Digital Design using HDLs

EE 4755

Midterm Examination

Friday, 21 October 2016  12:30–13:20 CDT

Problem 1  _________  (20 pts)
Problem 2  _________  (20 pts)
Problem 3  _________  (20 pts)
Problem 4  _________  (10 pts)
Problem 5  _________  (10 pts)
Problem 6  _________  (20 pts)

Alias  __________________________

Exam Total  _________  (100 pts)

Good Luck!
Problem 1: [20 pts] Write a Verilog description of the hardware illustrated below. The description must include the modules and instantiations as illustrated. The description can be behavioral or structural, but it must be synthesizable.
Problem 2: [20 pts] Appearing below is the `lookup_elt` module from Homework 4 and following that an incomplete module named `match_amt_elt`. Complete `match_amt_elt` so that the value at output port `md` is set to the number of bits in `clook` that match corresponding bits in `celt`. For example, if `clook=5'b00111` and `celt=5'b00111` then `md` should be 5, if `clook=5'b00101` and `celt=5'b00111` then `md` should be 4, and if `clook=5'b11000` and `celt=5'b00111` then `md` should be 0. Code must be synthesizable, but can be behavioral or structural.

- Complete the module so that `md` is set to the number of matching bits.
- Make sure that `md` is declared with sufficient width.

```verilog
module lookup_elt #( int charsz = 32 ) // This module is for reference only.
( output logic match,
  input uwire [charsz-1:0] char_lookup, char_elt );
always_comb match = char_lookup == char_elt;
endmodule

module match_amt_elt
  #( int charsz = 32 )
  ( output logic md,
    input uwire [charsz-1:0] clook,
    input uwire [charsz-1:0] celt );
nendmodule
```
Problem 3: [20 pts] Show the hardware that will be synthesized for the modules below.

(a) Show the hardware that will be inferred for the module below. Show `acme_ip_sqrt` as a box.

```verilog
module vmag( output uwire [31:0] mag, input uwire signed [31:0] v [3] );

logic [63:0] sos;
acme_ip_sqrt #(32) s1(mag,sos);

always_comb begin
    sos = 0;
    for ( int i=0; i<3; i++ ) sos += v[i] * v[i];
end
endmodule
```

- Show inferred hardware.
- Don’t forget `acme_ip_sqrt`.
- Clearly show input and output ports of `vmag`. 
Problem 3, continued:

(b) Show the hardware that will be inferred for the module below, before and after optimization. *Note: In the original exam the input was named vi.*

```verilog
module min_elt( output logic [1:0] idx_min, input uwire signed [31:0] v [3] );
    always_comb begin
        idx_min = 0;
        for ( int i=1; i<3; i++ ) if ( v[i] < v[idx_min] ) idx_min = i;
    end
endmodule
```

☐ Show inferred hardware. ☐ Clearly show input and output ports.

☐ Show hardware after some optimization.
Problem 4: [10 pts] Appearing in this problem are several variations on a counter.

(a) Show the hardware inferred for each counter below.

```verilog
module ctr_a( output uwire [9:0] count, input clk );
    logic [9:0] last_count;
    assign count = last_count + 1;
    always_ff @(posedge clk ) last_count <= count;
endmodule

module ctr_b( output logic [9:0] count, input clk );
    uwire [9:0] next_count = count + 1;
    always_ff @(posedge clk ) count <= next_count;
endmodule
```

Inferred hardware for `ctr_a` and `ctr_b`.

(b) There is a big difference in the timing of the outputs of `ctr_a` and `ctr_b`. Explain the difference and illustrate with a timing diagram.

Difference between two modules. Timing Diagram.
Problem 5: [10 pts] Appearing below is the solution to the 2015 midterm exam Problem 2. Estimate the cost of this module as illustrated but use variable s for the number of bits in sum (shown as sswid) and in each a element (shown as parameter f). Assume that the cost of a BFA is 10 units and that the cost of an n-input AND and OR gate is n – 1 units. Take into account the 0 input to one of the multiplexors.

```
module ssum #( int n = 3, int f = 4, int sswid = f + $clog2(n) )
(output logic [sswid-1:0] sum,
  input uwire [n-1:0] mask,
  input uwire [f-1:0] a[n] );
always @*
begin
  sum = 0;
  for ( int i=0; i<n; i++ ) if ( mask[i] ) sum += a[i];
end
endmodule
```

Cost of illustrated hardware. Account for 0 mux input.
Problem 6: [20 pts] Answer each question below.

(a) Show the values of the variables as indicated below:

```verilog
module tryout();
  logic [15:0] a;
  logic [0:15] b;
  logic [3:0][3:0] e;
  logic [3:0] x1, x2;

  initial begin
    a = 16'h1234;
    x1 = a[3:0]; // Value of x1 is:
    b = 16'h1234;
    x2 = b[0:3]; // Value of x2 is:
    e = 16'h1234;
    e[0] = e[0] + 'hf; // Value of e is:
    e = 16'h1234;
    e[0][0] = e[0][0] + 'hf; // Value of e is:
  end
endmodule
```

(b) Describe something that can be done during elaboration that cannot be done during simulation, and something that can be done during simulation, that cannot be done during elaboration.

- Something that can be done during elaboration but **not during simulation** is:

- Something that can be done during simulation but **not during elaboration** is:
(c) Appearing below are two alternatives for an integer division module, *Plan A* and *Plan B*. Both are impractical, but Plan A is not even synthesizable.

```verilog
module div_plan_a #( int w = 16 ) ( output logic [w-1:0] quo, input uwire [w-1:0] a, b );
    always_comb begin
        quo = 0; a > quo * b; quo++;
    end
endmodule

module div_plan_b #( int w = 16 ) ( output logic [w-1:0] quo, input uwire [w-1:0] a, b );
    localparam int LIMIT = 1 << w;
    always_comb begin
        quo = 0;
        for ( int i=0; i<LIMIT; i++ ) if ( a < i * b ) quo++;
    end
endmodule
```

- Why isn’t Plan A synthesizable? Be specific as possible.

- What might be a practical objection to the Plan B approach?

(d) The *magfp* module below is not synthesizable due to the use of the *real* data type. How would the module need to be changed so that it would be synthesizable and would operate on floating-point values.

```verilog
module magfp( output real mag, input real vi [3] );
    real sos;
    sqrt #(32) s1(mag,sos);
    always_comb begin
        sos = 0;
        for ( int i=0; i<3; i++ ) sos += vi[i] * vi[i];
    end
endmodule
```

- Show changes to port declaration for synthesizability.

- Explain with a few examples how the rest of the code would need to be changed.