Lee 19 > now 20

ECE 383 - Embedded **Computer Systems II** Lecture 19 - Soft Core (MicroBlaze) + Custom IP with Interrupt UNITED STATES **AIR FORCE** ACADEMY





MicroBlaze + Custom IP with Interrupt



MicroBlaze + Custom IP with Interrupt

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





What were doing Lab 3





Lab 2 – Architecture



Hints on the Flag Register

REGISTER Reset NS S R 0001 C O hold rest 0 set 0 X undefined 0 O hold 0 1 rold K undefined

SKIP

Process or CSA? Slag <= (flag or Set) and (not Clear) slvireg3 Roll FLAG if flag=1, set=0, clear=0 then clear = 1 then Set = 1 then clear=0; set=0 then set = 1 then clear = 0



How do interrupts work?

Polling versus Interrupts?

5 March 2021 Integrity - Service - Excellence

Interrupts



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

5 March 2021



Lec19.c

```
#include <xil_exception.h>
ul6 isr Count = Ø; // Global Variabk
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);
        Xil Out8(countClearReg, 0x00);
```

// Clear the flag and then you MUST
// allow the flag to be reset later

5 March 2021 Integrity - Service - Excellence



MicroBlaze + Custom IP – Workflow

- Follow Lec19_Install_short_version.pdf
- sion.pdf see last slide!
 - The following slides are for reference
 - The work flow has three main steps.
 - 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
- ***Deviation from Lecture 17 Tutorial***
 - 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

All Automation (2 out of 1 selected)	Description
imicroblaze_0	MicroBlaze connection automation generates local memory of selected size, and caches can be configured. MicroBlaze Debug Module, Peripheral AXI interconnect, Interrupt Controller, a clock source, Processor System Reset are also added and connected as needed. Instance: /microblaze_0 Options Local Memory: 32KB Local Memory ECC: None Cache Control Information: 16KB Debug Module: Debug Only Enabled
	Interrupt Controller: Clock Connection: New Clocking Wizard (100 MHz) 🔹



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.

🖁 🖁 Diagram 🗙 🔣 Address Editor	x 👎 IP Catalog 🛛 🛛				? 🗆 🖻 🤇	×	
Cores Interfaces	Cores Interfaces			Search: Q-			
Name Name User Repository (c:/Users	/Jeffrey.Falkinburg/Documen	AXI4 ts/Courses/ECE383/	Status Li	i cense _2017/ip	VLNV _repo)	•	
Image: Second	 Properties IP Settings Add Repository Refresh All Repositori Customize IP Edit in IP Packager 	Ctrl+E es	Pre-Production In	duded	usafa.edu:Maj_Jeff_Falkinburg	4	
Name: my_counter_ip_v1.0 Version: 1.0 (Rev. 2) Interfaces: AXI4	Disable IP Delete IP License Status Compatible Families Export to Spreadshee	Delete			? _ □ Ľ >	* III + X	

5 March 2021



Edit/Create New IP Package

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

🖁 🖁 Diagram 🗙 🔣 Address Editor	X 👎 IP Catalog 🛛 🛛				? 🗆 🖻 ×
Cores Interfaces	Search: Q-				
Name	'leffrev.Falkinburg/Documen	AXI4	Status L	icense VL	NV no)
AXI Peripheral	Properties	Ctrl+E	Pre-Production Ir	ncluded us	afa.edu:Maj_Jeff_Falkinburg
Alliance Partners	 IP Settings Add Repository Refresh All Repositori Customize IP Edit in IP Packager 	ies			
Details Name: my_counter_ip_v1.0 Version: 1.0 (Rev. 2) Interfaces: AXI4	Disable IP Celete IP License Status Compatible Families	Delete			•
	Export to Spreadshee	et			? _ 🗆 🖻 ×

5 March 2021



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 P button to add our v2.0 of our Lec
 10 Counter IP Core with the exposed roll signal.





Edit/Create New IP Package

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





5 March 2021

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.





5 March 2021



- 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 🗙							
9	Cell	Slave Interface	Base Name	Offset Address	Range	High Address	
	Z □·· I 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	
	men 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	0x000_0000	32K 💌	0x0000_7FFF	
	🛄 🚥 mig_7series_0	S_AXI	memaddr	0x8000_0000	512M	UX9FFF_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.

