The next problem is part of your final design project; you should work on this part of the problem set with your design-project partner.

2. You need to design the shifter for your final design project. Your shifter needs to support left logical shifts of 0 to 7 bits using an encoded control line shift_amount<0:2>. I would recommend a logarithmic shifter implementation using three-levels of 2-to-1 multiplexers (implemented with two nFETs). You should only have to layout the multiplexer once and reuse this cell 24 times. Watch your bit pitch. Remember that this should match your adder design; ultimately everything in the datapath needs a matched bit pitch.

To complete your shifter design, I expect:

- Sized schematics
- Bit-stack layout of your shifter design (remember to begin by making a tiling diagram and then stick layouts of your cells)
- HSPICE results for the delay of the critical path
- IRSIM runs to verify functionality
next_state <= s1;
end if;
when s2 =>
  out2 <= '1';
next_state <= s3;
when s3 =>
  if con2 = '0' then
    next_state <= s3;
  elsif con3 = '0' then
    out1 <= '0';
    next_state <= s2;
  else
    next_state <= s1;
  end if;
end case;
end process state_logic;

state_register: process (clk) is
begin
  if (clk = '1' and not(clk'stable)) then
    state <= next_state;
  end if;
end process state_register;
end architecture rtl;

Implement this finite-state machine as a PLA and a (four-bit) flip-flop. Present your results as (hand-drawn) schematics. Don’t worry about sizing or simulating your design. To design the PLA, use espresso for two-level logic optimization. You can assess the espresso man pages on the input file format by typing:

man -s 5 espresso

Additional documentation on espresso can also be found on the main espresso man pages:

man espresso
1. In this problem, I want you to consider the implementation of a controller using PLAs. We will take advantage of the tool %espresse% to optimize our logic.

Consider a controller defined by the following VHDL description:

```vhdl
architecture rtl of controller is
  subtype state_type is std_ulogic_vector(0 to 3);
  constant s0: state_type := "0001";
  constant s1: state_type := "0010";
  constant s2: state_type := "0100";
  constant s3: state_type := "1000";
  signal state, next_state: state_type;
  signal con1, con2, con3: std_ulogic;
  signal out1, out2: std_ulogic;
  signal clk;

begin
  state_logic: process(state, con1, con2, con3) is
    begin
      out1 <= '0';
      out2 <= '0';
      case state is
        when s0 =>
          out1 <= '0';
          out2 <= '0';
          next_state <= s1;
        when s1 =>
          out1 <= '1';
          if con1 = '1' then
            next_state <= s2;
        else
```