PmodBT2 Zedboard Tutorial

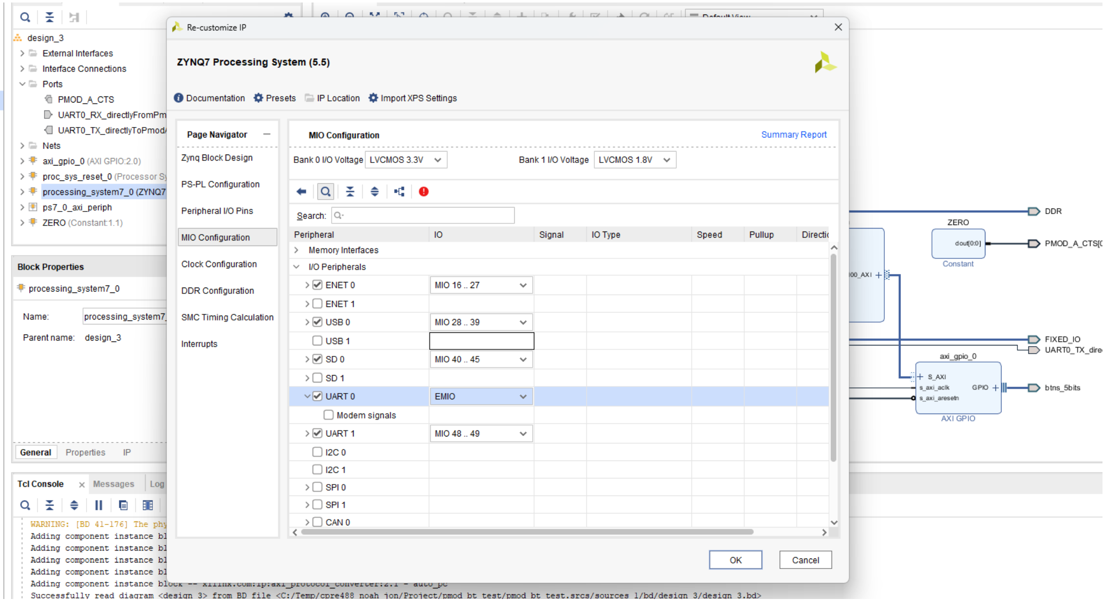
There are three parts to this, the Vivado design, Vitis software, and the hardware.

References:

- PmodBT2's Docs: https://digilent.com/reference/pmod/pmodbt2/reference-manual
- Gregory's Tutorial: https://ctrtl.com/docs/sparky.pdf (Chapter 6)

Vivado

1. In the processing system, enable UART 0 to EMIO



Re-customize IP

ZYNQ7 Processing System (5.5)

Documentation Presets IP Location Import XPS Settings

Page Navigator: Zynq Block Design, PS-PL Configuration, Peripheral I/O Pins, MIO Configuration, Clock Configuration, DDR Configuration, SMC Timing Calculation, Interrupts

MIO Configuration — Summary Report

Bank 0 I/O Voltage: LVCMOS 3.3V   Bank 1 I/O Voltage: LVCMOS 1.8V

Search:

Peripheral	IO	Signal	IO Type	Speed	Pullup	Directi
Memory Interfaces						
I/O Peripherals						
☑ ENET 0	MIO 16 .. 27					
☐ ENET 1						
☑ USB 0	MIO 28 .. 39					
☐ USB 1						
☑ SD 0	MIO 40 .. 45					
☐ SD 1						
☑ UART 0	EMIO					
☐ Modem signals						
☑ UART 1	MIO 48 .. 49					
☐ I2C 0						
☐ I2C 1						
☐ SPI 0						
☐ SPI 1						
☐ CAN 0						

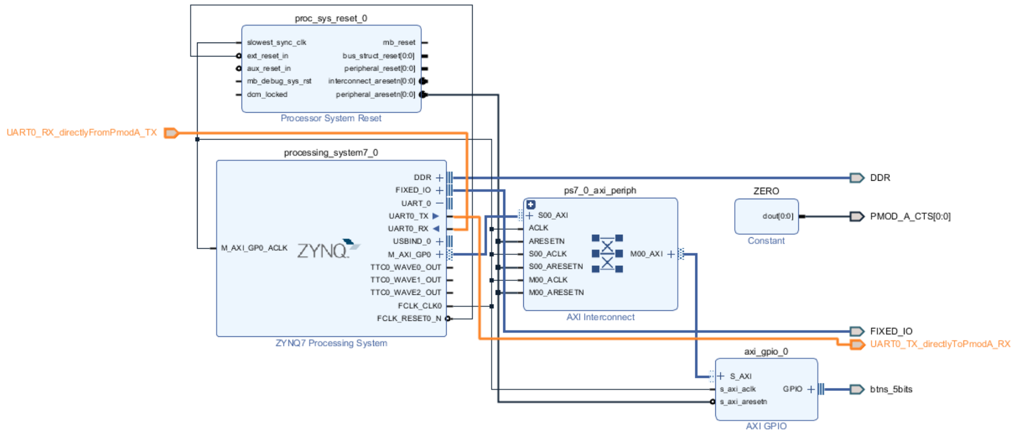
OK Cancel

design_3: External Interfaces, Interface Connections, Ports (PMOD_A_CTS, UART0_RX_directlyFromPm, UART0_TX_directlyToPmod), Nets, axi_gpio_0 (AXI GPIO:2.0), proc_sys_reset_0 (Processor S, processing_system7_0 (ZYNQ7, ps7_0_axi_periph, ZERO (Constant:1.1)

Block Properties: processing_system7_0 — Name: processing_system7_ — Parent name: design_3

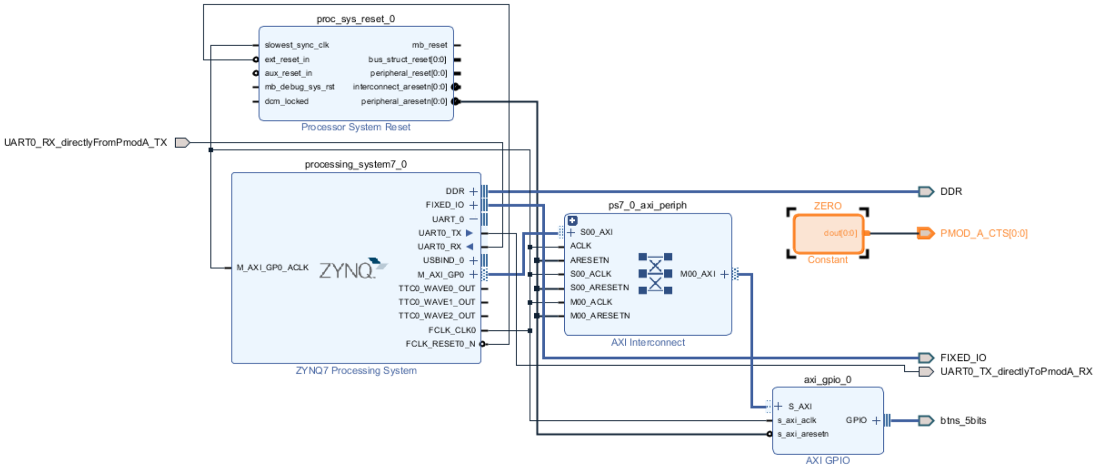
```
WARNING: [BD 41-176] The phy
Adding component instance bl
Adding component instance bl
Adding component instance bl
Adding component instance bl
Adding component instance block -- xilinx.com:ip:axi_protocol_converter:2.1 - auto_pc
Successfully read diagram <design 3> from BD file <C:/Temp/cpre488 noah ion/Project/pmod bt test/pmod bt test.srcs/sources 1/bd/design 3/design 3.bd>
```

2. Hook up UART0's Tx to the PmodBT2's Rx, and UART0's Rx to the PmodBT2's Tx

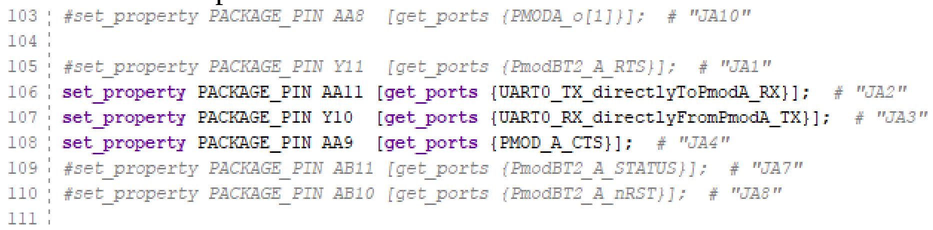


Make sure you hook up UART0's Tx to PmodBT2's Rx and UART0's Rx to PmodBT2's Tx, not UART0's Tx to PmodBT2's Tx and UART0's Rx to PmodBT2's Rx. The Tx and Rx pins can be found in the PmodBT2 docs.

3. Connect the PmodBT2’s CTS to constant zero



4. Now make sure the pins are set in the constrains file:



```
#set_property PACKAGE_PIN AA8  [get_ports {PMODA_o[1]}];  # "JA10"

#set_property PACKAGE_PIN Y11  [get_ports {PmodBT2_A_RTS}];  # "JA1"
set_property PACKAGE_PIN AA11 [get_ports {UART0_TX_directlyToPmodA_RX}];  # "JA2"
set_property PACKAGE_PIN Y10  [get_ports {UART0_RX_directlyFromPmodA_TX}];  # "JA3"
set_property PACKAGE_PIN AA9  [get_ports {PMOD_A_CTS}];  # "JA4"
#set_property PACKAGE_PIN AB11 [get_ports {PmodBT2_A_STATUS}];  # "JA7"
#set_property PACKAGE_PIN AB10 [get_ports {PmodBT2_A_nRST}];  # "JA8"

```

5. Generate bitstream and XSA and you should be good!

Vitis

1. Because we (the Zedboard) are talking to the PmodBT2 using UART (UART 0), and UART 0 is hooked up directly from the PS, we can use the built in UART API to talk to the PmodBT2.
2. Go to “pmodbt2_utils.c” to see the code, pmodBT2_init() is required to initialize the UART0 stuff. To send, checkout pmodBT2_send_and_just_send(), and to receive, checkout pmodBT2_receive_frame().
   Note: These functions are not generalized for all PmodBT2 applications, but is created for our 488 Final Project, but the idea is there.
3. Sender.c is the sender code, receiver.c is the receiver code. Which one is run is determined by the FLASHING_SENDER and FLASHING_RECEIVER definition in utils.h.
4. If you want, my code is based on the “Peripheral Tests” project.

Hardware

1. This is a tldr for Gregory’s docs and the RN42 chip’s docs (the Bluetooth chip) on the PmodBT2.
2. There are two modes: Command mode and Normal mode. Command mode is a pain, so we are using normal mode, which means that when two PmodBT2 is paired, they act like a UART wire.
3. To factory reset the PmodBT2:

   a. Short JP1, plug it into a Pmod port on the Zedboard, the green LED should blink rapidly for a quick second.
   b. Remove and reshort JP1 three times, the LEDs will go crazy for a bit, that’s when you know you did it right.
   c. Unplug the Pmod.
4. Pick one PmodBT2 to be the master, before powering it on, make sure JP2 and JP3 are shorted.
5. Pick another PmodBT2 to be the slave, before powering it on, make sure only JP2 is shorted.
6. Power both PmodBT2s.
7. The green LED should blink a bit and turn off (only red LED remain on) if connected. The LED pattern can be found here in 3.7.2 in the RN42 docs.

Sending and receiving

The code attached to this doc is proven to work with two Zedboards with a PmodBT2 plugged into the PMOD A slot. When a BTN C of the sender Zedboard is pressed, it will send a short string of characters starting with “SOF”, e.g. “SOF#######”. The receiver Zedboard has a polling function to wait to receive “SOF…”.

The code is just a demo.