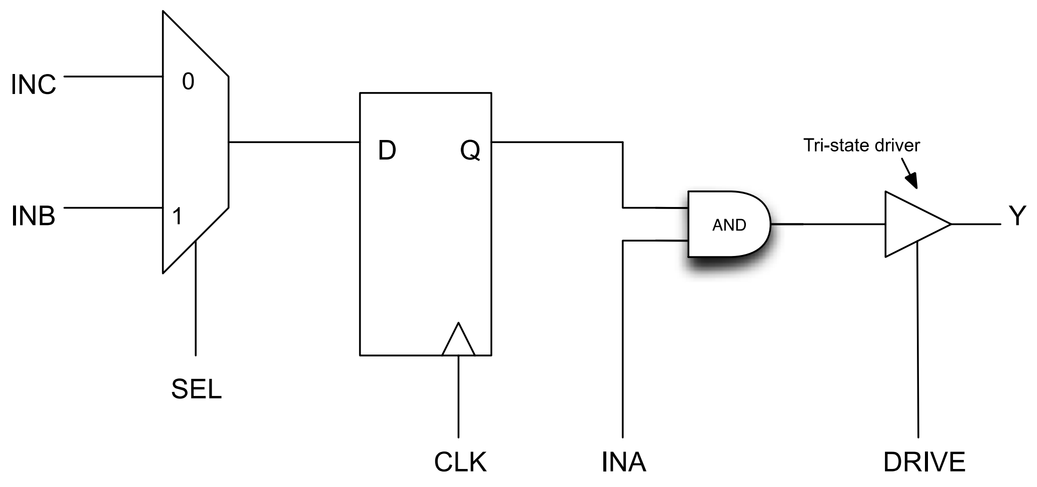
1. Write synthesizable VHDL code that has the same behaviour as the following circuit. Your code should consist of one entity containing as few processes as possible. Marks will be deducted for solutions which are overly long or complex.



2. Draw a schematic of a circuit that has the same behaviour as the following VHDL code. You can use flip-flops, basic gates, multiplexers, and/or tri-state buffers in your diagram

```
entity mycirc is
        port (  EN, B, C: in std_logic;
                Y : out std_logic_vector(1 downto 0));
end mycirc;
architecture defn1 of mycirc is
signal  F : std_logic_vector(2 downto 0);
begin
       Y <= F(1) & F(0);
       process(B)
       begin
          if (B = '1') then
             for i in 1 downto 0 loop
                  F(i) <= F(i+1);
             end loop;
             F(2) <= C xor F(1) xor F(0);
          end if;
        end process;
  end defn1;
```

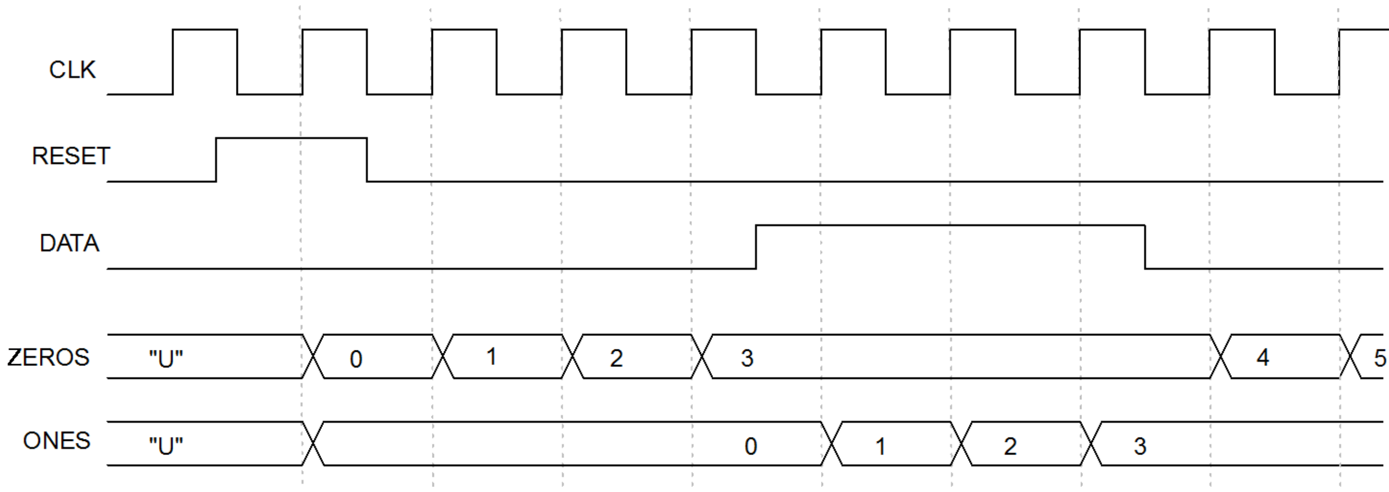
3. Draw a schematic of a circuit that has the same behaviour as the following VHDL code. You can use flip-flops, basic gates, multiplexers, and/or tri-state buffers in your diagram. On the surface, this looks almost the same as the previous question, however, it is a bit more tricky. Be sure you understand it.

```
entity mycirc is
        port ( A, B, D, E : in bit;
        F : out bit );
end mycirc;

architecture defn1 of  mycirc is
   signal W, Y : bit;
begin
        process(E)
        begin
                if (E='1') then
                    case A is
                        when '0' => W <= B;
                        when '1' => W <= not B;
                    end case;
                   F <= A and Y;
                end if;
        end process;

        process (W,D)
        begin
                Y <= D and W;
        end process;
end defn1;
```

4. Consider the following circuit. The inputs are CLK, RESET, and DATA (each one bit wide). The outputs are ZEROS and ONES, each 8 bits wide. The purpose of the circuit is to receive a sequence of one-bit data elements, one per clock cycle, and counts how many of these elements are 0, and how many are 1.

The circuit has a synchronous RESET. More precisely, on each rising clock edge, if RESET is 1, the outputs "ZEROS" and "ONES" are set to 0. On a rising clock edge, If RESET is 0, then if "DATA" is a 0, the output ZEROS is increased by 1, and if "DATA" is 1, the output ONES is increased by 1.

The following timing diagram shows an example of the operation of this circuit ("U" means "uninitialized"). Both outputs are uninitialized at the start of the simulation. At the second rising clock edge, RESET is high, so both outputs are set to 0. At the third rising clock edge, the input DATA is 0, so ZEROS is increased by 1. The same thing happens on the 4th and 5th rising clock edge. On the 6th rising clock edge, DATA is 1 so ONES is increased (and ZEROS maintains its value).



Write synthesizable VHDL code to specify the behaviour of this circuit. Your description should consist of as few processes as possible. Marks will be deducted for solutions which are overly long or complex. *Your code must be synthesizable*