UNIVERSIDAD PEDAGÓGICA Y TECNOLÓGICA DE COLOMBIA
FACULTAD SEDE SOGAMOSO
ESCUELA DE INGENIERÍA ELECTRÓNICA
LABORATORIO DE MICROPROCESADORES

PRACTICA N° 4

SISTEMAS EMBEBIDOS USANDO FPGAs: PROCESADOR NIOS II

I. OBJETIVOS

- Conocer los conceptos básicos relacionados a los sistemas embebidos con FPGAs.
- Desarrollar un sistema embebido que permita capturar, procesar y entregar datos, usando los software Quartus® II y Nios® II IDE.
- Implementar un sistema embebido en la tarjeta de desarrollo y educación: Altera DE1, considerando que esta tarjeta tiene como elemento central un FPGA.

II. INTRODUCCIÓN

Con el ánimo de satisfacer una necesidad de nuestra sociedad, es importante tener presente en el instante del desarrollo de un producto electrónico, que la portabilidad es un factor importante, aun así, esto no implica que la capacidad de procesamiento será reducida, por el contrario, tesis elaboradas en nuestra universidad han demostrado que un sistema embebido puede tener una capacidad de procesamiento, en una tarea específica, 10.000 veces mayor al otorgado por una laptop convencional.

Es por lo anterior, que un Ingeniero Electrónico al desarrollar un sistema embebido ha creado un dispositivo con un alto grado de portabilidad, mayor eficiencia, alta relación beneficio/costo y muchas otras características favorables.

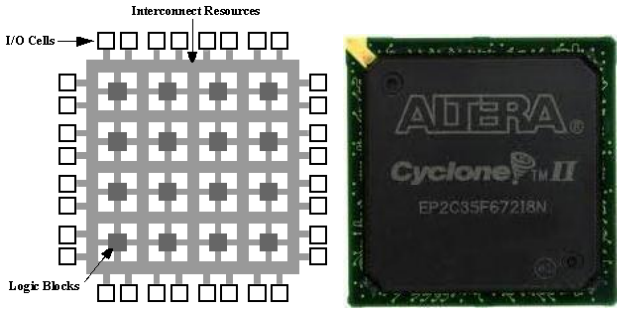


Fig 1: Arquitectura interna(Izquierda) y apariencia externa (derecha) de una FPGA.

Una FPGA (Field Programmable Gate Array, ver Figura 1) es un dispositivo semiconductor que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada 'in situ' (la programación *in situ* indica la capacidad de programar un dispositivo directamente en la etapa final, e inmediatamente antes de la ejecución [1]), todo esto mediante un lenguaje de descripción especializado, estos lenguajes de programación especiales son conocidos como HDL (lenguajes de descripción de hardware). Los HDLs más utilizados son:

- VHDL
- Verilog
- ABEL

La lógica programable permite reproducir funciones tan sencillas como las llevadas a cabo por una compuerta lógica hasta sistemas complejos.

Las FPGAs se utilizan en aplicaciones similares a los ASICs sin embargo son más lentas, tienen un mayor consumo de potencia y no pueden abarcar sistemas tan complejos como lo hacen los ASICs. A pesar de esto, las FPGAs tienen las ventajas de ser reprogramables (lo que añade una enorme flexibilidad al flujo de diseño), sus costes de desarrollo y adquisición son mucho menores para pequeñas cantidades de dispositivos y el tiempo de desarrollo es también menor [2].

SISTEMA EMBEBIDO

Un sistema embebido es una combinación de hardware y software, diseñados con el objetivo de realizar una función específica [3]. Expresado de otro modo, es un sistema computación diseñado para realizar funciones dedicadas, frecuentemente es un sistema de computación en tiempo real [4]. Los sistemas embebidos se diseñan para cubrir necesidades específicas, al contrario de lo que ocurre con los ordenadores de propósito general (por ejemplo una computadora personal o PC) que están diseñados para cubrir un amplio rango de necesidades.

Por lo general los sistemas embebidos se pueden programar en el lenguaje ensamblador del microcontrolador o microprocesador incorporado, o utilizando lenguajes de alto nivel.

En un sistema embebido la mayoría de los componentes se encuentran incluidos en la placa base (memorias, puertos, tarjetas de vídeo, audio, módem, etc) y muchas veces los dispositivos resultantes no tienen el aspecto de una computadora convencional como en el ejemplo de la Figura 2.

Fig 2: Ejemplo de sistema embebido.

III. HARDWARE

- Tarjeta de desarrollo y educación: Altera DE1 – DE2.

IV. SOFTWARE

- Quartus® II Web Edition V13 O superior.
- Nios® II Embedded Design Suite

Nota: Puede encontrar el software en la referencia [5].

V. DESCRIPCIÓN DEL

En el presente laboratorio se debe desarrollar un sistema embebido, primero se sintetiza y programa un sistema de 8 switches y 8 leds como modo de prueba y comprensión del procedimiento de síntesis y generación del código para el funcionamiento.

El laboratorio consta de tres partes:

1. Desarrollo del Hardware del sistema.
2. Desarrollo del Software del sistema.
3. Practica.

PRIMERA PARTE: DESARROLLO DEL HARDWARE

1. Inicie la aplicación Quartus® II Web Edition.

   Importante: Este proyecto debe estar ubicado en una ruta que no contenga espacios, es aconsejable que cree un directorio de trabajo para sus desarrollos en Quartus II.

2. Cree una carpeta dentro de su directorio de trabajo.

3. Diríjase a *"File/New Project Wizard…"* y cree un nuevo proyecto en la carpeta recientemente creada.

   Seleccione el dispositivo de acuerdo a la tarjeta de desarrollo que se desea usar.

   - DE1- Familia "*Cyclone II*" y su nombre es "*EP2C20F484C7*".
   - DE2- Familia "Cyclone IV E" y su nombre "EP4CE115F29C7"

4. Elabore un nuevo diagrama de bloques.

   En "File/New…" Expanda la opción "*Desing Files"* y cree un "*Block Diagram/Schematic File*"

   En "*File/Save As…*" guarde el diagrama de bloques con el mismo nombre de su proyecto.

5. Abra la herramienta Qsys Builder en "*Tools/Qsys*"

Del panel de la izquierda: "C*omponent Library*" (ver Fig 3), se agregan los componentes que se indican en el párrafo siguiente, realizando los cambios indicados, y una vez realizados los cambios, presione "*Finish*" para pasar a agregar otro componente.

NOTA: puede que se le indique presionar "*Finish*", y posteriormente realizar otro cambio antes de insertar el siguiente componente.

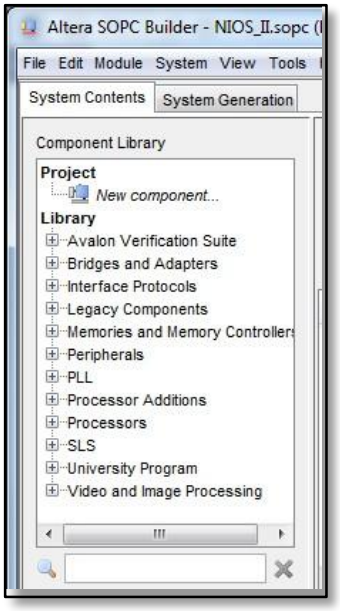


Fig 3: Librerías de la herramienta Qsys.

Listado de Componentes que debe insertar:

- *Memories and Memory Controller/On-Chip/On-Chip Memory (Ram or Rom).*

En "*Total memory size" digite "204800"*

Renombre el componente ubicándose sobre él y presionando "CTRL-R", asígnele un nombre que describa su función en este caso Ram.

- *Processors/Nios II Processor.*

  Conecte las señale de *clk, reset1 y s1* de la memoria ram a las señales *clk, reset_n, data_master e instruction_master* del procesador respectivamente.

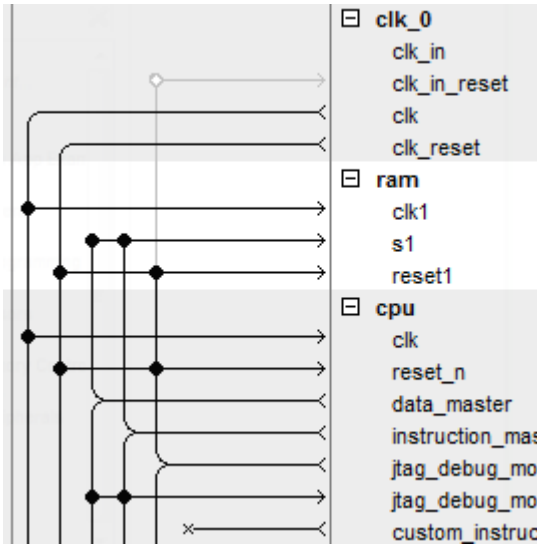


Fig 4: Conexión de la ram, cpu y fuente de reloj.

En la ventana de configuración de la CPU elegimos de "*Reset Vector*" y de "*Exception Vector*" laopción desplegable: "*ram.s1*"

- *Interface_Protocols/Serial/JTAG UART.*

  Esta ultima es la interfaz de programación y depuración del procesador Nios II.

- *Peripherials/MicrocontrollerPeripherias/ PIO (Parallel I/O)*

  Seleccione "*Input ports only*", presione "*Finish*" y renombre el componente como "*pio_switch*".

- *Peripherials/Microntroller Peripherials/PIO (parallel I/O)*

  Seleccione "*Output ports only*", presione "*Finish*" y renombre el componente como "*pio_led*".

- *Qsys Interconnect/Tri-state components /Generic Tri-state Controller.*

  En "*Address Width (bits):*" seleccione 23, posteriormente de click en la pestaña de "signal *Timing*" y digite 160,160,60 y 60 en los campos "Read wait time", "Write wait time","*Setup time*","data hold time" respectivamente. Renombrar el componente con "cfi_flash".

- *Qsys interconnect/Tri-state components/Tri-state conduit bridge.*

- *Qsys interconnect/Tri-state components/Tri-state conduit pin sharer.*

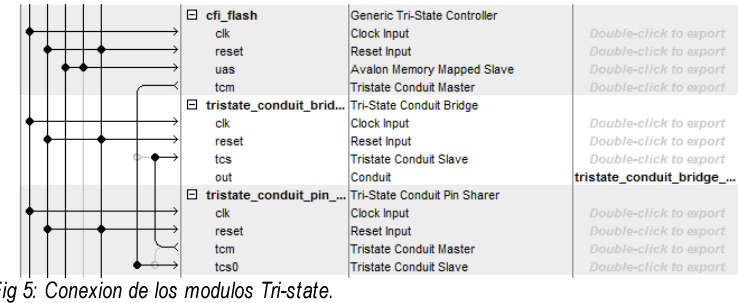


Fig 5: Conexion de los modulos Tri-state.

En la *Fig **5*** se observa las conexiones que se deben realizar en los componentes tri-estados agregados anteriormente. Despues abrimos la ventana *"tri-state conduit pin sharer" y en "number interfaces"* colocamos 1 y damos click en "*update interface table" y terminanos dando click en finish.*

- *Peripherals/Debug and Performance/System ID Peripheral.*

  Presione "*Finish*" y renombre el componente como "*sysid*"

Ya se han agregado todos los componentes, ahora en el menú "*System/Assing Base Addresses*" permitirá que el sistema genere todas las direcciones base de los componentes de manera automática. Para generar las direcciones de las interrupciones en la columna IRQ de la interfaz de conectamos las interrupciones disponibles y por ultimo "*system/Assign interrupt number*".

Para generar el sistema embido diseñado vamos a la pestaña "*generation*" y en el campo "synthesis" se selecciona VHDL y terminamos dando click en "*Generate*" y esperamos que en la ventana emergente no nos active el botón "close" y que no se generen errores.

6. Retornamos al entorno de desarrollo de Quartus y en el "block diagram" agregamos el bloque del sistema embebido creado con el Qsys.

Fig 6: Iconos para agregar el sistema creado y los pines.

Dé click en el primer botón de *Fig 6* "*Symbol Tool*" y agregue de la carpeta "*Projects*" el procesador generado. Con el segundo boton de la *Fig 6* agregue los pines de entrada o salida según corresponda, tenga encuenta que los pines de datos de la memoria flash cfi_flash_*tcm_data_out[7..0]* es bidireccional. Adicionalmente agregue dos pines de salida y conéctelo a Vcc (../primitives/other/vcc), el diagrama de bloques resultante lo puede observar en el ANEXO 1.

7. Compilación: Antes de poder compilar hay que agregar los archivos VHDL que describen el sistema embebido diseñado. Debemos dar click en "*Assignments/setting../file/add_all/OK*". Compilando dando click en "*start compilation*".

8. Pin planer: Realice la asignación de pines de acuerdo a las especificaciones de cada tarjeta, debe considerar que en cada una hay una memoria flash diferente y que una de estas cuenta con un pin extra.

- Clk: Cristal de 50 Mhz de la tarjeta.
- Reset_n: Pulsador.
- Leds: leds de la tarjeta.
- Switchs: Switch de la tarjeta.
- cfi_flash_tcm_write_n_out: FL_WE_N.
- cfi_flash_tcm_read_n_out: FL_OE_N
- cfi_flash_tcm_chipselect_n_out: FL_CE_N
- FL_RST: FL_RST_N
- FL_WP_N: FL_WP_N
- cfi_flash_tcm_data_out: FL_DQ[7..0]
- cfi_flash_tcm_addres_out: FL_ADDR[23..0].

Vuelva a compilar el proyecto y cárguelo en la tarjeta de desarrollo. Para esto conecte la tarjeta y en "*tolos/programer*" selecione el driver "*USB-blaster*" y agregue el archivo *.sof* correspondiente al proyecto de click en *start.*

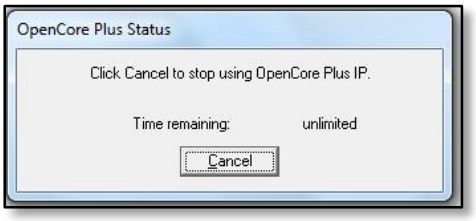


Fig 7: Ventana de carga exitosa del sistema embebido.

En este paso se ha cargado el diseño del sistema embebido diseñado usando Qsys. Recuerde no cerrar la ventana emergente de la *Fig 7*.

SEGUNDA PARTE: DESARROLLO DEL SOFTWARE.

1. Iniciar Nios II IDE. "*tolos/Nios II software buid tolos for eclipse".*

2. Seleccione como espacio de trabajo la carpeta donde creo el proyecto Quartus.

3. Creamos un proyecto nuevo en eclipse: "*File/New/Nios II Application and BSP from template".* En la ventana emergente damos el nombre del proyecto y seleccionamos el archivo descriptor del sistema embebido "*SOPC information file name",* diseñado con Qsys para que el IDE nos genere automáticamente los *headers (.h)* y código fuente (*.c)* necesario para compilar y generar los archivos (*.elf y bin)* usados para la depuración y ejecución del programa por el procesador del sistema embebido, Presione "*Finish*".

4. Borre el contenido del archivo emergente, y copie las líneas del ANEXO 2, analícelas.

5. En el "*Project Explorer*" seleccione la carpeta con el nombre de su proyecto y de click derecho y en el menú desplegable seleccione "*build Project".* Verifique que la compilación fue exitosa y en caso contrario verifique cual es la fuente del error según el compilador y siga los pasos que el arroja para solucionar el error.

6. Programación de la memoria flash: En esta memoria se guardara el código binario del programa desarrollado. Para su programación se debe indicarle al IDE el módulo de programación. Vamos a la barra de menús y seleccionamos "R*un/run" o "CTRL+F11".*

   Este operación compilara el proyecto e intentara cargar el proyecto en la memoria flash y al no encontrar configurar el modulo de programación no abre una ventana emergente que nos permite configurarlo.

   En el anexo 3 se observa la ventana de configuración para programar la memoria flash, damos click en "*Refresh connections"* y el programa nos detecta automáticamente la configuración del módulo de programación de la tarjeta. Debemos tener conectada la tarjeta y con el sistema embebido sintetizado previamente en la tarjeta para que el programa pueda detectar el módulo de programación.

7. Volvemos a compilar con "CTRL+F11" o "Run", el programa debe compilar y cargar automáticamente en la memoria flash del sistema embebido el código binario del programa desarrollado.

TERCERA PARTE: PRACTICA.

Los grupos deben modificar el sistema embebido diseñado usando Qsys, para que este sea compatible con una pantalla LCD 16x2 (agregar controlador disponible en la librería) y un teclado matricial.

Para estos nuevos componentes se deben desarrollar unas librerías siguiendo los conceptos de las librerías HAL (*Hardware abstraction layer)*. Que permitan desarrollar programas de manera rápida y con código limpio en el que el control de los periféricos se hace de manera transparente al desarrollador.

Con estos periféricos se debe desarrollar un reloj y un calendario que calcule el día que corresponde para la fecha ingresada con el teclado matricial. El reloj debe ser totalmente configurable y cuando este cambie de día el algoritmo debe recalcular el día correspondiente.

VI. BIBLIOGRAFÍA

[1] http://es.wikipedia.org/wiki/In_situ.

[2] http://es.wikipedia.org/wiki/FPGA.

[3] http://www.barrgroup.com/Embedded-Systems/Glossary-E.

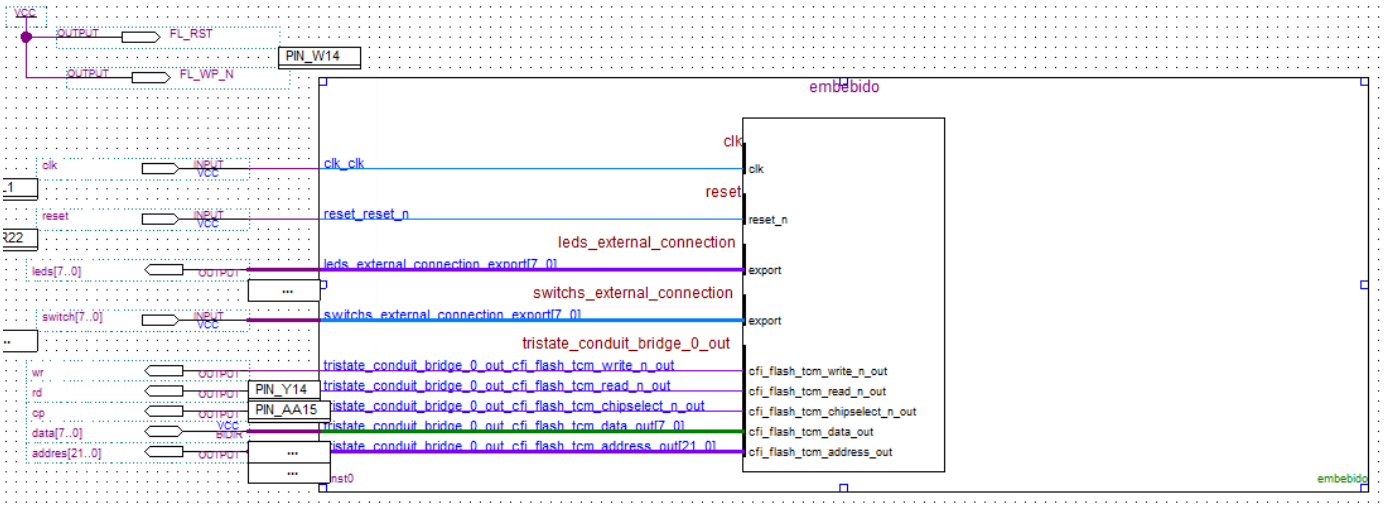
[4] http://es.wikipedia.org/wiki/Sistema_embebido.

[5] https://www.altera.com/support/software/download/nios2/dnl-nios2.jsp.

[6] Altera, DE1_UserManual_v1018.pdf

[7] http://www.altera.com/literature/lit-nio2.jsp.

VII. [illegible]



ANEXO 1: Diagrama de bloques del procesador NIOSII.

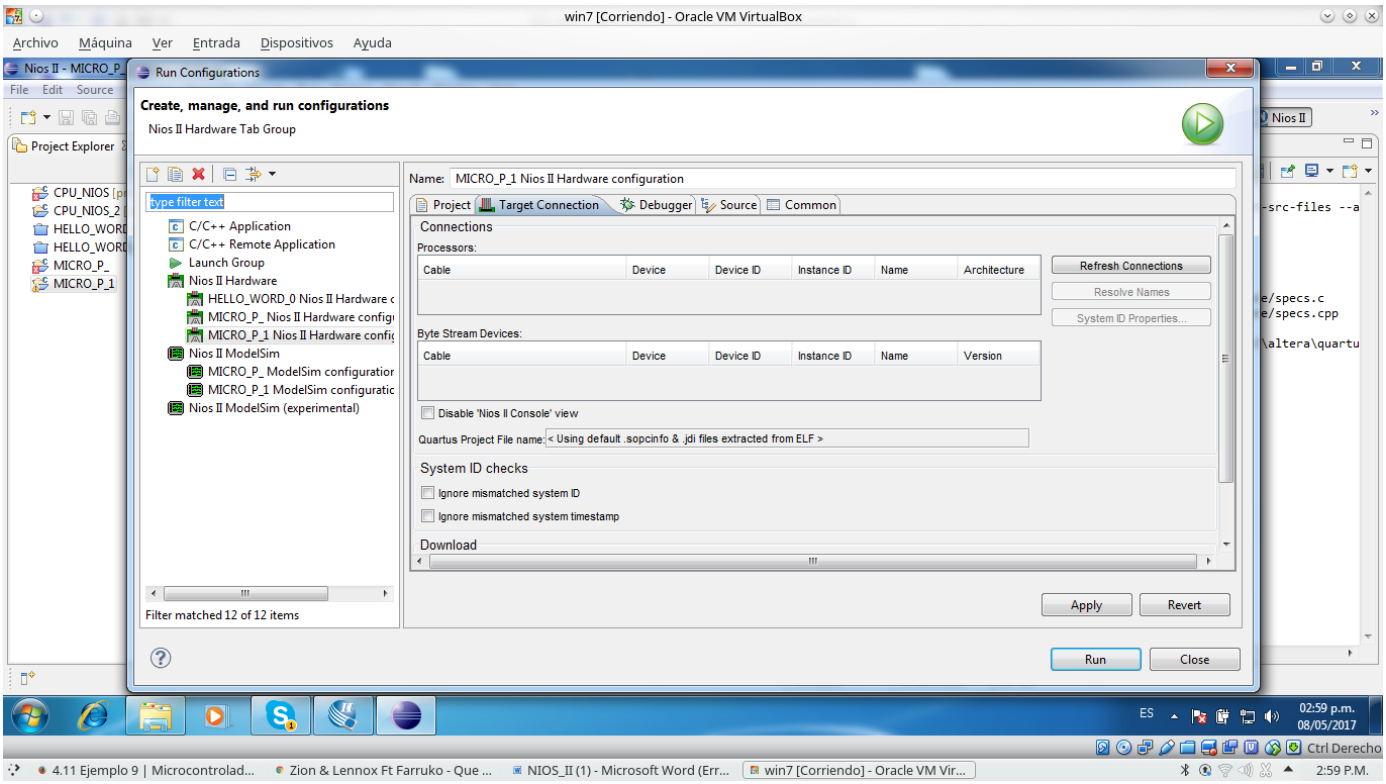
ANEXO 2: Código en C++ para el desarrollo del software del NIOS II.

```
#include "system.h"
#include "altera_avalon_pio_regs.h"

int main()
{
    int count,delay;
    count=0;
    delay=0;
    while(1)
    {
        count=IORD_ALTERA_AVALON_PIO_DATA(PI
        O_SWITCH_BASE);
        IOWR_ALTERA_AVALON_PIO_DATA(PIO_L
        ED_BASE,count); while (delay<1000000)
        {
            delay=delay+1;
        }
        delay=0;

    }
}
```

UNIVERSIDAD PEDAGÓGICA Y TECNOLÓGICA DE COLOMBIA
FACULTAD SEDE SOGAMOSO ESCUELA DE
INGENIERÍA ELECTRÓNICA LABORATORIO
DE MICROPROCESADORES



ANEXO 3: VENTANA DE CONFIGURACIÓN DE MEMORIA FLASH.