Lec 19 → now 20

ECE 383 - Embedded Computer Systems II

Lecture 19 - Soft Core (MicroBlaze) + Custom IP with Interrupt

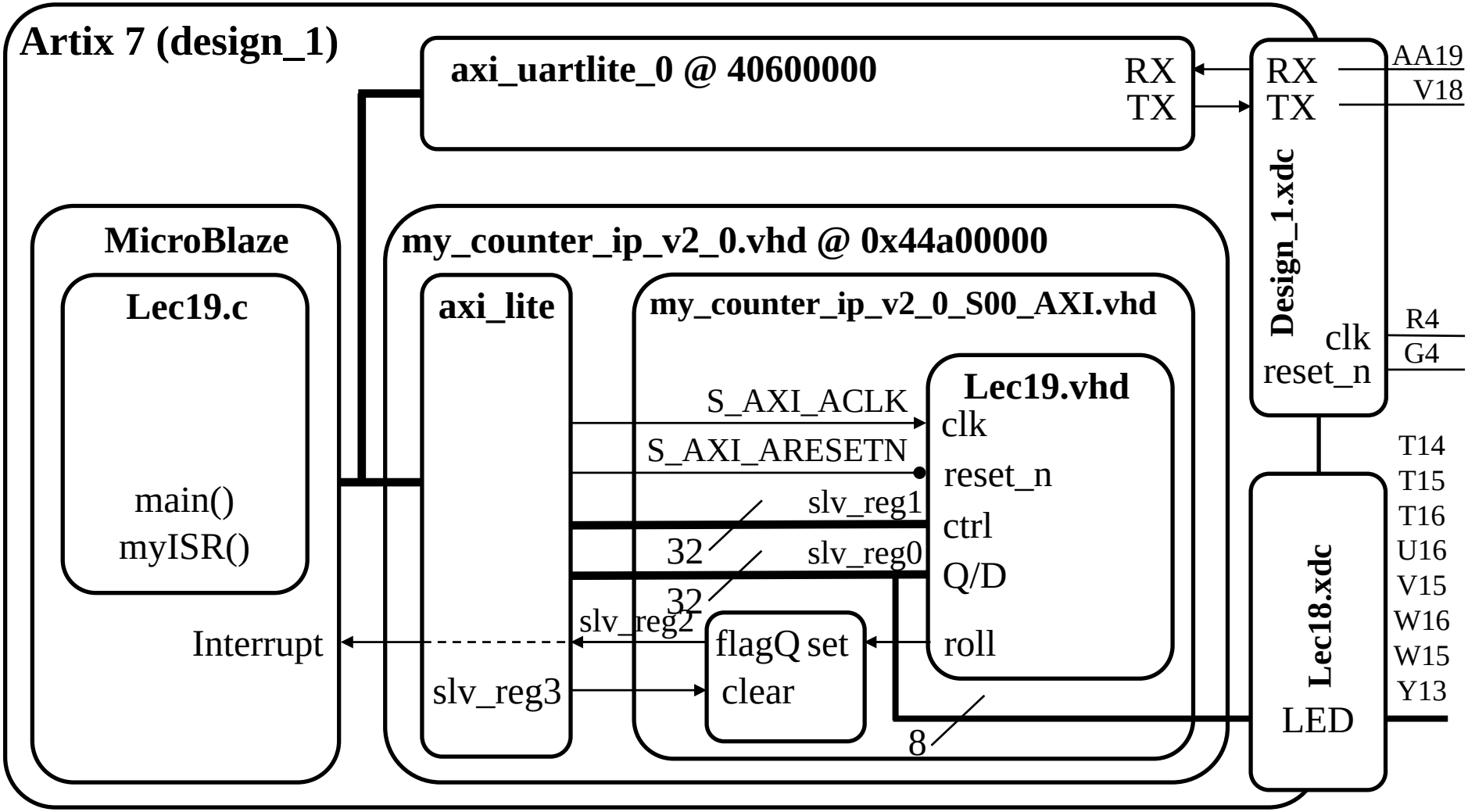
UNITED STATES AIR FORCE ACADEMY

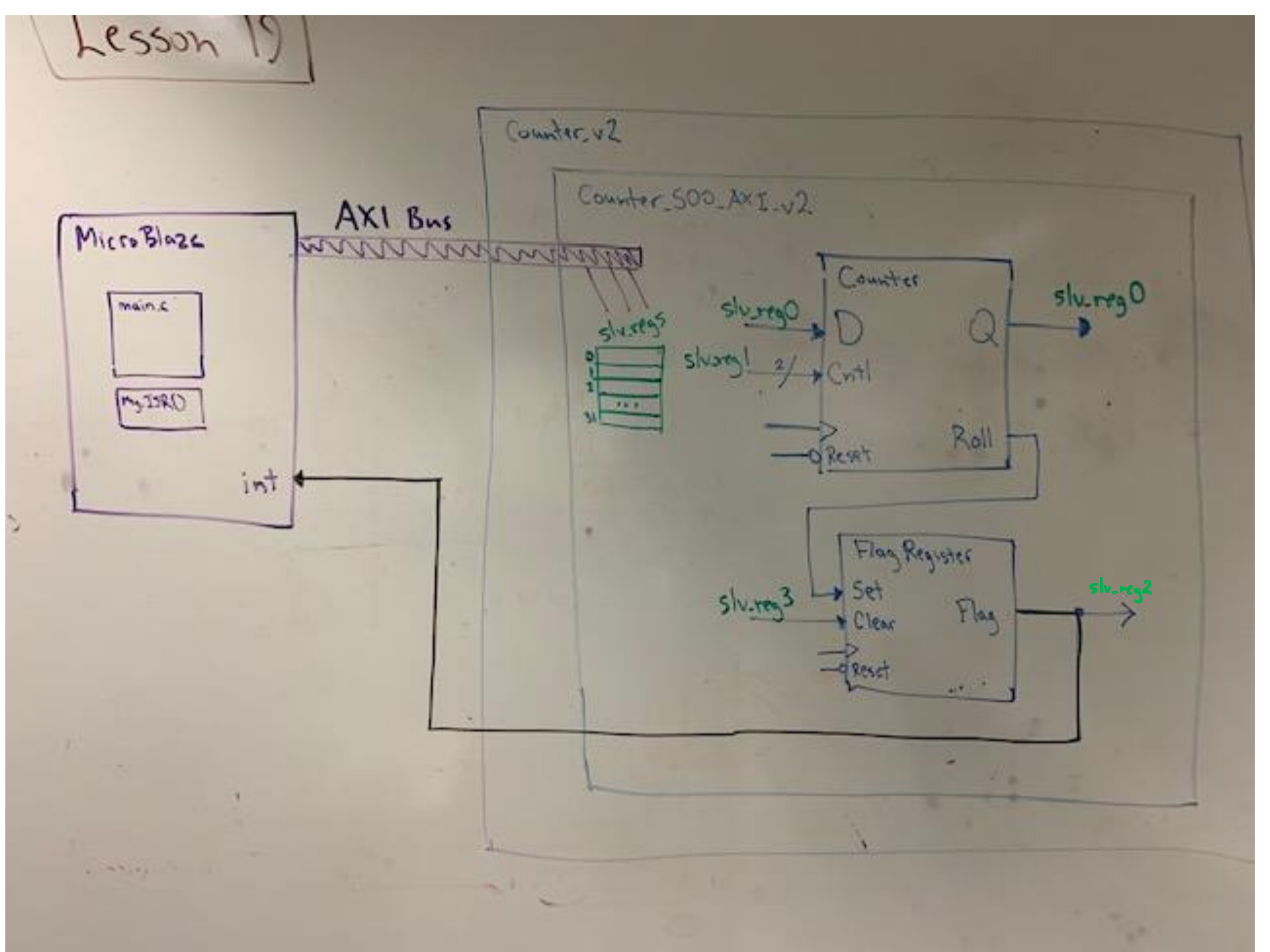
Lesson Outline

- **MicroBlaze + Custom IP with Interrupt**

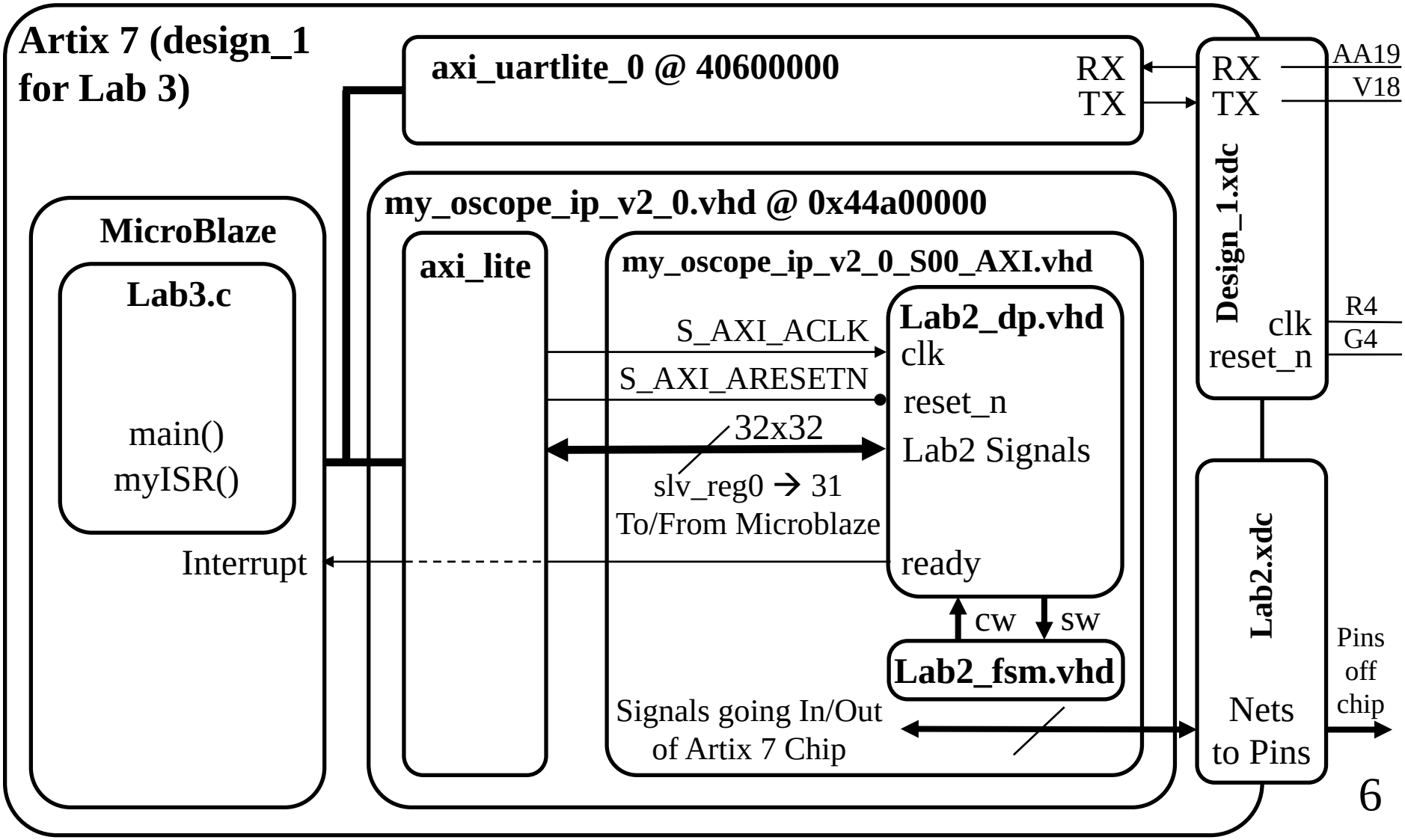
MicroBlaze + Custom IP with Interrupt

MicroBlaze + Custom IP with Interrupt
What we're building today

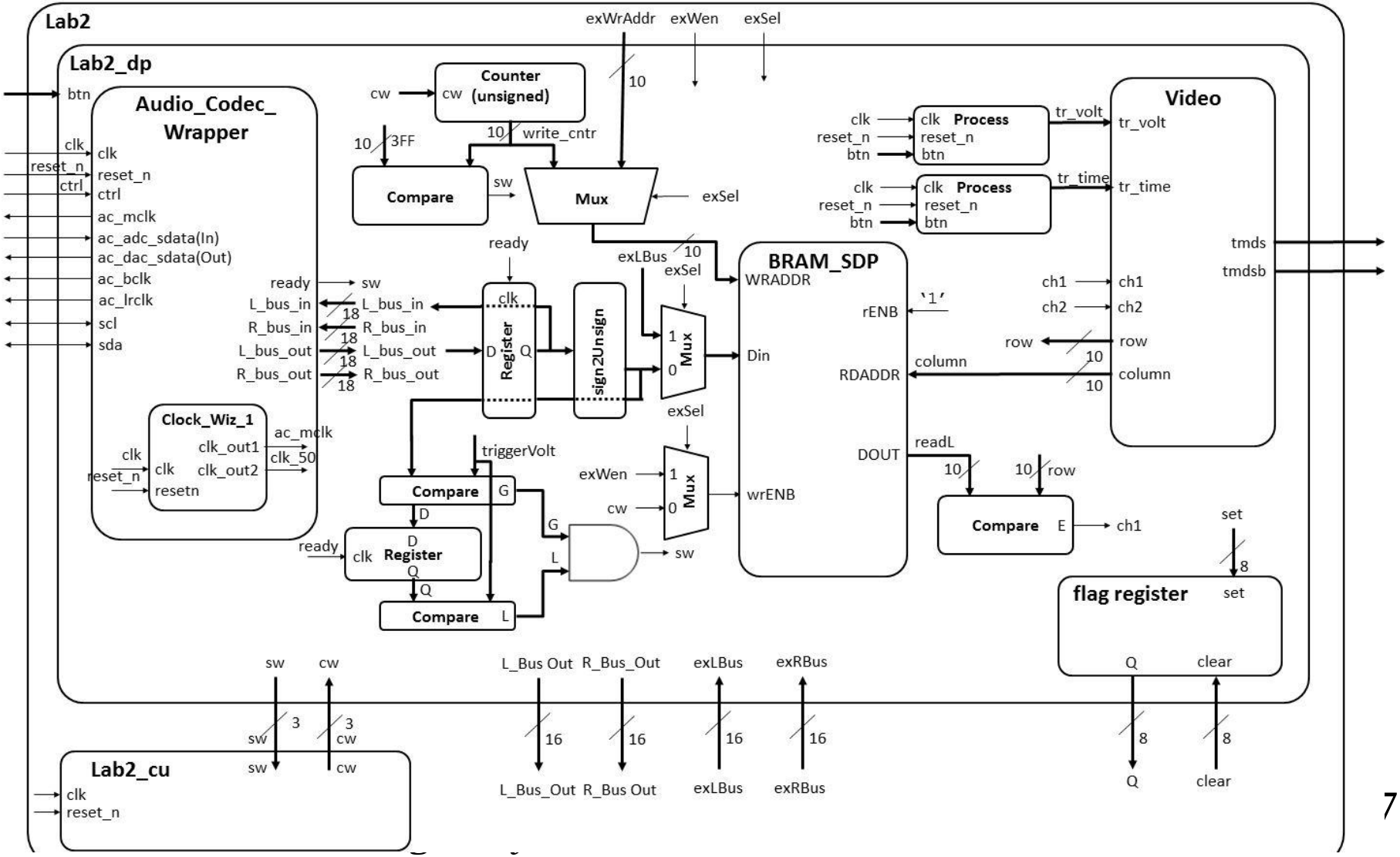



Lesson 19
Counter_v2
Counter_S00_AXI_v2
AXI Bus
MicroBlaze
main.c
myISR()
int
slv_regs
0
1
2
31
slv_reg0
slv_reg1
2
Counter
D
Q
Cntl
Reset
Roll
slv_reg0
Flag Register
Set
Clear
Reset
Flag
slv_reg3
slv_reg2

What were doing Lab 3

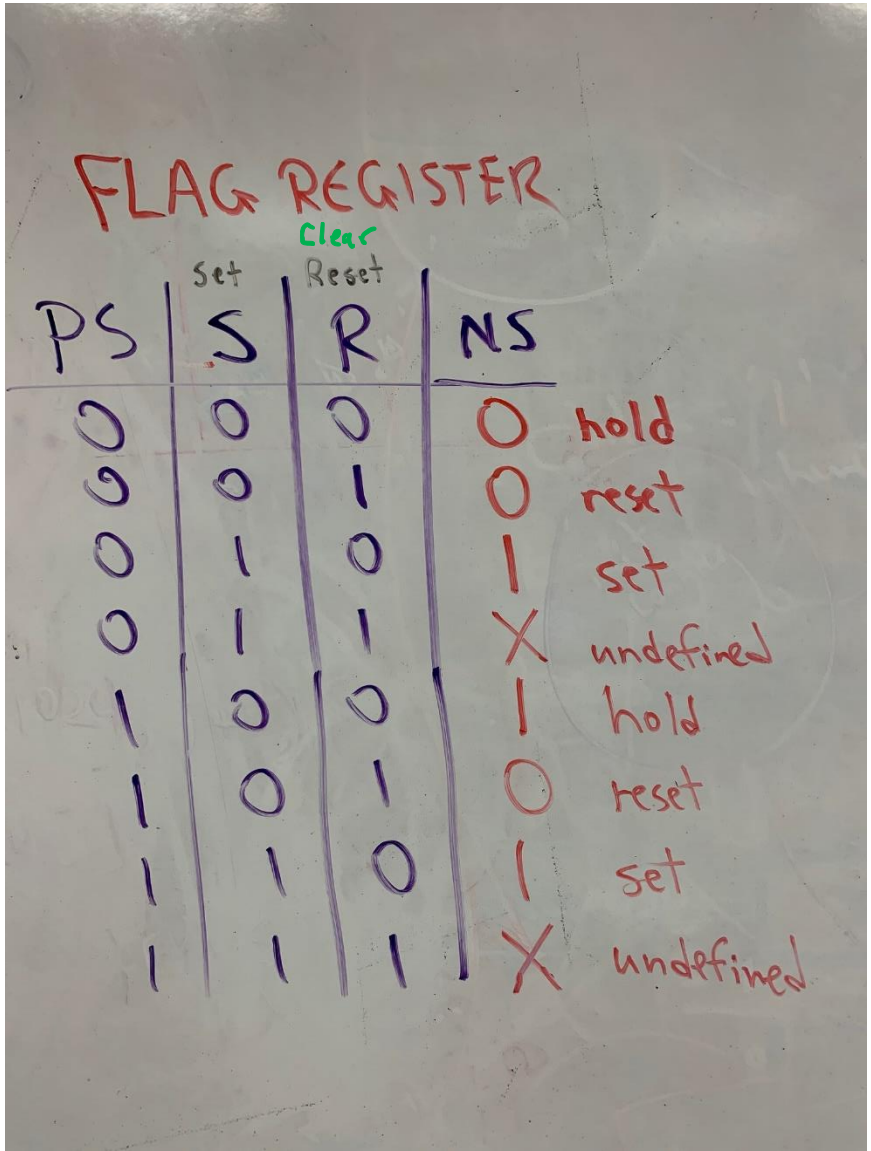


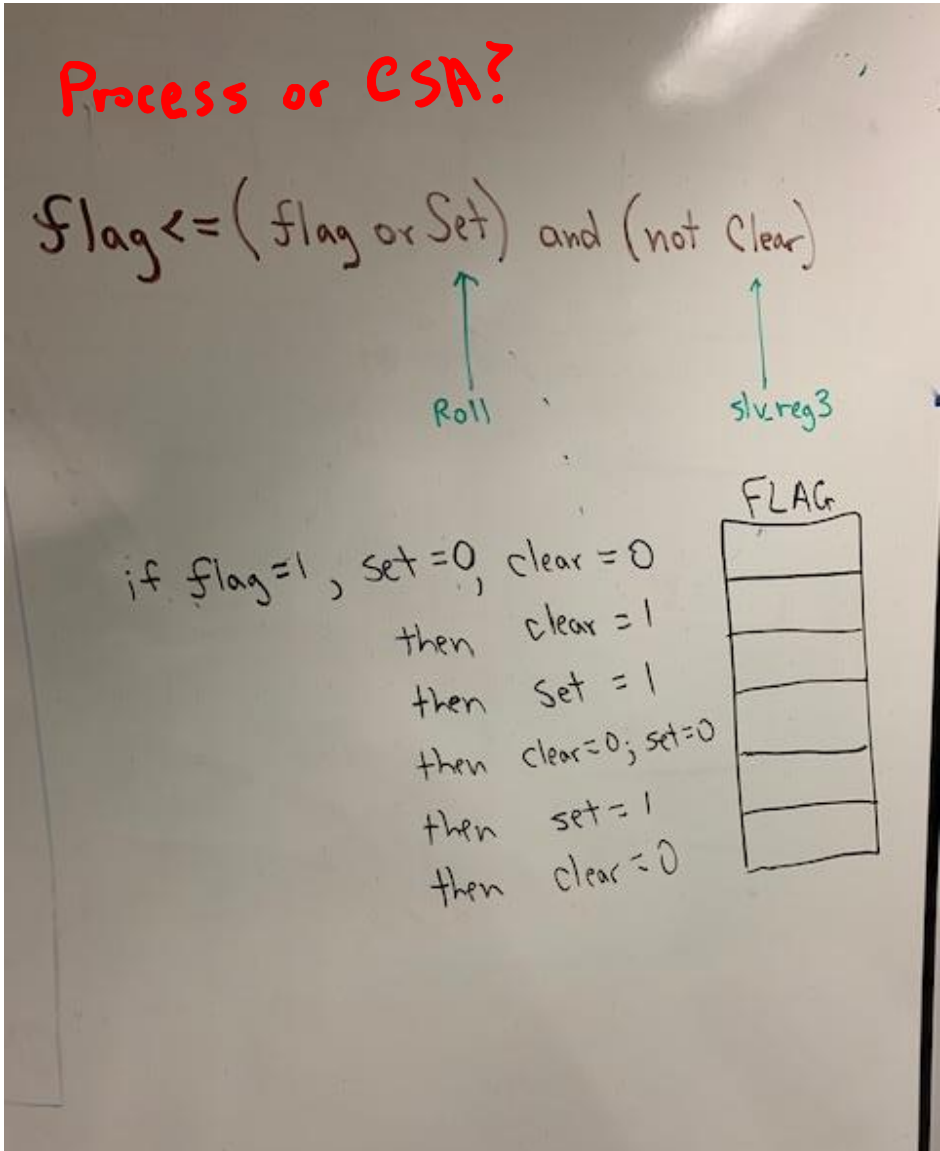
Lab 2 – Architecture



SKIP

Hints on the Flag Register





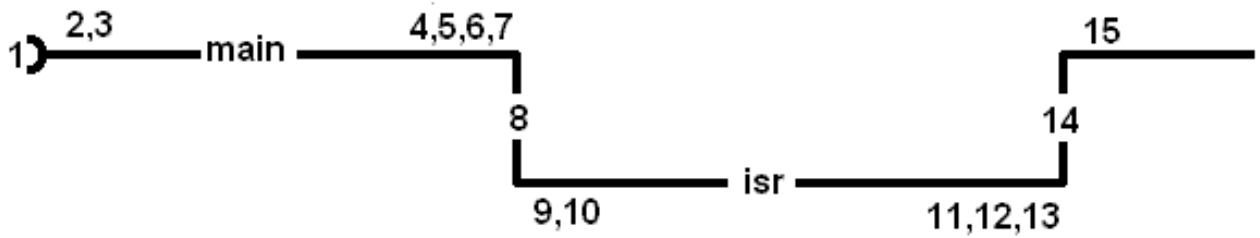
How do interrupts work?

- **Polling versus Interrupts?**

Interrupts

GENERIC

Interrupts are used when you want to your system to do more than one thing at a time. An interrupt service routine (ISR) is a subroutine called by hardware. The following figure illustrates the process of "calling" and returning from an ISR.



1. MCU powers up, jumps to RESET vector
2. MCU starts execution of main
3. Dynamic configuration
   - configure hardware
   - clear hardware interrupt flag
   - enable hardware interrupt
4. Event occurs which sets interrupt flag
5. MCU stops running main
6. MCU saves PC
7. MCU disables interrupts
8. Executes "GOTO ISR" at interrupt vector address
9. ISR: Poll interrupt flags
10. ISR: Execute appropriate code in ISR
11. ISR: Clear interrupt flag
12. ISR: executes rted
13. Interrupts are enabled
14. PC is restored
15. MCU resumes running main

Lec19.c

```
#include <xil_exception.h>
u16 isrCount = Ø;     // Global variable
void myISR(void);

int main(void) {

    microblaze_register_handler((XInterruptHandler) myISR, (void *) 0);
    microblaze_enable_interrupts();

    stuff();

} // end main


void myISR(void) {
        isrCount = isrCount + 1;
    Xil_Out8(countClearReg, 0x01);                                  // Clear the flag and then you MUST
        Xil_Out8(countClearReg, 0x00);                              // allow the flag to be reset later
}
```

MicroBlaze + Custom IP – Workflow

- Follow Lec19_Install_short_version.pdf see last slide!
  - The following slides are for reference

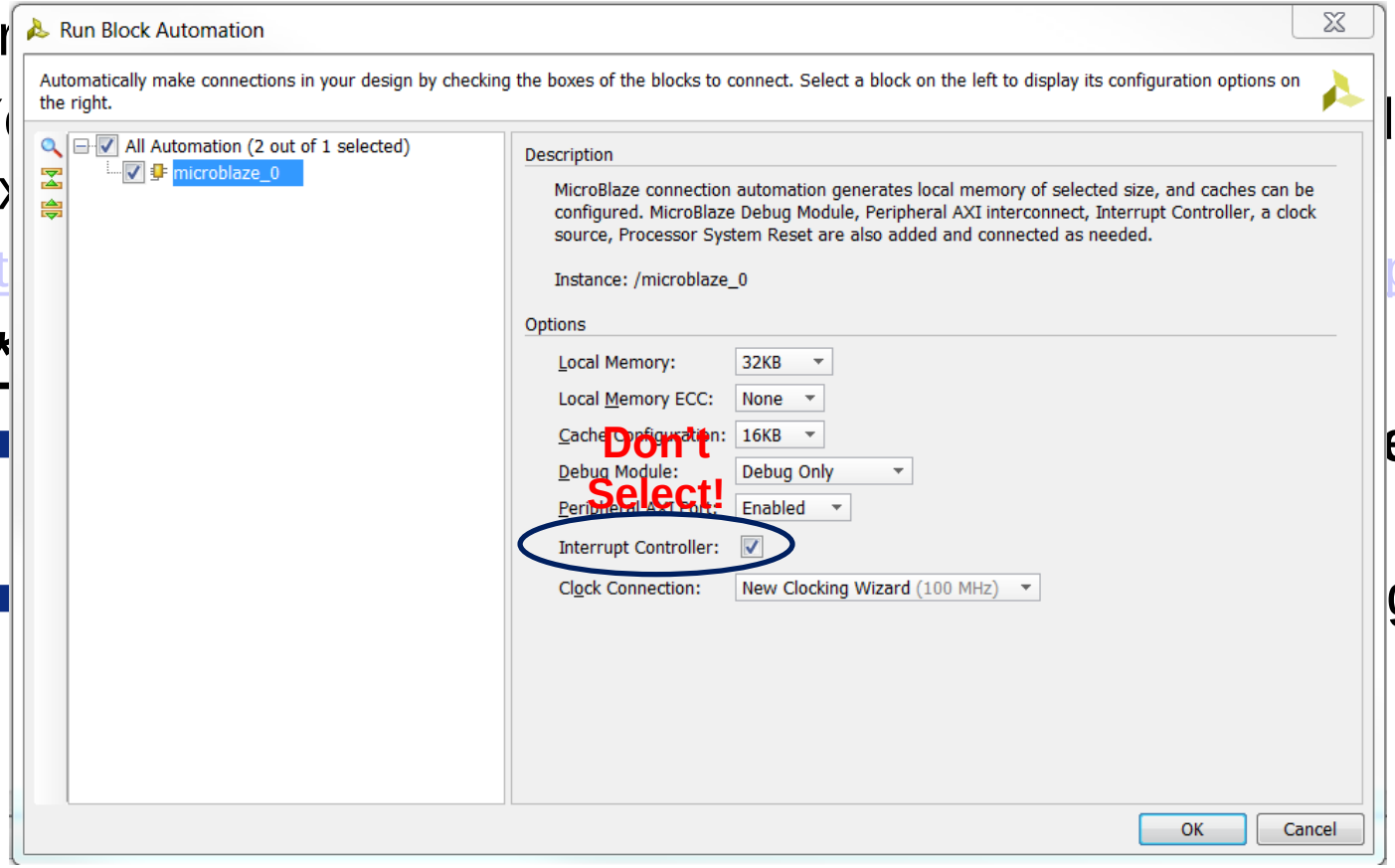
- The work flow has three main steps.
  1. Define a new hardware design (MicroBlaze + axi_uartlite) in Vivado IP Integrator (using the MicroBlaze Tutorial from Lecture 17)
  2. Create and package new custom IP (your custom hardware) and import it into your Vivado design
  3. Program the resulting hardware in the SDK environment.
- Lets start with the first step.

Xilinx Vivado – IP Integrator

- This step requires that you start a new hardware design (MicroBlaze + axi_uartlite) in Vivado IP Integrator in a new project called Lecture_19.
- You will add a new Block Design with a MicroBlaze and axi_uartlite following the MicroBlaze Tutorial.
- http://ece.ninja/383/hand/Nexys_Video_MicroBlaze_Tutorial.pdf
- **<u>***Deviation from Lecture 17 Tutorial***</u>**
  - **Do not include the MicroBlaze Interrupt Controller check box.**
  - If you do you could probably delete it from your design

Xilinx Vivado – IP Integrator

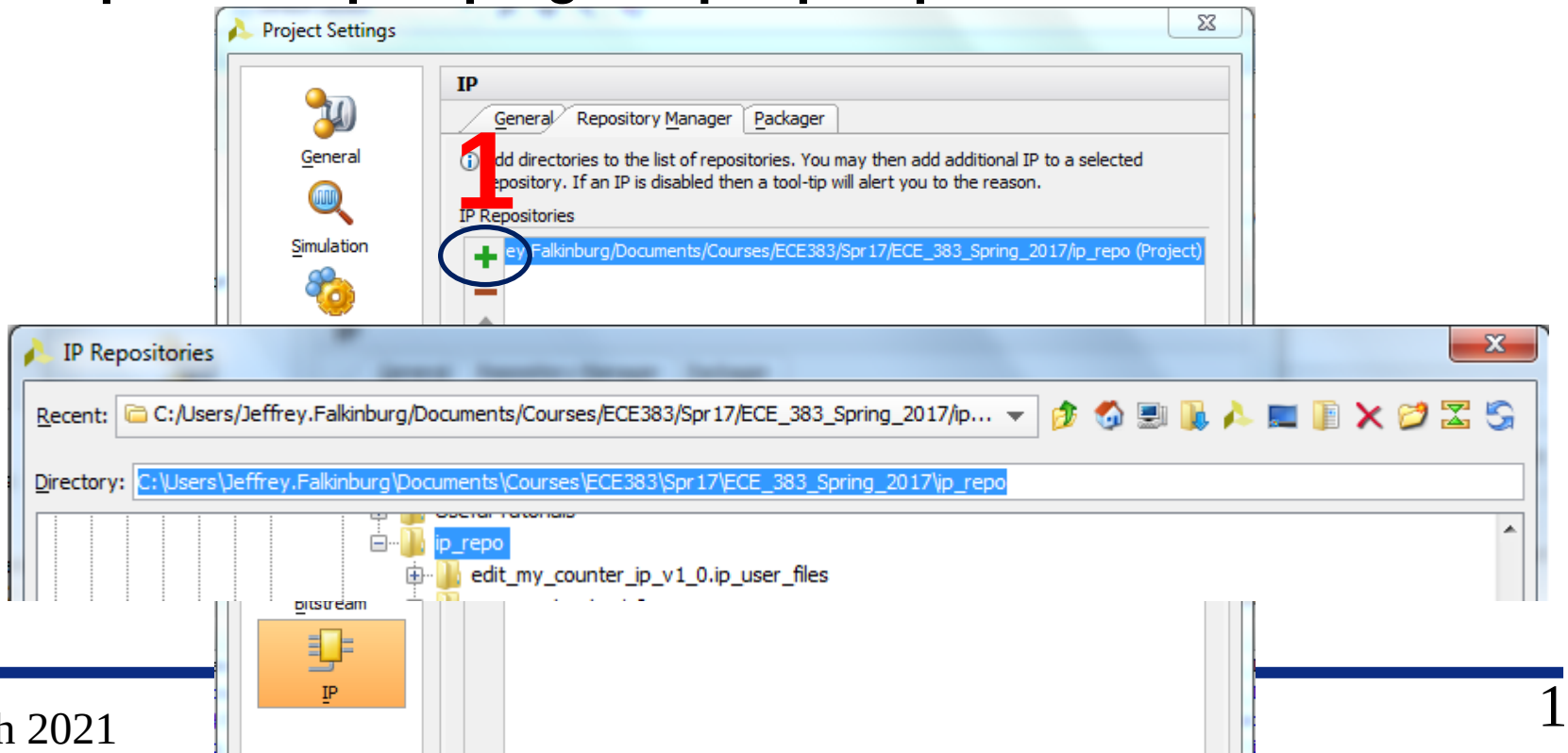
- This step requires that you start a new hardware design (MicroBlaze + axi_uartlite) in Vivado IP Integrator in a new pr
- Y l ax
- ht pdf
- **
  - er
  - gn



IP Catalog – Adding IP Repo

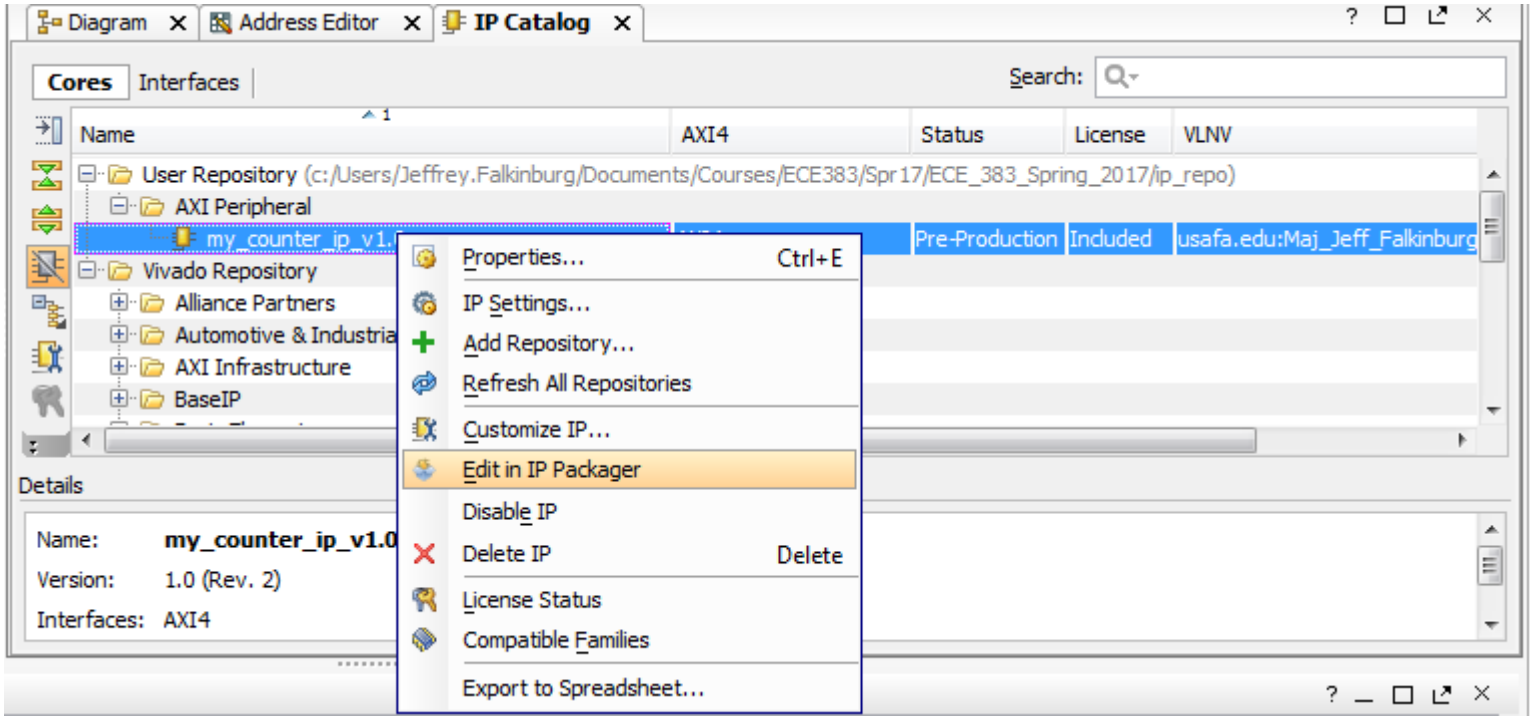
- **Open IP Catalog Settings and click on Repository Manager and add your IP Repo to your IP Repositories**

  /path_to_ip_repo/git_repo/ip_repo



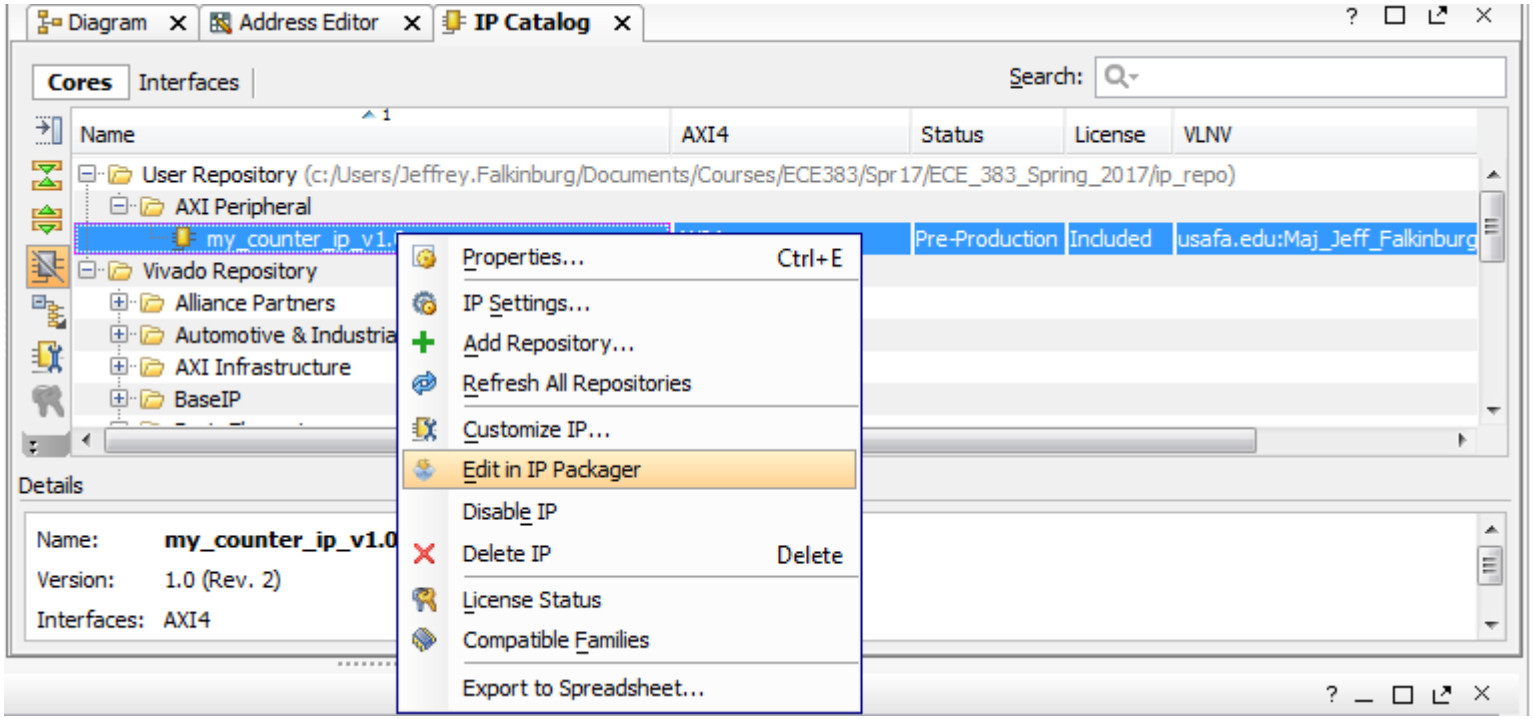
Edit/Create New IP Package

- **Edit Counter in IP Packager or create a new IP package**
- **I chose to create a new package with a new version.**



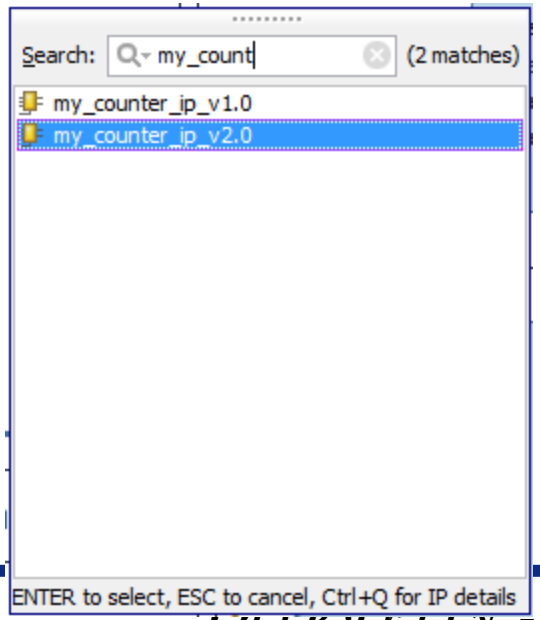
Edit/Create New IP Package

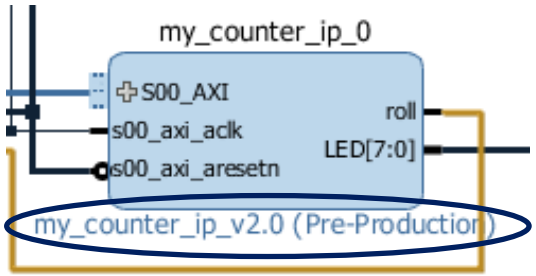
- **Expose the Roll Signal to the Artix 7 (design_1) block diagram by following the LED port maps**



Xilinx Vivado – Create and Package Custom IP

- 8. Add Custom IP to your design
  - 8.1) In the project manager page of the original window, click **Open Block Design**. This adds a block design to the project.
  - 8.2) Use the **Add IP** button to add our **v2.0 of our Lec 10 Counter IP Core** with the exposed roll signal.

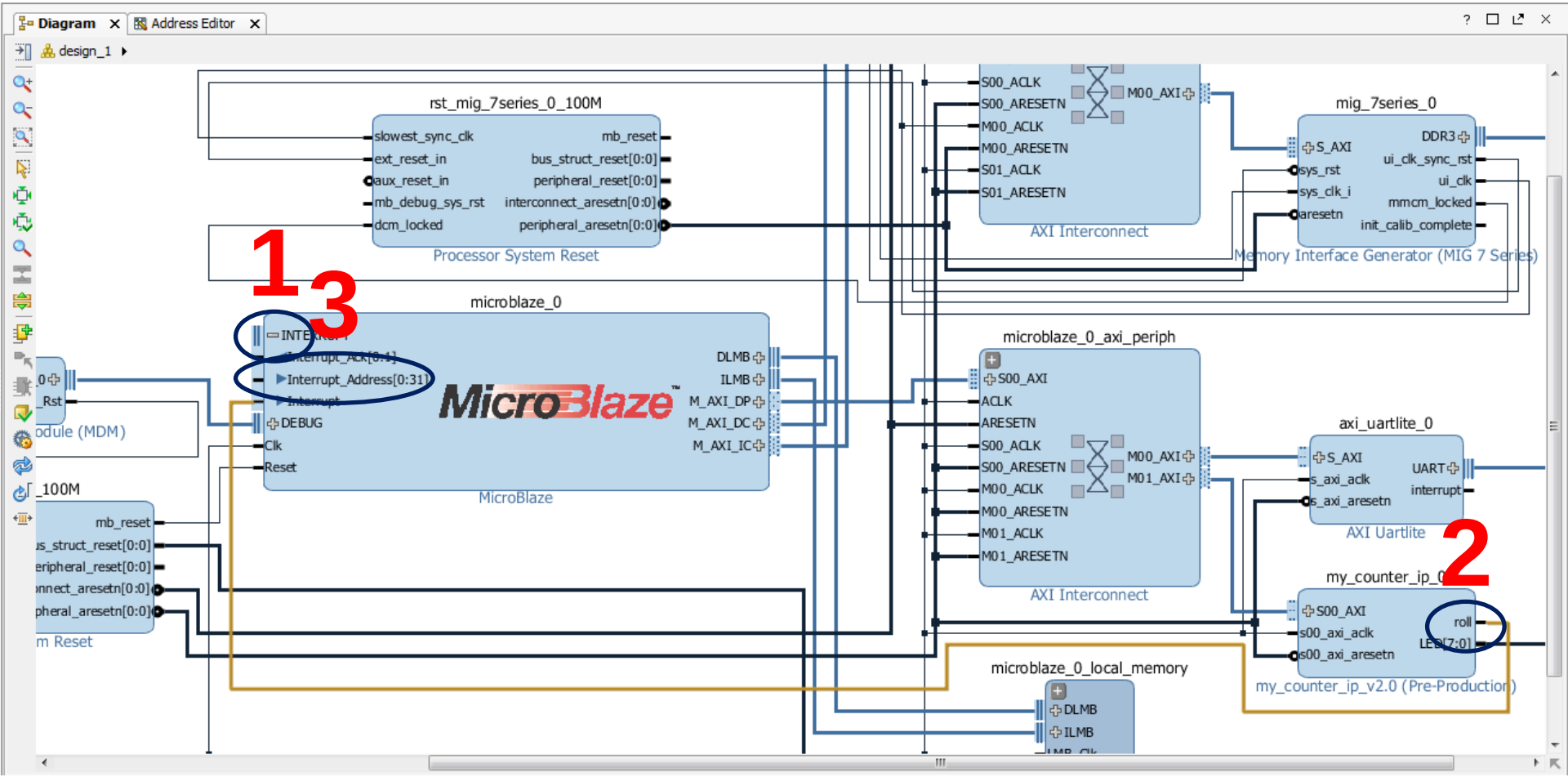




Notice it is v2.0

Edit/Create New IP Package

- **Click the '+' sign by the MicroBlaze to connect the Roll Signal to the MicroBlaze Interrupt input directly**



Verify Design

- You should verify the addressing for all your design components before continuing.
- Verify that the base addresses are the same addresses used in the template C-code.
- Should be no changes at this time.

1

Diagram | Address Editor

Cell	Slave Interface	Base Name	Offset Address	Range	High Address
microblaze_0					
Data (32 address bits : 4G)					
axi_uartlite_0	S_AXI	Reg	0x4060_0000	64K	0x4060_FFFF
microblaze_0_local_memory/dlmb_bram_if_cntlr	SLMB	Mem	0x0000_0000	32K	0x0000_7FFF
mig_7series_0	S_AXI	memaddr	0x8000_0000	512M	0x9FFF_FFFF
my_counter_ip_0	S00_AXI	S00_AXI_reg	0x44A0_0000	64K	0x44A0_FFFF
Instruction (32 address bits : 4G)					
microblaze_0_local_memory/ilmb_bram_if_cntlr	SLMB	Mem	0x0000_0000	32K	0x0000_7FFF
mig_7series_0	S_AXI	memaddr	0x8000_0000	512M	0x9FFF_FFFF

2

3

Verify Design

- You should verify the addressing for all your design components before continuing.
- Verify that the base addresses are the same addresses used in the template C-code.
- Should be no changes at this time.

Diagram | Address Editor

Cell	Slave Interface	Base Name	Offset Address	Range	High Address
microblaze_0					
Data (32 address bits : 4G)					
axi_uartlite_0	S_AXI	Reg	0x4060_0000	64K	0x4060_FFFF
microblaze_0_local_memory/dlmb_bram_if_cntlr	SLMB	Mem	0x0000_0000	32K	0x0000_7FFF
mig_7series_0	S_AXI	memaddr	0x8000_0000	512M	0x9FFF_FFFF
my_counter_ip_0	S00_AXI	S00_AXI_reg	0x44A0_0000	64K	0x44A0_FFFF
Instruction (32 address bits : 4G)					
microblaze_0_local_memory/ilmb_bram_if_cntlr	SLMB	Mem	0x0000_0000	32K	0x0000_7FFF
mig_7series_0	S_AXI	memaddr	0x8000_0000	512M	0x9FFF_FFFF

Validate and Export Design

1. First click validate design_1
2. Regenerate the design_1 HDL wrapper.
3. Finally you need to generate the Generate Design bitstream
4. Take a coffee break while it builds

SDK Project

- Start with a “Hello World” project once in the SDK.
- Rename the hello_world.c to Lec19.c and use the given Lec19.c code to get started
- Modify the code to handle the interrupt generated from the counter and increment a counter variable for display.