Name: ______________________ ID: ___________ Start Date: Tuesday, June 11, 2019
Name: ______________________ ID: ___________ Due Date: Wednesday, June 12, 2019

Software and Hardware Co-Design with Zybo, Spring 2019 HUST Lab #9 Part 2: Custom IP of a UART in Verilog and Standard Terminal to Display Two TMP101s from MIO and SelectIO

This lab is Part 2 of Lab #9. This is a group lab. Each group of two students should demonstrate your I2C TMP101 reading implementation on Zybo board and temperature displays on both standard terminal from ARM core and your own UAERT IP terminal and submit one pdf copy of report for this lab. Late lab submission will be accepted with a grade reduction of 10% for each day that it is late.

I. Available Source Files

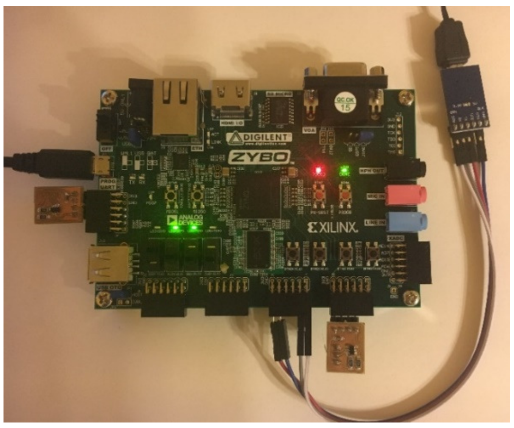
UARTmodule2018spring.zip contains all files of a uart module and it is available from the instructor.

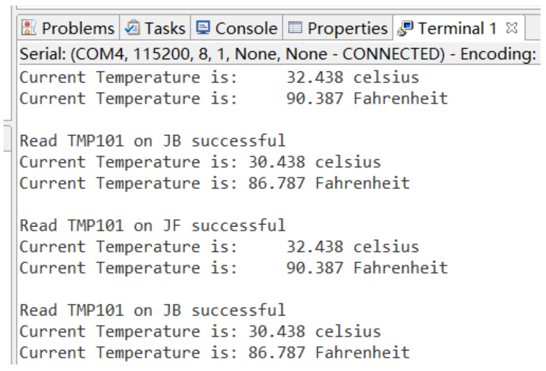
II. Deliverables

(1) *Demonstrate your implementation on your Zybo board.*
(2) *Submit a hard copy of your screen captures with two serial-port terminals to show two TMP101 temperatures. Submit a screen capture of your block design clearly readable.*
(3) *Submit a pdf copy of your C program for this part.*

III. Objectives

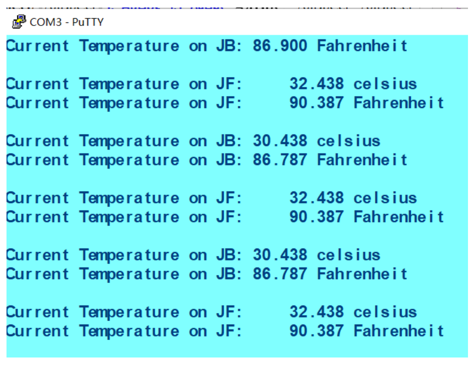
(4) *Create an IP for a UART in Verilog from uart source files in Verilog provided by the instructor. These files are similar to those in Lab #1.*
(5) *Create a new Vivado RTL project called lab9part2uartIP from Lab #9 Part 1. Add your UART IP to this block design.*
(6) *Write your program for this new hardware block design to display two temperatures from two TMP101 modules on both standard and your custom IP UART terminal that is connected to your FTDI breakout board.*





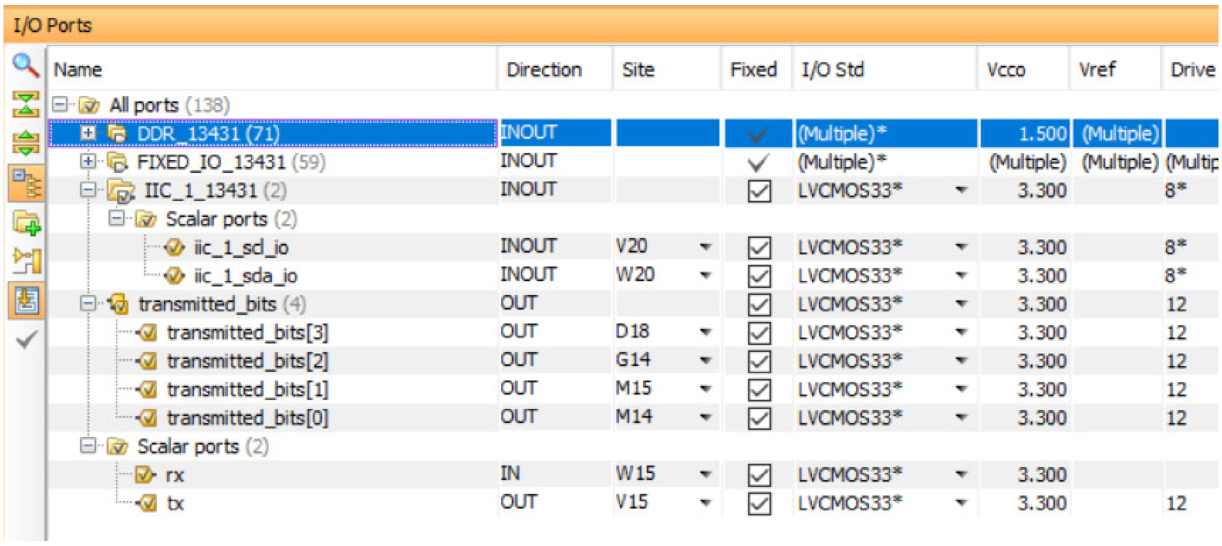
Serial: (COM4, 115200, 8, 1, None, None - CONNECTED) - Encoding:

```
Current Temperature is:     32.438 celsius
Current Temperature is:     90.387 Fahrenheit

Read TMP101 on JB successful
Current Temperature is: 30.438 celsius
Current Temperature is: 86.787 Fahrenheit

Read TMP101 on JF successful
Current Temperature is:     32.438 celsius
Current Temperature is:     90.387 Fahrenheit

Read TMP101 on JB successful
Current Temperature is: 30.438 celsius
Current Temperature is: 86.787 Fahrenheit
```

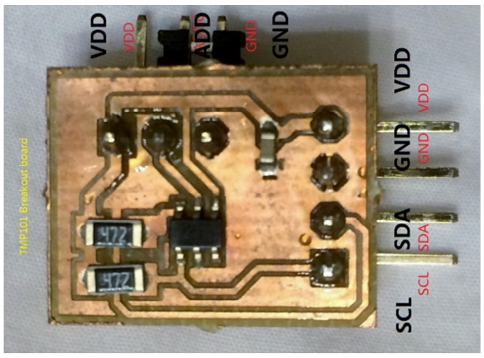


```
Current Temperature on JB: 86.900 Fahrenheit

Current Temperature on JF:     32.438 celsius
Current Temperature on JF:     90.387 Fahrenheit

Current Temperature on JB: 30.438 celsius
Current Temperature on JB: 86.787 Fahrenheit

Current Temperature on JF:     32.438 celsius
Current Temperature on JF:     90.387 Fahrenheit

Current Temperature on JB: 30.438 celsius
Current Temperature on JB: 86.787 Fahrenheit

Current Temperature on JF:     32.438 celsius
Current Temperature on JF:     90.387 Fahrenheit
```

The TMP101 breakout board in Port F is connected to the PS through MIO pins and is connected to Port F are as follows: SDK on JF9 (M14) and SDA on JF10 (M15). The TMP101 on Port B is connected to the PS through SelectIO pins and is connected so that JB3 (V20) is SCL and JB4 (W20) is SDA. The custom UART is connected to Pmod C. transmitted_bits are displayed on the four LEDs.


I/O Ports

Name	Direction	Site	Fixed	I/O Std	Vcco	Vref	Drive
All ports (138)							
DDR_13431 (71)	INOUT		✓	(Multiple)*	1.500	(Multiple)	
FIXED_IO_13431 (59)	INOUT		✓	(Multiple)*	(Multiple)	(Multiple)	(Multip
IIC_1_13431 (2)	INOUT		✓	LVCMOS33*	3.300		8*
Scalar ports (2)							
iic_1_sd_io	INOUT	V20	✓	LVCMOS33*	3.300		8*
iic_1_sda_io	INOUT	W20	✓	LVCMOS33*	3.300		8*
transmitted_bits (4)	OUT		✓	LVCMOS33*	3.300		12
transmitted_bits[3]	OUT	D18	✓	LVCMOS33*	3.300		12
transmitted_bits[2]	OUT	G14	✓	LVCMOS33*	3.300		12
transmitted_bits[1]	OUT	M15	✓	LVCMOS33*	3.300		12
transmitted_bits[0]	OUT	M14	✓	LVCMOS33*	3.300		12
Scalar ports (2)							
rx	IN	W15	✓	LVCMOS33*	3.300		
tx	OUT	V15	✓	LVCMOS33*	3.300		12



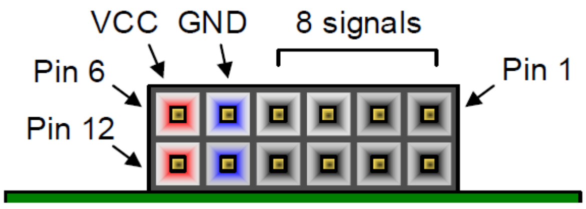
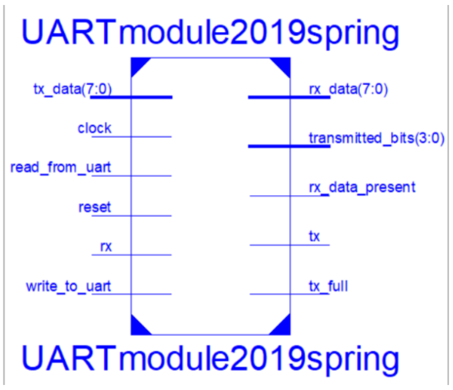


Figure 16. Pmod diagram.

Pmod JA (XADC)	Pmod JB (Hi-Speed)	Pmod JC (Hi-Speed)	Pmod JD (Hi-Speed)	Pmod JE (Std.)	Pmod JF (MIO)
JA1: N15	JB1: T20	JC1: V15	JD1: T14	JE1: V12	JF1: MIO-13
JA2: L14	JB2: U20	JC2: W15	JD2: T15	JE2: W16	JF2: MIO-10
JA3: K16	JB3: V20	JC3: T11	JD3: P14	JE3: J15	JF3: MIO-11
JA4: K14	JB4: W20	JC4: T10	JD4: R14	JE4: H15	JF4: MIO-12
JA7: N16	JB7: Y18	JC7: W14	JD7: U14	JE7: V13	JF7: MIO-0
JA8: L15	JB8: Y19	JC8: Y14	JD8: U15	JE8: U17	JF8: MIO-9
JA9: J16	JB9: W18	JC9: T12	JD9: V17	JE9: T17	JF9: MIO-14
JA10: J14	JB10: W19	JC10: U12	JD10: V18	JE10: Y17	JF10: MIO-15

IV. A UART Module, UARTmodule2019spring.v, in Verilog

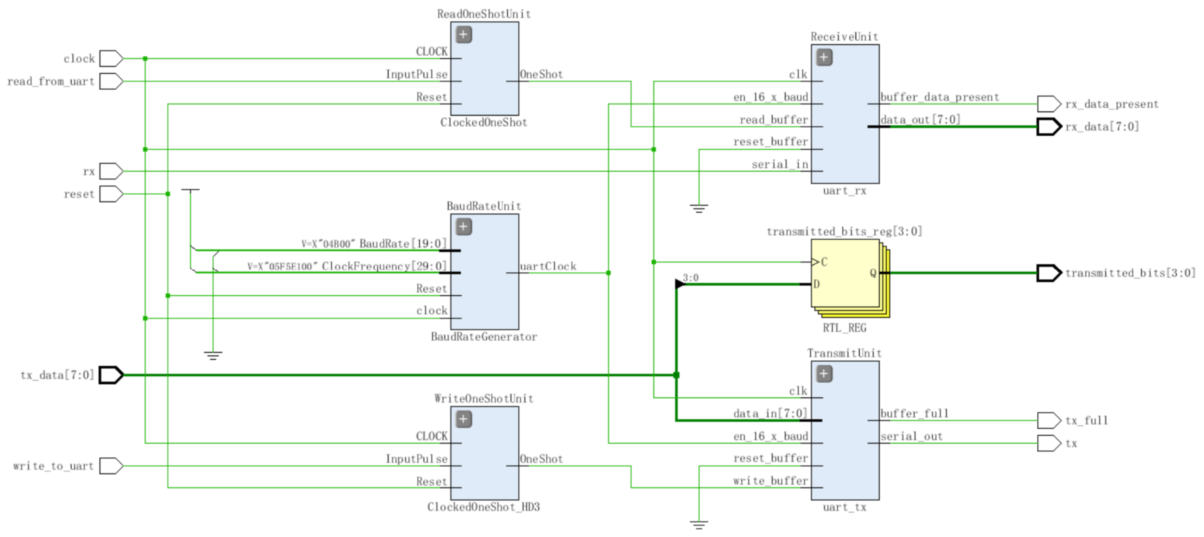


A uart module, called UARTmodule2019spring, is available from the class file site. It is based on UART transmit and receive modules in Verilog from Xilinx. The modules include 16-character circular buffers for both transmit and receive.

It has been revised to include a new baud rate generator with two parameters; BAUD RATE and SYSTEM CLOCK FREQUENCY. These parameters can be changed when the module is made an IP.

To send a character, check if tx_full is false. If it is, place data on tx_data and send a low to high pulse to write_to_uart. The module has an internal one-shot circuit to convert this pulse to a one-shot so that the data will be written into transmit buffer once only.

To receive, check if rx_data_present is true. If it is true, get data from rx_data and send a low to high pulse to read_from_uart to remove the data item from the receive buffer.

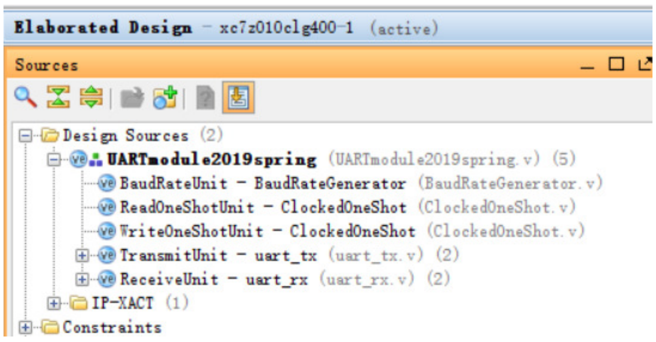


V. Make an IP for UARTmodule2019spring.v

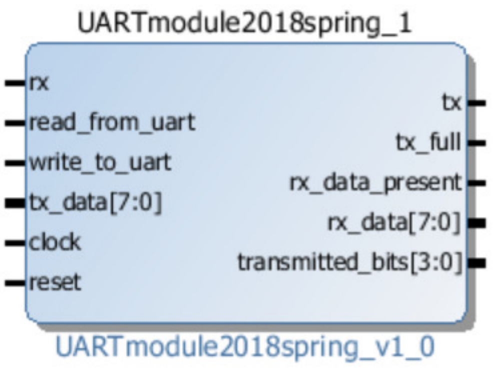
Get UARTmodule2019spring.v and its component files as UARTmodule source files.zip. Make an IP for it.

To make an IP for UARTmodule2019spring.v, create a Vivado RTL project for Zybo board and import all source files for UARTmodule2019spring.v to this project. **Change reset signals to active low as Zybo resets are all active low.**

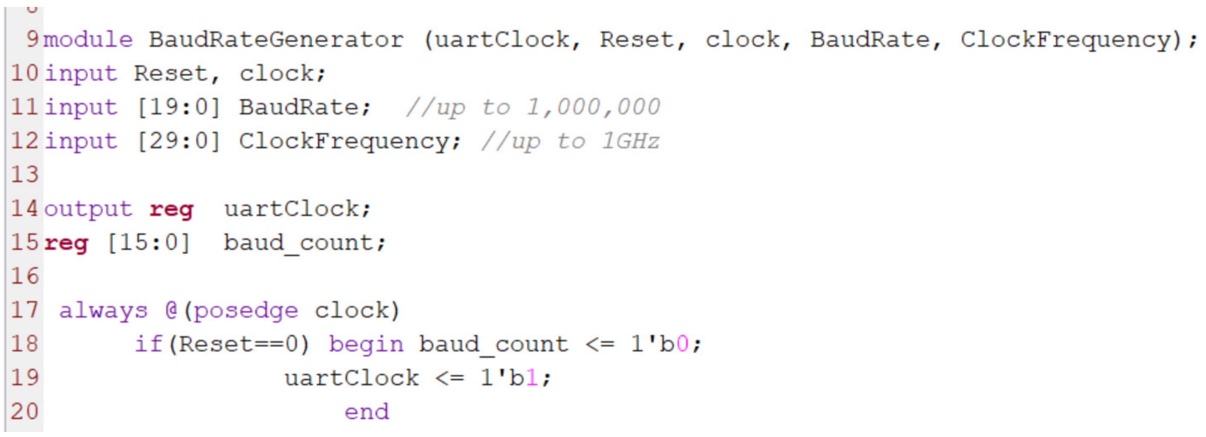
Open "Elaborated Design" and display its schematic. Run Tools->Create and Package IP. Choose "Package your current project". Choose "Include generated files". There is one waring about Reset pin being wrong polarity and it can be ignored. Choose "Review and Package" and run "Repackage IP". The following screen captures are from last year's implementation.



```
Elaborated Design - xc7z010clg400-1 (active)
Sources
Design Sources (2)
  UARTmodule2019spring (UARTmodule2019spring.v) (5)
    BaudRateUnit - BaudRateGenerator (BaudRateGenerator.v)
    ReadOneShotUnit - ClockedOneShot (ClockedOneShot.v)
    WriteOneShotUnit - ClockedOneShot (ClockedOneShot.v)
    TransmitUnit - uart_tx (uart_tx.v) (2)
    ReceiveUnit - uart_rx (uart_rx.v) (2)
  IP-XACT (1)
Constraints
```

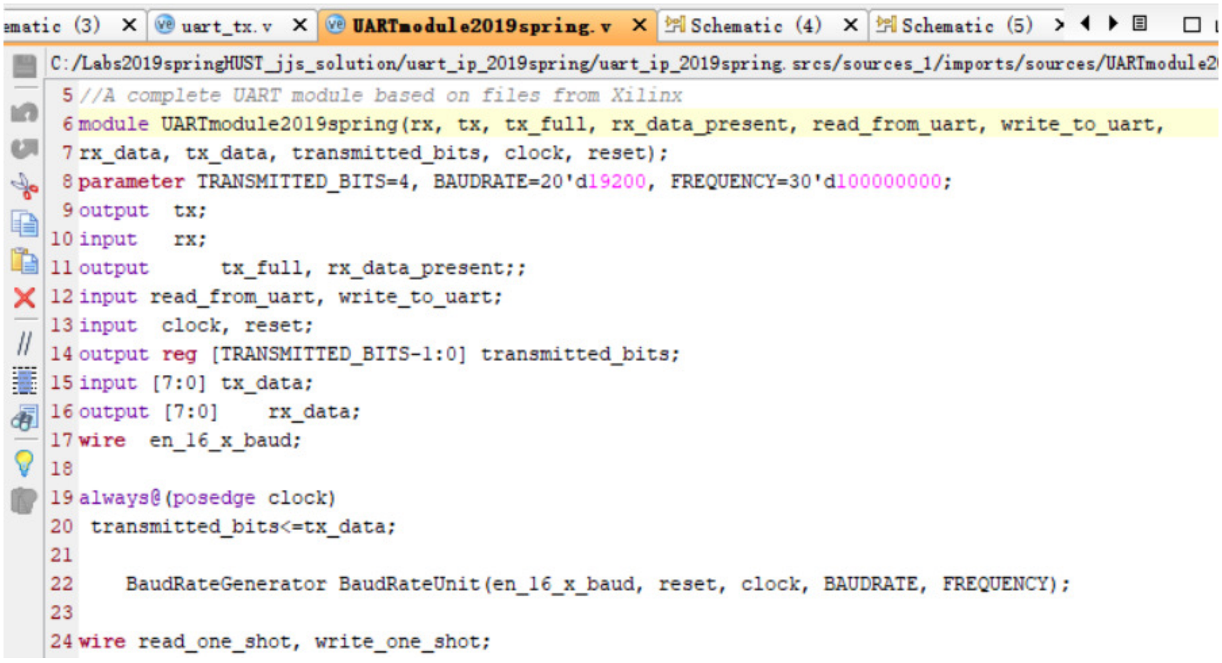


Change reset signal to active low in both baud rate generator module and clocked one-shot module.



```verilog
module BaudRateGenerator (uartClock, Reset, clock, BaudRate, ClockFrequency);
input Reset, clock;
input [19:0] BaudRate;   //up to 1,000,000
input [29:0] ClockFrequency; //up to 1GHz

output reg  uartClock;
reg [15:0]  baud_count;

 always @(posedge clock)
      if(Reset==0) begin baud_count <= 1'b0;
                    uartClock <= 1'b1;
                        end
```

Notice the baud rate and system clock frequency are defined as parameters.

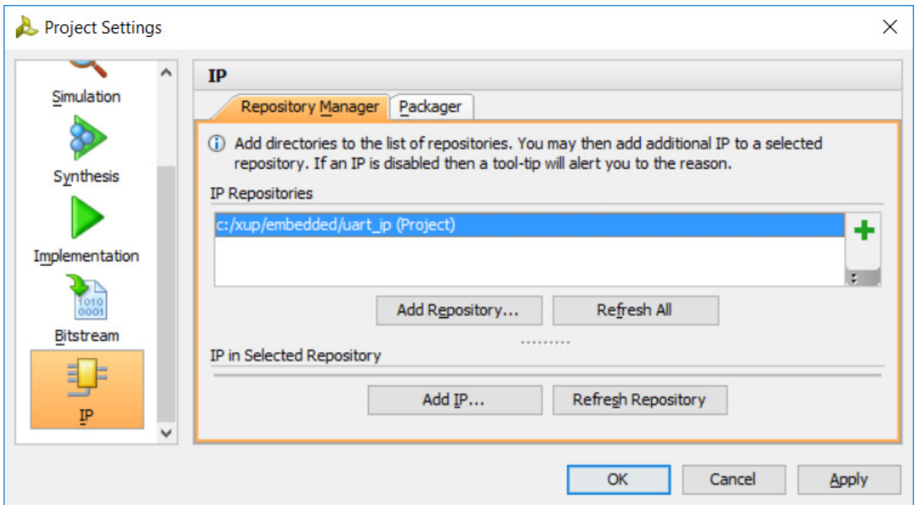


```verilog
ematic (3) ×  uart_tx.v ×  UARTmodule2019spring.v ×  Schematic (4) ×  Schematic (5)
C:/Labs2019springHUST_jjs_solution/uart_ip_2019spring/uart_ip_2019spring.srcs/sources_1/imports/sources/UARTmodule2
//A complete UART module based on files from Xilinx
module UARTmodule2019spring(rx, tx, tx_full, rx_data_present, read_from_uart, write_to_uart,
rx_data, tx_data, transmitted_bits, clock, reset);
parameter TRANSMITTED_BITS=4, BAUDRATE=20'd19200, FREQUENCY=30'd100000000;
output  tx;
input   rx;
output      tx_full, rx_data_present;;
input read_from_uart, write_to_uart;
input  clock, reset;
output reg [TRANSMITTED_BITS-1:0] transmitted_bits;
input [7:0] tx_data;
output [7:0]    rx_data;
wire  en_16_x_baud;

always@(posedge clock)
 transmitted_bits<=tx_data;

   BaudRateGenerator BaudRateUnit(en_16_x_baud, reset, clock, BAUDRATE, FREQUENCY);

wire read_one_shot, write_one_shot;
```

VI. Create a Block Design for Lab #9 Part 2 with two I2C Controller Modules and the UART IP

Create a new Vivado project, called lab9part2I2CuartIP, and block design based on Lab #9 Part 1 project. This project should work with the C code for Lab #9 Part 1 with two I2C TMP101 temperature sensors and the standard terminal. Each time an IP is added, execute “RUN CONNECTION AUTOMATION” if possible.

Add the UART IP to this block design by opening Project Settings. Choose IP and add Repository for uart IP to this project. Your IP may be located in a different folder.



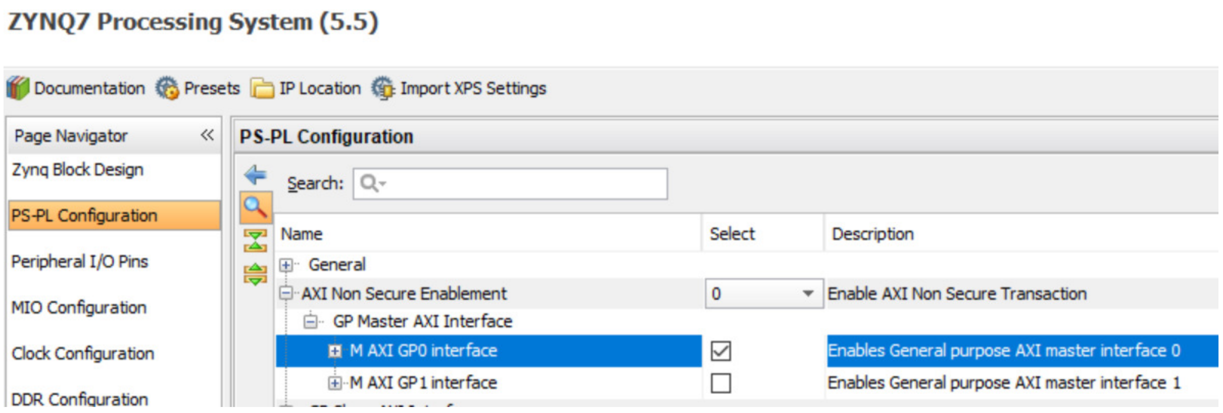
Add directories to the list of repositories. You may then add additional IP to a selected repository. If an IP is disabled then a tool-tip will alert you to the reason.

IP Repositories

c:/xup/embedded/uart_ip (Project)

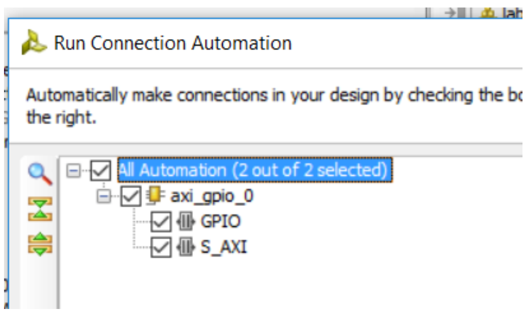
IP in Selected Repository

Choose PL clock to be FCLK_CLK0 under Clock Configuration and Enable Clock Resets to be FCLK_RESET0_N under PS-PL Configuration. Set FCLK_CLK0 to be 100MHz or whatever clock frequency for the IP to run both AXI and the UART IP. The reset signal is AXI: FCLK_RESET0_N, **which is active low**. Enable Master AXI GP0 interface for Zynq7.



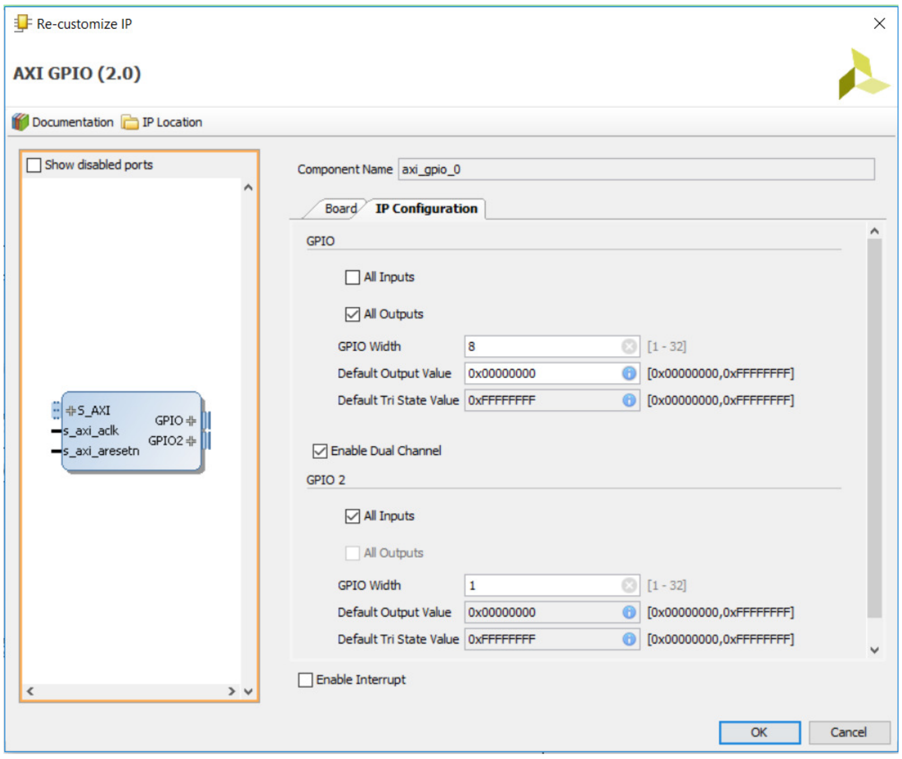
Name	Select	Description
General		
AXI Non Secure Enablement	0	Enable AXI Non Secure Transaction
GP Master AXI Interface		
M AXI GP0 interface	☑	Enables General purpose AXI master interface 0
M AXI GP1 interface	☐	Enables General purpose AXI master interface 1

Add axi_gpio IP module GPIO 0 as dual channels to connect 8-bit tx_data output and 1-bit write_to_uart output. Add IP axi_gpio IP module GPIO 1 for single channel for 1-bit tx_full input. When an IP is added, execute “Run Block Automation” and choose “All Automation”.



Automatically make connections in your design by checking the bo
the right.

- ☑ All Automation (2 out of 2 selected)
  - ☑ axi_gpio_0
    - ☑ GPIO
    - ☑ S_AXI


Re-customize IP

AXI GPIO (2.0)

Documentation IP Location

Show disabled ports

Component Name axi_gpio_0

Board | IP Configuration

GPIO

- [ ] All Inputs
- [x] All Outputs

GPIO Width	8	[1 - 32]
Default Output Value	0x00000000	[0x00000000,0xFFFFFFFF]
Default Tri State Value	0xFFFFFFFF	[0x00000000,0xFFFFFFFF]

- [x] Enable Dual Channel

GPIO 2

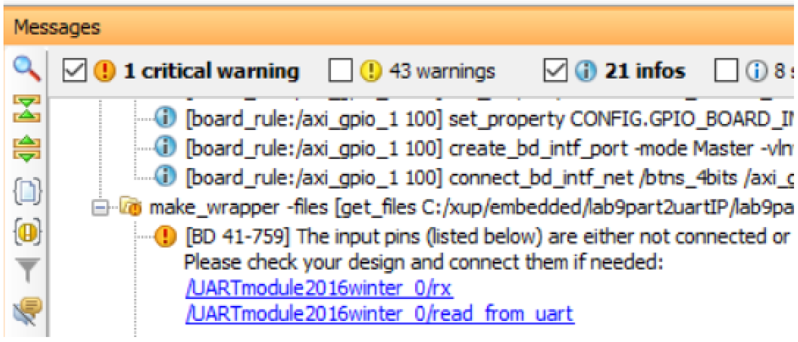
- [x] All Inputs
- [ ] All Outputs

GPIO Width	1	[1 - 32]
Default Output Value	0x00000000	[0x00000000,0xFFFFFFFF]
Default Tri State Value	0xFFFFFFFF	[0x00000000,0xFFFFFFFF]

- [ ] Enable Interrupt

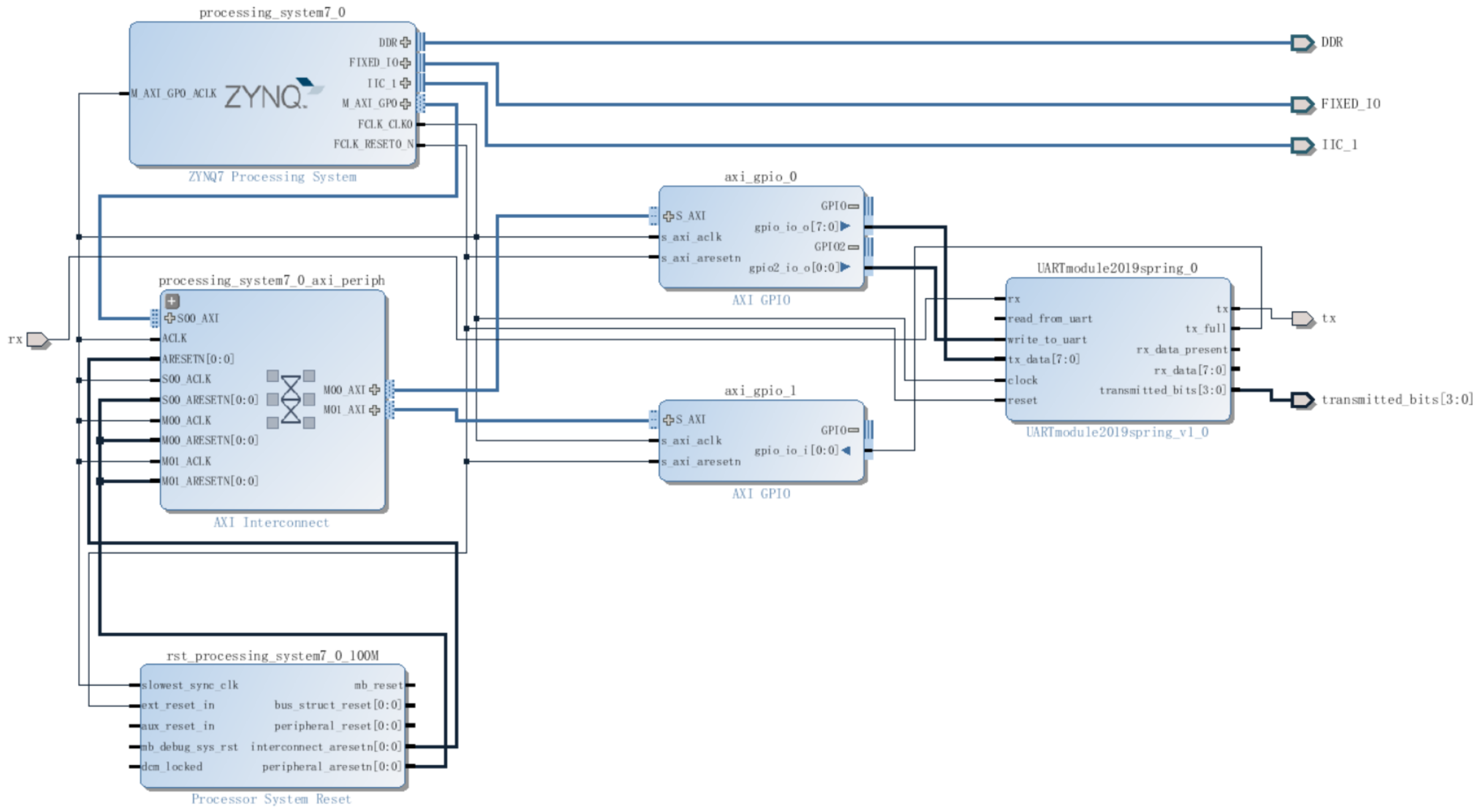
OK Cancel

Ignore the critical warning that the input pins of the uart IP are not connected because we do not need to read from the uart IP.

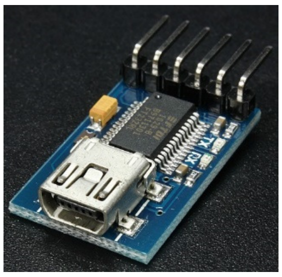

Messages

[x] 1 critical warning [ ] 43 warnings [x] 21 infos [ ] 8

```
[board_rule:/axi_gpio_1 100] set_property CONFIG.GPIO_BOARD_I
[board_rule:/axi_gpio_1 100] create_bd_intf_port -mode Master -vln
[board_rule:/axi_gpio_1 100] connect_bd_intf_net /btns_4bits /axi_
make_wrapper -files [get_files C:/xup/embedded/lab9part2uartIP/lab9pa
[BD 41-759] The input pins (listed below) are either not connected or
Please check your design and connect them if needed:
/UARTmodule2016winter_0/rx
/UARTmodule2016winter_0/read_from_uart
```


processing_system7_0
DDR
FIXED_IO
IIC_1
M_AXI_GP0_ACLK
M_AXI_GP0
FCLK_CLK0
FCLK_RESET0_N
ZYNQ
ZYNQ7 Processing System
axi_gpio_0
S_AXI
s_axi_aclk
s_axi_aresetn
GPIO
gpio_io_o[7:0]
GPIO2
gpio2_io_o[0:0]
AXI GPIO
processing_system7_0_axi_periph
S00_AXI
ACLK
ARESETN[0:0]
S00_ACLK
S00_ARESETN[0:0]
M00_ACLK
M00_ARESETN[0:0]
M01_ACLK
M01_ARESETN[0:0]
M00_AXI
M01_AXI
AXI Interconnect
rx
axi_gpio_1
S_AXI
s_axi_aclk
s_axi_aresetn
GPIO
gpio_io_i[0:0]
AXI GPIO
UARTmodule2019spring_0
rx
read_from_uart
write_to_uart
tx_data[7:0]
clock
reset
tx
tx_full
rx_data_present
rx_data[7:0]
transmitted_bits[3:0]
UARTmodule2019spring_v1_0
DDR
FIXED_IO
IIC_1
tx
transmitted_bits[3:0]
rst_processing_system7_0_100M
slowest_sync_clk
ext_reset_in
aux_reset_in
mb_debug_sys_rst
dcm_locked
mb_reset
bus_struct_reset[0:0]
peripheral_reset[0:0]
interconnect_aresetn[0:0]
peripheral_aresetn[0:0]
Processor System Reset

VII. SparkFun 3.3V FTDI Basic Breakout Board



Connect your FTDI basic breakout board to Pmod C and your laptop. Start a serial terminal on your laptop.

VIII. Software Development

VIII.1 Driver for the uart IP

The gpio interfaces are defined in xparameters.h as follows.

```
/*****************************************************

/* Definitions for driver GPIO */
#define XPAR_XGPIO_NUM_INSTANCES 2

/* Definitions for peripheral AXI_GPIO_0 */
#define XPAR_AXI_GPIO_0_BASEADDR 0x41200000
#define XPAR_AXI_GPIO_0_HIGHADDR 0x4120FFFF
#define XPAR_AXI_GPIO_0_DEVICE_ID 0
#define XPAR_AXI_GPIO_0_INTERRUPT_PRESENT 0
#define XPAR_AXI_GPIO_0_IS_DUAL 1


/* Definitions for peripheral AXI_GPIO_1 */
#define XPAR_AXI_GPIO_1_BASEADDR 0x41210000
#define XPAR_AXI_GPIO_1_HIGHADDR 0x4121FFFF
#define XPAR_AXI_GPIO_1_DEVICE_ID 1
#define XPAR_AXI_GPIO_1_INTERRUPT_PRESENT 0
#define XPAR_AXI_GPIO_1_IS_DUAL 0
```

VIII.2 Revised your Lab #9 Part 1 code to display temperatures on the uart IP terminal

Display two temperature values from two TMP101 on both the standard terminal and the custom UART IP terminal. Use XGPIO driver subroutines to read from and write to the GPIO IPs.

```
u32 XGpio_DiscreteRead(XGpio *InstancePtr, unsigned Channel);
void XGpio_DiscreteWrite(XGpio *InstancePtr, unsigned Channel, u32 data);
```

sprintf() can be used to generate a string array so that this array can be sent to the UART IP terminal.

```
int sprintf (char *str, const char *format, ….)
```

you could create a subroutine to send an ASCII array to the uart IP terminal as follows.

```
XGpio uart1;    //define a custom uart device
XGpio uart_tx_full;

void Print_message(char *Message) {
unsigned char count;
    count=0;
    for(count=0; count<strlen(Message); count++) {
        while(XGpio_DiscreteRead(&uart_tx_full, 1)==1) {};  //wait for buffer to be free
        XGpio_DiscreteWrite(&uart1, 1, Message[count]);
        XGpio_DiscreteWrite(&uart1, 2, 1);
        XGpio_DiscreteWrite(&uart1, 2, 0);
    }
}
```