Diseño de un sistema de parking mediante VHDL

Luca Scalambrin
Laboratorio III - Electrónica digital
Ing. en Telecomunicaciones - Instituto Balseiro, UNCuyo, CNEA, Argentina

8 de septiembre de 2019

En el presente trabajo se llevó a cabo el diseño y programación de un sistema de parking para un estacionamiento, capaz de contabilizar la ocupación de autos mediante una dupla de foto sensores. Para ello se dió uso del lenguaje VHDL y para el testing se utilizó una placa de desarrollo de FPGA Nexys 3 - Spartan 6, la cual permitió simular los sensores mediante switches.

1. Especificación

Se deberá diseñar e implementar un contador de ocupación para un estacionamiento, el cual cuenta con una única entrada/salida. Esto permitirá que mediante una única dupla de foto sensores, se pueda llevar a cabo dicho conteo. Para ello se dispondrá de una configuración se sensores como la que se observa en la FIg. 1 , los cuales definirán una serie de estados que permitirán identificar ingresos y egresos de automóviles al estacionamiento.

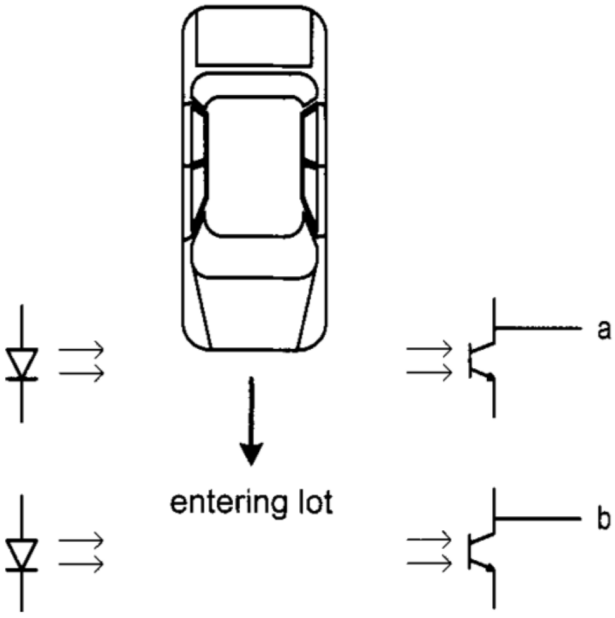

Figura 1: Esquema de sensores empleado para poder identificar egresos e ingresos de autos al estacionamiento.

Además, se deberá mostrar el conteo de autos en el display *BCD* siete segmentos de la propia placa, cuyo máximo será el número en hexadecimal $FFFF$. Para efectuar dicha muestra en el display será necesario dar uso del clock de la placa, de modo de que se puedan utilizar los cuatro módulos del display en simultaneo.

2. Diseño del módulo top

En la Fig. 2 se observa el diagrama del módulo TOP, el cual se obtuvo mediante el entorno de desarrollo empleado. Este cuenta con cuatro módulos, los cuales corresponden a cada una de las secciones que se necesitan llevar a cabo en el sistema de parking. Las entradas del TOP son cuatro, el clock, el bit de reset y las dos lecturas de los sensores, a y b. Cuando el sensor detecta un automóvil, el valor de dicha señal toma el valor "1", caso contrario, tendrá el valor "0".

Las salidas del módulo son cuatro, *min* y *max*, los cuales son bits que indican cuando el estacionamiento se encuentra vacío o lleno, $cod(7:0)$, siendo esta la salida que posee la codificación correspondiente al contador y $an(3:0)$ el selector de los displays siete segmentos.

3. Diseño, implementación y testing de cada módulo

A continuación se detallan cada uno de los módulos internos del módulo top. Tanto el código VHDL como el testbench correspondiente a cada uno de estos módulos se adjuntan con el presente informe. Además, en el repositorio digital [1] se

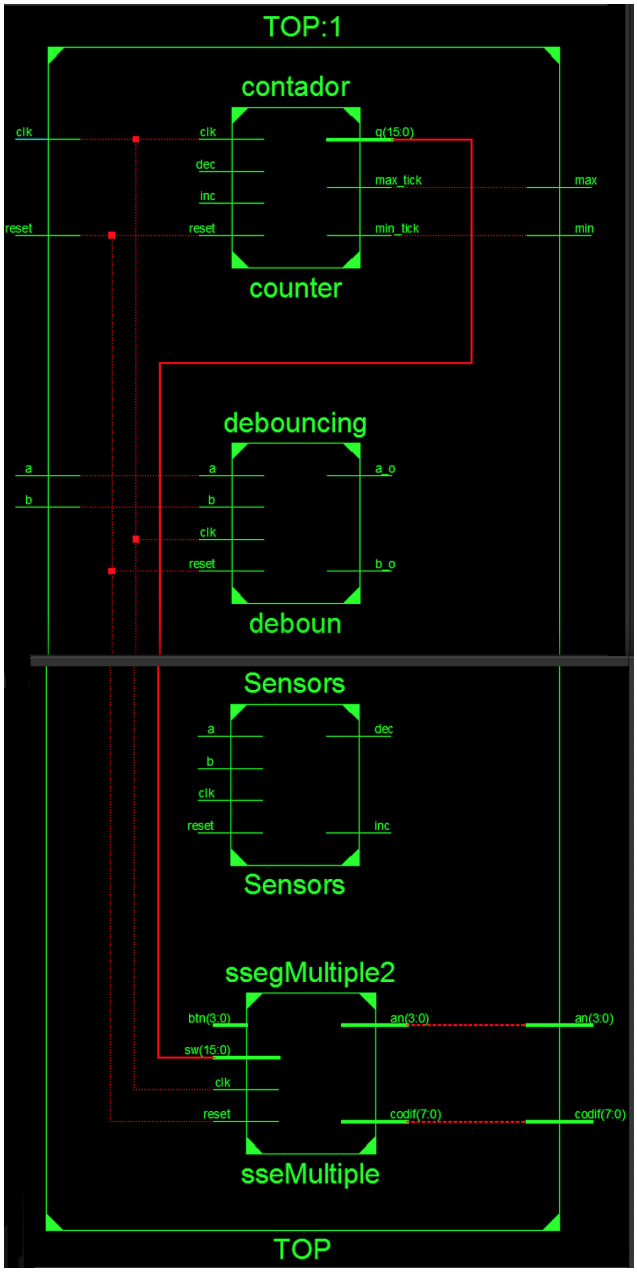

Figura 2: Diagrama en bloques del móludo top del sistema de parking. Se observan cada uno de los cuatro módulos empleados, estos son, el correspondiente al contador, el encargado de los sensores, el módulo capaz de mostrar los múltiples dígitos y por último el bloque encargado de estabilizar las señales de entrada, provenientes de los switches.

podrán encontrar todos los códigos correspondientes.

3.1. Módulo contador

El módulo contador posee cuatro entradas, estas son, el bit del clock y del reset, la señal de incremento y de decremento. El número máximo hasta el que podrá contar es el correspondiente a 2^{16}, encontrándose dicho número en la salida $q(15:0)$. Este número se incrementará cuando $inc = "1"$ y $dec = "0"$, decrementará cuando $dec = "1"$ e $inc = "0"$ y permanecerá constante en cualquier otro caso. Cuando se alcance el número máximo se encenderá la señal max_tick y cuando se alcance el mínimo ($"0000"$) se encenderá min_tick. Cabe aclarar que este código fue reutilizado, de modo que se debió realizar un módulo denominado "*adaptation*", el cual se encarga de setear en 1 o 0 la señales *en* y *up*, de modo de llevar a cabo el incremento y decremento de forma correcta.

Para el testing del módulo se realizaron incrementos y decrementos, corroborando el valor del contador, como también las señales de *max* y *min*. El código del testeo posee el nombre *testing_contador* y el del módulo es *contador*, los cuales se pueden consultar en el respositorio digital [1].

3.2. Módulo debouncing

El módulo debouncing se encarga de asegurar la estabilidad de la señal de entrada, correspondiente a los switches de la placa. Este es necesario debido a que los switches poseen una composición mecánica y por lo tanto, existirá un transitorio de la señal al colocarse en alguna de sus posiciones. Para ello, la lógica del módulo asegurará la estabilidad de la señal cuando esta permanezca estable durante al menos 20 ms.

3.3. Módulo sensores("*Sensors*")

En la Fig. 3 se observa la máquina de estados diseñada para los sensores. Dado que existe una única secuencia posible que permite el ingreso y egreso de un automóvil al estacionamiento, se dio uso de una máquina de finitos estados para modelar la situación. En este caso se dio uso de una máquina de Moore compuesta de nueve estados. El estado inicial corresponde a la situación en que no existe ningún obstáculo entre los sensores, esto es, no hay ningún automóvil ingresando o egresando del parking. Los demás estados representan cada una de las posibles combinaciones que se podrán dar en los sensores.

Las salidas del módulo son *inc* y *dec*, las cuales se activarán cuando ingrese o egrese un auto, respectivamente.

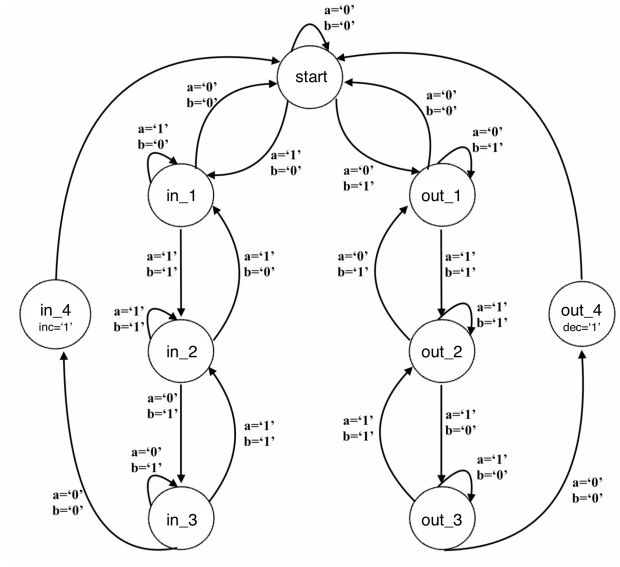

Figura 3: Diagrama de la máquina de estados finitos empleada para en el módulo de sensores.

Para llevar a cabo el testeo se introdujeron las secuencias correspondientes al ingreso y egreso de autos, corroborándose que las señales de *inc* y *dec* se activen cuando corresponda. El código correspondiente es *testing_sensor*, y junto al código del módulo se pueden consultar en el repositorio digital [1].

3.4. Módulo siete segmentos múltiple ("*ssegMultiple2*")

El módulo *ssegMultiple2* posee cuatro entradas y dos salidas. Una de sus entradas, $btn(3:0)$, se encuentra siempre en '1111', con el objetivo de que siempre estén habilitados los cuatro displays. Esto es debido a que se reutilizó un código ya desarrollado. Sus otras entradas son $sw(15:0)$, la cual indica el número a mostrar, el bit del clock y el de reset.

Este módulo se encarga de, una vez recibidos los cuatro números a mostrar(todos incluidos en la señal *sw*), generar la codificación del siete segmentos y a su vez, seleccionar cada uno de los displays mediante la señal $an(3:0)$. El cambio entre cada uno de los cuatro displays se efectúa cada 4 ms. El módulo encargado de contabilizar los 4 ms y generar la codificación es *disp_mux*, el cual es instanciado en el módulo *ssegMultiple2*, junto a los 4 flip-flops utilizados.

4. Test de integración total

Una vez completos todos los módulos, se instanciaron en el top, de modo de poder testear es sistema de parking por completo. Para ello, se llevó a cabo un mapeo entre los botones y switches de la placa a utilizar. En la Fig. 4 se pueden observar dichos mapeos. Los switches $sw00$ y $sw01$ se asignaron a los sensores a y b respectivamente, por medio del módulo encargado del *debouncing*, para poder asegurar estabilidad en dicha señal. Luego, se asignó el "$led00$" a la señal min para indicar que el estacionamiento se encuentra vacío y el "$led01$" para indicar que este se encuentra lleno. Por último se llevó a cabo el mapeo de los números codificados $codif(7:0)$ con los respectivos leds del display de siete segmentos.

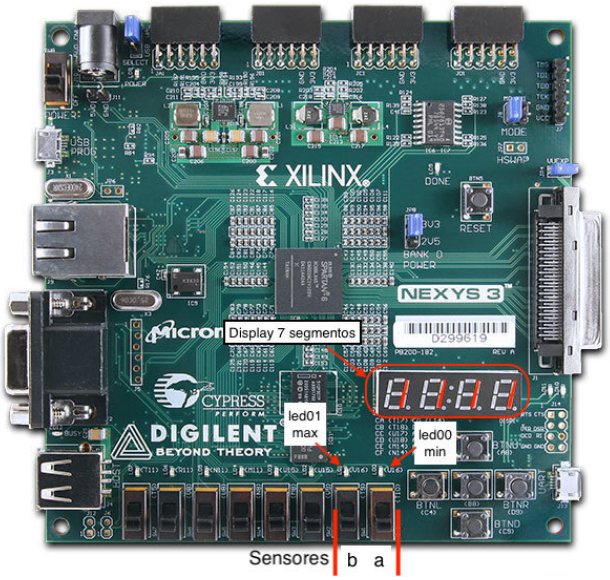

Figura 4: Esquema de la placa de desarrollo utilizada y los respectivos *switches* y *leds* mapeados.

Todos estos mapeos se encuentran realizados en el archivo "Nexys3_Master.ucf".

5. Conclusiones

Mediante la utilización del lenguaje $VHDL$ y conceptos de diseño y programación modular se pudo implementar el sistema de parking que cumple con la especificación. Para ello fue muy útil la utilización de módulos ya realizados, como el del contador y el multiplexor del display. Además, mediante la metodología de diseño sincrónico y la implementación de una máquina de estados fue

posible llevar a cabo el modelo de los sensores, para luego ser implementado mediante dos switches de la placa. Por último, fue importante considerar las fluctuaciones que existen en las señales debido a las teclas mecánicas empleadas; lo cual fue solucionado mediante el debouncing.

Las pruebas fueron realizadas en una placa de desarrollo de FPGA Nexys 3 - Spartan 6.

Todos los códigos desarrollados se pueden consultar en el repositorio digital de GitHub titulado "Sistema de parking con VHDL" [1].

Referencias

[1] Scalambrin Luca. *"Laboratorio III - Sistema de parking con VHDL".* https://github.com/LucaScalam/Sistema-de-parking-con-VHDL.git