Gebze Technical University Computer Engineering

CSE 331

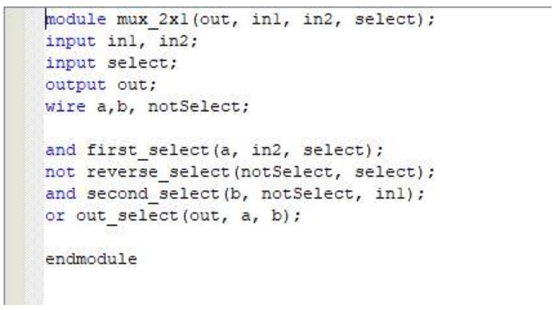
ASSIGNMENT 3 REPORT

ERSİN ALÇİN

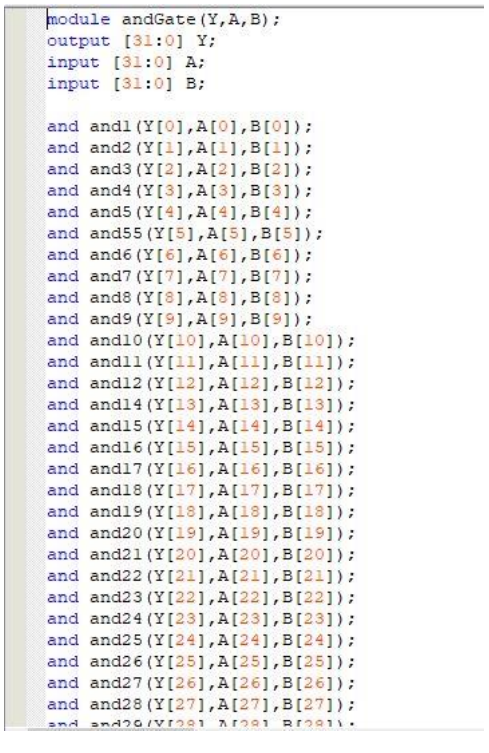
1801042692

Making of ALU To achieve an ALU which can select an operation in a list of 8 I used 8x1 mux to select which operation to return. Before that I calculated all outcomes and directed them to 8x1 mux. ALU Modules (Explained) Mux To assemble a 8x1 mux I needed 2 4x1 mux and 1 2x1 mux. And to construct a 4x1 mux I needed 3 more 2x1 mux. So, I started this assignment by building a 2x1 mux in Verilog. After first successful 2x1 mux I needed 32 bit input and output version of this mux to create 32 bit 4x1 and 8x1 muxes. In the way I used 32 times 2x1 muxes. . Simple Gates After successful 1 bit and 32-bit 2x1 mux, I wrote 4x1 mux in bot 1 bit and 32 bits. After them finally wrote the 32-bit 8x1 mux to create 32-bit ALU.

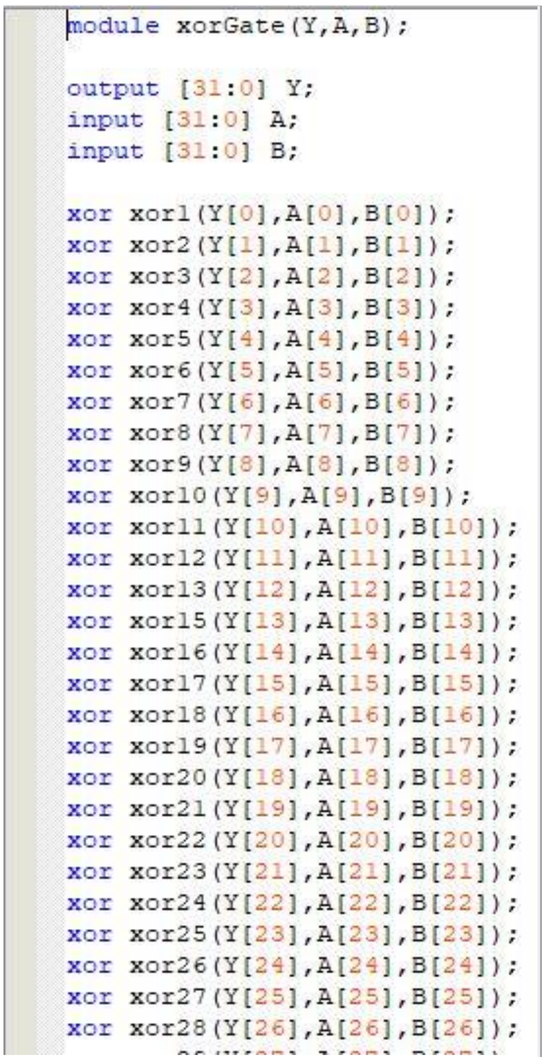
Then I wrote simple Gates and full_adder_32, subtracter_32 .



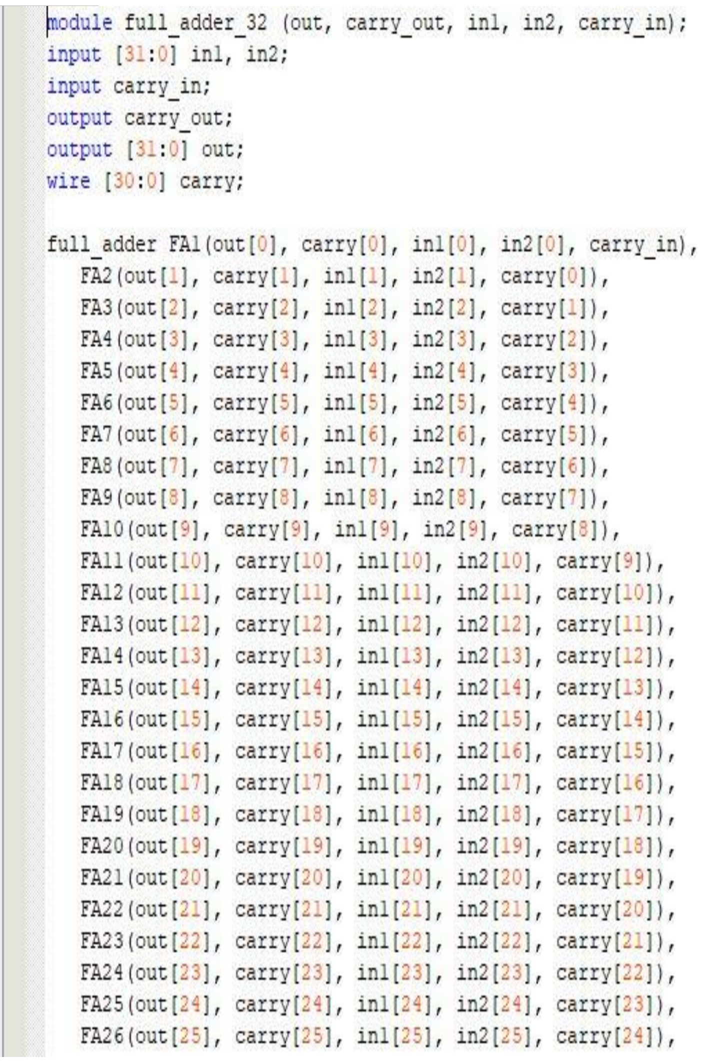
```
module mux_2x1(out, in1, in2, select);
input in1, in2;
input select;
output out;
wire a,b, notSelect;

and first_select(a, in2, select);
not reverse_select(notSelect, select);
and second_select(b, notSelect, in1);
or out_select(out, a, b);

endmodule
```



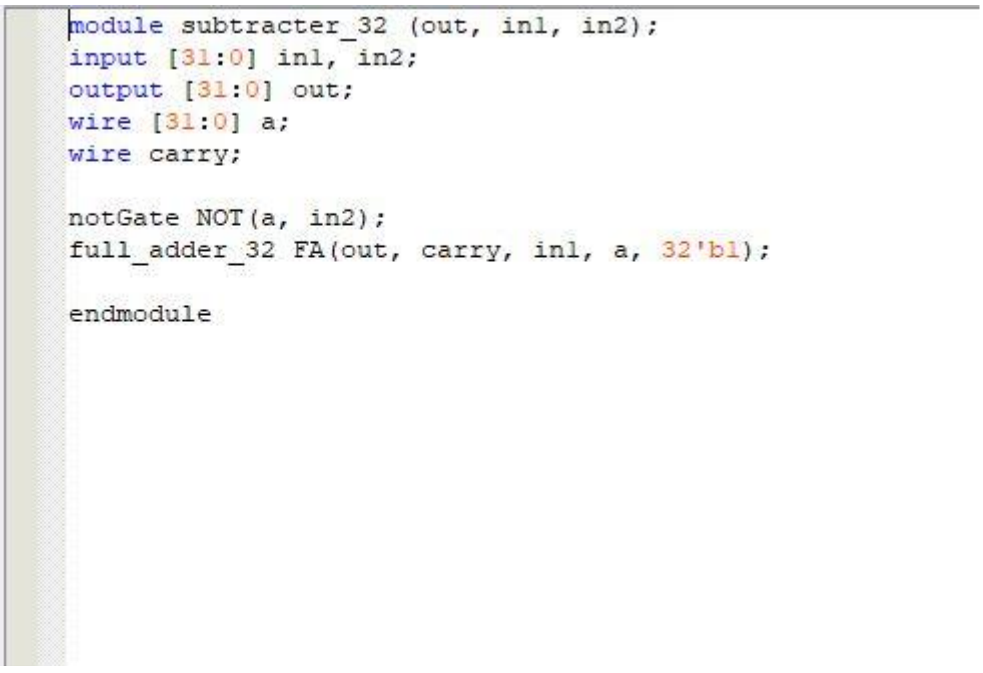
```
module andGate(Y,A,B);
output [31:0] Y;
input [31:0] A;
input [31:0] B;

and and1(Y[0],A[0],B[0]);
and and2(Y[1],A[1],B[1]);
and and3(Y[2],A[2],B[2]);
and and4(Y[3],A[3],B[3]);
and and5(Y[4],A[4],B[4]);
and and55(Y[5],A[5],B[5]);
and and6(Y[6],A[6],B[6]);
and and7(Y[7],A[7],B[7]);
and and8(Y[8],A[8],B[8]);
and and9(Y[9],A[9],B[9]);
and and10(Y[10],A[10],B[10]);
and and11(Y[11],A[11],B[11]);
and and12(Y[12],A[12],B[12]);
and and14(Y[13],A[13],B[13]);
and and15(Y[14],A[14],B[14]);
and and16(Y[15],A[15],B[15]);
and and17(Y[16],A[16],B[16]);
and and18(Y[17],A[17],B[17]);
and and19(Y[18],A[18],B[18]);
and and20(Y[19],A[19],B[19]);
and and21(Y[20],A[20],B[20]);
and and22(Y[21],A[21],B[21]);
and and23(Y[22],A[22],B[22]);
and and24(Y[23],A[23],B[23]);
and and25(Y[24],A[24],B[24]);
and and26(Y[25],A[25],B[25]);
and and27(Y[26],A[26],B[26]);
and and28(Y[27],A[27],B[27]);
and and29(Y[28],A[28],B[28]);
```



```
module xorGate(Y,A,B);

output  [31:0] Y;
input  [31:0] A;
input  [31:0] B;

xor xor1(Y[0],A[0],B[0]);
xor xor2(Y[1],A[1],B[1]);
xor xor3(Y[2],A[2],B[2]);
xor xor4(Y[3],A[3],B[3]);
xor xor5(Y[4],A[4],B[4]);
xor xor6(Y[5],A[5],B[5]);
xor xor7(Y[6],A[6],B[6]);
xor xor8(Y[7],A[7],B[7]);
xor xor9(Y[8],A[8],B[8]);
xor xor10(Y[9],A[9],B[9]);
xor xor11(Y[10],A[10],B[10]);
xor xor12(Y[11],A[11],B[11]);
xor xor13(Y[12],A[12],B[12]);
xor xor15(Y[13],A[13],B[13]);
xor xor16(Y[14],A[14],B[14]);
xor xor17(Y[15],A[15],B[15]);
xor xor18(Y[16],A[16],B[16]);
xor xor19(Y[17],A[17],B[17]);
xor xor20(Y[18],A[18],B[18]);
xor xor21(Y[19],A[19],B[19]);
xor xor22(Y[20],A[20],B[20]);
xor xor23(Y[21],A[21],B[21]);
xor xor24(Y[22],A[22],B[22]);
xor xor25(Y[23],A[23],B[23]);
xor xor26(Y[24],A[24],B[24]);
xor xor27(Y[25],A[25],B[25]);
xor xor28(Y[26],A[26],B[26]);
```



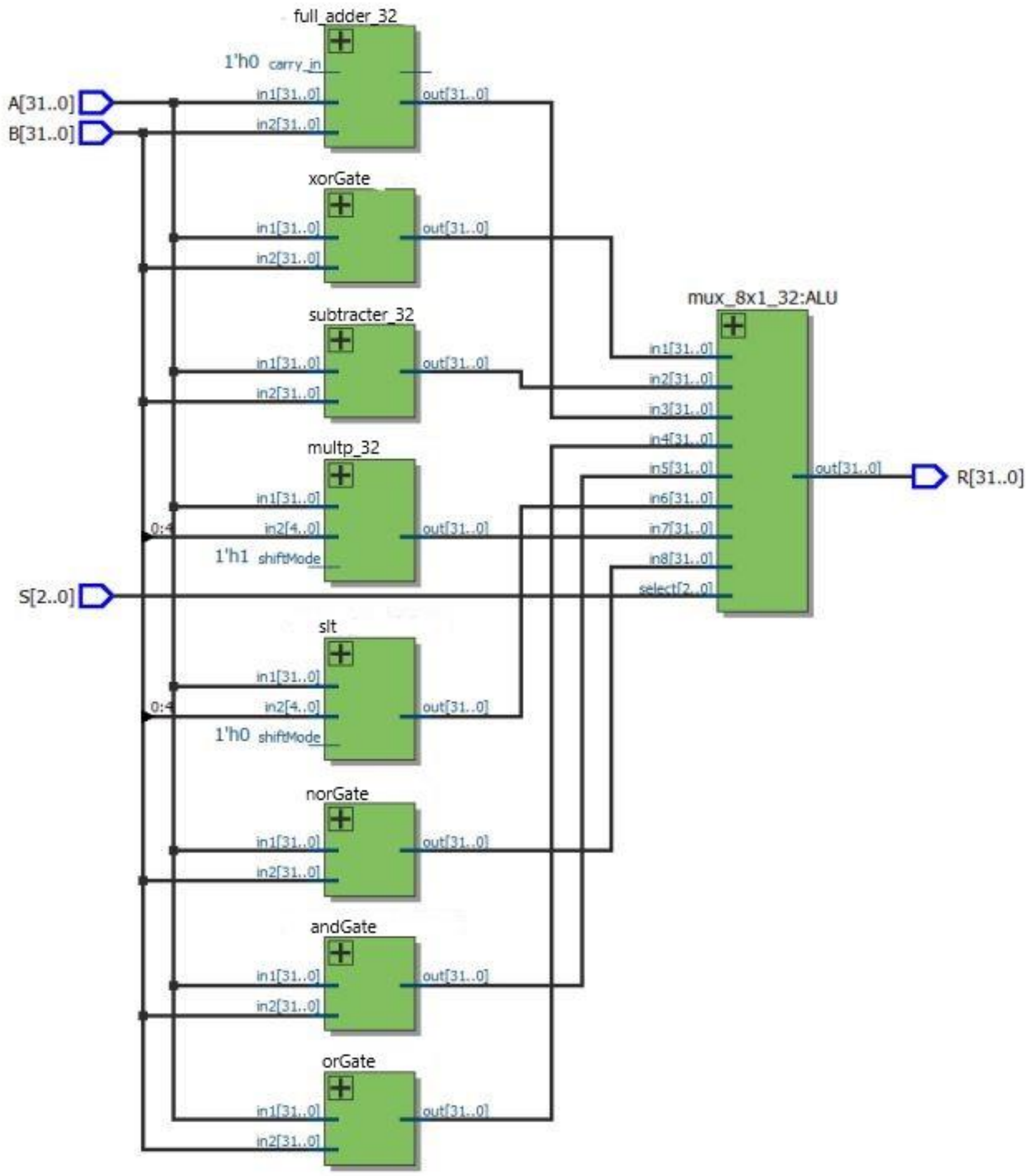
```
module full_adder_32 (out, carry_out, in1, in2, carry_in);
input [31:0] in1, in2;
input carry_in;
output carry_out;
output [31:0] out;
wire [30:0] carry;

full_adder FA1(out[0], carry[0], in1[0], in2[0], carry_in),
   FA2(out[1], carry[1], in1[1], in2[1], carry[0]),
   FA3(out[2], carry[2], in1[2], in2[2], carry[1]),
   FA4(out[3], carry[3], in1[3], in2[3], carry[2]),
   FA5(out[4], carry[4], in1[4], in2[4], carry[3]),
   FA6(out[5], carry[5], in1[5], in2[5], carry[4]),
   FA7(out[6], carry[6], in1[6], in2[6], carry[5]),
   FA8(out[7], carry[7], in1[7], in2[7], carry[6]),
   FA9(out[8], carry[8], in1[8], in2[8], carry[7]),
   FA10(out[9], carry[9], in1[9], in2[9], carry[8]),
   FA11(out[10], carry[10], in1[10], in2[10], carry[9]),
   FA12(out[11], carry[11], in1[11], in2[11], carry[10]),
   FA13(out[12], carry[12], in1[12], in2[12], carry[11]),
   FA14(out[13], carry[13], in1[13], in2[13], carry[12]),
   FA15(out[14], carry[14], in1[14], in2[14], carry[13]),
   FA16(out[15], carry[15], in1[15], in2[15], carry[14]),
   FA17(out[16], carry[16], in1[16], in2[16], carry[15]),
   FA18(out[17], carry[17], in1[17], in2[17], carry[16]),
   FA19(out[18], carry[18], in1[18], in2[18], carry[17]),
   FA20(out[19], carry[19], in1[19], in2[19], carry[18]),
   FA21(out[20], carry[20], in1[20], in2[20], carry[19]),
   FA22(out[21], carry[21], in1[21], in2[21], carry[20]),
   FA23(out[22], carry[22], in1[22], in2[22], carry[21]),
   FA24(out[23], carry[23], in1[23], in2[23], carry[22]),
   FA25(out[24], carry[24], in1[24], in2[24], carry[23]),
   FA26(out[25], carry[25], in1[25], in2[25], carry[24]),
```



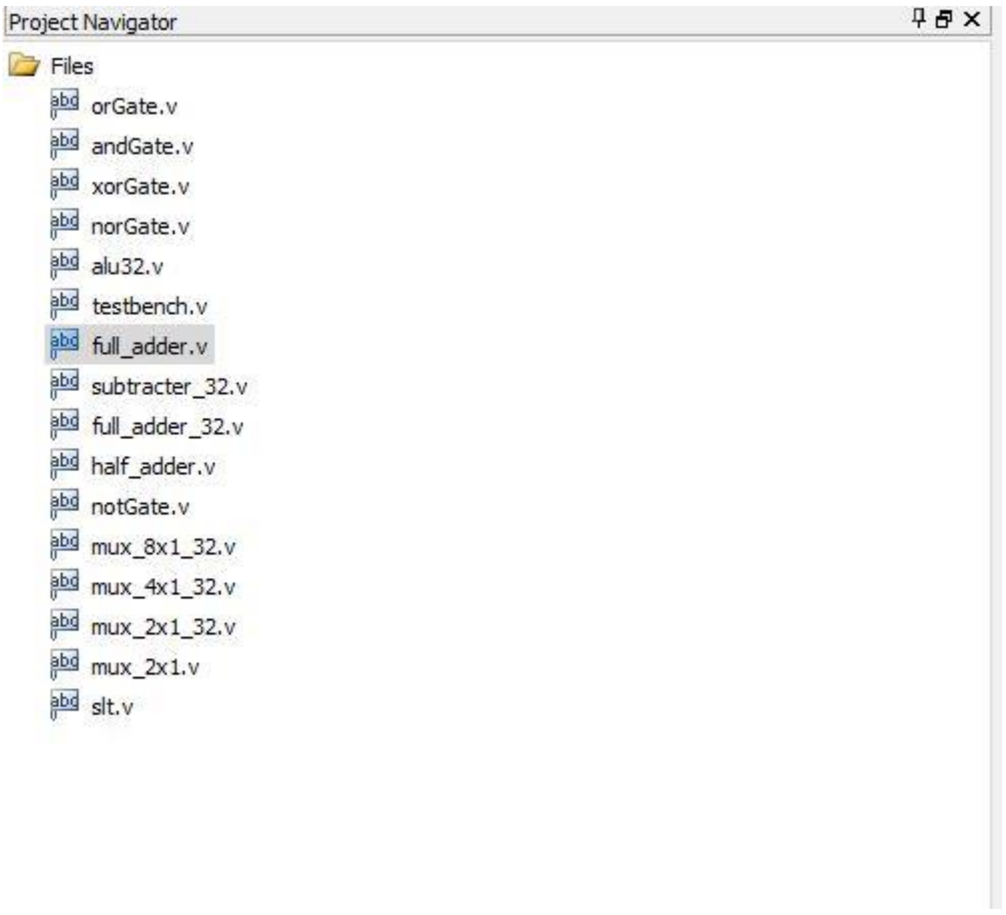
```
module subtracter_32 (out, in1, in2);
input [31:0] in1, in2;
output [31:0] out;
wire [31:0] a;
wire carry;

notGate NOT(a, in2);
full_adder_32 FA(out, carry, in1, a, 32'b1);

endmodule
```

Schematic Design

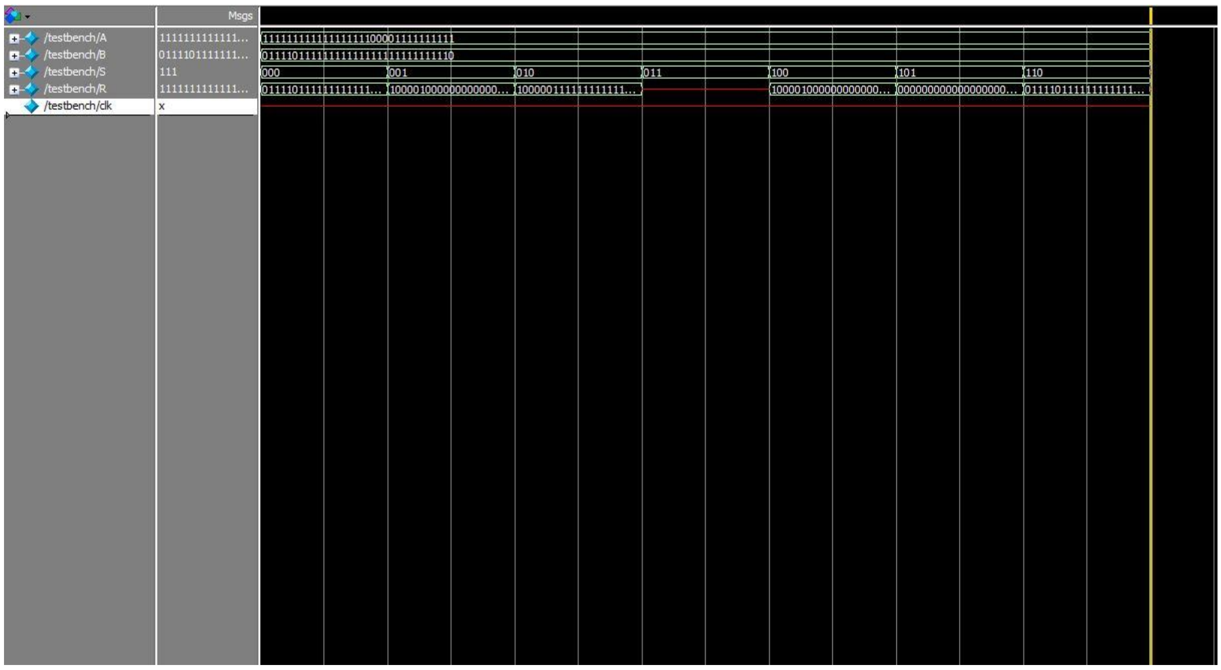
ALU



ModelSım Results



```
Project Navigator
Files
  orGate.v
  andGate.v
  xorGate.v
  norGate.v
  alu32.v
  testbench.v
  full_adder.v
  subtracter_32.v
  full_adder_32.v
  half_adder.v
  notGate.v
  mux_8x1_32.v
  mux_4x1_32.v
  mux_2x1_32.v
  mux_2x1.v
  slt.v
```



```
                   Msgs
/testbench/A       1111111111111...   11111111111111111111000011111111111
/testbench/B       0111101111111...   01111011111111111111111111111111110
/testbench/S       111                000  001  010  011  100  101  110
/testbench/R       1111111111111...   0111101111111111111...  10000100000000000000...  10000011111111111111...  10000100000000000000...  00000000000000000000...  0111101111111111111...
/testbench/clk     x
```