TRABAJO DE LABORATORIO III: Playa de Estacionamiento

Electrónica Digital - Laboratorio III

N. A. Pizarro

1 DESCRIPCIÓN DEL FUNCIONAMIENTO

Se implementó un contador de ocupación de una playa de estacionamiento en el software `VIVADO` a partir del lenguaje `VHDL`. Contiene una única salida `q` donde indica la cantidad de autos dentro de la playa. Los autos ingresan y egresan del mismo por una única entrada que contiene dos sensores `A` y `B`, siendo estas las entradas de dato al contador. Además cuenta con una entrada para resetear la cuenta denominada `clr` y una entrada de reloj `clk`. Los sensores `A` y `B` constan de un fototransmisor y un fotoreceptor que están dispuesto según la Figura 1.1, cuando hay un objeto entre medio de salida tenemos un '1' lógico y de lo contrario un '0' lógico.

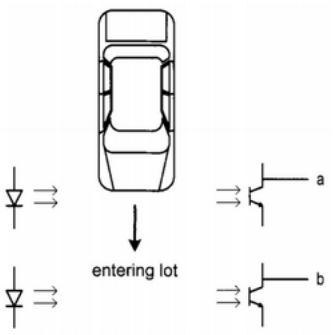

Figure 1.1: Ubicación de los fotosensores ubicado en la entrada/salida de la playa de estacionamiento.

El proyecto consta de dos bloques principales, *contador* y *FSM* (del inglés: máquina de estado finita). El bloque del *contador* se reutilizó del trabajo Práctico de Laboratorio II: un contador genérico universal, al que se le sacó la opción de cargar `load` y el reseteo `clr` se hizo asíncrono.

Por otro lado la *FSM* es una máquina de estado finita cuyo objetivo es, a través de sus dos salidas y de las entradas `A` y `B`, indicar a través de sus salidas `en` y `up` si ingresó o egresó un auto para poder llevar la cuenta de la cantidad de autos dentro de la playa. Se tuvieron en cuenta los casos dónde el auto intentaba entrar/salir pero retrocedía, si la playa de estacionamiento estaba llena (i.e. el contador estaba en el número máximo de la cuenta) no dejaba ingresarlo y si está vacía, ningún auto puede salir. La maquina de estado a implementar es de tipo *Mealy*, esto implica que los cambios en las salidas se producirán en las transiciones de estados.

¿Se podría haber implementado la carga de un valor inicial con `load`? Se me ocurrió hacer la carga de un valor cuando no este ni egresando ni ingresando ningún auto. Actualmente no está desarrollada esta opción pero vendrá incluida en la versión premium del programa.

¿Qué pasa si esta ingresando un auto pero uno de los sensores falla y eso produce un fallo en la cuenta? Si falla uno de los sensores es probable que no se cuente el ingreso o egreso del auto.

¿Si un auto es incinerado en medio de la playa de estacionamiento o es teletransportado a otra dimensión, se descontará de la cantidad de autos dentro de la playa de estacionamiento? No.

¿Se podría haber implementado con una FSM tipo *Moore*? Seguramente, pero simplemente vino a mi mente la idea de hacerlo tipo *Mealy*.

¿Cuenta otro tipo de vehículos que no sean autos? Depende, si es más grande que un auto seguro, y sino dependerá de que el largo total del vehículo sea mayor a la distancia entre sensores. Si no cuenta, es gratis.

¿Podría haber tenido el bloque *FSM* menos salidas? Tenemos tres estados posibles: contar auto, descontar auto o no contar nada, así que como mínimo necesitamos 2 bits.

¿Si el auto pasa muy rápido lo cuenta? Todo dependerá del tiempo del clock, si la frecuencia es de por ejemplo 1 kHz (un valor relativamente bajo para un clock), y los sensores están a una distancia de 1 m uno del otro, para que no detecte el auto éste debería pasar a más de 900 km/h y el auto más rápido del mundo alcanza los 509 km/h[1], así que la velocidad no es un problema actual.

[1] https://tinly.co/fUdL2

¿Y si pasan dos autos pegados cuenta uno solo? Si, pero no creo que los dueños de los autos quieran hacer esto, el arreglo de pintura y posible abolladuras es bastante más caro.

¿Tiene nombre la playa de estacionamiento? Se llama Pedroestación.

2 Diagrama en bloque

El diagrama en bloque del contador de ocupación se puede observar en la figura 2.1, donde se encuentran los dos bloques principales: *c_FSM* y *c_contador*. Las señales intermedias tienen el mismo nombre de la entrada pero anteponiendo "s_". Las señales involucradas son: `s_min_tick`,`s_max_tick`, `s_up` y `s_en`.

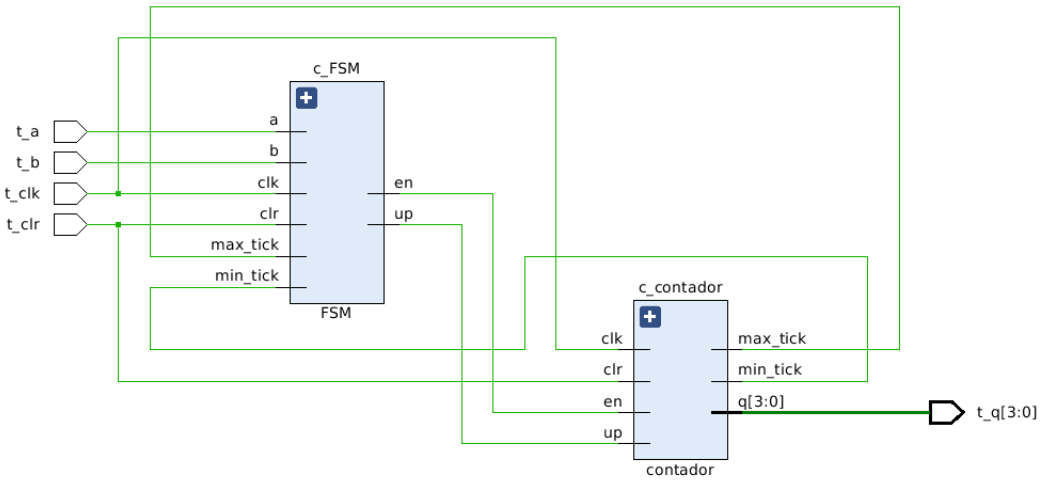

Figure 2.1: Esquema en bloques del contador de ocupación implementado.

3 Diagrama de estados

El diagrama de estado implementado es de tipo *Mealy* y se puede observar en la Figura 3.1. En dicho esquema para no sobrecargarlo se obvió el estado de las salidas `en` y `up`, en todas las transiciones donde no estén especificadas, ambas salidas toman el valor '0'.

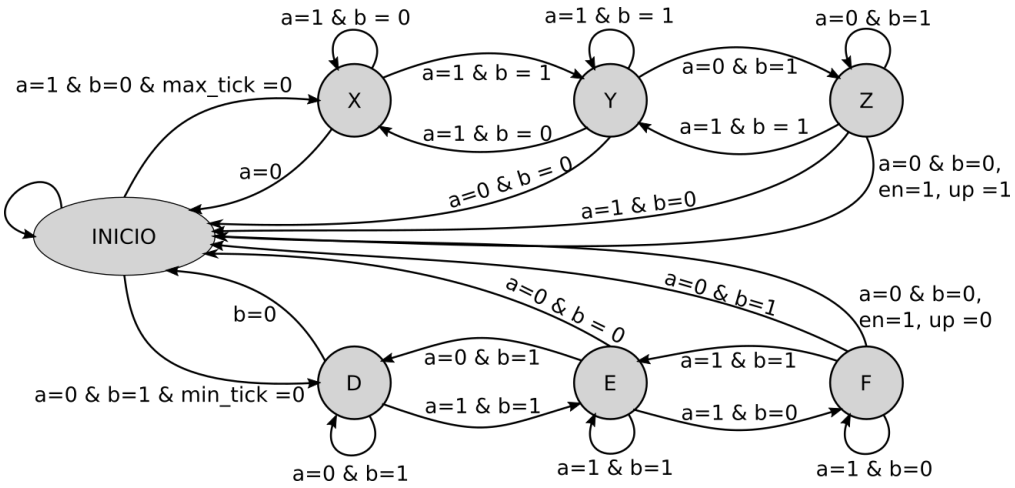

Figure 3.1: Diagrama de estado del bloque *FSM*.

4 Código

Los códigos se encuentran disponibles y descargables en el repositorio: https://github.com/AgustinaP/Pedrostacion[2].

5 Verificación comportamental

Para la corroboración del correcto funcionamiento se realizaron varios test para probas sus funcionalidades. Se implementó con el contador de 0 a 7 bits.
Primero, se resetearon las salidas y se verificó que se quede en el estado `INICIO` si un auto quiere salir ya que la cuenta es mínima, se ingresaron dos autos: el primero normal y el segundo en un punto retrocede un poco y sigue su camino. Se hace lo mismo con un auto que egresa. Seguidamente se prueban combinaciones que harían que vuelva al estado `INICIO`.

Dichos tests se simularon en el software `VIVADO` de forma exitosa. La simulación completa está disponible y descargable en el repositorio mencionado anteriormente. En la Figura 5.1 se observa parte de esta simulación.

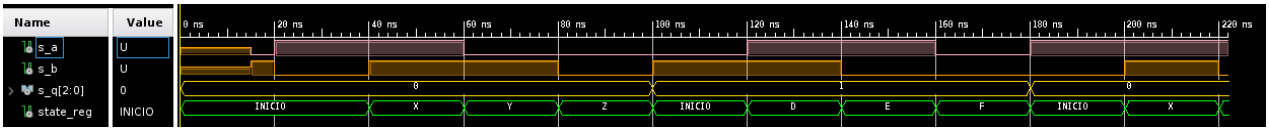

Figure 5.1: Simulación comportamental del contador de ocupación.

[2]Por cualquier inconveniente mandar un mail a nadia.pizarro@ib.edu.ar