PRAC2: Desarrollo e integración de circuitos digitales sobre dispositivos programables

Profesores responsables

Profesor responsable:

- Dr. Pere Tuset <peretuset@uoc.edu>

Profesores colaboradores:

- Dr. Francisco Vázquez <fvazquez@uoc.edu>

Presentación

Esta práctica se focaliza en la utilización del lenguaje VHDL para describir un diseño digital sencillo y poder verificar su correcto funcionamiento mediante bancos de pruebas y simulaciones. La implementación sobre una FPGA concreta debe permitir también obtener información adicional de frecuencia de funcionamiento, etc. En resumen, se trata de seguir un proceso de diseño lo más similar al real, pero en un entorno controlado y con un diseño sencillo. Se trata de llevar a la práctica los conocimientos adquiridos gracias al estudio del material base facilidad a la asignatura, en concreto los contenidos de los Módulos 4 y 5, y ampliarlos utilizando herramientas de diseño avanzado.

Competencias

Las competencias asociadas a esta actividad son las siguientes:

- Conocer las características generales y las herramientas involucradas en el proceso de diseño de un circuito integrado de aplicación específica (ASIC), específicamente una FPGA.
- Saber implementar funciones lógicas en dispositivos lógicos programables mediante lenguajes de descripción de hardware, en concreto, VHDL.
- Saber diseñar sobre FPGAs complejas, aplicando técnicas específicas, y verificar el correcto funcionamiento de la implementación resultante.

Objetivos

Los objetivos de aprendizaje asociados a esta actividad son los siguientes:

- Diseñar sistemas empotrados sobre FPGAs, basados en el procesador Nios-II y varios periféricos, utilizando una herramienta de diseño avanzado llamada Platform Designer (Intel) que se utiliza en el diseño digital y la integración de IP (Intellectual Property) cores.
- Diseñar un generador de ondas cuadradas en VHDL y convertirlo en periférico del procesador Nios-II a través de una interfaz Avalon Memory-Mapped (MM).
- Diseñar un sistema embebido formado por el procesador Nios-II y el periférico generador de ondas cuadradas, programar el procesador Nios-II y evaluar el funcionamiento del sistema mediante simulación con ModelSim-Altera.
- Entender cómo funcionan las herramientas de diseño avanzado que los fabricantes ponen a nuestro alcance para verificar los resultados de síntesis del diseño digital y el análisis temporal.

Recursos

Los recursos que se recomienda utilizar para esta práctica son los siguientes:

- **Recursos** básicos:
  - **Módulo** 4. "Introducción a los circuitos integrados de aplicación específica", Jordi Riera Baburés, UOC
  - **Módulo** 5. "Dispositivos lógicos programables", Jordi Riera Baburés, UOC
  - "Simulación de sistemas digitales mediante lenguajes descriptores de hardware", Juan Antonio Martínez Carrascal, UOC
  - "VHDL Simulation: Test bench", diapositivas UOC
- **Recursos** complementarios:
  - Intel Quartus Prime Standard Edition User Guide: Platform Designer https://www.intel.com/content/www/us/en/programmable/documentation/jrw1529444674987.html
  - Embedded Design Handbook https://www.intel.com/content/www/us/en/programmable/documentation/iga1446487888057.html
  - Introduction to Platform Designer (26 ') https://www.intel.com/content/www/us/en/programmable/support/training/course/oqsys1000.html
  - Creating a System Design with Platform Designer: Getting Started (28') https://www.intel.com/content/www/us/en/programmable/support/training/course/oqsyscreate.html

- Custom IP Development Using Avalon® and Arm AMBA AXI Interfaces (107 ')
  https://www.intel.com/content/www/us/en/programmable/support/training/course/oqsys3000.html
- Avalon® Interface Specifications
  https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/manual/mnl_avalon_spec.pdf

Criterios de valoración

Los criterios de valoración de los ejercicios que conforman esta actividad son los siguientes:

- Razonamiento de las respuestas en todos los ejercicios. Las respuestas sin justificación no recibirán puntuación.
- La valoración se indica en cada una de las partes de la práctica.
- **ATENCIÓN:** La práctica evalúa el conocimiento de VHDL y de las herramientas de diseño y simulación basadas en bancos de pruebas. Si estos dos aspectos no quedan acreditados, la puntuación máxima de la práctica será de D (2 puntos).

Formato y fecha de entrega

De cara a la entrega de esta actividad hay que tener en cuenta los siguientes aspectos:

- Hay que entregar la solución en un archivo ZIP, que contenga la solución de la práctica en formato PDF utilizando una de las plantillas entregadas conjuntamente con este enunciado, así como el archivo completo de los diseños de los ejercicios que requieren codificación VHDL (preferiblemente utilizando el herramienta del Quartus, *Project -> Archive Project*).
- La memoria en PDF debe incluir todo el código en VHDL, tanto del diseño como de los bancos de pruebas utilizados en las simulaciones, así como todas las gráficas obtenidas, junto con los comentarios adecuados. Piensa que si hay algún problema para reproducir los resultados de su diseño, esto será lo único que quedará para defender su trabajo.
- Se entregará a través de la aplicación de Entrega y registro de AC del apartado Evaluación de su aula.
- Para dudas y aclaraciones sobre el enunciado, diríjase al consultor responsable de su aula, preferiblemente mediante los foros (excepto en el caso de que se tenga que incluir código o detalles específicos sobre la resolución).
- La fecha límite de entrega es el **18 de Mayo** (a las 23:59 horas).

Descripción de la actividad

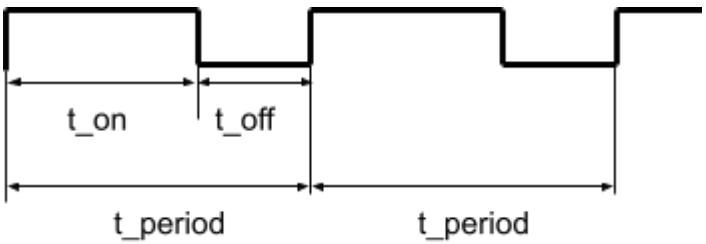
Esta práctica está compuesta por tres partes que pueden funcionar de manera más o menos independiente:

- **Parte 1** (30%): El objetivo de la primera parte consiste en desarrollar un generador de ondas cuadradas basado en una máquina de estados finitos y un contador. El funcionamiento del generador se verificará mediante simulación utilizando el Modelsim-Altera con un banco de pruebas.
- **Parte 2** (40%): En la segunda parte se utilizará la herramienta Platform Designer para diseñar un sistema empotrado formado por un procesador Nios-II, una UART, una memoria RAM y un periférico basado en el generador de ondas cuadradas diseñado en la primera parte. Se trata de desarrollar un periférico generador que se pueda conectar al procesador Nios-II mediante una interfaz Avalon-MM slave. A continuación, tendremos que diseñar el código del procesador Nios-II y comprobar con ModelSim-Altera el funcionamiento del sistema empotrado completo mediante un banco de pruebas.
- **Parte 3** (30%): En la última parte se trabajará con herramientas avanzadas disponibles en los entornos de diseño que los fabricantes de FPGAs ponen a nuestro alcance: visualizadores del resultado del proceso de síntesis y herramientas de análisis temporal.

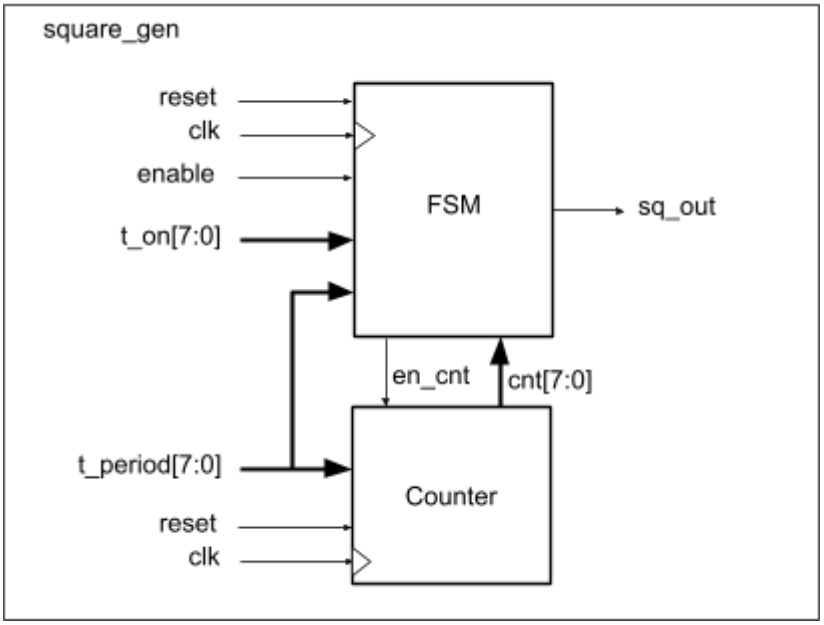
Enunciado

Parte 1 (30%)

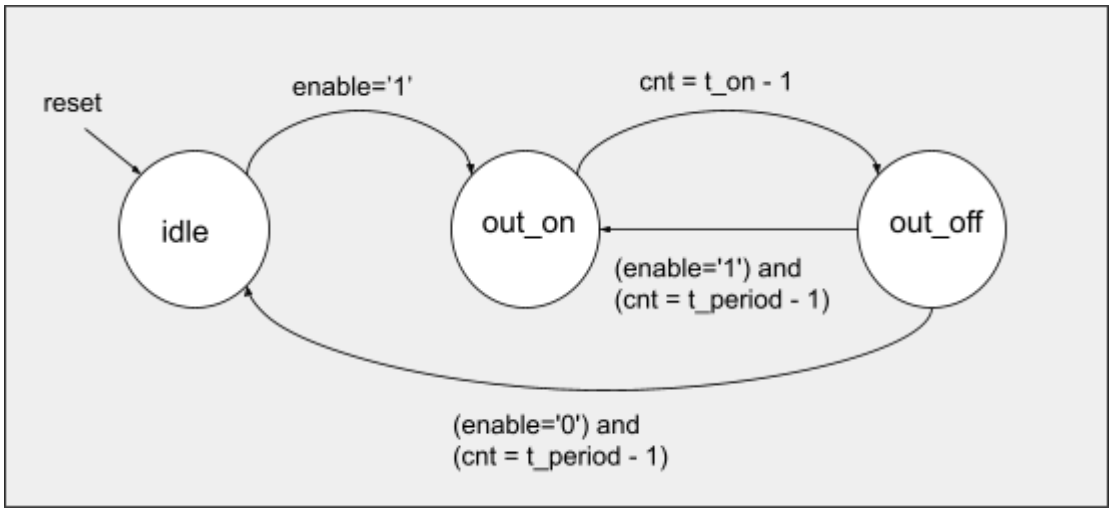
En esta primera parte diseñaremos un generador de ondas cuadradas denominado "square_gen". La salida del generador de ondas cuadradas es una señal digital periódica con un tiempo de ON y un tiempo de OFF (t_off) configurables. El tiempo de ON (t_on) es el tiempo en que la señal está a nivel lógico "1" y el tiempo de OFF (t_off) es el tiempo en que la señal está a nivel lógico "0". El periodo (t_period) de la señal periódica es la suma del tiempo de ON y el de OFF. La siguiente figura muestra un ejemplo de la forma de onda de la señal cuadrada.



La siguiente figura muestra el diagrama de bloques del generador de ondas cuadradas. Está formado por una máquina de estados finitos (FSM, Finite State Machine) y un contador de 8 bits. Las señales de entrada “t_on[7:0]” y ”t_period[7:0]” son de 8 bits y se utilizan para configurar el tiempo de ON y el periodo de la señal de salida, respectivamente, en número de ciclos de reloj. La señal de entrada “enable” se utiliza para activar/desactivar la generación de la señal de salida “sq_out” del generador. La señal “en_cnt” se utiliza para habilitar el contador. La salida del contador es “cnt[7:0]”.



La figura siguiente muestra el diagrama de estados de la FSM. Como se puede observar, la FSM debe tener 3 estados (idle, out_on y out_low) y las condiciones para conmutar entre estados se detallan en la figura. Los valores “t_on” y “t_period” sólo deben modificarse cuando la FSM se encuentra en el estado “idle”.



Es una máquina de Mealy y sus señales de salida deben tener los siguientes valores en cada estado:

- En el estado “idle”:
  - sq_out = '0'
  - en_cnt = '1' si (enable = '1'), si no, entonces en_cnt = '0'

- En el estado “out_on”:
  - sq_out = '1'
  - en_cnt = '1'
- En el estado “out_off”:
  - sq_out = '0'
  - en_cnt = ‘0' si (enable = '0') y (cnt = t_pwm - 1), si no, entonces en_cnt = '1'

El siguiente listado (dividido en 2 partes) corresponde al código VHDL del generador de ondas cuadradas "square_gen". El proceso “fsm_pr” describe las condiciones para cambiar de un estado a otro y el proceso “fsm_outputs” describe la asignación de valores a las salidas de la máquina. El proceso “counter_pr” describe el contador binario. Como se puede observar, la descripción VHDL de los procesos “fsm_pr” y “fsm_outputs” está incompleta. El objetivo de este ejercicio consiste en completar el diseño y verificar el funcionamiento del generador mediante simulación.

Listado VHDL del "square_gen.vhd"

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity square_gen is
    port (
        clk      : in std_logic;    -- Clock signal
        reset    : in std_logic;    -- Reset signal
        enable   : in std_logic;    -- Enable the generation of square output signal
        t_on     : in std_logic_vector(7 downto 0);-- Time ON in clock sycles
        t_period : in std_logic_vector(7 downto 0);-- Period of the output signal in clock sycles
        sq_out   : out std_logic    -- Output signal
    );
end entity square_gen;

architecture rtl of square_gen is

    type state_t is (idle, out_on, out_off);  -- states of fsm
    signal state: state_t;

    signal en_cnt : std_logic := '0';   -- counter enable signal
    signal cnt : unsigned(7 downto 0);  -- counter output

begin

    -- Finite states machine: conditions to switch between states
    fsm_pr : process(clk, reset)
    begin
          --------------------
          -- COMPLETAR AQUÍ
          --------------------

    end process fsm_pr;

    -- Finite states machine: outputs in each state
    fsm_outputs: process (state)
    begin
          --------------------
          -- COMPLETAR AQUÍ
          --------------------

    end process fsm_outputs;

    cnt_pr : process(clk, reset)
    begin
       if (reset = '1') then
          cnt <= (others => '0');
       elsif (rising_edge(clk)) then
          if (en_cnt = '1') then
             if (cnt = unsigned(t_period) - 1) then
                cnt <= (others => '0');
             else
                cnt <= cnt + 1;
             end if;
          end if;
       end if;
    end process cnt_pr;

end rtl;
```

Junto con el enunciado de la PRAC, se os ha proporcionado un archivo llamado "2020_PRAC2_SquareGen_Part1.qar" que corresponde al proyecto a partir del cual debe completar el diseño de la FSM del "square_gen". Con él se os pide hacer lo siguiente:

a) Recuperar el proyecto a partir del archivo proporcionado (a Quartus Prime, ir a *Project -> Restore Archived Project*).
b) Modificar el archivo "square_gen.vhd" para incluir el código que falta en la descripción de la FSM (procesos “fsm_pr” y “fsm_outputs”). La FSM debe cumplir con todos los requerimientos funcionales que se han descrito antes.
c) Compilar el diseño sobre una FPGA de Altera de la familia Cyclone IV E ejecutando *Processing → Start Compilation*. Se deben mostrar los resultados de ocupación y explicar brevemente los elementos lógicos, número de pines, etc.
d) El proyecto incluye un banco de pruebas (fichero “square_gen_tb.vht”) que debe completarse para verificar el funcionamiento del diseño del "square_gen" mediante simulación RTL con *Modelsim-Altera Starter Edition*. En particular, tenéis que verificar dos test cases:
    - **Test case 1**: configurar t_on = 8 y t_period = 16;
    - **Test case 2**: configurar t_on = 16 y t_pwm = 64.

    Se pide añadir el código que falta en el fichero “square_gen_tb.vht” para:
    (i) asignar los parámetros de configuración t_on y t_period en el generador, y
    (ii) habilitar el funcionamiento del generador.

e) **Lanzar la simulación con ModelSim** (clickar con botón derecho en *RTL Simulation* → clickar en *Start*). Si es necesario añadir más señales a la ventana Wave, en la ventana “sim - Default” debe seleccionarse “uut” para que aparezcan todas las señales internas del square_gen en la ventana “Objects”. En la ventana “Objects”, seleccionar todas las señales deseadas, clickar con botón derecho y seleccionar “Add Wave”. Finalmente, lanzar de nuevo la simulación ejecutando los siguientes comandos en la consola (ventana “Transcript”) de ModelSim:

    ```
    > restart
    > run 30 us
    ```

    ATENCIÓN: Se debe explicar detalladamente el resultado de la simulación de cada test case.

Parte 2 (40%)

En esta segunda parte de la práctica trabajaremos con una herramienta de diseño avanzado, llamada Platform Designer, que está integrada en el software Quartus Prime de Intel. El Platform Designer simplifica la tarea de definición e integración de IP cores (componentes VHDL) en un diseño digital con FPGA. El Platform Designer crea automáticamente la lógica de interconexión entre IP cores a partir de un diseño de interconexiones que nosotros definimos a alto nivel. La generación automática de la lógica de interconexión nos ahorra tiempo de diseño VHDL. El Platform Designer nos permite integrar los IP cores a partir de una interfaz gráfica donde se representa el sistema digital. Nos ofrece una librería de IP cores que se pueden configurar y soporta la integración de IP cores diseñados por terceros o componentes diseñados por nosotros.

En esta segunda parte de la práctica desarrollaremos un periférico para el procesador Nios-II a partir del diseño VHDL del generador de ondas cuadradas (square_gen) implementado en la primera parte de esta práctica. Para conectar el Nios-II con nuestro periférico utilizaremos un bus de tipo Avalon Memory-Mapped (MM). El periférico “square_gen” deberá funcionar como dispositivo slave del bus Avalon-MM y permitir accesos de escritura desde el Nios-II para configurar el periférico y accesos de lectura para comprobar su status. El procesador Nios-II actuará como dispositivo master del bus Avalon-MM. Una vez diseñada y verificada la lógica y los registros de la interfaz Avalon-MM slave con el Quartus Prime, añadiremos el periférico “square_gen” como IP core al Platform Designer, y crearemos un sistema empotrado formado por un procesador Nios-II, una memoria on-chip RAM, un periférico JTAG-UART y un periférico “square_gen”. Finalmente, programaremos el Nios-II y comprobaremos el funcionamiento del sistema mediante simulación con el ModelSim-Altera.

Para realizar esta segunda parte de la práctica es muy recomendable utilizar los siguientes recursos complementarios:

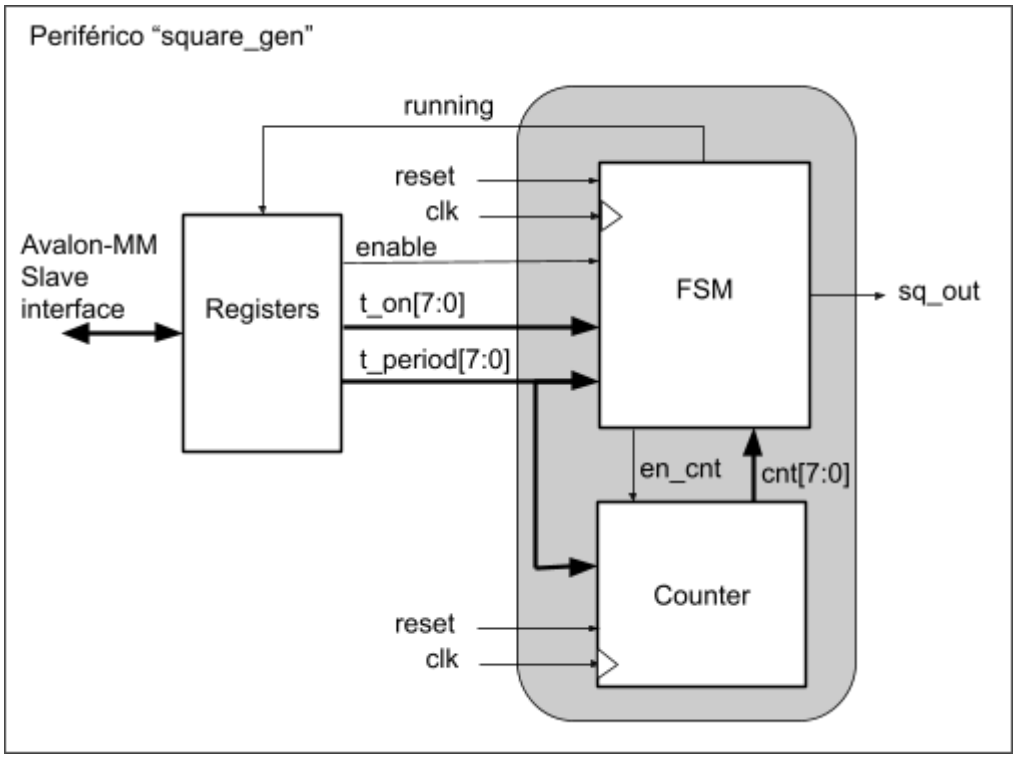
- Introduction to Platform Designer (26’)
  https://www.intel.com/content/www/us/en/programmable/support/training/course/oqsys1000.html
- Creating a System Design with Platform Designer: Getting Started (28’)
  https://www.intel.com/content/www/us/en/programmable/support/training/course/oqsyscreate.html
- Custom IP Development Using Avalon® and Arm AMBA AXI Interfaces (107’)
  https://www.intel.com/content/www/us/en/programmable/support/training/course/oqsys3000.html
- Avalon® Interface Specifications
  https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/manual/mnl_avalon_spec.pdf

Se pide lo siguiente:

1) **Ejercicio 1:** En este primer ejercicio diseñaremos el periférico “square_gen” para el procesador Nios-II utilizando el Quartus Prime. La siguiente figura muestra el diagrama de bloques del periférico. El periférico “square_gen” está formado por los siguientes elementos:

   (1) Un conjunto de registros que permiten configurar y controlar el funcionamiento del periférico a través de una interfaz Avalon-MM slave.
   (2) Una máquina de estados finitos que controla el funcionamiento del generador de señal y el tiempo de ON (a nivel "1") de la señal de salida.
   (3) Un contador que controla el período de la señal de salida.

   Como puede observarse en la figura, la máquina de estados y el contador incluidos en el periférico son exactamente los mismos elementos que se diseñaron en la parte 1 de esta práctica. **Además, se ha añadido una nueva señal de salida, "running",** a la máquina de estados para indicar si el generador está activo, es decir, se encuentra en los estados “out_on” o “out_off”. La descripción del resto de las señales de entrada/salida y las funcionalidades se detallan en el enunciado de la parte 1. En esta segunda parte de la práctica vamos a añadir los registros y la interfaz Avalon-MM slave para poder controlar el generador desde el procesador Nios-II.



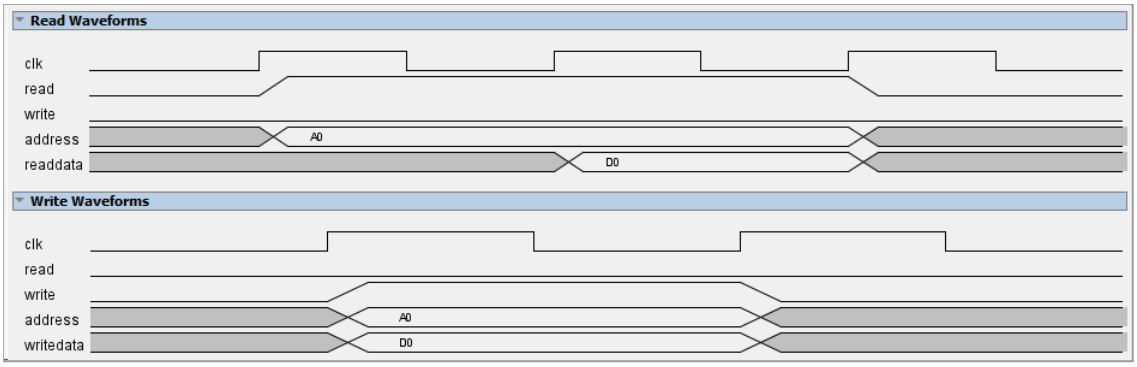
Los puertos de entrada y salida de la entidad del periférico se muestran en la siguiente figura. Como se puede observar, el periférico incluye una señal de clock de entrada (clk), una señal de reset), una señal de salida (sq_out), y un conjunto de señales de entrada/salida para la interfaz Avalon-MM de tipo slave.

```
entity square_gen is
   port (
      clk               : in std_logic;    -- Clock signal
      reset             : in std_logic;    -- Reset signal

      -- Avalon-MM slave interface
      avs_s0_address    : in std_logic_vector(2 downto 0);
      avs_s0_read       : in std_logic;
      avs_s0_readdata   : out std_logic_vector(7 downto 0);
      avs_s0_write      : in std_logic;
      avs_s0_writedata  : in std_logic_vector(7 downto 0);

      sq_out            : out std_logic    -- Output signal
   );
end entity square_gen;
```

En la tabla siguiente se detallan las características de los registros del periférico: su dirección (offset address) dentro de la interfaz Avalon-MM slave, el nombre del registro, los tipos de acceso (lectura, escritura) desde el procesador Nios-II, y la descripción de la funcionalidad asociada a cada registro.

Offset Address	Registro	Tipo de acceso	Descripción
0x00	control[7:0]	Write	0x01 = Activa el funcionamiento del generador 0x00 = Desactiva el funcionamiento del generador
0x01	t_on[7:0]	Read/Write	Número de ciclos de reloj en que la señal de salida “sq_out” está nivel "1"
0x02	t_period[7:0]	Read/Write	Número de ciclos de reloj que dura el período de la señal de salida “sq_out”
0x03	status[7:0]	Read	0x01 = Preparado para configurar parámetros del generador 0x00 = No preparado para configuración

En la siguiente figura se muestran los diagramas de tiempo de las señales del Avalon-MM slave interface en un acceso de lectura (read waveforms) del contenido de un registro desde un dispositivo master externo (p.e., Nios-II) y en un acceso de escritura (write waveforms) en un registro.



Junto con el enunciado de la PRAC, se os ha proporcionado un archivo llamado "2020_PRAC2_niosII_part2.qar" que corresponde al proyecto de diseño del periférico “square_gen”. Con este fichero se pide hacer lo siguiente:

a) Debéis recuperar el proyecto a partir del archivo proporcionado (en Quartus Prime, ir a *Project -> Restore Archived Project*). Una vez abierto el proyecto, abrir el archivo de diseño VHDL llamado "square_gen.vhd" disponible en el directorio “ip/square”. El siguiente listado corresponde a una parte del código VHDL del periférico "square_gen.vhd". Como se puede comprobar, la descripción VHDL está incompleta. Se debe modificar el fichero "square_gen.vhd" para incluir el código que falta en los siguientes procesos VHDL para cumplir con todos los requerimientos que se han descrito antes:
   - Proceso "fsm_pr": completar con el código diseñado en la parte 1 de esta práctica.
   - Proceso "fsm_outputs": completar con el código diseñado en la parte 1 de esta práctica y añadiendo la señal “running”, que se activa a nivel “1” cuando la máquina de estados está en “out_on” o “out_on”.
   - Proceso "write_reg_pr"
     - **Funcionalidades**: escribir el valor del bus "avs_s0_write_data" en los registros del periférico “square_gen”. Dependiendo del valor de la dirección "avs_s0_address" hay que asignarle el valor del bus "avs_s0_write_data" a un registro o a otro. Puede utilizarse un **demultiplexor** síncrono con una entrada (el bus "avs_s0_write_data") y varias salidas (los registros). Las señales de selección en el demultiplexor son las líneas de dirección "avs_s0_address".
     - **Añadir el código que falta** entre las líneas 92 y 98.
   - Proceso "read_reg_pr"
     - **Funcionalidades**: leer los registros del periférico “square_gen” a través de la interfaz Avalon-MM slave. Se trata de asignar el valor de los registros al bus de datos "avs_s0_read_data". Dependiendo del valor de la dirección "avs_s0_address" hay que asignarle el valor de

un registro. Puede utilizarse un **multiplexor** síncrono con varias entradas (los registros) y una salida (el bus "avs_s0_read_data"). Las señales de selección en el multiplexor son las líneas de dirección "avs_s0_address".

- **Añadir el código que falta** entre las líneas 108 y 114.

Listado VHDL parcial del “square_gen.vhd”

```
81   -----------------------------------------------
82   -- Avalon-MM slave interface logic
83   -----------------------------------------------
84
85   -- Write to internal registers from the Avalon-MM interface
86   write_reg_pr : process(clk, reset)
87   begin
88      if (reset = '1') then
89         control  <= (others => '0');
90         t_on     <= (others => '0');
91         t_period <= (others => '0');
92      elsif (rising_edge(clk)) then
93
94            --------------------
95            -- COMPLETAR AQUÍ
96            --------------------
97
98      end if;
99   end process write_reg_pr;
100
101  enable <= control(0);
102
103  -- Read from internal registers from the Avalon-MM interface
104  read_reg_pr : process(clk, reset)
105  begin
106     if (reset = '1') then
107        avs_s0_readdata   <= (others => '0');
108     elsif (rising_edge(clk)) then
109
110           --------------------
111           -- COMPLETAR AQUÍ
112           --------------------
113
114     end if;
115  end process read_reg_pr;
116
117  status(0) <= not(running);
118  status(7 downto 1) <= (others => '0');
119
```

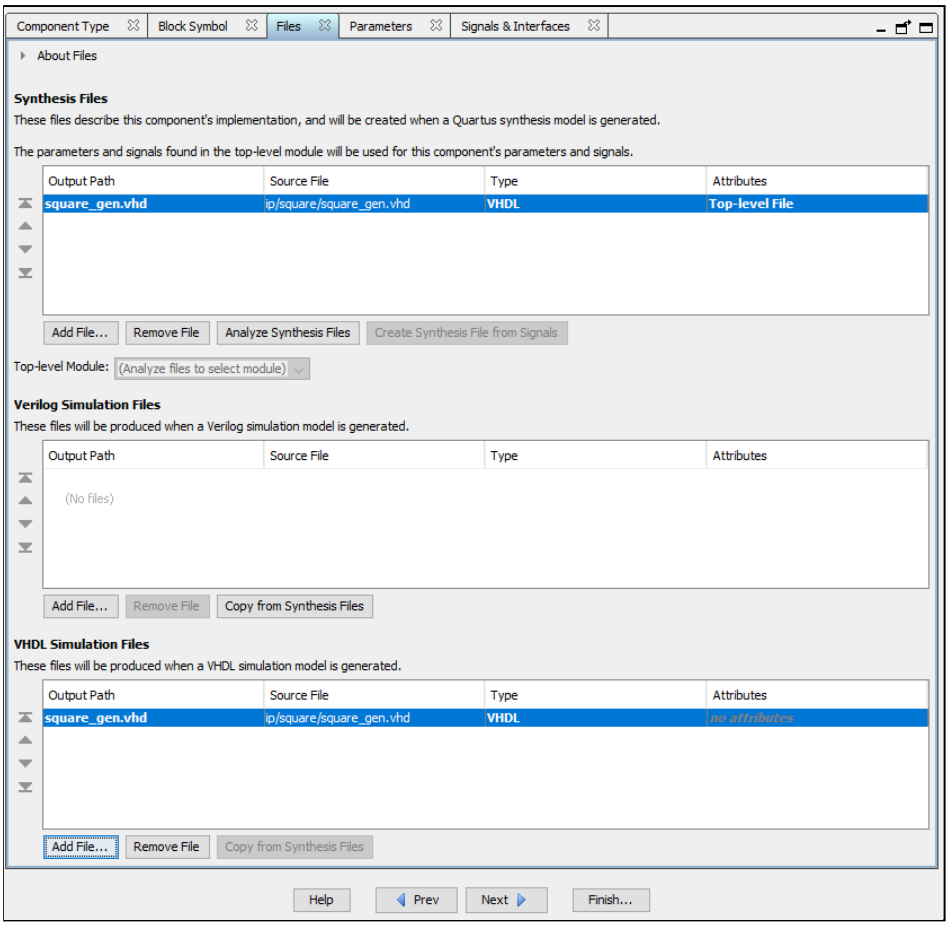
b) El proyecto incluye un banco de pruebas (fichero “square_gen_tb.vht” ubicado en el directorio \ip\square\) que debe completarse para verificar el funcionamiento del diseño del "square_gen.vhd" mediante simulación RTL con Modelsim-Altera Starter Edition. Con la simulación se debe demostrar lo siguiente:
   - Comprobar que se escriben correctamente los siguientes valores en los registros internos del periférico a través del Avalon-MM interface:
     - t_on = 8
     - t_period = 32
   - Comprobar que se lee correctamente el valor de los registros “t_on” y “t_period” a través del bus Avalon.
   - Comprobar que se activa el funcionamiento del generador de señal escribiendo 0x01 en el registro “control”.
   - Comprobar que el valor del registro “status” cambia de 0x01 (generador desactivado) a 0x00 indicando que el generador se ha activado.

- Comprobar que la señal de salida del generador funciona correctamente.
- Comprobar que se desactiva el funcionamiento del generador escribiendo 0x00 en el registro “control”.

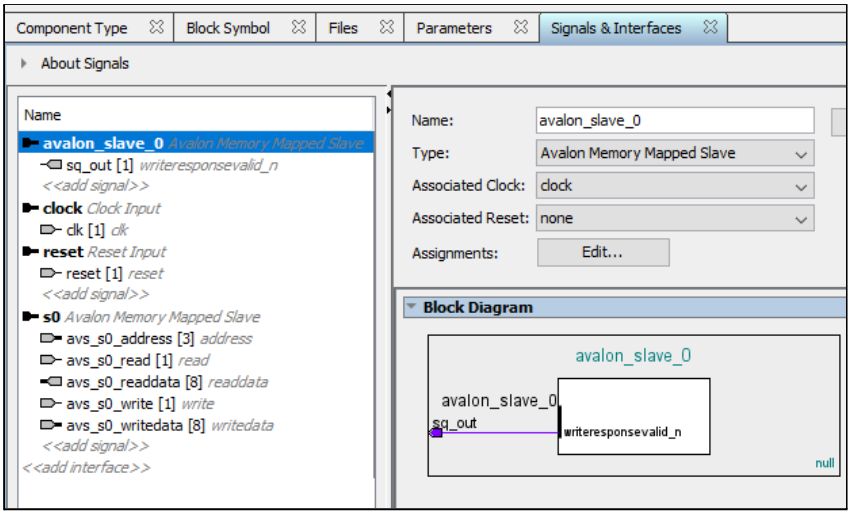
ATENCIÓN: Antes de lanzar la simulación hay que cambiar el nombre de la "Top-level entity" del proyecto,

1) En el project navigator, seleccionar Files.
2) Seleccionar el fichero “square_gen.vhd”
3) Clicar el botón derecho y seleccionar "Set as Top-level entity"

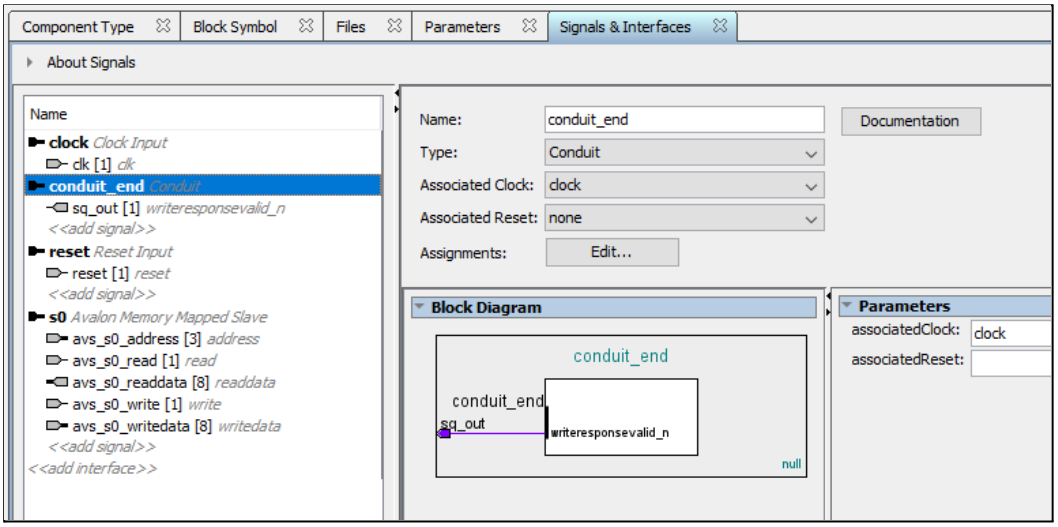
2) **Ejercicio 2:** En este segundo ejercicio añadiremos el periférico “square_gen” como nuevo componente al Platform Designer. Esto nos permitirá reutilizar nuestro periférico en el diseño de sistemas embebidos con el Platform Designer. Se debe hacer lo siguiente:
   a) Abrir el archivo de diseño del sistema empotrado (llamado "niosii_system.qsys") con el Platform Designer. En Quartus Prime, ir a *Tools -> Platform Designer* y abrir el archivo "niosii_system.qsys".
   b) Para crear un nuevo componente (IP core) en Platform Designer, ir a *File-> New Component*. Cuando se abra la ventana "Component Editor", hacer lo siguiente:
      - Ir a la pestaña "Component Type" e introducir lo siguiente: en los campos "Name" y "Display Name" poner "square_gen", y en el campo "Group" poner "Custom Logic".
      - Ir a la pestaña "Files" y en "Synthesis Files" clickar en "Add File..." para añadir el archivo con el diseño VHDL del periférico “square_gen.vhd” que se encontrará en el directorio "ip/square".
      - En "VHDL Simulation Files", clickar en "Add File..." para añadir el archivo con el diseño VHDL del periférico “square_gen.vhd” que se encontrará en el directorio "ip/square".



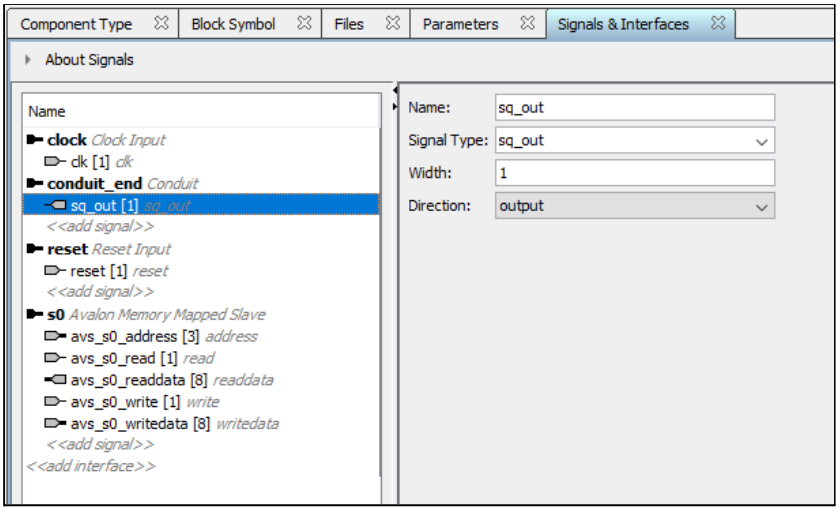
- Ir a la pestaña "Files" y clickar en "Analyze Synthesis Files" para comprobar que el diseño VHDL del periférico “square_gen” no tiene ningún error. Si se detecta algún error, se deberá volver a editar el código VHDL desde el Quartus Prime para corregirlo.
- Ir a la pestaña "Signals & Interfaces" para mostrar la lista de señales identificadas automáticamente por Platform Designer.



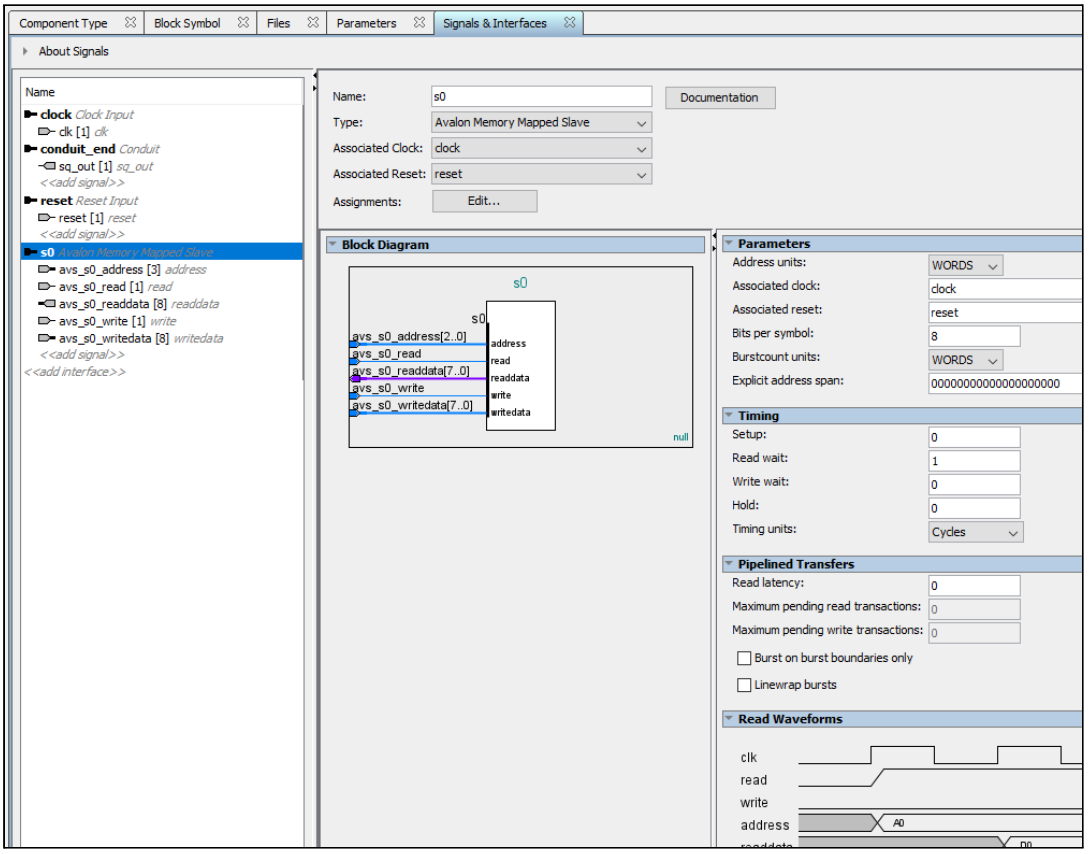
- Como se puede observar en la pestaña "Signals & Interfaces", la señal "sq_out" ha sido identificada erróneamente como una interfaz llamada "avalon_slave_0". Seleccionar la interfaz "avalon_slave_0" y cambiar "Type" por "Conduit". Las señales deben quedar tal como se muestra a continuación.



- Por defecto, se puede observar que la señal “sq_out” aparece como "writeresponsevalid_n". Para cambiarlo hay que seleccionar la señal "sq_out" (justo debajo de conduit_end, en lado izquierdo) y escribir "sq_out" en el campo "Signal Type" (en el lado derecho). Debe quedar como se muestra en la siguiente imagen.

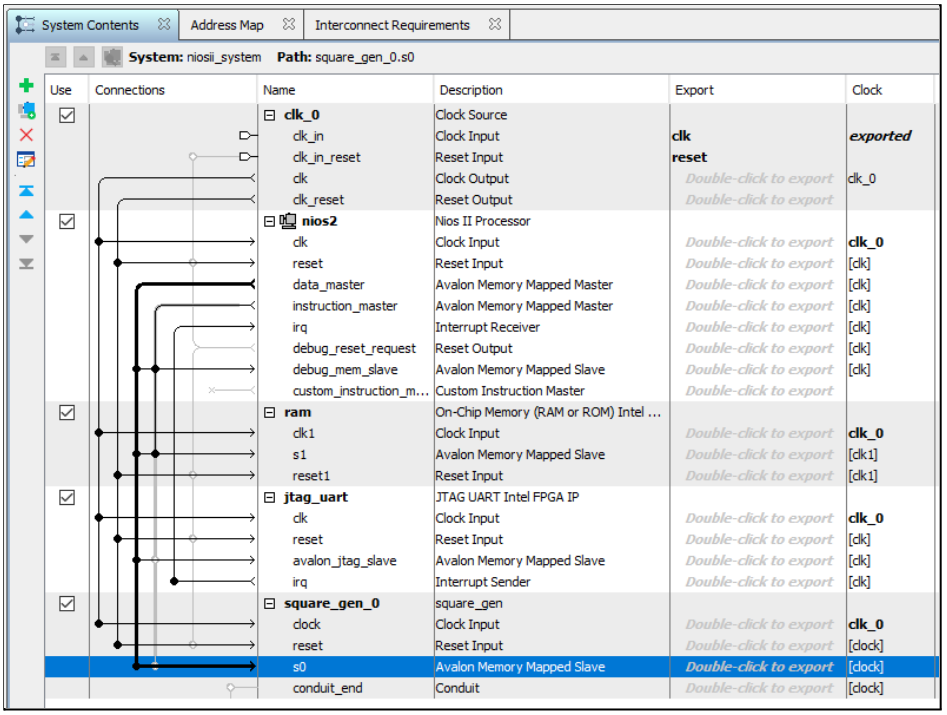


- Para eliminar el mensaje de error "s0: Interface must have an associated reset", hacer click en la interfaz "s0" y en el campo "Associated Reset" seleccionar la señal "reset". Las señales de la interfaz "s0" deben quedar tal como se muestra a continuación.

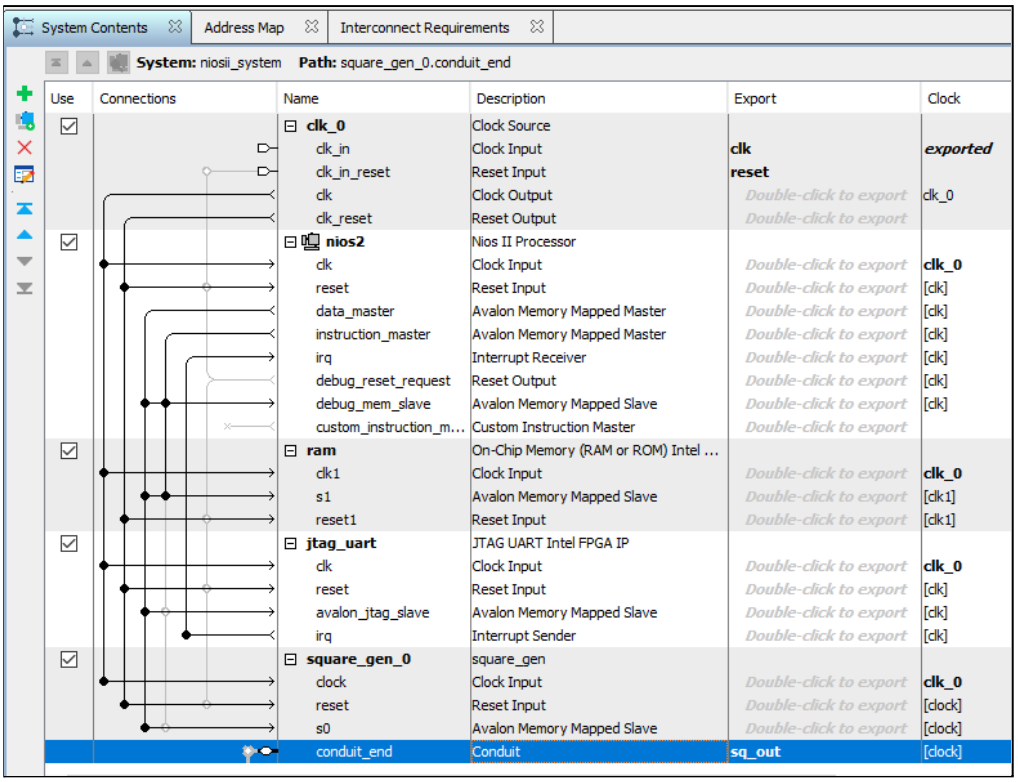


- Para guardar el componente "square_gen", ir a *File->Save*. El nuevo componente aparecerá en "IP Catalog" dentro del grupo "Custom Logic". Mostrarlo en el resultado de este ejercicio.

3) **Ejercicio 3:** En este tercer ejercicio añadiremos un periférico “square_gen” a nuestro sistema embebido "niosii_system.qsys", que está formado por un procesador Nios-II, una RAM y una JTAG-UART.
   a) Ir a la ventana "IP Catalog", buscar el “square_gen” en el grupo "Custom Logic" y hacer doble click sobre el “square_gen”. Cuando se abra la ventana del componente “square_gen”, hacer un click en "Finish". El Platform Designer creará automáticamente un nuevo componente llamado "square_gen_0".
   b) A través de la ventana "System Contents", conectar las señales clk (Clock input), reset (Reset input) y s0 (Avalon Memory Map Slave) del componente "square_gen_0" al resto del sistema. Debería quedar tal como se muestra en la siguiente figura.



   c) En la ventana "Messages" hay un warning que dice "*square_gen_0.conduit_end must be exported, or connected to a conduit matching*." Tendremos que conectar la salida digital del “square_gen_0” (de tipo Conduit) al exterior del sistema haciendo doble-click en la casilla "Export" asociada a "external_connection" y la llamaremos "sq_out". Debería quedar como se muestra en la siguiente figura.



4) **Ejercicio 4:** En este apartado prepararemos el testbench para evaluar más adelante el funcionamiento del sistema empotrado mediante simulaciones con ModelSim-Altera.
   a) En primer lugar, desde el Platform Designer y con el "niosii_system.qsys" abierto, ir a *Generate-> Generate VHDL*, seleccionar las siguientes opciones, y hacer un click en *Generate*.
      - *Create HDL design files for synthesis: None*
      - *Create block symbol file (.bsf): ON*
      - *Create simulation model: None*
   b) Para generar el testbench de manera automática, ir a *Generate->Generate Testbench System...*, seleccionar las siguientes opciones, y hacer un click en *Generate*. **El proceso de generación del testbench puede tardar unos minutos.** Mostrar el resultado y comprobar que no ha salido ningún error.
      - *Create testbench Platform Designer system: Standard, BFMs for standard Platform Designer interfaces*
      - *Create testbench simulation model: VHDL*

      NOTA: no hacer caso al siguiente warning "TB_Gen: niosii_system_inst_reset_bfm is not associated to any clock; connecting niosii_system_inst_reset_bfm to 'niosii_system_inst_clk_bfm.clk'"

c) Desde el Quartus Prime, abrir el archivo del testbench "niosii_system_tb.vhd" que se encuentra en el siguiente directorio:
... <directorio del proyecto> \ niosii_system \ testbench \ niosii_system_tb \ simulation

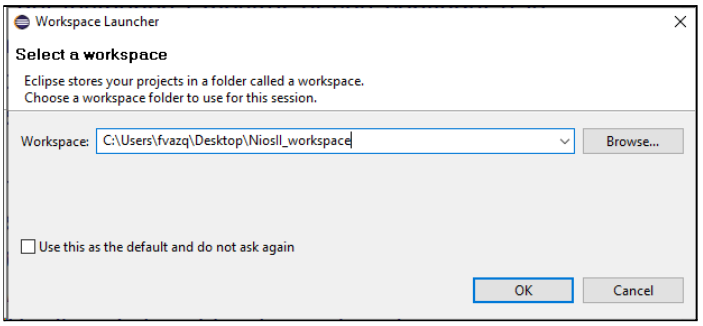
d) Seguir las siguientes indicaciones para modificar el fichero "niosii_system_tb.vhd" del testbench para acelerar la simulación:
- Eliminar la instancia del componente "altera_conduit_bfm" y
- Desconectar la salida "sq_out_sq_out" de la instancia “niosii_system” tal como se muestra en el siguiente listado VHDL.

```
54  begin
55
56     niosii_system_inst : component niosii_system
57        port map (
58           clk_clk       => niosii_system_inst_clk_bfm_clk_clk,       --    clk.clk
59           reset_reset_n => niosii_system_inst_reset_bfm_reset_reset, --  reset.reset_n
60           sq_out_sq_out => open -- niosii_system_inst_sq_out_sq_out          -- sq_out.sq_out
61        );
62
63     niosii_system_inst_clk_bfm : component altera_avalon_clock_source
64        generic map (
65           CLOCK_RATE => 50000000,
66           CLOCK_UNIT => 1
67        )
68        port map (
69           clk => niosii_system_inst_clk_bfm_clk_clk  -- clk.clk
70        );
71
72     niosii_system_inst_reset_bfm : component altera_avalon_reset_source
73        generic map (
74           ASSERT_HIGH_RESET    => 0,
75           INITIAL_RESET_CYCLES => 50
76        )
77        port map (
78           reset => niosii_system_inst_reset_bfm_reset_reset, -- reset.reset_n
79           clk   => niosii_system_inst_clk_bfm_clk_clk        --   clk.clk
80        );
81
82  -- niosii_system_inst_sq_out_bfm : component altera_conduit_bfm
83  --    port map (
84  --       clk           => niosii_system_inst_clk_bfm_clk_clk, --     clk.clk
85  --       sig_sq_out(0) => niosii_system_inst_sq_out_sq_out,   -- conduit.sq_out
86  --       reset         => '0'                                 -- (terminated)
87  --    );
88
89  end architecture rtl; -- of niosii_system_tb
90
```

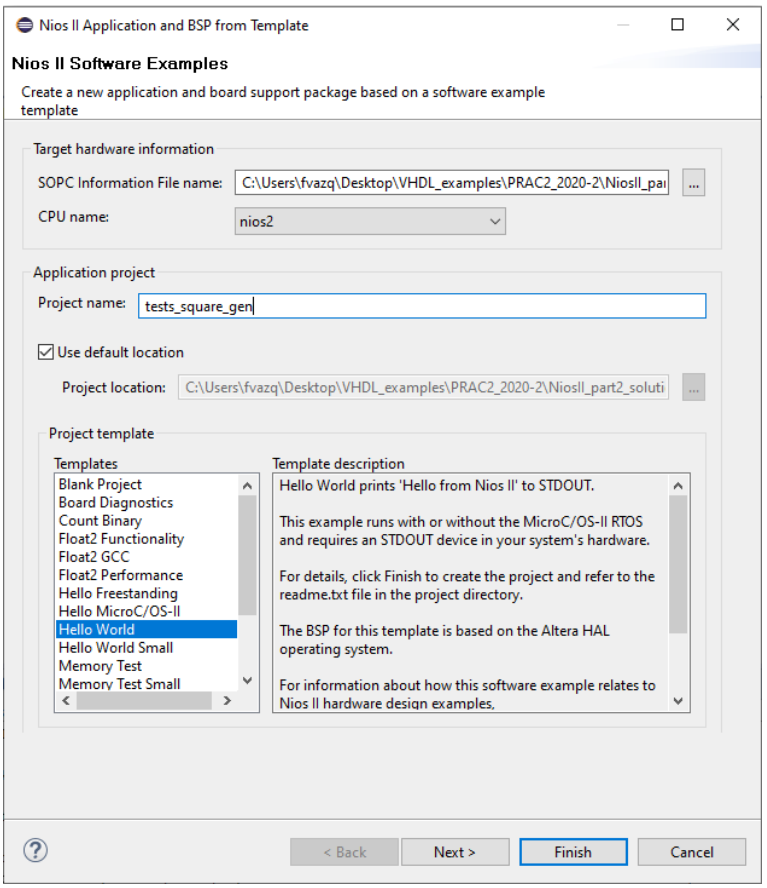
5) **Ejercicio 5:** En este apartado crearemos un proyecto software para el procesador Nios-II y diseñaremos el código en C para verificar el funcionamiento del sistema empotrado mediante simulación.

a) En primer lugar, desde el Platform Designer y con el "niosii_system.qsys" abierto, seleccionar *Tools -> Nios II Software Build Tools for Eclipse* para abrir el entorno de desarrollo Eclipse. A continuación, se deberá seleccionar el workspace o directorio donde se guardarán los proyectos software.

ATENCIÓN: los nombres de los directorios del workspace no deberían contener espacios en blanco.



b) Una vez abierto el entorno de desarrollo Eclipse, crearemos un nuevo proyecto software para el procesador Nios-II seleccionando *File -> New -> Nios II Application and BSP from Template* y seguir las siguientes indicaciones:
   - En el "SOPC Information File", seleccionar el fichero "niosii_system.sopcinfo" disponible **en el directorio del proyecto**.
   - En el "Project name", introducir "tests_square_gen".
   - En el "Project Template", seleccionar "Hello World".
   - Clickar en "Finish", comprobar que no sale ningún error, y mostrar el resultado.



c) Una vez creado el nuevo proyecto software, denominado "tests_square_gen", compilaremos el código siguiendo las indicaciones siguientes:
   - En la ventana "Project Explorer", hacer un click con el botón derecho sobre el nombre del proyecto que queremos compilar ("tests_square_gen") y seleccionar "Build Project".
   - Comprobar que no hay ningún error.

d) Modificar el código en C del fichero "Hello world" para dejarlo tal y como se muestra en la siguiente figura y comprobar que compila correctamente sin errores.

```
1  #include <stdio.h>
2  #include "io.h"
3  #include "system.h"
4
5  // Offset address of each register of the square_gen peripheral
6  #define REG_CONTROL     0x00
7  #define REG_T_ON        0x01
8  #define REG_T_PERIOD    0x02
9
10 int main()
11 {
12     //
13     // Write to registers of the square_gen peripheral: reg_t_on, reg_t_pwm
14     //
15
16     // t_on = 8 clock cycles
17     IOWR_8DIRECT(SQUARE_GEN_0_BASE, REG_T_ON, 8);
18
19     // t_period = 32 clock cycles
20     IOWR_8DIRECT(SQUARE_GEN_0_BASE, REG_T_PERIOD, 32);
21
22     //
23     // Enable the square signal generation by writing 0x01 to reg_control
24     //
25     IOWR_8DIRECT(SQUARE_GEN_0_BASE, REG_CONTROL, 0x01);
26
27     printf("Hello from Nios II!\n");
28
29     return 0;
30 }
31
```

6) **Ejercicio 6:** Una vez compilado el código del proyecto, procederemos a la simulación de nuestro sistema empotrado con ModelSim-Altera:

a) En la ventana "Project Explorer" del entorno Eclipse, hacer un click con el botón derecho sobre el nombre del proyecto software que queremos utilizar en la simulación ("tests_square_gen") y seleccionar *Run As -> Nios II ModelSim* para iniciar el software ModelSim y preparar los archivos de la simulación.

b) Añadir a la ventana “Wave” todas las señales del componente "square_gen_0" incluido dentro de "niosii_system_tb\niosii_system_inst". Para ello, en la ventana “sim - Default” debe seleccionarse el componente "square_gen_0" para que aparezcan

todas sus señales internas en la ventana “Objects”. En la ventana “Objects”, seleccionar todas las señales, clickar con botón derecho y seleccionar “Add Wave”.

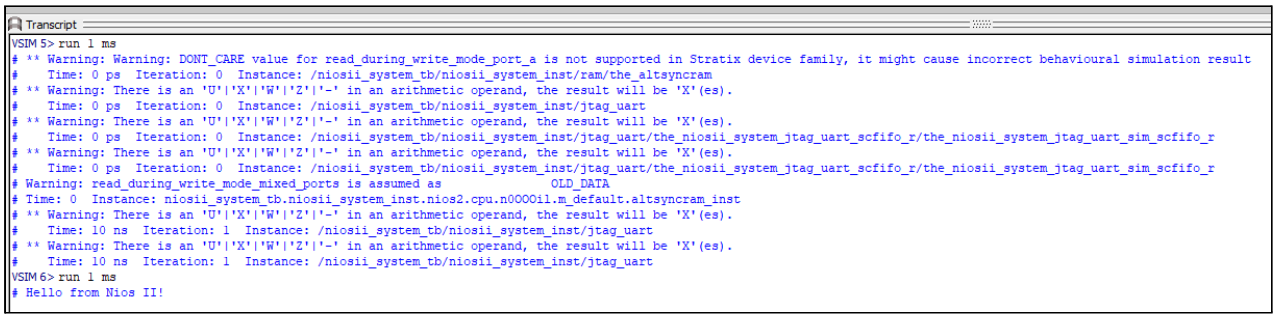
Lanzar la simulación con ModelSim ejecutando el comando "run 2 ms" en la consola. La simulación puede tardar entre 15 y 30 minutos dependiendo de la capacidad de vuestro ordenador.

VSIM 4> **run 2 ms**

Con la simulación se debe demostrar lo siguiente:

- Comprobar que se escriben correctamente los siguientes valores en los registros internos del periférico “square_gen” a través del Avalon-MM interface:
  - t_on = 8
  - t_period = 64
- Comprobar que se activa el funcionamiento del periférico “square_gen” escribiendo 0x01 en el registro control.
- Comprobar que el valor del registro “status” cambia de 0x01 (desactivado) a 0x00 indicando que el periférico se ha activado.
- Comprobar que la señal de salida del periférico “square_gen” funciona correctamente.

Cuando finaliza la simulación, el procesador Nios-II envía un mensaje (Hello from Nios II!) a través de la UART. En la siguiente figura se muestra la consola del ModelSim con el mensaje enviado por el Nios-II.



Parte 3: Utilización de herramientas avanzadas

EDA-CAD (30%)

El objetivo de esta última parte de la práctica consiste en conocer diversas herramientas avanzadas que los fabricantes de FPGAs ponen a nuestro alcance y con las que a menudo no hemos trabajado antes. Normalmente nos quedamos sólo con las herramientas de diseño VHDL, síntesis y simulación, pero incluso las versiones gratuitas del software ofrecido por Intel / Altera y Xilinx nos ofrecen mucho más.

Se pide lo siguiente:

1) Abrir el diseño del sistema embebido (llamado "niosii_system.qsys") creado en la Parte 2 y generar los ficheros VHDL para la síntesis con el Quartus Prime siguiendo las indicaciones siguientes:
    a) En el Quartus Prime, ir a *Tools->Platform Designer* y abrir el archivo "niosii_system.qsys".
    b) Desde el Platform Designer, generar los archivos de diseño VHDL para realizar el proceso de síntesis con el Quartus Prime. Ir a *Generate->Generate HDL...*, seleccionar las siguientes opciones, y hacer un click en *Generate*.
        - *Create HDL design files for synthesis: VHDL*
        - *Create block symbol file (.bsf): ON*
        - *Create simulation model: None*
2) Compilar el diseño VHDL del sistema desde el Quartus Prime siguiendo las indicaciones siguientes:
    a) En el Quartus Prime, antes de compilar se deben añadir todos los archivos de diseño VHDL del sistema empotrado. Ir a *Project-> Add / Remove Files in project...* y añadir el archivo "niosii_system.qip" disponible en el directorio "*<project folder>\niosii_system\synthesis*".
    b) En la ventana "Project Navigator" del Quartus Prime, seleccionar el fichero "niosii_system.qip", hacer un click en el botón derecho del ratón, y seleccionar "Set as Top-Level entity".
    c) En el Quartus Prime, ir a *Assignments-> Settings* y seleccionar "None" en "Tool Name" definido en "Simulation".
    d) En el Quartus Prime, ir a *Processing-> Start Compilation* para iniciar la compilación del sistema empotrado.
3) Una vez compilado el diseño VHDL del sistema desde el Quartus Prime, en este apartado utilizaremos diversas herramientas disponibles en el menú *Tools → Netlist Viewers*: el "RTL Viewer", el "State Machine Viewer" y el "Technology Map Viewer".
    a) Explicar brevemente cuál es la utilidad de estas tres herramientas.

b) Abrir el "RTL Viewer" y explicar cuáles son los elementos que se muestran. Añadir una imagen en la memoria.
c) Con el "RTL Viewer", mostrar los detalles de la instancia del módulo "square_gen" y comprobar si se ajusta a nuestras expectativas. Añadir una imagen en la memoria.
d) Clickar encima de la máquina de estados del módulo "square_gen" y se abrirá la herramienta del "State Machine Viewer". ¿Puedes reconocer los estados de tu código VHDL y las transiciones entre ellos? Añadir una imagen en la memoria.

4) Utilizar las herramientas de análisis temporal (Timing Analyzer en el caso de Intel / Altera) para obtener la máxima frecuencia de funcionamiento del diseño del sistema embebido.
   a) En primer lugar, tendremos que poner una restricción (constraint) en la señal de reloj para indicar la frecuencia a la que queremos que trabaje el diseño. Abrir la herramienta con *Tools → Timing Analyzer*. A continuación, vamos a la barra de menús y hacemos *Netlist → Create Timing Netlist*.
   b) Crear la constraint sobre el reloj con *Constraints → Create Clock*. Nos abre una ventana donde ponemos nombre a la señal (clock), seleccionamos el periodo deseado (10 ns, es decir, 100 MHz) y escogemos la señal de nuestro circuito (llamada "clk_clk") con '...' en targets. Clickar en "Run".
   c) Guardar el resultado en un fichero SDC con *Constraints → Write SDC*.
   d) Ahora tenemos que añadir en Quartus el fichero SDC que hemos guardado en el paso anterior. Cerramos el "Timing Analyzer" y en el Quartus añadimos el fichero SDC en *Assignments → Settings → Timing Analyzer*. Aquí también podemos indicar que nos genere los caminos críticos marcando la opción *Report Worst-case paths during compilation*.
   e) Compilamos el diseño de nuevo, y ahora ya tiene en cuenta la restricción impuesta.
   f) Ir a *Tools → Timing Analyzer*, en la ventana Tasks seleccionar *Datasheet → Report Fmax summary* para obtener la frecuencia máxima de funcionamiento (señal clock).