| CSE 308: Digital System Design | Homework 1 Solution |

Problem 1: Draw the logic circuit described by the Verilog code below.

module pie( x, y, a, b, c );

input a, b, c; output x, y;

wire t1, t2;

xor x1(t1,a,b);

not n1(x,t1);

and a1(t2,x,c);

or o1(y,t2,b);

endmodule

a

b

c

*Solution:*

pie

x

t1 t2

y

1

Problem 2: Draw the logic circuit described by the Verilog module twoterms below.

• Show the contents of each instantiated module. (That is, do not just show a box labeled term1100 or twoterms.)

• Show using AND, OR, and NOT gates, inferring the correct gate for the Verilog operators used in the assign expression.

• To the extent possible, label the diagram using the port names defined by twoterms (x, i, j, k, and m).

module term1100(x,a,b,c,d);

input a, b, c, d; output x;

assign x = a && b && !c && !d;

endmodule

module twoterms\_bundle(x,a);

input [3:0] a; output x;

wire tx101, t100x;

term1100 t0(tx101, a[2], a[0], a[1], 1’b0);

term1100 t1(t100x, a[3], a[3], a[1], a[2]);

or o1(x, tx101, t100x);

endmodule

module twoterms(x,i,j,k,m);

input i,j,k,m; output x;

wire [3:0] bundle;

assign bundle[3:0] = {i,j,k,m};

twoterms\_bundle t(x,bundle);

endmodule

*Solution appears below. The labels defined in module* twoterms *appear in* green bold*, they are shown in modules* twoterms bundle *and in* term1100 *alongside the signal names defined in the respective modules.* twoterms

t

This is 4 wires (4 bits).

a

4

msb lsb

twoterms\_bundle

**m** 1

lsb

3:3

a[3]

2:2

a[2]

1:1

a[1]

0:0

a[0]

t0

**k**

1

bundle

1 1 1 1

a b

**j**

**m**

term1100

**j i**

1

1

msb

**k**

This is 1 wire (1 bit).

0

cx d

t1

tx101

x

x

**i**

**i**

**m**

**i**

**i**

**k**

a b c d

**i i**

term1100

x

t100x

Port names de ned by twoterms. 

2

Problem 3: The logic below consists of three repeated parts. Write a Verilog explicit structural description of the logic which consists of two modules, one module, name it part, will be for the part that’s repeated, the other, name it whole, will instantiate part three times and interconnect them appropriately. Choose appropriate inputs and outputs for the two modules based on the diagram.

a[0] b[0]

c

x[0]

a[1] b[1]

x[1]

a[2] b[2]

x[2]y

*Solution appears below.*

// SOLUTION

module part(x,y,a,b,c);

input a, b, c;

output x, y;

wire nb;

not n1(nb,b);

and a1(x,a,nb);

or o1(y,x,c);

endmodule;

module whole(x,y,a,b,c);

input [2:0] a, b;

input c;

output [2:0] x;

output y;

wire y0, y1;

part p0( x[0], y0, a[0], b[0], c );

part p1( x[1], y1, a[1], b[1], y0 );

part p2( x[2], y, a[2], b[2], y1 );

endmodule;

Problem 4: Replace each assign statement below with explicit structural code. Consider each assign statement in isolation (they are not part of the same module). There is no need to show the module declarations.

*Solution appears below. There are several approaches to solving this. Perhaps the easiest is to first draw a truth table based on the* assign *and then determine the logic gates from the truth table. Another approach is to realize that the expression* I ? J : K *is equivalent to the logic* I · J + I · K*. For the first assign we get* (a · b) · 1 + a · b · 0 *which simplifies to* a · b*, just an AND gate.*

3

*A common mistake is using an AND gate to test equality. It’s an XNOR (not exclusive or) that tests equality. That is,* a == b *is equivalent to the Boolean expression* a ⊕ b *(an XNOR operation). So for the second* assign *we have*

a ⊕ b· 0 + a ⊕ b· 1 *which simplifies to* a ⊕ b*, a single XOR gate.*

assign x = a & b ? 1 : 0;

// SOLUTION

and a1(x,a,b);

assign x = a == b ? 0 : 1;

// SOLUTION

xor x1(x,a,b);

assign x = a ? b : c;

// SOLUTION

wire t1, t2, na;

or o1(x,t1,t2);

and a1(t1,a,b);

not n1(na,a);

and a2(t2,na,c);

4

Problem 5: Write a Verilog explicit structural description of the following logic.

x

y

z

w

*Solution appears below.*

module problem\_3(out,x,y,z,w);

input x,y,z,w;

output out;

and a1(xy,x,y);

not n1(nz,z);

and a2(xynz,x,y,nz);

and a3(xyw,x,y,w);

or o1(out,xy,xynz,xyw);

endmodule

Problem 6: Write a Verilog implicit structural description of the following logic.

x

y

z

w

*Solution appears below.*

module problem\_4(out,x,y,z,w);

input x,y,z,w;

output out;

assign out = x && y || x && y && ~z || x && y && w;

endmodule

5