ECE510 Post-Silicon Validation

Project 1

Learning Objectives:

- Understand TAP
- Practice coding in Python
- Gain experience interfacing & debugging a DUT

Project 1 demonstrations will be on May 10th
Project 1 deliverables are due by May 10th @ 11:00 PM

Project: Raspberry Pi TAP Controller

In this project you will build a Python JTAG TAP controller running on Raspberry Pi and utilize the GPIO pins of Raspberry Pi as the TAP controller pins to interface with a Digilent Nexys3 board. You will be given a skeleton design to start with. It provides a design framework to help you complete the design.

JTAG TAP

JTAG (Joint Test Action Group) is an IEEE 1149.1 standard defining TAP (Test Access Port) and boundary scan architecture. It is used in silicon devices for debugging, testing and programming purposes. The typical JTAG architecture is shown in Figure 1. The TAP pins are outlined in Table 1.

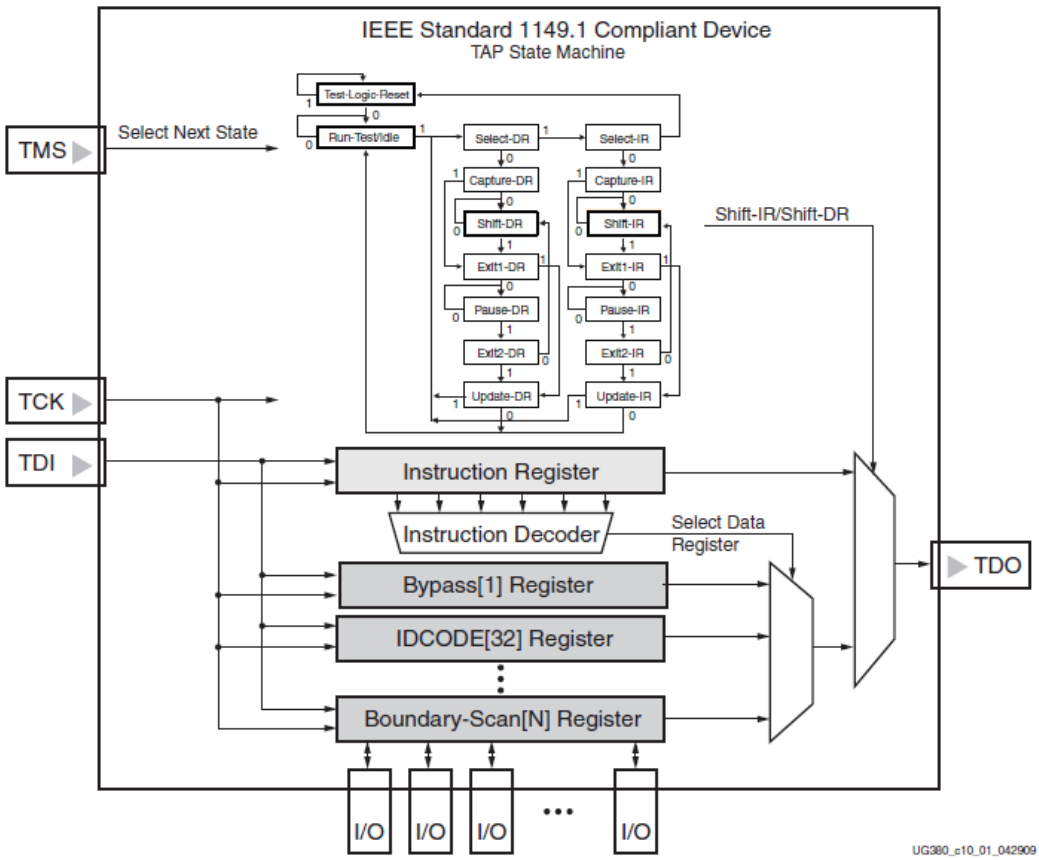

Figure 1: Typical JTAG Architecture

Table 1: TAP controller pins

Pin	Direction	Description
TDI	In	Test Data In, serial input to instruction & data registers
TDO	Out	Test Data Out, serial output for instruction & data registers
TMS	In	Test Mode Select, determine the states on the rising edge of TCK
TCK	In	Test Clock
TRST (Optional)	In	Test Reset

TAP state machine is shown in Figure 2.

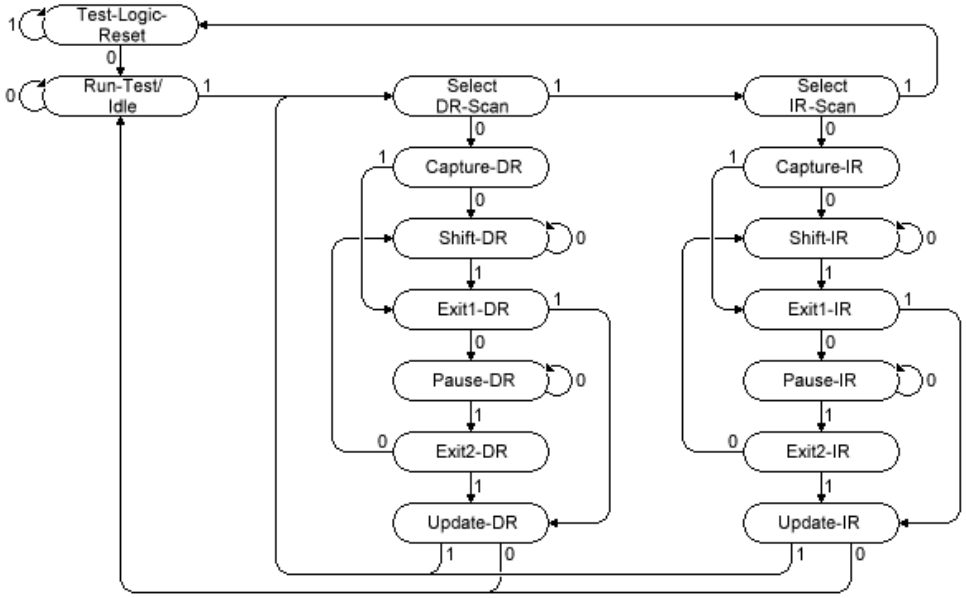

Figure 2: TAP State Machine

You will need to fill out the missing code from the reference design to complete this project. After complete coding the TAP controller, interface the Raspberry Pi with a Digilent Nexys3 board to read out the device code. Xilinx FPGAs support JTAG protocol for their configuration. Read the chapter 10 of the Spartan-6 FPGA configuration user guide to figure out how to read the content of the IDCODE.

Hardware

This project will be implemented in Raspberry Pi. It provides a number of GPIO pins that are easily accessible and programmed with Python. In addition to Raspberry PI, you will need to a Micro SD card with preferably 8GB of space, Ethernet cable or WiFi USB dongle, an HDMT cable display; you can purchase these components separately or just order a starter kit online.
https://www.raspberrypi.org/ has good information for you to get started.

Software

Operating system: you can use a flavor of Linux called Raspbian; get started using the following link:

https://www.raspberrypi.org/help/noobs-setup/
Adafruit has a quite a few tutorials:
First time configuration: https://learn.adafruit.com/adafruits-raspberry-pi-lesson-2-first-time-configuration
Network Setup: https://learn.adafruit.com/adafruits-raspberry-pi-lesson-3-network-setup
SSH: https://learn.adafruit.com/adafruits-raspberry-pi-lesson-6-using-ssh

Python Module Installation:
Two installation methods are commonly used: one through **apt-get**, the other through **pip**
If u can't find something in **apt-get**, **try pip**

```
~$ sudo apt-get install python3
~$ sudo apt-get install python3-pip
~$ sudo pip-3.2 install colorama
```

Python GPIO module: there are a whole slew of Python modules that makes it easy to read & control the GPIO pins. I recommend raspberry-gpio-python. This module should come pre-installed with the latest Raspbian distribution. To check if you have it, go into Python interactive interpreter as root from terminal and run the following code:

```
~$ sudo python
>>> import RPi.GPIO as GPIO
```

You are set if you don't get any error. Otherwise exit the interpreter and install the module like following:

```
~$ sudo apt-get update &  sudo apt-get upgrade
~$ sudo apt-get install python3-rpi.gpio
```

Verify installation by using interactive interpreter and import the module.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(25, GPIO.OUT)

while Ture:
   GPIO.output(25, GPIO.HIGH)
   time.sleep(1)
   GPIO.output(25, GPIO.LOW)
   time.sleep(1)
```

Project Tasks

Download the reference design

Download *project1.tar* from the course website. This .tar file contains the reference design that you can use to get started and examples for using TAP controller. The tar file contains the following files:

Project 1 Reference Design	
Name	**Description**
`project1\tap\code_docs\build\html\index.html`	Code documentation html page
`project1\tap\common\tap_gpio.py`	Tap_Gpio class with pin related setup & function calls
`project1\tap\common\tap.py`	Tap controller class
`project1\tap\common\loopback.py`	LoopBack class for GPIO pin monitoring
`project1\tap\log\logging.json`	Logging config file
`project1\tap\log\logging_setup.py`	Logging related functional calls
`project1\tap\log\*log`	Log files
`project1\tap\model\tap_model.py`	Helper class for state tracking
`project1\tap\regression\regression.py`	Tap controller regression script
`project1\tap\regression\tests\*py`	Regression tests
`project1\tap\examples\tap_prog.py`	An example code that instantiates `tap.py`
Documentation	
`project1\docs\project1.pdf`	This document
`project1\docs\xilinx_ug380.pdf`	Xilinx Spartan-6 FPGA configuration user guide
`project1\docs\adafruits-raspberry-pi-lesson*`	Adafruit Raspberry Pi setup lessons

Design TAP Controller

- Complete blank routines in `tap.py`.
- Complete testReset, testReset2ShiftIR, testExit1IR2ShiftDR, testReadDeviceCode routines in smoke.py; utilize the routines from LoopBack class to monitor TAP state transitions for validation.
- (Optional) hook up the TAP pins to a scope and verify the waveforms.

Connect to Nexys3 Board

Connect the Raspberry Pi to a Digilent Nexys3 board to read out the device IDCODE in the terminal.

Device	ID Code (Hex)
6SLX4	0xX4000093
6SLX9	0xX4001093
6SLX16	0xX4002093
6SLX25	0xX4004093
6SLX25T	0xX4024093
6SLX45	0xX4008093
6SLX45T	0xX4028093
6SLX75	0xX400E093
6SLX75T	0xX402E093
6SLX100	0xX4011093
6SLX100T	0xX4031093
6SLX150	0xX401D093