Universidad de Granada

Sistemas Críticos

Diseño de una plataforma de criticidad mixta

Marvin Matías Agüero Torales

maguero@correo.ugr.es

Curso 2016-2017

Sumario

Enunciado de la práctica

Esta práctica se pretende el desarrollo de un SoC para aplicaciones de criticidad mixta. El software a utilizar es el Vivado de Xilinx[1]. Se debe programar la tarjeta y desarrollar una pequeña aplicación con el SDK para validar su funcionamiento.
Se utilizarán como documentación de referencia los laboratorios 1 y 2 de Vivado (Xilinx Inc., 2015).

Configuración

Antes que nada debemos instalar Vivado, para ello escogemos la versión System Edition (WebaPack), marcando incluir también el SDK. Como advertencia, este software es muy pesado y requiere un volumen de disco duro considerable (unos 23 GB). Es importante eliminar opciones no necesarias como familias de dispositivos distintos de la Zynq-7000 o Artix, así como paquetes como HLS o SystemGenerator para ahorrar espacio de disco duro.

La instalación se puede hacer de dos maneras, la primera es online y la segunda offline, se recomienda la segunda, puesto que con la primera suele dar problemas, ya que puede haber un corte de red o similar, aunque no se interrumpa la descarga e instalación, puede corromper algunos elementos importantes del software y dar problemas más adelante.

Cabe mencionar, que la instalación debemos hacerla con nuestro usuario o con root, y ejecutarlo de la misma manera, para que no debe problemas de permisos o inconsistencias.

Una vez instalado el Vivado, debemos instalar los cable drivers, primero los localizamos

```
sudo find /opt/Xilinx/ -name install_drivers*
```

Luego podemos instalarlos

```
./opt/Xilinx/Vivado/2016.4/data/xicom/cable_drivers/lin64/install_script/install_drivers/install_drivers
```

y/o

```
./opt/Xilinx/SDK/2016.4/data/xicom/cable_drivers/lin64/install_script/install_drivers/install_drivers
```

Como servidor de licencias se usará atccongresos.ugr.es y será suficiente con usar la variable 2100@atccongresos.ugr.es para el acceso al servidor en el "Xilinx License Manager" del menu Help de Vivado. En Ubuntu podemos usar export y agregarlos al ~/.bashrc de nuestro usuario, para que siempre utilice la licencia, o crear un script para arrancar Vivado con la licencia así:

```
#!/bin/bash

export XILINXD_LICENSE_FILE=2100@atccongresos.ugr.es

export SWT_GTK3=0

export WORKSPACE=”/media/marvin/8868F00968EFF43A/Academico/MII/SC/Vivado”

cd ${WORKSPACE}

source /opt/Xilinx/Vivado/2016.4/settings64.sh
```

1 http://www.xilinx.com/support/download.html

source /opt/Xilinx/SDK/2016.4/settings64.sh

/opt/Xilinx/Vivado/2016.4/bin/vivado

Desarrollo

El desarrollo se hace sobre una portátil HP Envy con procesador i7 3o generación con 8 cores y 8

GB de RAM, con SO Ubuntu 16.04 LTS de 64 bits.

Se siguió el material de la asignatura para llevar acabo este trabajo.

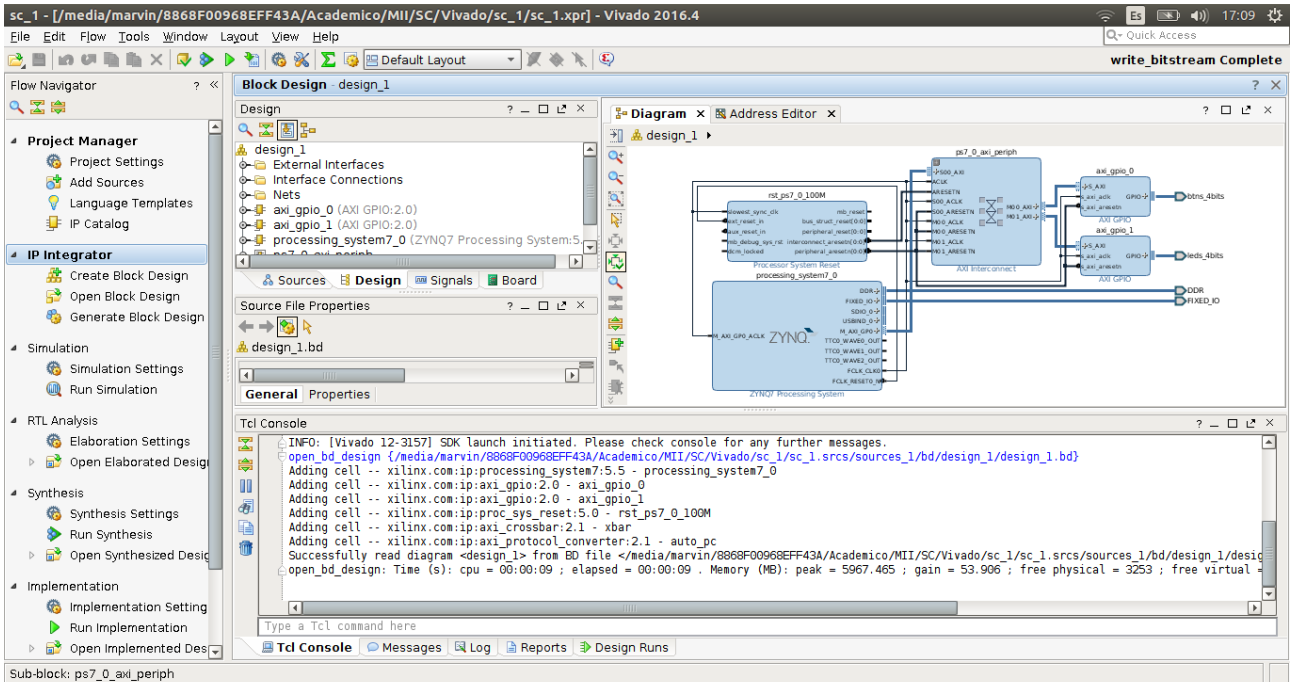
Prerrequisitos

Para crear un procesador el ARM Cortex-A9 necesitamos una tarjeta Zybo[2], la cual debemos importar su definición al lugar correspondiente, que en mi caso es /opt/Xilinx/Vivado/2016.4/data/boards/board_files.

Construir un Embedded System

Crear un proyecto Vivado

Creamos un proyecto en Vivado del tipo RTL, Target Language: VHDL, Simulator Language: Mixed, Board la Zybo de Digilent Inc. Dentro del proyecto, creamos un diseño mínimo: Create block desgin, añadimos el SoC Zynq de Xilinx: Add IP, automatizamos las conexiones: Run Block Automation, añadimos el GPIO: Add IP, automatizamos las conexiones: Run Connection Automation (en GPIO, led de 4 bits, boton de 4 bits), regeneramos el layout del diseño, validamos el diseño: pulsando F6.



Una vez validado el diseño, pasamos a la generación de los ficheros HDL: IP Integrator, Generate Block Design; creamos el Bitstream: Program and Debug, Generate Bitstream.

2 https://www.xilinx.com/support/documentation/university/vivado/workshops/vivado-embedded-design-flow-zynq/materials/2015x/ZYBO/zybo.zip

Luego exportamos el diseño de la plataforma de Vivado al SDK. Para ello, en el proyecto Vivado creado, en File, Export, Export Hardware y lo guardamos dentro del mismo proyecto, luego File, Launch SDK.

Ahora debemos crear un Board Support Package, en File, New, con OS Platform, Standalone e in/out con interacción normal (teclado/ratón).

Crear una aplicación en el SDK

En el SDK, vamos a New, File, Application Project, HelloWorld en C.

Modificamos el helloWorld para tomar los botones y leds GPIO, como punteros de entrada/salida (en xparameters.h encontramos las constantes de CPU, dirección de GPIOs), asignándole funciones del GPIO: como leer leer y escribir: pulsar botón, encender led …

El programa quedo así:

```
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include "xparameters.h"
#include "xgpio.h"


int main()
{
    init_platform();

    print("Hello World\n\r");

        XGpio btn, led;
        int btn_check;

        xil_printf("***START***\r\n");

        XGpio_Initialize(&btn, XPAR_GPIO_0_DEVICE_ID);
        XGpio_SetDataDirection(&btn, 1, 0xffffffff);

        XGpio_Initialize(&led, XPAR_GPIO_1_DEVICE_ID);
       XGpio_SetDataDirection(&led, 1, 0xffffffff);

    while(1){

        btn_check = XGpio_DiscreteRead(&btn, 1);
        xil_printf("Gpio Btn Read \r\n", btn_check);

        XGpio_DiscreteWrite(&led, 1, btn_check);
        xil_printf("Gpio Led Write \r\n");

        sleep(1);
    }

    cleanup_platform();
    return 0;

}
```

Prueba en hardware

Conectamos la tarjeta con un cable micro-usb y lo encedemos. Podriamos utilizar el terminal de Vivado, desde Ventana> Mostrar vista> Terminal, pero por alguna razón no funciona, así que utilizamos Minicomp, debemos seleccionar el puerto USB objetivo y el rate en el más alto.

sudo apt-get install minicom

minicom -D /dev/ttyUSB0

Ya dentro de minicom, en Options, serial port setup, configuramos de acuerdo a nuestra plataforma[3].

Inconvenientes

La primera instalación del Vivado la hice “online”, teniendo luego problemas para lanzar la aplicación y con los cable drivers, por lo que tuve que escribir un script de arranque de Vivado[4] descrito en el apartado de configuración. Este script también ayudó a solucionar el problema al lanzar el SDK[5], agregando

```
export SWT_GTK3=0
```

La terminal de Vivado por alguna razón a veces tiene fallas y otras no, por lo que tuvimos que configurar la terminal con Minicom (apartado Prueba en Hardware).

Conclusiones

Vivado permite el diseño de una plataforma de criticidad mixta muy rápidamente de una manera guiada y muy gráfica. Una vez definido el sistema, se puede exportar el hardware y se puede invocar el SDK de Vivado. Incluso este SDK proporciona varias plantillas de aplicación base.

EN este trabajo pude comprobar la operación del hardware, creando una aplicación de prueba, ejecutándose en el procesador, y observando la salida en la ventana de terminal serie y en la placa misma, prendiendo y apagando las luces led.

Bibliografía

Xilinx Inc. (2015). Embedded System Design Flow on Zynq using Vivado. Recuperado 7 de marzo de 2017, a partir de http://www.xilinx.com/support/university/vivado/vivado-workshops/Vivado-embedded-design-flow-zynq.html

Anexos

SDK del proyecto comprimido.

3 http://www.wiki.xilinx.com/Setup+a+Serial+Console
4 https://forums.xilinx.com/t5/Installation-and-Licensing/Installed-Vivado-2014-1-webpack-on-Linux-Ubuntu-and-there-is-no/td-p/463838
5 https://www.xilinx.com/support/answers/68490.html