BLOQUE TEMÁTICO 1

TÍTULO DE LA ACTIVIDAD: **Análisis del módulo gen_SCL**	**CÓDIGO:** **BT1.A3_P4**

FECHA:			
NOMBRE:		APELLIDOS:	

MODALIDAD:	Libre. Resolución de problemas.	TIPO:	Presencial	DURACIÓN:	60 minutos

CALENDARIO:	Sesión 9	REQUISITOS:	

CRITERIO DE ÉXITO:	

COMENTARIOS E INCIDENCIAS:

TIEMPO DEDICADO:	minutos	AUTOEVALUACIÓN: [entre 0 y 10 puntos]	No procede

Introducción

En esta actividad se analizará y simulará una realización del modelo VHDL sintetizable del módulo que genera la señal SCL del bus I2C (gen_SCL) acorde con la especificación de tiempos de la interfaz master I2C desarrollada en actividades anteriores. El código incluye el modelado de salidas en colector abierto.

Modelo VHDL sintetizable del genrador de SCL

Descargue de Moodle el fichero BT1_A3_Z2.zip y descomprímalo en una carpeta. Podrá comprobar que, debajo de la carpeta *BT1_A3_Z2* hay dos carpetas (*hdl* y *modelsim*) que contienen, respectivamente, un modelo sintetizable del módulo y un test-bench. Debe crear un proyecto en la carpeta modelsim y añadir ambos ficheros a dicho proyecto. Compile ambos ficheros.

Análisis del código

En primer lugar, debe abrir el fichero *gen_SCL.vhd* empleando el editor de texto de ModelSim. Observe que las primeras líneas del fichero son comentarios que documentan la funcionalidad del módulo y su interfaz. En base a la información contenida en dichos comentarios, conteste a las siguientes cuestiones.

1. ¿Qué señal habilita la generación de SCL?

2. En la primera parte del ciclo de generación de un periodo de SCL, esta señal está a nivel alto. ¿Por qué?

3. ¿Qué condiciones determinan el instante de activación de las señales *ena_out_SDA* y *ena_in_SDA*?

4. ¿Para qué sirven las salidas *ena_stop_i2c* y *ena_start_i2c*?

5. ¿En qué instante de tiempo se genera la señal *SCL_up*?

El núcleo del módulo es un contador (*cnt_SCL*). El proceso que lo modela controla también el estado de un flip-flop (*start*). En base al estado de cuenta de *cnt_SCL* se generan las salidas de temporización del módulo. Analice el código para contestar las siguientes cuestiones.

1. ¿Cuál es el módulo de cuenta de *cnt_SCL*?

2. El contador modelado mediante *cnt_SCL* cuenta mientras *ena_SCL* vale '1'. Sabiendo que *ena_SCL* se desactiva tras el flanco de reloj en que *SCL_up* vale '1', explique cómo evoluciona la cuenta a partir de ese momento y deduzca el porqué de este funcionamiento. Indique, además, cuando se para la cuenta, en qué estado quedan *cnt_SCL* y *start* y por qué.

3. Analice el valor que toma *start* durante los ciclos de cuenta de *cnt_SCL* y observe el uso que se hace de esta señal en las sentencias concurrentes para deducir su utilidad y explicarla aquí.

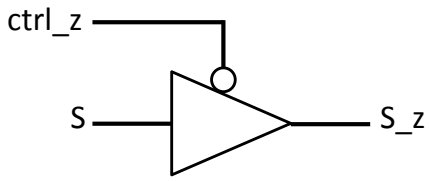
La señal *n_ctrl_SCL* tiene la forma (periodo y duración de los niveles alto y bajo) que se desea dar al reloj del bus I2C (salida *SCL*). Pero no puede ser la señal de salida para el pin de la FPGA conectado a la línea SCL, porque dicha salida debe comportarse como una salida en colector abierto (a nivel alto la salida debe estar en alta impedancia y a nivel bajo debe forzar un '0' "fuerte", como una salida de tensión con baja impedancia). El modelado de dicha salida se hace de la siguiente manera:

a) Se modela una señal interna (*n_ctrl_SCL*) que se comporte como la salida en colector abierto.

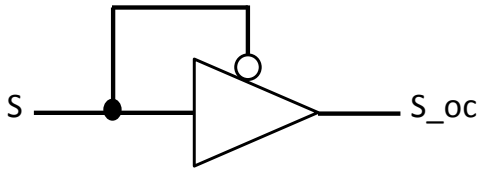
b) En VHDL, una salida con control de alta impedancia, *S_z*, puede modelarse de la siguiente manera:

```
S_z <= S when ctrl_z = '0' else
       'Z';
```

La salida *S_z* toma el valor de S ('0' ó '1') mientras la entrada de control (*ctrl_z*) esté activa –a nivel bajo, como en el ejemplo, que es lo típico-; cuando la entrada control se desactiva (*ctrl_z = '1'*), la salida se pone en alta impedancia. Un sintetizador lógico interpreta este fragmento de código como el modelo sintetizable de un *buffer* con salida en alta impedancia:



c) Una salida en colector abierto se puede modelar en base al modelo del *buffer* con salida en alta impedancia, utilizando como señal de control la entrada del *buffer*:




```
S_oc <= S when S = '0' else
        'Z';
```

Compruebe que el modelo de la salida *SCL* corresponde a una salida en colector abierto. Justifique su comprobación.

La salida *SDA_out* del módulo *reg_out_SDA*, cuyo modelo completó en la actividad BT1_A3_P3, también debe ser una salida en colector abierto. Modifique el modelo de *reg_out_SDA* para que sea así.

Simulación del modelo

EL fichero *test_gen_SCL* codifica un conjunto de pruebas que permiten verificar el funcionamiento del modelo *gen_SCL*. Abra el fichero con el editor de ModelSim. El test consiste en la generación de tres secuencias de generación de SCL con 18 pulsos cada una.

Debajo de la sentencia de emplazamiento del modelo, hay una sentencia de asignación que modela la resistencia de *pull-up* que se conecta a las salidas en colector abierto:

```
-- Resistencia de pull_up
  SCL <= 'H';
```

Como bien sabe, su función es establecer el nivel de tensión correspondiente al nivel lógico alto de la línea:

a) cuando una salida en colector abierto está a nivel bajo, fuerza una tensión baja (cercana a 0 V) con una salida que tiene una impedancia mucho menor que el valor de la resistencia de *pull-up*; por tanto, la salida tiene "más fuerza" para poner un '0' que la resistencia de *pull-up* (que intenta establecer "débilmente" la tensión de alimentación en el nodo de salida) –se dice que la salida pone un '0' fuerte y la resistencia de *pull-up* un '1' débil-;

b) cuando una salida en colector abierto está a nivel alto, se "desconecta" del nodo de salida, es decir, presenta una altísima impedancia de salida que no se opone a que la resistencia de *pull-up* establezca un nivel alto en el nodo de salida.

El modelado de la resistencia de *pull-up* se basa en que la salida *SCL* es de tipo *std_logic* y en que este tipo dispone de una función de resolución capaz de determinar el valor que efectivamente toma un objeto de este tipo (como *SCL*), cuando le asigna valor más de un *driver* (más de un proceso o sentencia concurrente). La función de resolución funciona de la siguiente manera:

a) La función considera que los 9 valores del tipo *std_logic* se clasifican en tres grupos, atendiendo a la fuerza con que establecen un nivel lógico (o metalógico); por orden de mayor a menor fuerza, en:
    a. Valores fuertes: 'U', '0', '1', 'X' y '-'
    b. Valores débiles: 'H', 'L' y 'W' –este último valor significa indeterminación débil-
    c. Valor muy débil: 'Z'
b) Cuando al objeto se le asignan simultáneamente dos o más valores contradictorios ('0' y '1', por ejemplo) con la misma fuerza, la función asigna al objeto el valor 'X' (indeterminación)

c) Cuando al objeto se le asignan simultáneamente dos o más valores contradictorios ('L' y 'Z', por ejemplo) pero con distinta fuerza, la función asigna al objeto el valor el valor con mayor fuerza, 'L' (nivel bajo débil)

Por tanto, al modelar la resistencia de *pull-up* como una sentencia de asignación (un *driver*) de 'H' a *SCL*, lo que ocurre es que:

a) Si la salida del módulo *gen_SCL* asigna el valor '0' a la salida *SCL*, la función de resolución se encuentra con una doble asignación simultánea a la señal *SCL*, '0' y 'H', y asigna a *SCL* el valor '0'.

b) Si la salida del módulo *gen_SCL* asigna el valor ‘Z’ a la salida *SCL*, la función de resolución se encuentra con una doble asignación simultánea a la señal *SCL*, ‘Z’ y ‘H’, y asigna a *SCL* el valor ‘H’.

En definitiva, la señal *SCL* alternará entre dos valores, ‘0’ y ‘H’, que representan un nivel lógico bajo y alto.

Revise el código del proceso que genera las pruebas de test y realice una simulación. Compruebe que tanto el reloj (*SCL)* como las señales de temporización cumplen con los tiempos de la especificación.