Week 5

Objectives

1. Be able to communicate with the JTAG pins of the board
2. Test the concrete implementation of the FPGA's JTAG interface

Communicate with the board

Create the config file

Info about the JTAG interface :

```
$ openfpgaloader --detect
Jtag frequency : requested 6.00MHz    -> real 6.00MHz
index 0:
    idcode 0x41113043
    manufacturer lattice
    family ECP5
    model  LFE5U-85
    irlength 8
```

```
$ openfpga --scan-usb
empty
Bus device vid:pid       probe type      manufacturer serial               product
020 005    0x0403:0x6010 FTDI2232        FTDI         none                 Dual RS232-HS
```

Create config file for openOCD : openocd docs

The config i have done seems to work correctly

Use openOCD to interact

From the OpenOCD user guides the states are :

- RESET... stable (with TMS high); acts as if TRST were pulsed
- RUN/IDLE... stable; don't assume this always means IDLE
- DRSELECT
- DRCAPTURE
- DRSHIFT... stable; TDI/TDO shifting through the data register
- DREXIT1
- DRPAUSE... stable; data register ready for update or more shifting
- DREXIT2
- DRUPDATE
- IRSELECT
- IRCAPTURE
- IRSHIFT... stable; TDI/TDO shifting through the instruction register
- IREXIT1
- IRPAUSE... stable; instruction register ready for update or more shifting
- IREXIT2
- IRUPDATE

Note that only six of those states are fully "stable" in the face of TMS fixed (low except for reset) and a free-running JTAG clock. For all the others, the next TCK transition changes to a new state. • From drshift and irshift, clock transitions will produce side effects by changing register contents. The values to be latched in upcoming drupdate or irupdate states may not be as expected. • run/idle, drpause, and irpause are reasonable choices after drscan or irscan com- mands, since they are free of JTAG side effects. • run/idle may have side effects that appear at non-JTAG levels, such as advancing the ARM9E-S instruction pipeline. Consult the documentation for the TAP(s) you are working with.

Open OCD provides 4 main commands to interact with the board using low level JTAG:

1. `irscan <tap_name> <instruction>`: Load the **instruction** in the instruction register of **tap_name**
2. `drscan <tape_name> <number_of_bits> <data_to_transfer>`: Send the **number_of_bits** first bits of **data_to_transfer** to the instruction chain selected
3. `pathmove <state1> (<state2> <state3> ...)`: move to the state1 (then state2, state3 ...) and stay in this state if it is persistant
4. `runtest <num_cycle>`: Stay in RUN/IDLE for **num_cycle** clock cycles

Recall the IPCORE

The ipcore implemented works using 2 instructions : 0x32 to decide the color of the first 3 leds of the selected column, 0x38 to select the column.

Both instructions use a shift register to shift the data before sending it to the corresponding outputs:

- 0x32: [0:2]Led 0, [3:5]Led 1, [6:8]Led 2
- 0x38: [0:4]The column to be selected

Concrete execution

Here is a concrete example on how to use openOCD to communicate with the ipcore to light up specific leds:

- syntethize and load the design on the Board
- run `openocd myconf.cfg` to run the server used to communicate with the JTAG interface
- on a new terminal access the server : `telnet localhost 4444`
- on the server :
  - `irscan ecp5.tap 0x32` select the 0x32 instruction
  - `drscan ecp5.tap 10 0xdb` light all thirst 3 leds with the same value (10 is used even if the shift register is 9 bits long since the JTDI is one clock cycle late)
  - `irscan ecp5.tap 0x32` select the 0x38 instruction
  - `drscan ecp5.tap 5 0xb` select the last led column

Questions

No question this week since starting a new step in the project, maybe how does the RAM works in the architecture