EG3205 Part II. Multiprocessor and Multicore Systems

Laboratory Exercise 5

Contents

Lab 5 Introduction

This lab exercise aims to introduce you to a dual-core design based on the Nios II soft processor core, which you will serve as a stepping stone for your design work in Assignment 2.

The purpose of Lab 5 is to create a dual-core processor as the slave for a two-node distributed system similar to the one made in Lab 4. The two nodes will be connected to a CAN bus, implemented through the use of CAN-SPI modules.

Lab 5 Description

In the dual-core processor node, one core, which is referred to as Event Triggered (ET) core, is able to handle event-triggered tasks. The other core, which is referred to as Time Triggered (TT) core, is able to handle time-triggered tasks.

Inter-processor communication is achieved via a shared message buffer and the access of the two cores to the buffer is ensured by a mutual exclusion (mutex) IP core. The details on the hardware configuration are given below in "Creating your Qsys Design"

The ET core of the dual-processor node will be connected to the CAN bus as a slave. The other node on the CAN bus, which will act as a master, will be assumed to generate "bursts" of data at random time intervals. Your ET core will receive and process these data.

The dual-core processor design is intended for use in safety-critical embedded applications.

Task 1

To begin with, you need to develop your Qsys design that we will be using in this Lab as illustrated in Fig. 1:

- The Nios II processor configuration should be designed as illustrated in section "Creating your Qsys Design" below which provides a step-by-step guide to the design process.
- The boards must be connected via the CAN-SPI modules as you have previously done

Once completed move on to Task 2.

Task 2

Your **second task** is to analyse the provided code and understand:

- the code supporting the mutex IP core;
- the functions performed by the ET and TT schedulers in making the slave application run;
- the code related to the CAN controller MCP2515 on the CAN-SPI board.

Upload a brief report with a summary of your findings to Blackboard.

Files provided

```
ET_slave_app.zip
TT_slave_app.zip
```

Acknowledgements:

The first version of this exercise was created by Dr Keith Athaide (March 2012).
The second version of this exercise was developed by Dr. M. Fayyaz (June 2015).
This version of the exercise was developed by Sam Kennedy (November 2018).

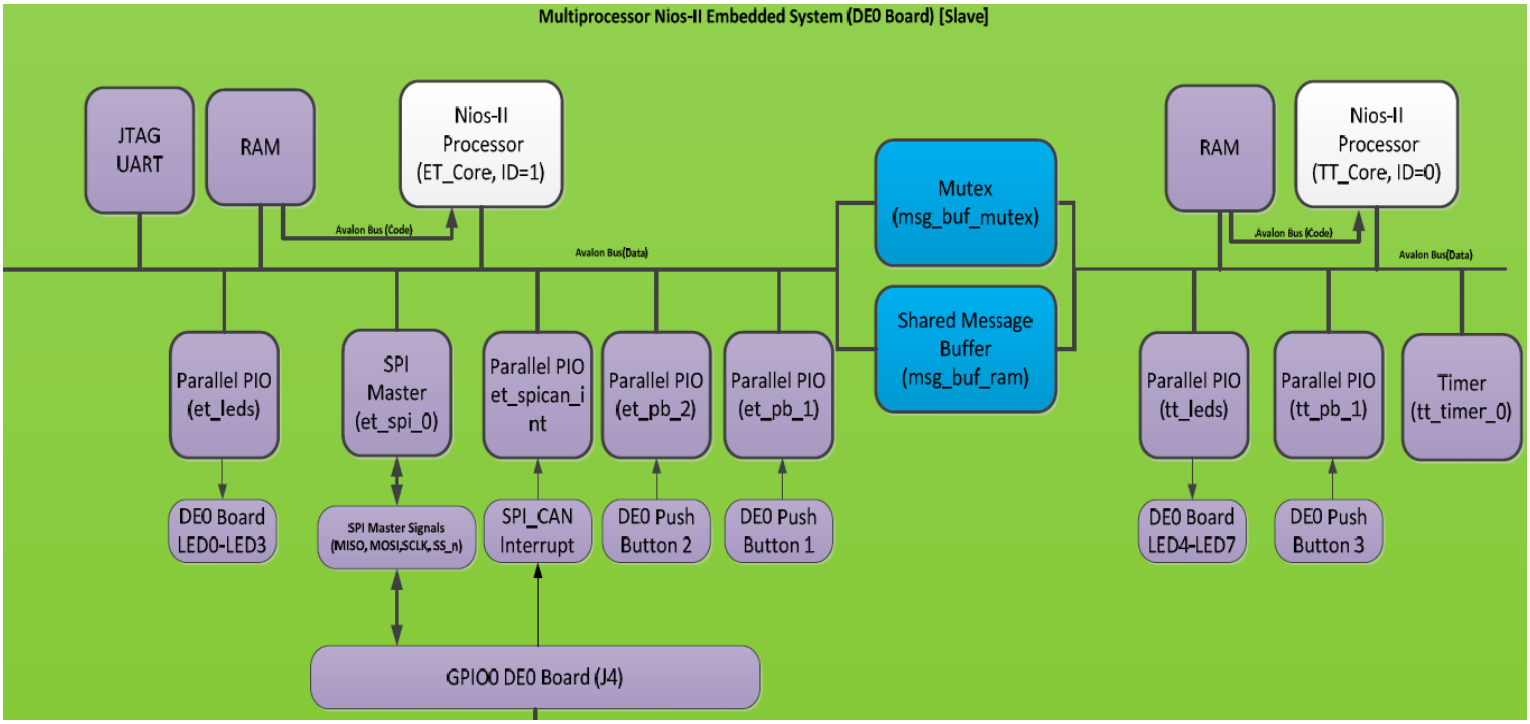
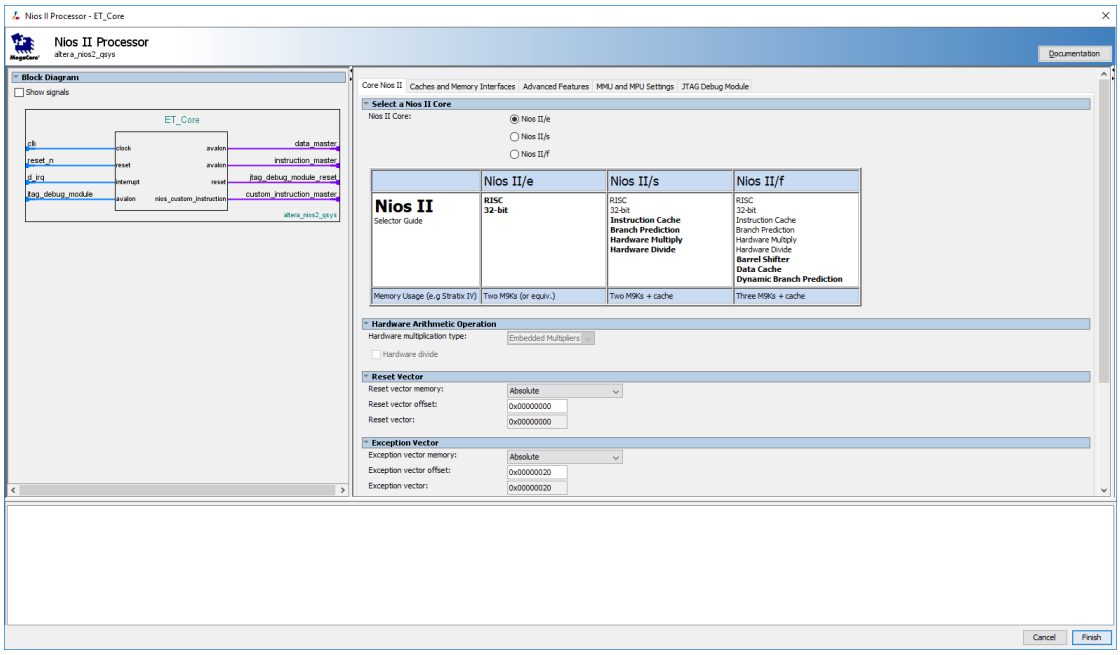


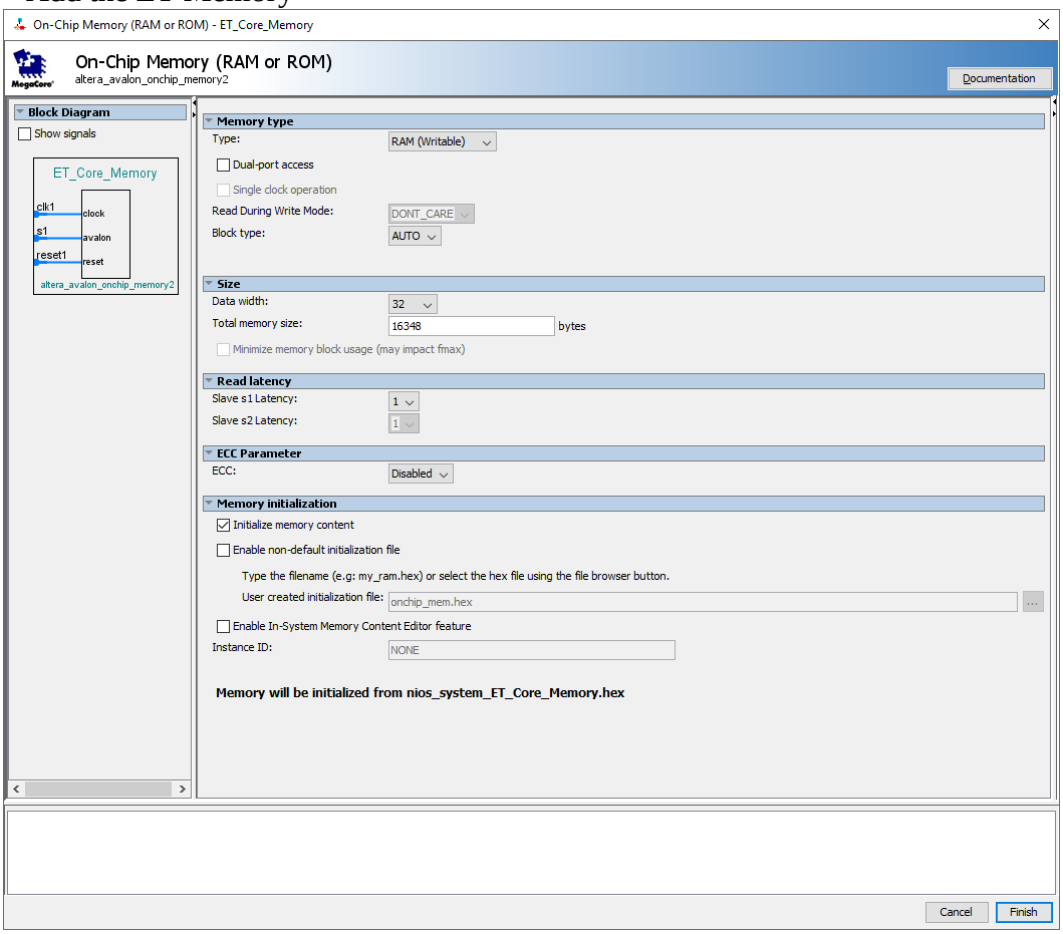
Figure 1. A dual-core design based on the Nios II soft processor core serving as a slave node in a multi-processor system.

Creating your Qsys Design

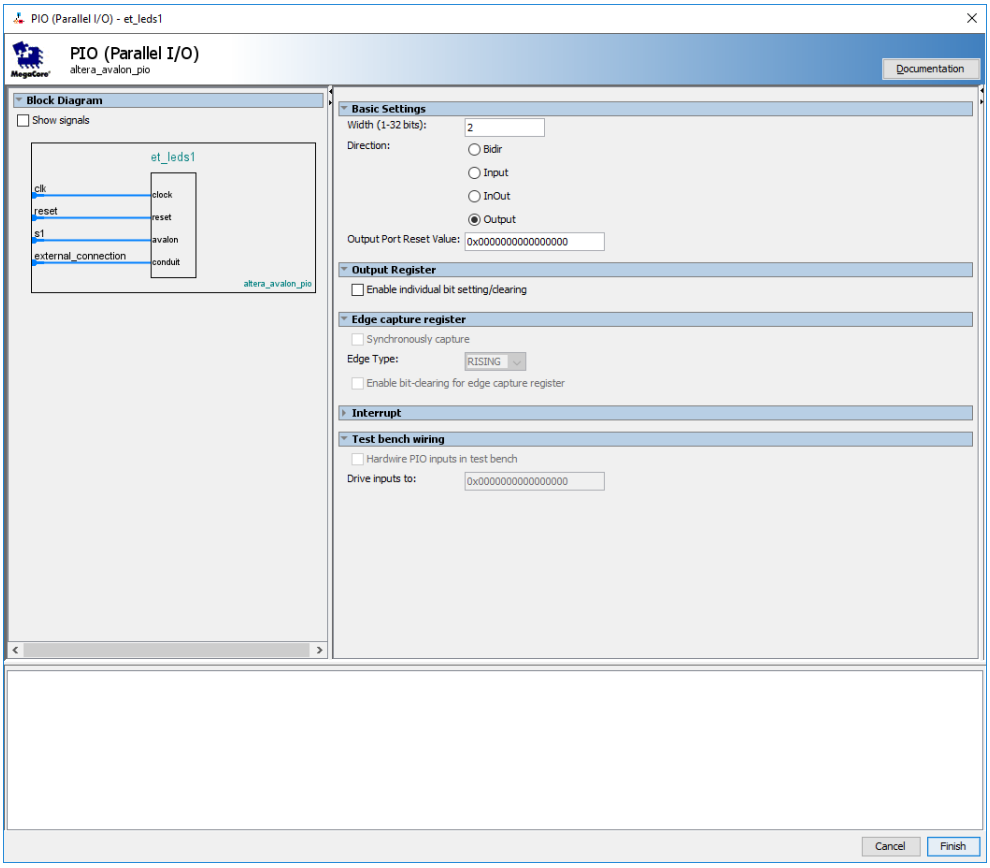
1. Add the Nios Processor for the ET core



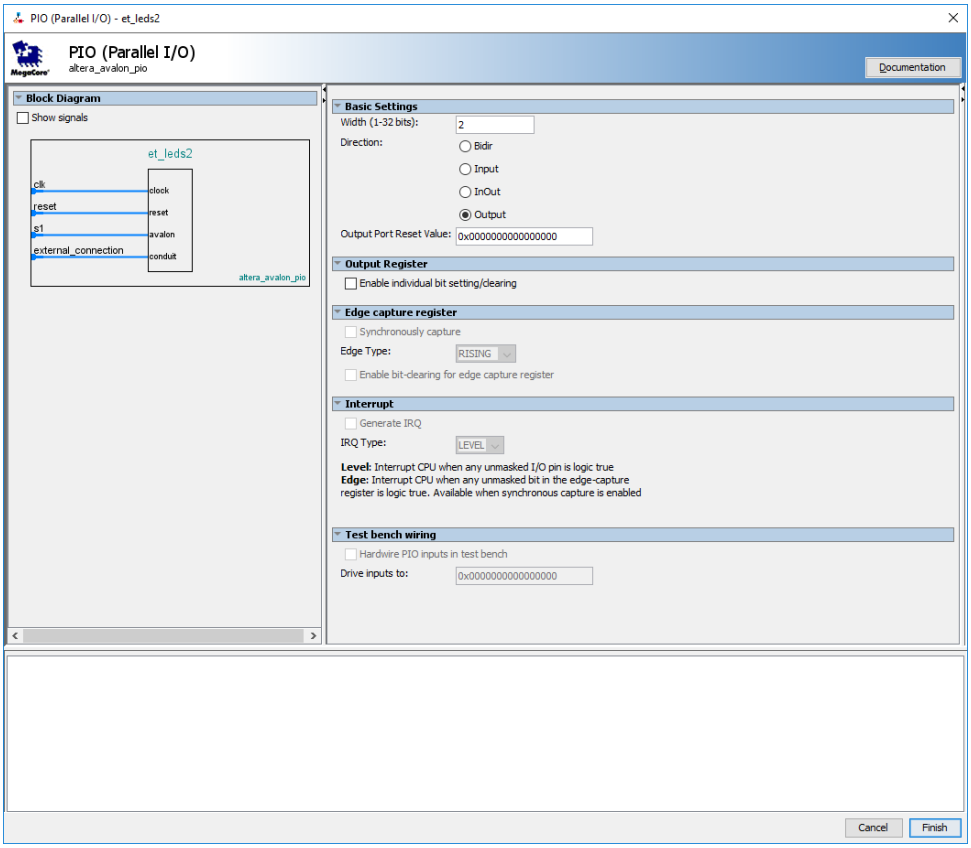
2. Add the ET Memory



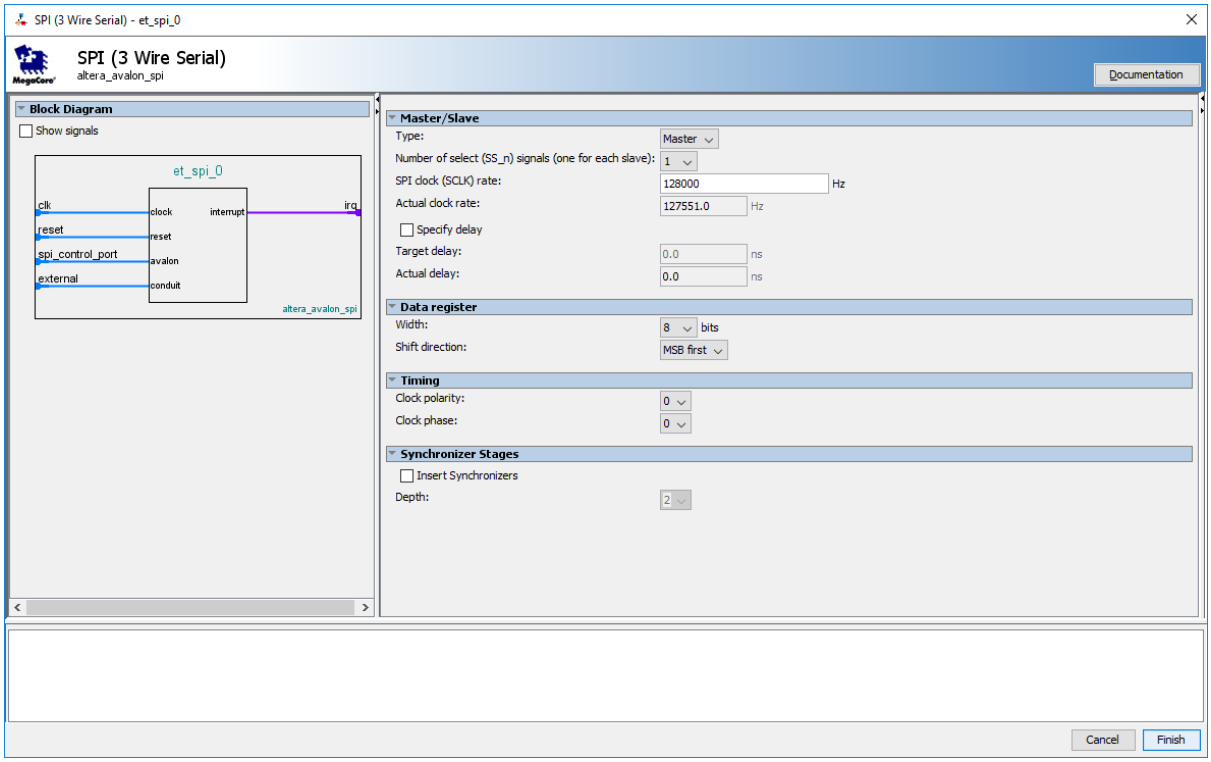
3. Add the first ET LED PIO



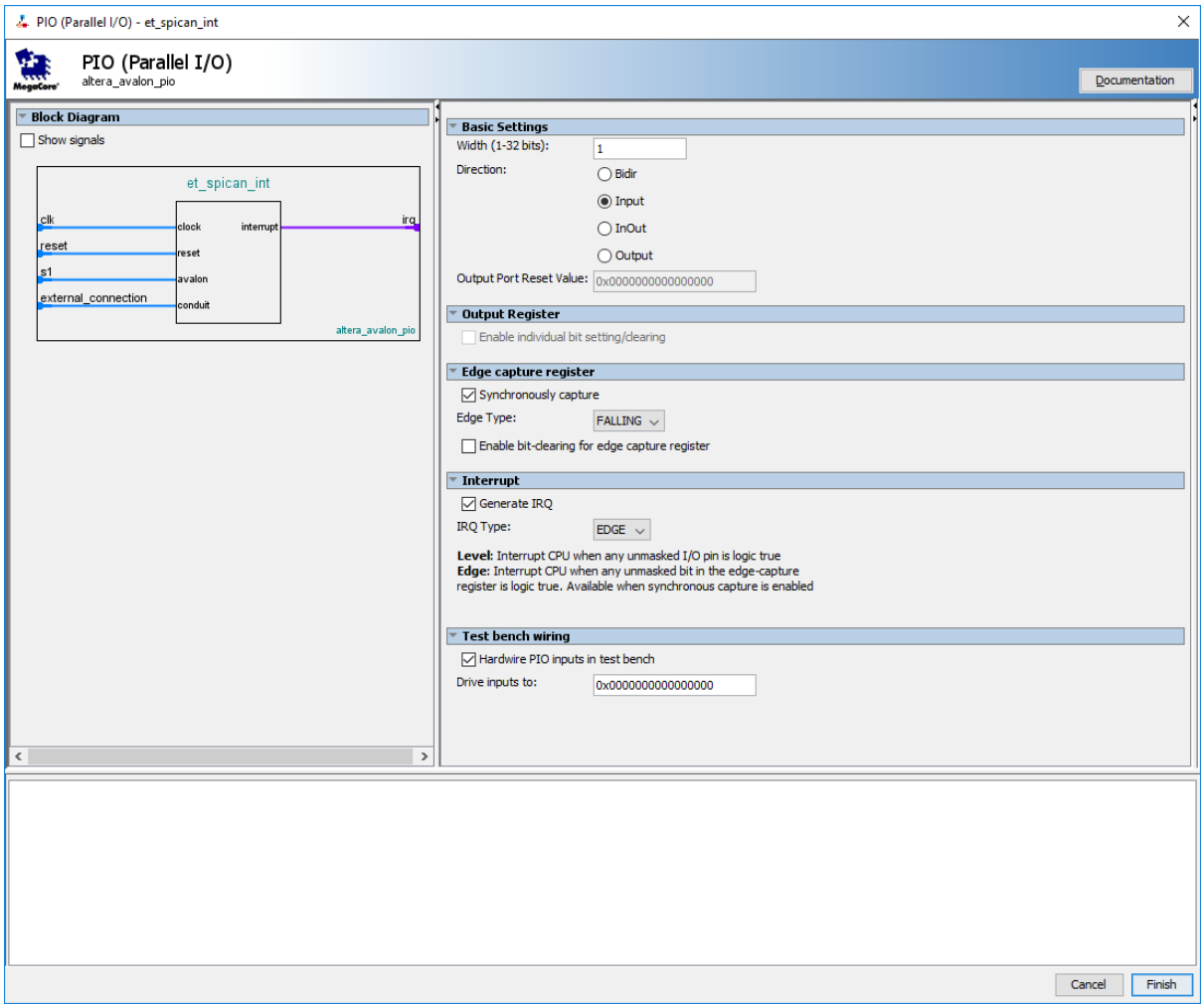
4. Add the second ET LED PIO



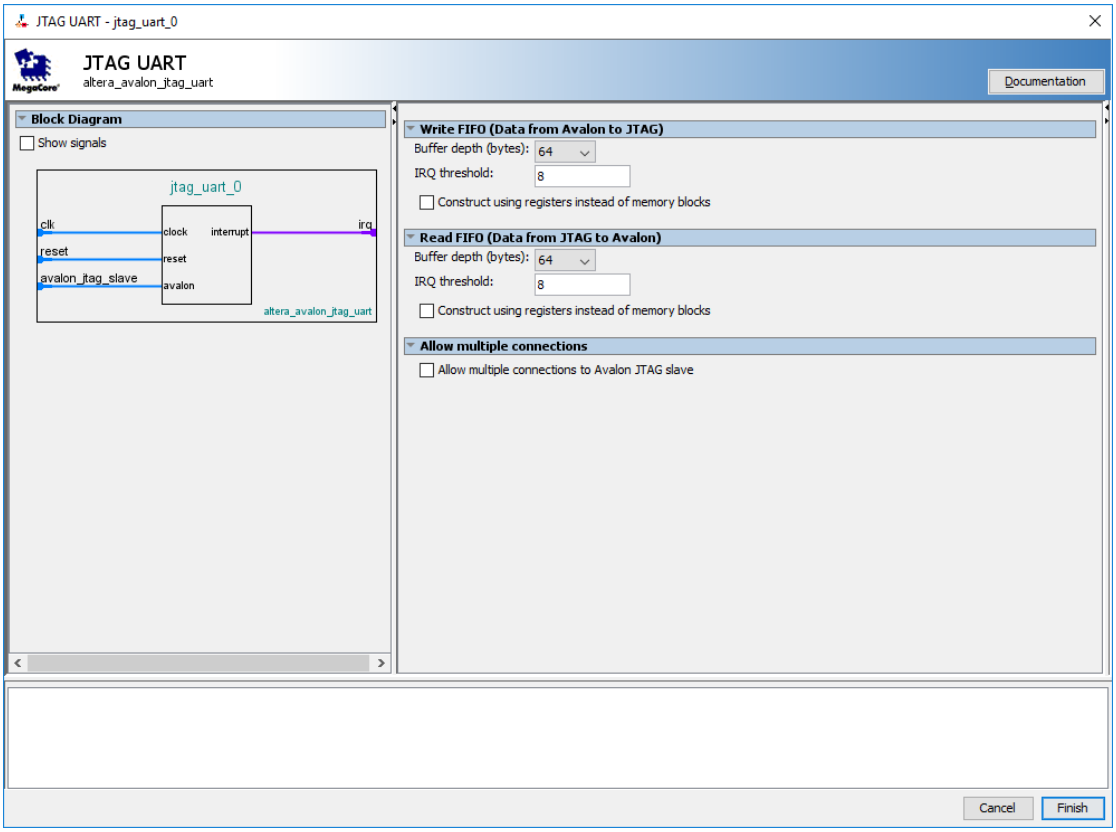
5. Add the ET SPI



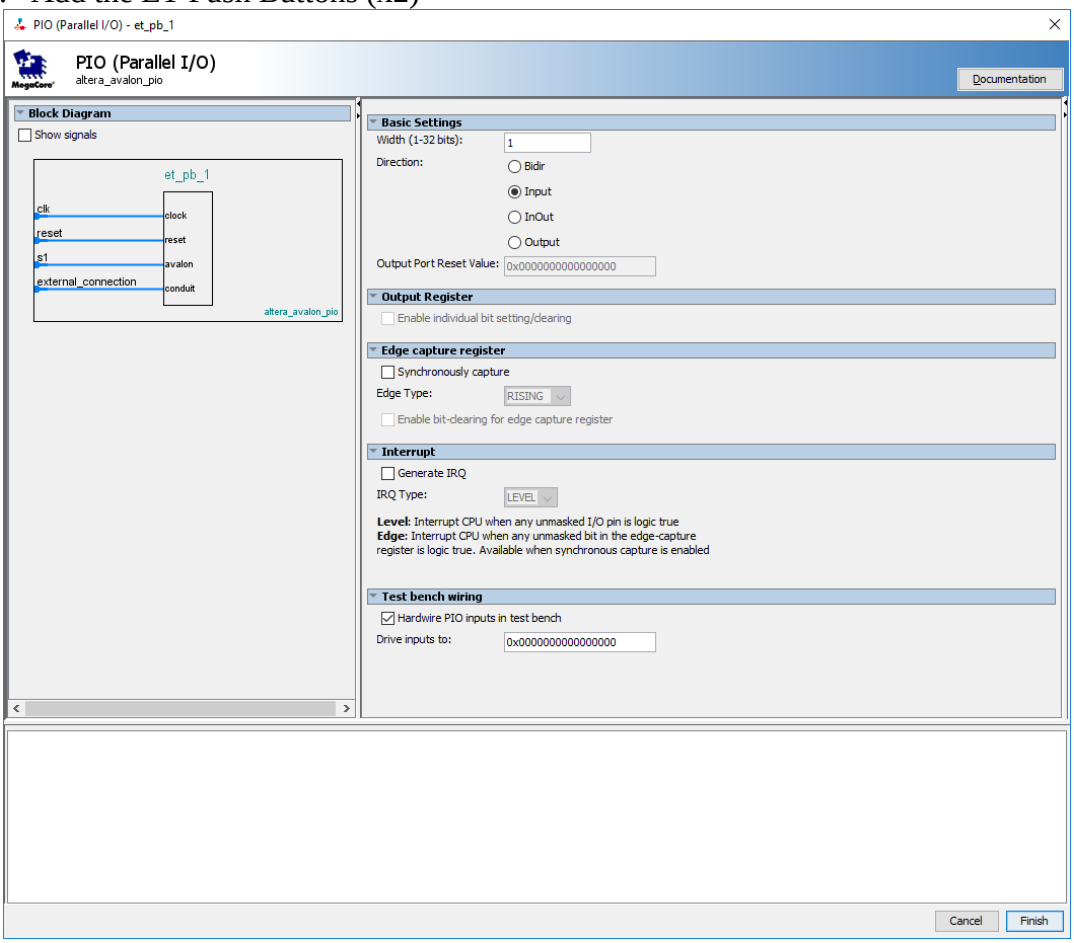
6. Add the ET SPICAN PIO



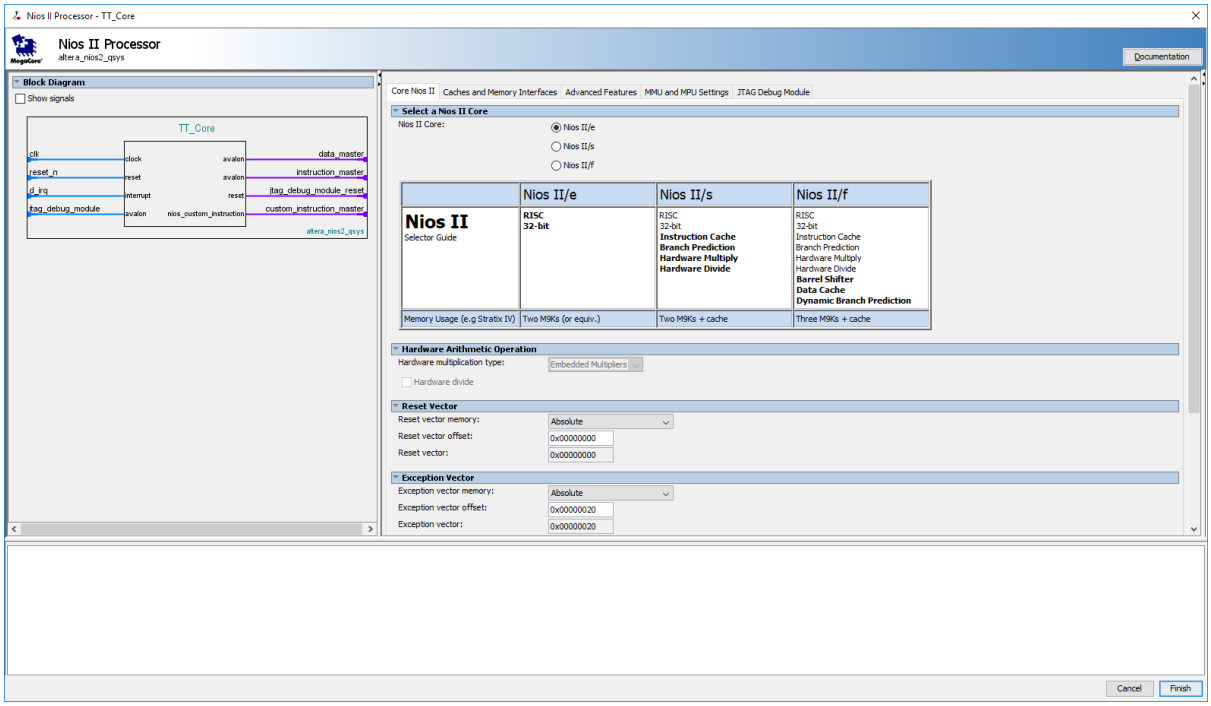
7. Add the JTAG UART



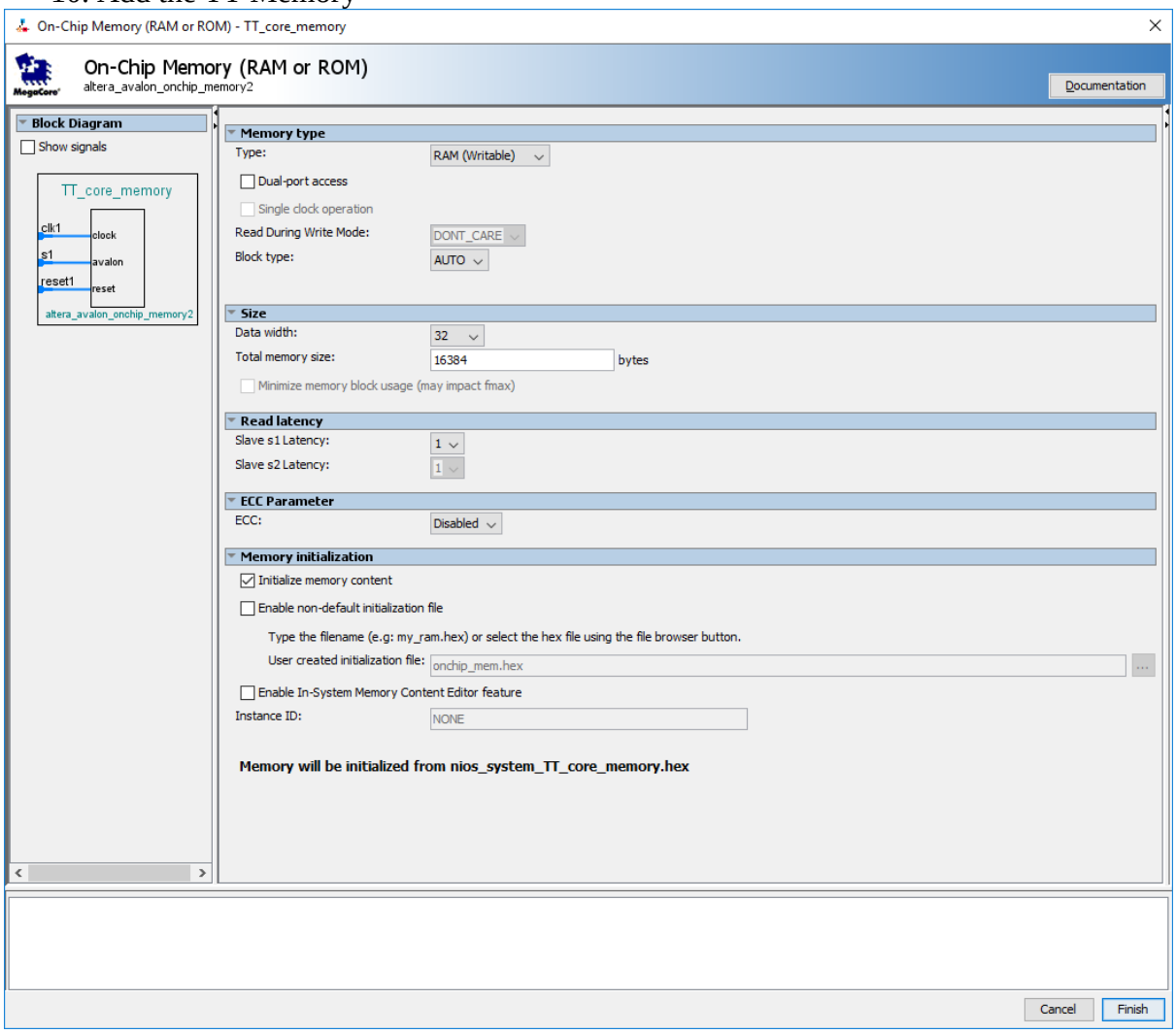
8. Add the ET Push Buttons (x2)



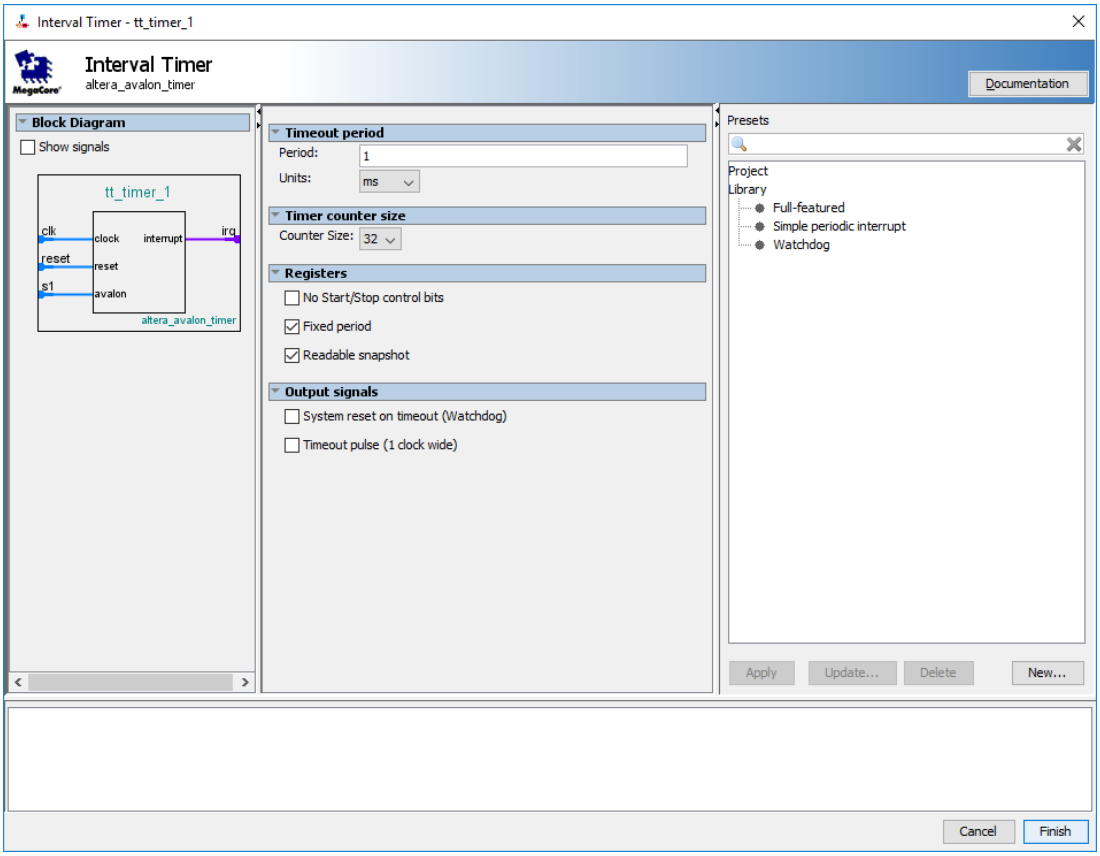
9. Add the TT Core



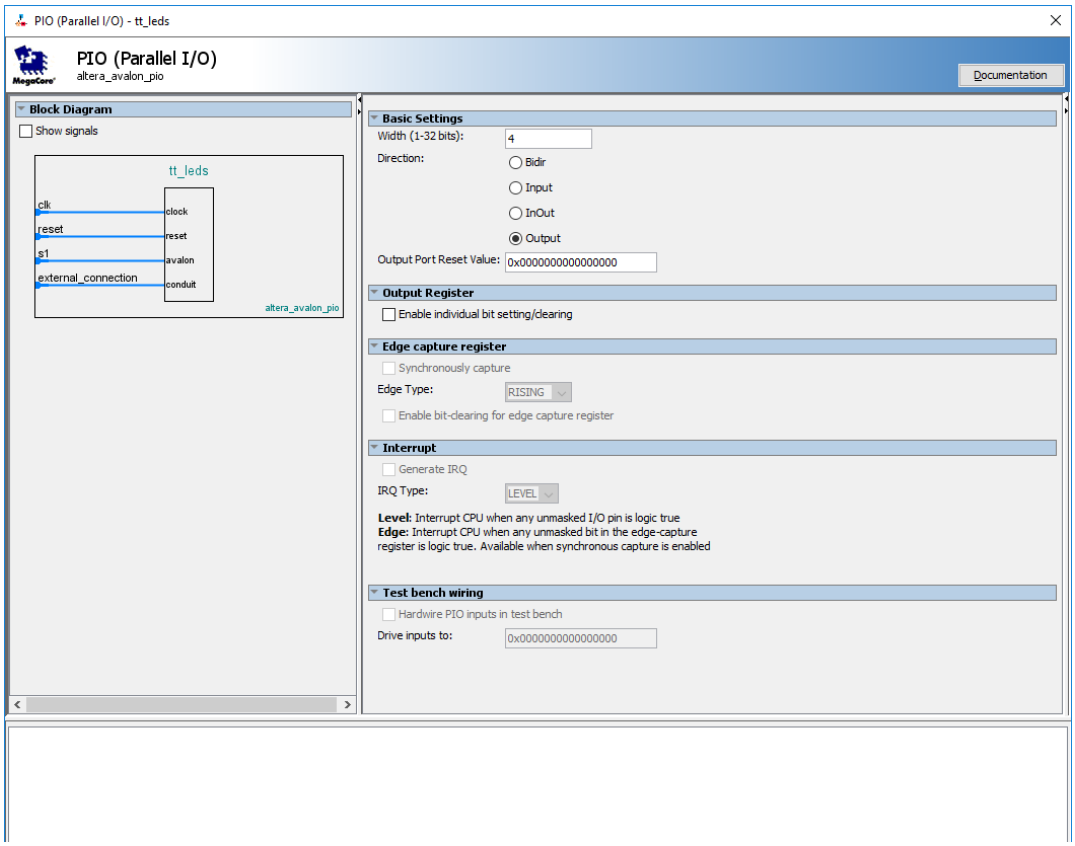
10. Add the TT Memory



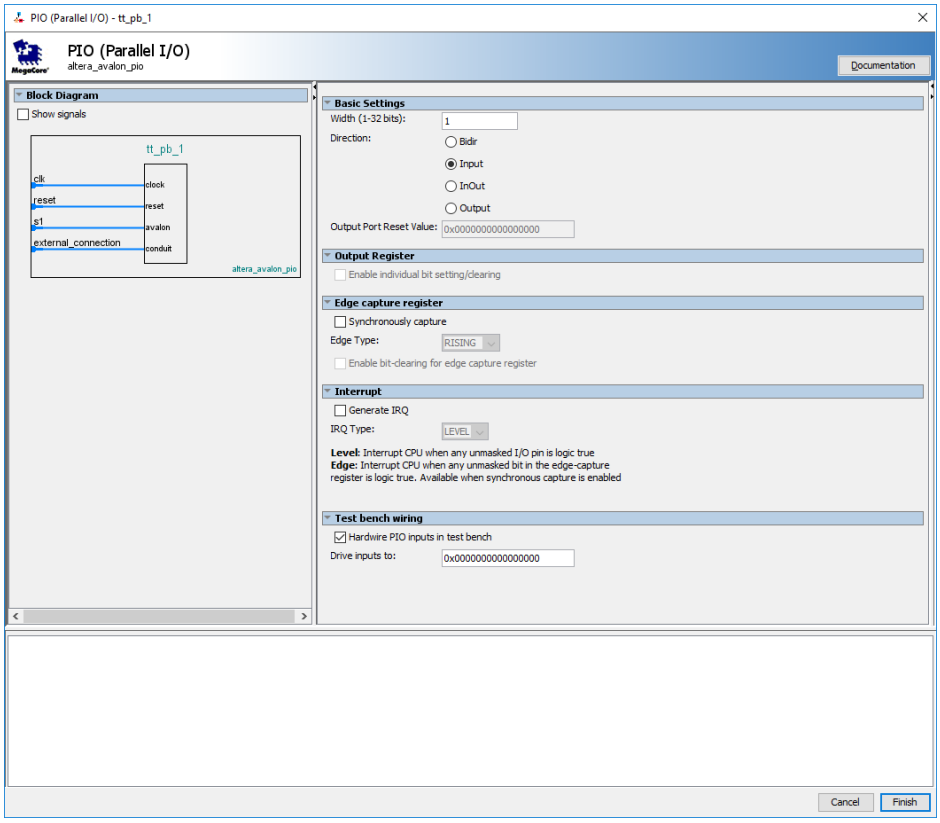
11. Add the TT Timer



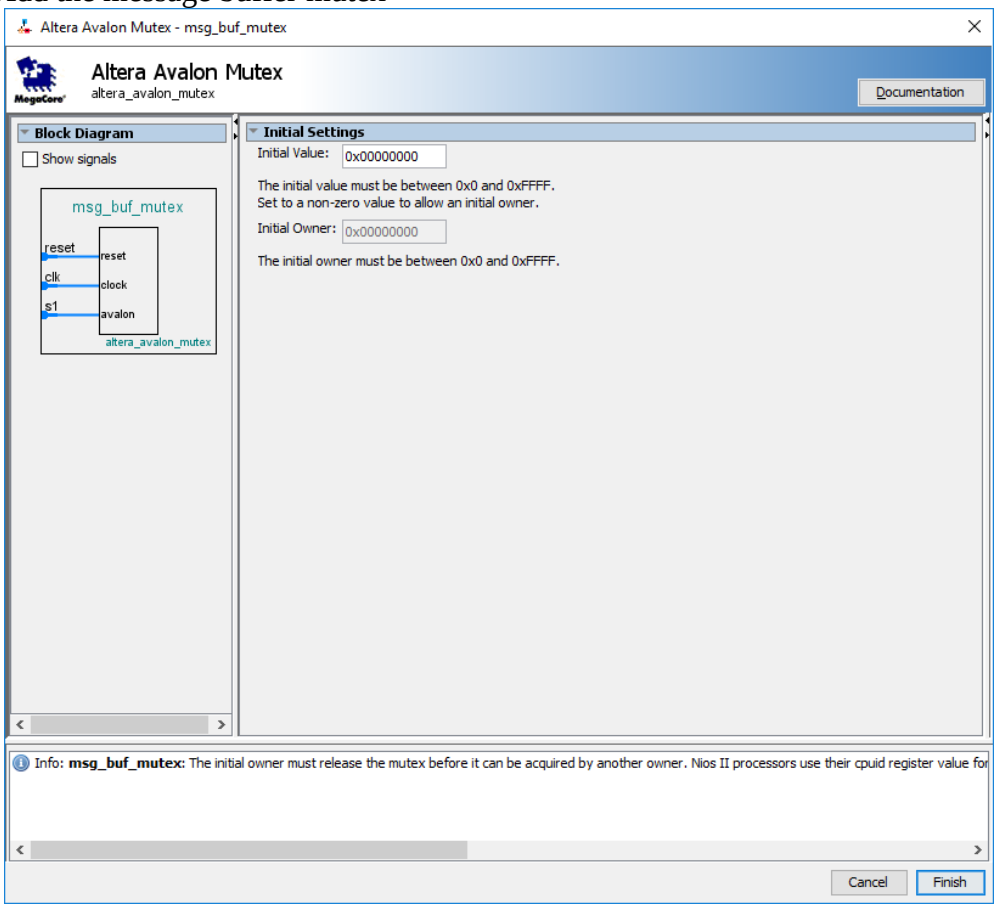
12. Add the TT LED PIO



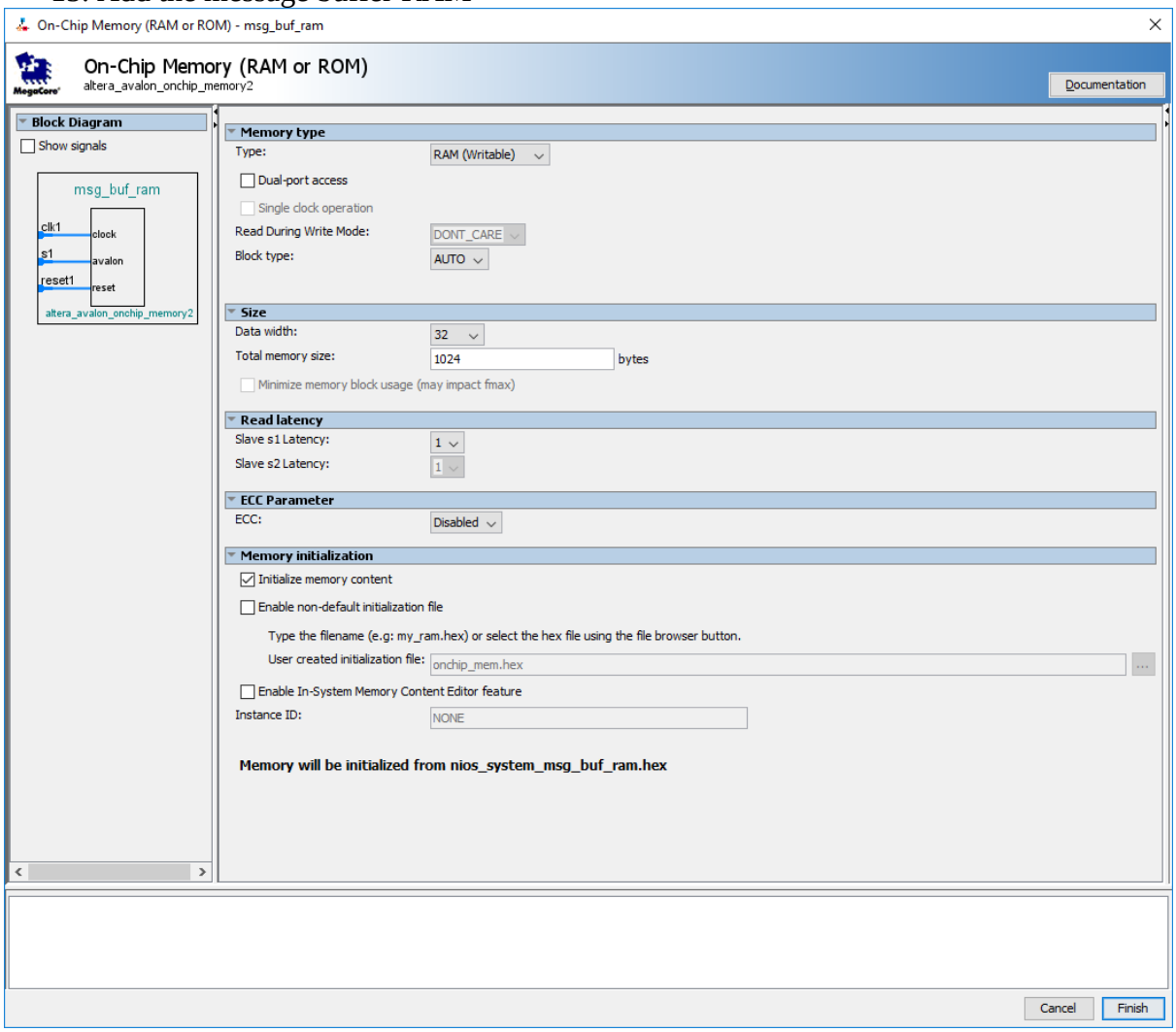
13. Add the TT Push Button



14. Add the message buffer mutex



15. Add the message buffer RAM



Once all components are added, and connect them, export the conduits, set memory addresses, and IRQs so that your design looks like the image on the next page.

Make sure to name the components identically and match up your base memory addresses

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ
☑		clk_0	Clock Source					
		clk_in	Clock Input	clk	exported			
		clk_in_reset	Reset Input	Double-click to export				
		clk	Clock Output	Double-click to export	clk_0			
		clk_reset	Reset Output	Double-click to export				
☑		ET_Core	Nios II Processor					
		clk	Clock Input	Double-click to export	clk_0			
		reset_n	Reset Input	Double-click to export	[clk]			
		data_master	Avalon Memory Mapped Master	Double-click to export	[clk]	IRQ 0	IRQ 31	
		instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]			
		jtag_debug_module_r...	Reset Output	Double-click to export	[clk]			
		jtag_debug_module	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x8800	0x8fff	
		custom_instruction_m...	Custom Instruction Master	Double-click to export				
☑		ET_Core_Memory	On-Chip Memory (RAM or ROM)					
		clk1	Clock Input	Double-click to export	clk_0			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	0x0000	0x3fdb	
		reset1	Reset Input	Double-click to export	[clk1]			
☑		et_leds1	PIO (Parallel I/O)					
		clk	Clock Input	Double-click to export	clk_0			
		reset	Reset Input	Double-click to export	[clk]			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x9470	0x947f	
		external_connection	Conduit	et_led1				
☑		et_leds2	PIO (Parallel I/O)					
		clk	Clock Input	Double-click to export	clk_0			
		reset	Reset Input	Double-click to export	[clk]			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0xa000	0xa00f	
		external_connection	Conduit	et_led2				
☑		et_spi_0	SPI (3 Wire Serial)					
		clk	Clock Input	Double-click to export	clk_0			
		reset	Reset Input	Double-click to export	[clk]			
		spi_control_port	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x9420	0x943f	1
		external	Conduit	spi_0				
☑		et_spican_int	PIO (Parallel I/O)					
		clk	Clock Input	Double-click to export	clk_0			
		reset	Reset Input	Double-click to export	[clk]			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x9460	0x946f	0
		external_connection	Conduit	spi_can_int_n				
☑		jtag_uart_0	JTAG UART					
		clk	Clock Input	Double-click to export	clk_0			
		reset	Reset Input	Double-click to export	[clk]			
		avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x9488	0x948f	3
☑		et_pb_1	PIO (Parallel I/O)					
		clk	Clock Input	Double-click to export	clk_0			
		reset	Reset Input	Double-click to export	[clk]			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x9450	0x945f	
		external_connection	Conduit	et_pb_1				
☑		et_pb_2	PIO (Parallel I/O)					
		clk	Clock Input	Double-click to export	clk_0			
		reset	Reset Input	Double-click to export	[clk]			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x9440	0x944f	
		external_connection	Conduit	et_pb_2				
☑		TT_Core	Nios II Processor					
		clk	Clock Input	Double-click to export	clk_0			
		reset_n	Reset Input	Double-click to export	[clk]			
		data_master	Avalon Memory Mapped Master	Double-click to export	[clk]	IRQ 0	IRQ 31	
		instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]			
		jtag_debug_module_r...	Reset Output	Double-click to export	[clk]			
		jtag_debug_module	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x8800	0x8fff	
		custom_instruction_m...	Custom Instruction Master	Double-click to export				
☑		TT_core_memory	On-Chip Memory (RAM or ROM)					
		clk1	Clock Input	Double-click to export	clk_0			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	0x0000	0x3fff	
		reset1	Reset Input	Double-click to export	[clk1]			
☑		tt_timer_1	Interval Timer					
		clk	Clock Input	Double-click to export	clk_0			
		reset	Reset Input	Double-click to export	[clk]			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x9400	0x941f	0
☑		tt_leds	PIO (Parallel I/O)					
		clk	Clock Input	Double-click to export	clk_0			
		reset	Reset Input	Double-click to export	[clk]			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x9430	0x943f	
		external_connection	Conduit	tt_led				
☑		tt_pb_1	PIO (Parallel I/O)					
		clk	Clock Input	Double-click to export	clk_0			
		reset	Reset Input	Double-click to export	[clk]			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x9420	0x942f	
		external_connection	Conduit	tt_pb_1				
☑		msg_buf_mutex	Altera Avalon Mutex					
		reset	Reset Input	Double-click to export	[clk]			
		clk	Clock Input	Double-click to export	clk_0			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x9480	0x9487	
☑		msg_buf_ram	On-Chip Memory (RAM or ROM)					
		clk1	Clock Input	Double-click to export	clk_0			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	0x9000	0x93ff	
		reset1	Reset Input	Double-click to export	[clk1]			
☑		pio_0	PIO (Parallel I/O)					
		clk	Clock Input	Double-click to export	clk_0			
		reset	Reset Input	Double-click to export	[clk]			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x94a0	0x94af	
		external_connection	Conduit	tt_test				

Implementing the Hardware Design

Quartus IDE

1. Open the Pin Planner tool in Quartus and complete the connections as seen below, **make sure to alter the I/O Standard voltage for the SPI I/Os**.

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	S
clk_clk	Unknown	PIN_G21	6	B6_N1	2.5 V (default)		8mA (default)	
spi_0_MISO	Unknown	PIN_AB16	4	B4_N1	3.3-V LVCMOS		2mA (default)	
spi_0_MOSI	Unknown	PIN_AA16	4	B4_N1	3.3-V LVCMOS		2mA (default)	
spi_0_SCLK	Unknown	PIN_AA15	4	B4_N1	3.3-V LVCMOS		2mA (default)	
spi_0_SS_n	Unknown	PIN_AB15	4	B4_N1	3.3-V LVCMOS		2mA (default)	
spi_can_int_n_export	Unknown	PIN_AB14	4	B4_N1	3.3-V LVCMOS		2mA (default)	
tt_led_export[0]	Unknown	PIN_F2	1	B1_N0	2.5 V (default)		8mA (default)	
tt_led_export[1]	Unknown	PIN_E1	1	B1_N0	2.5 V (default)		8mA (default)	
tt_led_export[2]	Unknown	PIN_C1	1	B1_N0	2.5 V (default)		8mA (default)	
tt_led_export[3]	Unknown	PIN_C2	1	B1_N0	2.5 V (default)		8mA (default)	
et_pb_1_export	Unknown	PIN_H2	1	B1_N1	2.5 V (default)		8mA (default)	
et_pb_2_export	Unknown	PIN_G3	1	B1_N0	2.5 V (default)		8mA (default)	
tt_pb_1_export	Unknown	PIN_F1	1	B1_N0	2.5 V (default)		8mA (default)	
tt_test_export	Unknown	PIN_AA20	4	B4_N0	3.3-V LVCMOS		2mA (default)	
et_led1_export[1]	Unknown	PIN_J2	1	B1_N1	2.5 V (default)		8mA (default)	
et_led1_export[0]	Unknown	PIN_J1	1	B1_N1	2.5 V (default)		8mA (default)	
et_led2_export[1]	Unknown	PIN_H1	1	B1_N1	2.5 V (default)		8mA (default)	
et_led2_export[0]	Unknown	PIN_J3	1	B1_N1	2.5 V (default)		8mA (default)	

2. Generate a bit stream file by compiling in Quartus II under the Compilation tab or using the compilation button.

Programmer

3. Download the design to the Cyclone III FPGA on the slave DE0 board by using Programmer under the Tools menu.

Software Implementation

Now we have two applications for the slave. There is now code for the Event-Triggered Core and the Time-Triggered Core.

Eclipse IDE

Start by opening the Eclipse based Software Build Tool from the Tools tab in Quartus II. Make sure that if you are opening Nios from the Quartus process that has the Lab 5 project open.

As we have a dual-core slave, where the cores perform different functions, we will need to create two **separate** applications, one called **ET_slave_app** and one called **TT_slave_app**, each with their **own** board support package. Follow the below instructions below **twice**, first for master, and then a second time for the slave.

To create the applications, follow the following steps for each application:

1. Open the Nios II Software Build Tool for Eclipse and select File -> New -> Nios II Application and BSP from template.
2. Select the "SOPC Information File and Blank project" option in the displayed window. Give the name for the project (ET_slave_app or TT_slave_app) and select Finish to complete this. (Do not click Next on the window).

3. Once done, a Board Support Package (BSP) is automatically generated in the Nios II Software Build Tool for Eclipse.
4. Right click on your project (ET or TT) and select “Import” from the menu, to import the code (download from blackboard).
5. Expand “General” and select “File System”. A “File System” dialog will open. Click browse to locate the source folder that includes the correct (ET or TT) files. Click on ‘Select All’ to include all files into the project. Click Finish to complete the process the projects should appear as in the screenshot below. Once the folder is added, clean and build the project.
6. If you have errors, try cleaning and building again to see if this clears any issues.

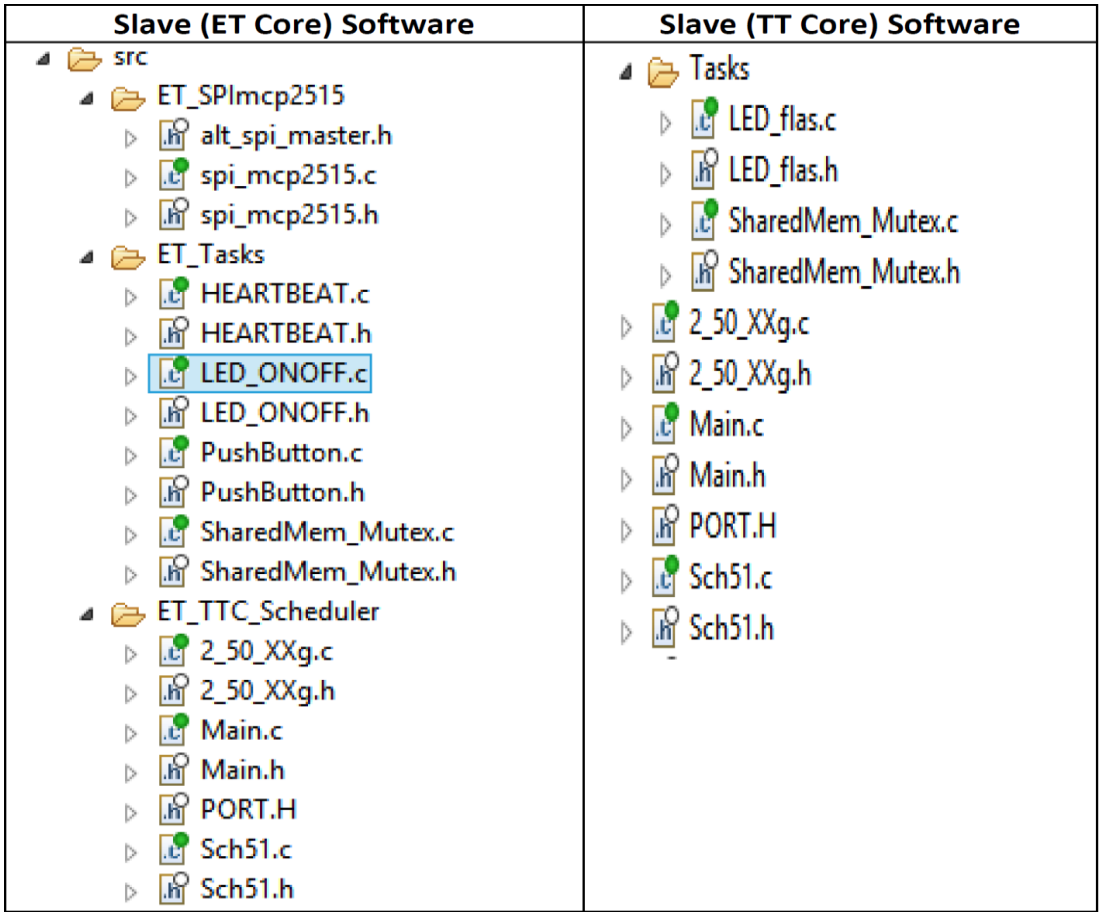


Figure 2. Software Configurations of Slave Node (ET core and TT core).

Prof. Tanya Vladimirova

20 November 2018