

□ GRI postpartum



UNITED STATES
AIR FORCE
ACADEMY

ECE 383 - Embedded Computer Systems II Lecture 17 - Soft CPU - “MicroBlaze”

Lesson Outline

- ~~Today's Agenda~~
- HW# 9 BOC Next Lesson!
- Soft CPU – MicroBlaze
 - MicroBlaze Intro
 - MicroBlaze Tutorial
 - Adding LEDs GPIO

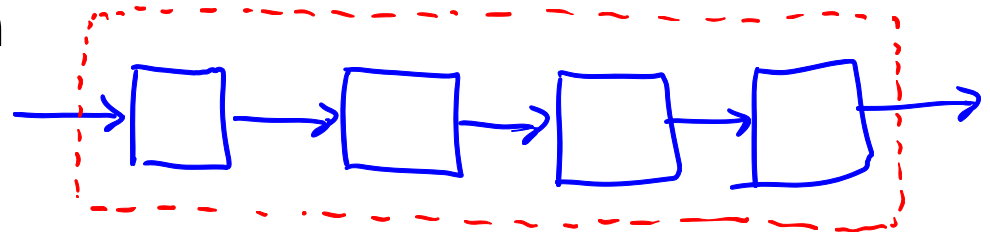
- While I talk, go ahead and do step 1 of
 - [MicroBlaze_Install_short_version.pdf](#)
 - Warning: short path name!
 - Got SDK installed?

Lesson #s off one lesson (due to extra lab2 day)

L17	Lab2 - Data acquisition, storage and display		Lab2 Functionality	COB L17
<u>L18</u>	Soft CPU		Lab2 Write-up HW #9	COB L18 BOC L19
L19	Soft CPU		HW #10	BOC L20
L20	Soft CPU		HW #11	BOC L21
L21	Lab3 - O'scope control	Final Project Ideas		
L22	Lab3 - O'scope control		Gate Check 1	End of L22
L23	Lab3 - O'scope control		Gate Check 2	End of L23
L24	Lab3 - O'scope control		Lab3 Functionality	COB L24
L25	Lab3 - O'scope control		Lab 3 Functionality	COB L25
L26	Direct Digital Synthesis	Final Project Ideas	Lab3 Write-up Final Project Proposal Section 2	COB L26 BOC L27

Debugging?

- Understand the lab
- Babysteps vs Homerun
- Divide-n-Conquer
- Methods



① ■ Code Walk-through

② ■ Testbench

- Small “play space”

- Large lab2

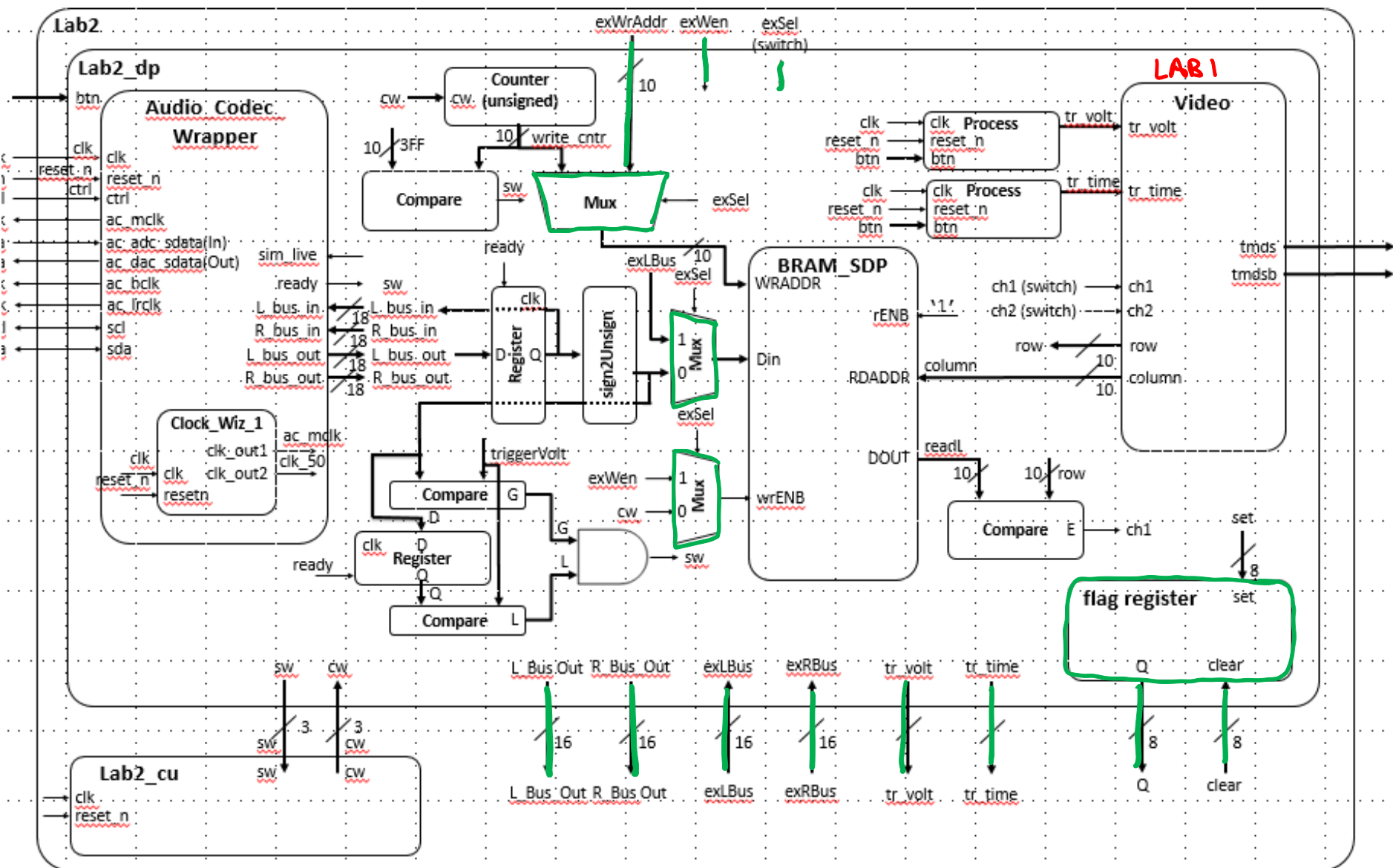
count is 10-bits ¹⁰²³ 0 11 1111 1111
00 0000 0000

sw <= '0' when count < 1024 else '1'

sw is Never '1' ... why?

Lab2 → Lab3

exSel and Flag Register?





UNITED STATES
AIR FORCE
ACADEMY

MicroBlaze Intro

Pay attention:

- **Install microblaze 3 times over the next 3 lessons**
 - (save vs redo?)
- **HW 9, 10, and 11 crucial to lab3**

Not a "Toy" Processor

'pico blaze?

Microblaze Intro

- The goal of today's class is to bring you up to speed on how to instantiate a microBlaze processor on our Artix 7, integrate a custom piece of VHDL code to the processor, and then to write some C code to run on the microBlaze to control the custom VHDL module. Here are some specifications on the microBlaze processor:
- It has thirty-two 32-bit general purpose registers.
- It uses a 32-bit instruction word with three operands, and has two addressing modes.
- It has a 32-bit address bus.
- It uses a single issue (3 or 5)-stage pipeline.

Microblaze Intro

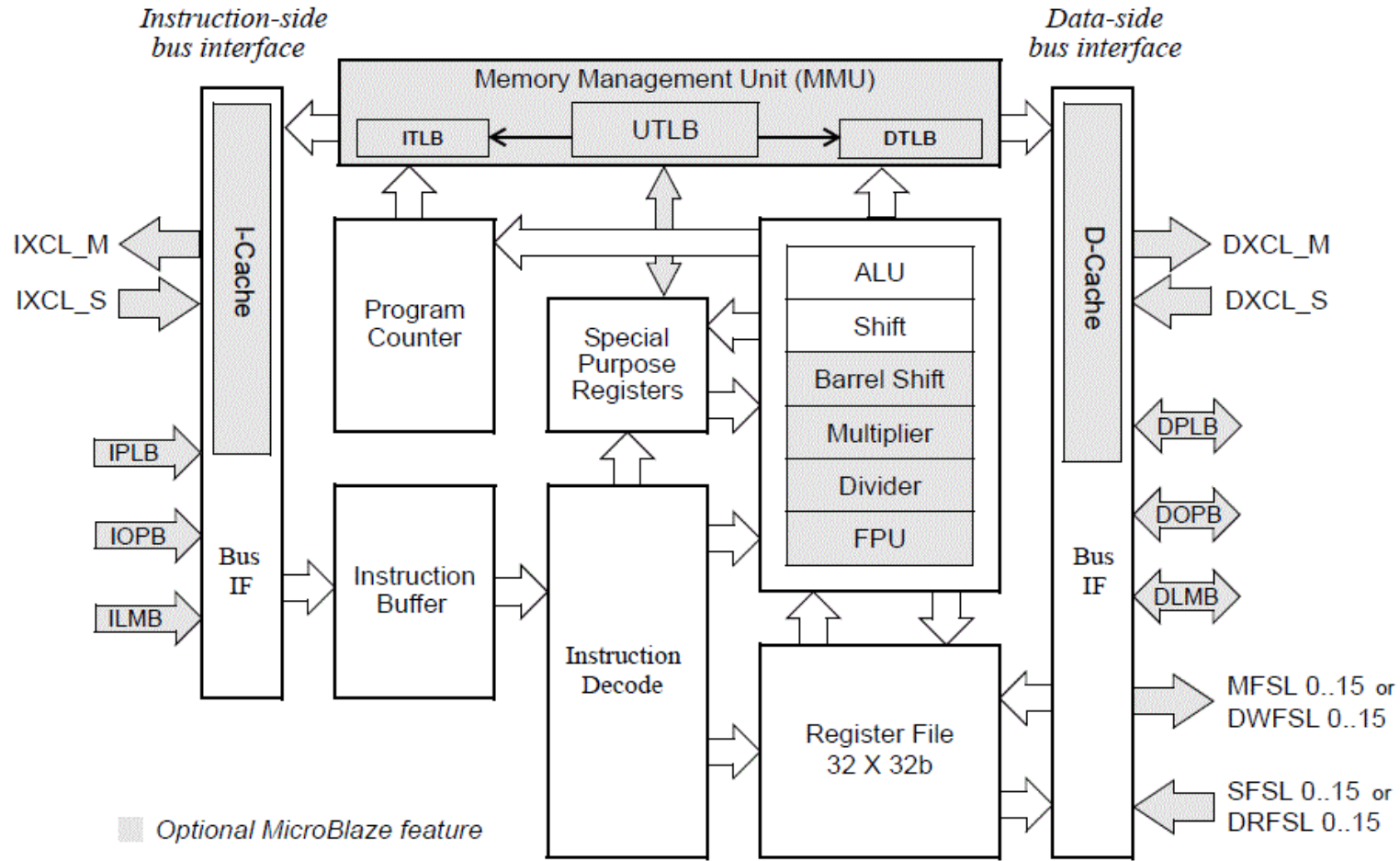


Figure 1-1: MicroBlaze Core Block Diagram



MicroBlaze Tutorial

History:

- [MicroBlaze Tutorial](#) <- Diligent Website
- Nexys_Video_MicroBlaze_Tutorial.pdf ← 46 pages
- MicroBlaze_Install_short_version.pdf ← 3 pages

Lecture:	17
Homework	HW #9
Status	Complete
MicroBlaze Tutorial on Digilent Website	MicroBlaze Tutorial
MicroBlaze Tutorial with ECE 383 Deviations	MicroBlaze Tutorial MicroBlaze Tutorial Short Version
Supplemental Lesson Slides	ECE_383_Lec17.pptx
Code	Lec17.c (Initial Hello World) Lec17_v2.c (Example code to interface with GPIO LEDs)

Microblaze - Tutorial Overview

- In this tutorial, you will be introduced to the tool flow for simple MicroBlaze designs. Specifically, you will create a design that continuously reads the input from **UART** and ~~writes that value to the LEDs~~. The UART will be connected from the FPGA to your computer via a micro USB cable.
- You will follow the tutorial ~~here~~ step by step.

Microblaze - Tutorial Deviations

■ **Deviations** from the Tutorial

- Call your project "L17" or "Lesson17".
- Keep in mind that you can zoom in and out on your block diagram.
- The AXI bus is the bus the Microblaze uses, similar to a PCI bus in normal PC's.
- What does the UART actually do? Although you learned about it in ECE382, you can read more about UART's for a refresher.
- UART Controller?
- Ignore step 6.3 completely. When you do step 6.2, just check the "reset" box in the automatic connections dialogue under clock wizard. You do not need to make the connection manually.
- The Memory Interface Generator (MIG) is used essentially add BRAM (it is SDRAM in this case).
- Although you didn't need to make the first connection manually, you do need to make the 2nd connection manually (with the RAM).

Microblaze - Tutorial

- <https://reference.digilentinc.com/learn/programmable-logic/tutorials/nexys-video-getting-started-with-microblaze/start>
- <https://reference.digilentinc.com/learn/programmable-logic/tutorials/pmod-ips/start>
- [https://reference.digilentinc.com/nexys/nexysvideo/gsm?sb\[0\]=ip&sb\[0\]=integrator](https://reference.digilentinc.com/nexys/nexysvideo/gsm?sb[0]=ip&sb[0]=integrator)
- <https://reference.digilentinc.com/learn/programmable-logic/tutorials/zedboard-creating-custom-ip-cores/start>

Microblaze - Getting Started

- What you need before proceeding with this guide

- Software

- Xilinx Vivado with the **SDK package.**

- Follow this Wiki guide ([Installing Vivado](#)) on how to install and activate Vivado 2018.2

- ✓ **■ Board Support Files**

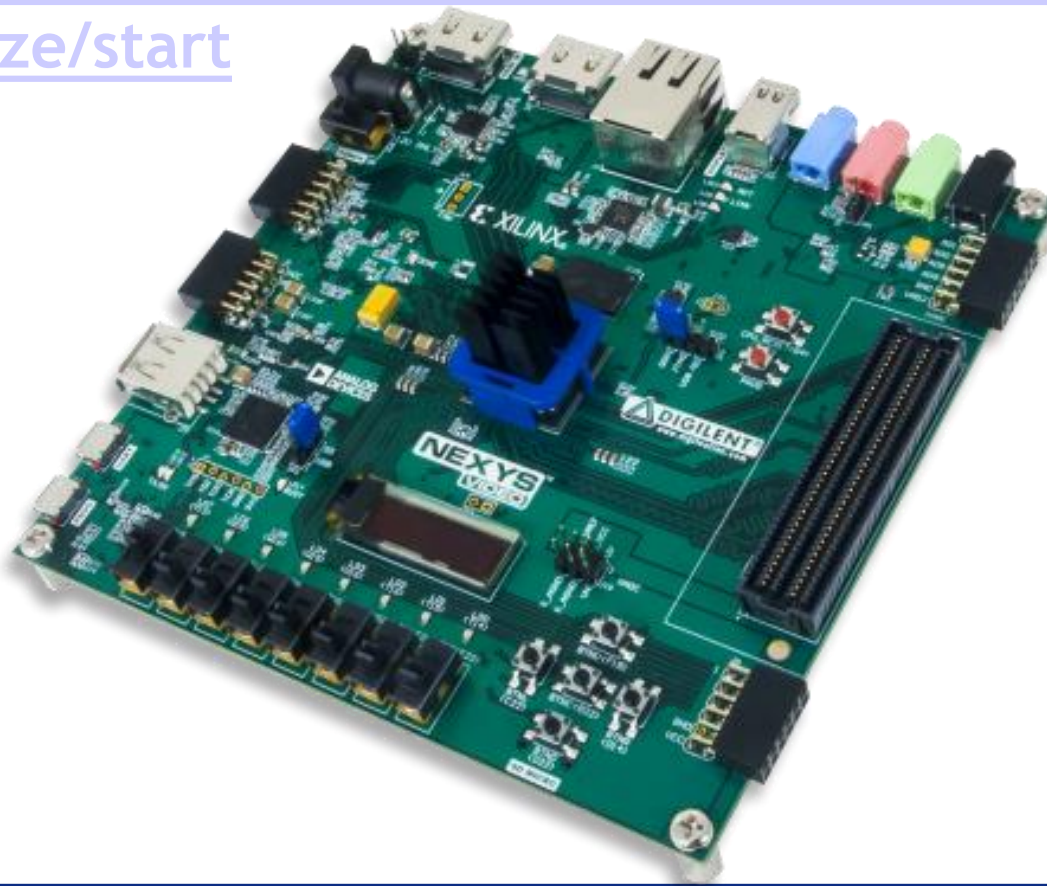
- Board Support Files. These files will describe GPIO interfaces on your board and make it easier to select your FPGA board and add GPIO IP blocks.

- Follow this Wiki guide ([Vivado Board Files for Digilent 7-Series FPGA Boards](#)) on how to install Board Support Files for Vivado 2018.2

- ✓ **■ Hardware**

- Digilent Nexys Video FPGA Board and Micro USB Cable for UART communication and JTAG programming
-

- <https://reference.digilentinc.com/learn/programmable-logic/tutorials/nexys-video-getting-started-with-microblaze/start>



Microblaze - Introduction

remember

clockwiz?

- Microblaze is a soft IP core from Xilinx that will implement a microprocessor entirely within the Xilinx FPGA general purpose memory and logic fabric. For this tutorial, we are going to add a Microblaze IP block using the Vivado IP Integrator tool.
- In addition to the Microblaze IP block, we would also like to make use of the DDR3 SDRAM component on the Nexys Video. Therefore a MIG (Memory Interface Generator) IP block will be added to our design.
- Finally, a UART (Universal Asynchronous Receiver/Transmitter) IP block will be added to communicate between the host PC and the soft processor core running on the Nexys Video.

□ TeraTerm?

■ General Design Flow - Vivado

- Open Vivado and select Nexys Video board
- Create an new Vivado Project
- Create empty block design workspace inside the new project
- Add required IP blocks using the IP integrator tool and build Hardware Design
- Validate and save block design
- Create HDL system wrapper
- Run design Synthesis and Implementation
- Generate Bit File
- Export Hardware Design including the generated bit stream file to SDK tool
- Launch SDK → then do C programming!

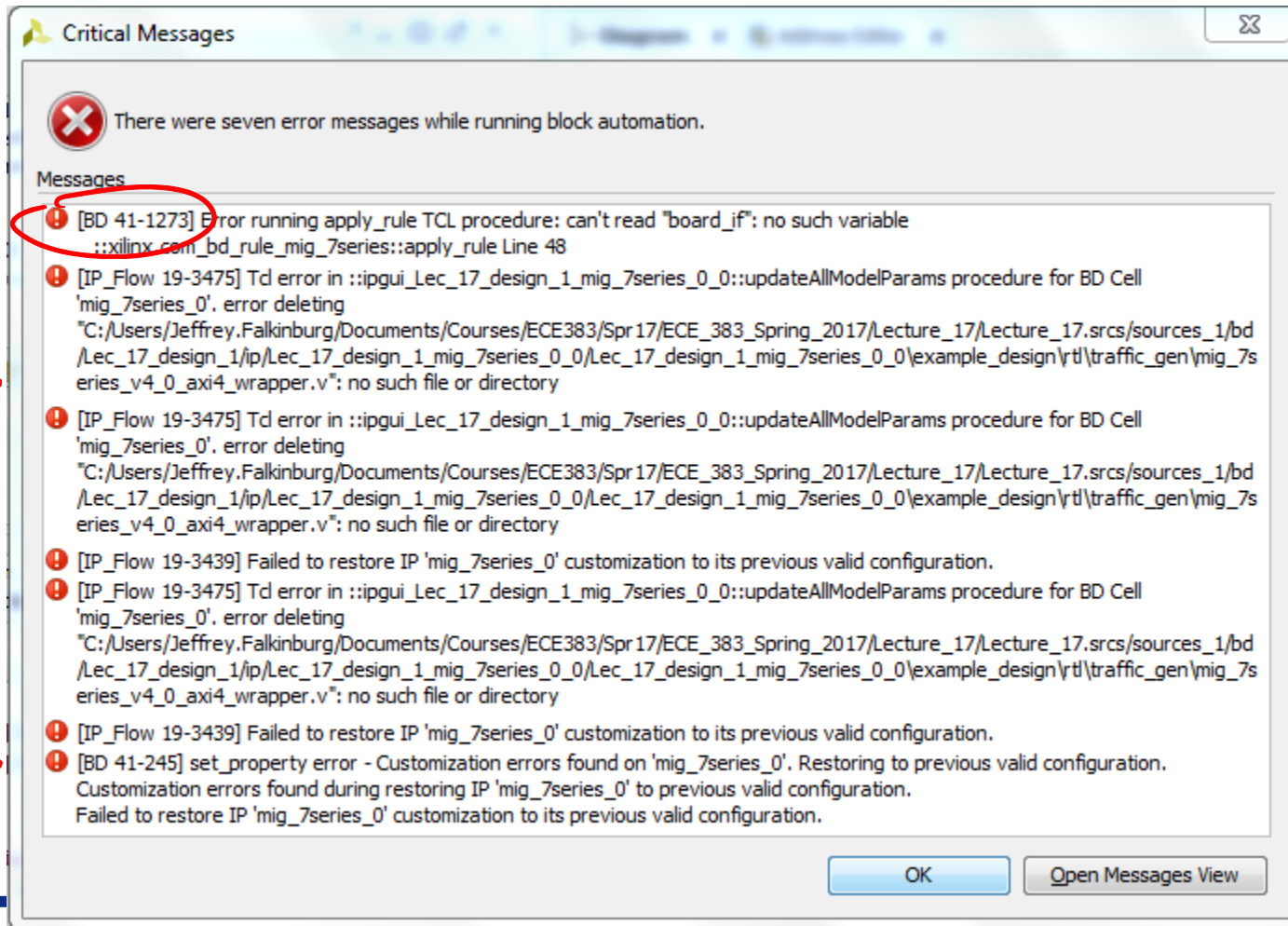
- Now the Hardware design is exported to the SDK tool. The Vivado to SDK hand-off is done internally through Vivado. We will use SDK to create a Software application that will use the customized board interface data and FPGA hardware configuration by importing the hardware design information from Vivado.
- General Design Flow - SDK
 - Create new application project and select default Hello World template
 - Program FPGA
 - Run configuration by selecting the correct UART COM Port and Baud Rate **9600**

Microblaze - Issues

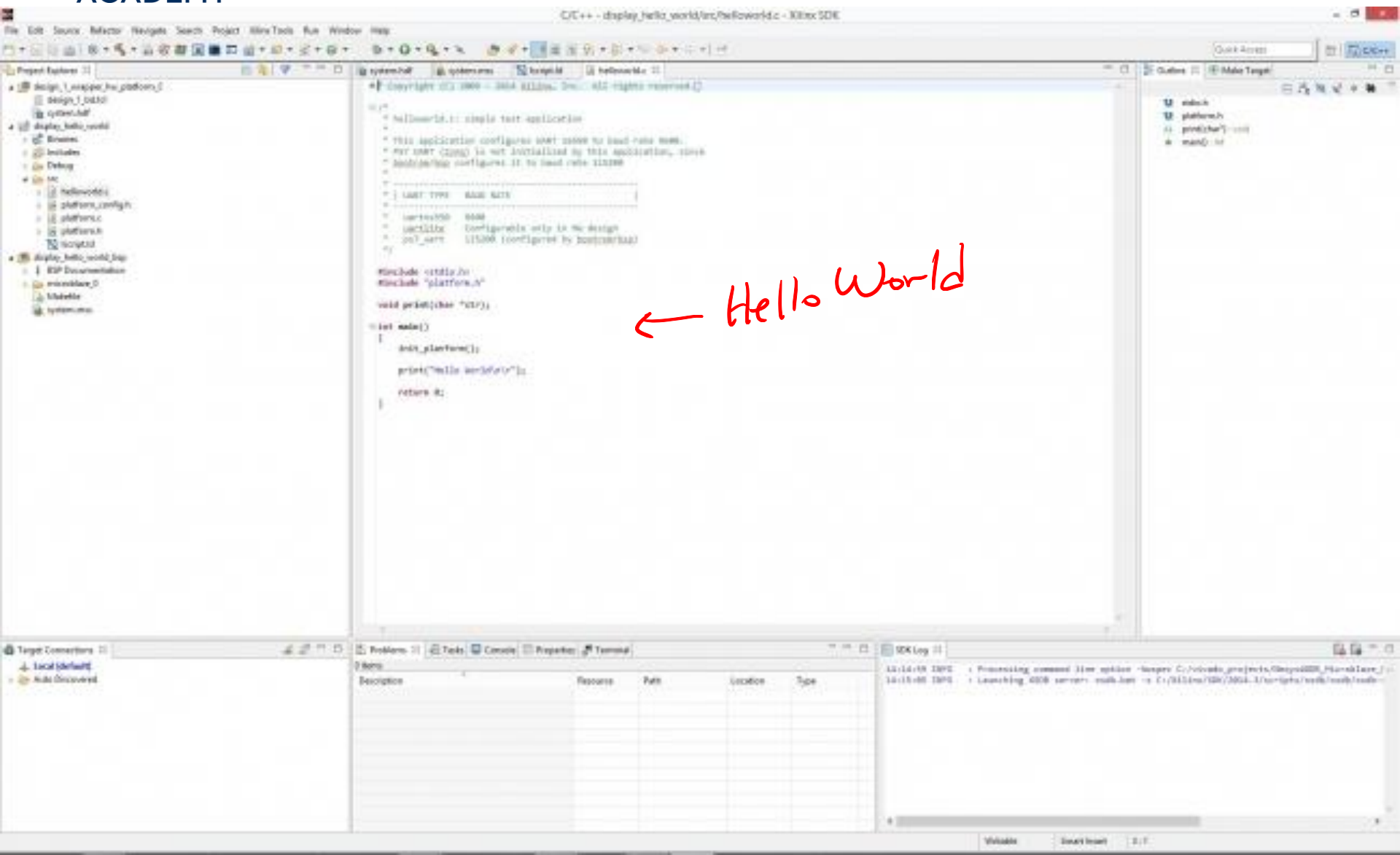
will for HW 9 - 10. Not HW 11 and Lab 3

- Ensure at this time don't use the MicroBlaze Interrupt Controller on the MicroBlaze Block Automation.
- Trouble with too many errors with MIG Block Automation. Should only have the error message [BD 41-1273] ← the "good" error
 - Design won't validate or build with more error messages.
- Block Design name may need to start with "design"

■ Errors after MIG Block Automation Run



Microblaze - Xilinx Vivado **SDK**



Microblaze - Xilinx Vivado SDK

The screenshot displays the Xilinx Vivado SDK IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Xilinx Tools, Run, Window, and Help. The toolbar contains various icons for file operations, editing, and running. The Project Explorer on the left shows the project structure: design_1_wrapper_hw_platform_0, display_hello_world (with subfolders Binaries, Includes, Debug, and src), and display_hello_world_bsp. The main editor window shows the source code for 'helloworld.c', which is a simple test application that configures UART 16550 to a baud rate of 9600 and prints 'Hello World\n\r'. The code includes 'stdio.h' and 'platform.h'. The Outline pane on the right shows the project's header files and the main function. The bottom status bar shows the current target connection as 'Local [default]' and 'Auto Discovered'. The bottom console pane displays the output of the application, showing 'Hello World' and a series of log messages indicating that the processor reset is completed for microblaze_0.

```
/*
 * Copyright (C) 2009 - 2014 Xilinx, Inc. All rights reserved.
 */
/*
 * helloworld.c: simple test application
 *
 * This application configures UART 16550 to baud rate 9600.
 * PS7 UART (Zynq) is not initialized by this application, since
 * bootrom/bsp configures it to baud rate 115200
 *
 * -----
 * | UART TYPE   BAUD RATE |
 * -----
 * uarts550     9600
 * uartlite     Configurable only in HW design
 * ps7_uart     115200 (configured by bootrom/bsp)
 */

#include <stdio.h>
#include "platform.h"

void print(char *str);

int main()
{
    init_platform();

    print("Hello World\n\r");

    return 0;
}
```

16:19:28 INFO : Processor reset is completed for microblaze_0
16:20:39 INFO : Processor reset is completed for microblaze_0
16:21:23 INFO : Processor reset is completed for microblaze_0
16:21:51 INFO : Processor reset is completed for microblaze_0
16:22:25 INFO : Processor reset is completed for microblaze_0
16:24:15 INFO : Processor reset is completed for microblaze_0
16:24:26 INFO : Processor reset is completed for microblaze_0
16:26:19 INFO : Processor reset is completed for microblaze_0
16:26:49 INFO : Processor reset is completed for microblaze_0
16:27:24 INFO : Processor reset is completed for microblaze_0
16:28:03 INFO : FPGA configured successfully with bitstream "C:/vivado_projects/Nexys4DDR_Microblaze_MIG/Hellow_World
16:28:25 INFO : Processor reset is completed for microblaze_0

Microblaze - Xilinx Vivado SDK

The screenshot displays the Xilinx Vivado SDK interface. The main editor shows the source code for `helloworld.c`, which is a simple test application. The code includes comments about UART configuration and baud rates, and defines a `print` function using `stdio.h`. The `main` function calls `init_platform` and `print` to output "Hello World".

Overlaid on the code editor is a terminal window titled "COM4:9600baud Tera Term V...". The terminal displays the output "Hello World", which is circled in red. A red arrow points from the handwritten text "Tera Term" to the terminal window.

The bottom status bar shows the "SDK Log" with the following messages:

```
16:21:51 INFO : Processor reset is completed for microblaze_0
16:22:25 INFO : Processor reset is completed for microblaze_0
16:24:15 INFO : Processor reset is completed for microblaze_0
16:24:26 INFO : Processor reset is completed for microblaze_0
16:26:19 INFO : Processor reset is completed for microblaze_0
16:26:49 INFO : Processor reset is completed for microblaze_0
16:27:24 INFO : Processor reset is completed for microblaze_0
16:28:03 INFO : FPGA configured successfully with bitstream "C:/vivado_projects/Nexys4DDR_Microblaze_MIG/Hellow_World/
16:30:03 INFO : FPGA configured successfully with bitstream "C:/vivado_projects/Nexys4DDR_Microblaze_MIG/Hellow_World/
16:30:14 INFO : Processor reset is completed for microblaze_0
```


Microblaze - Issues with SDK

- **ERROR: Specified device 'Digilent Nexys Video 210276723218B/1-xc7a200t' is not found on the board**
 - This essentially means that the .bit file you created was for a different board and the Artix 7 is rejecting it.
 - Solution: Regenerate .bit file in Vivado and re-export to SDK



Adding LEDs GPIO

(was part of HW09)

{ SKIP for this
year

What is HW09?

Microblaze - Adding LEDs GPIO

Lecture_17 - [C:/Users/Jeffrey.Falkinburg/Documents/Courses/ECE383/Spr17/ECE_383_Spring_2017/Lecture_17/Lecture_17.xpr] - Vivado 2016.4

File Edit Flow Tools Window Layout View Help

Quick Access

write_bitstream Complete

Flow Navigator

- IP Catalog
- IP Integrator**
 - Create Block Design
 - Open Block Design
 - Generate Block Design
- Simulation
 - Simulation Settings
 - Run Simulation
- RTL Analysis
 - Elaboration Settings
 - Open Elaborated Design
- Synthesis
 - Synthesis Settings
 - Run Synthesis
 - Open Synthesized Design
- Implementation
 - Implementation Settings
 - Run Implementation
 - Open Implemented Design
- Program and Debug
 - Bitstream Settings
 - Generate Bitstream
 - Open Hardware Manager

Block Design - design_1

Board

- DDR3 SDRAM
- Onboard Micro SD Slot
- QSPI Flash
- GPIO (0 out of 4 connected)
- 5 Push Buttons
- 8 LEDs
- 8 Sw
- Onb
- HDMI (0)

Board Component Properties... Ctrl+E

Connect Board Component...

Auto Connect

Board Component Properties

8 LEDs

Description: LEDs 7 to 0

General Properties

Diagram

design_1

MicroBlaze

Tcl Console

```
Run output will be captured here: C:/Users/Jeffrey.Falkinburg/Documents/Courses/ECE383/Spr17/ECE_383_Spring_2017/Lecture_17/Lecture_17.runs/impl_1/runme
launch_runs: Time (s): cpu = 00:00:19 ; elapsed = 00:08:33 . Memory (MB): peak = 1380.512 ; gain = 47.523
file mkdir C:/Users/Jeffrey.Falkinburg/Documents/Courses/ECE383/Spr17/ECE_383_Spring_2017/Lecture_17/sdk
file copy -force C:/Users/Jeffrey.Falkinburg/Documents/Courses/ECE383/Spr17/ECE_383_Spring_2017/Lecture_17/Lecture_17.runs/impl_1/design_1_wrapper.sysd
launch_sdk -workspace C:/Users/Jeffrey.Falkinburg/Documents/Courses/ECE383/Spr17/ECE_383_Spring_2017/Lecture_17/Lecture_17.sdk -hwspec C:/Users/Jeffrey.
INFO: [Vivado 12-393] Launching SDK...
INFO: [Vivado 12-417] Running xsdk -workspace C:/Users/Jeffrey.Falkinburg/Documents/Courses/ECE383/Spr17/ECE_383_Spring_2017/Lecture_17/Lecture_17.sdk -
INFO: [Vivado 12-3157] SDK launch initiated. Please check console for any further messages.
```

Type a Tcl command here

Tcl Console Messages Log Reports Design Runs

Connect Board Component

Windows Taskbar: 11:00 PM 1/1/2017

Microblaze - Adding LEDs GPIO

Lecture_17 - [C:/Users/Jeffrey.Falkenburg/Documents/Courses/ECE383/Spr17/ECE_383_Spring_2017/Lecture_17/Lecture_17.xpr] - Vivado 2016.4

File Edit Flow Tools Window Layout View Help

Quick Access

write_bitstream Complete

Flow Navigator

- IP Catalog
- IP Integrator**
 - Create Block Design
 - Open Block Design
 - Generate Block Design
- Simulation
 - Simulation Settings
 - Run Simulation
- RTL Analysis
 - Elaboration Settings
 - Open Elaborated Design
- Synthesis
 - Synthesis Settings
 - Run Synthesis
 - Open Synthesized Design
- Implementation
 - Implementation Settings
 - Run Implementation
 - Open Implemented Design
- Program and Debug
 - Bitstream Settings
 - Generate Bitstream
 - Open Hardware Manager

Block Design - design_1

Board

- DDR3 SDRAM
- Onboard Micro SD Slot
- QSPI Flash
- GPIO (0 out of 4 connected)
- 5 Push Buttons
- 3 LEDs
- 8 Switches
- Onboard OLED
- HDMI (0 out of 4 connected)

Sources Design Signal

Board Component Properties

8 LEDs

Description: LEDs 7 to 0

General Properties

Tcl Console

```
Run output will be ca
launch_runs: Time (s)
file mkdir C:/Users/J
file copy -force C:/U

launch_sdk -workspace C:/Users/Jeffrey.Falkenburg/Documents/Courses/ECE383/Spr17/ECE_383_Spring_2017/Lecture_17/Lecture_17.sdk -hwspec C:/Users/Jeffrey.
INFO: [Vivado 12-393] Launching SDK...
INFO: [Vivado 12-417] Running xsdk -workspace C:/Users/Jeffrey.Falkenburg/Documents/Courses/ECE383/Spr17/ECE_383_Spring_2017/Lecture_17/Lecture_17.sdk -
INFO: [Vivado 12-3157] SDK launch initiated. Please check console for any further messages.
```

Type a Tcl command here

Connect Board Component

Messages Log Reports Design Runs

Connect Board Component

Select an IP block interface for connecting board component '8 LEDs'.

Name	VLNV
Create new IP	
AXI GPIO	xilinx.com:ip:axi_gpio:2.0
GPIO	
GPIO2	
IOModule	xilinx.com:ip:iomodule:3.0
GPIO1	
GPIO2	
GPIO3	
GPIO4	
MicroBlaze MCS	xilinx.com:ip:microblaze_mcs:3.0
GPIO1	
GPIO2	

OK Cancel

MicroBlaze

383_Spring_2017/Lecture_17/Lecture_17.runs/impl_1/runme
= 47.523
e_17/Lecture_17.sdk
Lecture_17/Lecture_17.runs/impl_1/design_1_wrapper.sysde

11:02 PM
1/1/2017

Microblaze - Adding LEDs GPIO

- Run Connection Automation

