Digital Design Using HDLs

LSU EE 4755

Midterm Examination

Wednesday, 30 October 2019  10:30–11:20 CDT

Problem 1  ________  (20 pts)
Problem 2  ________  (25 pts)
Problem 3  ________  (27 pts)
Problem 4  ________  (28 pts)

Alias  ____________________________  Exam Total  ________  (100 pts)

Good Luck!
Problem 1: [20 pts] Appearing below is one of the solutions to Homework 2, the count leading zeros module.

```verilog
module clz_bi_tree #( int w = 19, int ww = $clog2(w+1) )
  ( output uwire [ww:1] nlz, input uwire [w:1] a );
if ( w == 1 ) begin
  assign nlz = ~ a;
end else begin
  localparam int wlo = w/2, whi = w - wlo;
  localparam int wwlo = $clog2(wlo+1), wwhi = $clog2(whi+1);
  uwire [wwlo:1] lz_lo;
  uwire [wwhi:1] lz_hi;
  clz_bi_tree #(wlo) clo( lz_lo, a[wlo:1] );
  clz_bi_tree #(whi) chi( lz_hi, a[w:whi+1] );
  assign nlz = lz_lo < wlo ? lz_lo : wlo + lz_hi;
end
endmodule
```

Show the hardware that will be inferred for the module for \(w > 1\). Just show one level, don’t show what is inside of clo and chi.

☐ Show synthesized hardware for one level.  ☐ Be sure to show clo and chi (but not their contents).
Problem 2: [25 pts] In Homework 2 a clz (count leading zeros) module was constructed recursively by splitting the input bit vector and connecting each half to a smaller instance. The incomplete module below is similar except that the input vector is to be split into thirds and each third connected to a recursive instance. Complete the module.

Complete so that clz_tri_tree computes clz.

```verilog
module clz_tri_tree #( int w = 19, int ww = $clog2(w+1) )
( output uwire [ww-1:0] nlz, input uwire [w-1:0] a );

if ( w == 1 ) begin
assign nlz = ~ a;

// Make any needed changes to terminal case(s).
end else begin

// Finish these localparams.
localparam int wlo =

localparam int wmi =

localparam int whi =

localparam int wwlo = $clog2(wlo+1), wwmi = $clog2(wmi+1), wwhi = $clog2(whi+1);

uwire [wwlo-1:0] lz_lo; // No need to change these four lines.

uwire [wwmi-1:0] lz_mi;

uwire [wwhi-1:0] lz_hi;

// Finish module connections below.

clz_tri_tree #(wlo) clo( lz_lo, a[ ] );

clz_tri_tree #(wmi) cmi( lz_mi, a[ ] );

clz_tri_tree #(whi) chi( lz_hi, a[ ] );

// Finish nlz.
assign nlz =

end
endmodule
```
Problem 3: [27 pts] Appearing below are modules that test if two bit vectors are equal in some way.

(a) Show the hardware for the module below at the default size using basic gates: AND, OR, XOR, NOTs, and bubbled inputs and outputs. Do not use something like $==$.

```verilog
module eq #( int w = 4 )( output uwire equal, input uwire [w-1:0] a, b );
    assign equal = a == b;
endmodule
```

Show hardware using basic gates at default size.

(b) Show the cost and delay of the module in terms of $w$ (the value of parameter $w$) using the simple model.

In terms of $w$: Cost and Delay.
(c) The module below also tests equality but it does so after shifting the first operand. Show the hardware in terms of basic gates after optimization.

```verilog
module eqs #( int w = 6, int s = 2 ) ( output uwire equal, input uwire [w-1:0] a, b );
    localparam logic [w+s-1:0] zero = 0;
    assign equal = zero + ( a << s ) == b;
endmodule
```

Show hardware at default size after optimization.

---

(d) The module below performs a different operation than the one above. Explain the difference and show an example.

```verilog
module eqt #( int w = 16, int s = 5 ) ( output uwire equal, input uwire [w-1:0] a, b );
    assign equal = ( a << s ) == b;
endmodule
```

Difference between operation eqs and eqt.

Show a value for a and b for which the output of eqs and eqt are different.
Problem 4: [28 pts] Answer each question below.

(a) Appearing below is synthesis data taken from the solution to Homework 2. The Delay Target column shows the maximum delay constraint given to the synthesis program.

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Area</th>
<th>Delay</th>
<th>Delay Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>clz_w32</td>
<td>26290</td>
<td>3.110</td>
<td>10.000 ns</td>
</tr>
<tr>
<td>clz_tree_w32</td>
<td>21706</td>
<td>1.425</td>
<td>10.000 ns</td>
</tr>
<tr>
<td>clz_w32_1</td>
<td>36476</td>
<td>1.007</td>
<td>0.100 ns</td>
</tr>
<tr>
<td>clz_tree_w32_5</td>
<td>37356</td>
<td>0.577</td>
<td>0.100 ns</td>
</tr>
</tbody>
</table>

In general, which result should be used if the only goal were to minimize area, the results for the 10.0 ns Target or for the 0.1 ns Target? Explain.

(b) Provide w-bit declarations requested below.

```vhdl
// Declare each object to be w bits and consistent with its name.
//
ulw [ ] bit_zero_is_msb;
ulw [ ] bit_zero_is_lsb;
ulw [ ] bit_zero_is_middle;
```
(c) The module fragment below starts with six declarations (the object names starting with r), each providing a value (either a+b or x+y). Some of those declarations will result in compile errors. Identify them and explain the problem. If possible fix the problem without changing the object kind (localparam, uwire, var).

```verilog
module my_mod
  #( int w = 10, int x = 11, int y = 12 )
  ( input uwire [w:1] a, b );

  localparam logic [w:1] r1p = a + b;
  localparam logic [w:1] r2p = x + y;
  uwire [w:1] r1w = a + b;
  uwire [w:1] r2w = x + y;
  logic [w:1] r1l = a + b;
  logic [w:1] r2l = x + y;

  Indicate which ones are wrong and the reason that they are wrong.
  Indicate which can't be fixed and and explain why not.
```

(d) Explain what $realtobits does, and what hardware will be synthesized for it, if any.

```verilog
always_comb begin
  x = $realtobits(r);
end
```

Purpose of realtobits.

Synthesized hardware.