SISTEMAS EMBEBIDOS
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Segundo Cuatrimestre de 2019

Laboratorio N° 6
Hardware Programable, FPGAs y HDLs

Fecha de evaluación: Lunes 04/11/19
Objetivo: El objetivo de este laboratorio consiste en introducir a los alumnos en el uso de hardware programable en Sistemas Embebidos. Se persigue que los alumnos comprendan el proceso de desarrollo en este tipo de sistemas,utilicen un conjunto de herramientas propias de dicho proceso y experimenten con hardware programable real, en particular con FPGAs.
Desarrollo: El laboratorio deberá realizarse en comisiones de no más de 2 alumnos. Al finalizar la evaluación, se deberá comprimir y enviarse por mail respetando el siguiente formato: *LaboratorioX-ApellidosComision.zip.*

Descripción del hardware

El Hardware a utilizar se compone de:

- Una placa Terasic DE1-SoC, que integra un FPGA Altera Cyclone V, con procesador ARM Cortex-A9, y numerosos dispositivos de entrada-salida, interfaces de comunicación, audio, video, puertos de propósito general, memorias, etc [1].

Actividad 1: Integración de módulos de hardware programable.

Durante esta actividad introductoria del conjunto de herramientas de desarrollo Quartus Prime de Altera, se pretende programar un elemento de hardware sencillo en el FPGA, para poder probar el circuito utilizando las facilidades de E/S de la placa Altera DE1-SoC. Además de los módulos brindados por la cátedra, se pretende la integración de un IP (*Intellectual Property*) sencillo disponible en el entorno. La actividad se realizará en lenguaje Verilog.

1. Analice el ejemplo brindado por la cátedra [3] y determine los periféricos de la Placa Altera DE1-SoC utilizados.

2. En base al punto anterior, configure un proyecto con la herramienta `DE1SoC_SystemBuilder.exe`[2], disponible con la distribución de la placa.

3. Ejecute el entorno Quartus Prime y familiarícese con los principales componentes y opciones. Utilizando Quartus Prime, abra el proyecto generado en el punto anterior. Copie y pegue la lógica del ejemplo `Actividad1.v`, en el módulo *top level* del proyecto. Luego, integre al proyecto el archivo `Temporizador.v` [3].

4. Genere un módulo denominado `Multiplexor`, compatible con la instanciación realizada en el módulo *top level*, a partir de una variación de la IP de librería `LPM_MUX` (IP Catalog >Library >Basic Functions >Miscellaneous) para multiplexar entre dos datos de 4 bits cada uno.

5. Examine la asignación de pines realizada por la herramienta en el punto 2.

6. Sintetice el diseño.

7. Examine el resultado de la síntesis mediante los visores de *netlists* (accesibles desde Tools >Netlist Viewers). Use el RTL Viewer al completar el análisis y síntesis, y los Technology Map Viewers al completar las etapas de mapeo, posicionamiento y ruteo (Mapping and Fitting). Con el diseño ya posicionado, utilice el Chip Planner (Tools >Chip Planner) para examinar la disposición final del sistema en el FPGA.

8. Analice cada una de las etapas realizadas, las tareas llevadas a cabo por el ambiente de desarrollo durante la compilación del diseño, los archivos que se utilizan y se generan, y la relación existente con los distintos niveles de abstracción vistos en teoría, y que son provistos por los Lenguajes de Descripción de Hardware.

9. Descargue el *bitstream* generado usando el programador en modo JTAG ¿Qué diferencias encuentra al programar el FPGA de esta manera en lugar de hacerlo en modo *Active (quad) Serial* usando el dispositivo de con figuración EPCS128? ¿En qué escenarios es más adecuado el uso de cada una de estas alternativas?

10. Pruebe el circuito programado.

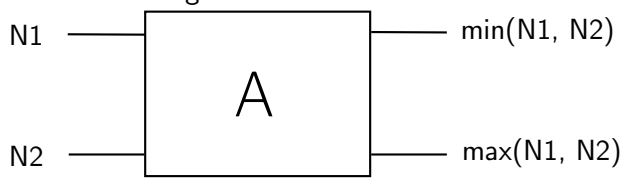
Actividad 2: Desarrollo de módulos de hardware programable

Se desea implementar un algoritmo de ordenamiento en hardware siguiendo una estructura de tipo *butterfly*. Este circuito deberá ser capaz de ordenar 8 valores enteros de 8 bits cada uno. Se incorporará una línea de registros para almacenar los operandos, y un mecanismo para multiplexar las entradas y demultiplexar las salidas, de manera de poder operar el sistema con las llaves, pulsadores y LEDs de la placa Altera DE1-SoC. Adicionalmente se deberá desarrollar una lógica que permita observar los resultados mediante los displays de 7 segmentos.

1. Configure un proyecto mediante la utilidad `DE1SoC_SystemBuilder.exe` con llaves, pulsadores, LEDs y displays de 7 segmentos.

2. Siguiendo la figura 1, especifique en Verilog un módulo combinacional (denominado A), que ordene 2 datos numéricos. Para ello deberá contar con dos entradas, N1 y N2, y dos salidas, S1 y S2, de 8 bits cada una. Deberá emitir la menor de las entradas por la salida S1, y la mayor de las entradas por la salida S2.

Figura 1: Módulo combinacional A



3. A partir de lo realizado en el punto anterior, implemente un módulo combinacional (denominado B) que utilizando el módulo A, permita realizar parte del ordenamiento *butterfly*, tomando 4 entradas numéricas de 8 bits. Este procesamiento debe realizarse de acuerdo al esquema de la figura 2.

4. Implemente un tercer módulo combinacional (denominado D) que utilizando el módulo A, realice otro de los segmentos del ordenamiento *butterfly*, de acuerdo al esquema de la figura 3.

5. Utilizando los módulos combinacionalaes A, B y D, implemente un módulo C, que contenga la estructura *butterfly* de ordenamiento paralelo. El esquema de conexiones se indica en la figura 4.

Figura 2: Módulo combinacional B

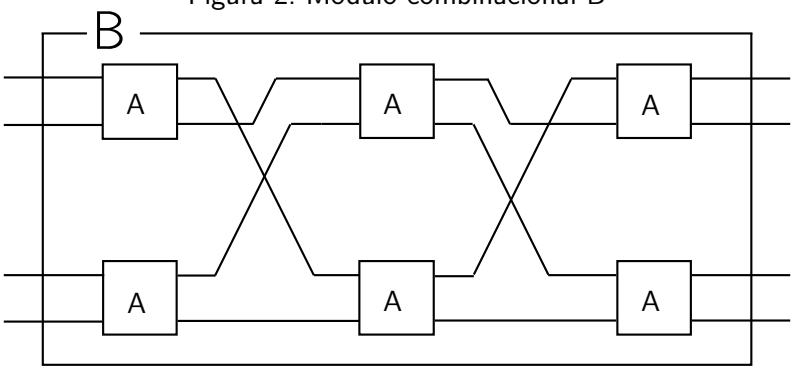
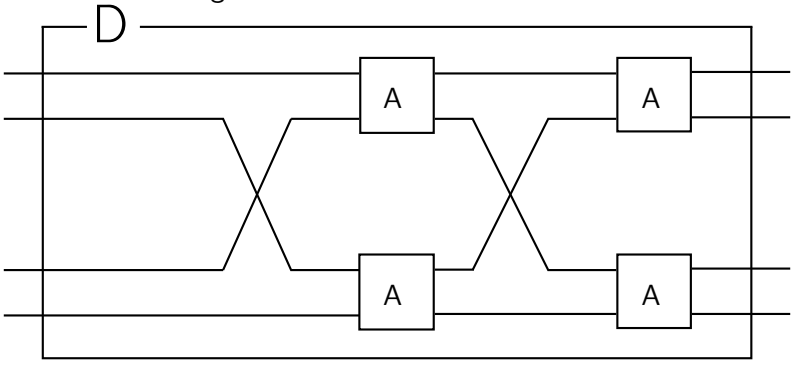


Figura 3: Módulo combinacional D



6. En el módulo *top level* añada:

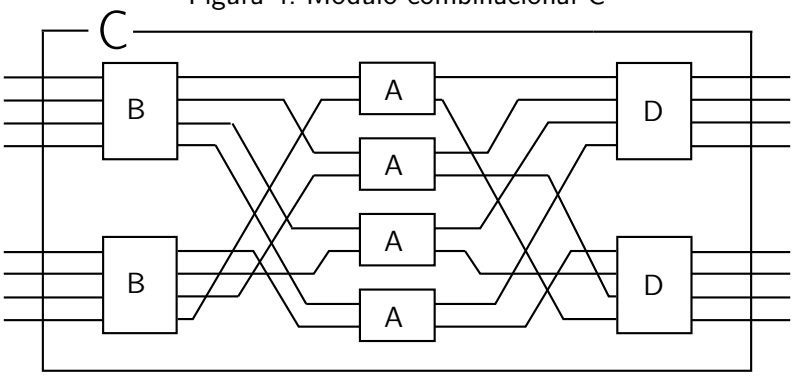
   a) Ocho registros de 8 bits c/u, previo a cada entrada del módulo C, que permitan almacenar los valores de entrada, ante la ocurrencia de un pulso sincrónico.

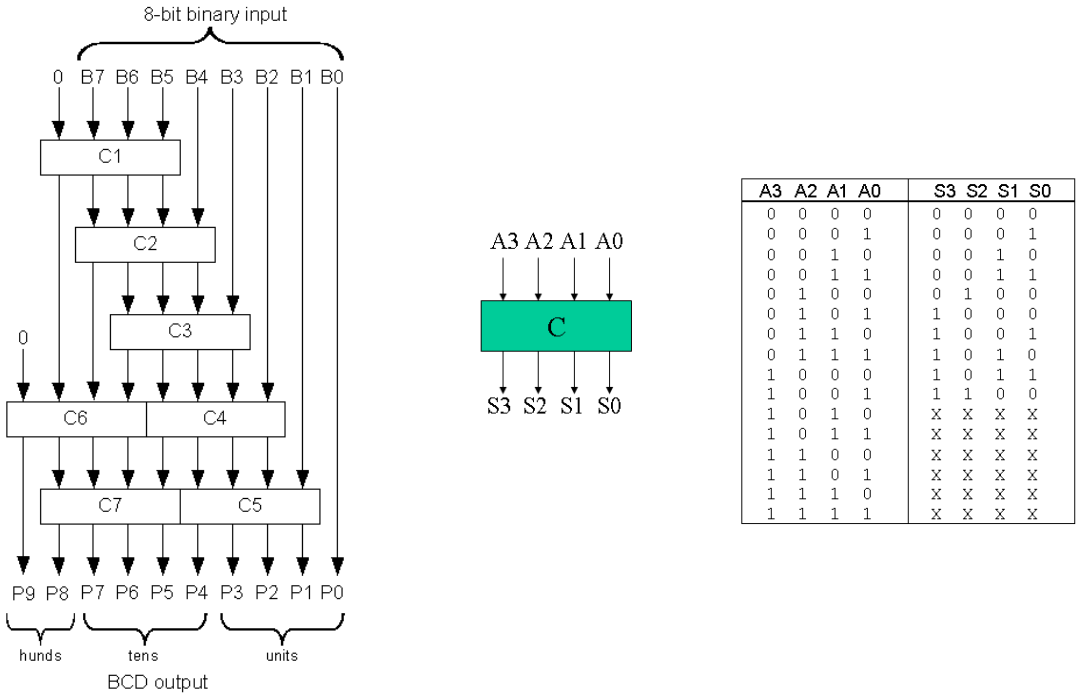
   b) Un demultiplexor `DM1`, que permita mediante 3 bits de selección, conectar las llaves `SW[7:0]`, con cada una de las 8 entradas de los registros del punto anterior.

   c) Un demultiplexor `DM2` que permita, a partir de los mismos tres bits de selección del punto b, conectar una única línea de pulso sincrónico (`KEY[2]`) a cada una de las entradas de reloj de los registros del inciso a).

   d) Un contador ascendente, `Cin`, de 3 bits incrementado mediante el pulsador `KEY[3]`, para

Figura 4: Módulo combinacional C





A3	A2	A1	A0	S3	S2	S1	S0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

Figura 5: Decodificador de Binario a BCD (algoritmo ShiftAndAdd3). Cada bloque C implementa la tabla de verdad que se ilustra.

generar los tres bits de selección de los demultiplexores de los incisos b) y c).

e) Un multiplexor, `MS`, (de salida) que permita seleccionar mediante 3 bits, cuál de las 8 salidas del módulo C se conecta a los `LEDR[7:0]`.

f) Un contador ascendente, `Cout`, de 3 bits incrementado mediante el pulsador `KEY[1]`, para generar los tres bits de selección del multiplexor del inciso e).

7. Implemente un circuito decodificador combinacional, que permita representar el resultado del multiplexor MS, utilizando los displays de 7 segmentos `HEX0` (unidades), `HEX1` (decenas) y `HEX2` (centenas). Para ello defina un módulo `BCD2HEX` que permita representar los dígitos en un display individual, a partir de recibir como entrada un dígito BCD, y con dicho módulo instanciado 3 veces y el decodificador Binario a BCD, implemente el decodificador combinacional completo. Tenga en cuenta que los segmentos de los displays se encienden cuando la línea de control asociada toma un valor lógico bajo [1].

8. Sintetice el diseño y verifique el mismo examinando las NetLists generadas.

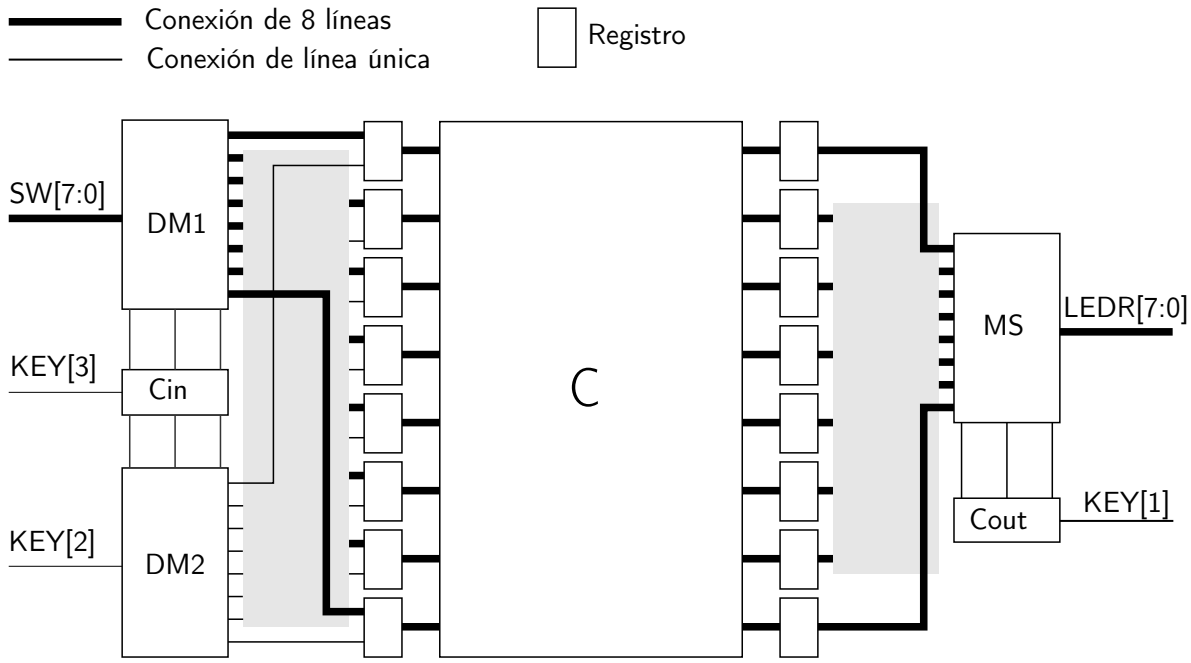
9. Descargue el diseño y pruebe el sistema.

Actividad 3: Una aplicación sencilla sobre procesadores soft-core

En esta actividad se pretende especificar en Verilog un sistema completo compuesto por un procesador, memoria y dispositivos simples de entrada/salida. Los puntos a desarrollar permitirán integrar una solución de hardware y software para un sistema sencillo sobre el procesador soft-core Nios II Gen 2 de Altera.

1. Configure un proyecto con la herramienta `DE1SoC_SystemBuilder.exe` contemplando el uso de *clock*, pulsadores, llaves y LEDs. Abra el proyecto en Quartus Prime. Abra la herramienta de integración de sistemas Plataform Designer (Ex.Qsys). Familiarícese con las principales opciones.

Figura 6: Actividad 2, inciso 6



En función de mantener a la imagen legible, no se muestran todas las conexiones. Se encuentran implícitas en las zonas grises.

2. Integre un sistema SoPC compuesto por: Módulo de clock a 50Mhz, procesador soft-core Nios II Gen2, memoria *on-chip*, un puerto PIO de entrada, un puerto PIO de salida, una unidad JTAG UART, y un System ID Peripheral. Conecte estos componentes y defina el mapa de memoria e interrupciones. Finalmente, genere el sistema.

3. Utilizando el entorno Quartus Prime, instancie el SoPC conectando la línea de clock a `CLOCK_50`, el reset a `KEY[0]`, el puerto de salida a `LEDR[7:0]` y el puerto de entrada a `SW[7:0]`.

4. Sintetice el diseño. Verifique el mismo examinando las *netlists* generadas.

5. Abra el entorno Nios II Software Build Tools for Eclipse. Utilizando el asistente cree una nueva aplicación, utilizando la opción Nios II Application and BSP from template, a partir del template Hello World, referenciando el archivo `.sopcinfo` generado

6. Configure y genere el BSP para el sistema.

7. Añada/copie el código del ejemplo propuesto por la cátedra [4] en el archivo principal de la aplicación. Construya el proyecto completo.

8. Con el programador de Quartus Prime descargue el diseño generado en el punto 2 al FPGA.

9. En el entorno NiosII SBT for Eclipse defina una Run Configuration para un *target* NiosII. Establezca una conexión con el SoPC programado en el FPGA y descargue el firmware. Compruebe el funcionamiento del sistema.

10. Compare el código utilizado en la actividad, con el ejemplo `nohal.c` [5] ¿Qué diferencias encuentra? ¿Qué ventajas/desventajas puede mencionar entre ambos programas? Justifique.

11. Describa el proceso de desarrollo realizado al utilizar un SoC implementado sobre un procesador soft-core ¿Qué ventajas presenta este esquema en cuanto a la posibilidad de introducir cambios en el diseño durante el desarrollo y/o modificaciones a futuro?

Actividad 4: Integración de un Sistema Embebido Soft-Core con unidades en Hardware Programable

En esta actividad se desea desarrollar un sistema que integre la estructura *butterfly* de ordenamiento paralelo, con un sistema SoPC Nios II. El usuario podrá ingresar valores numéricos enteros (no signados) que el sistema deberá mantener ordenados. Al ingresar un valor, el sistema reemplazará el menor de los 8 valores almacenados por el nuevo. Estos valores se ingresan utilizando los `SW[7:0]`, que definen la representación binaria del número, seguido del pulso de `KEY[1]`. El software deberá realizar el reemplazo sólo al detectar un cambio en el valor del puerto.

El sistema mostrará cíclicamente, mediante LEDs y displays de 7 segmentos, los 8 valores almacenados ordenados de manera ascendente. Cada 1 segundo, mostrará automáticamente el siguiente número, de acuerdo al ordenamiento. Al llegar al final de la secuencia, inicia nuevamente desde el menor elemento.

1. Configure un proyecto con la herramienta `DE1SoC_SystemBuilder.exe`, contemplando el uso de *clock*, pulsadores, llaves, LEDs y displays de 7 segmentos. Abra el proyecto en Quartus Prime.

2. Mediante Plataform Designer, integre un sistema SoPC compuesto por módulo de *clock* a 50Mhz, procesador soft-core Nios II Gen2, memoria, puertos PIO de entrada y salida para interfaz con el módulo C (actividad 2), un puerto PIO de entrada adicional, un puerto de salida PIO adicional, una unidad JTAG UART, un *timer* configurado cada 1 segundo, y un System ID Peripheral.

   En relación a los PIOs de E/S conectan la interfaz del módulo C, para determinar las cantidades y anchos necesarios, se deberán distribuir los 64 pines de entrada y 64 pines de salida, teniendo en cuenta que, desde el software, se desea acceder a estos pines con arreglos. Por ello deberá existir continuidad en el mapa de memoria. Conecte los componentes, defina el mapa de memoria e interrupciones, y genere el código HDL.

3. ¿Qué alternativas existen para distribuir los pines de los puertos de E/S en el módulo C? ¿Qué ventajas y desventajas encuentra entre estas opciones, desde el punto de vista del programa que accede a esa región del mapa de memoria mediante? Explique.

4. Utilizando Quartus Prime, instancie el SoPC conectando la línea de clock a `CLOCK_50`, el *reset* a `KEY[0]`, los puertos PIO al módulo C de la Actividad 2, el puerto de entrada adicional a un registro de 8 bits (cuyas entradas a su vez se conectan a `SW[7:0]` y su *clock* a `KEY[1]`), y el puerto de salida adicional simultáneamente a `LEDR[7:0]` y al decodificador binario a BCD de la actividad 2.

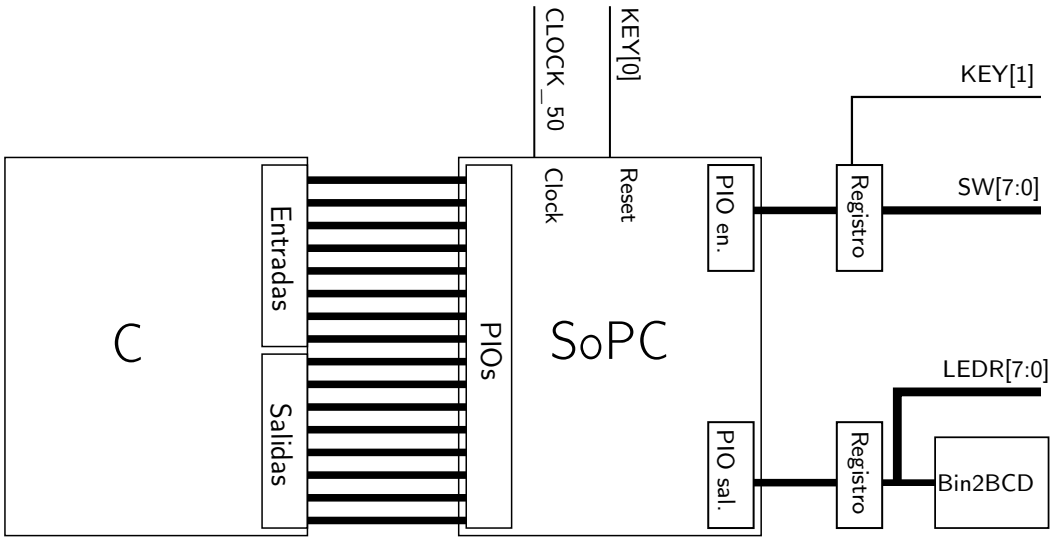
5. Sintetice el diseño. Verifique el mismo examinando las NetLists generadas.

6. Desarrolle un programa en C, que utilizando un BSP generado a partir del SoPC, implemente la funcionalidad del sistema. Identifique las tareas e ISRs del sistema. Utilice las interrupciones del timer para determinar qué número se muestra. Defina un algoritmo para la rotación de los números, que aproveche la estructura *butterfly* de ordenamiento.

7. ¿Qué ventajas y desventajas observa en contar con una unidad de ordenamiento por hardware, respecto de realizar una búsqueda u ordenamiento por software? Explique.

8. Construya el BSP y la aplicación.

9. Descargue el hardware y el firmware. Pruebe el sistema.



Referencias

[1] *DE1-SoC User Manual*. Terasic Technologies Inc.

[2] `www.terasic.com/download/cd-rom/de1-soc/DE1SoC_SystemBuilder.zip`. Terasic Technologies Inc.

[3] `Actividad1.v` y `Temporizador.v`, presente en la página de Sistemas Embebidos, en los archivos asociados al Laboratorio 6.

[4] `main.c`, presente en la página de Sistemas Embebidos, en los archivos asociados al Laboratorio 6.

[5] `nohal.c`, presente en la página de Sistemas Embebidos, en los archivos asociados al Laboratorio 6.

[6] *Tutorial: Quartus II Introduction Using Verilog Design*. Altera Corp., 2011.

[7] *Tutorial: Using Library Modules in Verilog Designs*. Altera Corp., 2011.

[8] *Chapter 9,10,11,12 y 13 Embedded SoPC Design with NIOSII Processor and Verilog*