**SISTEMAS ELECTRÓNICOS DIGITALES**

**Cruce de semáforos**

**Introducción**

En este trabajo programaremos en lenguaje VHDL un cruce de semáforos y lo implementaremos en una FPGA Spartan3 y un módulo externo que simula mediante botones y LEDs el cruce.

El cruce será la intersección entre una carretera principal y una secundaria, cada una de ellos con su correspondiente paso de peatones, Para dificultar el enunciado inicial, lo hemos complementado con la adición de un tren en el mismo cruce, que tendrá completa prioridad ante el paso de los coches.

Cambiamos el uso previsto de los 4 botones del módulo externo. Dos de los botones serán los que indiquen la presencia de peatones que quieren cruzar, otro botón indicará que hay un coche en la vía secundaria y el último botón, del que nos interesa si su estado es alto o bajo, indicará cuando, o cuando no, hay un tren en el cruce.

Por la adición del tren, las dos luces auxiliares del módulo externo las usaremos para identificar el estado del tren, si éste está llegando, se está yendo o si no hay tren.

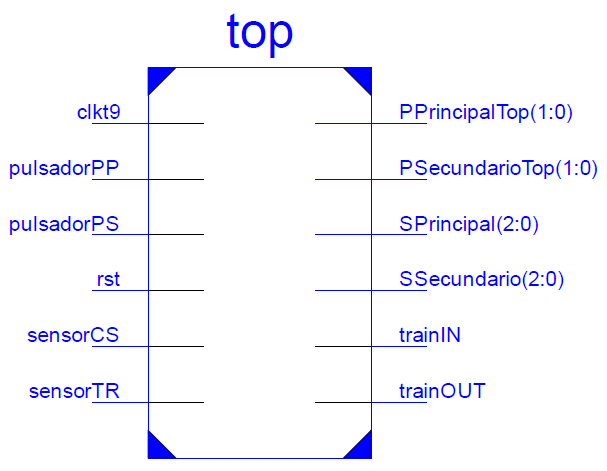
Debido a que la FPGA funciona a 3,3V mientras que el módulo de semáforos externo funciona a 5V, hemos optado por cambiar los buffer del módulo externo por unos de tecnología TTL CMOS disponibles en el laboratorio. Estos buffer incluyen puertas NOT, por lo que para la programación en VHDL hemos tenido que optar por usar lógica negativa, es decir, el valor 0 como valor lógico alto.

Para la realización del trabajo hemos optado por usar un repositorio en github para facilitar el trabajo individual. El repositorio se puede encontrar en el siguiente enlace <https://github.com/APerezG95/Semaforos-SED>

**Top**

En la entidad Top, la más genérica de todas, agrupamos todos los componentes necesarios para la realización del proyecto. Asignamos las entradas y salidas de cada componente, además del valor de los genéricos necesarios para el correcto funcionamiento.

El esquemático de las entradas y salidas de esta entidad es el siguiente.



Al final de la memoria añadiremos un esquemático interno de la entidad.

La entidad Top está conformada por componentes de las siguientes identidades:

**Máquina de estados**

Para este proyecto usaremos un total de 13 estados. 6 de esos estados se podrían considerar principales, mientras que el resto son estados auxiliares usados para resolver la problemática de contabilizar el tiempo a partir de un estímulo externo y para generar un flanco en el reset de los contadores entre dos estados temporizados.

Estos estados principales son *S0, S1, S2, S3, T1 y T2*.

El estado *S0* corresponde al tránsito "normal" de la vía, en el que los coches de la vía principal podrán seguir su camino de forma usual.

El estado *S1* corresponde a un estado de transición entre el *S0* y el *S2*, en el que el semáforo principal pasa a ser ámbar.

El estado *S2* corresponde al estado en el que los vehículos del camino secundario tienen preferencia.

El estado *S3* corresponde a un estado de transición entre el *S2* y el *S0*, en el que el semáforo secundario pasa a ser ámbar.

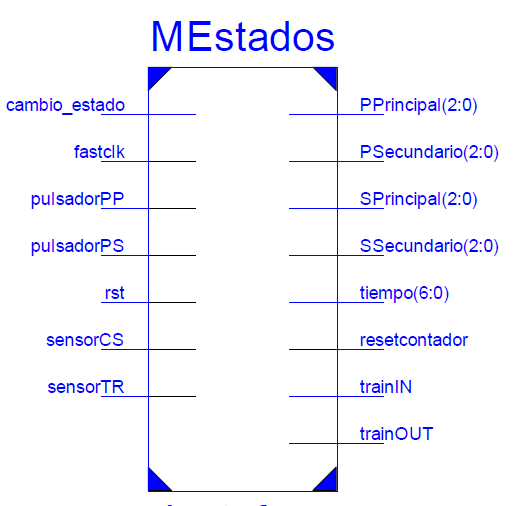
El estado *T1* corresponde a la llegada y paso del tren. Ningún vehículo podrá moverse en este momento, aunque los peatones podrán cruzar con seguridad.

El estado *T2* corresponde a un estado de transición entre *T1* y el estado *S0*.

Resumimos los estados en la siguiente tabla, en la que las siglas *SVP* corresponde a *Semáforo Vía Principal*, *SVS* corresponde a *Semáforo Vía Secundaria*, *PPVP* a *Paso Peatones de la Vía Principal*, *PPVS* *a Paso Peatones de la Vía Secundaria* y *P* a *Parpadeando.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Estado S0 | Estado S1 | Estado S2 | Estado S3 | Estado T1 | Estado T2 |
| SVP | Verde | Ámbar | Rojo | Rojo | Rojo | Rojo |
| SVS | Rojo | Rojo | Verde | Ámbar | Rojo | Rojo |
| PPVP | Rojo | Rojo | Verde | Verde P | Verde | Verde P |
| PPVS | Verde | Verde P | Rojo | Rojo | Verde | Verde |
| Tren IN | Nivel Bajo | Nivel Bajo | Nivel Bajo | Nivel Bajo | Nivel Alto | Nivel Bajo |
| Tren OUT | Nivel Bajo | Nivel Bajo | Nivel Bajo | Nivel Bajo | Nivel Bajo | Nivel Alto |

El esquemático de la entidad de la máquina de estados es el siguiente



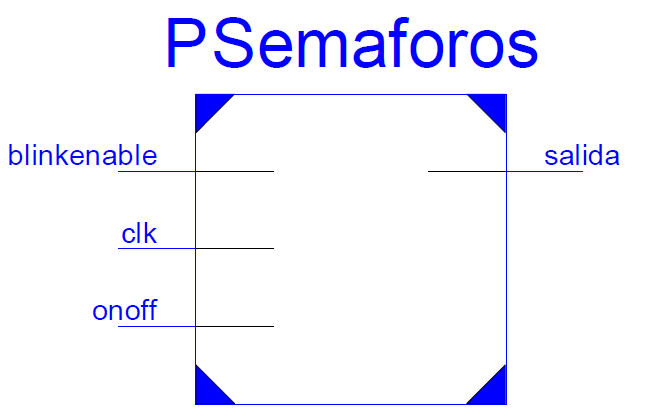
**Parpadeo de semáforos**

Para lograr el parpadeo de los semáforos hemos optado por crear una entidad que reciba como entrada un reloj, una señal de encendido/apagado y si el parpadeo está activo o no.

La salida de esta entidad será directamente el estado del semáforo, es decir, de esta entidad obtendremos una señal a nivel alto, bajo o parpadeante en función de las entradas.

Usaremos dos componentes de esta entidad, una para cada paso de peatones.

El esquemático es el siguiente



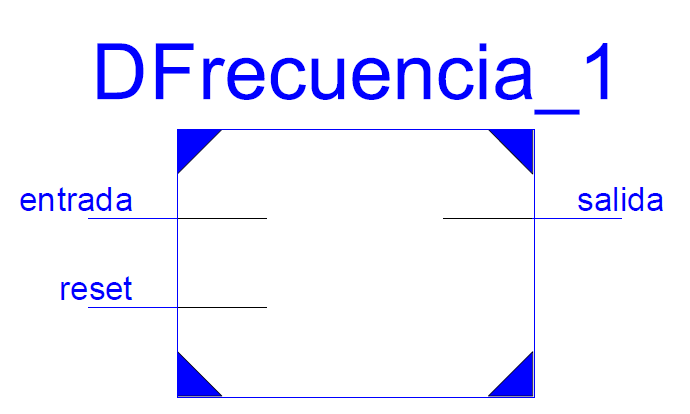
**Divisor de frecuencias**

Como la FPGA tiene un reloj interno, llamado *fastclk*, demasiado rápido (50MHz) hemos optado por un divisor de frecuencia para obtener una señal de reloj, *clock*, más adecuada.

El funcionamiento se basa en aumentar una unidad a una señal de tipo entero llamada *contador* a cada ciclo del *fastclk* hasta llegar a un *dato*, que se trata de un genérico de tipo entero. Al llegar al dato introducido, se niega la salida de la entidad y se resetea el *contador*. Así logramos conseguir un *clock* de la frecuencia que queramos, según el *dato*.

Usamos dos divisores de frecuencia en nuestro programa, uno para lograr un reloj de 1Hz más adecuado para nuestros fines, y otro de 0,6Hz para el parpadeo de los semáforos. La frecuencia de estos relojes puede ser variada a través del *dato* asignado en la entidad Top.

El esquemático es el siguiente



**Contador**

Esta entidad se encarga de poner su salida en valor lógico alto cuando ha pasado el tiempo indicado.

Tiene tres entradas, reset, un tiempo y el reloj. Dispone de una única salida que se activa cuando se ha llegado al tiempo recibido en la entrada.

Esta entidad ha causado muchos problemas debido a la necesidad de dos procesos, uno que calcule el tiempo que ha pasado y otro que active, cuando proceda, la salida. Debido a la imposibilidad de sintetizar el programa si asignas un valor a una señal en dos procesos distintos o mezclando flancos de subida con flancos de bajada, nos vimos obligados a cambiar el código original, totalmente funcional en el *testbench*.

El esquemático es el siguiente

