**LSU EE 4755** **Homework 8** **Due: 27 November 2018**

**Problem 1:** Appearing below is the output of the simulator and synthesis script, showing data for the Homework 7 solution modules. Modules are simulated and synthesized for $w = 32$.

```
Module Name                              Area  Period  Period
                                               Target  Actual
mult_seq_ds_prob_1_w32_m1              157813    1000   14926
mult_seq_ds_prob_1_w32_m2              185493    1000   15431
mult_seq_ds_prob_1_w32_m4              242568    1000   16296

mult_seq_d_prob_2_w32_m1               288580    1000   31944
mult_seq_d_prob_2_w32_m2               301203    1000   32204
mult_seq_d_prob_2_w32_m4               329226    1000   32192

For Prob 1 Deg 1       ran  400 tests,    0/   0/   0 errors found. Avg cyc 33.0
For Prob 1 Deg 2       ran  400 tests,    0/   0/   0 errors found. Avg cyc 17.0
For Prob 1 Deg 4       ran  400 tests,    0/   0/   0 errors found. Avg cyc 9.0

For Prob 2 Deg 1       ran  400 tests,    0/   0/   0 errors found. Avg cyc 9.5
For Prob 2 Deg 2       ran  400 tests,    0/   0/   0 errors found. Avg cyc 7.3
For Prob 2 Deg 4       ran  400 tests,    0/   0/   0 errors found. Avg cyc 5.0
Modules instantiated with w = 32.
```

The Problem 1 modules are based on the streamlined multiplier and so are faster. But the Problem 2 modules skip zeros. Based on the data above, indicate the ways, if any, that the Problem 2 modules are better than the Problem 1 modules. Explain using the numbers above.

*There are more problems on the next pages.*

**Problem 2:** Appearing below is a solution to Homework 7, Problem 1, the streamlined degree-$m$ multiplier with handshaking. The complete solution is at `https://www.ece.lsu.edu/koppel/v/2018/hw07-sol.v.html`. For this problem assume that $w$ and $m$ are both powers of 2.

```
module mult_seq_ds_prob_1 #( int w = 16, int m = 2 )
   ( output logic [2*w-1:0] prod,  output logic  out_avail,
     input uwire clk, in_valid,    input uwire [w-1:0] plier, cand );

   localparam int iterations = ( w + m - 1 ) / m;
   localparam int iter_lg = $clog2(iterations);

   uwire [iterations-1:0][m-1:0] cand_2d = cand;

   bit [iter_lg:0] iter;
   logic [2*w-1:0] accum;

   always_ff @( posedge clk ) begin

      if ( in_valid ) begin

         accum = cand;
         iter = 0;
         out_avail = 0;

      end else if ( !out_avail && iter == iterations ) begin

         out_avail = 1;
         prod = accum;

      end

      accum = { 0 + plier * accum[m-1:0] + accum[2*w-1:w], accum[w-1:m] };
      iter++;
   end

endmodule
```

(*a*) Show the hardware that will be inferred for this module. The Inkscape SVG format diagram of the hardware for the streamlined sequential module from the class demo notes can be used as a starting point. It is at `https://www.ece.lsu.edu/koppel/v/2018/ill-mul-seq-str.svg`.

(*b*) Compute the cost and delays for this module using the simple model. Show these in terms of $w$ and $m$. Clearly show the critical path on your diagram.

*There is a problem on the next page.*

**Problem 3:** Appearing below is a solution to Homework 7, Problem 2, the streamlined degree-$m$ multiplier with handshaking. The complete solution is at `https://www.ece.lsu.edu/koppel/v/2018/hw07-sol.v.html`. For this problem assume that $w$ and $m$ are both powers of 2.

```
module mult_seq_d_prob_2 #( int w = 16, int m = 2 )
   ( output logic [2*w-1:0] prod,   output logic out_avail,
     input uwire clk, in_valid,     input uwire [w-1:0] plier, cand );

   localparam int iterations = ( w + m - 1 ) / m;
   localparam int iter_lg = $clog2(iterations);

   uwire [iterations-1:0][m-1:0] cand_2d = cand;

   bit [iter_lg-1:0] iter;
   logic [2*w-1:0] accum;

   always_ff @( posedge clk ) begin

      logic [iter_lg-1:0] next_iter;

      if ( in_valid ) begin
         iter = 0;
         accum = 0;
         out_avail = 0;
      end else if ( !out_avail && iter == 0 ) begin
         prod = accum;
         out_avail = 1;
      end

      accum += plier * cand_2d[iter] << ( iter * m );

      next_iter = 0;
      for ( int i=iterations-1;  i>0;  i-- )
        if ( i>iter && cand_2d[i] ) next_iter = i;
      iter = next_iter;
   end

endmodule
```

($a$) Show the hardware that will be inferred for this module.

($b$) Compute the cost and delays for this module using the simple model. Show these in terms of $w$ and $m$. Clearly show the critical path on your diagram.