 3.5 Wave of SignExtend testbench

(6) PC: An instruction address register, the entity includes an clock signal, a reset signal, a write signal, and two instruction address signals. Since asynchronous reset is required, there is no need to wait for the clock signal. I use “if” statement to set the address to clear when the reset signal is **1**. When the reset signal is not **1**, the address can be output only when the writing signal is 1 at the same time as the rising edge signal. I take 10ns as a clock cycle, and use “now” to terminate the clock signal. When the time exceeds 100ns, the clock signal will stop and will not cycle indefinitely. For the blue rectangle, since the reset signal is **1**, even if the write signal is **1**, the signal will not be output at the rising edge. For green rectangle, the reset signal is **0**, write signal = **1**, at the rising edge, the signal is successfully output. For the red part, even at the rising edge, if the writing signal is **0**, the signal will not be written, and the last output signal will still be maintained. For the last yellow part, when the reset signal =**1**, the AddressOut be set 0x0.

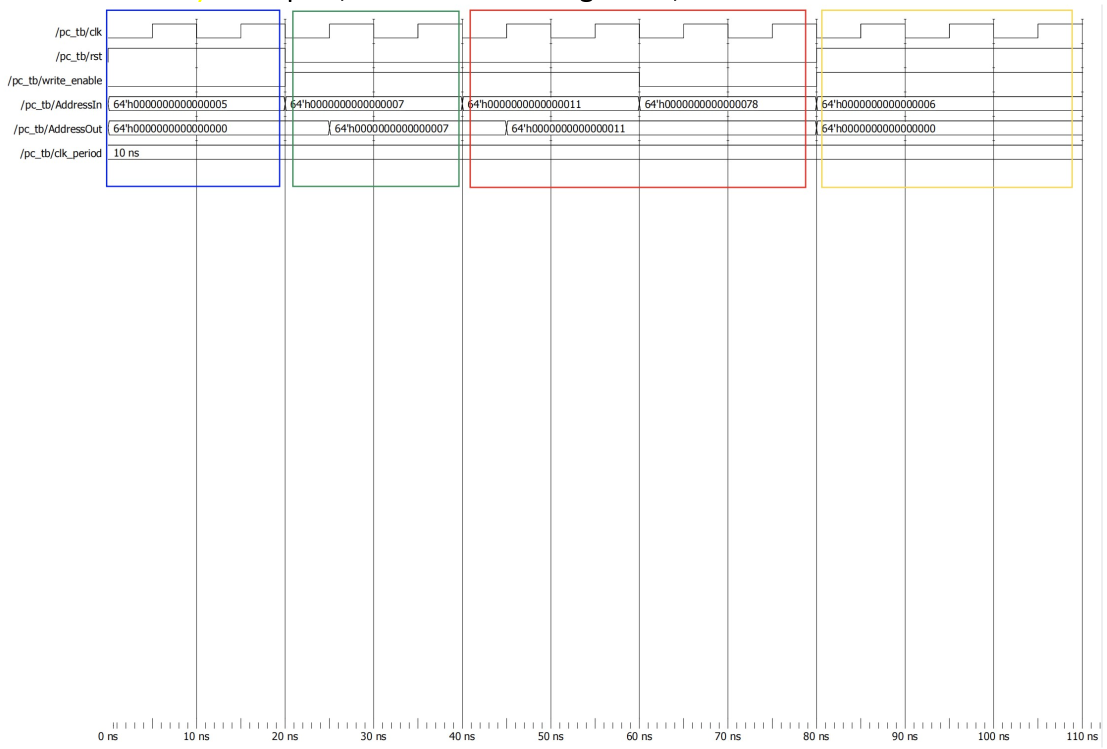

Figure 3.6 Wave of PC testbench