

Procedimiento para almacenar la plataforma de hardware basado en NIOS II y la aplicación de usuario (software) desarrollado en Eclipse en la memoria EPICS4

Por: Kalun José Lau Gan

2021

1

Revisión (changelog) del documento:

- 23-mayo-2021: Se logró programar la EPICS4 para que configurara la plataforma NIOS II al FPGA, copiara el programa a la memoria RAM e iniciara la aplicación.
- 24-mayo-2021: Creación del documento
- 25-mayo-2021: Redacción inicial del documento

2

Aspectos iniciales:

- En el desarrollo de la plataforma NIOS II se validó la implementación de su hardware y su aplicación (software) desarrollada en C en una FPGA, pero éstos al estar basados en memoria RAM pierden su contenido de configuración cuando se le desconecta la energía eléctrica.
- Las tarjetas de desarrollo basadas en FPGA por lo general siempre vienen con una memoria externa no volátil por lo que en muchos de los casos se emplea esta última para almacenar la configuración de hardware del FPGA. Cuando se energiza la tarjeta lo primero que hará la FPGA es consultar en la memoria externa si tiene alguna configuración a ser transferida, si la hay obtendrá dicha configuración y terminada la transacción empezará a operar.

3

Aspectos iniciales:

- Se está empleando el entorno de software Quartus II versión 13.0 SP1.
- El procedimiento explicado por Intel (ex Altera) en sus documentos técnicos no está del todo claro, cuando se usa el Flash Programmer no funciona. Para ello se tendrá que generar los archivos de programación utilizando la ventana de comandos del NIOS II.

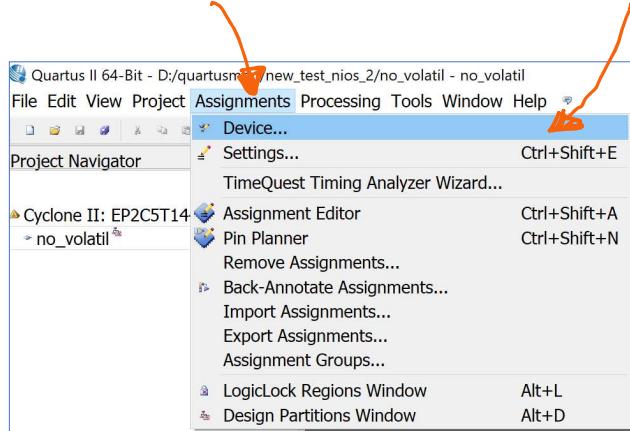


- Este procedimiento es para circuitos o tarjetas de desarrollo que tienen una memoria de configuración no volátil EPCS4 conectado a un FPGA Cyclone II, para alojar tanto la configuración del hardware como la aplicación que correrá el procesador NIOS II.

4

Procedimiento:

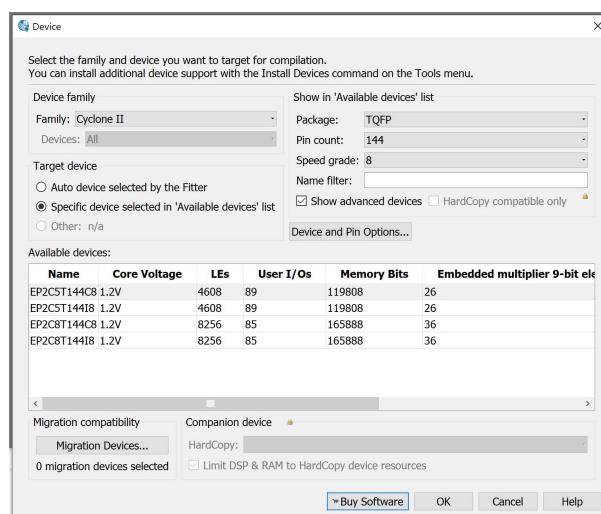
Luego de creado el proyecto en el Quartus II configurar los siguientes parámetros de puerto:



5

Procedimiento

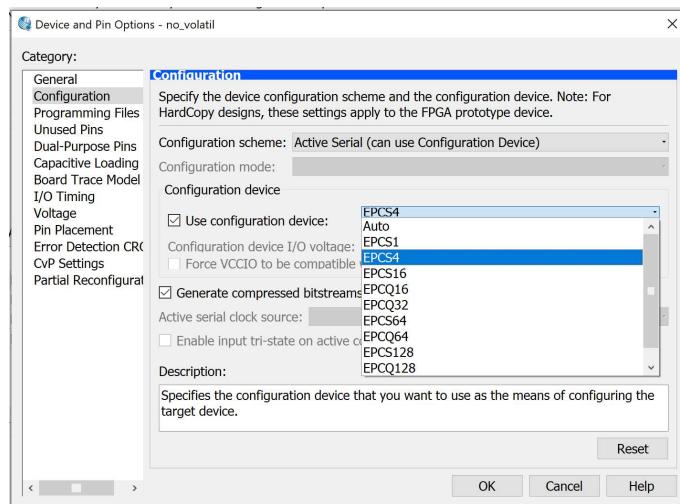
- Hacer clic en el botón “Device and Pin Options”



6

Procedimiento

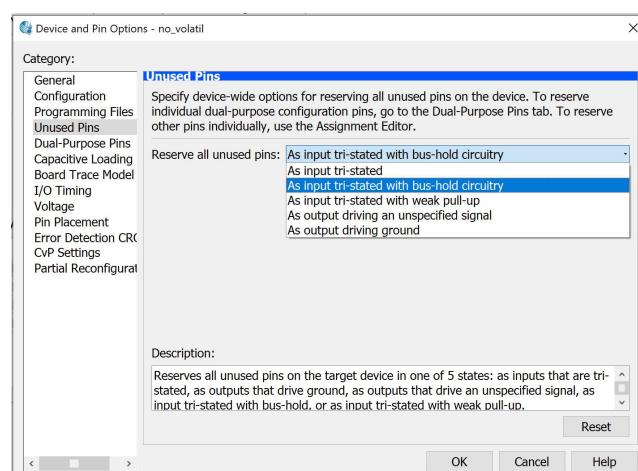
- Seleccionar en el lado izquierdo “Configuration”
- Seleccionar el dispositivo (memoria) de configuración EPCS4



7

Procedimiento

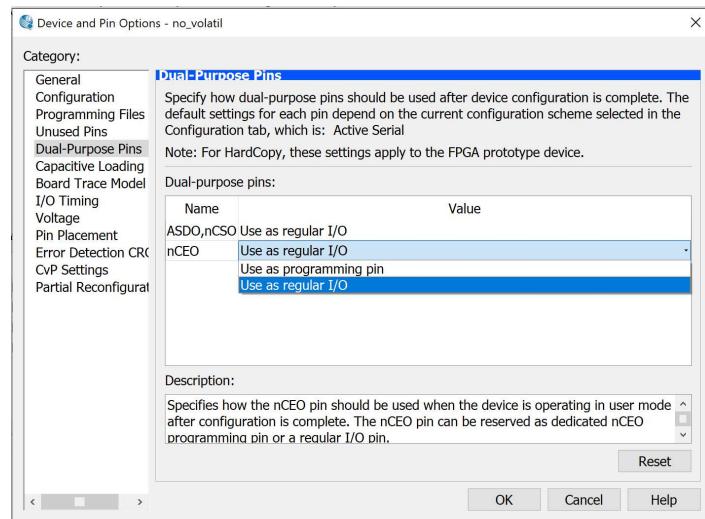
- Seleccionar en el lado izquierdo “Unused Pins”
- Seleccionar al lado derecho la opción de Reserve all unused pins: “As input tri-stated with bus-hold circuitry”



8

Procedimiento

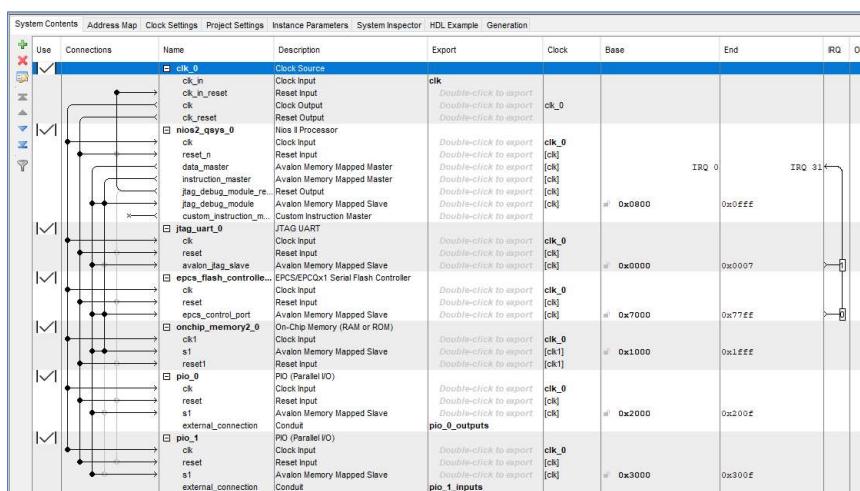
- Seleccionar en el lado izquierdo “Dual-Purpose Pins”
- Colocar todos los pines del lado derecho con la opción “Use as regular I/O”



9

Procedimiento

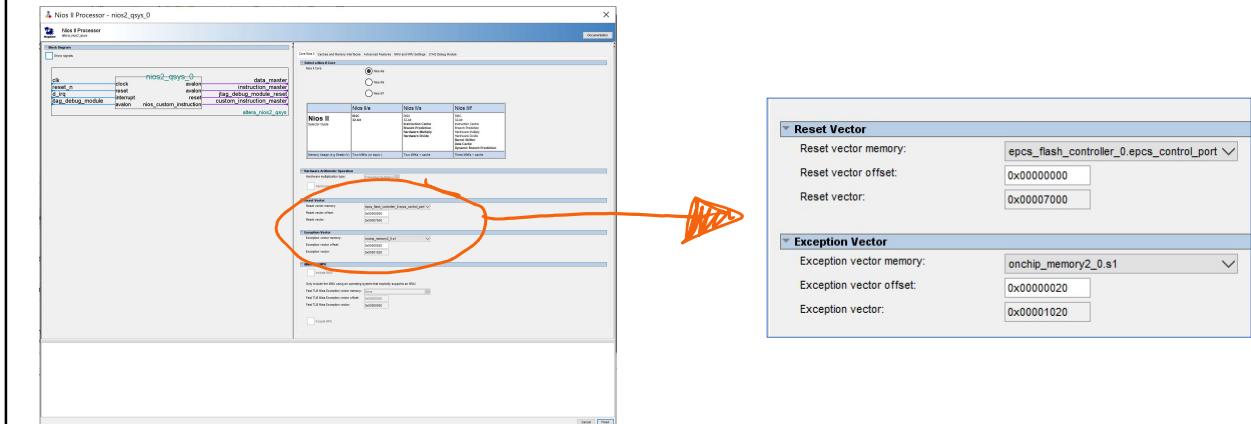
- En el Qsys se llamará al módulo EPCS/EPCQx1 Serial Flash Controller, conectarle los buses Avalon de instrucciones y datos provenientes del procesado NIOSII
- Colocar dicho módulo en la dirección 0x7000 (muy importante!)



10

Procedimiento

- En las propiedades del procesador NIOS II/e colocar:
 - Reset Vector con el epc5_flash_controller
 - Exception Vector seguirá apuntando a onchip_memory



11

Procedimiento

- En el Qsys generar el sopcinfo.
- Regresar al Quartus y crear el VHDL estructural donde se instanciará el procesador NIOS II

```

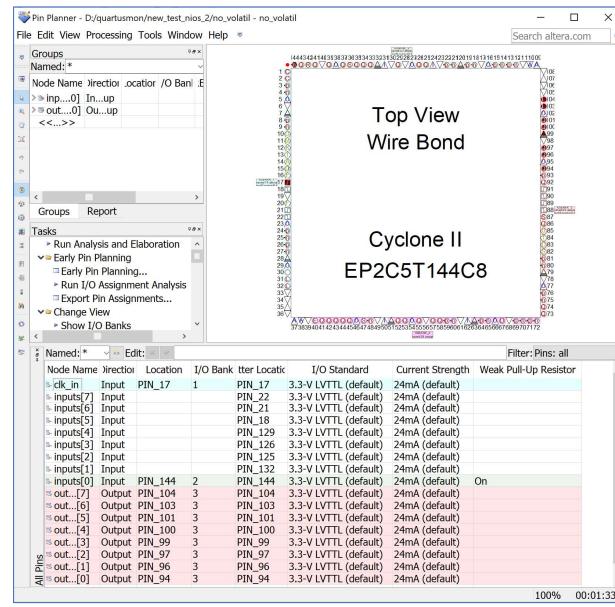
1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity no_volatile is
5   port( Clk_in: in std_logic;
6        outputs: out std_logic_vector(7 downto 0);
7        inputs:  in std_logic_vector(7 downto 0));
8 end no_volatile;
9
10 architecture estructura of no_volatile is
11
12   component nios_core is
13     port (
14       clk_clk           : in std_logic := 'X';
15       pio_0_outputs_export : out std_logic_vector(7 downto 0) := 'X';
16       pio_1_inputs_export : in std_logic_vector(7 downto 0) := (others => 'X');
17     );
18   end component nios_core;
19
20 begin
21   u0 : component nios_core
22     port map (
23       clk_clk          => clk_in, -- |clk.clk
24       pio_0_outputs_export => outputs, -- pio_0_outputs.export
25       pio_1_inputs_export  => inputs  -- pio_1_inputs.export
26     );
27 end estructura;

```

12

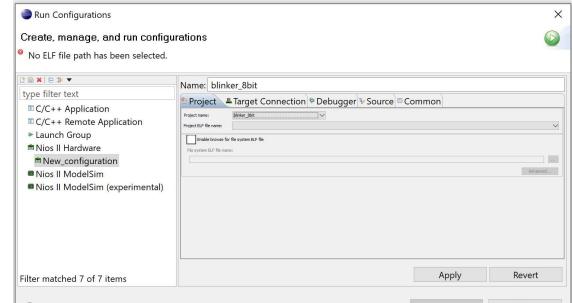
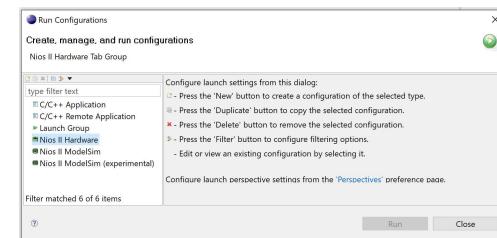
Procedimiento

- Revisar la asignaciones de pines del FPGA para con el proyecto.
- Toda modificación en el PinPlanner deberá de acompañarse después con un proceso de Compilación en el Quartus II
- Proceder a realizar una primera grabación del FPGA con el Programmer del Quartus empleando el archivo sof generado.

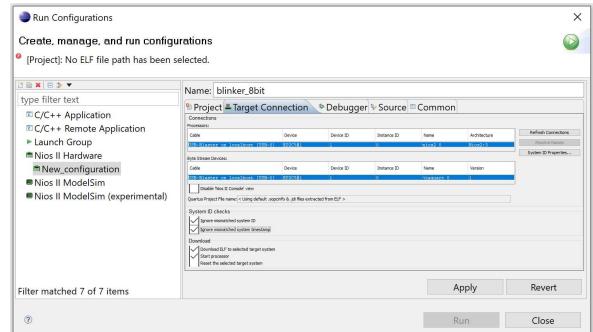


13

Procedimiento



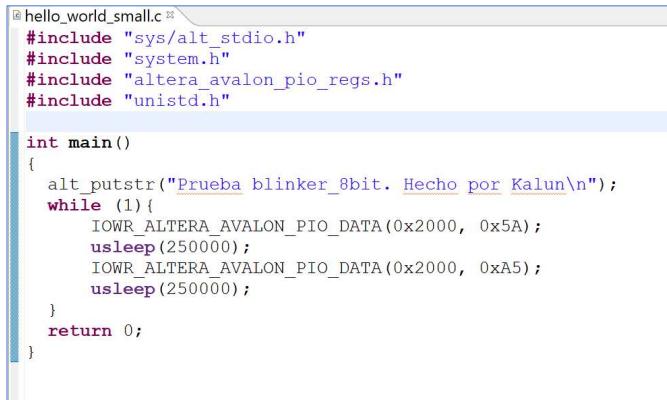
- En el Eclipse luego de crear el proyecto establecer la conexión con el NIOS II mediante la opción Run Configurations



14

Procedimiento

- Este es el programa ejemplo en C que correrá el procesador NIOSII
- Se está empleando la librería “unistd.h” para suprimir el mensaje de advertencia por usar “usleep”



```

hello_world_small.c
#include "sys/alt_stdio.h"
#include "system.h"
#include "altera_avalon_pio_regs.h"
#include "unistd.h"

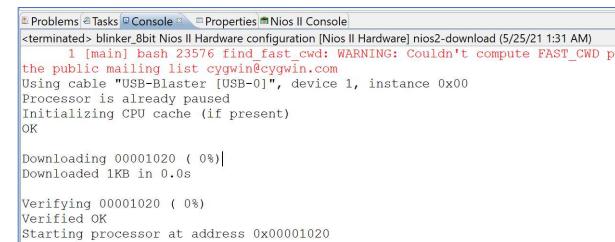
int main()
{
    alt_putstr("Prueba blinker_8bit. Hecho por Kalun\n");
    while (1){
        IOWR_ALTERA_AVALON_PIO_DATA(0x2000, 0x5A);
        usleep(250000);
        IOWR_ALTERA_AVALON_PIO_DATA(0x2000, 0xA5);
        usleep(250000);
    }
    return 0;
}

```

15

Procedimiento

- Proceder a armar el proyecto usando la opción “Build Project” para generar el archivo ELF.
- Correr el programa en el FPGA mediante la opción “Run As... NIOS II Hardware”



```

Problems Tasks Console Properties Nios II Console
<terminated> blinker_8bit Nios II Hardware configuration [Nios II Hardware] nios2-download (5/25/21 1:31 AM)
  1 [main] bash 23576 find_fast_cwd: WARNING: Couldn't compute FAST_CWD path
the public mailing list cygwin@cygwin.com
Using cable "USB-Blaster [USB-0]", device 1, instance 0x00
Processor is already paused
Initializing CPU cache (if present)
OK

Downloading 00001020 ( 0%)
Downloaded 1KB in 0.0s

Verifying 00001020 ( 0%)
Verified OK
Starting processor at address 0x00001020

```



```

Problems Tasks Console Properties Nios II Console
blinker_8bit Nios II Hardware configuration - cable: USB-Blaster on localhost [USB-0] device ID: 1 instance ID: 0 name: jtjuart_0
Prueba blinker_8bit. Hecho por Kalun

```

16

Procedimiento

- Detener la comunicación JTAG UART con el “NIOS II Console” antes de seguir con los siguientes pasos.

```

Nios II - blinker_8bit/hello_world_small.c - Eclipse
File Edit Source Refactor Navigate Search Run Project Nios II Window Help
Project Explor... > hello_world_small.c
  Binaries
  Includes
  Obj
  System
  hello_world_small.c
  blinker_bitelf
  blinker_bitmap
  blinker_bitobjdump
  create-this-app
  Makefile
  README.txt
  blinker_bitlisp [ ]
  
```

```

#include "sys/alt_stdio.h"
#include "altera_avalon_pio_regs.h"
#include "unisysd.h"

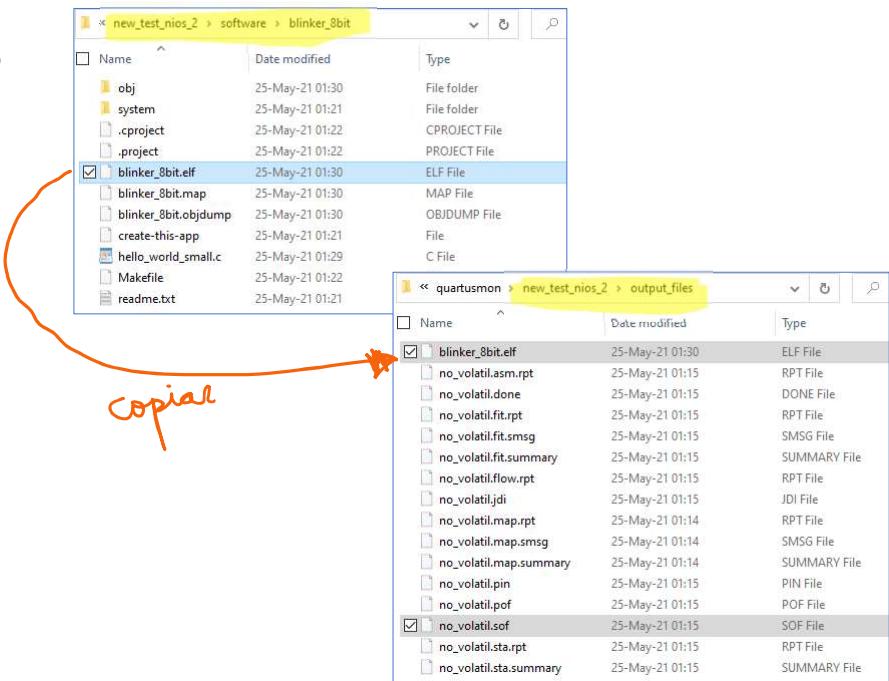
int main()
{
    alt_putstr("Prueba blinker_8bit. Hecho por Kalman\n");
    while (1) {
        IOMR_ALTERA_AVALON_PIO_DATA(0x2000, 0xA5);
        usleep(250000);
        IOMR_ALTERA_AVALON_PIO_DATA(0x2000, 0xA5);
        usleep(250000);
    }
    return 0;
}
  
```

Problems Tasks Properties Nios II Console

17

Procedimiento

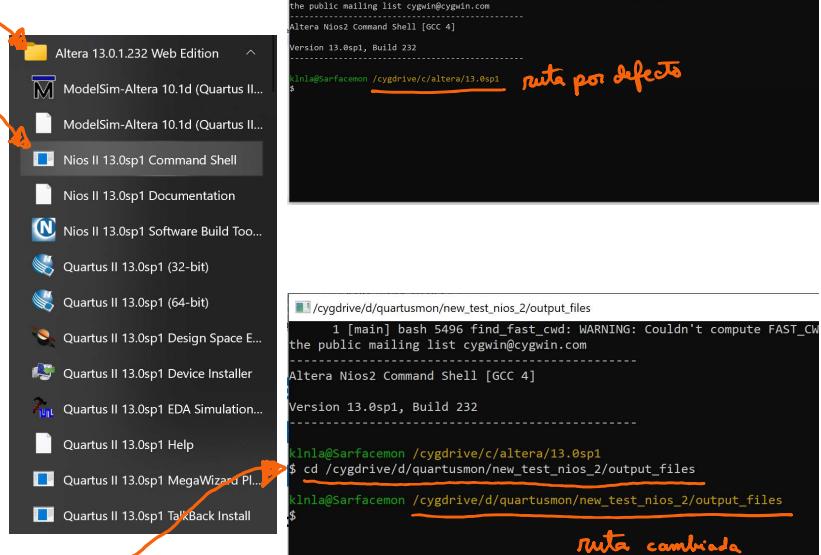
- Abrir una ventana de File Explorer
- Dirigirse a la carpeta en donde se encuentre el archivo compilado *.elf por el software Eclipse
- Copiar el archivo *.elf hacia la carpeta output_files donde se encontrará el archivo *.sof generado por el Quartus II



18

Procedimiento

- En el menú de aplicaciones de Windows ingresar a la carpeta “Altera 13.0.1.232 Web Edition” y abrir el Nios II 13.0sp1 Command Shell.
- Cambiar la ruta hacia la carpeta output_files dentro de la carpeta del proyecto utilizando comandos similares a Linux.



19

Procedimiento

- Estos tres comandos que deberán ser ejecutados en el Command Shell de manera secuencial permitirán obtener el archivo HEX necesario para adjuntarlo en el siguiente proceso de conversión de archivos de programación.

```
sof2flash --input=[nombre del hw].sof --output=[nombre del hw].flash --epcs --verbose
elf2flash --input=[nombre del sw].elf --output=[nombre del sw].flash --epcs --after=[nombre del hw].flash --verbose
nios2-elf-objcopy --input-target srec --output-target ihex [nombre del sw].flash [nombre del sw].hex
```

[nombre del hw] será el nombre de tu archivo sof generado por el Quartus II al hacer un "Compile"
 [nombre del sw] será el nombre de tu archivo elf generado por el Eclipse al hacer un "Build Project"

- Para el ejemplo:

```
sof2flash --input=no_volatile.sof --output=no_volatile.flash --epcs --verbose
elf2flash --input=blinker_8bit.elf --output=blinker_8bit.flash --epcs --after=no_volatile.flash --verbose
nios2-elf-objcopy --input-target srec --output-target ihex blinker_8bit.flash blinker_8bit.hex
```

20

Procedimiento

- Es importante que estos tres comandos se ejecute de manera correcta como se muestra a continuación:

```

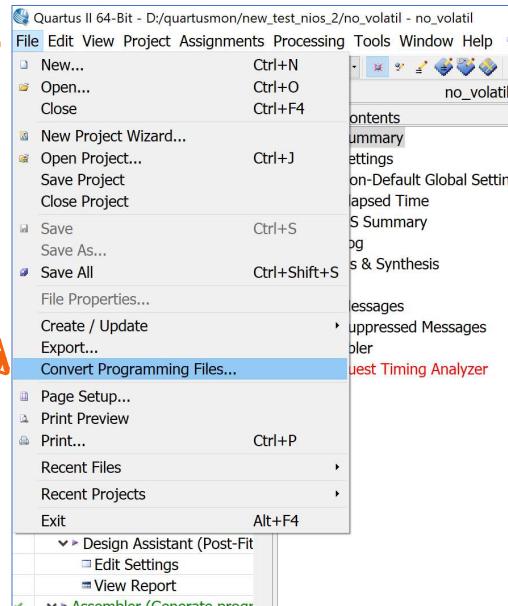
1 [main] bash 5496 find_fast_cwd: WARNING: Couldn't compute FAST_CWD pointer. Please report this problem to the public mailing list cygwin@cygwin.com
-----
Altera Nios2 Command Shell [GCC 4]
Version 13.0sp1, Build 232
-----
klmla@Sarfacemon /cygdrive/c/altera/13.0sp1
$ cd /cygdrive/d/quartusmon/new_test_nios_2/output_files
klmla@Sarfacemon /cygdrive/d/quartusmon/new_test_nios_2/output_files $ sof2flash --input=no_volatile.sof --output=no_volatile.flash --epcs --verbose
May 25, 2021 1:54:52 AM - (FINE) sof2flash: Starting
May 25, 2021 1:54:59 AM - (FINE) sof2flash: Done
-----
klmla@Sarfacemon /cygdrive/d/quartusmon/new_test_nios_2/output_files $ elf2flash --input=blinker_8bit.elf --output=blinker_8bit.flash --epcs --after=no_volatile.flash --verbose
May 25, 2021 1:55:08 AM - (INFO) elf2flash: args = --input=blinker_8bit.elf --output=blinker_8bit.flash --epcs --after=no_volatile.flash --verbose
May 25, 2021 1:55:08 AM - (FINE) elf2flash: Starting
May 25, 2021 1:55:08 AM - (FINER) elf2flash: Program Record: 928 bytes destined for 0x1020
May 25, 2021 1:55:08 AM - (FINER) elf2flash: Start Record: 1020
May 25, 2021 1:55:08 AM - (FINE) elf2flash: Done
-----
klmla@Sarfacemon /cygdrive/d/quartusmon/new_test_nios_2/output_files $ nios2-elf-objcopy --input-target srec --output-target ihex blinker_8bit.flash blinker_8bit.hex
klmla@Sarfacemon /cygdrive/d/quartusmon/new_test_nios_2/output_files $

```

21

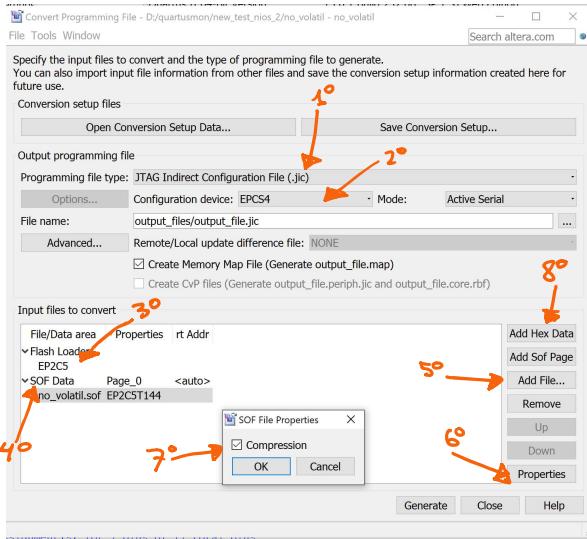
Procedimiento

- 1º Revisar si se generó el archivo HEX luego de haber ejecutado los tres comandos en el Command Shield.
- 2º Dirigirse al Quartus II y abrir el proceso “Convert Programming Files...”

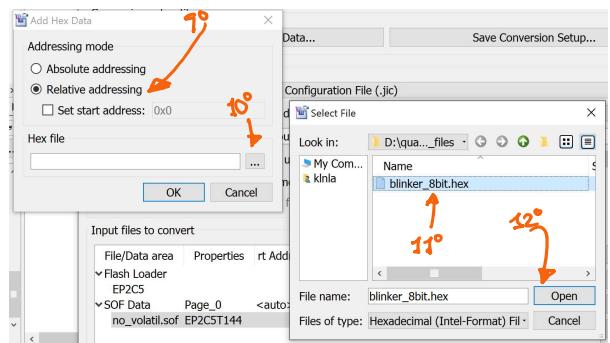


22

Procedimiento



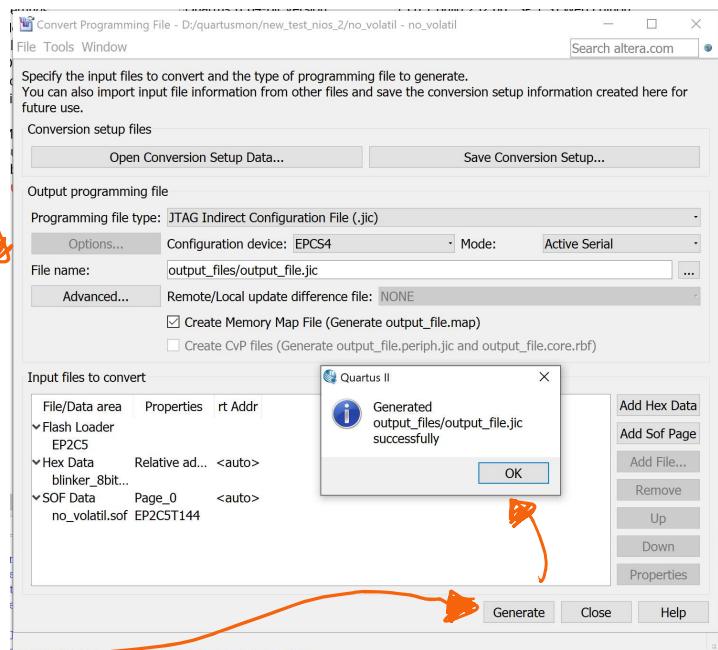
- Seleccionar el tipo de archivo de programación a JTAG Indirect Configuration File (.jic)
- Seleccionar el dispositivo EPCS4
- En Flash Loader Seleccionar el dispositivo FPGA Cyclone II EP2C5
- En SOF Data escoger el archivo sof generado por la compilación del proyecto en el Quartus II.
- Entrar a propiedades del SOF y habilitar la compresión.
- Agregar en Hex Data el archivo HEX generado con los tres comandos en el Command Shell y seleccionar Relative addressing



23

Procedimiento

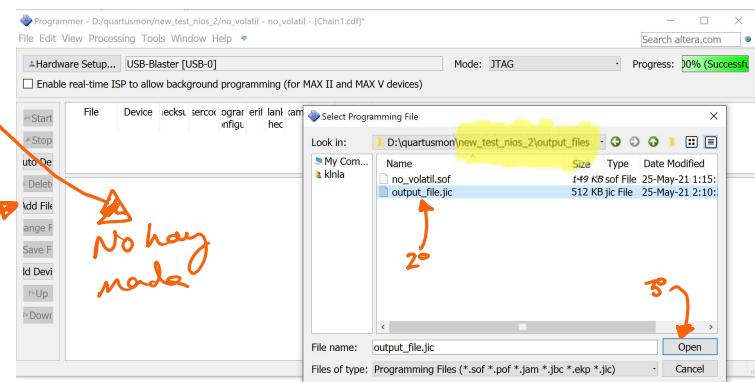
- Revisar que se haya seleccionado las configuraciones y archivos correctos.
- Hacer clic en "Generate" y esperar el mensaje de generación satisfactoria, se habrá generado el archivo JIC necesario para el siguiente paso



24

Procedimiento

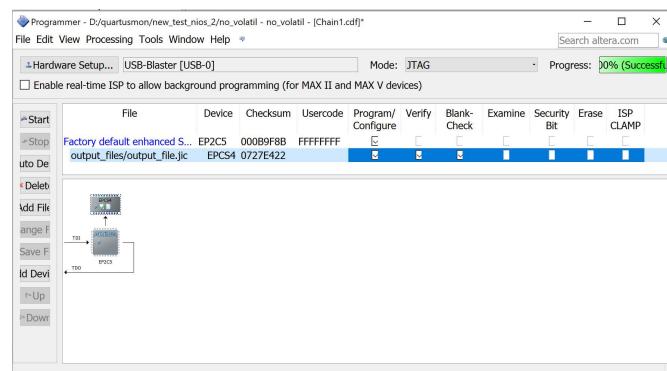
- Abrir nuevamente el programador del Quartus II
- Borrar todo dispositivo presente
- Agregar el archivo JIC generado



25

Procedimiento

- Proceder con el botón Start para grabar la memoria EPICS4 mediante JTAG Indirecto
- Esperar a que termine de grabar (barra de estado al 100%). Si es que no procede con la grabación puede que no se haya detenido la comunicación JTAG UART en el Eclipse, se debe de detener dicha comunicación para liberar el JTAG y permita la grabación de la EPICS4.
- Desconectar y conectar la tarjeta (cycle energy) y verificar si funciona como lo planeado.



26

Consideraciones finales

- Tener en cuenta que se ha utilizado en esta oportunidad el archivo QSYS en lugar del archivo QIP para llamar a la configuración del procesador NIOS II en el Quartus II (al momento de hacer Add File...)