Sistemas de Tempo Real: FPGAs
2013/14

Práctica 6: Sistema de adelantamiento inteligente

Objetivo

Desarrollar un sistema de control guiado por una máquina de estados. Inicialmente se explicará cómo se puede implementar fácilmente una máquina de estados genérica y a continuación se expondrán los requerimientos del sistema de control a desarrollar.

Creación de Máquinas de Estado en VHDL

Es muy habitual hacer uso de máquinas de estados finitos (FSMs, *Finite-State Machines*) para la implementación de sistemas de control robustos y autómatas. Entre los distintos tipos de máquinas de estado, probablemente las más conocidas sean las de Moore y las de Mealy. En esta práctica vamos a tratar la implementación de máquinas de Moore, cuyas salidas están determinadas únicamente por su estado actual (la salida de las de Mealy dependen también de la entrada). En la Figura 1 se puede ver un ejemplo sencillo de una máquina de Moore de cuatro estados entre los que conmuta en función de una entrada binaria.

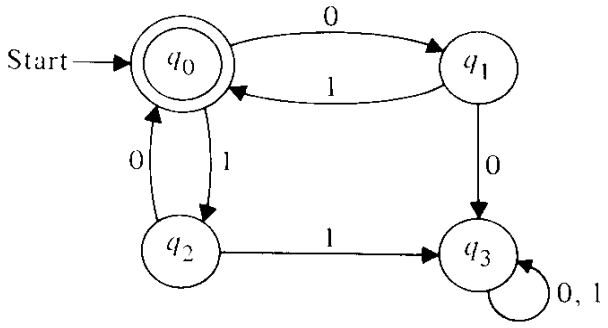

Figura 1: Ejemplo de máquina binaria de cuatro estados.

Para implementar en una FPGA la máquina de estados de la Figura 1, se procedería de la siguiente manera:

- (1) Se definen las entradas y las salidas del sistema:

```
entity maquina is
port(  entrada:      in std_logic;
           clk:      in std_logic;
         start:      in std_logic;
        salida:      out std_logic);
end maquina;
```

- (2) Dentro de la arquitectura, se comienza por crear un tipo de dato enumerado que contenga todos los estados. Asimismo, se declaran dos señales que representarán al estado actual y al siguiente:

```
type estados is (q0, q1, q2, q3);
signal siguiente_estado, estado_actual: estados;
```

- (3) A continuación se define un primer proceso que controla el inicio de la máquina de estados y la conmutación entre el estado actual y el siguiente:

```
actualizacion_estado: process(clk, start)
begin

  if (start='1') then
        estado_actual <= q0;
  elsif (clk'event and clk='1') then
        estado_actual <= siguiente_estado;
  end if;

end process;
```

- (4) Por último, en un segundo proceso, se define el comportamiento de la máquina al conmutar entre los distintos estados. En este caso, dado que en la Figura 1 no está especificado, vamos a suponer que como salida se generan siempre ceros al conmutar a q0, q1 y q2, y 1 al conmutar a q3.

```
logica_de_control: process(estado_actual, entrada)
begin

  case estado_actual is

   when q0 => salida <= '0';
     if entrada='0' then
         siguiente_estado <= q1;
     elsif entrada='1' then
         siguiente_estado <= q2;
     end if;

   when q1 => salida <= '0';
     if entrada='0' then
         siguiente_estado <= q3;
     elsif entrada='1' then
         siguiente_estado <= q0;
     end if;

   when q2 => salida <= '0';
     if entrada='0' then
         siguiente_estado <= q0;
     elsif entrada='1' then
         siguiente_estado <= q3;
     end if;

   when q3 => salida <= '1';
     if entrada='0' then
         siguiente_estado <= q3;
     elsif entrada='1' then
         siguiente_estado <= q3;
     end if;

   when others =>
     salida <= '0';
     siguiente_estado <= q0;

  end case;

end process;
```

Práctica a realizar

Cada año suceden en las carreteras numerosos accidentes de tráfico debido a maniobras imprudentes. Una de las más comunes son los adelantamientos temerarios que se realizan en zonas prohibidas o cuando no se verifica por el retrovisor si hay vehículos precedentes que puedan ocasionar una colisión.

En esta práctica, vamos a suponer que poseemos un vehículo que tiene instalado un sistema de cámaras conectadas a una FPGA. Este sistema entra en funcionamiento sólo cuando el conductor utiliza los intermitentes al señalizar un adelantamiento, evitando así su uso en casos en los que se precisa realizar maniobras rápidas para evitar colisiones.

Una de las cámaras estaría situada bajo un retrovisor exterior y permitiría saber si existen o no vehículos en el carril adyacente, bien sea porque se aproximan a nuestro vehículo por dicho carril, o bien porque circulan por el ángulo muerto. En el caso de que existiese posibilidad de colisión, el sistema prevendría al conductor emitiendo un pitido.

Por otro lado, una segunda cámara estaría colocada en la parte frontal del vehículo y serviría para detectar si las líneas del carril son continuas o discontinuas (se puede ver un ejemplo de aplicación real en la Figura 1). Si el conductor intenta realizar el adelantamiento en una zona con líneas continuas, el sistema intentará evitar la maniobra, indicando al sistema de control central (por ejemplo, a través de la ECM o la ECU) que debe incrementarse la resistencia de la dirección.

Figura 1: Sistema de detección de carriles con cámaras CCD[1].

Para modelar este sistema de control de adelantamientos se pide:

1) Diseñar el diagrama de estados de la lógica de control del sistema. Debe tenerse en cuenta que va a haber una señal inicial **INTERMITENTE_ENCENDIDO** que indica que se va realizar un adelantamiento. Además, se considerará como estado inicial el estado **OBTENER_IMAGENES** y la información suministrada por cada cámara se representará por un único bit (carril adyacente libre/ocupado y líneas continuas/discontinuas). Cada estado estará asociado a una variable binaria distinta que indicará si se lleva a cabo la acción o no.
2) Crear un proyecto nuevo en ISE con un módulo de VHDL (**p6**) en donde se implementará la máquina de estados.
3) Crear un **Test Bench** con un reloj a 25 MHz que suministrará los estímulos necesarios para simular las cuatro posibles situaciones que pueden darse al combinar la información resultante de ambas cámaras.

[1] Fuente: J. M. Armingol et al.,"IVVI: Intelligent Vehicle based on Visual Information", *Robotics and Autonomous Systems*, Volume 55, Issue 12, Dec. 2007, pp. 904-916.

4) Redactar una breve memoria explicativa sobre el diseño realizado incluyendo una ilustración del diagrama de estados implementado. Enviar esta memoria por correo electrónico junto con una copia comprimida del directorio del proyecto a tiago.fernandez@udc.es .