Lab #7: Run and Debug IP

04/19/2018

4190.309A: Hardware System Design
(Spring 2018)

Overview

- Custom IP Design
  - Processing System + BRAM + Connectivity + Custom IP (shifter)
- Integrated Logic Analyzer (ILA)
  - Extract values of nets and regs on runtime
- Practice
  - Write a short report about HW and SW of Lab 7

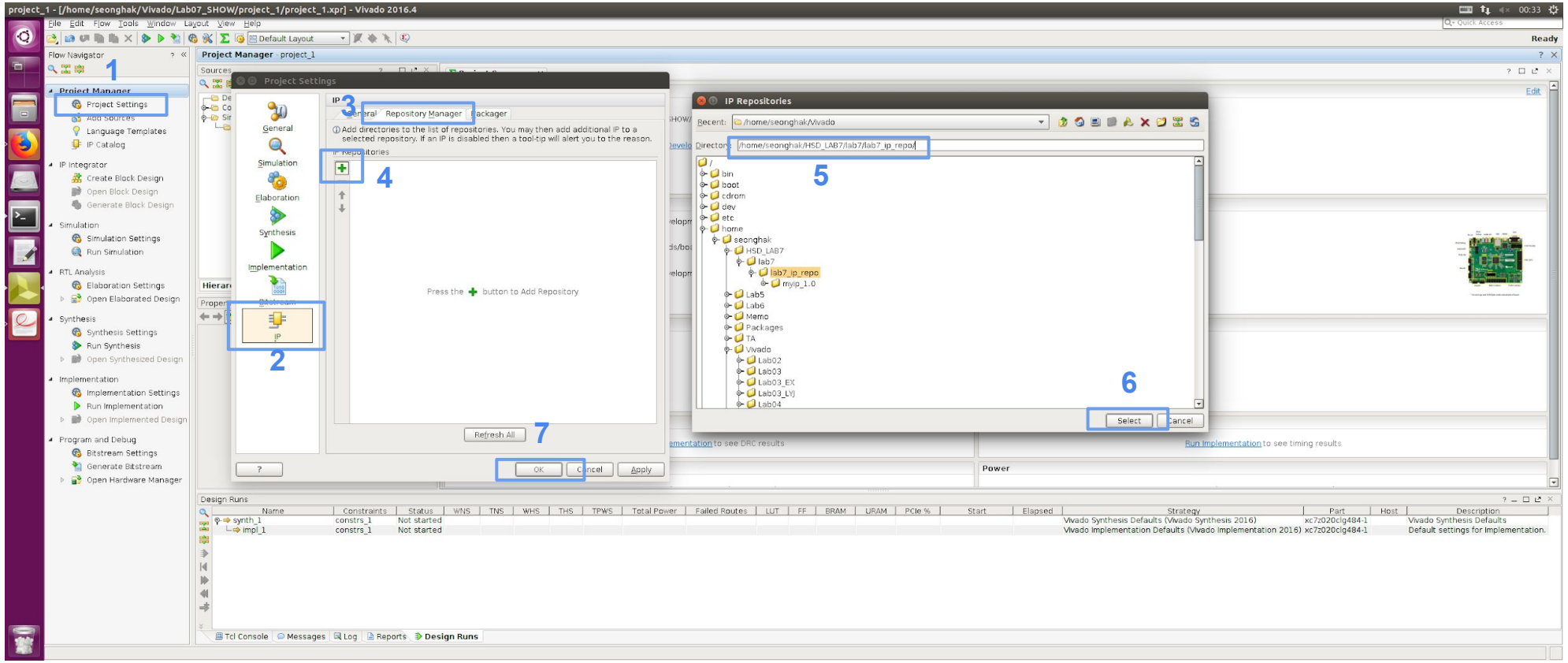
Tutorial: Custom IP Design

Notations

- `HOST$ XXX`
  - Type XXX at the terminal of your Ubuntu-PC.
- `BOARD$ YYY`
  - Type YYY at the terminal of ZedBoard (on minicom).
- `TCL$ ZZZ`
  - Type ZZZ at the Tcl console of Vivado.

Custom IP Design - Apply Block Design (1)

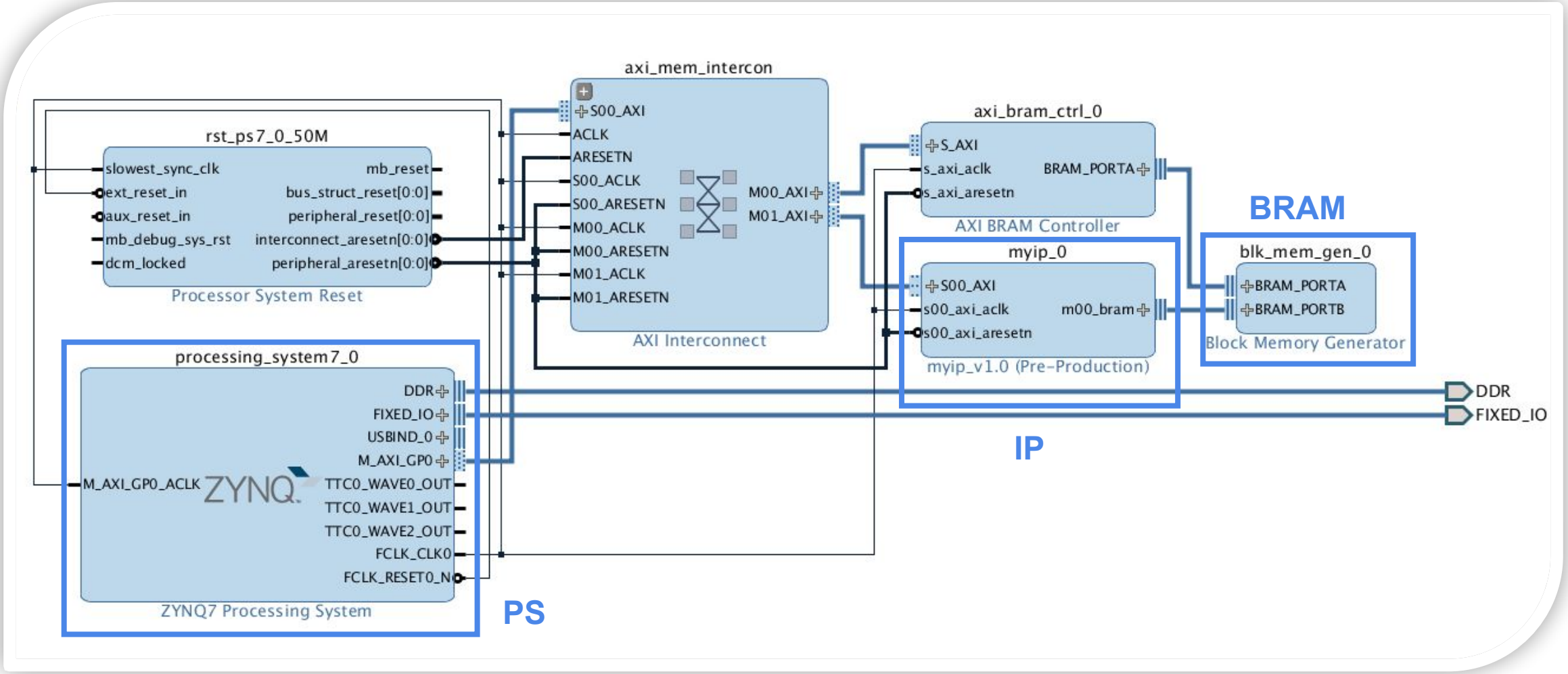
- Download an example IP
  - `HOST$ git clone https://github.com/K16DIABLO/HSD_LAB7.git`
  - `HOST$ tar -zxvf lab7_ip_repo.tar.gz`



Custom IP Design - Apply Block Design (2)

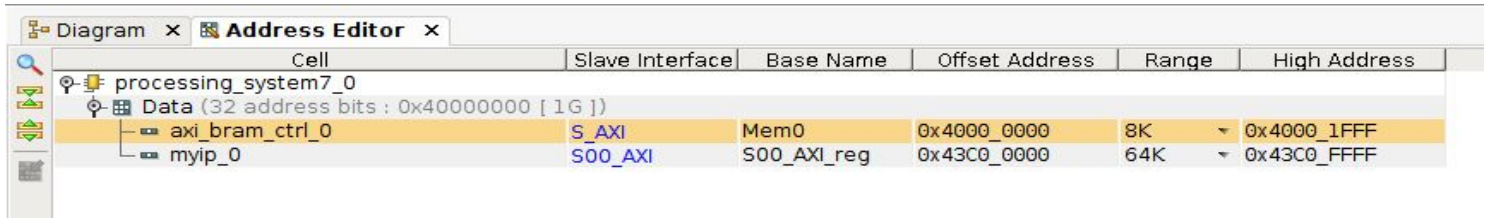
- Execute TCL scripts of the example block design.
  1. Select Tcl Console tab
  2. `TCL$ cd $(git_clone_dir)`
  3. `TCL$ source block_design.tcl`

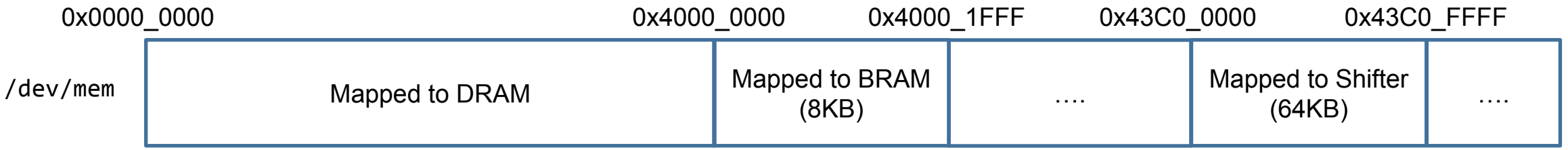
Custom IP Design - Summary



Memory map of ZYNQ SoC (1)

- BRAM is mapped to address 0x4000_0000 ~ 0x4000_1FFF
- Shifter is mapped to address 0x43C0_0000 ~ 0x43C0_FFFF



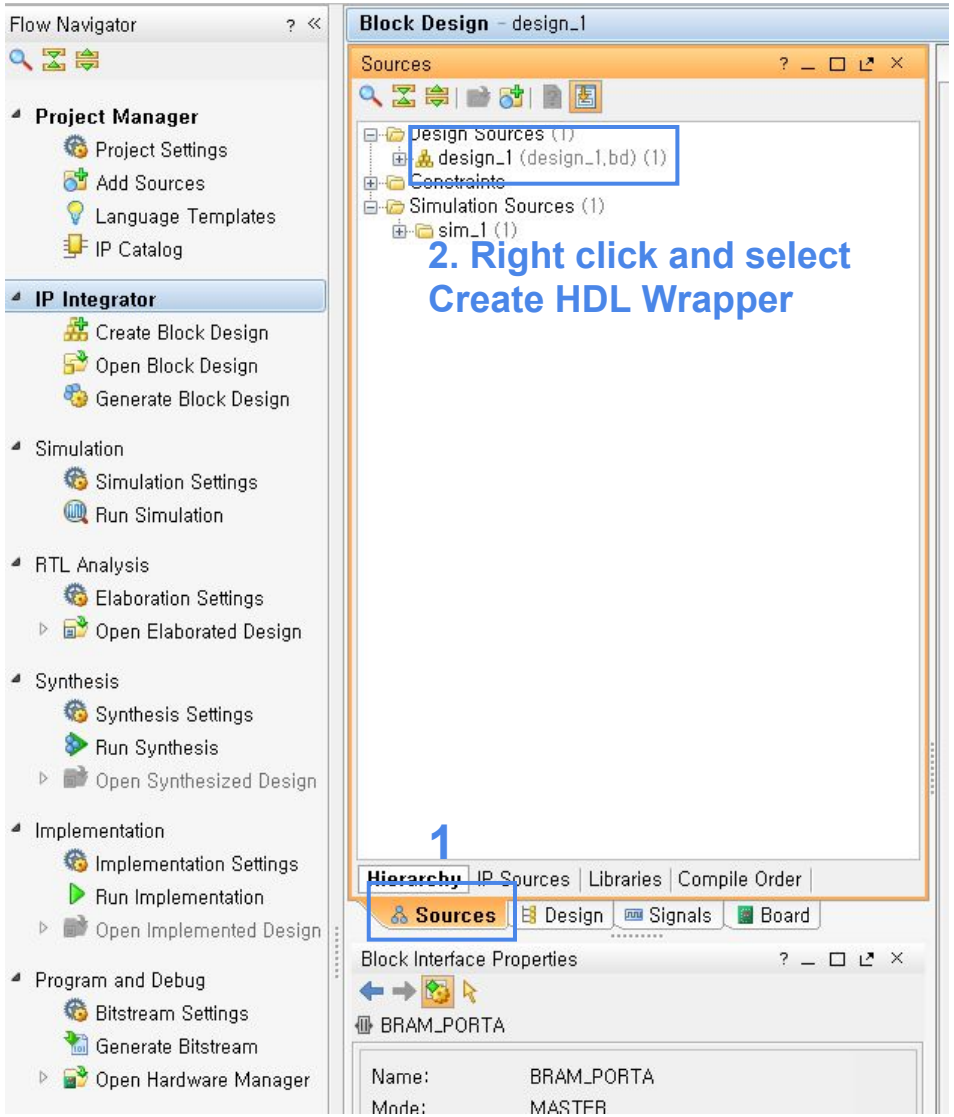


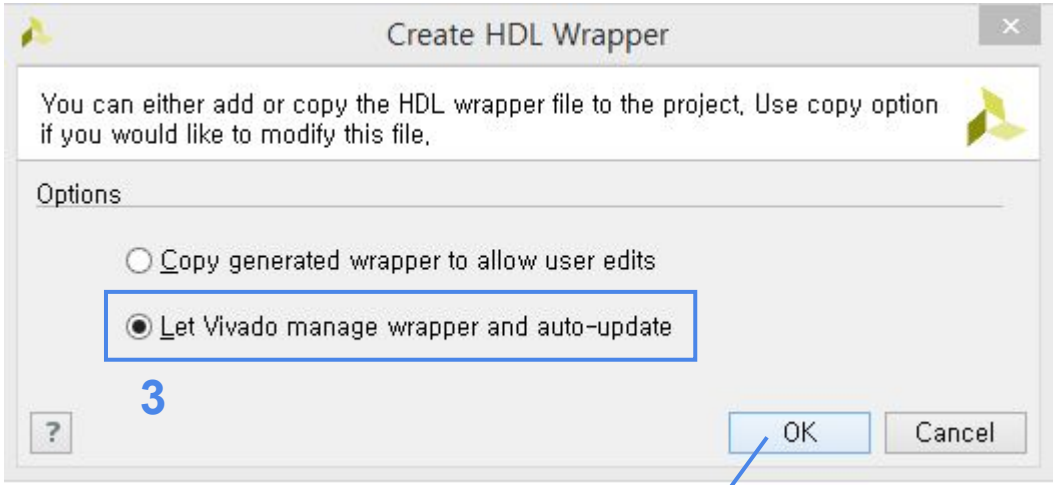
Memory map of ZYNQ SoC (2)

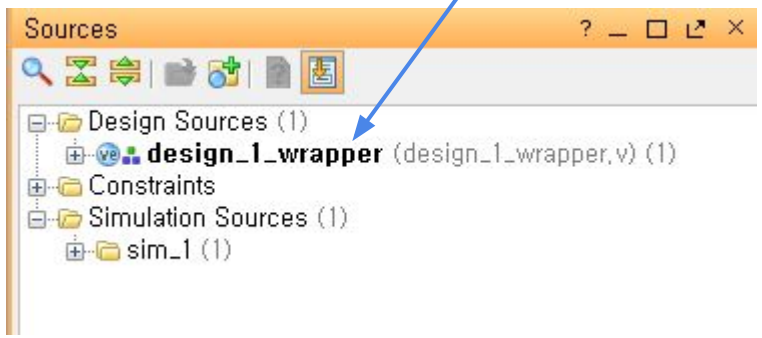
- Shifter is activated when *(0x43C0_0000) == 0x5555

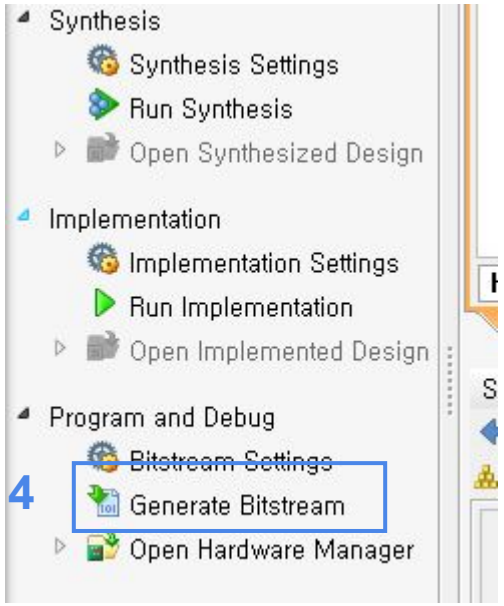
```
422     wire magic_code = (slv_reg0 == 32'h5555);
423
424     always @( posedge S_AXI_ACLK )
425     begin
426       if ( S_AXI_ARESETN == 1'b0 )
427         bram_state <= BRAM_IDLE;
428       else
429         case (bram_state)
430           BRAM_IDLE: bram_state <= (magic_code)? BRAM_READ : BRAM_IDLE;
431           BRAM_READ: bram_state <= BRAM_WAIT;
432           BRAM_WAIT: bram_state <= BRAM_WRITE;
433           BRAM_WRITE: bram_state <= (run_complete)? BRAM_IDLE: BRAM_READ;
434           default  : bram_state <= BRAM_IDLE;
435         endcase
436     end
```

Custom IP Design - Create Wrapper



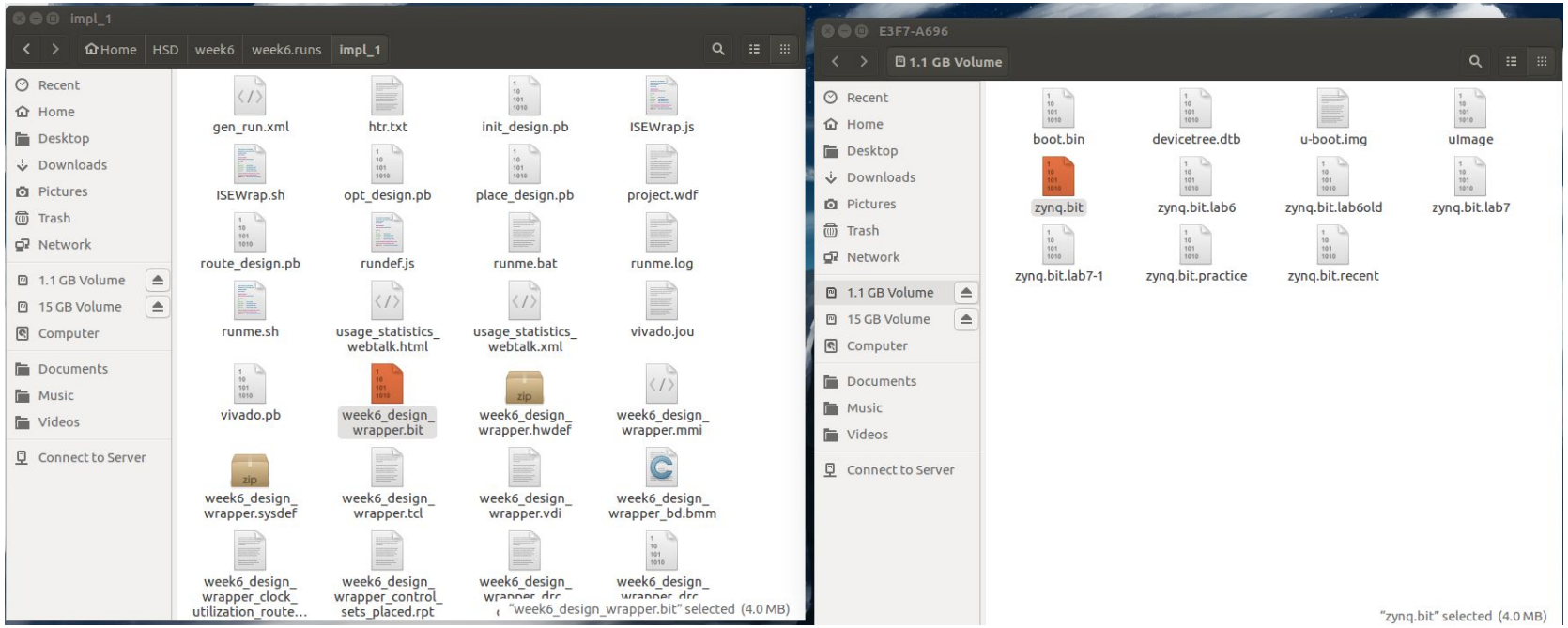






Review: Use your bitstream

- Copy the generated bitstream to your SD card.
  - Replace zynq.bit in the 1.1GB partition of SD card with your own bitstream.
  - Location of the generated bitstream:
    - `Host$ $(your_project)/$(your_project).runs/impl_1/design_1_wrapper.bit`



[ZEDBoard Hardware User's Guide, 2014]

Preparing the Board

1. Set configuration mode pins to **01100** (SD boot mode).
2. Insert USB-JTAG cable at J14.
3. Insert power cable at J20.
   then turn the power on with SW8.
   - LD13 lights up when it is on
4. Insert SD card,
   with **your own bitstream** in it.

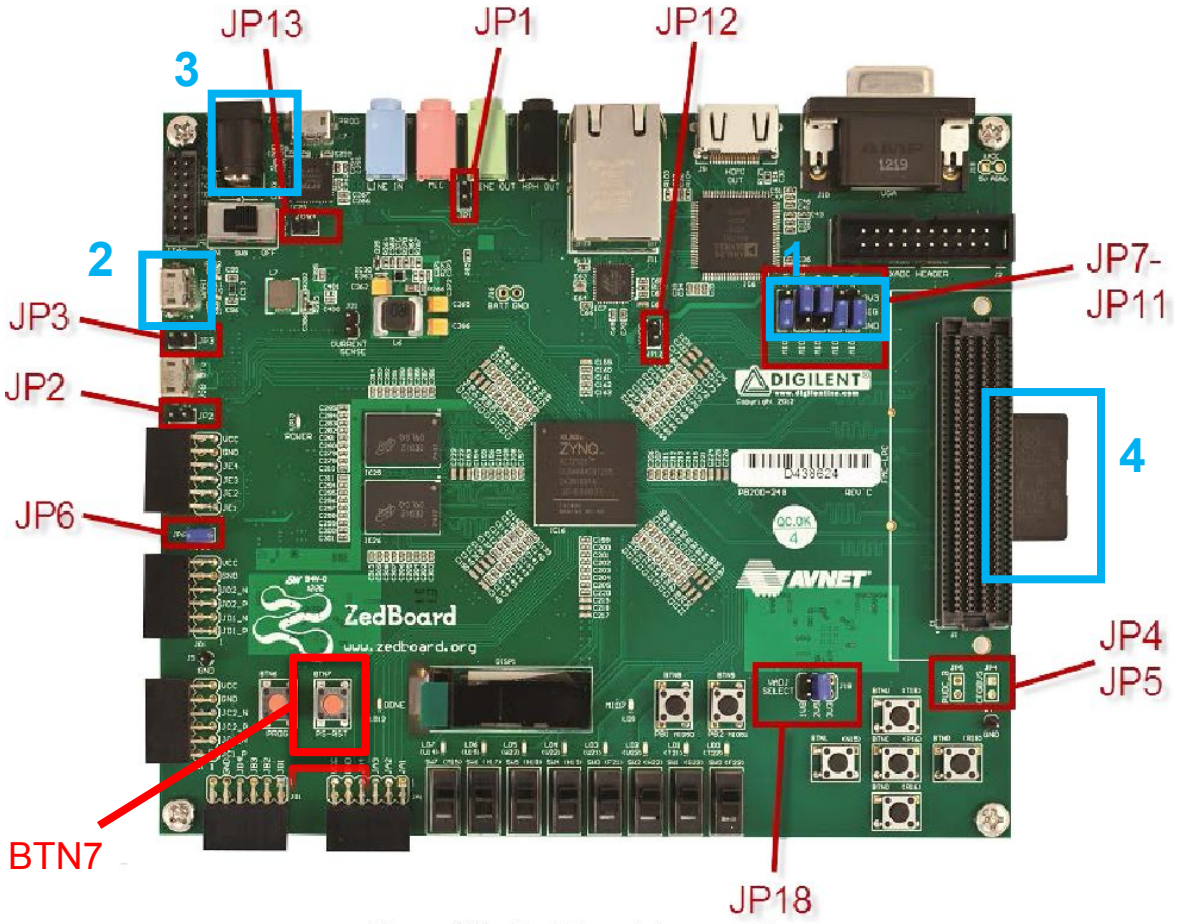
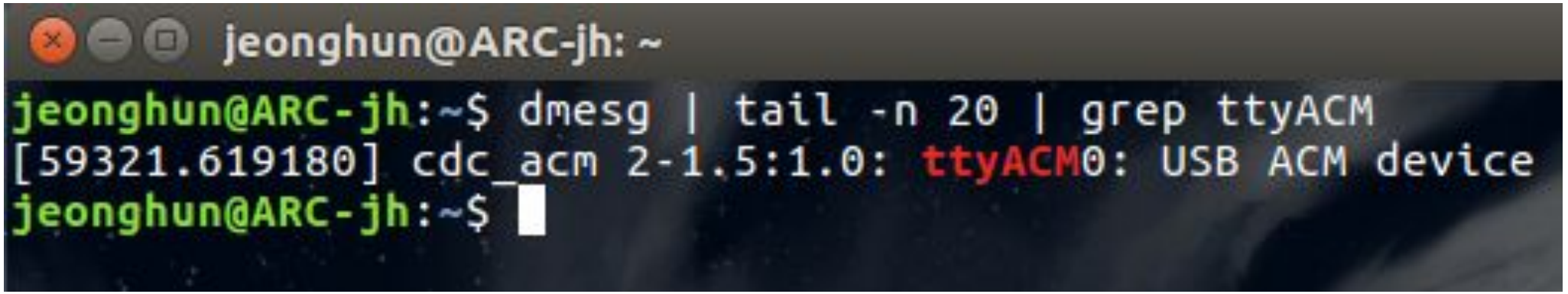


Figure 20 - ZedBoard Jumper Map

※ CAVEAT: J14 pin is very fragile!
If you cannot pull out a USB-JTAG cable from J14, press hooks with something pointed (sharp pencil, etc.).

Review: Connect your board to host (1)

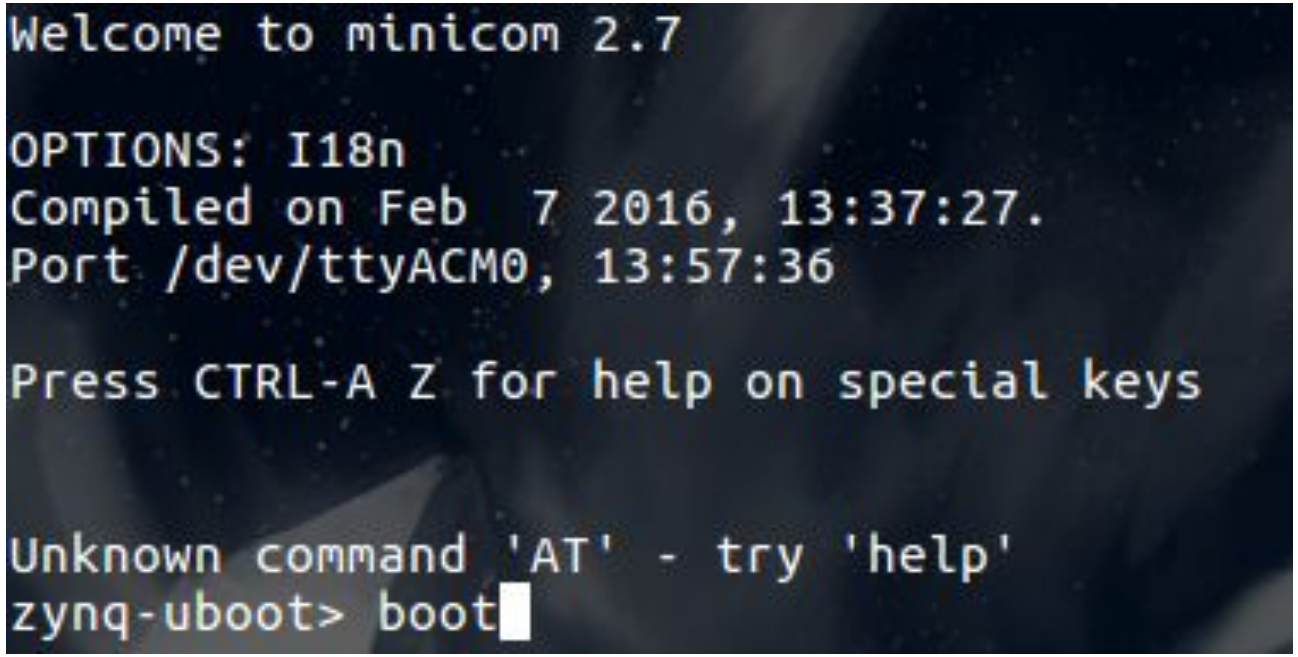
- Setup from host terminal
  - `Host$ dmesg | tail –n 20 | grep ttyACM`



- Connect to terminal of your board
  - `Host$ minicom –D /dev/ttyACMx`
    - Replace x with the number from above

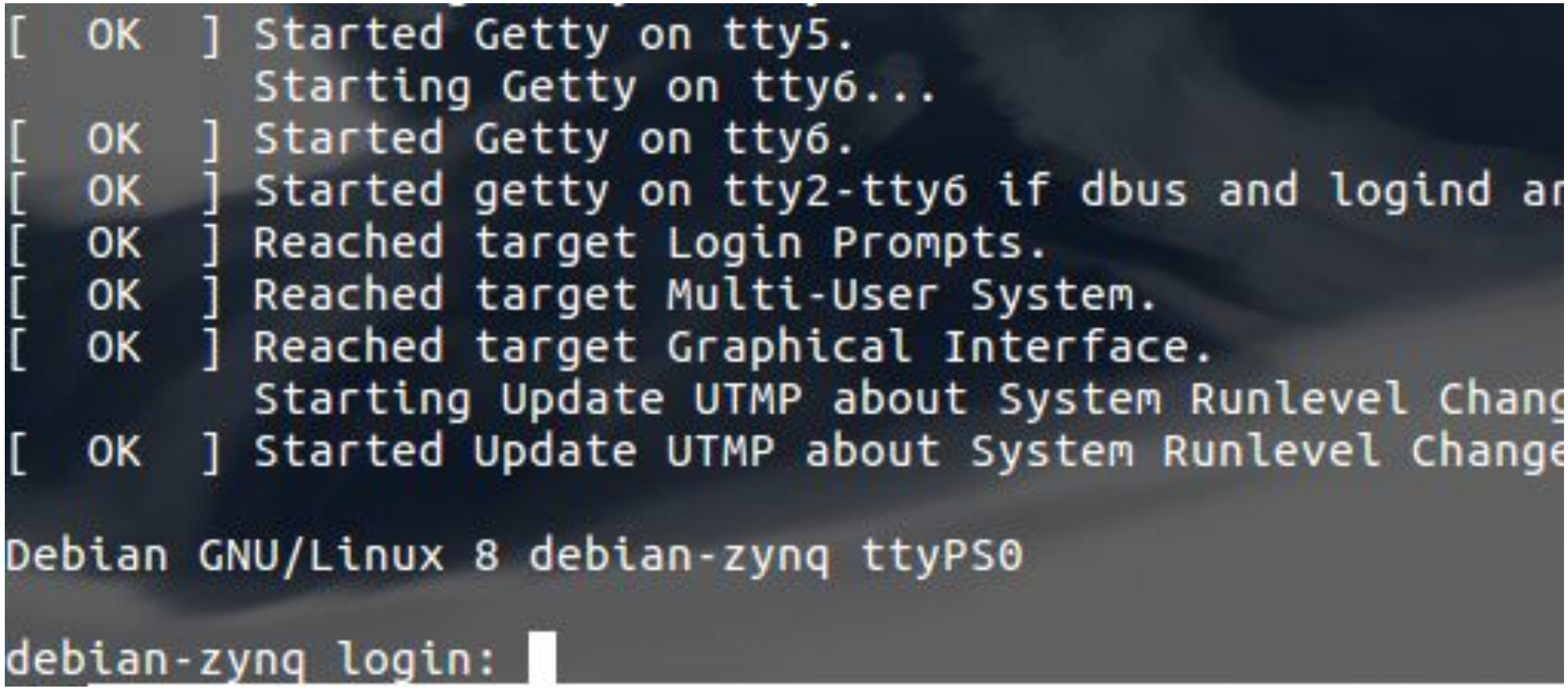
Review: Connect your board to host (2)

- Boot your board
  - `Board $> boot`
  - Wait 1~2 minutes for board to boot.
- If your Zedboard does not respond
  - Press BTN7(PS-RST).
- Before turn off the board
  - `BOARD$ sudo poweroff`
- To quit minicom
  - ctrl + a, q



Review: Connect your board to host (3)

- Login to Debian Linux
  - ID : zed / PW : zedzed



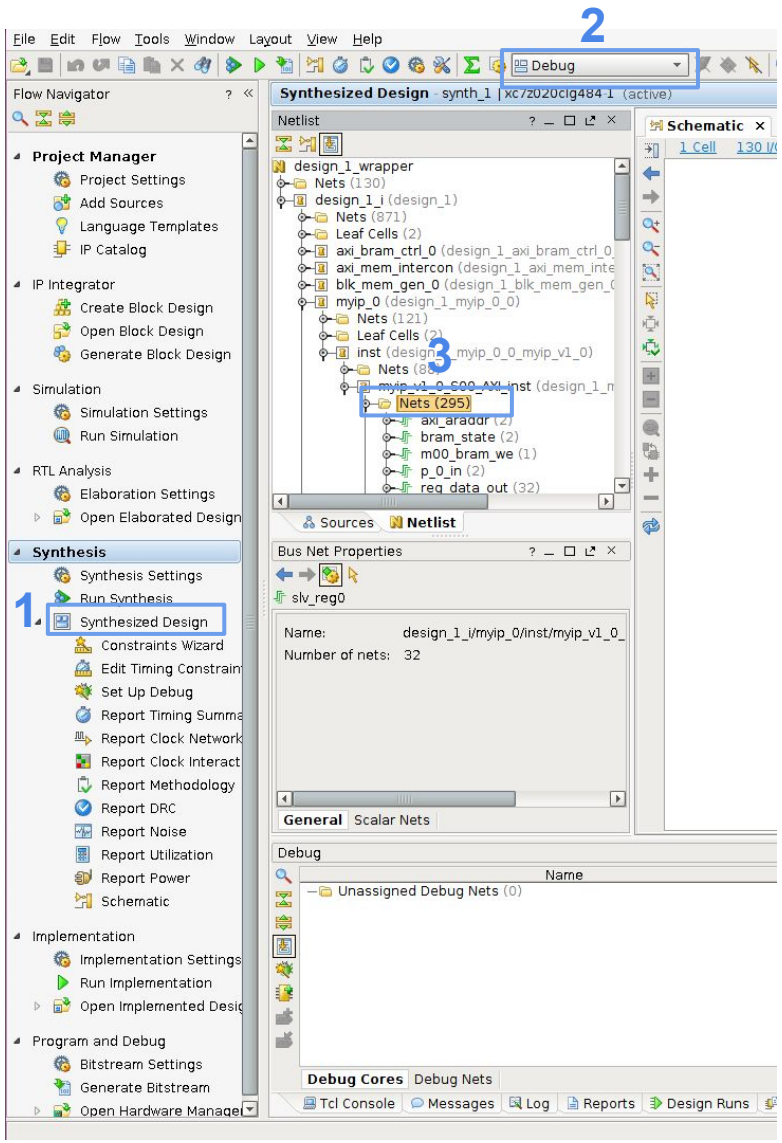
Run example program

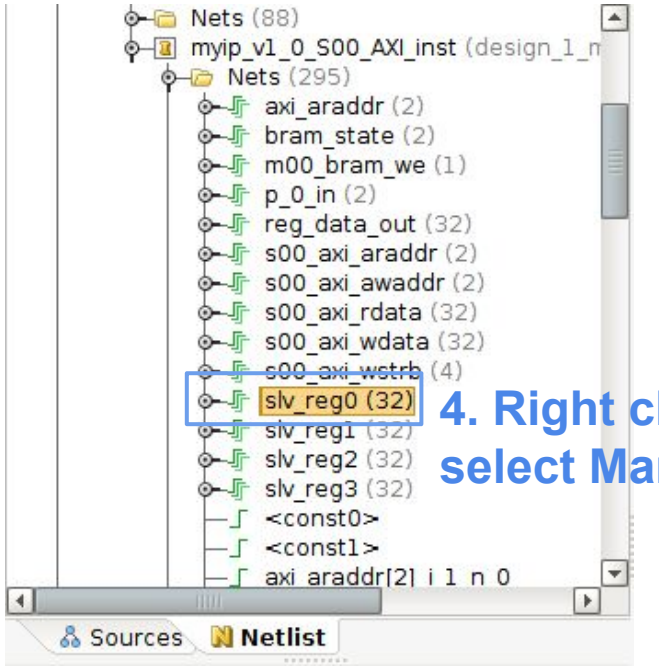
1. `BOARD$ git clone https://github.com/K16DIABLO/HSD_LAB7.git`
   - If that command does not work…
     - `sudo apt-get update`
2. `BOARD$ cd HSD_LAB7/lab7`
3. `BOARD$ make`

addr	FPGA(hex)	
0	0	
1	2	
2	4	
3	6	
4	0	
5	0	
6	0	
7	0	
addr	FPGA(hex)	
0	0	
1	2	
2	4	
3	6	
4	0	*(0) << 1
5	4	*(1) << 1
6	8	*(2) << 1
7	C	*(3) << 1

Tutorial: Integrated Logic Analyzer (ILA)

ILA - Mark nets to debug



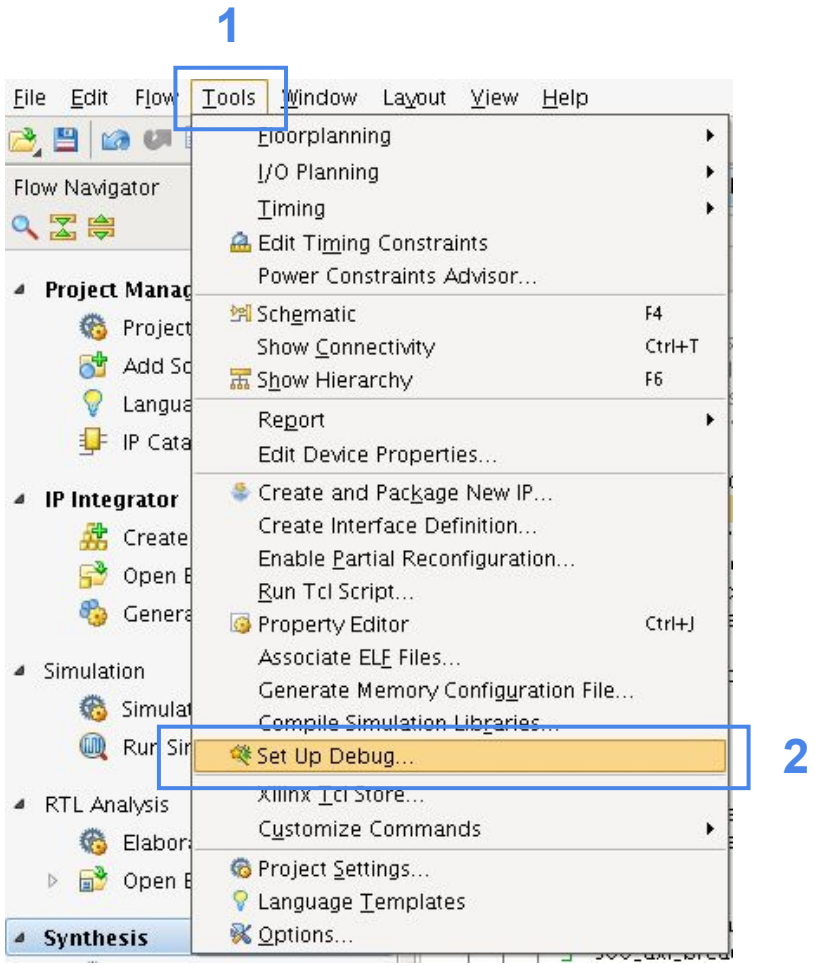


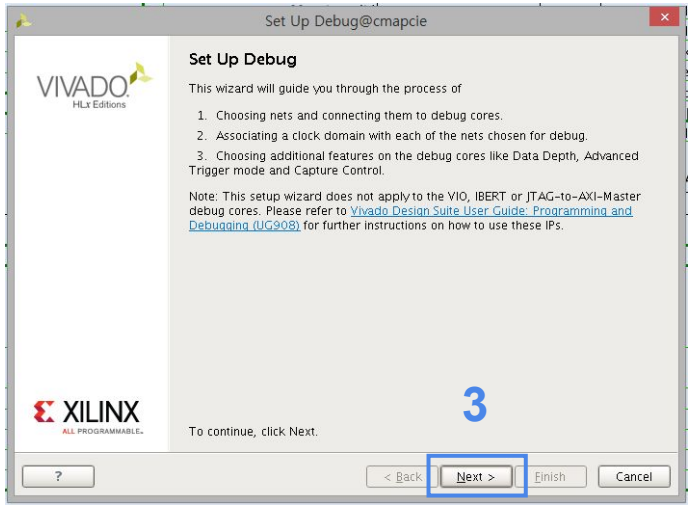
4. Right click and select Mark Debug

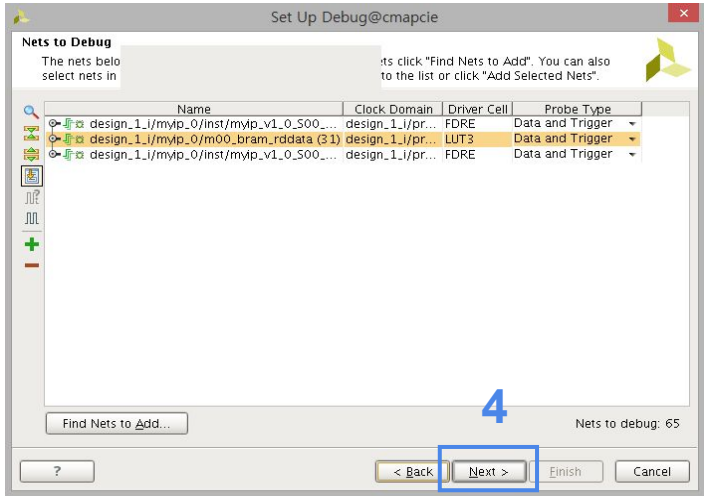
5. Mark Debug
design_1_i/myip_0/inst/myip_v1_0_S00_AXI_inst/Nets/bram_state (2)

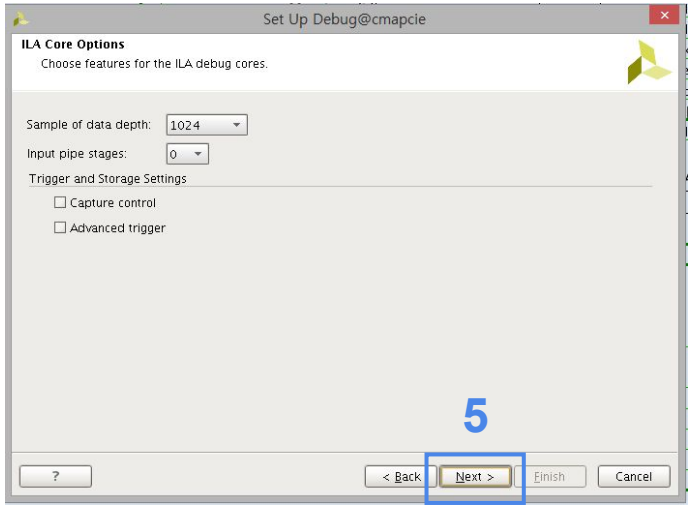
6. Mark Debug
design_1_i/myip_0/Nets/m00_bram_rddata (31)

ILA - Set up debugger(1)

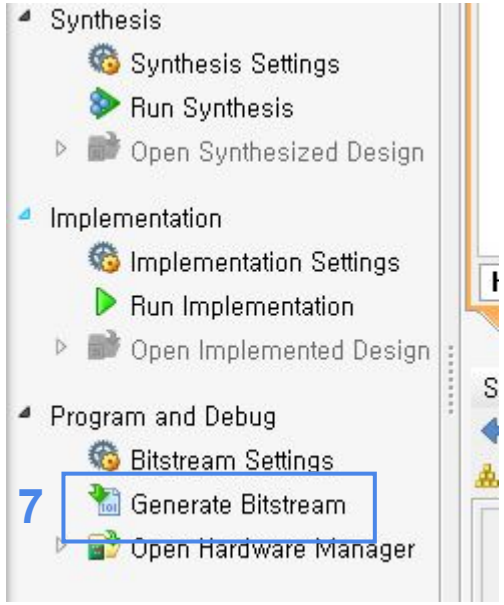


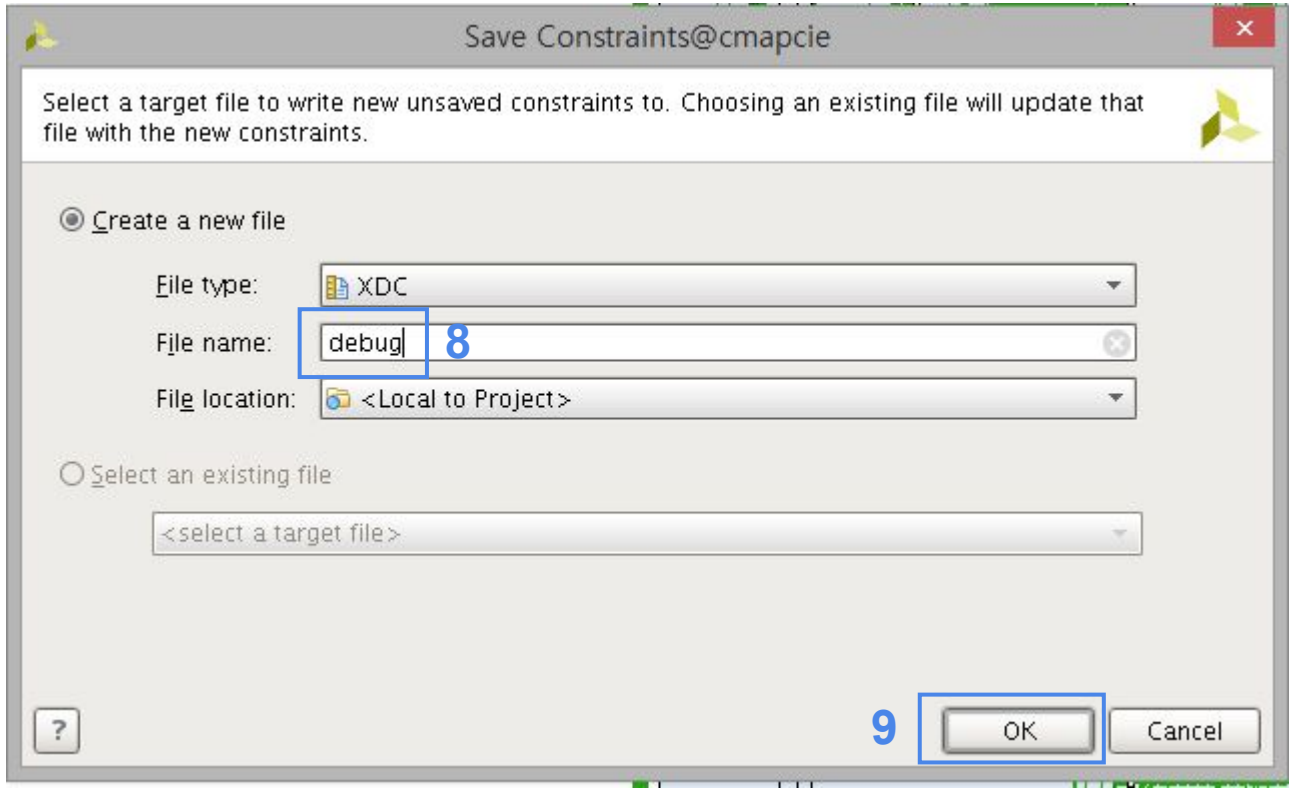






ILA - Set up debugger(2)





[ZEDBoard Hardware User's Guide, 2014]

Preparing the Board

1. Set configuration mode pins to **01100** (SD boot mode).
2. Insert USB-JTAG cable at J14 & J17.
3. Insert power cable at J20.
   then turn the power on with SW8.
   - LD13 lights up when it is on
4. Insert SD card,
   with **your own bitstream** in it.

※ CAVEAT: J14 pin is very fragile!
If you cannot pull out a USB-JTAG cable from J14, press hooks with something pointed (sharp pencil, etc.).

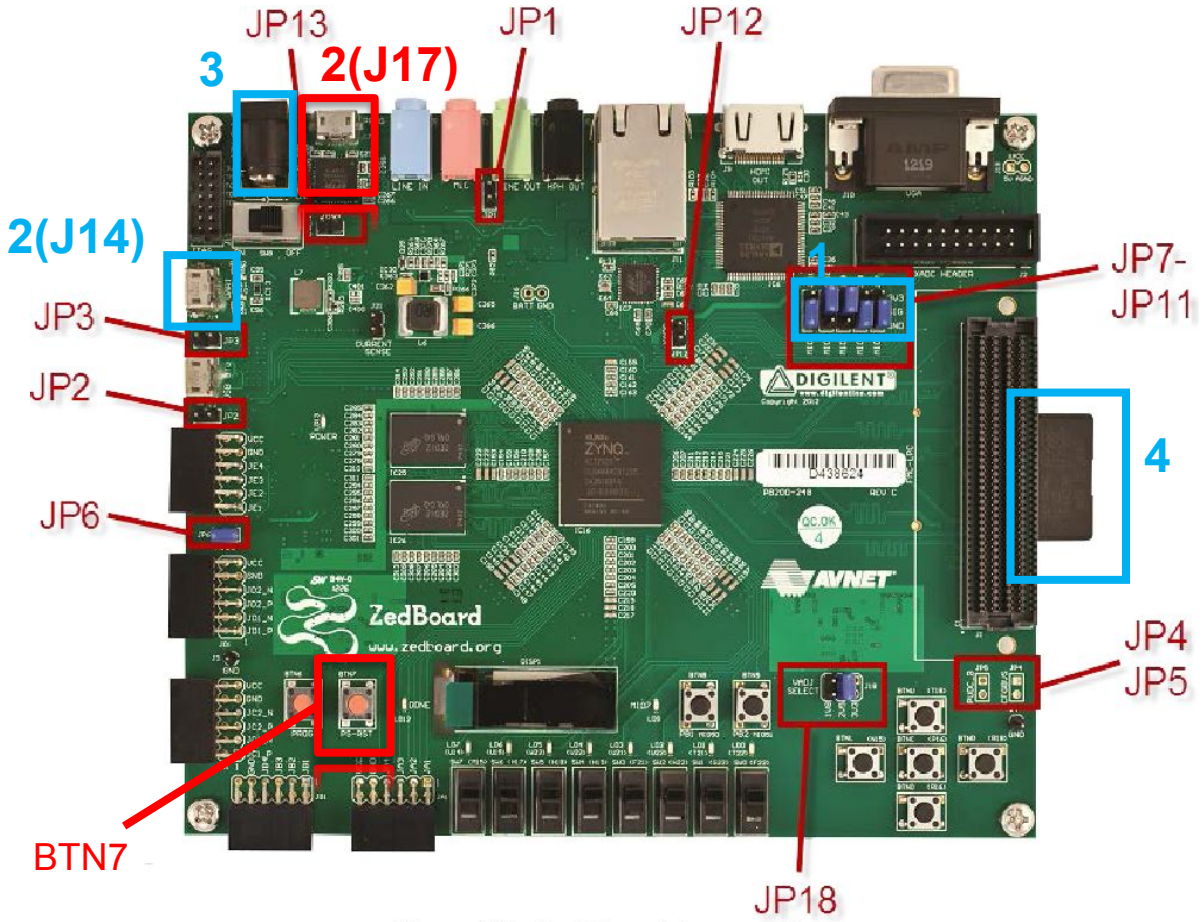
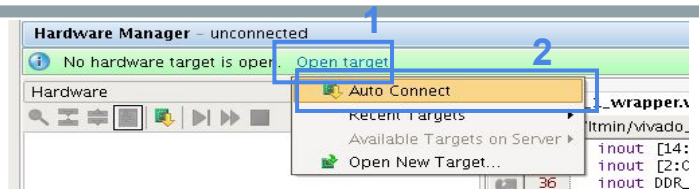
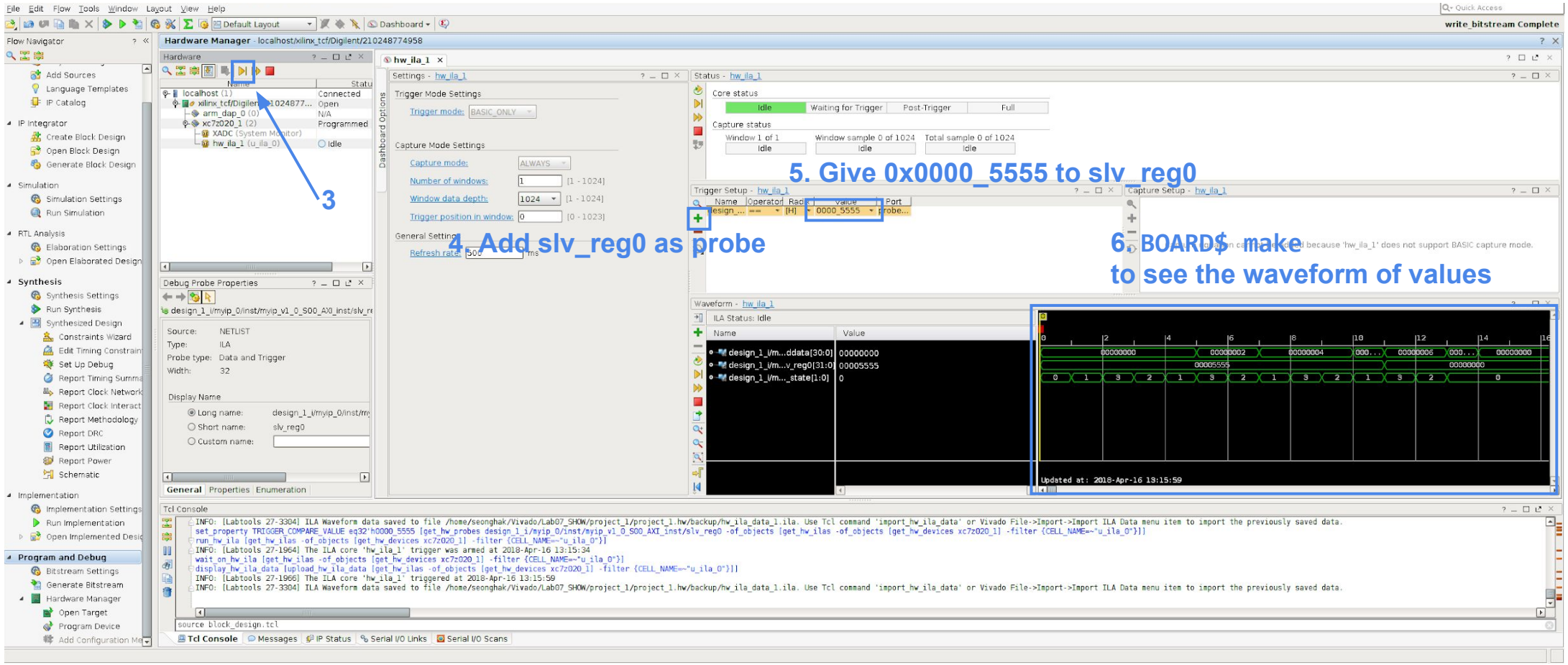


Figure 20 - ZedBoard Jumper Map

ILA - Debug


Hardware Manager - unconnected
No hardware target is open
Open target
Hardware
Auto Connect
Recent Targets
Available Targets on Server
Open New Target...
1
2


Hardware Manager - localhost/xilinx_tcf/Digilent/210248774958
write_bitstream Complete
3
5. Give 0x0000_5555 to slv_reg0
4. Add slv_reg0 as probe
6. BOARD$ make
to see the waveform of values
Updated at: 2018-Apr-16 13:15:59

Practice: Report

Practice

- Submit a 1-page report with brief explanation of the following:
  - HW IP: Behavior of the shifter IP based on design sources
    - myip_v1_0.v
    - myip_v1_S00_AXI.v
  - HW System: Composition of the block design of Lab 7
  - SW: Role of each partition of the SD card and program codes.
    - 1st partition (1.1GB partition)
    - 2nd partition (15GB partition)
    - main.c
    - Makefile

Grading policy

- Check lists
  - Shifter IP runs (30 points)
  - Waveform of marked variables is displayed (30 points)
  - Finished tutorial in class (10 points)
  - Report (30 points)
- Submit “L7.**pdf**” only on eTL
  - Due : 4/24 (Tue) PM 11:59
- Show that your board works
  - In class / office hour (Tue) recommended
  - If you cannot come to the office hour, take a screen shot of your working version and mail us.