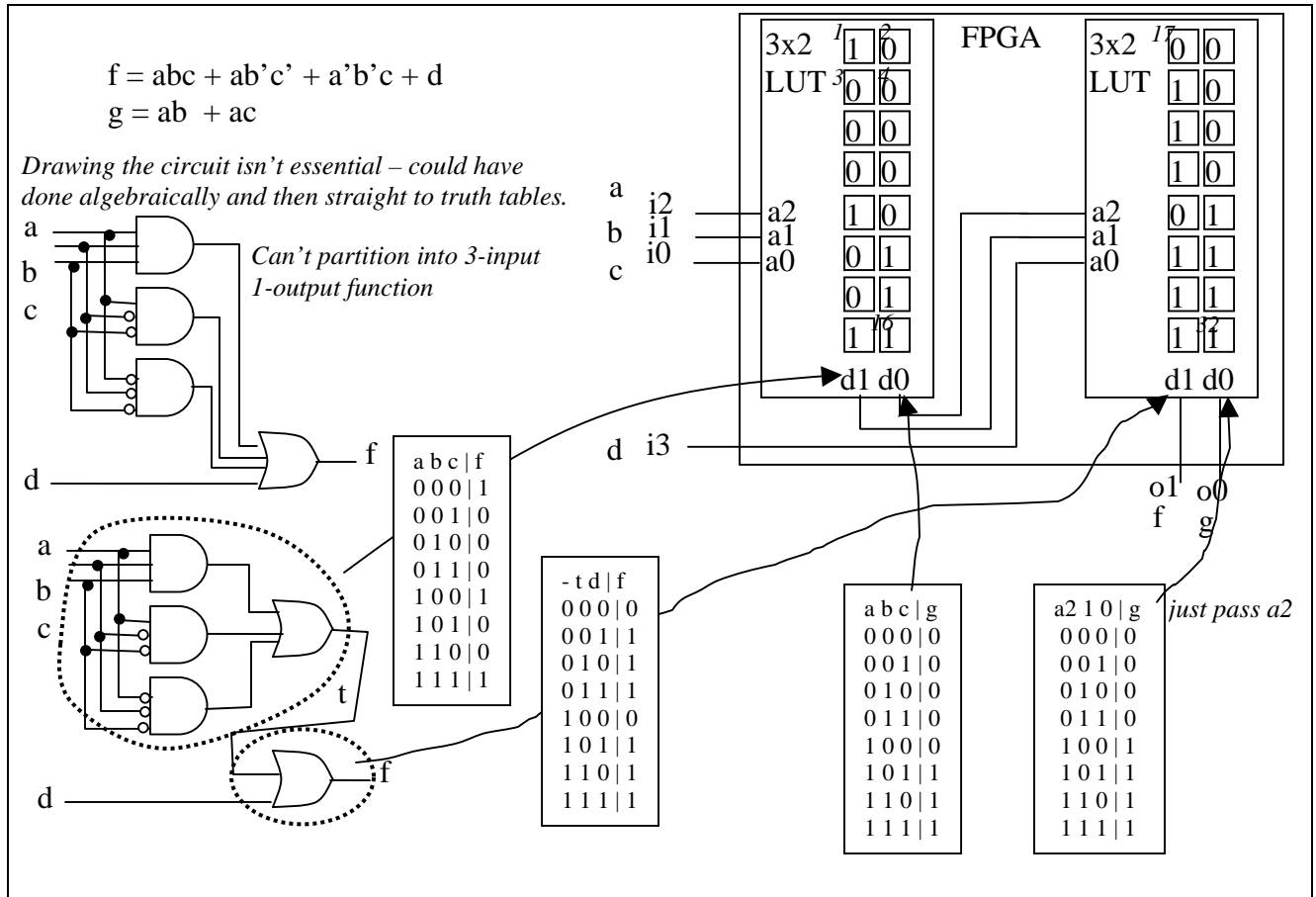


1. (a) Show how to map the following functions to the FPGA structure. If a bit is a don't care, program it as an "X." Show all work, including any intermediate truth tables or circuits, for full credit.

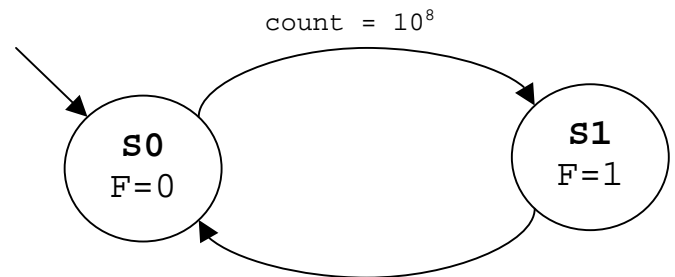


- (b) Show the bitfile that would be shifted into the FPGA to program the FPGA for that circuit. We have labeled some of the lookup table bits to make clear how they are connected as one logical shift register (known as a "scan chain"). Your bit file should consist of one row of 32 bits, with the rightmost bit representing lookup table bit 32, and the leftmost bit representing bit 1.

Bitfile: 1000000010010111 0010101001111111

(Turn page over for problem 2)

2. Draw an FSM and capture the FSM in VHDL code to toggle an FPGA output “F” every one second – one second high, one second low, forever. The FPGA has an input “clk” connected to a 100 MHz clock. Your VHDL code should be as close as possible to being synthesizable by Xilinx tools (e.g., do not use “wait for” statements).



```

architecture bhv of toggler is
  type states is (S0, S1);
  signal current_state: states := S0;
  signal next_state: states := S1;
  signal count: integer := 0;
begin
  process (clk)
  begin
    if (clk = '1' and clk'event) then
      current_state <= next_state;
    end if;
  end process;

  process (clk)
  if (clk = '1' and clk'event) then
    if (current_state = next_state) then
      count <= count + 1;
    else
      count <= 0;
    end if;
  end if;
end process;

  process (current_state, count)
  begin
    case current_state is
    when S0 =>
      F <= '0';
      if (count = 100000000) then
        next_state <= S1;
      else
        next_state <= S0;
      end if;
    when S1 =>
      F <= '1';
      if (count = 100000000) then
        next_state <= S0;
      else
        next_state <= S1;
      end if;
    end case;
  end process;
end bhv;

```

END QUIZ