LEIS- UNSL - 2017

Curso de:
Programación avanzada de Sistemas Embebidos en RTOS y Linux embebido

Pasos para ejecutar una aplicación en Petalinux usando JTAG

I- GENERAR SISTEMA OPERATIVO

1- Crear un sistema base con Vivado
    a) Crear un proyecto en Vivado
    b) Agregar el IP Zynq Processing System
    c) Run Automation Connection
    d) Conectar la entrada M_AXI_GPO_ACKL al reloj FCLK_CLK0
    e) Crear el Wrapper HDL
    f) Generar el bitstream y exportar el hardware generado, incluyendo el bitstream

Antes de continuar!
Configurar el ambiente de Petalinux desde la terminal:
$ source <path-to-installed-PetaLinux>/settings.sh

2- Crear un proyecto nuevo en petalinux desde la terminal
    petalinux-create --type project --template zynq --name <nombre>

3- Importar Hw creado en punto 1
    a) Ir al directorio del proyecto petalinux creado en punto 2
    b) Ejecutar
    petalinux-config --get-hw-description=<path al directorio .sdk creado en el punto 1 donde está el archivo .hdf>

4- Crear la Imagen del sistema

a) Ir al directorio del proyecto petalinux creado en el punto 2

b) Ejecutar

petalinux-build

Si da un error del tipo

INFO] install linux/kernel

[ERROR] ERROR: Invalid ELF file '/home/tochibow9/Xilinx-ZC706-2015.4/images/linux/

ERROR: Failed to build linux

Ejecutar

LANG= LANG_C= petalinux-build

5- Generar Imagen de Boot

a) Ir al directorio del proyecto petalinux creado en punto 2

b) Ir al directorio images/linux

c) Cambiar el nombre del archivo .bit que está en el directorio imges/linux por download.bit

d) Volver al directorio raíz del proyecto y ejecutar

petalinux-package --boot --fsbl <archivo fsbl.elf (genralmente está en image/linus/zynq-fsbl.elf> --fpga <archivo .bit (que está en el mismo directorio images/linux) –u-boot

NOTA: tanto para --fsbl como para --fpga poner el path y el archivo correspondiente

6- Empaquetar la imagen prebuilt

a) Ir al directorio del proyecto petalinux

b) Ejecutar

petalinux-package --prebuilt --fpga <archivo .bit>

7- Boot con petalinux-boot usando la imagen creada

a) Conectar la placa de desarrollo Zedboard dos cables usb para JTAG y el puerto serie

b) Encender la placa

c) Abrir terminal serie (mincom o tera term)

d) Ejecutar

petalinux-boot --jtag --prebuilt 3

Esto cargará el bitstream a la FPGA y la imagen de Linux,
En el terminal serie se debe observar el proceso de boot.
Una vez finalizado logearse con:

usuario= root,
passw= root.

Dar un paseo por el SO cargado

II CREAR UNA APLICACIÓN

Volvemos al ordenador

8- Ir al directorio del proyecto petalinux

a) Crear una aplicación ejecutando

petalinux-create -t apps --name <nombreapp> --enable

Esto crea una aplicación .c por defecto. Si se desea crear una aplicación c++ el comando es

petalinux-create -t apps –template c++ --name <nombreapp> --enable

9- Ver los archivos creados por petalinux

a) Ir al directorio de la aplicación creada

cd components/apps/nombreapp

Deben estar los siguientes archivos: Kconfig, Makefile, README y nombreapp.c

Abrir el archivo nombreapp.c (es un hola mundo)

Modificar para la funcionalidad deseada. (por ahora lo ejecutaremos como está)

10- Hacer un build de la aplicación creada

a) Ir al directorio del proyecto petalinux

b) Ejecutar

petalinux-build

(recordar lo de la configuración LANG por si hiciera falta)

c) instalar la aplicación creada ejecutando

petalinux-build -c rootfs/nombreapp -x install

11- Volver a compilar y generar imagen de petalinux

repetir los pasos 5-6 y 7

12 – probar ejecutando desde el prompt # nombreapp

NOTA: la aplicación se guarda en le directorio /bin

III AGREGAR UNA APLICACION CREADA FUERA DE PETALINUX, AL ROOT FILE SYSTEM (OPCIONAL)

(SOLO SE PUEDE HACER CON APLICACIONES COMPILADAS PARA ARM)

Suponiendo que la aplicación creada fuera de petalinux se llama myapp

13- Ir all directorio del proyecto de petalinux

a) Crear una aplicación con el mismo nombre de la aplicación a agregar ejecutando

petalinux-create -t apps --template install --name myapp –enable

b) Ir al directorio de components/apps/myapp

c) Editar el archivo MAKEFILE y descomentar la linea

$ (TARGETINST) -d/data/myapp /bin/myapp

d) Ir al directorio components/bin/myapp/data

e) Borrar myapp existente

d) Copiar la aplicación myapp a agregar en este directorio

14- Repetir los pasos 10 a 12 anteriores

También se pueden agregar librerías propias o ya compiladas como Qt.
Todo esto está en las guías UG1144 y UG981 que se adjuntan

Se puede crear una aplicación en Ubuntu y compilarla para ARM
C_COMPILER arm-linux-gnueabihf-gcc
CXX_COMPILER arm-linux-gnueabihf-g++

y luego ir al paso 13