LAB 2

HOME ASSIGNMENT

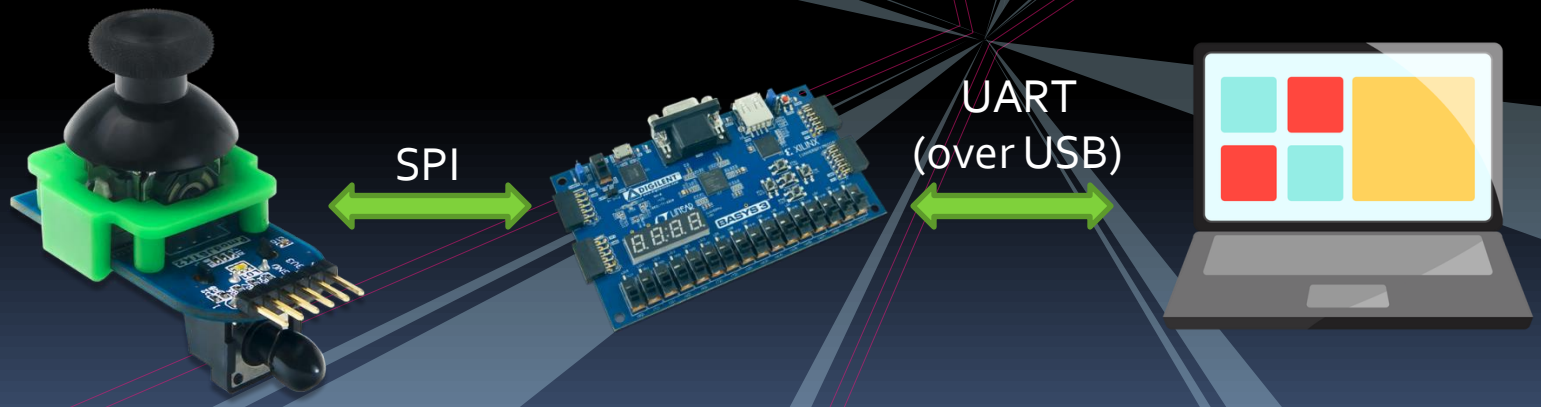
Home assignment Goal

At the beginning of LAB2, we will give each one of you a Digilent Pmod JSTK2 module.

This can be connected to your Basys3 board through the Pmod connectors.

Home assignment Goal

The objective of the LAB2 home assignment is to receive data and control the module's LEDs from the PC, using the provided AXI4-Stream UART module.

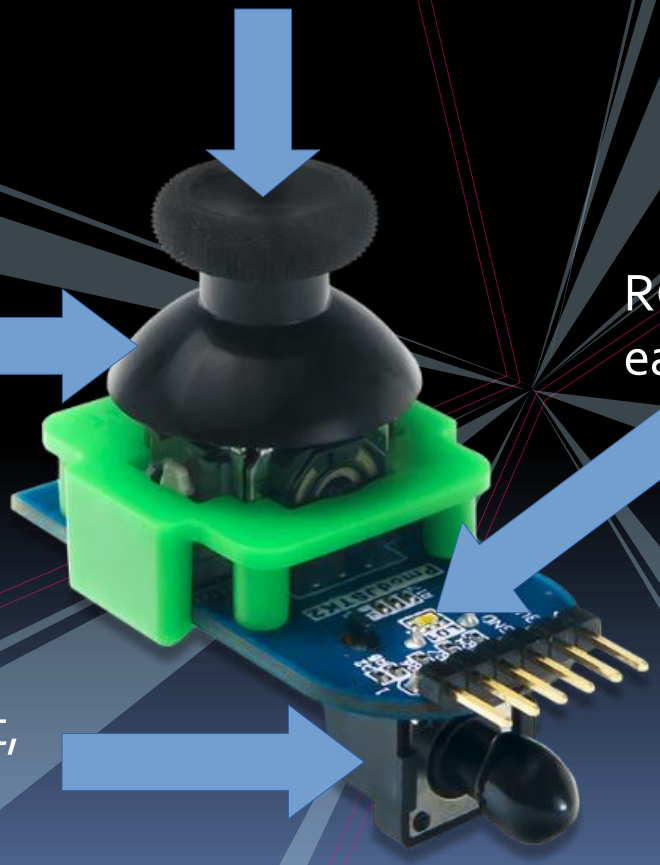


Digilent Pmod JSTK2 components

Joystick button (1 bit, pressed/not-pressed)

Joystick position (10 bits X-axis, 10 bits Y-axis)

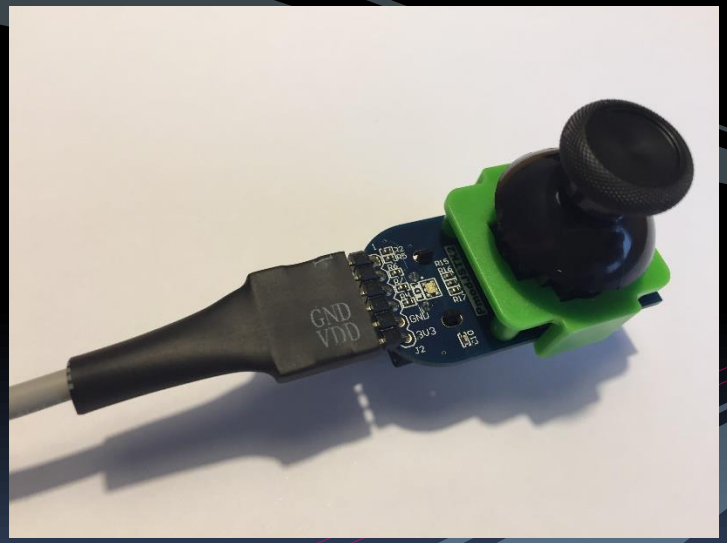
RGB LED (8 bits for each channel)

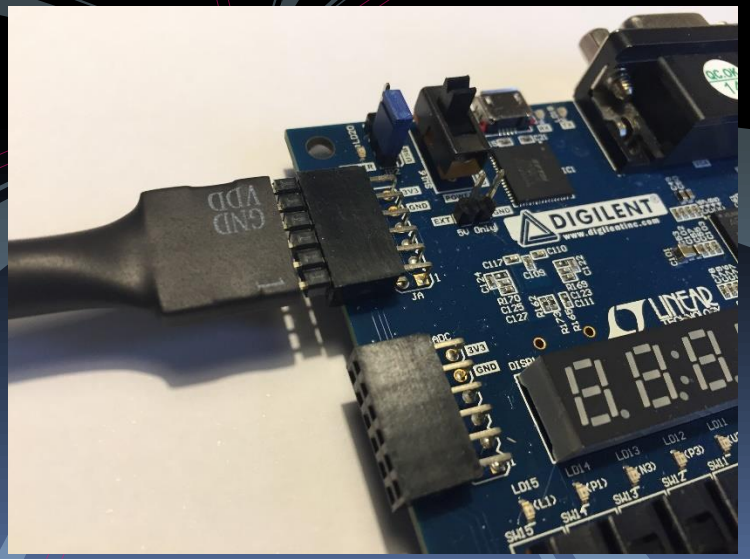


"Trigger" button (1 bit, pressed/not-pressed)

How to connect the Joystick

Connect the Joystick with the provided cable to "JA", top row, paying attention to the VCC and GND position.

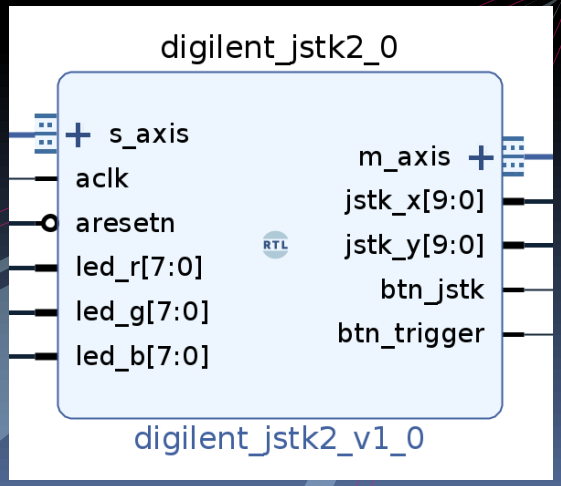




Home assignment Goal #1

You will have to build 2 modules:

- digilent_jstk2, to "talk" to the module and expose its "state" (jstk_x, jstk_y, jstk_btn, trigger_btn) and "controls" (led_r, led_g, led_b) as std_logic or std_logic_vector.



JSTK2 interfacing

The Digilent Pmod JSTK2 module protocol is fully described in its reference manual.

In short, it uses the SPI protocol to receive "commands" and send back the "readings" of the Joystick position and the buttons state.

SPI protocol

The Serial Peripheral Interface (SPI) is a very popular synchronous protocol for off-chip communication.

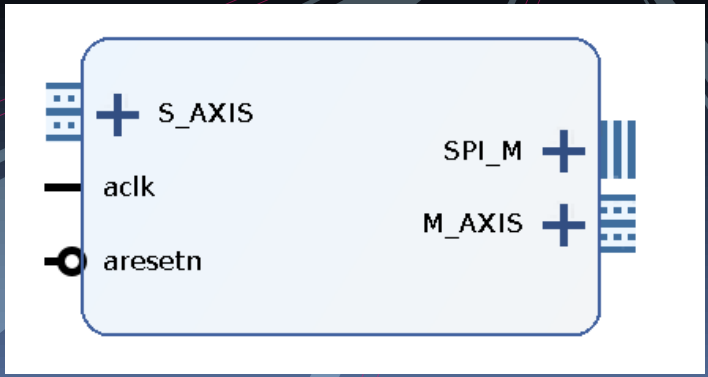
In its basic form it is composed by 4 signals:

- Serial CLocK (SCLK)
- Master-Out Slave-In (MOSI)
- Master-In Slave-Out (MISO)
- Chip Select (CS)

SPI IP-Core

While SPI is a simple protocol, describing it in VHDL is not immediate.

To ease your work, we will give you a "AXI4-Stream SPI" IP-Core, similar to the UART one.



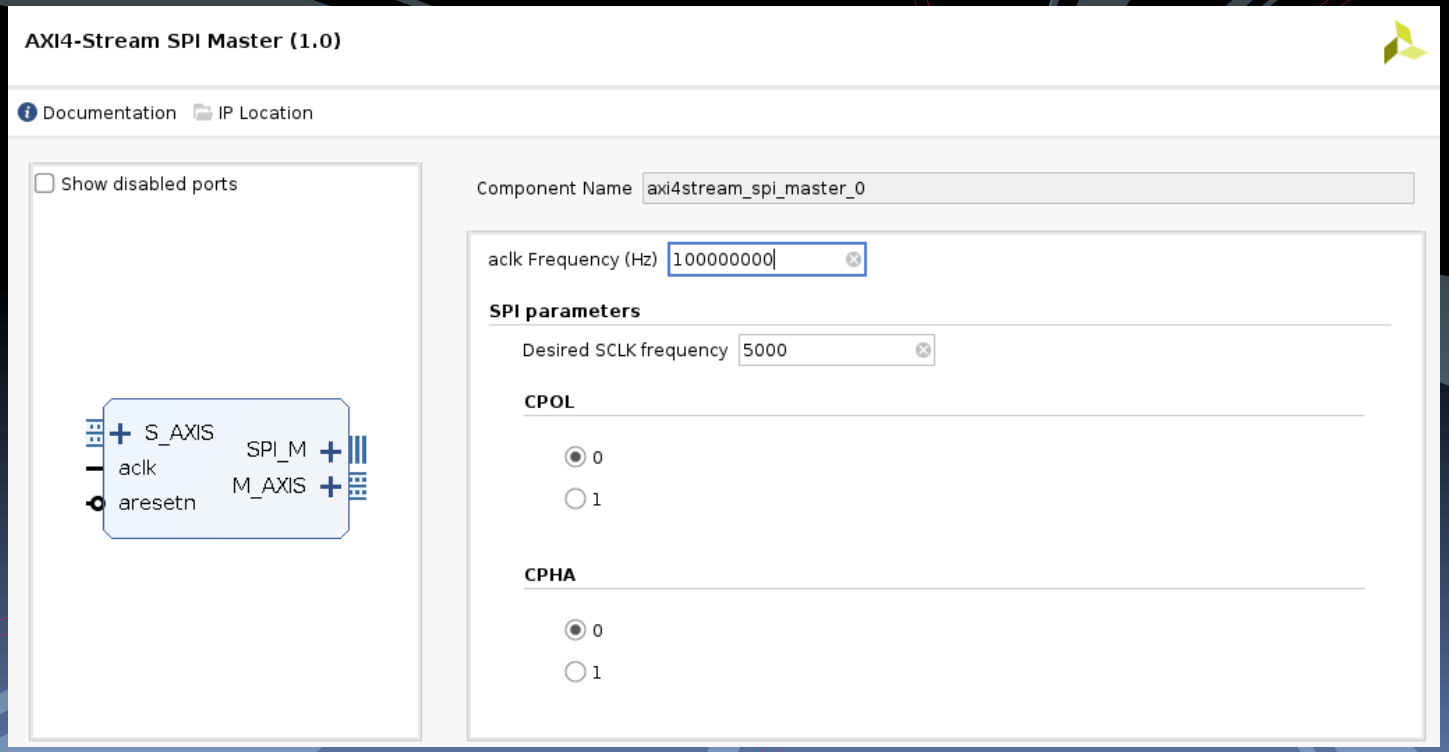
SPI IP-Core

You will find the details of its behavior in the README file. In short:

- Whatever you send to S_AXIS is sent to the SPI "slave".
- Whatever is received from the SPI "slave" is sent to you through the M_AXIS interface.

SPI IP-Core

To respect the timing of the Digilent Pmod JSTK2 module (see here), use these parameters in the SPI IP-Core.



JSTK2 commands

Using this IP-Core, you can easily "talk" to the module, following the protocol described in the reference manual.

We suggest you to use the cmdSetLedRGB command so that, in a single command, you can get the position of the joystick and the state of the button, and control the LED color.

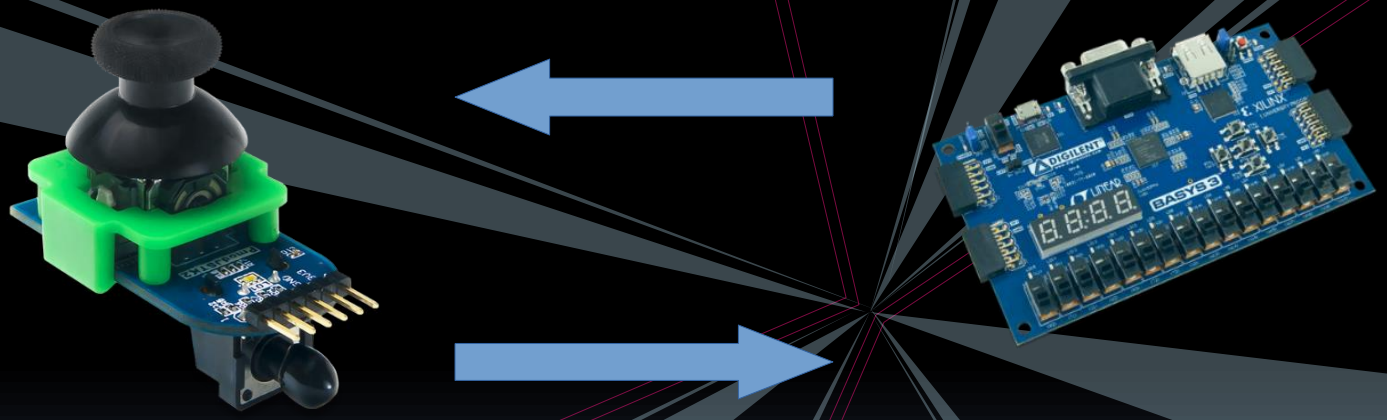
JSTK2 commands - cmdSetLedRGB

		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
cmdSetLedRGB	**(0x84)**	1	0	0	0	0	1	0	0

Parameters

PARAM1 – Red LED duty cycle
PARAM2 – Green LED duty cycle
PARAM3 – Blue LED duty cycle
PARAM4 – ignored

Set the duty cycles for the Red, Green, and Blue LEDs.



	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
MOSI	COMMAND / 0	PARAM1 / DUMMY	PARAM2 / DUMMY	PARAM3 / DUMMY	PARAM4 / DUMMY
MISO	smpX (Low Byte)	smpX (High Byte)	smpY (Low Byte)	smpY (High Byte)	fsButtons

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
fsButtons	EXTPKT	0	0	0	0	0	TRIGGER	JOYSTICK

EXTPKT: Extended Packet Status Bit
1 = additional data corresponding to the command byte is available and may be retrieved after this byte
0 = standard response packet, no additional data follows this byte

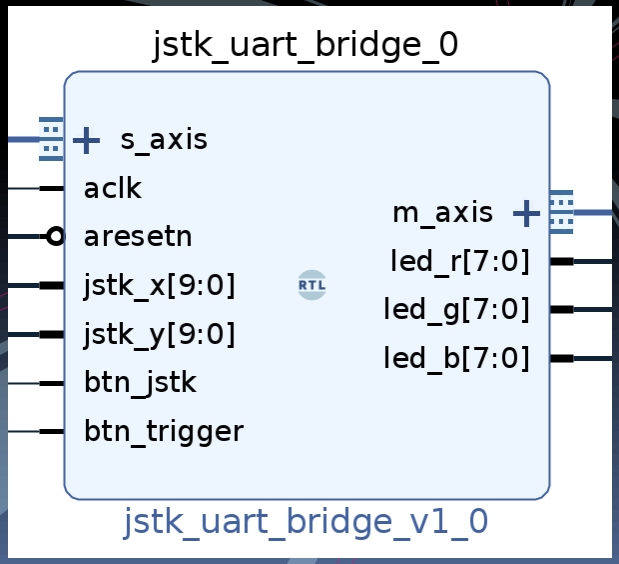
TRIGGER: Trigger Button Status Bit
1 = trigger button is currently pressed
0 = trigger button is not being pressed

JOYSTICK: Joystick Center Button Status Bit
1 = joystick center button is currently pressed
0 = joystick center button is not being pressed

Home assignment Goal #2

Now that we can "talk" to the JSTK2 module, we have to send those values to the PC with:

- jstk_uart_bridge, to send and receive those values to the UART module.



Home assignment Goal #2

You have to manage data to and from the PC:

- PC -> FPGA: RGB LED control
- FPGA -> PC: position of joystick axis, joystick button and trigger button. This should be sent periodically.

As we have to send more than just a single byte, we have to build a "packet", so that the receiver can correctly understand which value it is receiving.

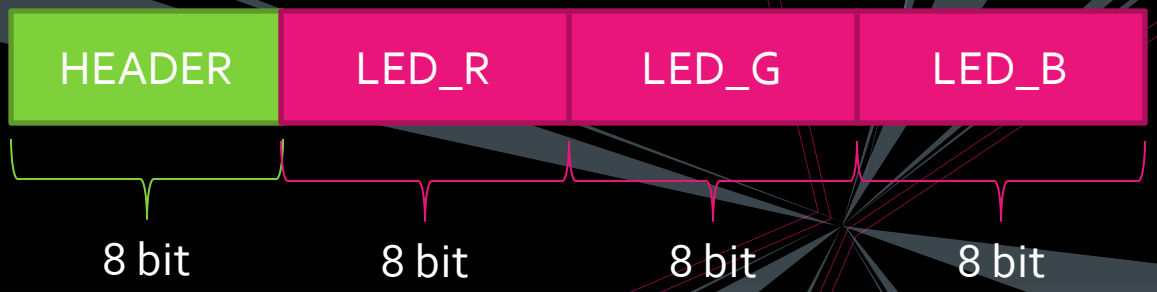
Packet specification

All packets, in both directions, must have this stucture:

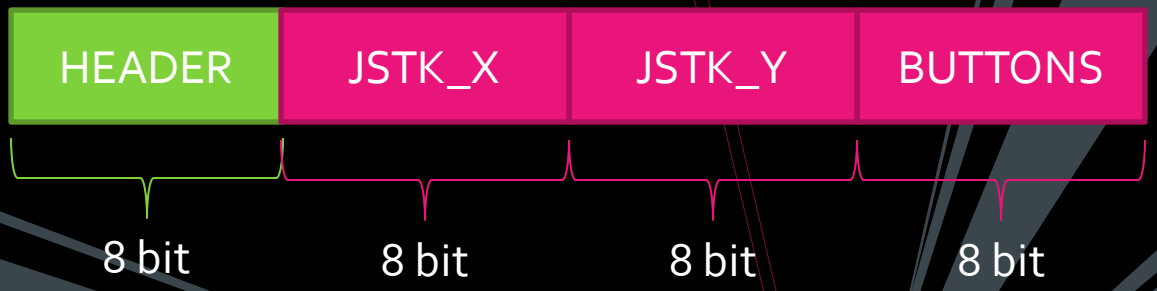
A header let the receiver understand when a new packet begins.

In our case, we choose HEADER = 0xC0.

PC → FPGA: LEDs control



FPGA → PC: Joystick and buttons



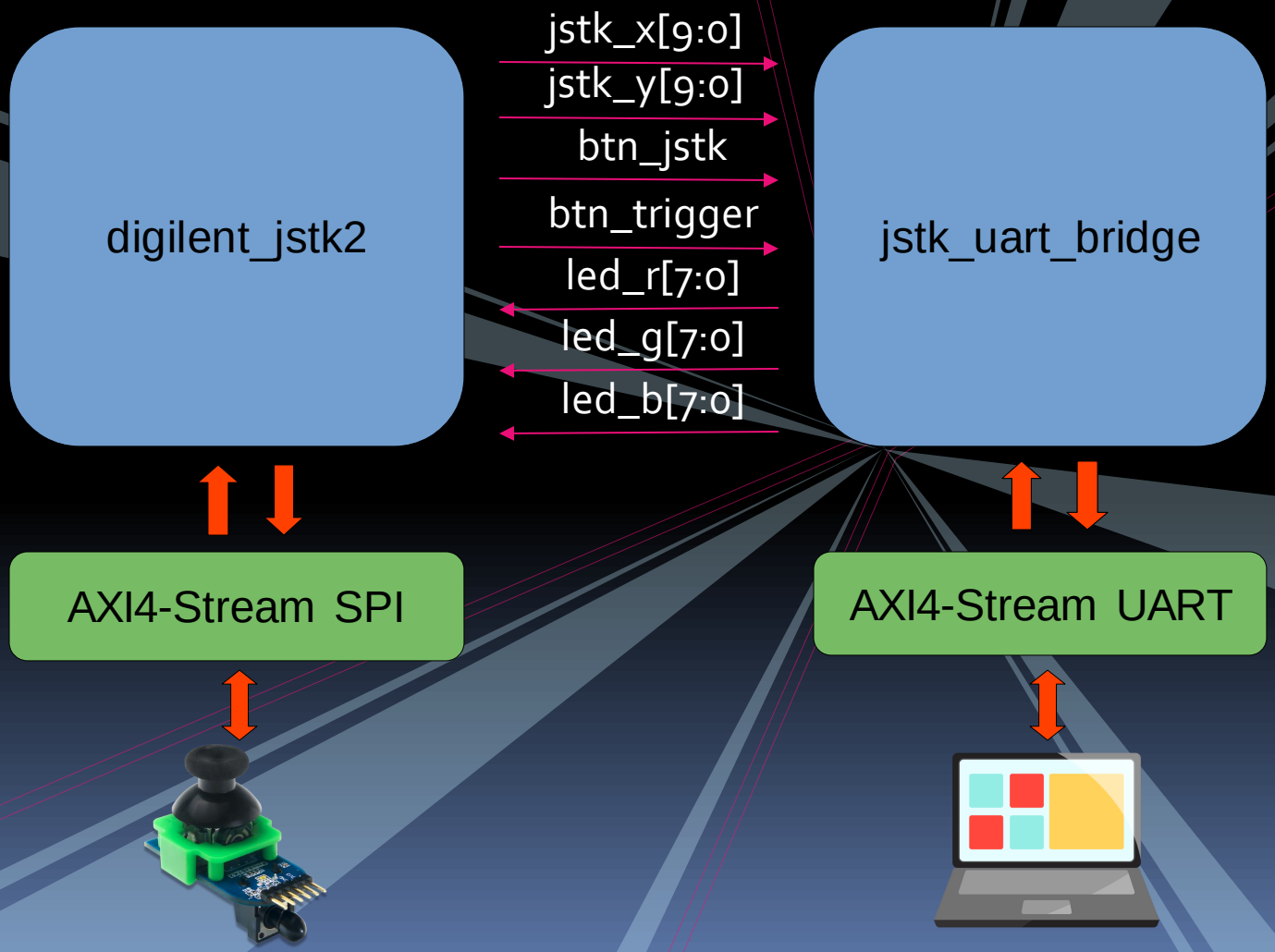
JSTK_X and JSTK_Y are the eight MSbits of the data provided by the digilent_jstk2 module.

"BUTTONS" must have this structure:

- BUTTONS(7 downto 2) <= '0'
- BUTTONS(1) <= btn_trigger
- BUTTONS(0) <= btn_jstk

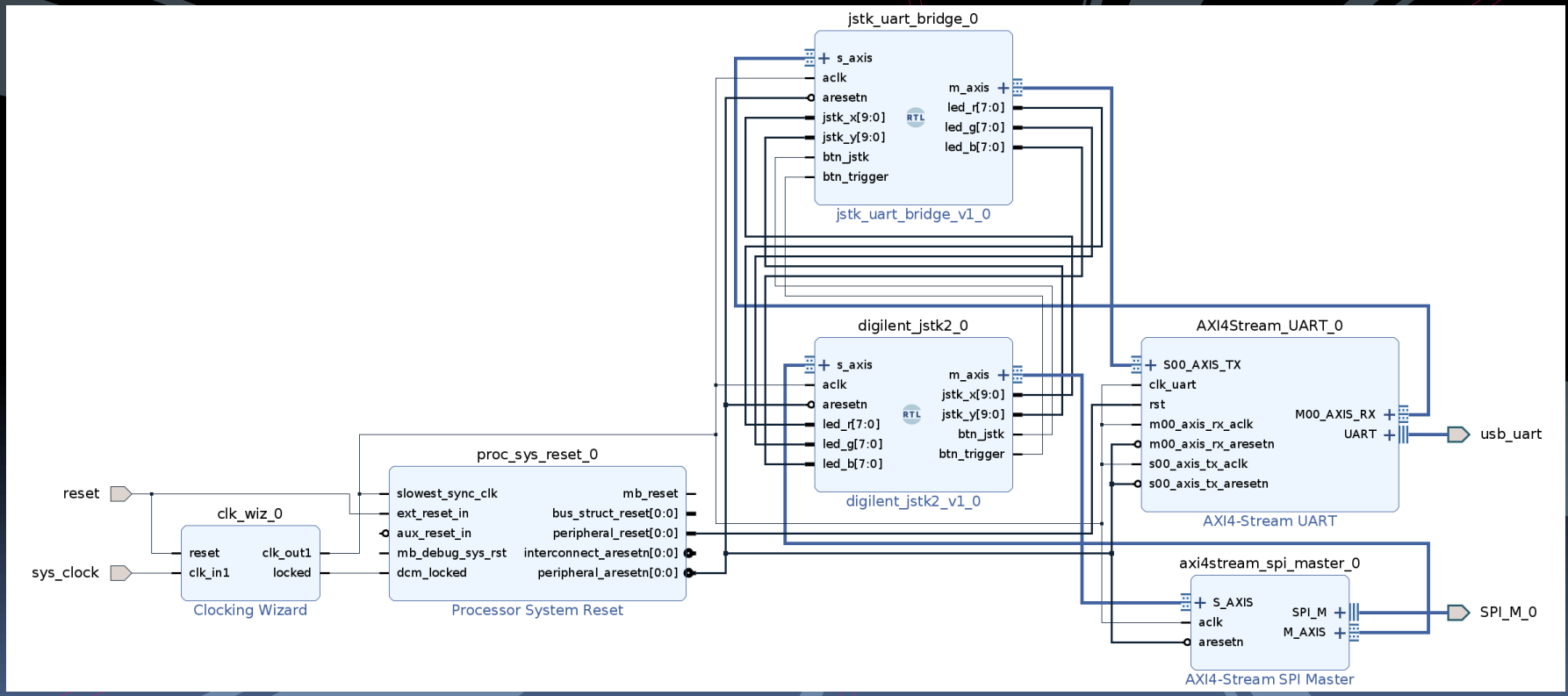
Overall structure

At the end, your project should have this structure:



Overall structure – Block Design

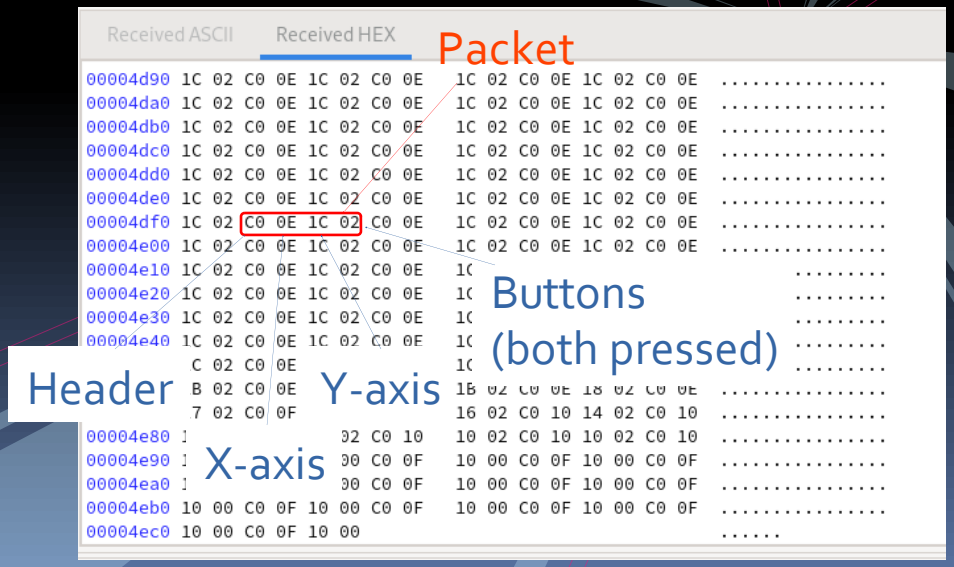
At the end, your project should have this structure:



Testing: FPGA → PC

As usual, the easiest way to test your design on hardware is by using a serial terminal.

Open it and see if you receive the expected data when you move the joystick and press the buttons:



Testing: PC → FPGA

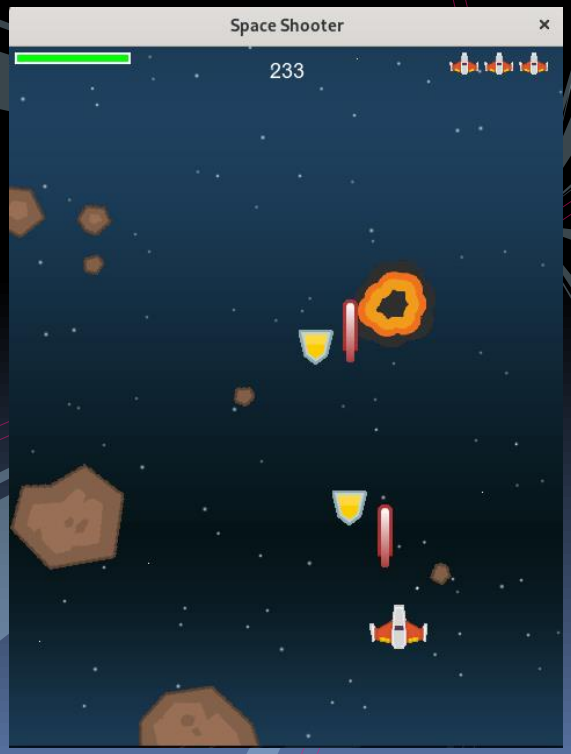
The same for the PC → FPGA direction: send some data in the proper format and see if the LED changes color.

For example:

- C0 ff ff ff → white
- C0 00 00 00 → OFF
- C0 ff 00 00 → red
- C0 ff ff 00 → yellow
- C0 10 00 10 → light purple

Even better testing

Of course, having a joystick connected to the PC and controlling it only through the serial would be quite boring…



Even better testing

You will find the “spaceShooter” game in the LAB2 material, modified to communicate through the serial port with your modules. You will be able to control the spaceship with the joystick, shoot with the trigger button and pause the game with the joystick button.

The LEDs on the board will show your life (blue for high life, yellow for mid, red for low).