Name: ____________________ ID: ____________ Start Date: Tuesday, Aug. 10, 2021
Due Date: Wednesday, Aug. 11, 2021

Software and Hardware Co-Design with Zybo, Summer 2021 HUST
Lab #1 Xilinx Tool Flow, Pins Assignment, UART on Zybo PL

1 Objectives

(1) *Implement the UART circuit from Xilinx on the PL fabric of Zybo only.*

(2) *Connect a SparkFun 3.3V FTDI Basic Breakout to SelectIO pins on a Pmod*

(3) *Demonstrate your UART transmit and receive on a HyperTerminal of your laptop.*

(4) *This is an individual lab. Each student will need to demonstrate his or her own work to get credit for this lab.*

This lab is an individual, take-home lab to be performed on your own time. The main ideas and instructions for this lab are from Labs #1 Xilinx Tool Flow Lab for Nexys 3 and Lab #2 Architecture Wizard and Pins Assignment Lab for Nexys 3 of Xilinx University Program at https://www.xilinx.com/support/university.html.

2 Deliverables

(1) Submit your report in pdf format. Name your report to be lab1yourlastname in pdf.

(2) Your report should be of memo style that includes your name, date, subject title, a summary and the following data and brief description of each table or figure.

- *Your own simulation waveforms similar to Figure 11, Figure 13, and Figure 14 from "Xilinx Tool Flow Lab Workbook"..*
- *Your own design summary similar to Table 17 from "Xilinx Tool Flow Lab Workbook".*
- *A few screenshots of your HyperTerminal that displays functions of the real-time clock. You could use snipping tool of Word 10 to obtain screen images.*

3 Part 1: Xilinx Tool Flow

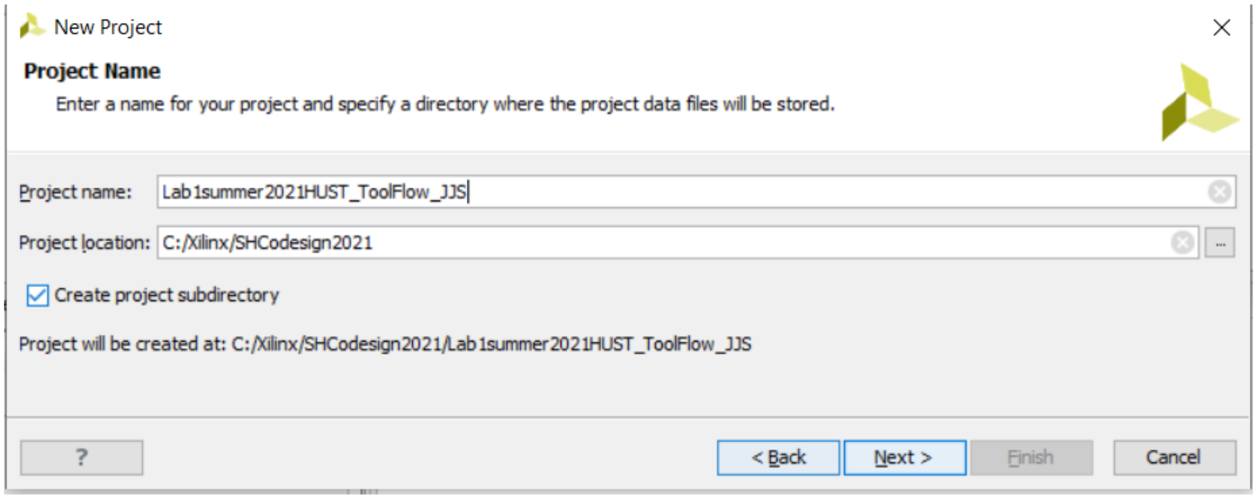
This part is based on a lab handout "Xilinx Tool Flow Lab Workbook" from Xilinx University [1]. A revised version of the workbook is available as "Part 1 of Lab 1 Xilinx Tool Flow". The four files are available as lab1part1sources.zip. The file names are given below.

testbench.v   INT_TEST.V   kcpsm3.v   kcpsm3_int_test.v

File kcpsm3.v is an 8-bit microcontroller based on PicoBlaze. File kcpsm3_int_test.v is a circuit to test interrupt services of this 8-bit microcontroller. File INT_TEST.v contains the instructions in binary for the PicoBlaze microcontroller. File testbench.v is a test bench to simulate this whole microcontroller circuit.

3.1 Create, Simulate and Implement kcpsm3_int_test.v Circuit

Create a Vivado RLT project for Zybo board under your SHCodesign2021 folder as C:/Xilinx/SHCodesign2021. Name your project Lab1summer2021HUST_ToolFlow_JJS, where JJS is your name initials. Add three Verilog source files to Design Sources folder and add the test bench to Simulation Sources folder of your project. An example is shown below.



New Project

Project Name

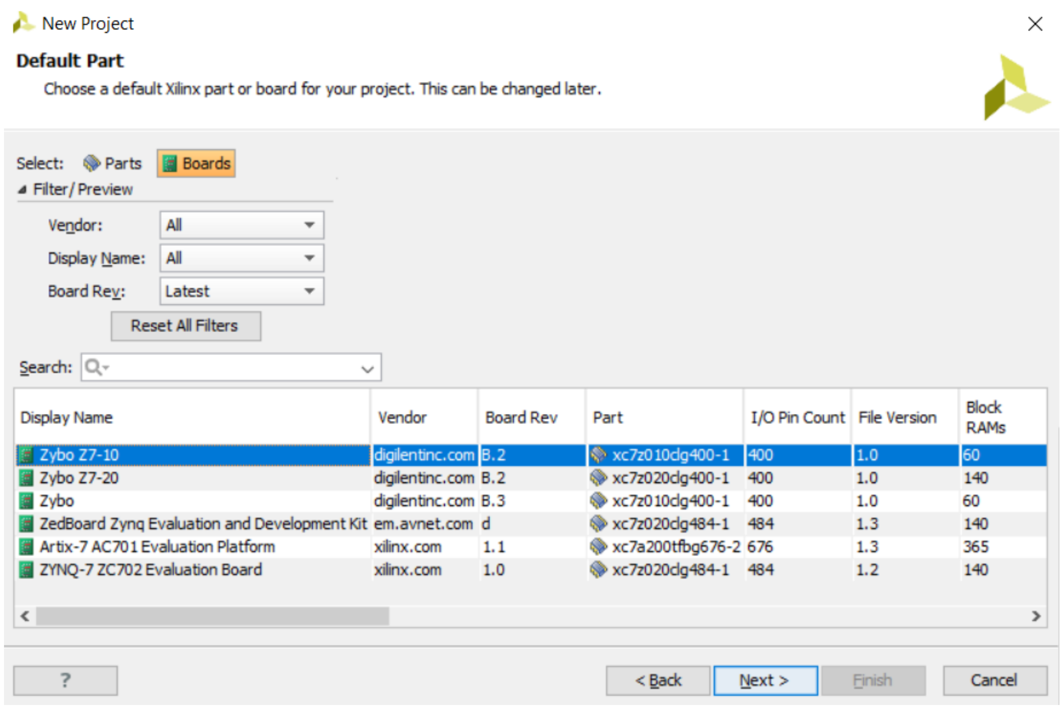
Enter a name for your project and specify a directory where the project data files will be stored.

Project name: Lab1summer2021HUST_ToolFlow_JJS

Project location: C:/Xilinx/SHCodesign2021

☑ Create project subdirectory

Project will be created at: C:/Xilinx/SHCodesign2021/Lab1summer2021HUST_ToolFlow_JJS



New Project

Default Part

Choose a default Xilinx part or board for your project. This can be changed later.

Select: Parts [✓ Boards]

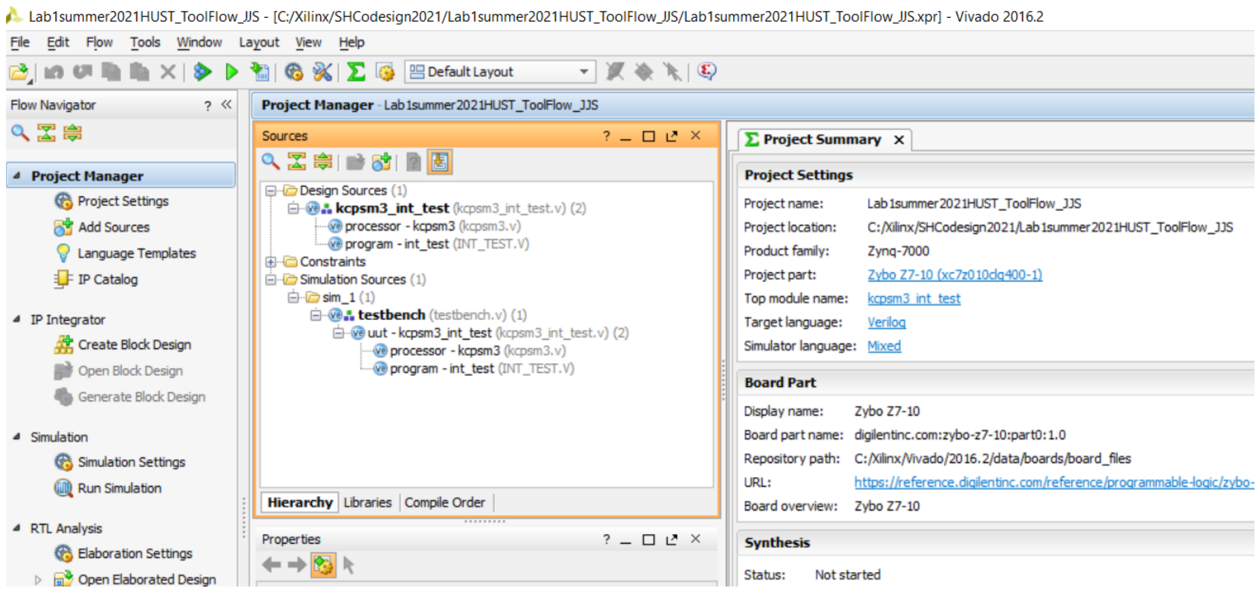
Filter/ Preview

Vendor: All

Display Name: All

Board Rev: Latest

Display Name	Vendor	Board Rev	Part	I/O Pin Count	File Version	Block RAMs
Zybo Z7-10	digilentinc.com	B.2	xc7z010clg400-1	400	1.0	60
Zybo Z7-20	digilentinc.com	B.2	xc7z020clg400-1	400	1.0	140
Zybo	digilentinc.com	B.3	xc7z010clg400-1	400	1.0	60
ZedBoard Zynq Evaluation and Development Kit	em.avnet.com	d	xc7z020clg484-1	484	1.3	140
Artix-7 AC701 Evaluation Platform	xilinx.com	1.1	xc7a200tfbg676-2	676	1.3	365
ZYNQ-7 ZC702 Evaluation Board	xilinx.com	1.0	xc7z020clg484-1	484	1.2	140



Lab1summer2021HUST_ToolFlow_JJS - [C:/Xilinx/SHCodesign2021/Lab1summer2021HUST_ToolFlow_JJS/Lab1summer2021HUST_ToolFlow_JJS.xpr] - Vivado 2016.2

Project Manager - Lab1summer2021HUST_ToolFlow_JJS

```
Design Sources (1)
  kcpsm3_int_test (kcpsm3_int_test.v) (2)
    processor - kcpsm3 (kcpsm3.v)
    program - int_test (INT_TEST.V)
Constraints
Simulation Sources (1)
  sim_1 (1)
    testbench (testbench.v) (1)
      uut - kcpsm3_int_test (kcpsm3_int_test.v) (2)
        processor - kcpsm3 (kcpsm3.v)
        program - int_test (INT_TEST.V)
```

Project Settings

Project name:	Lab1summer2021HUST_ToolFlow_JJS
Project location:	C:/Xilinx/SHCodesign2021/Lab1summer2021HUST_ToolFlow_JJS
Product family:	Zynq-7000
Project part:	Zybo Z7-10 (xc7z010clg400-1)
Top module name:	kcpsm3_int_test
Target language:	Verilog
Simulator language:	Mixed

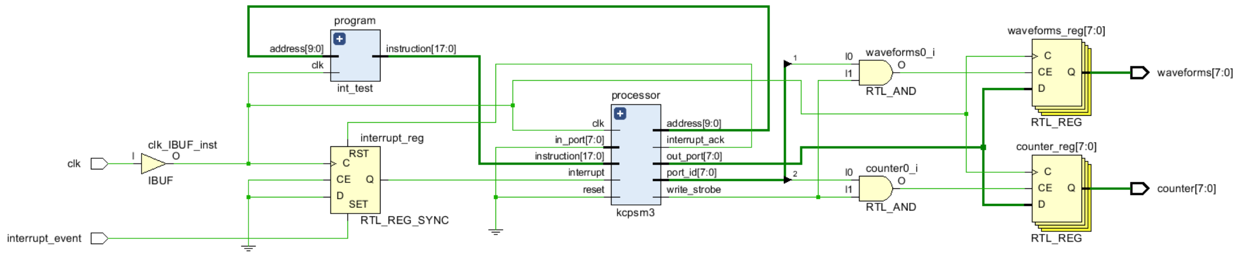
Board Part

Display name:	Zybo Z7-10
Board part name:	digilentinc.com:zybo-z7-10:part0:1.0
Repository path:	C:/Xilinx/Vivado/2016.2/data/boards/board_files
URL:	https://reference.digilentinc.com/reference/programmable-logic/zybo-
Board overview:	Zybo Z7-10

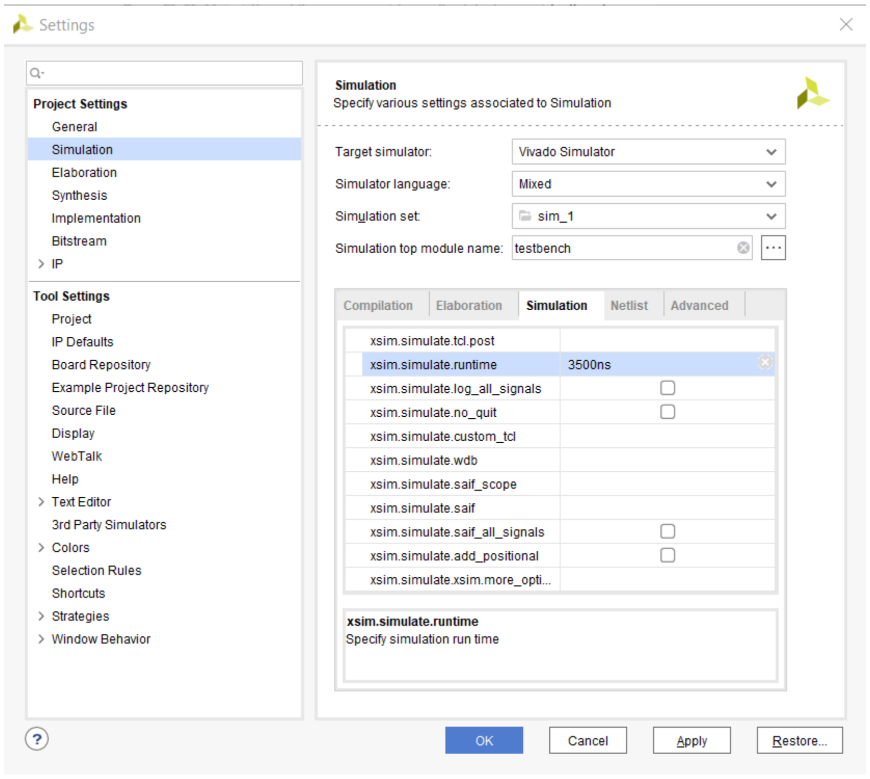
Synthesis

Status:	Not started

Select file kcpsm3_int_test.v as the top-level circuit. Choose "Open Elaborated Design" to see the top-level circuit schematic of this file as follows.



File testbench.v specifies the total simulation time to be 150*PERIOD=3000 ns, where PERIOD is 20ns. The clock period is 20ns. The total number of clock cycles is 150. The default maximum simulation time is 1,000 ns. To change this maximum limit, select Tools->Settings to open Settings Menu. Select Simulation under Project Settings column and Simulation under Simulation options. As you can see below that xsim.simulate.runtime parameter has been changed to 3,500ns. Click "Apply" and "OK".


Settings

Simulation
Specify various settings associated to Simulation

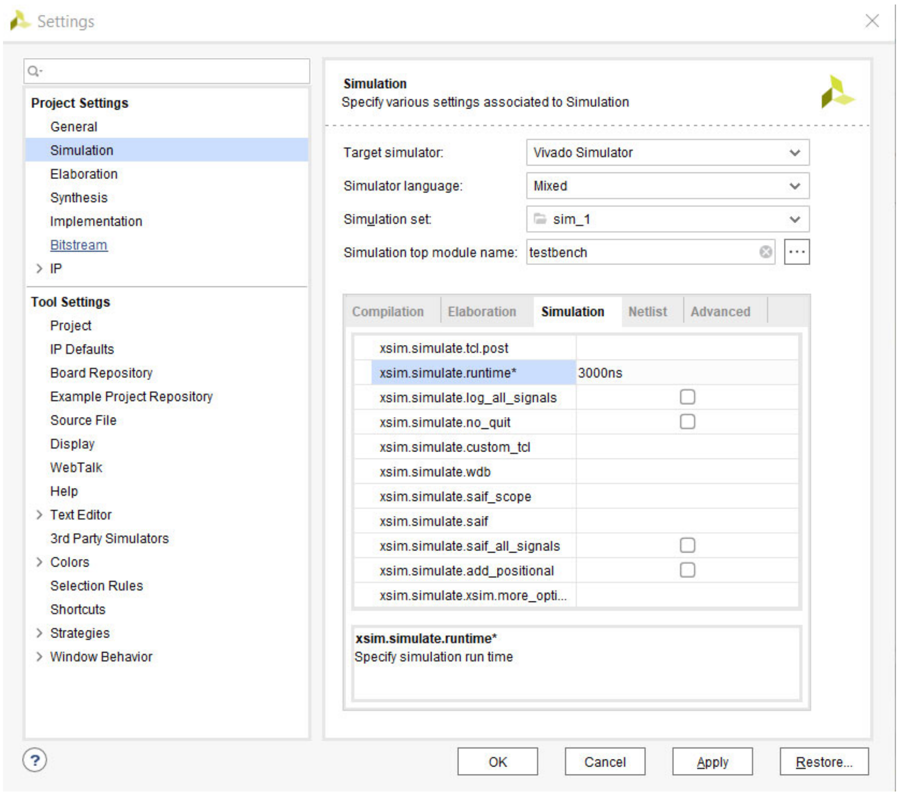
Target simulator: Vivado Simulator
Simulator language: Mixed
Simulation set: sim_1
Simulation top module name: testbench

xsim.simulate.tcl.post	
xsim.simulate.runtime	3500ns
xsim.simulate.log_all_signals	☐
xsim.simulate.no_quit	☐
xsim.simulate.custom_tcl	
xsim.simulate.wdb	
xsim.simulate.saif_scope	
xsim.simulate.saif	
xsim.simulate.saif_all_signals	☐
xsim.simulate.add_positional	☐
xsim.simulate.xsim.more_opti...	

xsim.simulate.runtime
Specify simulation run time

3.2 Simulate Your Circuit

After testbench.v is added to Simulation Sources folder, you can see file kcpsm3_int_test.v is under this test bench. Right click on Run Simulation to open Simulation Settings. Choose Simulation column and change xsim.simulate.runtime to 3000ns.


Settings

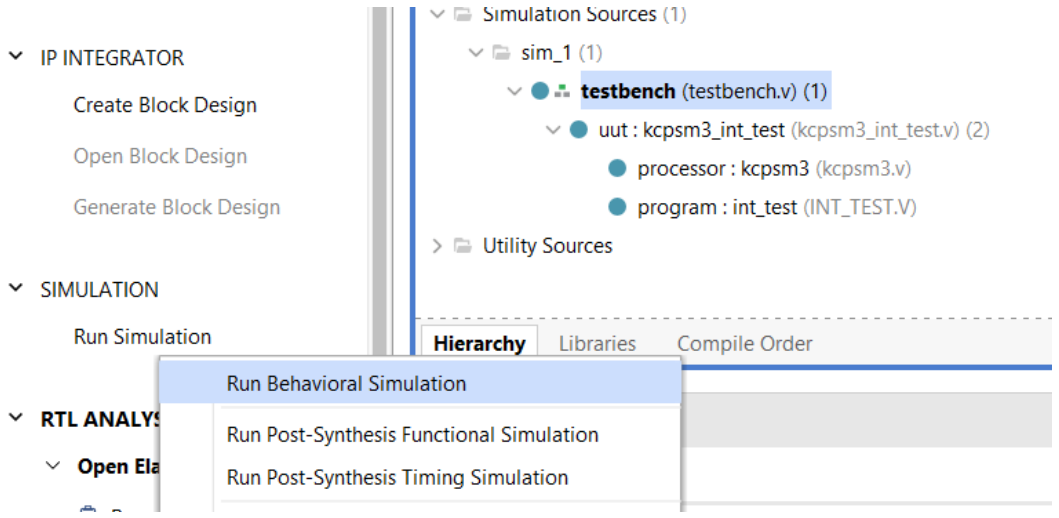
Simulation
Specify various settings associated to Simulation

Target simulator: Vivado Simulator
Simulator language: Mixed
Simulation set: sim_1
Simulation top module name: testbench

xsim.simulate.tcl.post	
xsim.simulate.runtime*	3000ns
xsim.simulate.log_all_signals	☐
xsim.simulate.no_quit	☐
xsim.simulate.custom_tcl	
xsim.simulate.wdb	
xsim.simulate.saif_scope	
xsim.simulate.saif	
xsim.simulate.saif_all_signals	☐
xsim.simulate.add_positional	☐
xsim.simulate.xsim.more_opti...	

xsim.simulate.runtime*
Specify simulation run time

Select testbench.v file under Project Manager and click “Run Simulation->Run Behavioral Simulation” to generate simulation results from testbench.v file. If your Vivado has another simuation opened, you may need to click File->Close Simulation first.


IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYS
Open Ela
Simulation Sources (1)
sim_1 (1)
testbench (testbench.v) (1)
uut : kcpsm3_int_test (kcpsm3_int_test.v) (2)
processor : kcpsm3 (kcpsm3.v)
program : int_test (INT_TEST.V)
Utility Sources
Hierarchy
Libraries
Compile Order
Run Behavioral Simulation
Run Post-Synthesis Functional Simulation
Run Post-Synthesis Timing Simulation

Click on Untitle1 in the following case to see the resulting simulation waveforms. You may need to enlarge this display area and click Zoon Fit button to show the complete simulation waveforms at once.

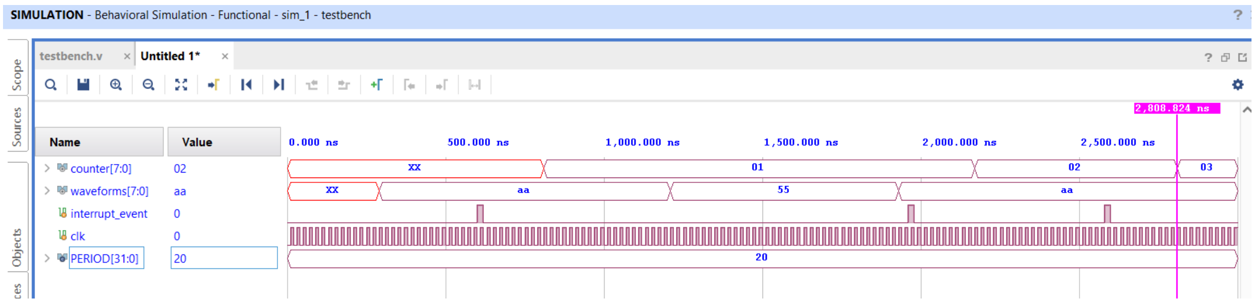


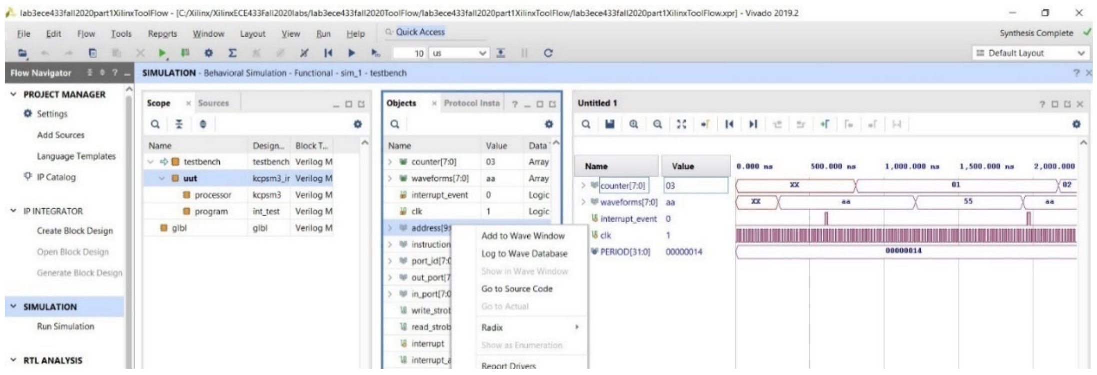
Figure 11. Behaviour Simulation Results [1] runtime=3000ns from Lab #1 of Xilinx

3.3 Analyze the Internal Signals of a Circuit over Scope

The steps below are for illustrative purposes only and show how to analyze the internal signals of the design. The first step shows how to add internal signals to the waveform. The second step shows how to analyze the interrupt process. The third step shows how to analyze the output waveform process. You may optionally complete these steps if you have additional time at the end of the lab.

3.3.1 Monitor internal signals by adding them to the design.

Choose Scope instead of Sources under Simulation. You need to select the desired module entry [uut] in **Scope** window, and then select the desired signal [address] in Processes window. Right-click on it and select **Add to Wave Window**. Similarly add interrupt, interrupt_ack, and instruction signals to the waveform display.

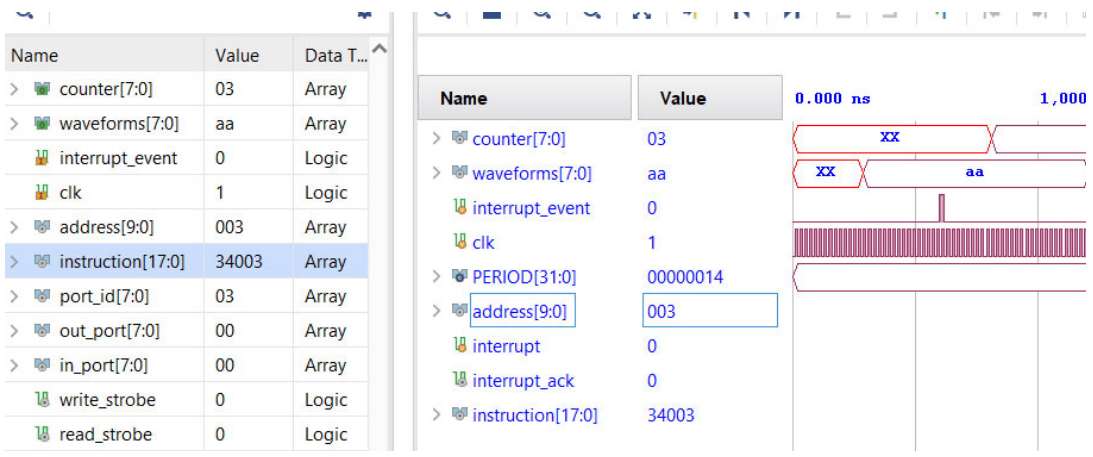


lab3ece433fall2020part1XilinxToolFlow - [C:/Xilinx/XilinxECE433Fall2020labs/lab3ece433fall2020ToolFlow/lab3ece433fall2020part1XilinxToolFlow/lab3ece433fall2020part1XilinxToolFlow.xpr] - Vivado 2019.2

SIMULATION - Behavioral Simulation - Functional - sim_1 - testbench

Name	Design...	Block T...
testbench	testbench	Verilog M
uut	kcpsm3_ir	Verilog M
processor	kcpsm3	Verilog M
program	int_test	Verilog M
glbl	glbl	Verilog M

Name	Value
counter[7:0]	03
waveforms[7:0]	aa
interrupt_event	0
clk	1
PERIOD[31:0]	00000014

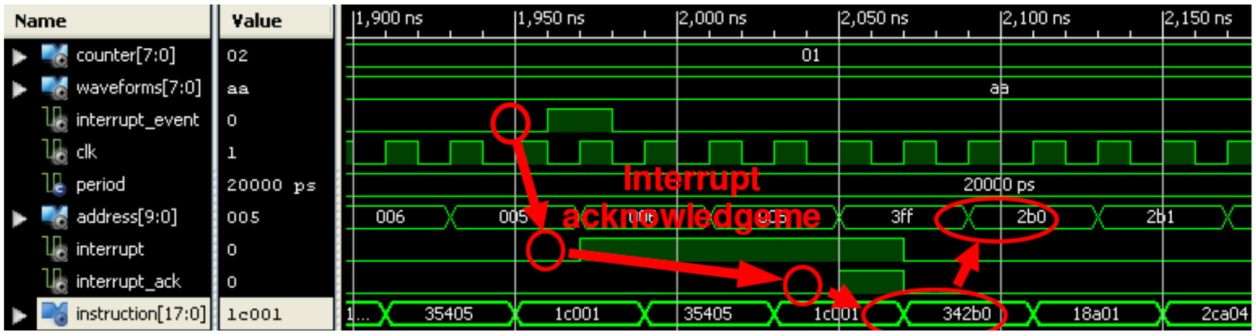


Name	Value	Data T...
counter[7:0]	03	Array
waveforms[7:0]	aa	Array
interrupt_event	0	Logic
clk	1	Logic
address[9:0]	003	Array
instruction[17:0]	34003	Array
port_id[7:0]	03	Array
out_port[7:0]	00	Array
in_port[7:0]	00	Array
write_strobe	0	Logic
read_strobe	0	Logic

Name	Value
counter[7:0]	03
waveforms[7:0]	aa
interrupt_event	0
clk	1
PERIOD[31:0]	00000014
address[9:0]	003
interrupt	0
interrupt_ack	0
instruction[17:0]	34003

3.3.2 Change radix of address and instructions to hexadecimal and Re-Simulate

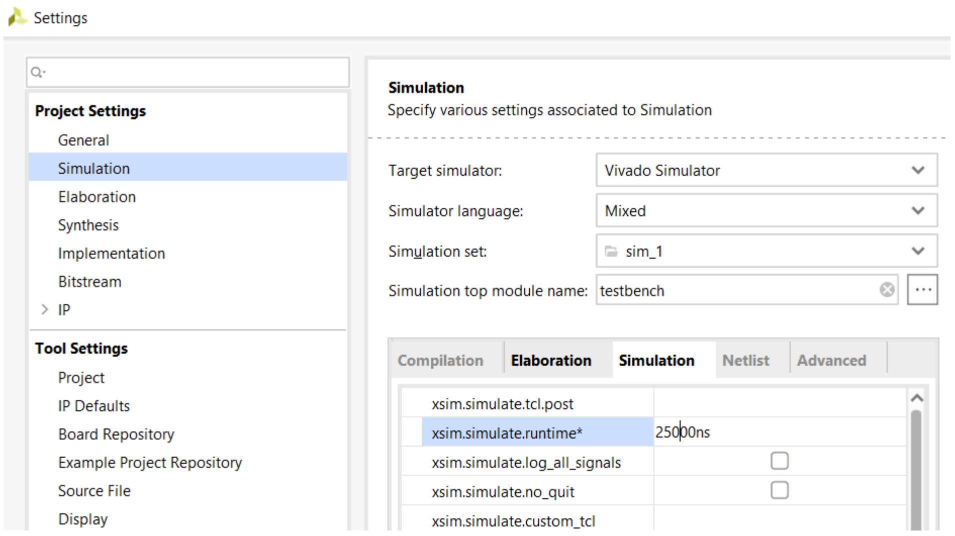
Click the settings wheel on the top right of the waveform window to check if radix is hexadecimal**.** Re-launch simulation to see the added signals, click on the recycle icon shown in the circle below. Zoom into the waveform to see interrupt Service Routine process by looking for an interrupt event that occurred three times and their responses, i.e., Acknowledgement as follows.



Name	Value
counter[7:0]	02
waveforms[7:0]	aa
interrupt_event	0
clk	1
period	20000 ps
address[9:0]	005
interrupt	0
interrupt_ack	0
instruction[17:0]	1c001

Figure 13. Interrupt Service Routine from Lab #1 of Xilinx.

Change simulation run time to 25000ns, add write_strobe signal and re-simulate the design. Analyze the output waveform process (**Figure 14**).



Settings

Project Settings
- General
- Simulation
- Elaboration
- Synthesis
- Implementation
- Bitstream
- IP

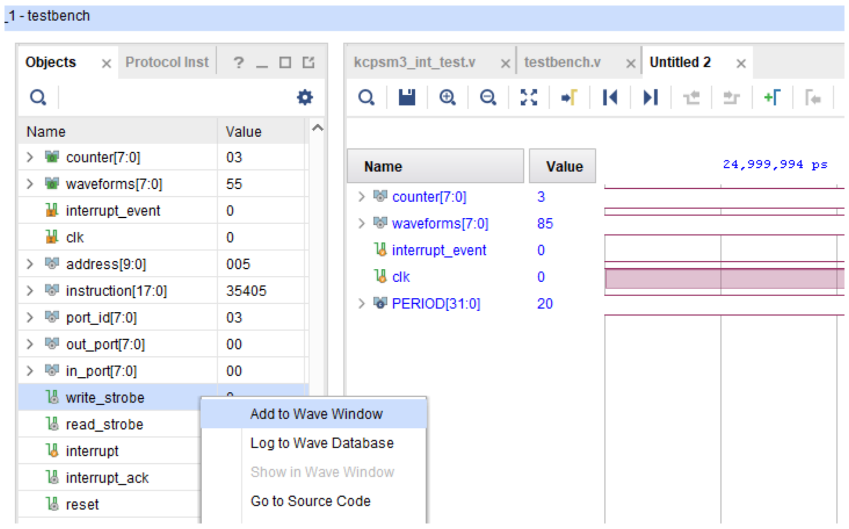
Tool Settings
- Project
- IP Defaults
- Board Repository
- Example Project Repository
- Source File
- Display

Simulation
Specify various settings associated to Simulation

Target simulator:	Vivado Simulator
Simulator language:	Mixed
Simulation set:	sim_1
Simulation top module name:	testbench

Compilation | Elaboration | Simulation | Netlist | Advanced

xsim.simulate.tcl.post	
xsim.simulate.runtime*	25000ns
xsim.simulate.log_all_signals	☐
xsim.simulate.no_quit	☐
xsim.simulate.custom_tcl	



_1 - testbench

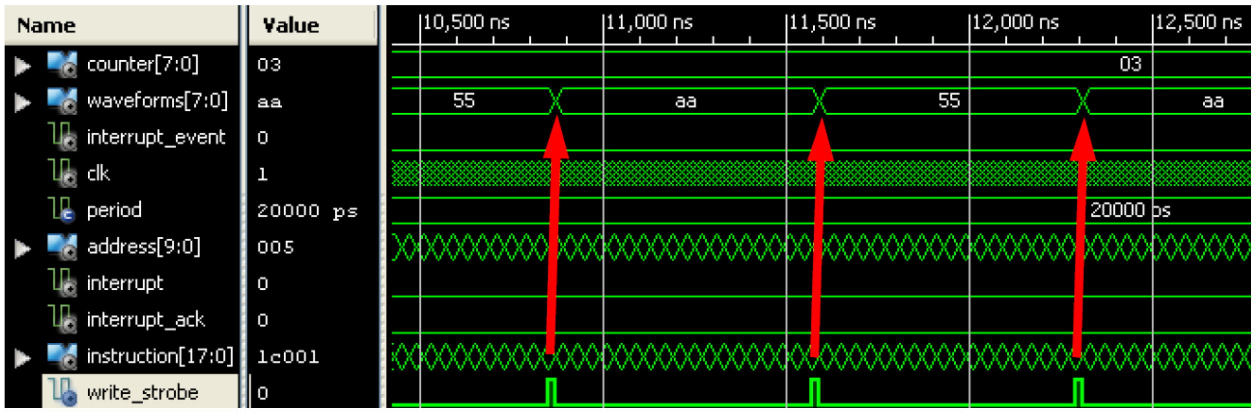
Objects | Protocol Inst

Name	Value
counter[7:0]	03
waveforms[7:0]	55
interrupt_event	0
clk	0
address[9:0]	005
instruction[17:0]	35405
port_id[7:0]	03
out_port[7:0]	00
in_port[7:0]	00
write_strobe	0
read_strobe	
interrupt	
interrupt_ack	
reset	

kcpsm3_int_test.v | testbench.v | Untitled 2

Name	Value
counter[7:0]	3
waveforms[7:0]	85
interrupt_event	0
clk	0
PERIOD[31:0]	20

Analyze the output waveform process in **Figure 14** to see how write_strobe signal works.



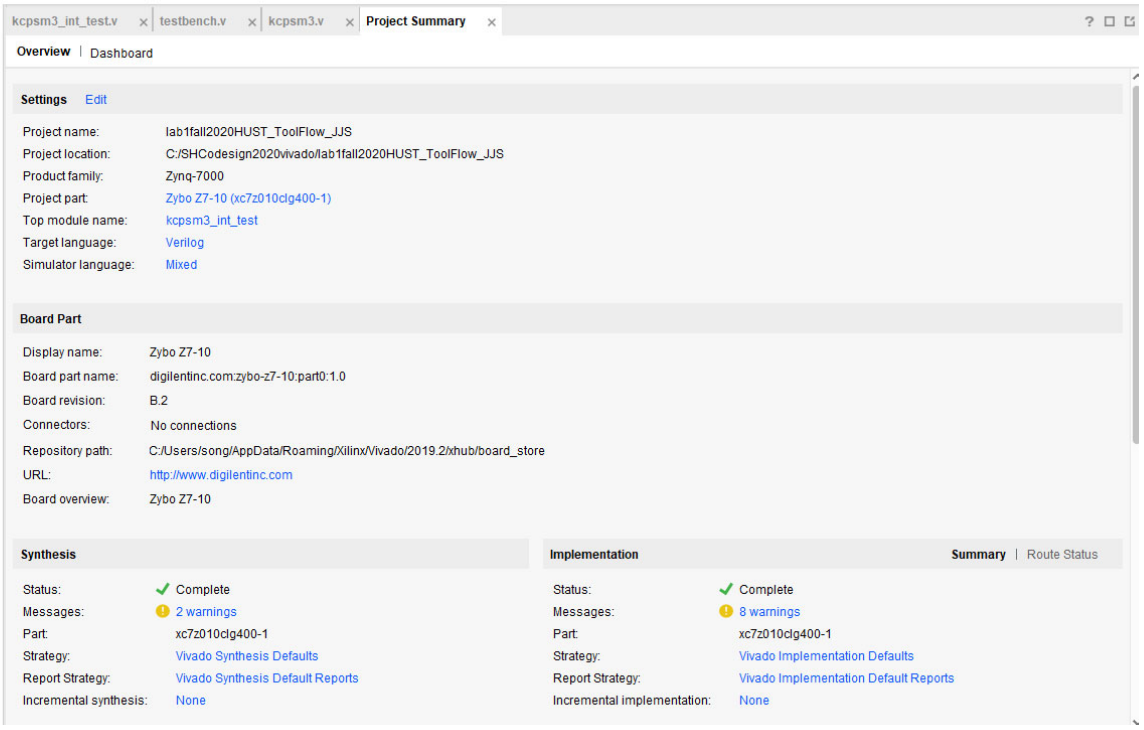
Name	Value
counter[7:0]	03
waveforms[7:0]	aa
interrupt_event	0
clk	1
period	20000 ps
address[9:0]	005
interrupt	0
interrupt_ack	0
instruction[17:0]	1c001
write_strobe	0

Figure 14. Output Waveforms from Lab #1 of Xilinx.

3.4 Implement the project and Check Design Summary Report

Make sure kcpsm3_int_test.v is the top-level circuit. Choose "Run Implement" to implement the project. When the implementation is completed, review design utilization in the Project Summary window as follows. You can also open Project Summary by clicking on Window.

Click on Project Manager at the upper left corner to see Project Summary. Project Summary Overview is copied below.



Settings

Project name:	lab1fall2020HUST_ToolFlow_JJS
Project location:	C:/SHCodesign2020vivado/lab1fall2020HUST_ToolFlow_JJS
Product family:	Zynq-7000
Project part:	Zybo Z7-10 (xc7z010clg400-1)
Top module name:	kcpsm3_int_test
Target language:	Verilog
Simulator language:	Mixed

Board Part

Display name:	Zybo Z7-10
Board part name:	digilentinc.com:zybo-z7-10:part0:1.0
Board revision:	B.2
Connectors:	No connections
Repository path:	C:/Users/song/AppData/Roaming/Xilinx/Vivado/2019.2/xhub/board_store
URL:	http://www.digilentinc.com
Board overview:	Zybo Z7-10

Synthesis

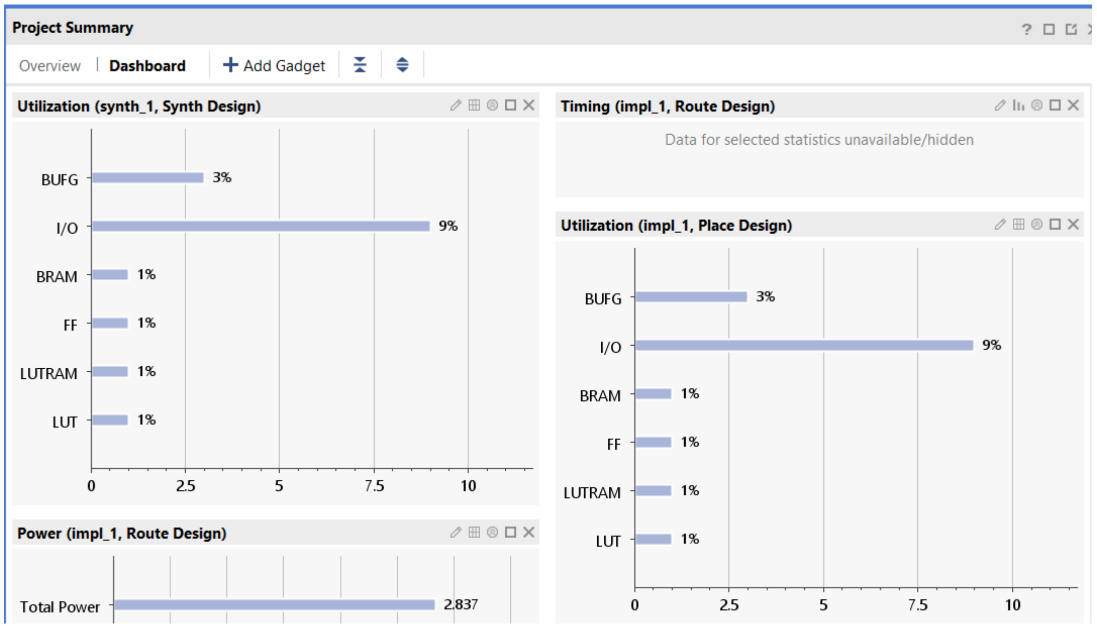
Status:	✓ Complete
Messages:	2 warnings
Part:	xc7z010clg400-1
Strategy:	Vivado Synthesis Defaults
Report Strategy:	Vivado Synthesis Default Reports
Incremental synthesis:	None

Implementation

Status:	✓ Complete
Messages:	8 warnings
Part:	xc7z010clg400-1
Strategy:	Vivado Implementation Defaults
Report Strategy:	Vivado Implementation Default Reports
Incremental implementation:	None

Table 17. Design Summary from Lab #1 of Xilinx.

You can also see Project Summary Dashboard to see Device Utilization, Timing, power etc.



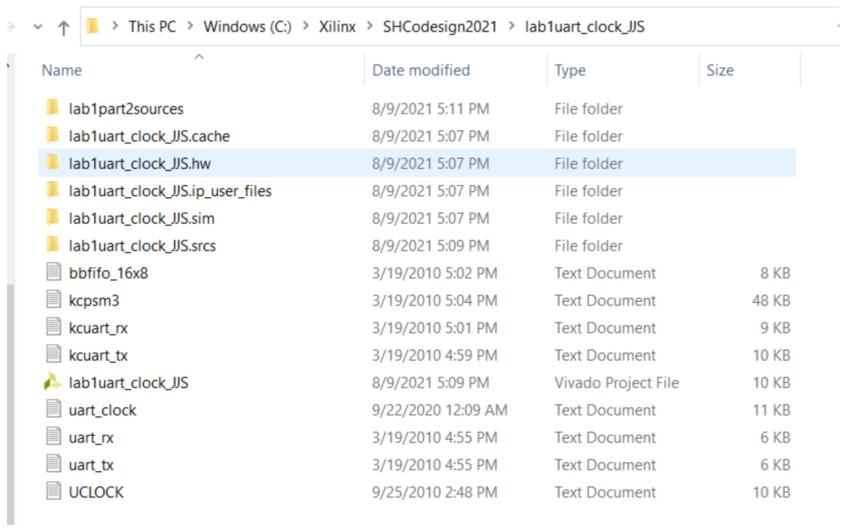
Data for selected statistics unavailable/hidden

4 Part 2: Architecture Wizard and Pins Assignment from Lab #2 of Xilinx

This part is based on “Architecture Wizard and Pins Assignment Lab Workbook” by Xilinx University Program to create a UART real-time clock. The source files are provided as lab1part2sources.zip.

4.1 Create A Project Called lab1uart_clock_JJS

Create a new project called lab1uart_clock_JJS, where JJS is your name initials, and copy source files from lab1part2sources folder to this project. Set uart_clock.v as top-level file if it is not at top level.



This PC > Windows (C:) > Xilinx > SHCodesign2021 > lab1uart_clock_JJS

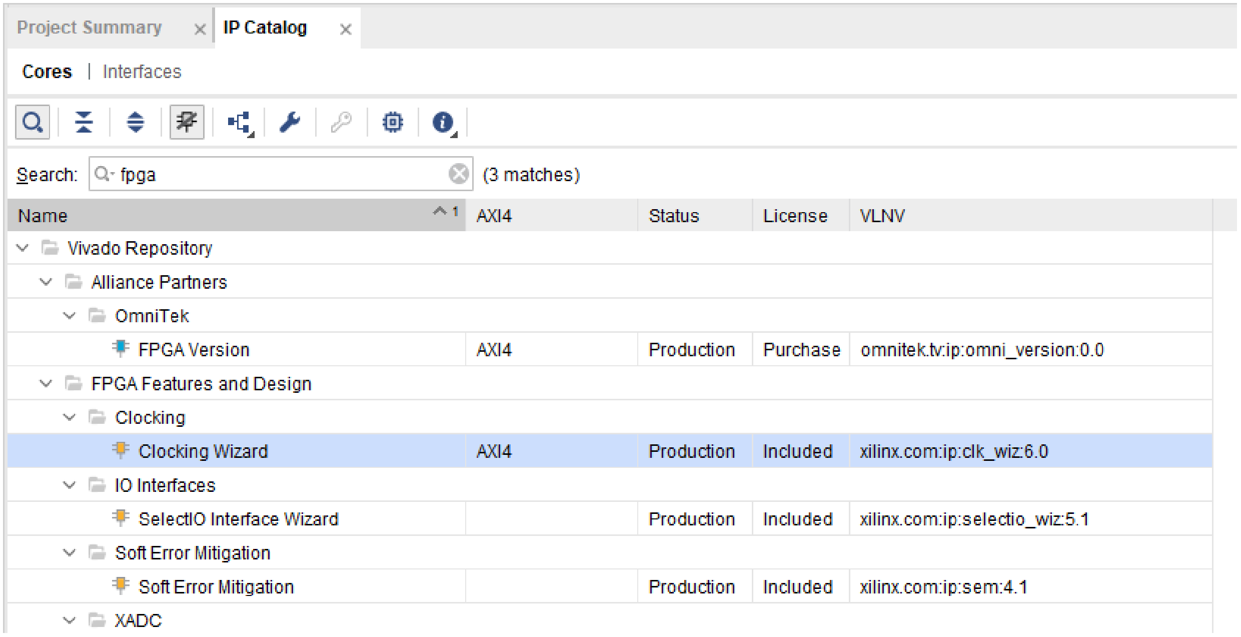
Name	Date modified	Type	Size
lab1part2sources	8/9/2021 5:11 PM	File folder	
lab1uart_clock_JJS.cache	8/9/2021 5:07 PM	File folder	
lab1uart_clock_JJS.hw	8/9/2021 5:07 PM	File folder	
lab1uart_clock_JJS.ip_user_files	8/9/2021 5:07 PM	File folder	
lab1uart_clock_JJS.sim	8/9/2021 5:07 PM	File folder	
lab1uart_clock_JJS.srcs	8/9/2021 5:09 PM	File folder	
bbfifo_16x8	3/19/2010 5:02 PM	Text Document	8 KB
kcpsm3	3/19/2010 5:04 PM	Text Document	48 KB
kcuart_rx	3/19/2010 5:01 PM	Text Document	9 KB
kcuart_tx	3/19/2010 4:59 PM	Text Document	10 KB
lab1uart_clock_JJS	8/9/2021 5:09 PM	Vivado Project File	10 KB
uart_clock	9/22/2020 12:09 AM	Text Document	11 KB
uart_rx	3/19/2010 4:55 PM	Text Document	6 KB
uart_tx	3/19/2010 4:55 PM	Text Document	6 KB
UCLOCK	9/25/2010 2:48 PM	Text Document	10 KB

4.2 A UART Real-Time Clock

Generate a 55MHz clock module from 125 MHz crystal oscillator.

4.2.1 Create a Clock IP to Generate 55MHz Clock from 125MHz Crystal Oscillator

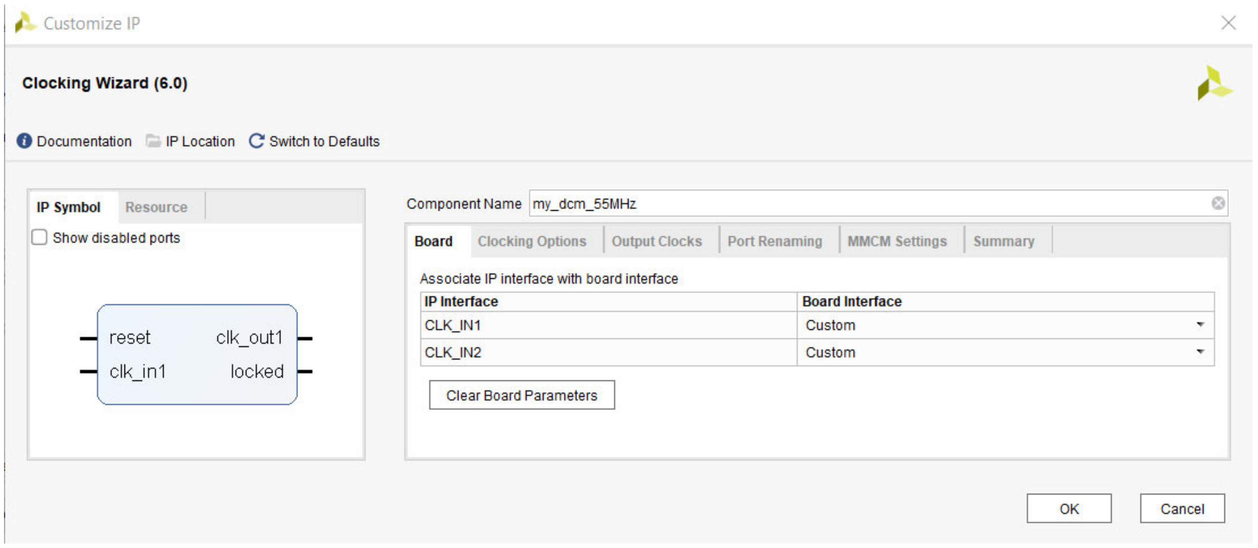
Open IP Catalog and double click on **FPGA Features and Design->Clocking->Clocking Wizard** to build a clock module in Verilog to have 125MHz input clock and 55MHz output clock. Type FPGA in search entrance to see the following. Double click on Clocking Wizard.



Search: fpga (3 matches)

Name	AXI4	Status	License	VLNV
Vivado Repository				
Alliance Partners				
OmniTek				
FPGA Version	AXI4	Production	Purchase	omnitek.tv:ip:omni_version:0.0
FPGA Features and Design				
Clocking				
Clocking Wizard	AXI4	Production	Included	xilinx.com:ip:clk_wiz:6.0
IO Interfaces				
SelectIO Interface Wizard		Production	Included	xilinx.com:ip:selectio_wiz:5.1
Soft Error Mitigation				
Soft Error Mitigation		Production	Included	xilinx.com:ip:sem:4.1
XADC				

Name this digital clock module my_dcm_55MHz.


Customize IP

Clocking Wizard (6.0)

Documentation | IP Location | Switch to Defaults

IP Symbol | Resource

☐ Show disabled ports

Component Name: my_dcm_55MHz

Board | Clocking Options | Output Clocks | Port Renaming | MMCM Settings | Summary

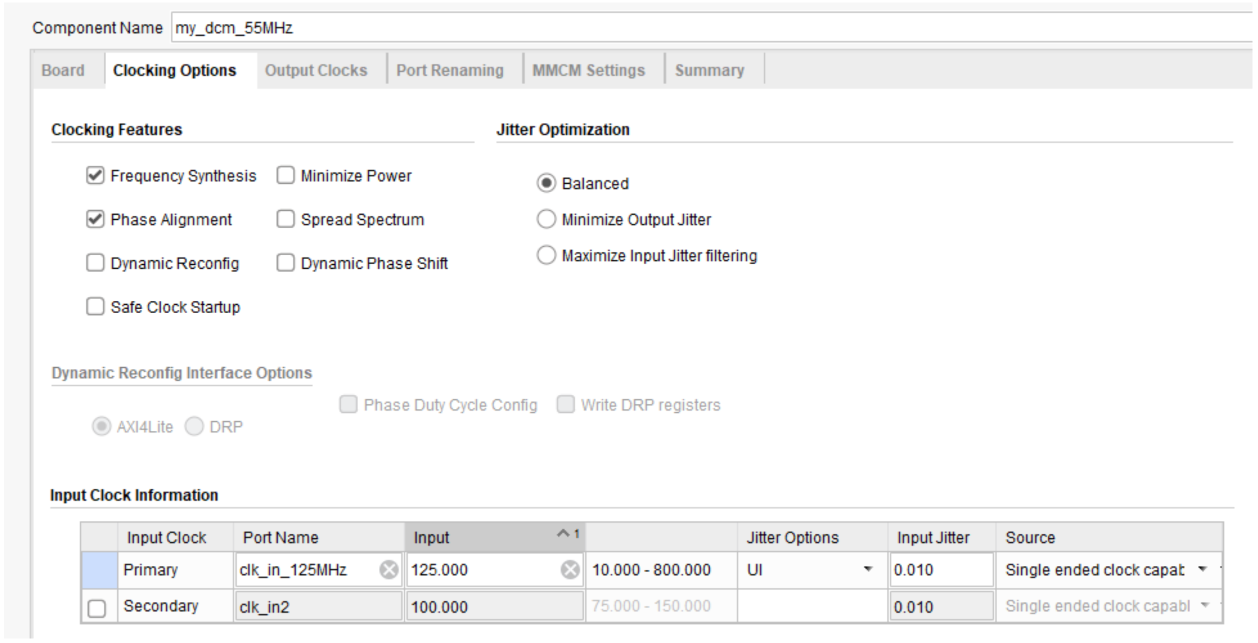
Associate IP interface with board interface

IP Interface	Board Interface
CLK_IN1	Custom
CLK_IN2	Custom

Clear Board Parameters

OK | Cancel

Choose Clocking Options and enter 125MHz as primary input clock and is named clk_in125MHz.


Component Name: my_dcm_55MHz

Board | Clocking Options | Output Clocks | Port Renaming | MMCM Settings | Summary

Clocking Features

- [✓] Frequency Synthesis
- [ ] Minimize Power
- [✓] Phase Alignment
- [ ] Spread Spectrum
- [ ] Dynamic Reconfig
- [ ] Dynamic Phase Shift
- [ ] Safe Clock Startup

Jitter Optimization

- ● Balanced
- ○ Minimize Output Jitter
- ○ Maximize Input Jitter filtering

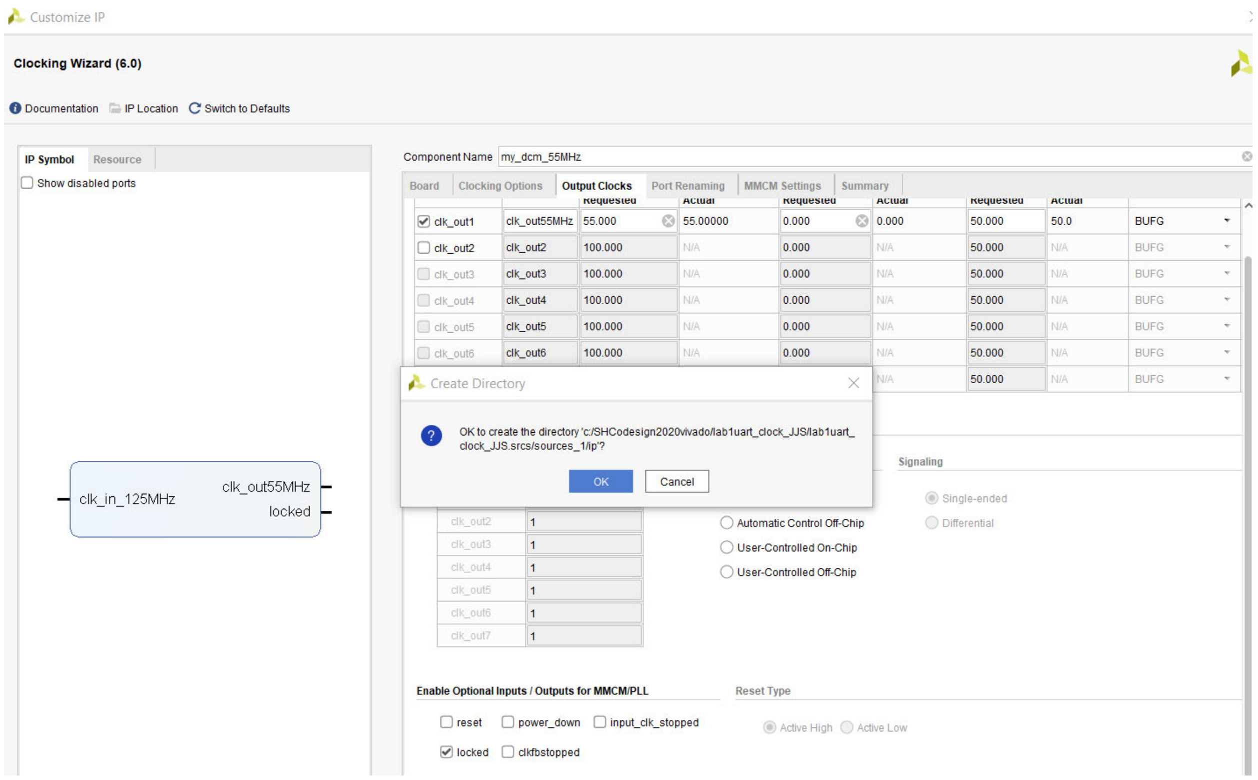
Dynamic Reconfig Interface Options

● AXI4Lite ○ DRP ☐ Phase Duty Cycle Config ☐ Write DRP registers

Input Clock Information

	Input Clock	Port Name	Input		Jitter Options	Input Jitter	Source
	Primary	clk_in_125MHz	125.000	10.000 - 800.000	UI	0.010	Single ended clock capat
☐	Secondary	clk_in2	100.000	75.000 - 150.000		0.010	Single ended clock capabl

Select Output Clocks icon. Enter Output Freq (MHz) to be 55MHz is named clk_out55MHz.
Disable reset signal at the bottom of this menu. Click OK. Click Generate Output Products if prompted.



Component Name my_dcm_55MHz

		Requested	Actual	Requested	Actual	Requested	Actual	
✓ clk_out1	clk_out55MHz	55.000	55.00000	0.000	0.000	50.000	50.0	BUFG
clk_out2	clk_out2	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
clk_out3	clk_out3	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
clk_out4	clk_out4	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
clk_out5	clk_out5	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
clk_out6	clk_out6	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
					N/A	50.000	N/A	BUFG

Create Directory

OK to create the directory 'c:/SHCodesign2020vivado/lab1uart_clock_JJS/lab1uart_clock_JJS.srcs/sources_1/ip'?

OK Cancel

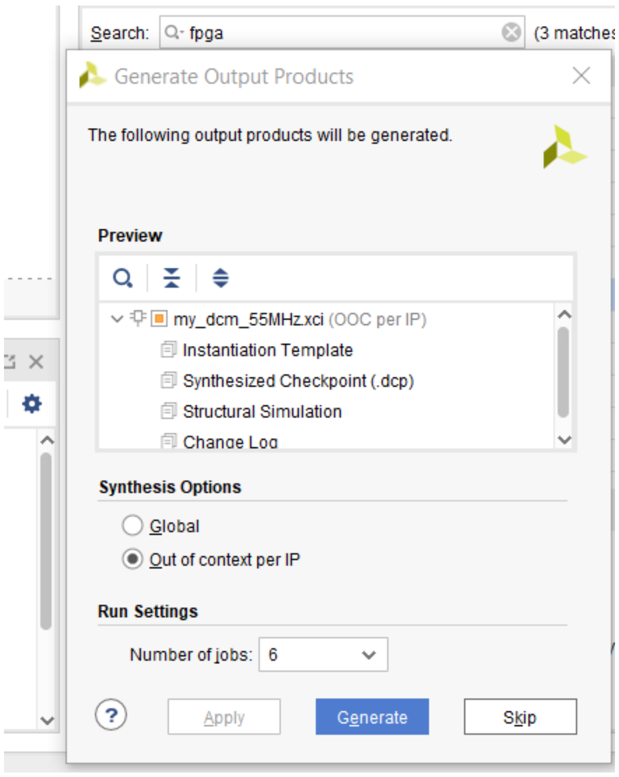
clk_out2	1
clk_out3	1
clk_out4	1
clk_out5	1
clk_out6	1
clk_out7	1

Automatic Control Off-Chip
User-Controlled On-Chip
User-Controlled Off-Chip

Signaling
[✓ Single-ended]
Differential

Enable Optional Inputs / Outputs for MMCM/PLL
reset power_down input_clk_stopped
[✓ locked] clkfbstopped

Reset Type
[✓ Active High] Active Low


Search: fpga (3 matches

Generate Output Products

The following output products will be generated.

Preview

- my_dcm_55MHz.xci (OOC per IP)
  - Instantiation Template
  - Synthesized Checkpoint (.dcp)
  - Structural Simulation
  - Change Log

Synthesis Options
Global
[✓ Out of context per IP]

Run Settings
Number of jobs: 6

4.2.2 Module my_dcm_55MHz.v

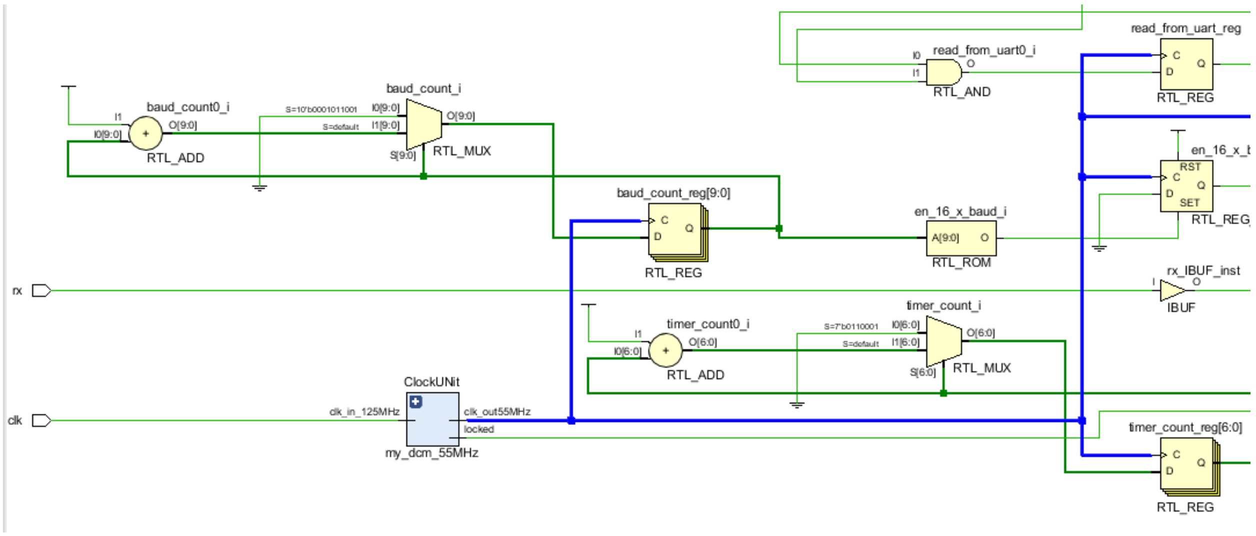
Module my_dcm_55MHz is a wrapper for a Xilinx library clock generation module. You can use it to generate a 55MHz clock signal from a 125MHz input system clock by envoking it as follow. Module uart_clock.v defines input clock is clk and internal clock as clk55MHz.

```
module my_dcm_55MHz
(
 // Clock out ports
 output      clk_out55MHz,
 // Status and control signals
 output      locked,
// Clock in ports
 input       clk_in_125MHz
);
```

Under line 170 of uart_clock file, insert this clock module as follow.

```
// Insert DCM component here

my_dcm_55MHz ClockUNit(.locked(lock), .clk_out55MHz(clk55MHz), .clk_in_125MHz(clk));

```

Open your top-level schematic to see the clock module is connected to your circuit as shown below.



4.2.3 Add One Signal lock to uart_clock.v

Add output wire lock to your uart_clock.v. wire lock is from signal locked of my_dcm_55MHz clock module and is tied to an LED to indicate status of a clock module. Signal lock is logic “1” if the clock module functions correctly and logic “0” if the clock module does not function.

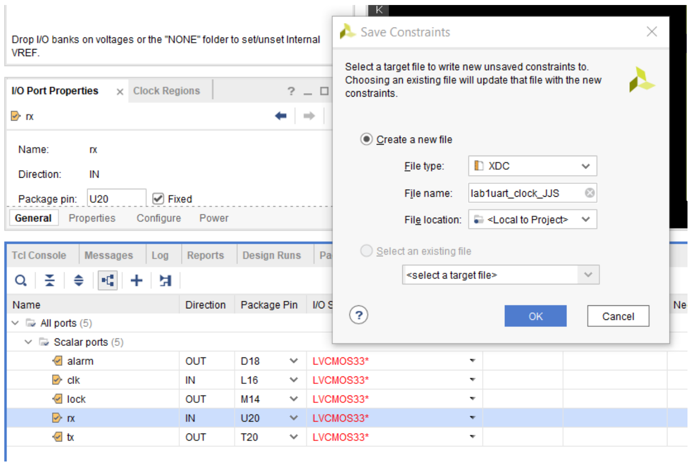
```
module uart_clock
(   tx,
    rx,
    alarm,
    clk,
    lock);

output  tx;
input   rx;
output  alarm;
input   clk;
output  lock;
```

4.3 Add Zybo pins to module uart_clock.v and Generate bit stream file

Run Synthesis to create Synthesized Design and open I/O Ports. The system clock is the 125MHz crystal oscillator from pin L16. We will connect signal lock to LD0 and signal alarm to LD3. An external UART to USB adapter is connected to Pmod B of Zybo. wire tx from uart_clock module must be connected to rx pin of the module and wire rx from uart_clock needs to be connected to tx pin of the adapter.

I/O wires	Physical Pins
alarm	LD3 (Pin D18)
lock	LD0 (Pin M14)
clk	Pin L16 125MHz crystal oscillator
rx	JB0 Pin U20 on Pmod JB
tx	JB1 Pin T20 on Pmod JB

Here are the pin assigment, where wire tx is connected to JB1 (pin T20) and wire rx on JB2 (pin U20).



Drop I/O banks on voltages or the "NONE" folder to set/unset Internal VREF.

I/O Port Properties | Clock Regions

rx

Name: rx
Direction: IN
Package pin: U20 [✓ Fixed]

General | Properties | Configure | Power

Save Constraints

Select a target file to write new unsaved constraints to. Choosing an existing file will update that file with the new constraints.

[● Create a new file]
File type: XDC
File name: lab1uart_clock_JJS
File location: <Local to Project>

○ Select an existing file
<select a target file>

OK | Cancel

Tcl Console | Messages | Log | Reports | Design Runs

Name	Direction	Package Pin	I/O S
All ports (5)			
Scalar ports (5)			
alarm	OUT	D18	LVCMOS33*
clk	IN	L16	LVCMOS33*
lock	OUT	M14	LVCMOS33*
rx	IN	U20	LVCMOS33*
tx	OUT	T20	LVCMOS33*

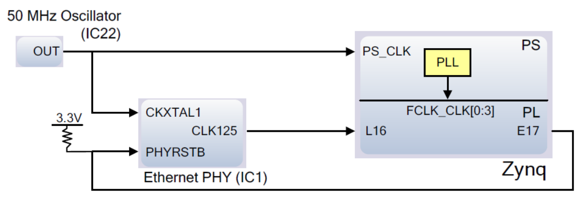
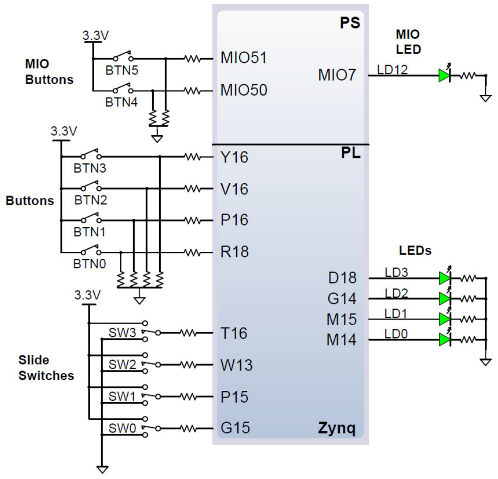


Figure 13. ZYBO clocking.



The serial-to-USB FTDI breakout board can be connected to any SelectIO pins on Pmod connectors. Four pins are connected: TX, RX, VCC and GND. FTDI TX is connected to Zybo RX and FTDI RX is connected to Zybo TX. Notice pins are numbers from left to right as 1 to 6 and top to bottom: 1 to 6 and 7 to 12. I have used Pmod JB, pins 1 for tx from uart_clock, 2 for rx from uart_clock, 5 for ground, and 6 for power. Power pin is not necessary as the adapter is powered by USB cable from the laptop.

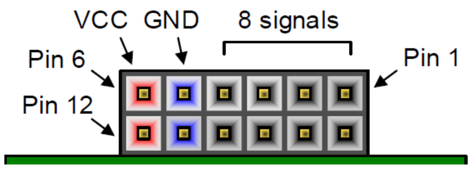


Figure 16. Pmod diagram.

4.4 Connect the uart to usb adapter board to your Zybo

Pmod JB connect is used as pin 1 for tx from uart_clock, pin 2 for rx from uart_clock, pin 5 for ground, and pin 6 for power. Power pin is not necessary as the adapter is powered by USB cable from the laptop. Use jumper wires to connect your The serial-to-USB FTDI breakout board to your Pmod JB and use a mini-USB to USB cable to connect it to your laptop.

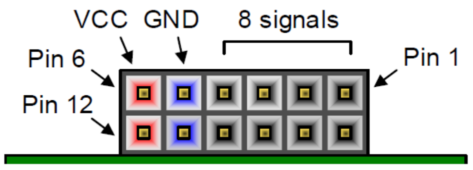
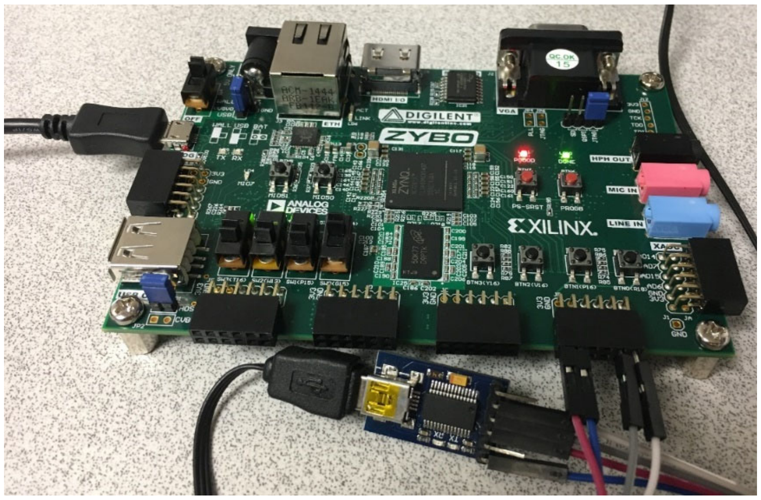
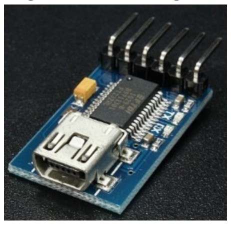
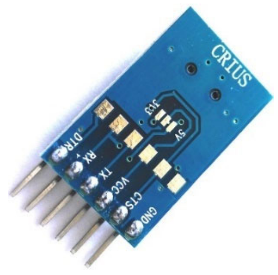
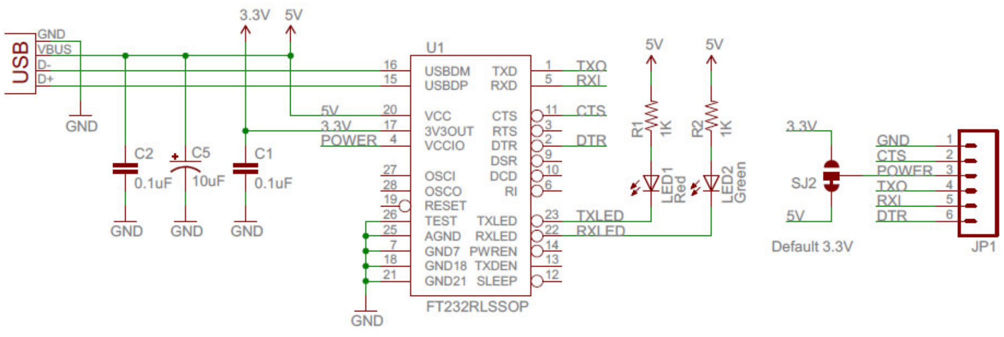


Figure 16. Pmod diagram.

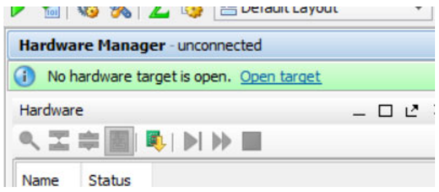








4.5 Program your Zybo board

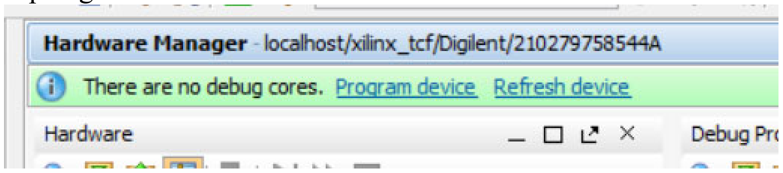

Hardware Manager - unconnected

No hardware target is open. Open target

Set your Zybo board to power by USB and connect your Zybo board to a USB port.

Choose Program and Debug->Open Hardware Manger at the bottom of the Flow Navigator under Vivado.

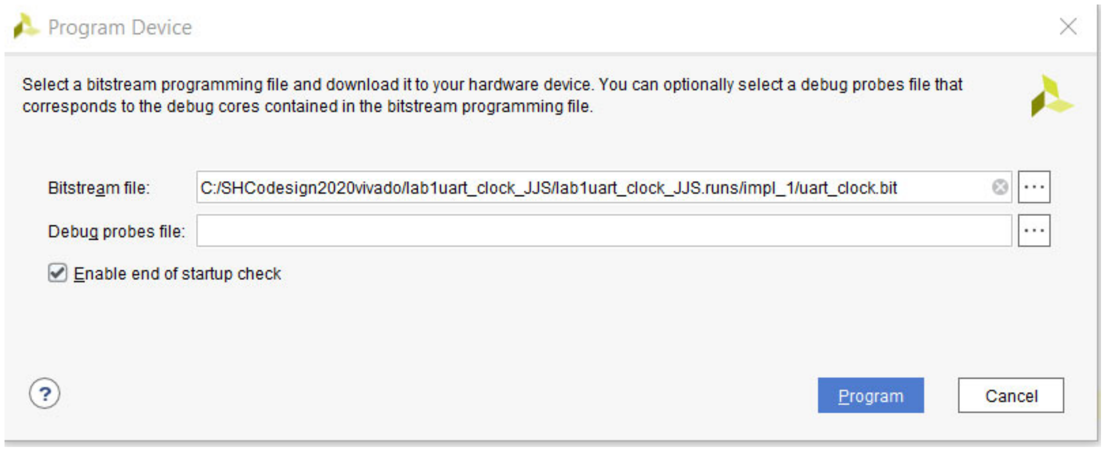
Click Open Target-> Auto Connect on top Right corner of Hardware Manger.


Hardware Manager - localhost/xilinx_tcf/Digilent/210279758544A

There are no debug cores. Program device Refresh device

Click Program Derive->xc7z010_1 to program your Zybo PL.

Deselect Debug probes file to program Zybo with Bitstream file only. Click Program.


Program Device

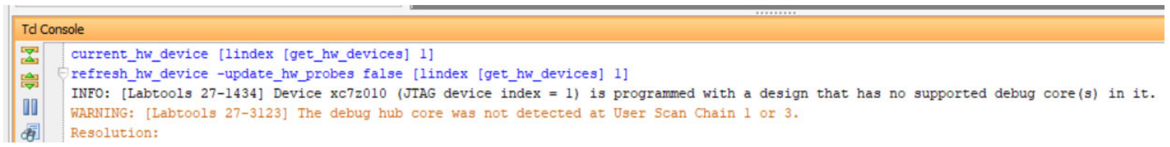
Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.

Bitstream file: C:/SHCodesign2020vivado/lab1uart_clock_JJS/lab1uart_clock_JJS.runs/impl_1/uart_clock.bit

Debug probes file:

☑ Enable end of startup check

You should see a message that your device is programmed with a design that has no supported debug core(s) in it.

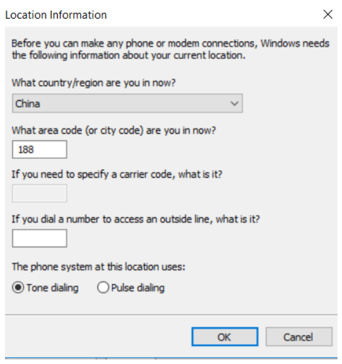

Tcl Console

```
current_hw_device [lindex [get_hw_devices] 1]
refresh_hw_device -update_hw_probes false [lindex [get_hw_devices] 1]
INFO: [Labtools 27-1434] Device xc7z010 (JTAG device index = 1) is programmed with a design that has no supported debug core(s) in it.
WARNING: [Labtools 27-3123] The debug hub core was not detected at User Scan Chain 1 or 3.
Resolution:
```

4.6 Test your uart_clock circuit with a serial terminal

One terminal software is HyperTerminal. Open HyperTerminal for the first time to see the following. Use Device Manager to find out the comm port number for your uart to usb adaptor, which is COM5 in the following case.

You can now test your circuit with a HyperTerminal. Enable "Append line feeds to incoming ends so that your HyperTerminal will respond to line feed, i.e., carriage return.


Location Information

Before you can make any phone or modem connections, Windows needs the following information about your current location.

What country/region are you in now? China

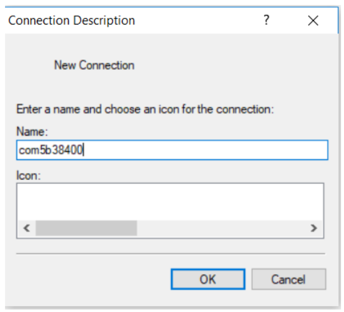
What area code (or city code) are you in now? 188

If you need to specify a carrier code, what is it?

If you dial a number to access an outside line, what is it?

The phone system at this location uses: [✓ Tone dialing] Pulse dialing

OK Cancel


Connection Description

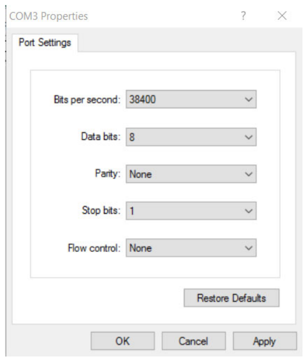
New Connection

Enter a name and choose an icon for the connection:

Name: com5b38400

Icon:

OK Cancel


COM3 Properties

Port Settings

Bits per second: 38400

Data bits: 8

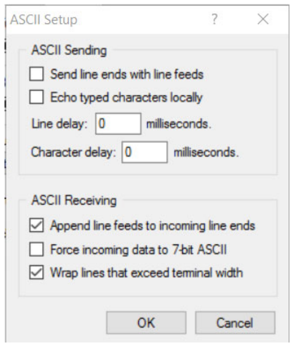
Parity: None

Stop bits: 1

Flow control: None

Restore Defaults

OK Cancel Apply


ASCII Setup

ASCII Sending

☐ Send line ends with line feeds

☐ Echo typed characters locally

Line delay: 0 milliseconds.

Character delay: 0 milliseconds.

ASCII Receiving

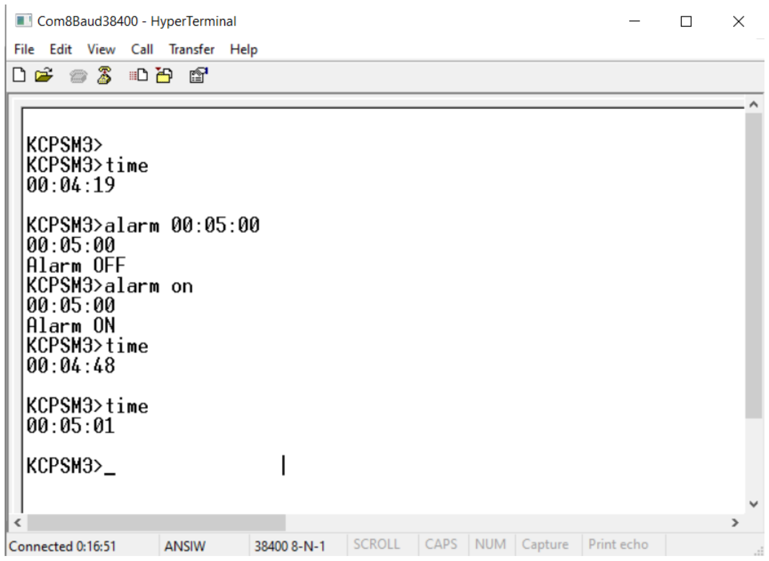
☑ Append line feeds to incoming line ends

☐ Force incoming data to 7-bit ASCII

☑ Wrap lines that exceed terminal width

OK Cancel

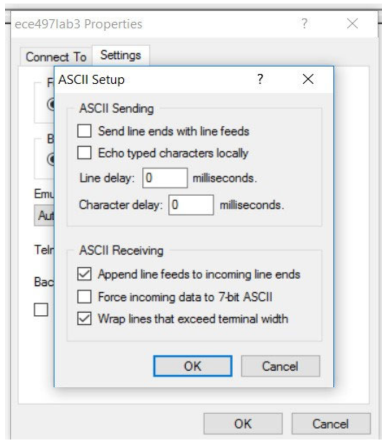
Your real-time clock circuit should display on your HyperTerminal after the board is powered. Turn on the alarm and set the alarm time. Wait until LD3 is turned on when the alarm time is equal to the real time clock time.


Com8Baud38400 - HyperTerminal

File Edit View Call Transfer Help

```
KCPSM3>
KCPSM3>time
00:04:19

KCPSM3>alarm 00:05:00
00:05:00
Alarm OFF
KCPSM3>alarm on
00:05:00
Alarm ON
KCPSM3>time
00:04:48

KCPSM3>time
00:05:01

KCPSM3>_
```

Connected 0:16:51 ANSIW 38400 8-N-1 SCROLL CAPS NUM Capture Print echo


ece497lab3 Properties

Connect To Settings

ASCII Setup

ASCII Sending

☐ Send line ends with line feeds

☐ Echo typed characters locally

Line delay: 0 milliseconds.

Character delay: 0 milliseconds.

ASCII Receiving

☑ Append line feeds to incoming line ends

☐ Force incoming data to 7-bit ASCII

☑ Wrap lines that exceed terminal width

OK Cancel

OK Cancel

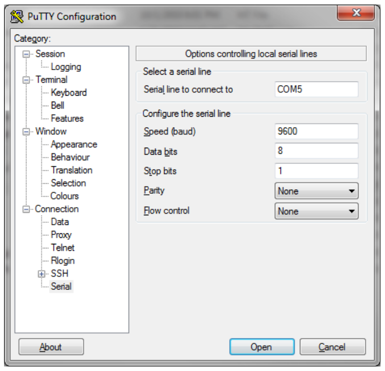
5 Serial Terminals: HyperTerminal, Putty and TeraTerm

5.1 HyperTerminal

HyperTerminal is a serial terminal software for Windows XP. Its binary code can be ported to other Windows except for Windows 10. Therefore, HyperTerminal should not be used if not necessary.

5.2 PuTTY Terminal Setup

Configure your PuTTY session by selecting Serial to set the communication port to be the right baud rate, 8-bit data, 1-bit stop, no parity bit and no flow control. Then select Session and choose Serial. Click Open to start a serial session. You can save a session by typing a saved session name and "Save".



Options controlling local serial lines

Select a serial line

Serial line to connect to: COM5

Configure the serial line

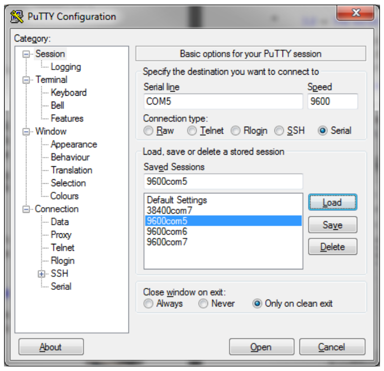
Speed (baud): 9600

Data bits: 8

Stop bits: 1

Parity: None

Flow control: None



Basic options for your PuTTY session

Specify the destination you want to connect to

Serial line: COM5 Speed: 9600

Connection type: Raw Telnet Rlogin SSH [✓ Serial]

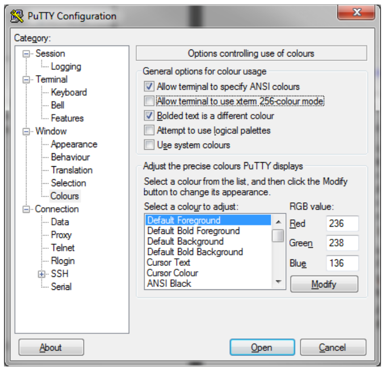
Load, save or delete a stored session

Saved Sessions: 9600com5

Default Settings
38400com7
9600com5
9600com6
9600com7

Close window on exit: Always Never [✓ Only on clean exit]

To change color, choose “Colors” in PuTTY Configuration window as follows. Select Default Foreground and choose Modify. Click on the color bars to choose a color and then click “add to custom colors” to save it. Click OK. Do the same for Default Background to change it as well. When you save a session, the color changes will be saved at the same time.



Options controlling use of colours

General options for colour usage

☑ Allow terminal to specify ANSI colours

☐ Allow terminal to use xterm 256-colour mode

☑ Bolded text is a different colour

☐ Attempt to use logical palettes

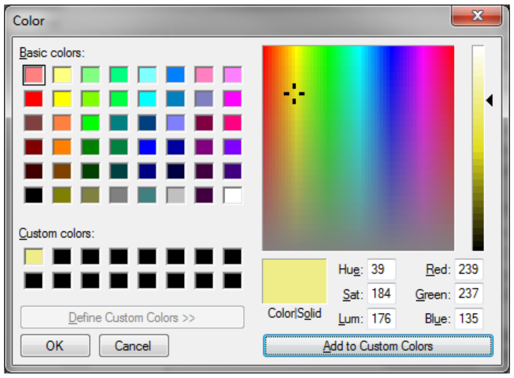
☐ Use system colours

Adjust the precise colours PuTTY displays

Select a colour from the list, and then click the Modify button to change its appearance.

Select a colour to adjust:
Default Foreground
Default Bold Foreground
Default Background
Default Bold Background
Cursor Text
Cursor Colour
ANSI Black

RGB value: Red 236 Green 238 Blue 136



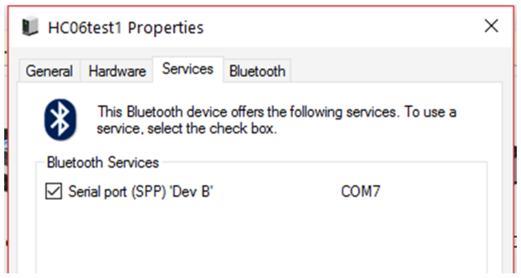
Hue: 39 Sat: 184 Lum: 176 Red: 239 Green: 237 Blue: 135

5.3 Tera Term Terminal Emulator Installation and Configuration

You can download the ***teraterm-4.67.exe*** installation file onto your desktop from the ECE230 Lab 6 Moodle folder or you can download the latest release from

http://de.sourceforge.jp/projects/ttssh2/

Click on this installation file to install Tera Term on your PC once it has been downloaded. Select all installation defaults, except for when you see the “Select Components” window, it is important to deselect (uncheck) all boxes except for the “***Tera Term & Macro***” box (which cannot be unchecked). Otherwise, many extra unneeded features will be downloaded along with the Tera Term terminal emulator.



This Bluetooth device offers the following services. To use a service, select the check box.

Bluetooth Services

☑ Serial port (SPP) 'Dev B' COM7

After pairing your laptop with the Bluetooth device, click on the Tera Term icon, and a Tera Term “New Connection Window” should appear. Click on the “Serial” button and click on the down arrow in the Port selection box. Select the COM port that corresponds to your HC-06. Your PC may create two ports for the Bluetooth device. If it does, check the properties of the Bluetooth device in “Devices and Printers”, go to the Services tab, and check which port is listed as the “Dev B” port. See the image below.

A Tera Term Communication Window should appear that is running at the default settings of "9600 baud, 8 data bits, no parity, no flow control (no handshaking)." To change the settings for this window (if necessary), click on the "Setup Tab" at the top of the Tera Term window, and then select "Serial Port". A Tera Term "Serial Port Setup" window appears that lets you change the Port, Baud Rate, Parity, Number of Stop Bits, and Flow Control settings, as needed. Click OK when done.

You should now be viewing the Tera Term Communication Window. With the mouse clicked somewhere inside this window, you can type on your PC keyboard (which is now acting as the Tera Term keyboard), and you are now sending serial characters out on the TX-O pin. However, you will not see the typed characters echoed on the screen unless you perform a hardware loopback test, or of course, unless you are running a program on your PIC that sends received characters back out to the terminal display.