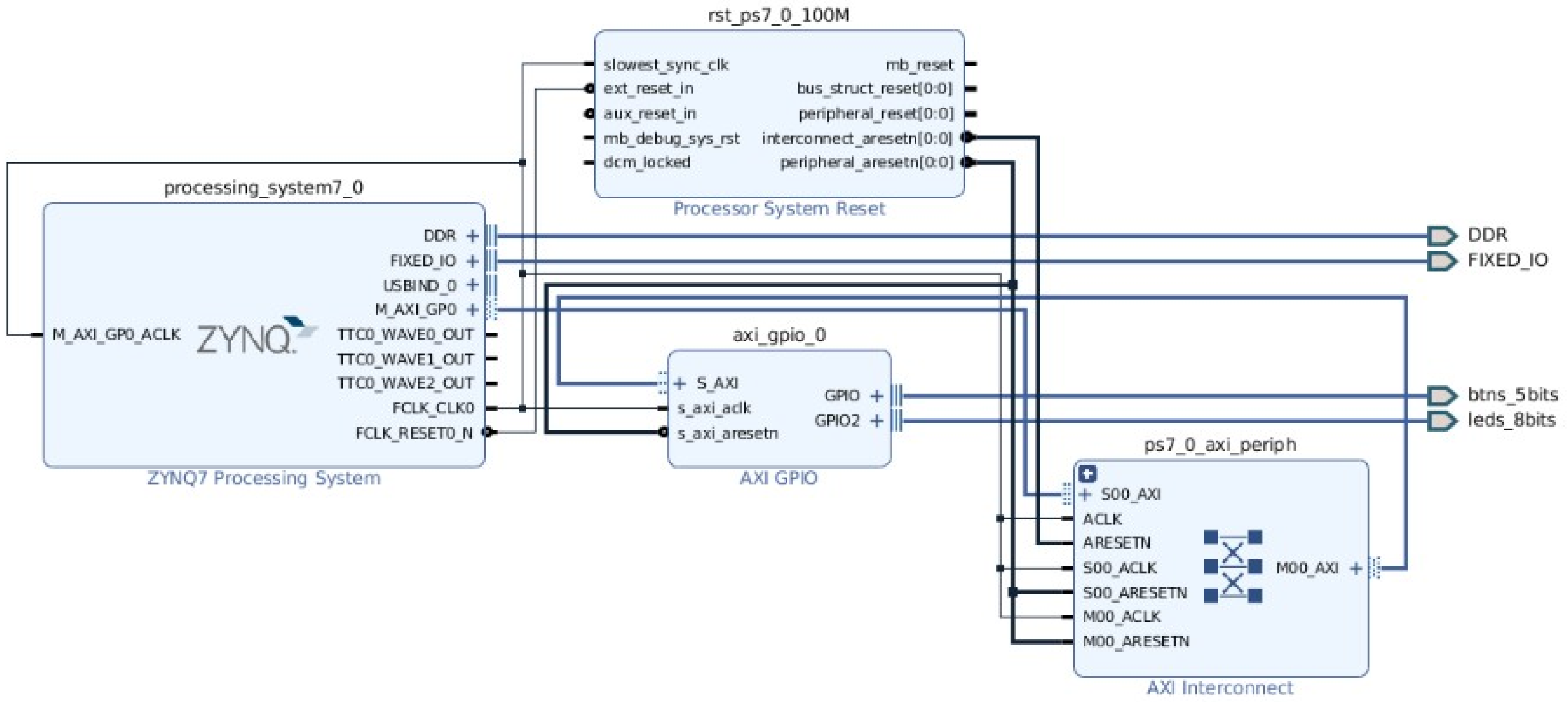
Exercises FreeRTOS

Exercise 1:
Create a Design in Vivado using a Zynq7_PS and a AXI_GPIO configured with btns_5bits and leds_8bits for GPIO0 and GPIO2 respectively.
The design should be something like this



Exercises FreeRTOS

Exercise 1:

Now that the design is complete, run PROGRAM & DEBUG→ Generate Bitstream to complete the back-end flow.

Once finished we just need to Export→Export Hardware (also selecting include bitstream) to generate the necessary files for the software development part of the project using the SDK

Once the hardware platform has been generated, the rest of the development will continue in the SDK environment. You can launch SDK directly from the Vivado tool by executing File→Launch SDK.

Exercises FreeRTOS

Exercise 1:

Once the platform specification is available, it is the time to generate the FreeRTOS Board Support Package, upon which the FreeRTOS applications will later be developed.
The BSP will provide the FreeRTOS run-time, as well as the Application Programming Interface (API) that the applications require to make of the services and resources provided by the Operating System:

1. Click on File→New→Board Support Package and select **freertos10_xilinx** in the Board Support Package OS entry in the bottom left side of the resulting window. The 10 in the name refers to the FreeRTOS version, and may vary depending on the Vivado Tools version installed.

Finally, click on Finish to generate the BSP.

Exercises FreeRTOS

Exercise 1:

2- Create Appliction Project

File→New→Application Project.

a. Project Name: led_buttons
b. OS platform: freertos10_xilinx
c. board support package: use existing freertos10_xilinx_bsp_0. The one that we just generated in the previous step.
d. Now click on Next to select the type of initial project to generate from a list of Templates. Here choose the FreeRTOS Hello World application.
e. By clicking on Finish the application will be generated.

Exercises FreeRTOS

Exercise 1:

- Open the freeRtos_hello_world file
- Identify the main function
- Identify the taskCreate functions
  - How many tasks are being created?
  - Which is the purpose of each task?
  - Which task has the highest priority?
  - Which is the purpose of the timer?
  - How does the application finish message transmission?

- Identify the function xQueueCreate
  - How many elements can be stored in the queue?

Exercises FreeRTOS

Exercise 1:

3- Run the project to check that the hello world application is working properly

a - Configure the SDK Terminal (find the tab in the lower part of the SDK), clicking on the "+" icon and choosing the right port and baudrate (115200 by default). This is necessary since the standard input and output of the project are mapped to the PS UART, and therefore we would need a serial terminal in the PC to have access to them.

b- Select the application project (led_buttons) on the left side of the SDK (Project Explorer)

c- Run selecting Run as - Launch on Hardware (GDB)

Exercises FreeRTOS

Exercise 2:

- Invert the priorities of the task
  - What should happen?

    Run the code and verify

- Add a vTaskDelay(x1second) to the RxTask to force it to go to the Blocked State
  - What should happen now?

    Run the code and verify

- Put the same prority to both tasks and run the code.

Exercises FreeRTOS

Exercise 3:

- Change the generated code to perform the following behaviour:

  - Every time the Tx task is enabled, it will read the status of the 5 buttons in the board. The resulting value will be send to the queue used for task communication.

  - Every time the Rx task is enabled, it will pop a new value from the queue and display it though the activation of the appropriate leds. Since there are only 5 buttons available in the board, only 5 leds will be really used for the display.

  - This behavior should be keep indefinitely.

Exercises FreeRTOS

Exercise 3:

- Include the reference to the GPIO driver library at the top of the file: #include "xgpio.h"
- declare the variable gpio of type XGpio. This variable will held the reference to the GPIO driver. We need the declaration to be a global one, so insert the following text just before the main function:

```
XGpio gpio;
```

- Get into the main function and insert the following code to configurethe GPIO peripheral: Note that the gpio variable is always passed by reference (&).

```
XGpio_Initialize(&gpio, XPAR_AXI_GPIO_0_DEVICE_ID);
XGpio_SetDataDirection(&gpio, 1, 0xff);
XGpio_SetDataDirection(&gpio, 2, 0x00);
```

Exercises FreeRTOS

Exercise 3:

- Modify the xqueue to be created

```
xQueue = xQueueCreate( 1, sizeof( unsigned int ) );
```

- Let's switch to the **prvTxTask function**. Here, instead of a fixed string, we'll send the value obtained from reading the current status of the buttons. To do so:

1. include the following declaration before the for clause: unsigned int value;

2. read the status of the buttons (inside the for clause):

Notice how value is again passed by reference.

```
value = XGpio_DiscreteRead(&gpio, 1);
```

3. and replace the xQueueSend call with this one:

```
xQueueSend( xQueue,
&value,
0UL );
```

Exercises FreeRTOS

Exercise 3:

- In the **prvRxTask** we should do the complimentary changes:
    1. include the declaration of value:

```
unsigned int value;
```

    2. replace the xQueueReceive call with this one:

```
xQueueReceive( xQueue, &value, portMAX_DELAY );
```

and finally write the value into the leds

```
XGpio_DiscreteWrite(&gpio, 2, value);
```

Exercises FreeRTOS

Exercise 3:

- Now remove the **xTimerCreate** function call in the main function, as well as the assert and xTimerStart function that immediately follow.

- Remove the **vTimerCallBack** function at the end of the source code file.

  Run the application again after build the project

  You should observe that the response from the board is not immediate, but there is some delay from the button has been pressed until the value is reflected on the leds. Even the leds are not immediately turn off after releasing the button. Why do you thing this is happening?

  How can you fix such high latency between the pressing and the display?