Plataforma de hardware recofigurable

JTAG – Configuración OOCD-Links, (*Hardware & Software*)

Luis A. Guanuco

13 de junio de 2013

1. Introducción

Una gran parte del proyecto *Plataforma de Hardware Reconfigurable (PHR)* se basa en la implementación de códigos de programación en diferentes lenguajes. En esta etapa El presente reporte desarrolla la continuación del armado, testeo de las distintas placas que conformarán la *PHR.*Gran parte del informe se basa en el *software* que permitirá la comunicación entre la PC y el programador Joint Test Action Group (JTAG).

2. Programador JTAG

Gran parte de los *scripts* utilizados se obtuvieron del manual de usuario de Open On-Chip Debugger (OpenOCD) que se encuentra publicado en la web. Se recuerda que anteriormente al uso de OpenOCD, se intentó utilizar el *software* UrJTAG, pero debido a la limitaciones que presentaba la versión estable, se optó por cambiarlo. Además existía mayor documentación de OpenOCD que de UrJTAG.

2.1. *Software* OpenOCD

En el proceso de lograr obtener el *software* funcional, se realizó un gran trabajo que se podría diferenciar en tres etapas:

- Instalación del *software* OpenOCD
- *Interface*
- *Target*

2.1.1. Instalación del *software* OpenOCD

Para la instalación se recurrió a la documentación del *software*. En dicho documento se hace referencia a los requerimientos para cada sistema operativo (SO). Como ya se mencionó en anteriormente reportes, se utilizará herramientas libres, por lo que se realizará la instalación en un SO GNU/Linux Debian “Squeeze” 6.0.

2.1.2. *Interface*

El *Interface* hace referencia al *hardware* que realiza el enlace entre el dispositivo que queremos programar y la PC. En éste caso el *interface* será nuestra placa OOCD-Links. OpenOCD dispone de *scripts* para varios *interface* entre los cuales se encuentra nuestra placa programadora. La información que proporciona éste código hace a que tipo de driver se utilizará como así también identificación del dispositivo central. A continuación se puede ver la estructura del archivo `oocdlink.cfg`.

```
#
# Joern Kaipf's OOCDLink
#
# http://www.joernonline.de/contrexx2/cms/index.php?page=126
#

interface ft2232
ft2232_device_desc "OOCDLink"
ft2232_layout oocdlink
ft2232_vid_pid 0x0403 0xbaf8
jtag_khz 5
```

2.1.3. *Target*

Target define las especificaciones del *hardware* al cual queremos acceder con el *interface* OOCD-Links. Al igual que con los *interfaces*, OpenOCD dispone de *scripts* con varias *targets* conocidas. En nuestro caso vamos a utilizar éstos archivos como ejemplo para configurar y adaptar a nuestra placa de prueba. Por último se probará agregar varios dispositivos al archivo que se generé y poder programarlos a la vez con la placa OOCD-Links.

2.2. Documentación de comandos y *scripts*

Luego de la configuración del *software* OpenOCD, se procede a generar código de programa que nos permita realizar las tareas que necesitamos para nuestros proyecto *PHR*. En principio se agregarán comandos útiles que servirán en general para cualquier dispositivo. Entre ellos se puede destacar,

- Identificar información de un dispositivo sin datos del proveedor, para agregarlo a una cadena JTAG como Test Access Port (TAP).
- Configurar dispositivo en modo *halt*.

2.2.1. Agregar TAP

Para acceder a un dispositivo mediante el protocolo JTAG, se debe proporcionar información del *hardware* al *interface*. Muchas veces ésta información no se encuentra fácil de acceder en la hoja de datos de dichos dispositivos por lo que OpenOCD dispone de un modo *Autoprobing*, para más información sobre éste modo se puede obtener del Manual de Usuario de OpenOCD en la sección 10.7 Autoprobing. En definitiva, para capturar la información del dispositivo conectado se debe generar una archivo `openocd.cfg` base como el que se muestra a continuación,

```
# OpenOCD configuration script
# 2013-03-25 lguanuco
# Hardware:
# - OOCD-Link-s

# ("3": max. verbosity level)
debug_level 3
# (set log file)
log_output out.log

source [find core.cfg]

# 3. Other configs

reset_config trst_and_srst
jtag_rclk 8
```

Luego de correr OpenOCD, se debe acceder al archivo de salida con el nombre que se asignó en el archivo `openocd.cfg`, `out.log`. Este archivo registra todos los mensajes de salida que genera OpenOCD, a continuación se muestra parte de éste archivo donde se puede observar información útil para generar el TAP de dicho dispositivo.

```
...
Debug: 129 273 core.c:323 jtag_call_event_callbacks(): jtag event: TAP reset
Warn : 130 359 core.c:1145 jtag_examine_chain(): AUTO auto0.tap - use "jtag newtap auto0
Debug: 131 359 core.c:1323 jtag_tap_init(): Created Tap: auto0.tap @ abs position 0, irle
Debug: 132 359 core.c:1208 jtag_validate_ircapture(): IR capture validation scan
Warn : 133 369 core.c:1244 jtag_validate_ircapture(): AUTO auto0.tap - use "... -irlen 4"
Debug: 134 369 core.c:1267 jtag_validate_ircapture(): auto0.tap: IR capture 0x01
Debug: 135 369 openocd.c:145 handle_init_command(): Examining targets...
...
```

Una vez obtenida éstas lineas se puede rescatar los datos más importantes que son,

- *id*
- *irlen*
- *capture*
- *mask*

2.2.2. Modo *Halt*

El modo *halt*, permite congelar al dispositivo tanto para agregar *breakpoints* como también programar el dispositivo. Un problema común es que dicho modo no puede ser aplicado, para solucionar ésto es necesario realizar un *reset*, con el comando `reset halt`. Luego de esto se puede pasar al modo *halt* sin problemas.

2.2.3. Programación Complex Programmable Logical Device (CPLD)

Para los CPLD, primero se debe tener el archivo Serial Vector Format (SVF), que es generado por la herramienta con la que se diseña y sintetiza el código VHDL o Verilog. En éste caso se utiliza el *software* ISE Xilinx. Una vez que se tiene el archivo SVF, desde OpenOCD, se corre el siguiente comando;

```
svf {file.svf} quiet
```

Lo que generaría una salida como la que sigue,

```
svf processing file: "file.svf"
500 kHz

Time used: 0m7s478ms
svf file programmed successfully for 8468 commands
```

En el caso de que se tenga problemas en la transferencia del archivo, se podría probar bajando la frecuencia de la comunicación JTAG. Recordar que esto quizá se deba cambiar también desde el entorno ISE Xilinx.

En la conexión de los puertos JTAG, hemos encontrado un problema y es el nivel de tensión que maneja éstos pines con respecto a los de la placa OOCD-Links. La placa OOCD-Links cuenta con un dispositivo central, FT2232D. Éste dispositivo posee dos alimentaciones, por una parte la alimentación de núcleo del circuito integrado y por otro lado la alimentación de los *interfaces* de salida. La primera alimentación es tomada desde el puerto USB al que se encuentra conectado, mientas que la alimentación de sus puertos de conexión lo toma del conector *P1*. Éste último debería tener el mismo nivel de tensión que la placa con la que se conecte. Es decir, si se hace una conexión entre la placa OOCD-Links y una placa con un CPLD XC9572XL, se debería asegurar que ambas tengan la misma tensión en sus puertos JTAG, en éste caso 3.3 V.

3. Documentación

La documentación resulta fundamental en ésta etapa del desarrollo. Si bien se quiere lograr el correcto funcionamiento de las placas, la documentación sirve para realizar correciones a las versiones futuras de cada placa. Otro objetivo es documentar el funcionamiento de cada dispositivo que sirvan al reporte final como así también a los usuarios de la *Plataforma de Hardware Reconfigurable*.

A. Acrónimos

PHR Plataforma de Hardware Reconfigurable

OpenOCD Open On-Chip Debugger

JTAG Joint Test Action Group

TAP Test Access Port

SVF Serial Vector Format

CPLD Complex Programmable Logical Device

SO sistema operativo

UTN-FRC Universidad Tecnológica Nacional – Facultad Regional Córdoba

B. Repositorio de proyecto

El proyecto se encuentra alojado en los servidores de *OpenCores*. Por lo que se puede acceder a los repositorios mediante el siguiente link, `http://opencores.org/project,phr` De todas formas se pueden comunicar por correo, `guanucoluis@gmail.com`.