Tecnología de Computadores

Práctica 2

UNIVERSIDAD NEBRIJA

1. INTRODUCCIÓN

En esta segunda sesión de laboratorio se estudiarán circuitos secuenciales sencillos para entender el funcionamiento del reloj en la FPGA, así como máquinas de estado de Moore y de Mealy.

Una vez terminada la práctica el alumno será capaz de:

- Distinguir entre constantes y señales en VHDL.
- Crear contadores en VHDL.
- Diseñar circuitos secuenciales síncronos en VHDL.
- Diseñar máquinas de estado síncronas en VHDL.

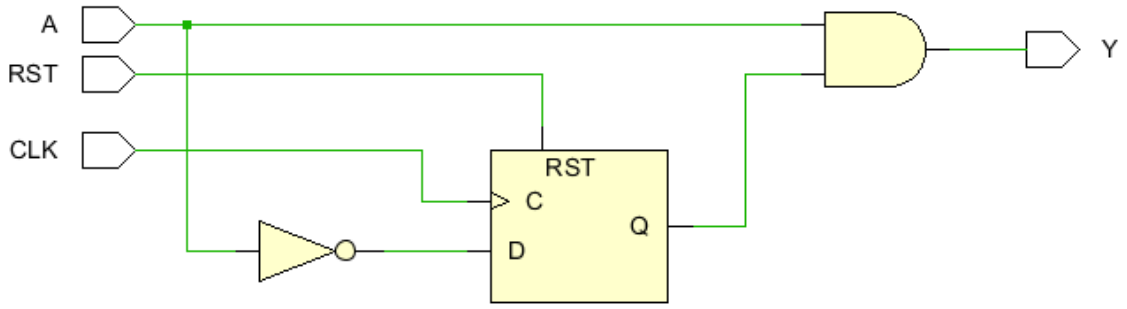
2. CIRCUITOS SECUENCIALES

Comience la práctica descargando los códigos "*blink.vhd*" y "*constraints.xdc*" del Campus Virtual y cree un nuevo proyecto en Vivado. Añada estos dos ficheros al proyecto creado e implemente el diseño en la FPGA para verificar su funcionamiento.

Una vez entendido el propósito del circuito responda a las siguientes cuestiones:

a) ¿Qué haría el circuito si la constante BLINK_FREQ pasase a valer 5?
b) ¿Por qué es necesaria la señal AUX_SIG?
c) ¿Por qué "*LED <= AUX_SIG;*" no está dentro del *process*?

A continuación, observe el siguiente circuito y deduzca su funcionamiento.



Flip-flop D

Diseñe el circuito anterior tomando como referencia/ayuda el código "*blink.vhd*". Para comprobar el funcionamiento del circuito, modifique el fichero de constraints conectando la señal de reset al botón BTNC (pin N17), la señal A al interruptor SW0 (pin J15) y la salida Y al LED0 (pin H17).

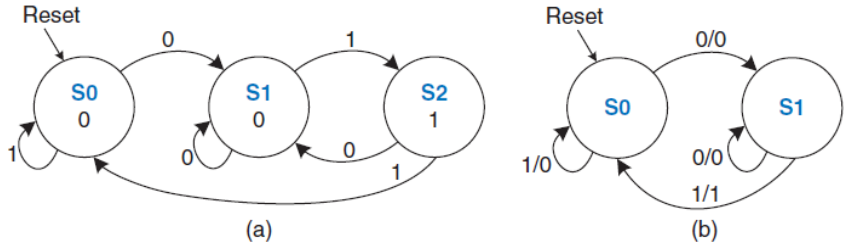
Implemente el diseño en la FPGA y pruebe a mover el interruptor SW0. ¿Observa algún cambio en el LED0? ¿A qué cree que es debido este funcionamiento? Piense dónde puede estar el problema y comente con el profesor de laboratorio lo que propone para solucionarlo.

Implemente su solución y muéstrele el correcto funcionamiento del circuito al profesor.

3. MÁQUINAS DE ESTADO FINITO

Las máquinas de estado finito (FSM) son circuitos secuenciales utilizados en muchos sistemas de control. En esta parte de la práctica se pretende que el alumno se familiarice con la representación de las máquinas de estado de Moore y de Mealy.

Para empezar, observe los dos diagramas de estado presentados a continuación:



A la izquierda (a) se muestra el diagrama de una máquina de estados de Moore (la salida únicamente depende del estado actual), mientras que a la derecha (b) se muestra el diagrama de una máquina de estados de Mealy (la salida depende del estado actual y del valor de la entrada). Ambos diagramas de pueden describir en VHDL de la siguiente forma:

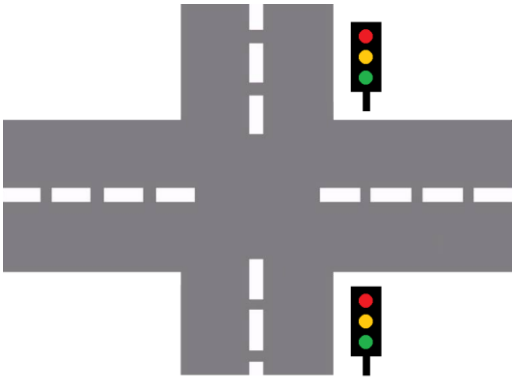
```
library IEEE; use IEEE.STD_LOGIC_1164.all;

entity patternMoore is
  port(clk, reset: in  STD_LOGIC;
       a:          in  STD_LOGIC;
       y:          out STD_LOGIC);
end;

architecture synth of patternMoore is
  type statetype is (S0, S1, S2);
  signal state, nextstate: statetype;
begin
  -- state register
  process(clk, reset) begin
    if reset then                state <= S0;
    elsif rising_edge(clk) then state <= nextstate;
    end if;
  end process;

  -- next state logic
  process(all) begin
    case state is
      when S0 =>
        if a then nextstate <= S0;
        else      nextstate <= S1;
        end if;
      when S1 =>
        if a then nextstate <= S2;
        else      nextstate <= S1;
        end if;
      when S2 =>
        if a then nextstate <= S0;
        else      nextstate <= S1;
        end if;
      when others =>
                  nextstate <= S0;
    end case;
  end process;

  --output logic
  y <= '1' when state = S2 else '0';
end;
```

```
library IEEE; use IEEE.STD_LOGIC_1164.all;

entity patternMealy is
  port(clk, reset: in  STD_LOGIC;
       a:          in  STD_LOGIC;
       y:          out STD_LOGIC);
end;

architecture synth of patternMealy is
  type statetype is (S0, S1);
  signal state, nextstate: statetype;
begin
  -- state register
  process(clk, reset) begin
    if reset then                state <= S0;
    elsif rising_edge(clk) then state <= nextstate;
    end if;
  end process;

  -- next state logic
  process(all) begin
    case state is
      when S0 =>
        if a then nextstate <= S0;
        else      nextstate <= S1;
        end if;
      when S1 =>
        if a then nextstate <= S0;
        else      nextstate <= S1;
        end if;
      when others =>
                  nextstate <= S0;
    end case;
  end process;

  -- output logic
  y <= '1' when (a = '1' and state = S1) else '0';
end;
```

Observe ambos códigos y responda a las siguientes cuestiones:

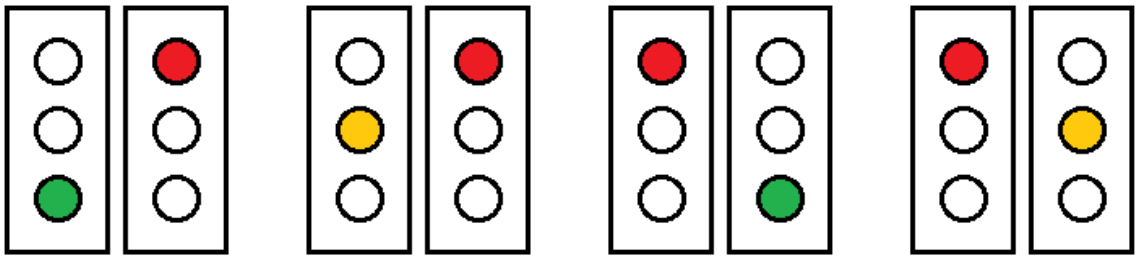
a) Ambos códigos describen el mismo funcionamiento de un sistema, ¿cuál es este funcionamiento?
b) ¿Cuáles son las principales diferencias entre ambos códigos?
c) ¿Cuál de los dos códigos cree que producirá un circuito con menor número de recursos? ¿Y cuál generará el circuito más rápido?

Para terminar la práctica, se le propone al alumno que realice un sistema de control de un cruce con semáforos como el de la siguiente imagen:



El sistema de semáforos controla dos calles, una calle vertical con más tráfico que la calle horizontal que la cruza. Debido a esto, el semáforo de abajo permanece más tiempo en verde que el otro. La descripción del sistema es la siguiente:

· Se comienza en el estado 1, transcurridos 10 segundos se pasa al estado 2.
· Estando en el estado 2, se pasa al estado 3 transcurridos 2 segundos.
· Estando en el estado 3, se pasa al estado 4 transcurridos 5 segundos.
· Estando en el estado 4, se pasa al estado 1 transcurridos 2 segundos.



Dibuje el diagrama de estados y, ayudándose de los códigos de Moore/Mealy presentados antes, escriba el código VHDL que controla este sistema de semáforos. Mapee las 6 luces de los dos semáforos a 6 LEDs cualesquiera de la FPGA e implemente el diseño. Muestre el correcto funcionamiento del sistema al profesor.