# Lab 06 – Worksheet

|  |  |  |
| --- | --- | --- |
| Name: Manqad Raza  Abdullah Amir | ID:08456,  08453 | Section: T2 |

## Task 1.

**Code: Design module & testbench**

*Provide appropriately commented code for designed module & its testbecnch, code should contain meaningful variable naming.*

*\*Add snippet or Copy paste the code written in the Editor window. Make sure the irrelevant area of the snip is cropped.*

|  |
| --- |
| **Module**  `timescale 1ns / 1ps  module ALU\_1\_bit(  input wire a,  input wire b,  input wire CarryIn,  input [3:0]ALUop,  output wire Result,  output wire CarryOut  );    // carryout calculation  assign CarryOut = (a&CarryIn) | (b&CarryIn) | (a&b);  wire mux1out, mux2out;  wire ainvert = ALUop[3], binvert = ALUop[2];  wire abar = ~a, bbar = ~b;    // using two mux to select between the value and its complement  assign mux1out = ainvert ? abar : a;  assign mux2out = binvert ? bbar : b;    // calculating Result based on the [1:0] ALUop bits  assign Result = ALUop[1] ? (mux1out + mux2out): ALUop[0] ? (mux1out || mux2out) : (mux1out & mux2out);  endmodule  **Testbench:**  `timescale 1ns / 1ps  module test\_ALU( );  reg a;  reg b;  reg CarryIn;  reg [3:0]ALUop;  wire Result;  wire CarryOut;  ALU\_1\_bit A(a, b, CarryIn, ALUop, Result, CarryOut);    initial  begin  a = 1'b1;  b = 1'b1;  CarryIn = 1'b0;    //testing AND  ALUop = 4'b0000;    #25  //testing OR  ALUop = 4'b0001;    #25  //testing Addition  ALUop = 4'b0010;    #25  // testing Subtraction  CarryIn = 1'b1;  ALUop = 4'b0110;    #25  //testing NOR  ALUop = 4'b1100;    end  endmodule  **Schematic:** |

## Results (Waveforms)

*\*Add snip of relevant signals’ waveforms. Make sure the irrelevant area of the snip is cropped.*

|  |
| --- |
|  |

## Comments

*\*Observation/Comments on the obtained results/working of code.*

|  |
| --- |
| This module takes two 1-bit numbers, a and b, and depending on the value of the ALUop, choose one of the five operations: AND, OR, Addition, Subtraction and NOR using multiplexers. The desired operation is performed at the suitable selection bits of ALUop and the is the given as output Result. Whereas the Carryout bit is calculated separately using the CarryIn bit given as input. |

## Task 2.

**Code: Design module & testbench**

*Provide appropriately commented code for designed module & its testbecnch, code should contain meaningful variable naming.*

*\*Add snippet or Copy paste the code written in the Editor window. Make sure the irrelevant area of the snip is cropped.*

|  |
| --- |
| **Module:**  `timescale 1ns / 1ps  module ALU\_8\_bit(  input wire [7:0]a,  input wire [7:0]b,  input wire CarryIn,  input [3:0]ALUop,  output wire [7:0]Result,  output wire CarryOut  );  wire Carry0, Carry1, Carry2, Carry3, Carry4, Carry5, Carry6, Carry7;    // instantiating the ALU 1-bit module 8 times with different carry outs  ALU\_1\_bit A0(a[0], b[0],CarryIn, ALUop,Result[0], Carry0);    ALU\_1\_bit A1(a[1], b[1],Carry1, ALUop,Result[1], Carry1);    ALU\_1\_bit A2(a[2], b[2],Carry1, ALUop,Result[2], Carry2);    ALU\_1\_bit A3(a[3], b[3],Carry2, ALUop,Result[3], Carry3);    ALU\_1\_bit A4(a[4], b[4],Carry3, ALUop,Result[4], Carry4);    ALU\_1\_bit A5(a[5], b[5],Carry4, ALUop,Result[5], Carry5);    ALU\_1\_bit A6(a[6], b[6],Carry5, ALUop,Result[6], Carry6);    ALU\_1\_bit A7(a[7], b[7],Carry6, ALUop,Result[7], CarryOut);    endmodule  **Testbench:**  module bit8test();  reg [7:0]a;  reg [7:0]b;  reg CarryIn;  reg [3:0]ALUop;  wire [7:0]Result;  wire CarryOut;    ALU\_8\_bit M(a ,b, CarryIn, ALUop, Result, CarryOut);    initial  begin    a=8'b11111111;  b=8'b00000000;  CarryIn=1'b0;    //testing AND    ALUop=4'b0000;    #50  // testing OR  ALUop=4'b0001;    #50  // testing Addition  ALUop=4'b0010;    #50  // testing Subtraction  CarryIn = 1'b1;  ALUop = 4'b0110;    #50  // testing NOR  ALUop=4'b1100;    end    endmodule  **Schematic:** |

## Results (Waveforms)

*\*Add snip of relevant signals’ waveforms. Make sure the irrelevant area of the snip is cropped.*

|  |
| --- |
|  |

## Comments

*\*Observation/Comments on the obtained results/working of code.*

|  |
| --- |
| In this ALU\_8\_bit module, we instantiate the ALU 1-bit module 8 times, each instantiation calculates the result bit for one of the 8 bits in the two 8 bit inputs, a and b, passed in the call. The carryout from the first bit instantiation is passed onto the next one as carryin input, until the last bit module when the carryout is the resultant carryout. In this way, we perform the five operations of the ALU 1-bit module for 8 bit numbers. |

**Task3**

**Code: Design module & testbench**

*Provide appropriately commented code for designed module & its testbecnch, code should contain meaningful variable naming.*

*\*Add snippet or Copy paste the code written in the Editor window. Make sure the irrelevant area of the snip is cropped.*

|  |
| --- |
| **Module**:  `timescale 1ns / 1ps  module ALU\_64\_bit(  input [63:0]a,  input [63:0]b,  input [3:0]ALUop,  output reg [63:0]Result,  output wire Zero  );    always @(\*)  begin    case (ALUop)    // when AND  4'b0000:  Result = (a&b);    // when OR  4'b0001:  Result = (a|b);    // when Addition  4'b0010:  Result = (a+b);  // when Subtraction  4'b0110:  Result = (a-b);  // when NOR  4'b1100:  Result = ~(a|b);    default:  Result = 64'h0000;    endcase  end    // giving zero high if the result = 0 and low otherwise  assign Zero = Result ? 0 : 1;    endmodule  **Testbench**:  `timescale 1ns / 1ps  module test\_64bit\_ALU();  reg [63:0]a;  reg [63:0]b;  reg [3:0]ALUop;  wire [63:0]Result;  wire Zero;    ALU\_64\_bit S(a, b, ALUop, Result, Zero);    initial  begin    a=64'h1111;  b=64'h0000;    //testing AND  ALUop=4'b0000;    #50  // testing OR  ALUop=4'b0001;    #50  // testing Addition  ALUop=4'b0010;      #50  // testing NOR  ALUop=4'b1100;    end    endmodule  **Schematic:** |

## Results (Waveforms)

*\*Add snip of relevant signals’ waveforms. Make sure the irrelevant area of the snip is cropped.*

|  |
| --- |
|  |

## Comments

|  |
| --- |
| In this ALU 64-bit module, we take two 64-bit inputs, a and b, and depending on the input value of the ALUop, we have different case for performing the different operations of the ALU and calculating the result. There is another output bit, zero, which is high when the result is equal to 0 and low otherwise. |

*\*Observation/Comments on the obtained results/working of code.*