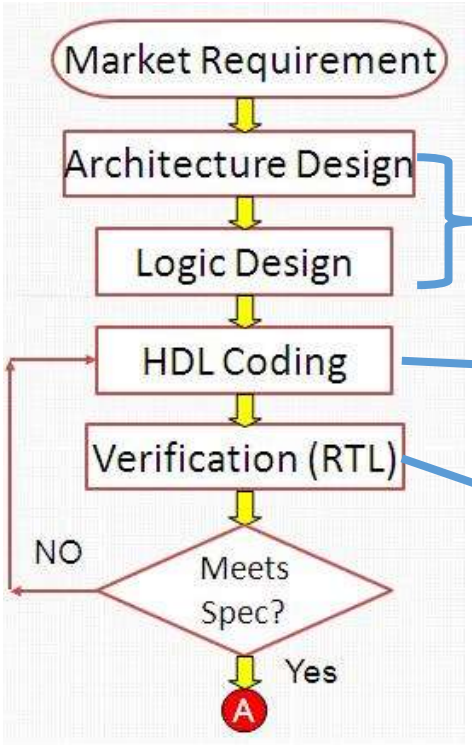
Modelsim Tutorial2

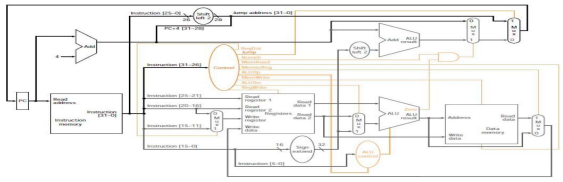
ICSL

Seok-jin Eo
TA of Computer Architecture
Department of Computer Engineering
Kyung Hee University, Korea

Design Flow of Digital Systems



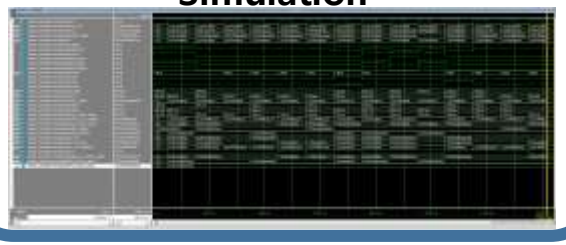
MIPS structure



Verilog coding

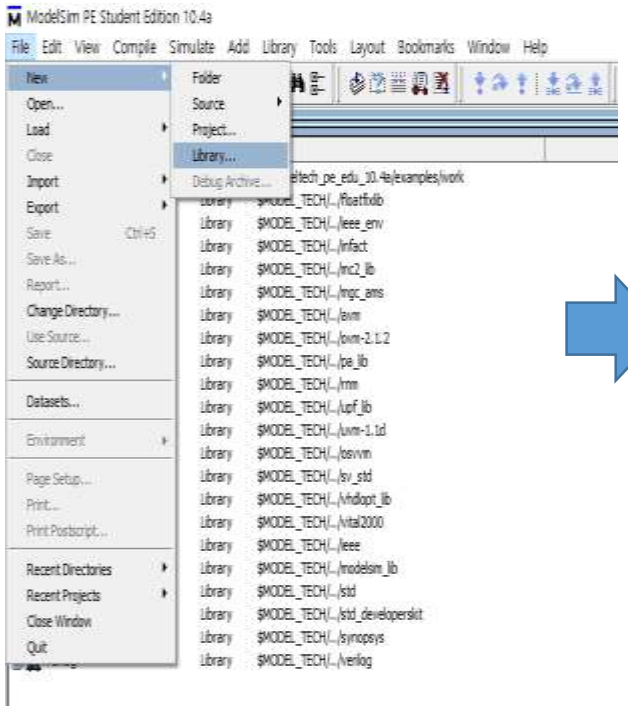
```
Ex) module adder (A,B, S,C);
                input A,B;
                output S,C;
                assign S = A^B;
                assign C = A&B;
    endmodule
```

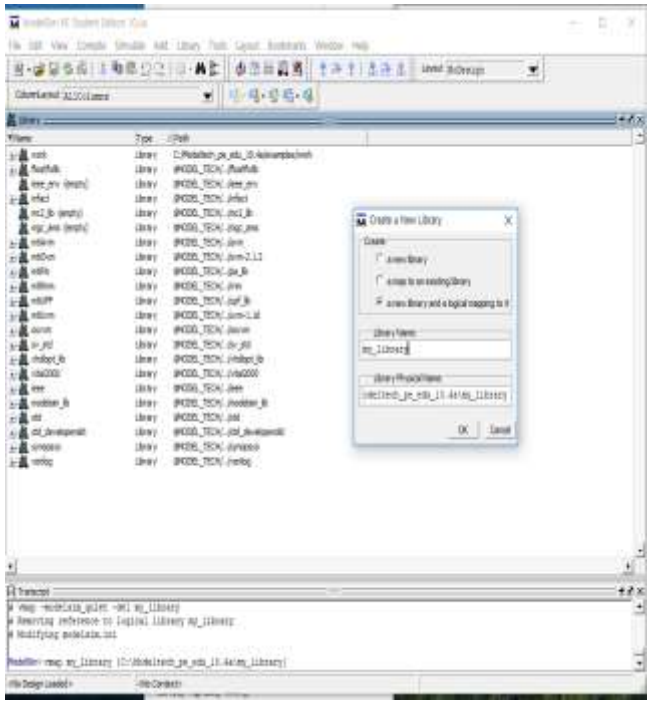
Simulation



How to Make Simulation to Obtain Waveform

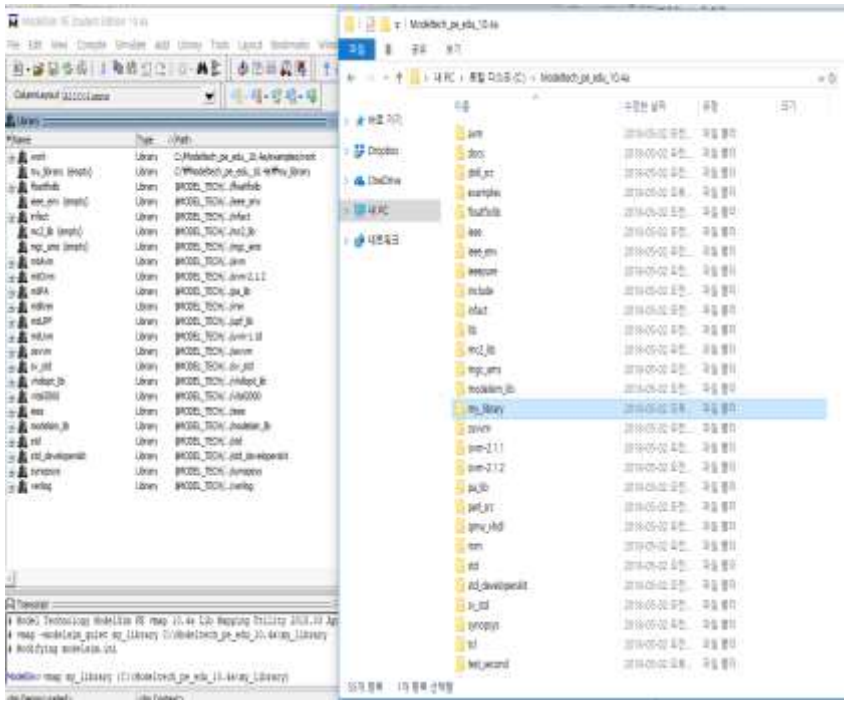
- 1. Make library (not necessary)





Library Physical Name:

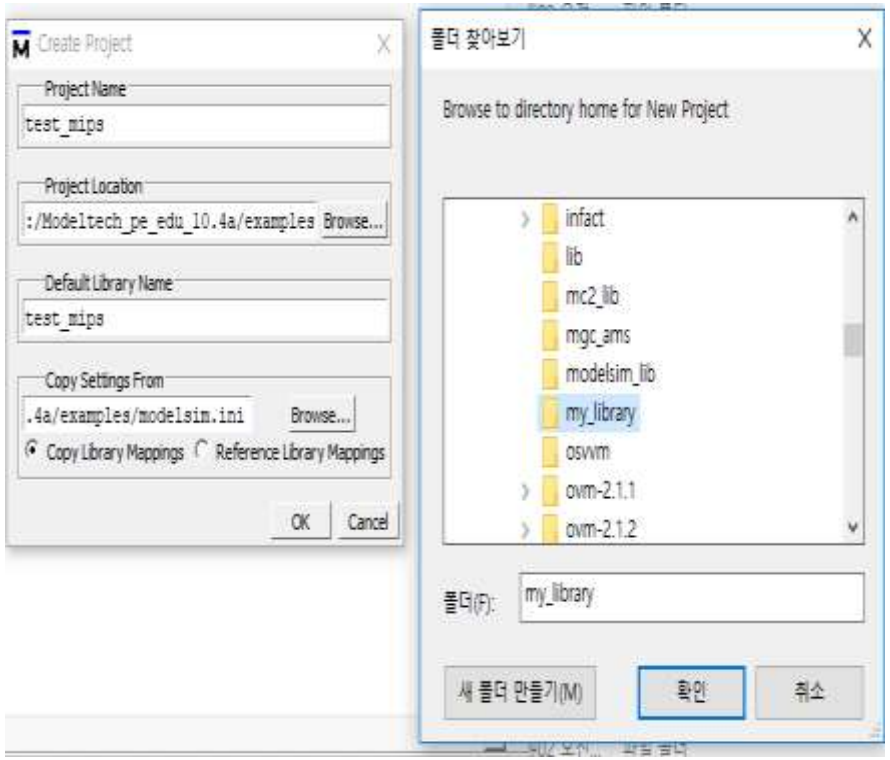
=> C:₩Modeltech_pe_edu_10.4a₩my_library

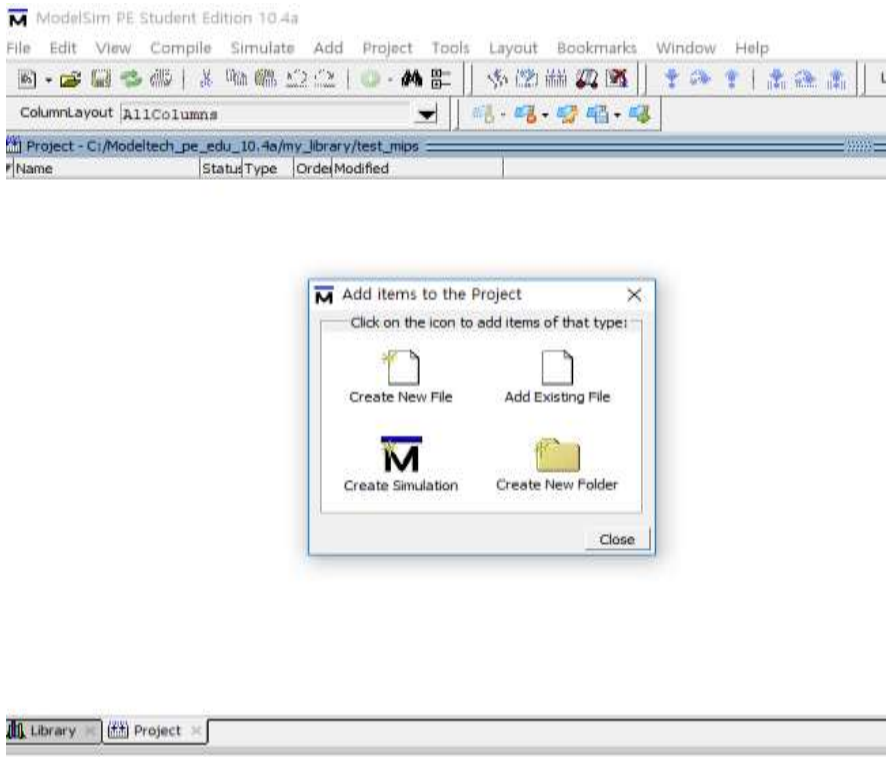


(Result)

How to Make Simulation to Obtain Waveform

- 2. Make project

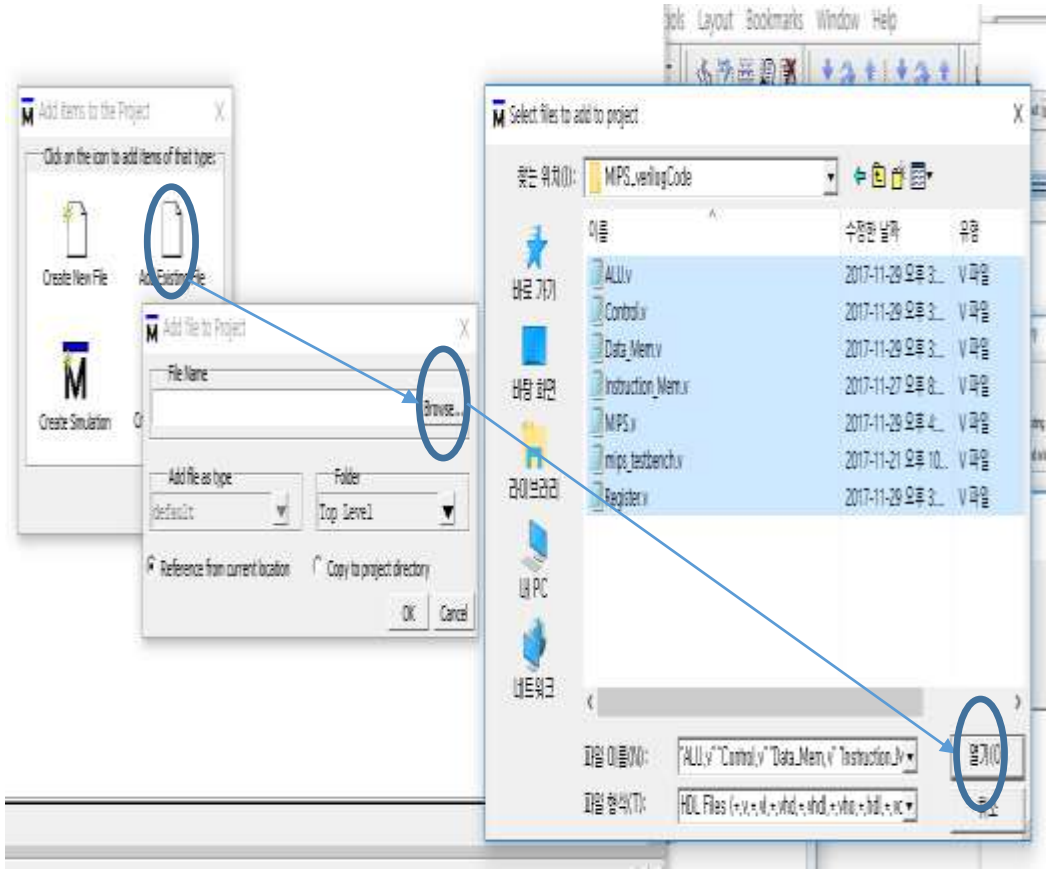


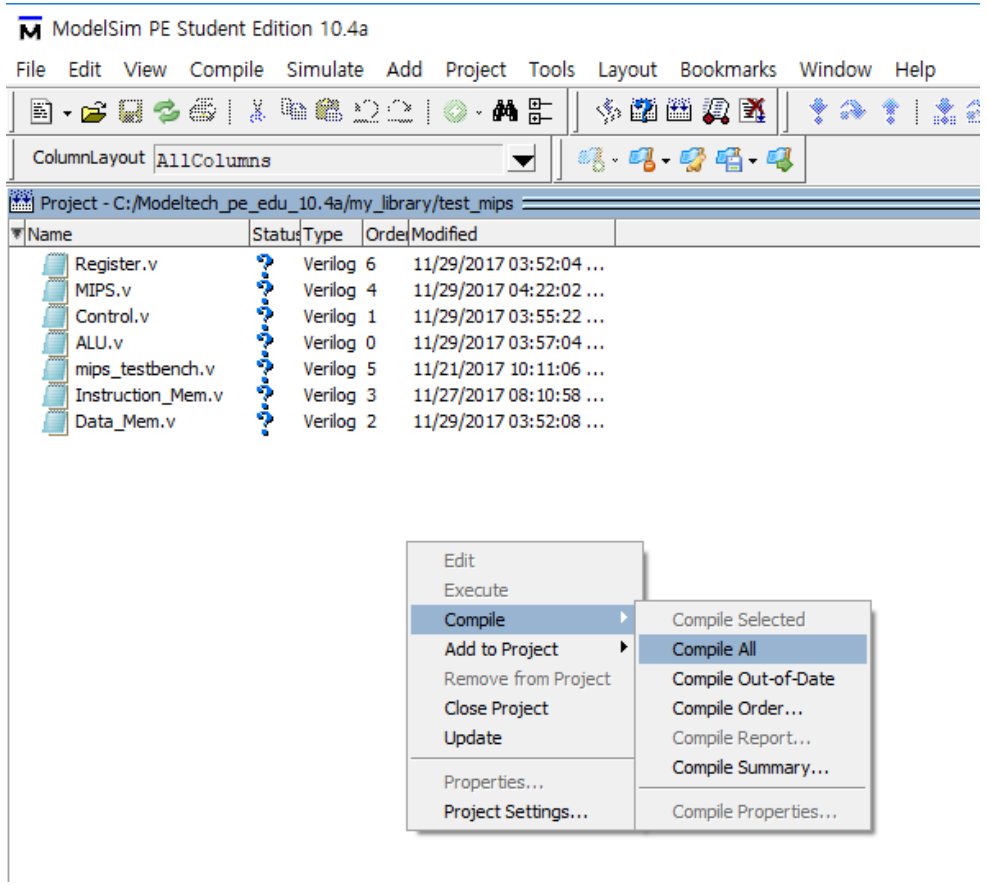


(Result)

How to Make Simulation to Obtain Waveform

- 3. Add file

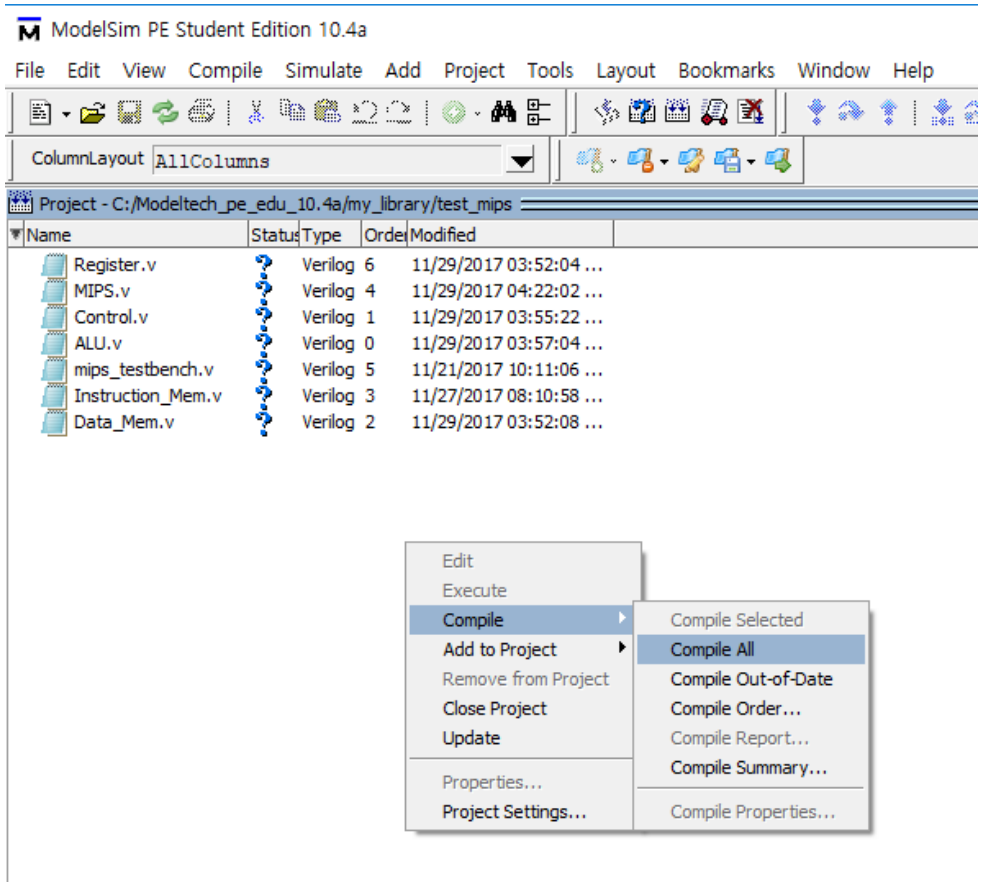


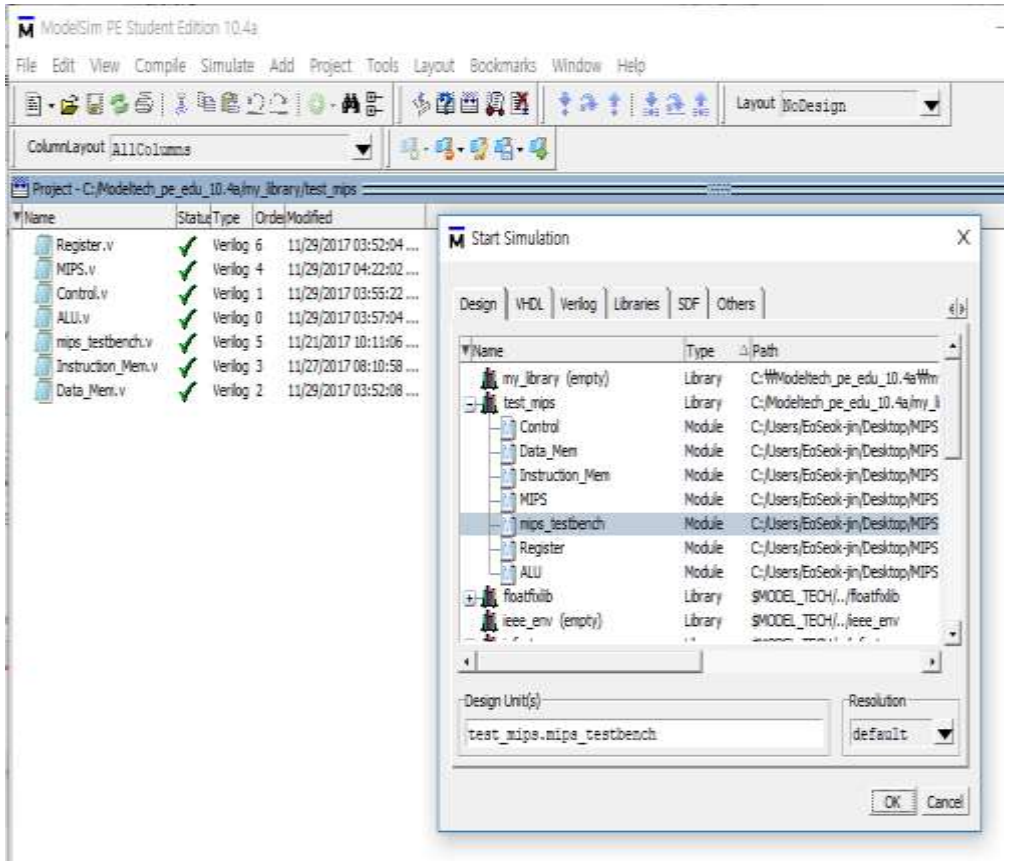


(Result)

How to Make Simulation to Obtain Waveform

- 4. Compile

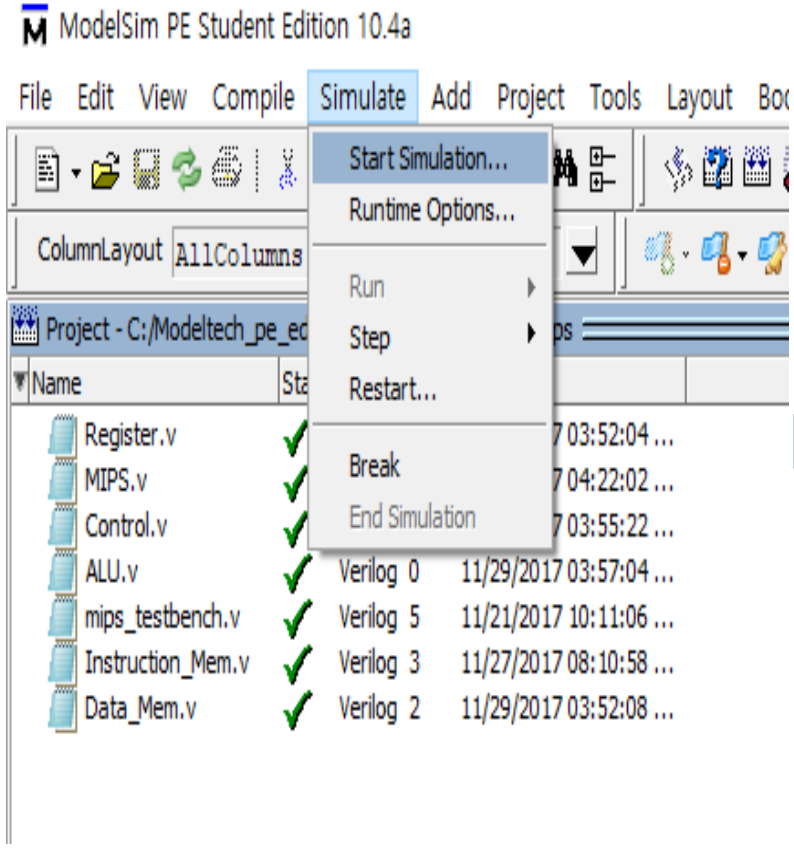


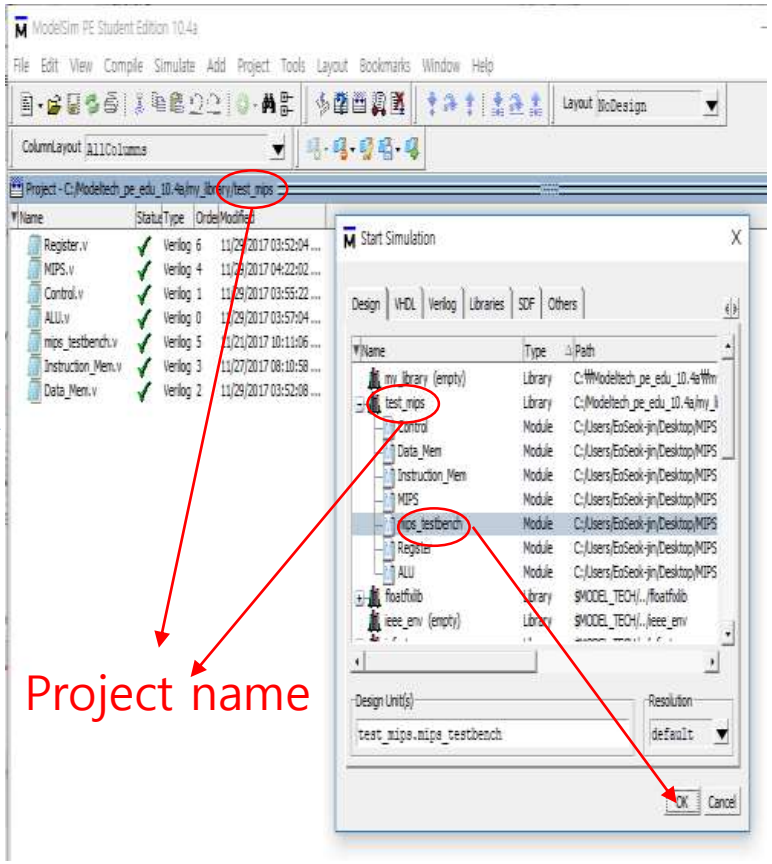


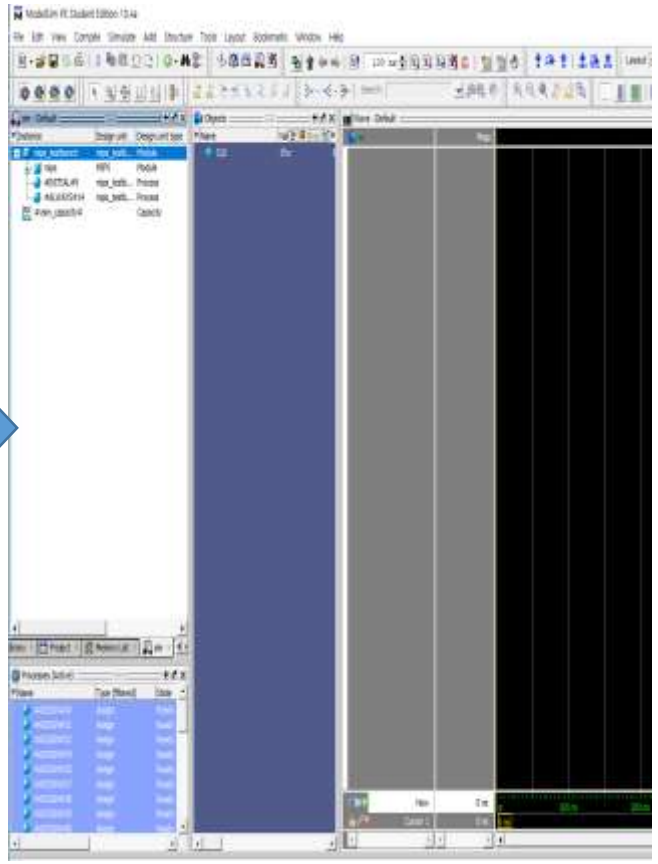
(Result)

How to Make Simulation to Obtain Waveform

- 5. Start simulation



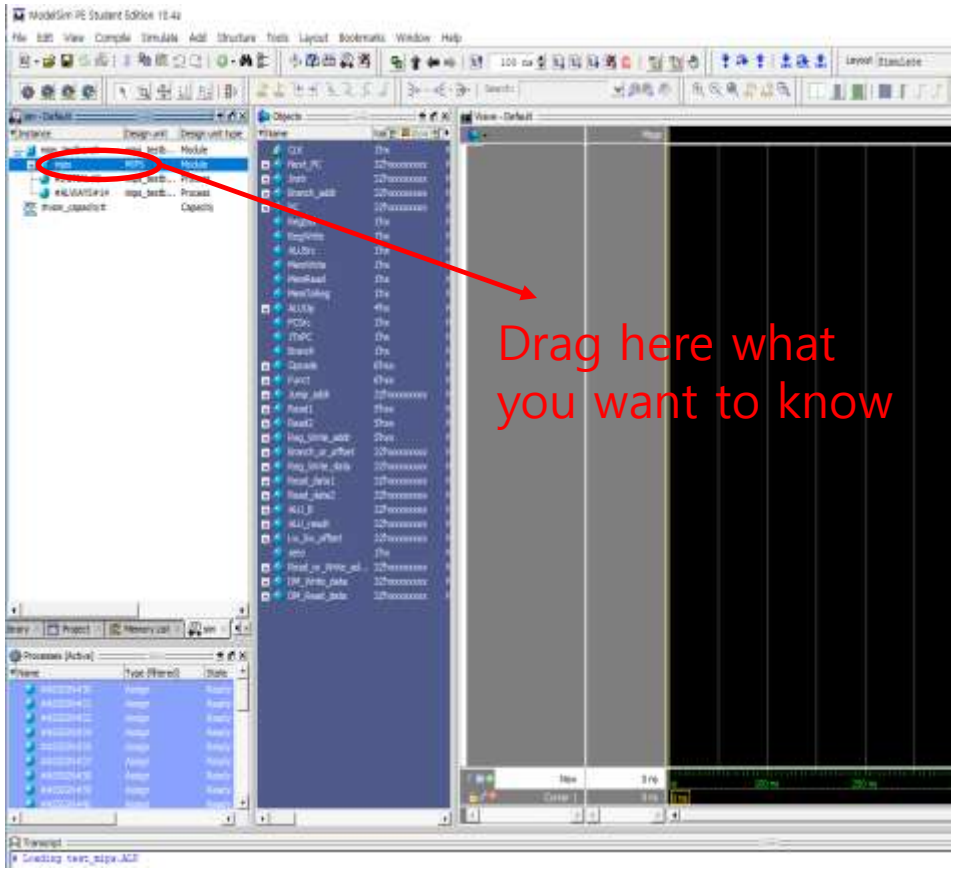


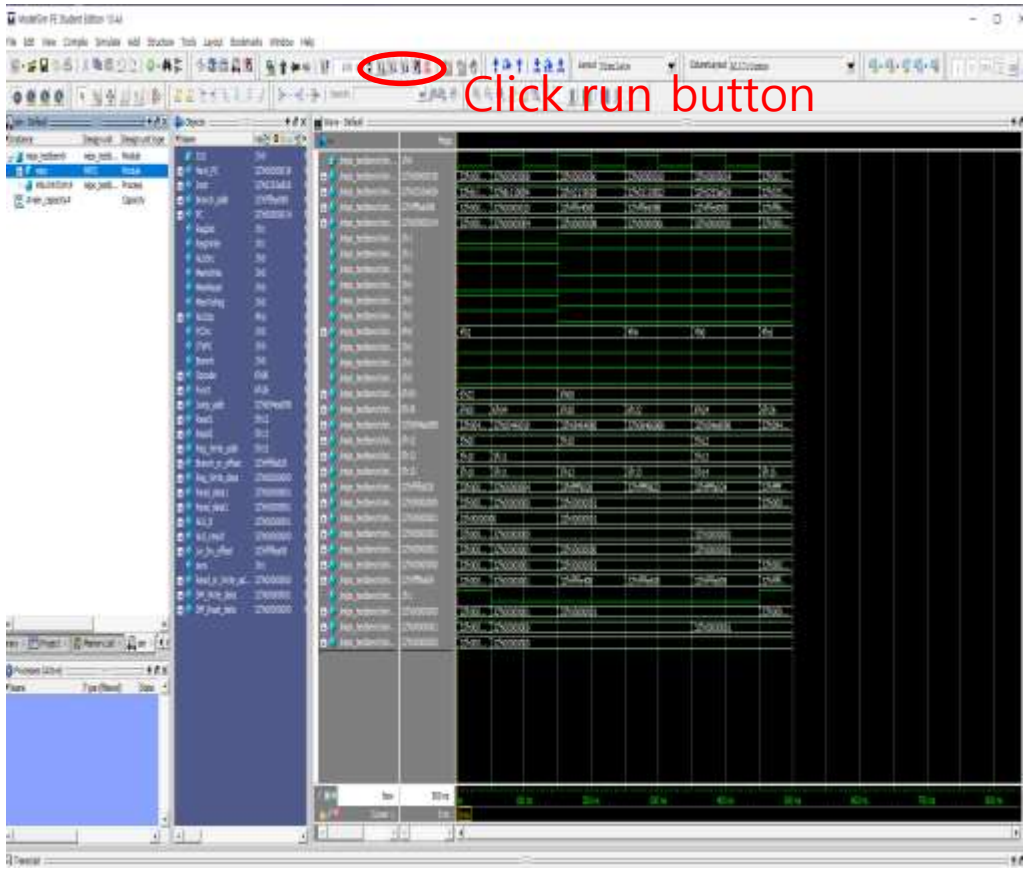


(Result)

How to Make Simulation to Obtain Waveform

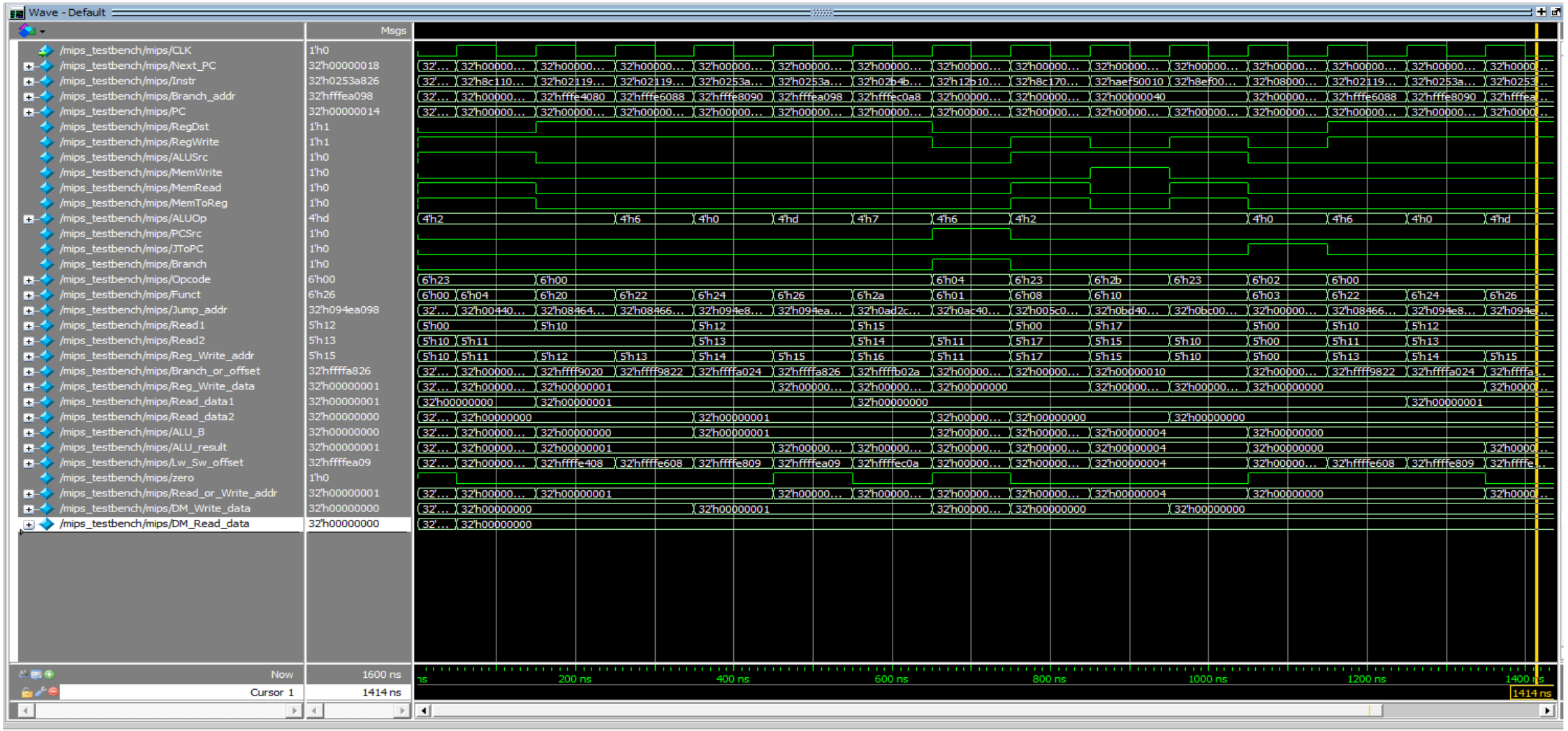
- 6. Drag & Run



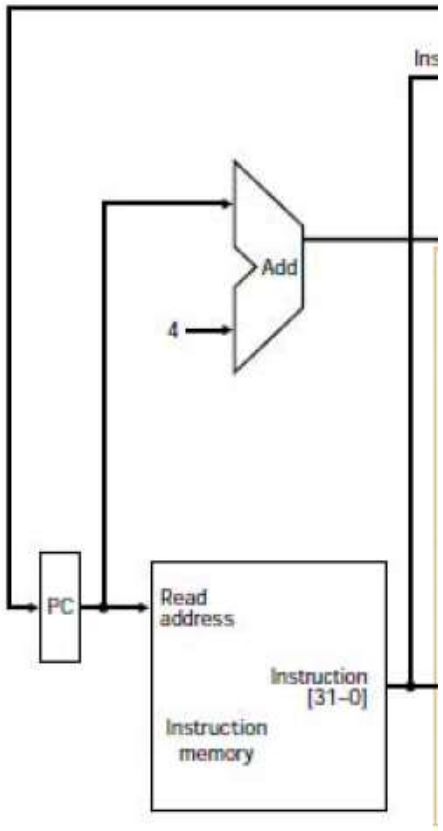


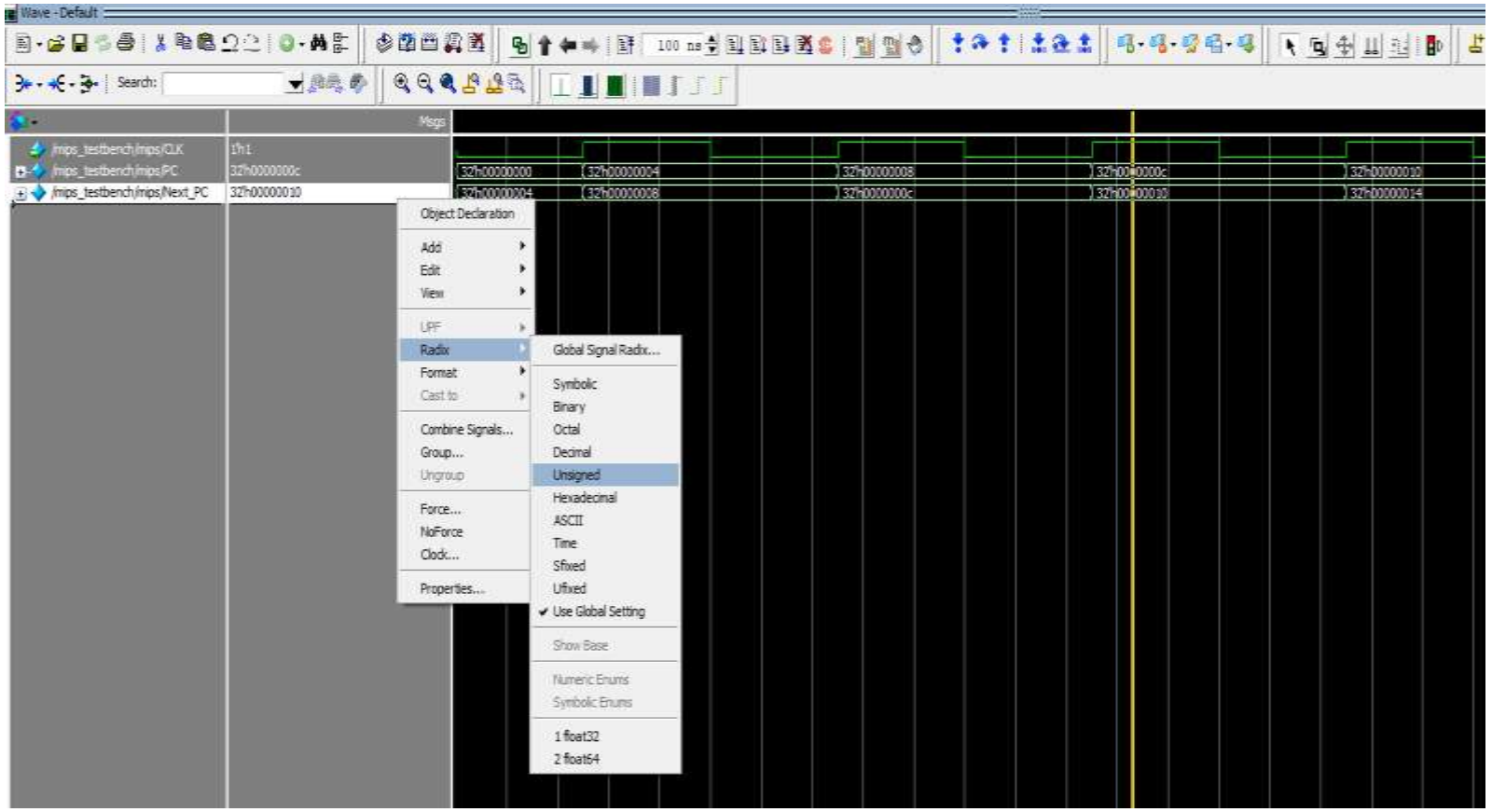
(Result)

How do extract meaning from complex waveform?



First of all, understand data-updating as a clk timing (with radix-modifying)



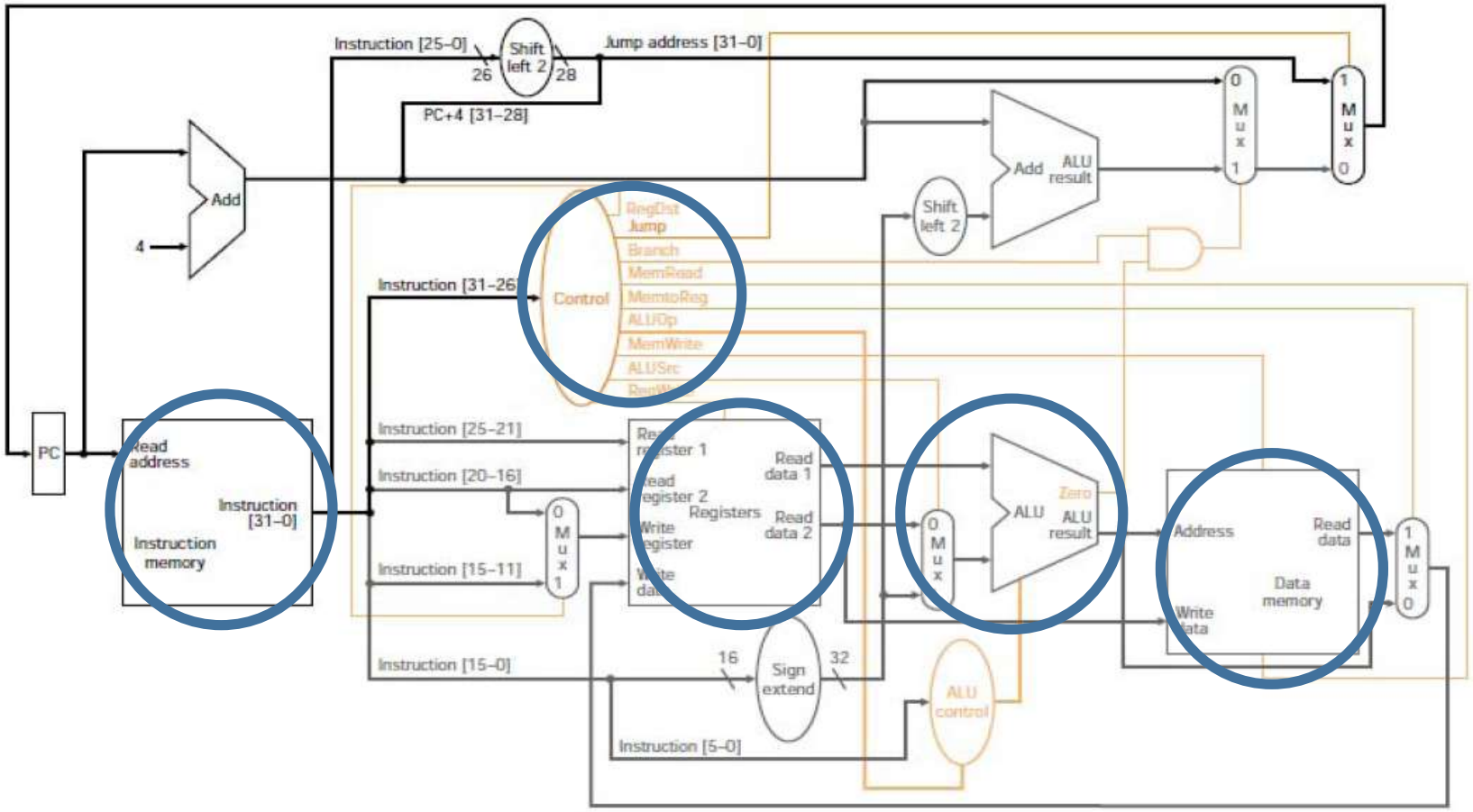


Next, understand meaning of instruction as a machine language.

Test Assembly Code

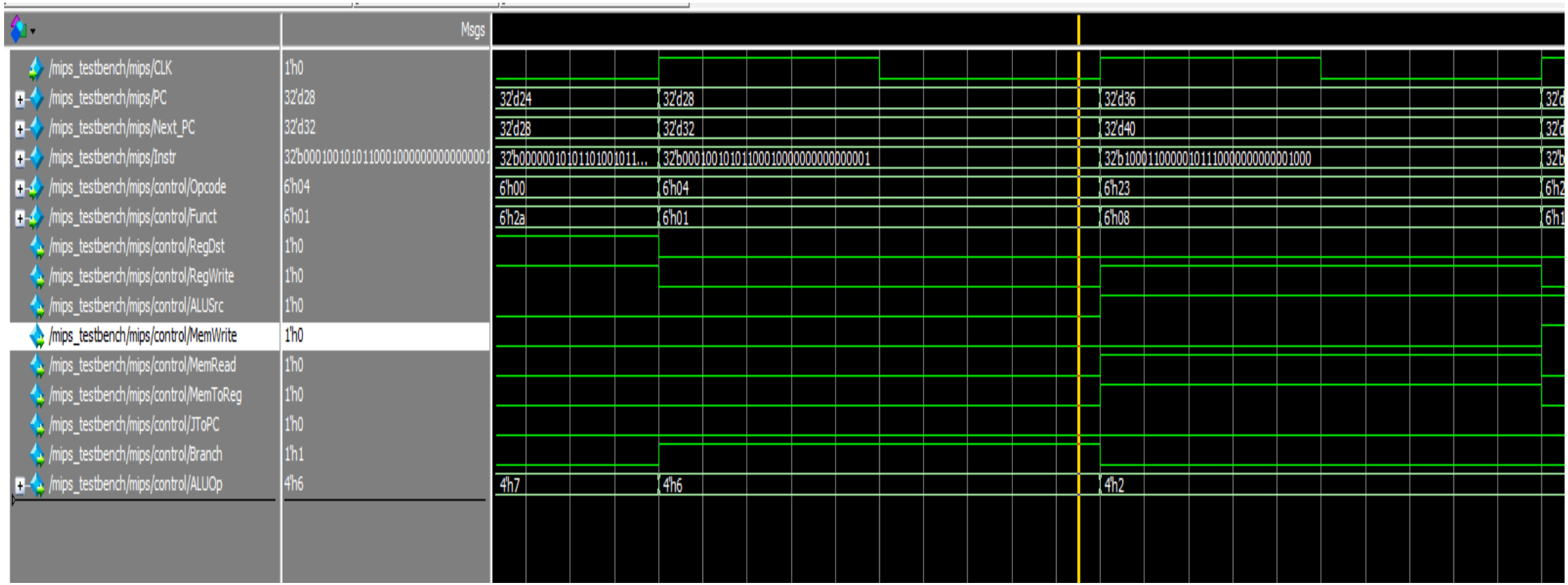
```
mem[0] = 32'b100011_00000_10000_00000_00000_000000; // lw s0 0($zero) , s0 = 1
mem[1] = 32'b100011_00000_10001_00000_00000_000100; // lw s1 4($zero) , s1 = 0
mem[2] = 32'b000000_10000_10001_10010_00000_100000; // add s2 s0 s1 , s2 = 1
mem[3] = 32'b000000_10000_10001_10011_00000_100010; // sub s3 s0 s1 , s3 = 1
mem[4] = 32'b000000_10010_10011_10100_00000_100100; // and s4 s2 s3 , s4 = 1
mem[5] = 32'b000000_10010_10011_10101_00000_100110; // xor s5 s2 s3 , s5 = 0
mem[6] = 32'b000000_10101_10100_10110_00000_101010; // slt s6 s5 s4 , s6 = 1
mem[7] = 32'b000100_10101_10001_00000_00000_000001; // beq s5 s1 1 , s5 = s1 = 0
mem[8] = 32'b000010_00000_00000_00000_00000_001101;
// j 13 , it is not executed at first, but is executed after the jump. (mem[12])
mem[9] = 32'b100011_00000_10111_00000_00000_001000; // lw s7 8($zero) , s7 = 0
mem[10]= 32'b101011_10111_10101_00000_00000_010000; // sw s5 16(s7) , mem[4] = 0
mem[11]= 32'b100011_10111_10000_00000_00000_010000; // lw s0 16(s7) , s0 = mem[4] = 0
mem[12]= 32'b000010_00000_00000_00000_00000_000011; // j 3
mem[13]= 32'b000000_10000_00000_10111_00000_100111;
// nor s7 s0 $zero , s7 = 32'b1111_1111_1111_1111_1111_1111_1111_1111;
mem[14]= 32'b100011_00000_01000_00000_00000_101000;
// lw t0 40($zero) , t0 = 32'b0101_0101_1010_1010_0101_0101_1010_1010;
```

And then, analyze by each parts

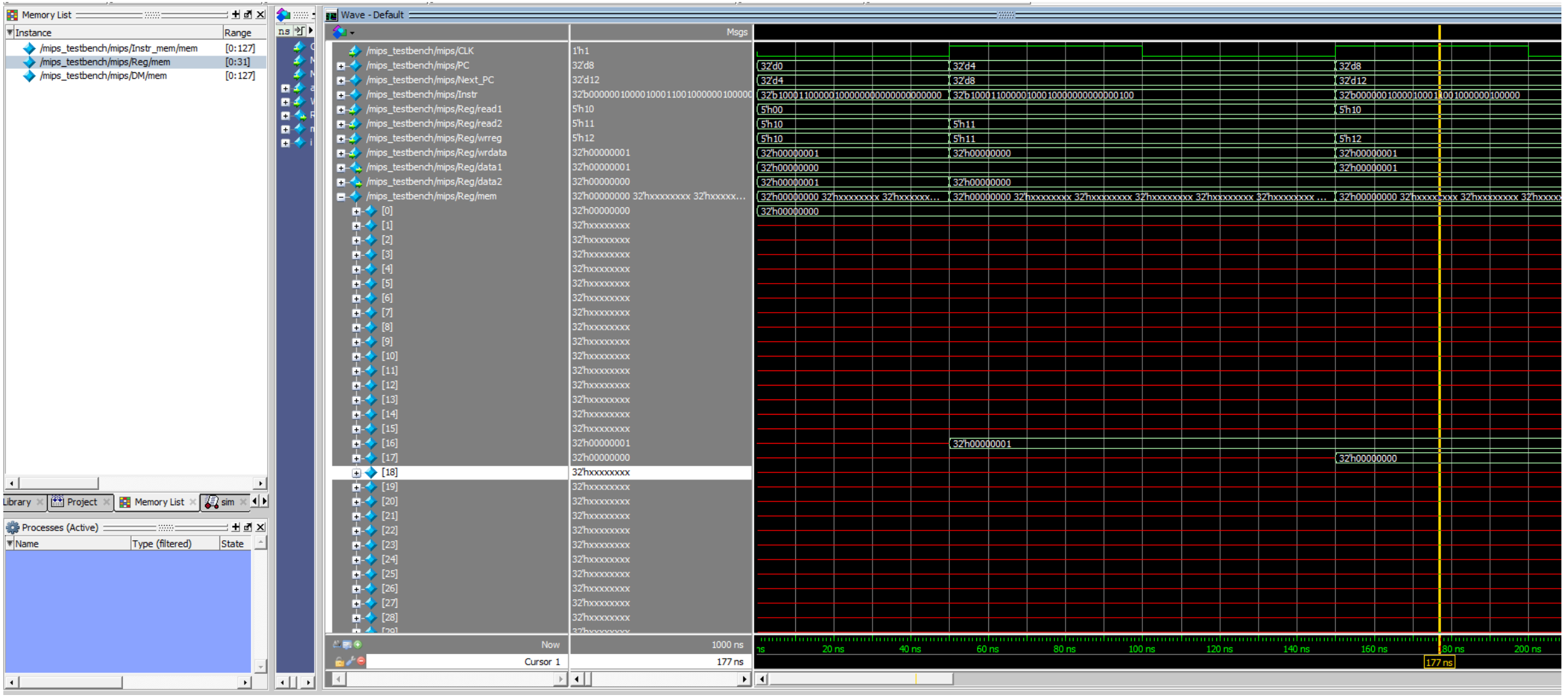


Control signal table

Operation	RegDst	RegWrite	ALUSrc	ALUOp	MemWrite	MemRead	MemToReg	JToPC
beq	X	0	0	110	0	0	X	0



How to check memory elements in submodule



Q & A