EC-204

<LAB- 2>

NITK SURATHKAL

INBASEKARAN.P

201EC226

Prof: Sumam S

1. Half adder (a) using gates (b) using dataflow model (using assign statement).

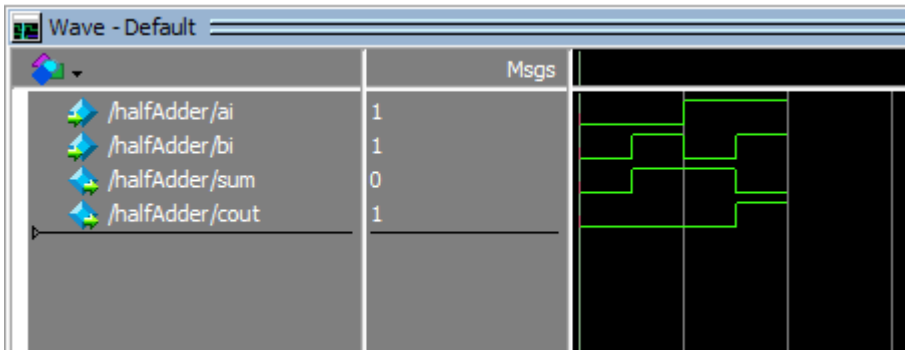
Solution:

a) Verilog code using gates

question1 > halfAdder.v

```
module halfAdder(input ai,bi, output sum,cout);
  // Structural model for half adder using gates
  xor(sum, ai, bi);
  and(cout, ai, bi);
endmodule
```

Simulation waveform using gates

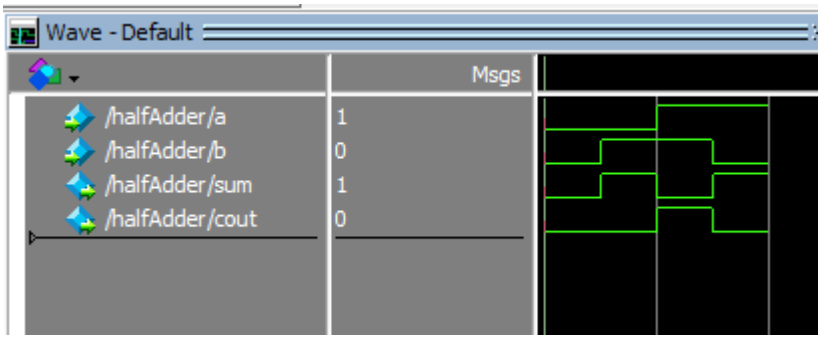


b) Verilog code using dataflow model

question1 > halfAdderDataFlow.v

```
module fullAdder(input a,b, output sum,cout);
  // Data flow model of half adder
  assign sum = a^b;
  assign cout = (a&b);
endmodule
```

Simulation waveform using dataflow model



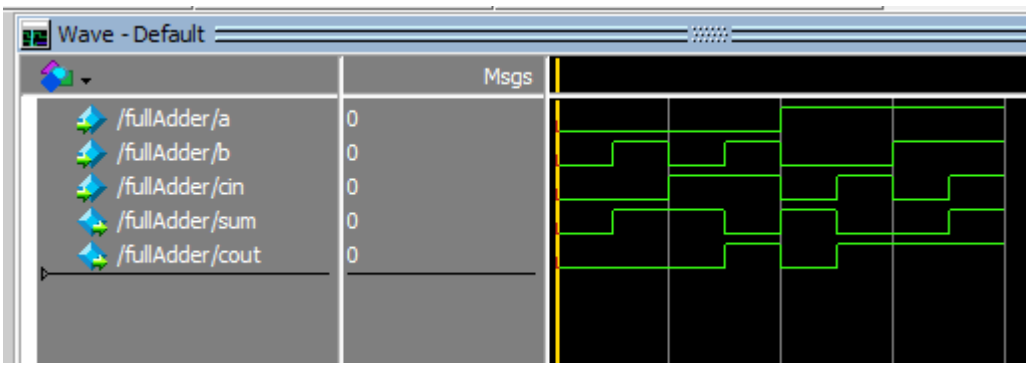
2. Full adder (a) using dataflow model (using assign statement) (b) using two half adders and an OR gate.

Solution:

Verilog code using dataflow model

```
question2 > fullAdder_dataFlowModel.v
1   module fullAdder(input a,b,cin, output sum,cout);
2     // Data flow model of full adder
3     assign sum = a^b^cin;
4     assign cout = (a&b)|(a&cin)|(b&cin);
5   endmodule
```

Simulation waveform using dataflow model



Verilog code

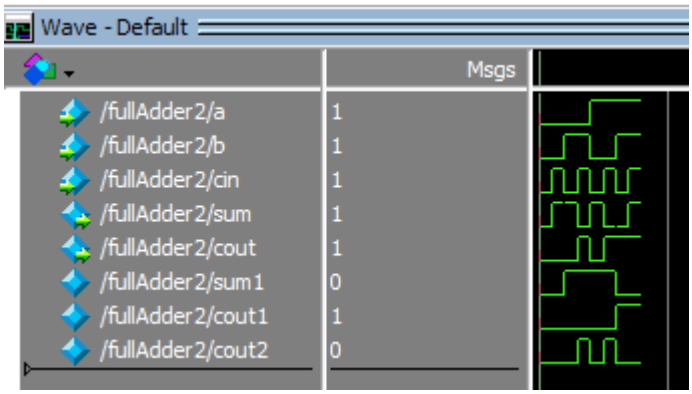
question2 > halfAdder.v

```verilog
module halfAdder(input ai,bi, output sum,cout);
  // Structural model for half adder using gates
  xor(sum, ai, bi);
  and(cout, ai, bi);
endmodule
```

question2 > fullAdder_2Half.v

```verilog
module fullAdder2(input a,b,cin, output sum,cout);
  // Full adder using two half adders
  `include "halfAdder.v"
  wire sum1, cout1, cout2;
  halfAdder
    h1(a, b, sum1, cout1),
    h2(cin, sum1, sum, cout2);
  assign cout = cout1|cout2;
endmodule
```

Simulation waveform



3. Two to one multiplexer using behavioral model (using if statement).

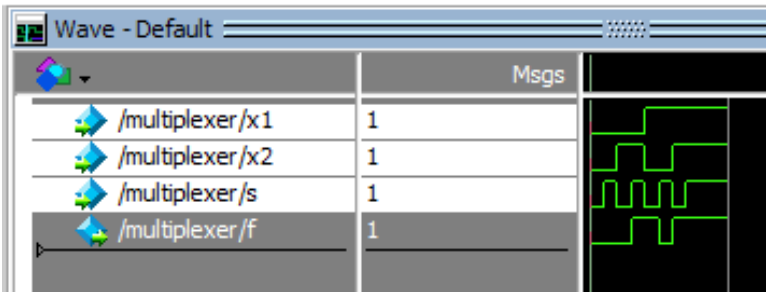
Solution:

Verilog code

question3 > multiplexerBehav.v

```
module multiplexer(x1,x2,s,f);
  // Behavioural model of multiplexer
  input x1,x2,s;
  output f;
  reg f;
  always @(x1,x2,s)
  begin
    if (s==0)
      begin
        f = x1;
      end
    else
      begin
        f = x2;
      end
  end
endmodule
```

Simulation waveform



4. 4-bit ripple carry adder using full adder modules.

Solution:

Verilog code

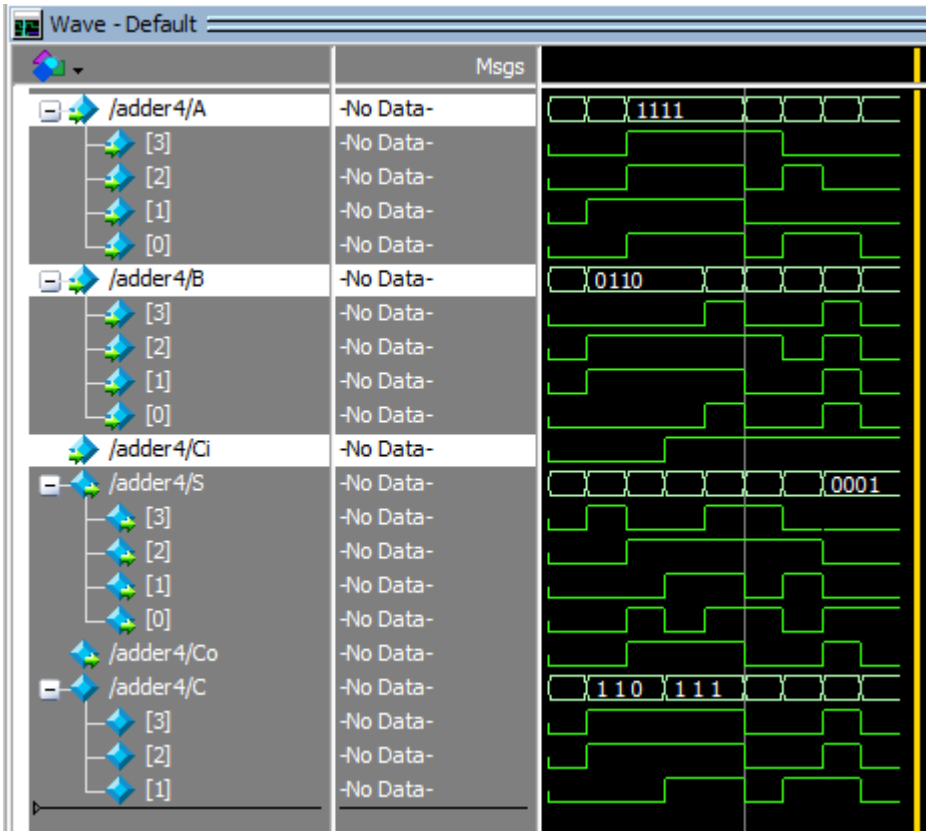
question4 > fullAdder.v

```
module fullAdder(input a,b,cin, output sum,cout);
  // Data flow model of full adder
  assign sum = a^b^cin;
  assign cout = (a&b)|(a&cin)|(b&cin);
endmodule
```

question4 > adder4.v

```
module adder4 (input [3:0] A, input [3:0] B, input Ci, output [3:0] S, output Co);
  wire C [3:1];
  `include "fullAdder.v"
  fullAdder
    ufa1(.a(A[0]), .b(B[0]), .cin(Ci), .sum(S[0]), .cout(C[1])),
    ufa2(A[1], B[1], C[1], S[1], C[2]),
    ufa3(A[2], B[2], C[2], S[2], C[3]),
    ufa4(A[3], B[3], C[3], S[3], Co);
endmodule
```

Simulation waveform



5. A four variable logic function that is equal to 1 if any three or all four of its variables are equal to 1 is called a majority function. Write a Verilog code that implements this majority function. Use the Boolean equation derived in Lab1 in assign statement.

Solution:

Verilog code

question5 > majorityFucntion.v

```
module moduleName (a, b, c, d, y);
  // Verilog code that implements this majority function
  input a,b,c,d;
  output y;
  wire aPb;
  or(aPb, a, b);
  wire aPc;
  or(aPc, a, c);
  wire aPd;
  or(aPd, a, d);
  wire bPc;
  or(bPc, b, c);
  wire bPd;
  or(bPd, b, d);
  wire cPd;
  or(cPd, c, d);
  and(y, aPb, aPc, aPd, bPc, bPd, cPd);
endmodule
```

Simulation waveform