Problem 1: The homework Verilog file, hw05.v, contains something similar to the streamlined multiplier presented in class, mult_seq_stream, and even more streamlined versions of the multiplier, mult_seq_stream_2, and mult_seq_stream_3. These modules are reproduced at the end of this assignment. For an HTML version visit http://www.ece.lsu.edu/koppel/v/2015/hw05.v.html. See the 2014 midterm exam for similar problems.

(a) Show the hardware that will be synthesized for each module for the default parameters. Show the module after optimization.

(b) Estimate the clock frequency of each module based on the following assumptions:
   - Latch delay: 10 units. Multiplexor latency: 2 units. Latency of a n-bit adder: $5\lceil \lg n \rceil$ units.
   - Latency of an n-input gate: $\lceil \lg n \rceil$ units.

(c) Why would module mult_seq_stream_3 provide a result in less time than the other two, even assuming that the clock frequency for all the modules was the same?
module mult_seq_stream #( int wid = 16 )
(output logic [2*wid-1:0] prod,
 input logic [wid-1:0] plier,
 input logic [wid-1:0] cand,
 input clk);

localparam int wlog = $clog2(wid);

logic [wlog-1:0] pos;
logic [2*wid-1:0] accum;

always @( posedge clk ) begin

logic [wid:0] pp;

if ( pos == 0 ) begin

prod = accum;
accum = cand;
pos = wid - 1;
end else begin

pos--;
end

// Note: the multiplicand is in the lower bits of the accumulator.
// pp = accum[0] ? { 1'b0, plier } : 0;

// Add on the partial product and shift the accumulator.
// accum = { { 1'b0, accum[2*wid-1:wid] } + pp, accum[wid-1:1] };

end
endmodule
module mult_seq_stream_2 #( int wid = 16 )
  ( output logic [2*wid-1:0] prod,
    input logic [wid-1:0] plier,
    input logic [wid-1:0] cand,
    input clk);

  localparam int wlog = $clog2(wid);

  logic [wlog-1:0] pos;
  logic [2*wid-1:0] accum;

  always @( posedge clk ) begin
    if ( pos == 0 ) begin
      prod = accum;
      accum = { 1'b0, cand[0] ? plier : wid'(0), cand[wid-1:1] };
      pos = wid - 1;
    end else begin
      logic [wid:0] pp;

      // Note: the multiplicand is in the lower bits of the accumulator.
      // pp = accum[0] ? plier : 0;

      // Add on the partial product and shift the accumulator.
      // accum = { { 1'b0, accum[2*wid-1:wid] } + pp, accum[wid-1:1] };

      pos--; end
  end
endmodule
module mult_seq_stream_3 #( int wid = 16 )
( output logic [2*wid-1:0] prod,
  input logic [wid-1:0] plier,
  input logic [wid-1:0] cand,
  input clk);

localparam int wlog = $clog2(wid);

logic [wlog-1:0] pos;
logic [2*wid-1:0] accum;

always @( posedge clk ) begin
  if ( pos == 0 ) begin
    accum = { 1'b0, cand[0] ? plier : wid'(0), cand[wid-1:1] };
    pos = wid - 1;
  end else begin
    logic [wid:0] pp;

    // Note: the multiplicand is in the lower bits of the accumulator.
    //
    pp = accum[0] ? plier : 0;

    // Add on the partial product and shift the accumulator.
    //
    accum = { { 1'b0, accum[2*wid-1:wid] } + pp, accum[wid-1:1] };

    if ( pos == 1 ) prod = accum;
    pos--;
  end
end
endmodule