Lab 1

1. Simple TestBench

Code:

```
module lab1simpletestbench(); // Testbench has no inputs, outputs
reg [7:0]A, B;
reg [3:0] ALU Sel; // Will be assigned in initial block
wire [7:0] ALU Out;
wire CarryOut;
// instantiate device under test
alu dut (.A(A), .B(B), .ALU Sel(ALU Sel), .ALU Out(ALU Out), .CarryOut(CarryOut));
// apply inputs one at a time
integer i;
//Select 0 - 15
initial begin // sequential block
       ALU Sel = 4'b0000;
       A = 8'b00001010; B = 8'b00001010;#10; // apply inputs, wait 10ns
       for (i = 0; i < 16; i = i + 1) begin
              display("A = \%h", A);
              display("B = \%h", B);
              $display("ALU Sel = %h", ALU Sel);
              $display("ALU Out = %h", ALU Out);
              $display("CarryOut = %h", CarryOut);
              ALU Sel = ALU Sel + 4'b0001;#10;
       end
end
endmodule
```

Display Output: *All inputs and outputs are in hexadecimal

```
INDIA 12% THE
                # A = 0a
# A = 0a
                # B = 0a
# B = 0a
                # ALU_Sel = 8
# ALU Sel = 0
                # ALU Out = 0a
# ALU Out = 14
                # CarryOut = 0
# CarryOut = 0
                # A = 0a
# A = 0a
                # B = 0a
#B = 0a
                # ALU Sel = 9
# ALU Sel = 1
               # ALU Out = 0a
# ALU Out = 00
                # CarryOut = 0
# CarryOut = 0
                # A = 0a
#A = 0a
                # B = 0a
# B = 0a
                # ALU Sel = a
# ALU_Se1 = 2
                # ALU Out = 00
# ALU Out = 64
                # CarryOut = 0
# CarryOut = 0
                # A = 0a
#A = 0a
                #B = 0a
#B=0a
                # ALU_Sel = b
# ALU Sel = 3
                # ALU_Out = f5
# ALU_Out = 01
                # CarryOut = 0
# CarryOut = 0
                #A = 0a
#A = 0a
                # B = 0a
#B=0a
                # ALU Sel = c
# ALU Sel = 4
               # ALU_Out = f5
# ALU Out = 14
                # CarryOut = 0
# CarryOut = 0
                #A = 0a
# A = 0a
                #B = 0a
# B = 0a
                # ALU Sel = d
# ALU Sel = 5
                # ALU Out = ff
# ALU Out = 05
                # CarryOut = 0
# CarryOut = 0
                A = 0a
#A = 0a
                #B = 0a
#B = 0a
                # ALU_Sel = e
# ALU Sel = 6
               # ALU Out = 00
# ALU_Out = 14
                # CarryOut = 0
# CarryOut = 0
                # A = 0a
# A = 0a
                #B = 0a
#B=0a
                # ALU Sel = f
# ALU Sel = 7
                # ALU_Out = 01
# ALU Out = 05
                # CarryOut = 0
# CarryOut = 0
```

	8'h0a		8'h0a											
	8'h0a		8'h0a											
- /lab 1simpletestbench/ALU_Sel	4'h0	4'h0		4h1	4'h2	4h3	4'h4	4'h5	4'h6	4'h7	4'h8	4'h9	4'ha	4'hb
-/ /lab1simpletestbench/ALU_Out	8'h14		8'h14	8'h00	8'h64	8'h01	8'h14	8'h05	8'h14	8'h05	8'h0a		8'h00	8'hf5
/lab1simpletesthench/CarryOut	1'b0													

+- /lab 1 simpletestbench/A 8"h	h0a	8'h0a												
IIIIIIIIIIIII	h0a	8'h0a												
/ab1simpletestbench/ALU_Sel 4'h	hb	4h5	4'h6	4'h7	4'h8	4'h9	4'ha	4'hb	コ	4'hc	4'hd	4'he	4hf	4'h0
<u>→</u> /lab1simpletestbench/ALU_Out 8"h	hf5	8'h05	8'h14	8'h05	8'h0a	•	8'h00	8'hf5			8'hff	8'h00	8'h01	8'h14
/lab1simpletestbench/CarryOut 1'h	h0													

2. Self-checking test-bench

Code:

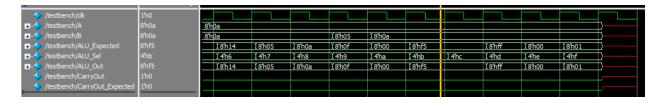
```
module testbench();
reg clk, reset; // clock and reset are internal
reg [7:0]A, B, ALU Expected;
reg [3:0] ALU_Sel; // Will be assigned in initial block
wire [7:0] ALU Out;
wire CarryOut;
reg CarryOut Expected;
reg [31:0] vectornum, errors; // bookkeeping variables
reg [28:0] testvectors[15:0];// array of testvectors
// instantiate device under test
alu dut (.A(A), .B(B), .ALU Sel(ALU Sel), .ALU Out(ALU_Out), .CarryOut(CarryOut));
// generate clock
always begin
       clk = 1; #5; clk = 0; #5; // 10ns period
end
initial begin
       $readmemb("alu.tv", testvectors); // Read vectors
       vectornum = 0; errors = 0; // Initialize
       reset = 1; \#27; reset = 0; // Apply reset wait
end
// Note: $readmemh reads testvector files written in
// hexadecimal
// apply test vectors on rising edge of clk
always @(posedge clk) begin
       #1; {ALU Sel, A, B, ALU Expected, CarryOut Expected} = testvectors[vectornum];
end
// check results on falling edge of clk
always @(negedge clk)
       if (~reset) begin
              if (ALU Out !== ALU Expected || CarryOut !== CarryOut Expected) begin
                     $display("Error: inputs = ALU Sel:%b A:%b B:%b ALU Out:%b",
ALU_Sel, A, B, ALU_Out);
                     $display(" outputs = ALU Out:%b ALU Expected:%b CarryOut:%b
CarryOut Expected: %b", ALU Out, ALU Expected, CarryOut, CarryOut Expected);
                     errors = errors + 1;
              end
```

```
vectornum = vectornum + 1;
            if (testvectors[vectornum] === 29'bx) begin
                   $display("%d tests completed with %d errors", vectornum, errors);
                   $finish; // End simulation
             end
      end
endmodule
alu.tv:
0000 10000000 10000010 00000010 1
0001 00001010 00001010 00000000 0
0010 00000010 00000011 00000110 0
0011 00001010 00001010 00000001 0
0100 00001010 00001010 00010100 0
0101 \ 00001010 \ 00001010 \ 00000101 \ 0
0110\ 00001010\ 00001010\ 00010100\ 0
0111 00001010 00001010 00000101 0
1000 00001010 00001010 00001010 0
1001 00001010 00000101 00001111 0
1010 00001010 00001010 00000000 0
1011 00001010 00001010 11110101 0
1100 00001010 00001010 11110101 0
1101 00001010 00001010 111111111 0
1110 00001010 00001010 00000000 0
1111 00001010 00001010 00000001 0
```

Display Output: *All inputs and outputs are in hexadecimal

```
| 16 tests completed with 0 errors
| ** Note: $finish : C:/Modeltech_pe_edu_10.4a/examples/testbench.v(43)
| Time: 185 ns Iteration: 1 Instance: /testbench
| 1
| Break in Module testbench at C:/Modeltech_pe_edu_10.4a/examples/testbench.v line 43
```

/testbench/dk	1'h0												
 /testbench/A	8'h0a	8'h80	(8'h0a) 8"h02) 8'h0a								
■ - / /testbench/B	8'h0a	8'h82	(8'h0a	(8h03	(8'h0a						(8'h05	(8'h0a	
<u>→</u> /testbench/ALU_Expected	8'hf5	8'h02	(8'h00	(8'h06	(8h01	(8h14	(8'h05	(8h14	(8'h05	(8'h0a	(8'h0f	(8'h00	(8'hf5
II	4'hb	4'h0	(4h1) 4h2	(4h3	(4h4	(4h5) 4'h6	(4h7	(4h8	(4h9	(4ha	∬4'hb
	8'hf5	8'h02	(8'h00	(8'h06	(8h01	(8h14	(8'h05	(8'h14	(8'h05	(8'h0a	(8'h0f	(8'h00	(8'hf5
/testbench/CarryOut	1'h0												
/testbench/CarryOut_Expected	1'h0												



Part-B: Memory testing using tasks

```
Memory.v
```

```
module memory (addr, ce, wren, rden, wr_data, rd_data, clk);
  input [7:0] addr, wr_data;
  output reg [7:0] rd_data;
  input ce, wren, rden;
  input clk;
  reg [7:0] mem [0:255];
 always @ (posedge clk) if (ce)
p begin
    if (rden)
        rd_data <= mem[addr];
    else if (wren)
        mem[addr] <= wr_data;
  end
 endmodule
module testbench();
reg clk, reset; // clock and reset are internal
reg [7:0]addr, wr data;
wire [7:0] rd data;
reg ce, wren, rden;
// instantiate device under test
memory dut (.addr(addr), .ce(ce), .wren(wren), .rden(rden), .wr data(wr data),
.rd data(rd data), .clk(clk));
// generate clock
always begin
        clk = 1; #5; clk = 0; #5; // 10ns period
end
// Memory write task
task mem write;
 input [7:0] address;
 input [7:0] data;
 begin
  $display ("%g Memory Write task with address: %h Data: %h", $time, address,data);
  $display ("%g -> Driving CE, WREN, WR data and ADDRESS on to bus", $time);
  @ (posedge clk);
// addr, ce, wren, wr data are the signals connected to the memory model
  addr = address;
  ce = 1;
```

```
wren = 1;
  wr data = data;
  @ (posedge clk);
  addr = 0;
  ce = 0;
  wren = 0;
  $display ("====
 end
endtask
// Memory Read task
task mem read;
 input [7:0] address;
 input [7:0] data;
 begin
  $display ("%g Memory Read task with address: %h", $time, address);
  $display ("\%g -> Driving CE, RDEN, RD data and ADDRESS on to bus", $time);
  @ (posedge clk);
// addr, ce, rden, rd data are the signals connected to the memory model
  addr = address;
  ce = 1;
  rden = 1;
  @ (posedge clk);
  addr = 0;
  ce = 0;
  rden = 0;
  @ (posedge clk);
  data = rd data;
  $display ("%g Value at address : %h = %h", $time, address, data);
  $display ("======");
 end
endtask
initial begin
      reset = 1; \#27; reset = 0; // Apply reset wait
       #1 mem write (8'h11, 8'hAA); // writing 0xAA to the memory address 0x11
       #1 mem read (8'h11, rd data); // reading from memory address 0x11 to the rd data
signal
       #1 mem write (8'h12, 8'hA1); // writing 0xA1 to the memory address 0x12
       #1 mem read (8'h12, rd data); // reading from memory address 0x12 to the rd data
signal
       #1 mem write (8'h13, 8'hA2); // writing 0xA2 to the memory address 0x13
```

```
#1 mem read (8'h13, rd data); // reading from memory address 0x13 to the rd data
signal
       #1 mem write (8'h14, 8'hA3); // writing 0xA3 to the memory address 0x14
       #1 mem read (8'h14, rd data); // reading from memory address 0x14 to the rd data
signal
       #1 mem write (8'h15, 8'hA4); // writing 0xA4 to the memory address 0x15
       #1 mem read (8'h15, rd data); // reading from memory address 0x15 to the rd data
signal
       #1 mem write (8'h16, 8'hA5); // writing 0xA5 to the memory address 0x16
       #1 mem read (8'h16, rd data); // reading from memory address 0x16 to the rd data
signal
       #1 mem write (8'h17, 8'hA6); // writing 0xA6 to the memory address 0x17
       #1 mem read (8'h17, rd data); // reading from memory address 0x17 to the rd data
signal
       #1 mem write (8'h18, 8'hA7); // writing 0xA7 to the memory address 0x18
       #1 mem read (8'h18, rd data); // reading from memory address 0x18 to the rd data
signal
       #1 mem write (8'h19, 8'hA8); // writing 0xA8 to the memory address 0x19
       #1 mem read (8'h19, rd data); // reading from memory address 0x19 to the rd data
signal
       #1 mem write (8'h1a, 8'hA9); // writing 0xA9 to the memory address 0x1a
       #1 mem read (8'h1a, rd data); // reading from memory address 0x1a to the rd data
signal
end
endmodule
```

	→ /memoryTestBench/dk	1h1																		
Immon/TestBench/u, data Shap Sh	/memoryTestBench/reset	1'h0																		
Immony/TestBench/uk 10 10 10 10 10 10 10 1	/memoryTestBench/addr	8'h00				8'h11	8'h00	8h11	8'h00		8h12	8'h00	8h12	8'h00		8h13	[8h00	8h13	[8/h00	
/memoryTestBendylice	/memoryTestBench/wr_data	8'ha9				8'haa					8'ha1					8ha2				
	/memoryTestBench/rd_data	8'ha9							8 haa					8'ha1					[8ha2	
	/memoryTestBench/ce	1h0																		
Morary TestBench/(ck	/memoryTestBench/wren	1h0																		
Internor/TestBendrick	/memoryTestBench/rden	1h0																		
MemoryTesBendylds																				
Internor/TestBendrick																				
Internor/TestBench/lest 170 Internor/TestBench/lest 17																				
/memory/TestBendy/dx																				
Internor/TestBench/leaset 17:0 Internor/TestB	•	Msgs																		
Internor/TestBench/lada Sh00 Sh14 Sh00 Sh14 Sh00 Sh15 S	/memoryTestBench/dk	1h1																		
InemaryTestBench/ur_data 8ha9	/memoryTestBench/reset	1h0																		
	/memoryTestBench/addr	8'h00	8h14	8'h00	[8h14	8*h00		8h15	8'h00	8h15	8'h00		8h16	8'h00	8h16	(8'h00		8'h17	8'h00	8'h
✓ [nemary/TestBench]/ce 1h0 ✓ [nemary/TestBench]/wren 1h0	/memoryTestBench/wr_data	8'ha9	(8'ha3					8ha4					8ha5					8'ha6		
→ /memoryTestBench/wren 1h0	/memoryTestBench/rd_data	8'ha9	8ha2			8 ha 3					8'ha4					[8'ha5				
	/memoryTestBench/ce	1h0																		
memoryTestBench/rden 1h0	/memoryTestBench/wren	1h0																		
	/memoryTestRench/rden	1h0																		
	manory resource;																			
	menory research preci																			
Mogs I																				
	•	Msgs																		
→ //memoryTestBench/ck Ih1 Ih1 Ih1 Ih1 Ih1 Ih1 Ih1 Ih1		Msgs																		
✓ /memory/TestBench/ck 1h1 ✓ /memory/TestBench/reset 1h0		Msgs 1h1 1h0	8500		8h18	T8/b00	8h18	18h00		8519	8500	87119	8000		Shia	8500	Shia.	8000		
Immort/TestBendy/dext 1h1 Immort/TestBendy/dext 1h0	/memoryTestBench/dk /memoryTestBench/addr	Mags 17h1 17h0 87h00				I8'h00	8h18	[8]h00			(8°h00	8h19	8°h00			[8h00	Sh1a	8°h00		
/memoryTestBench/lck	/memoryTestBench/ck /memoryTestBench/reset /memoryTestBench/wr_data	Msgs 1h1 1h0 8h00 8ha9	8ha6			(8'h00	[8h18				(8h00	8h19				X8'h00				
	// // // // // // // // // // // // //	Mogs 1h1 1h0 8h00 8ha9 8ha9	8ha6			[8h00	8h18				[8"h00	8h19				X8'h00				
memoryTestBendylex 1h1	/memoryTestBench/dk /memoryTestBench/prest /memoryTestBench/sdr /memoryTestBench/sdr /memoryTestBench/rd_data /memoryTestBench/rd_data	Msgs 151 150 8500 8549 8549 150 150 150 150 150 150 150 150 150 150	8ha6			[8h00	8h18				(8°h00	8h19				(8'h00				
InternoryTestBench/let 101	/memoryTestBench/dk /memoryTestBench/leset /memoryTestBench/ledr /memoryTestBench/leds /memoryTestBench/leds /memoryTestBench/leds /memoryTestBench/led /memoryTestBench/led	Msgs 1h1 1h0 8h00 8ha9 8ha9 1h0 1h0 1h0	8ha6			[Sh00	8h18				(8'h00	8h19				I8h00				

```
|YDIM DZ> run
|# 28 Memory Write task with address : 11 Data : aa
|# 28 -> Driving CE, WREN, WR data and ADDRESS on to bus
341 Memory Read task with address: 17
341 -> Driving CE, RDEN, RD data and ADDRESS on to bus
370 Value at address: 17 = a6
121 Memory Write task with address: 13 Data: a2
121 -> Driving CE, WREN, WR data and ADDRESS on to bus
141 Memory Read task with address: 13
141 -> Driving CE, RDEN, RD data and ADDRESS on to bus
170 Value at address: 13 = a2
                                                                         371 Memory Write task with address : 18 Data : a7
                                                                         371 -> Driving CE, WREN, WR data and ADDRESS on to bus
# 171 Memory Write task with address: 14 Data: a3
# 171 -> Driving CF, WREN, WR data and ADDRESS on to bus
# 191 Memory Read task with address: 14
# 191 -> Driving CF, RDEN, RD data and ADDRESS on to bus
# 220 Value at address: 14 = a3
                                                                         391 Memory Read task with address : 18
                                                                         391 -> Driving CE, RDEN, RD data and ADDRESS on to bus
                                                                         420 Value at address: 18 = a7
                                                                         421 Memory Write task with address : 19 Data : a8
  221 Memory Write task with address : 15 Data : a4
221 -> Driving CE, WREN, WR data and ADDRESS on to bus
                                                                         421 -> Driving CE, WREN, WR data and ADDRESS on to bus
441 Memory Read task with address: 19
                                                                         441 -> Driving CE, RDEN, RD data and ADDRESS on to bus
                                                                         470 Value at address: 19 = a8
  271 Memory Write task with address : 16 Data : a5
                                                                         471 Memory Write task with address : la Data : a9
  271 -> Driving CE, WREN, WR data and ADDRESS on to bus
                                                                         471 -> Driving CE, WREN, WR data and ADDRESS on to bus
  291 Memory Read task with address : 16
291 -> Driving CE, RDEN, RD data and ADDRESS on to bus
320 Value at address : 16 = a5
                                                                         491 Memory Read task with address : la
                                                                         491 -> Driving CE, RDEN, RD data and ADDRESS on to bus
  321 Memory Write task with address : 17 Data : ad
321 -> Driving CE, WREN, WR data and ADDRESS on to bus
                                                                         520 Value at address : la = a9
```