De1-SoC Board Peripheral Buttons

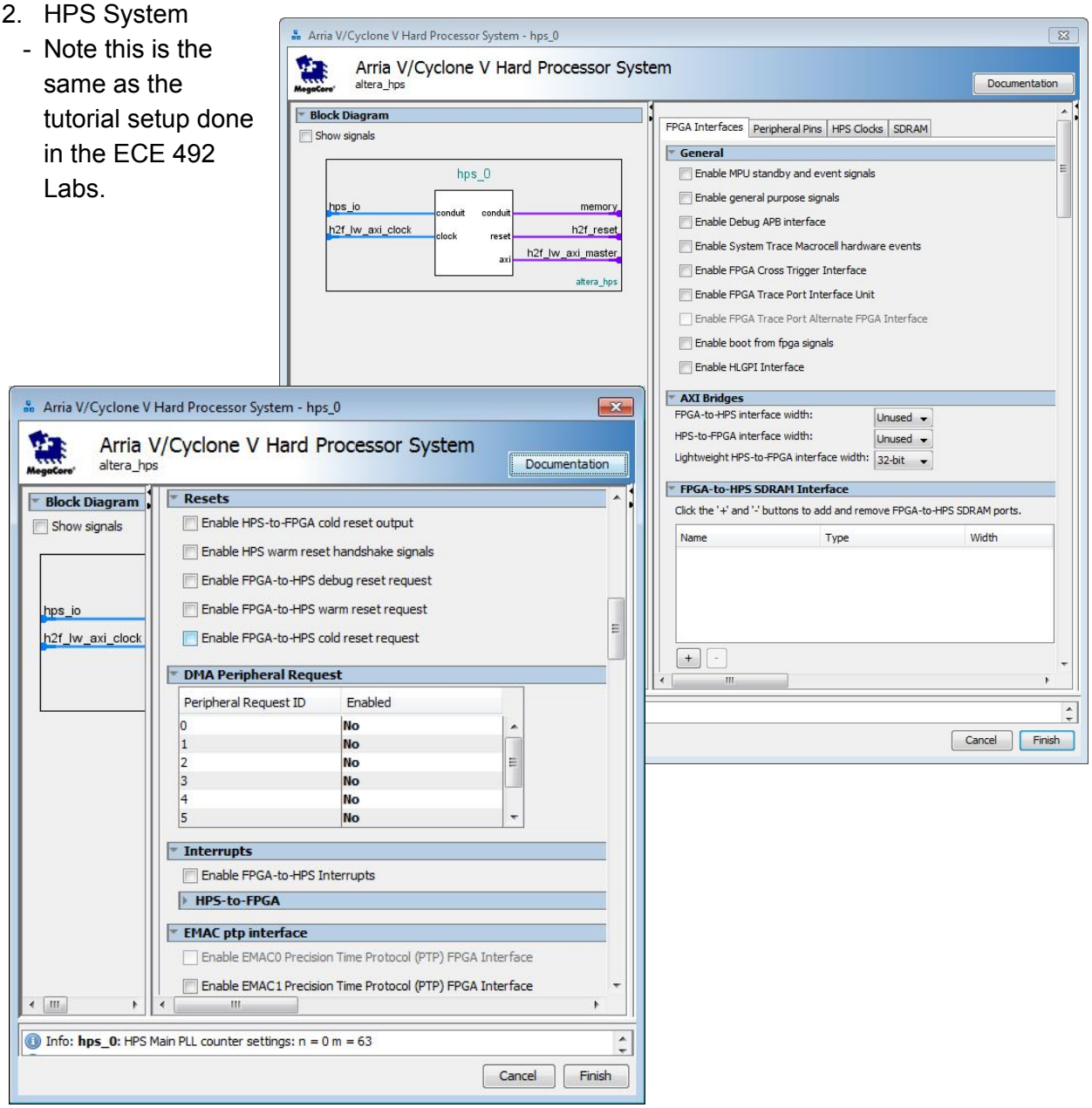
Introduction

These notes will show how to setup and configure the on board buttons, as well as provide a sample task that shows button reactions on the red LEDS.

Hardware Procedure

The following images show the minimum required components in the Qsys system as well as the required settings.

1. Clock
    - Use the default clock source created by the New Project Wizard
2. HPS System
    - Note this is the same as the tutorial setup done in the ECE 492 Labs.



Arria V/Cyclone V Hard Processor System - hps_0

Arria V/Cyclone V Hard Processor System

altera_hps

Documentation

FPGA Interfaces | Peripheral Pins | HPS Clocks | SDRAM

Block D

Show s

hps_io

h2f_lw_

Ethernet Media Access Controller

EMAC0 pin:	Unused
EMAC0 mode:	N/A
EMAC1 pin:	HPS I/O Set 0
EMAC1 mode:	RGMII

NAND Flash Controller

Quad SPI Flash Controller

SD/MMC Controller

SDIO pin:	HPS I/O Set 0
SDIO mode:	4-bit Data

USB Controllers

USB0 pin:	Unused
USB0 PHY interface mode:	N/A
USB1 pin:	HPS I/O Set 0
USB1 PHY interface mode:	SDR with PHY clock output mode

SPI Controllers

SPIM0 pin:	Unused
SPIM0 mode:	N/A
SPIM1 pin:	HPS I/O Set 0
SPIM1 mode:	Single Slave Select
SPIS0 pin:	Unused
SPIS0 mode:	N/A
SPIS1 pin:	Unused
SPIS1 mode:	N/A

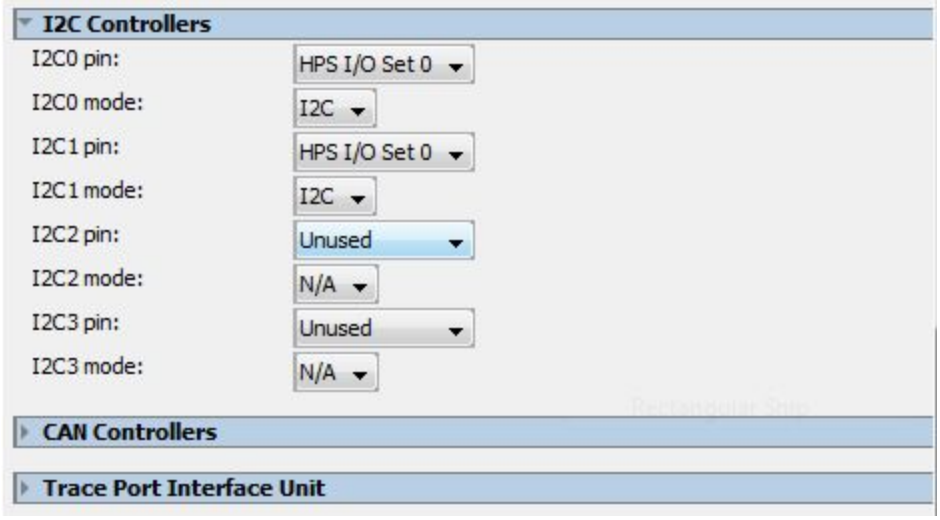
UART Controllers

UART0 pin:	HPS I/O Set 0
UART0 mode:	No Flow Control
UART1 pin:	Unused
UART1 mode:	N/A

Info: **hps_0**: HPS Main PLL counter settings: n = 0 m = 63

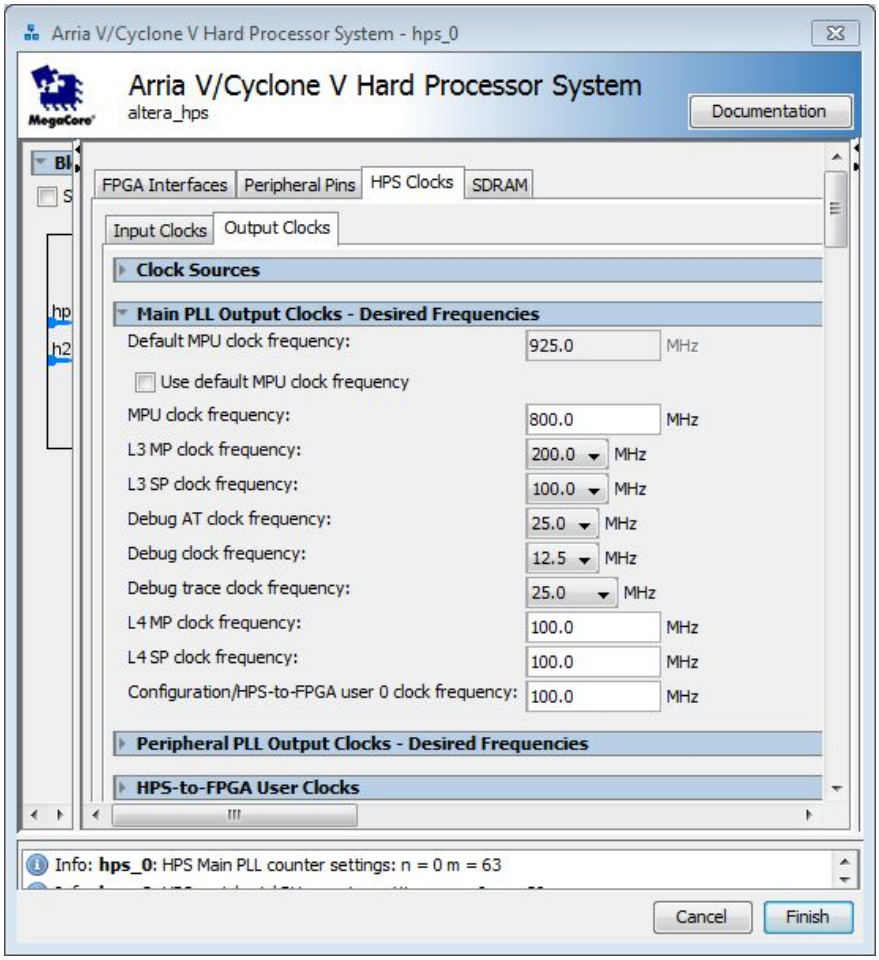
Info: **hps_0**: HPS peripherial PLL counter settings: n = 0 m = 39

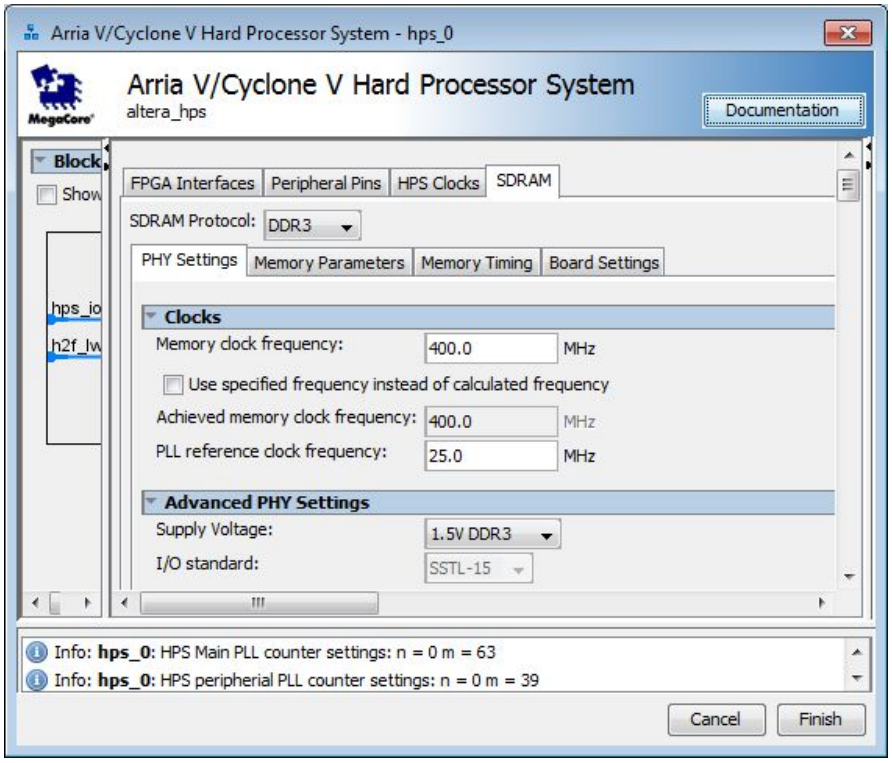
Cancel | Finish

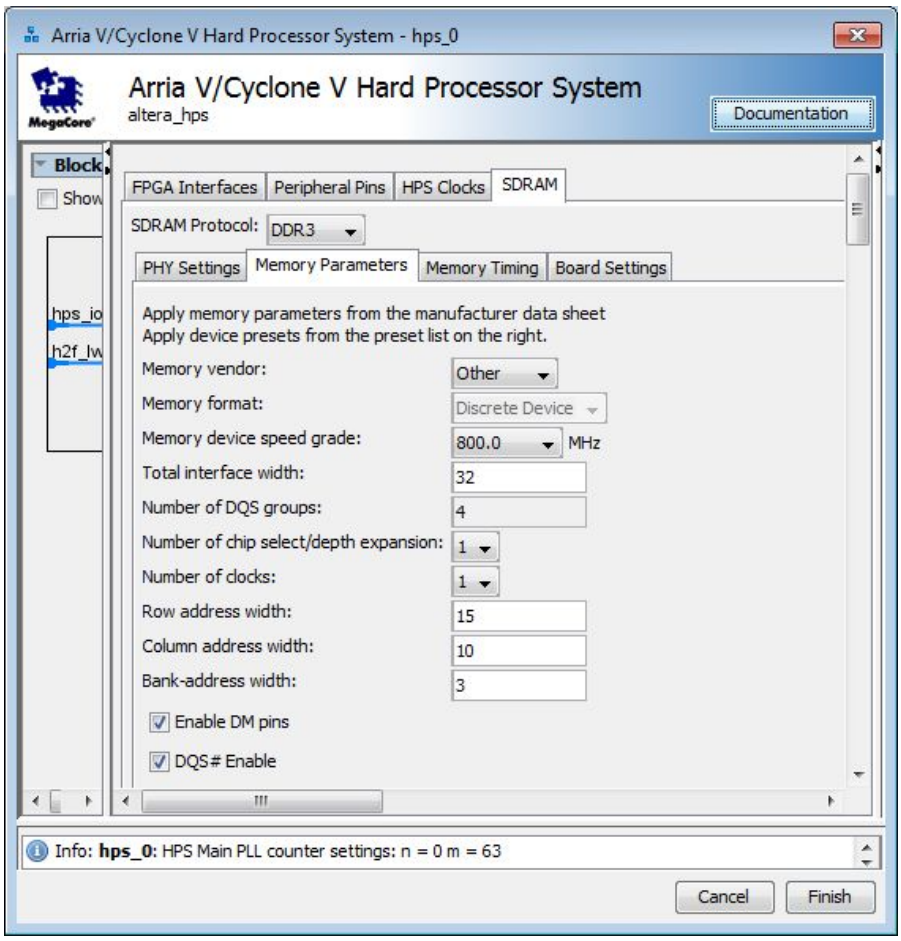


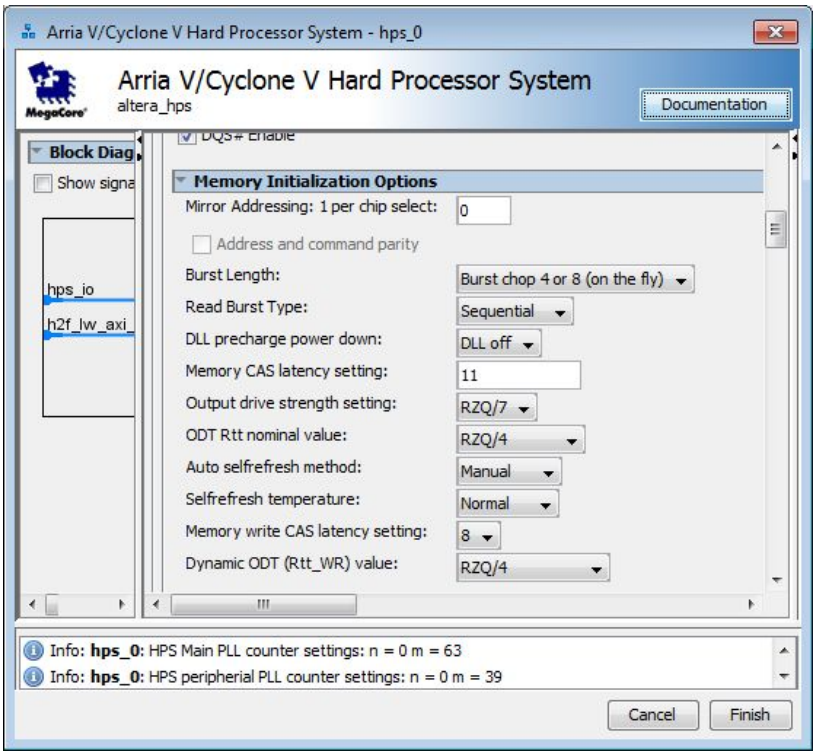
In the Peripheral Mux Table click on the GPIO09, GPIO35, GPIO40, GPIO48, GPIO53, GPIO54 and GPIO61 pins to appropriately map them manually. .

Under the HPS Clocks tab leave the Input Clocks tab as the default.
The output clocks are as follows:



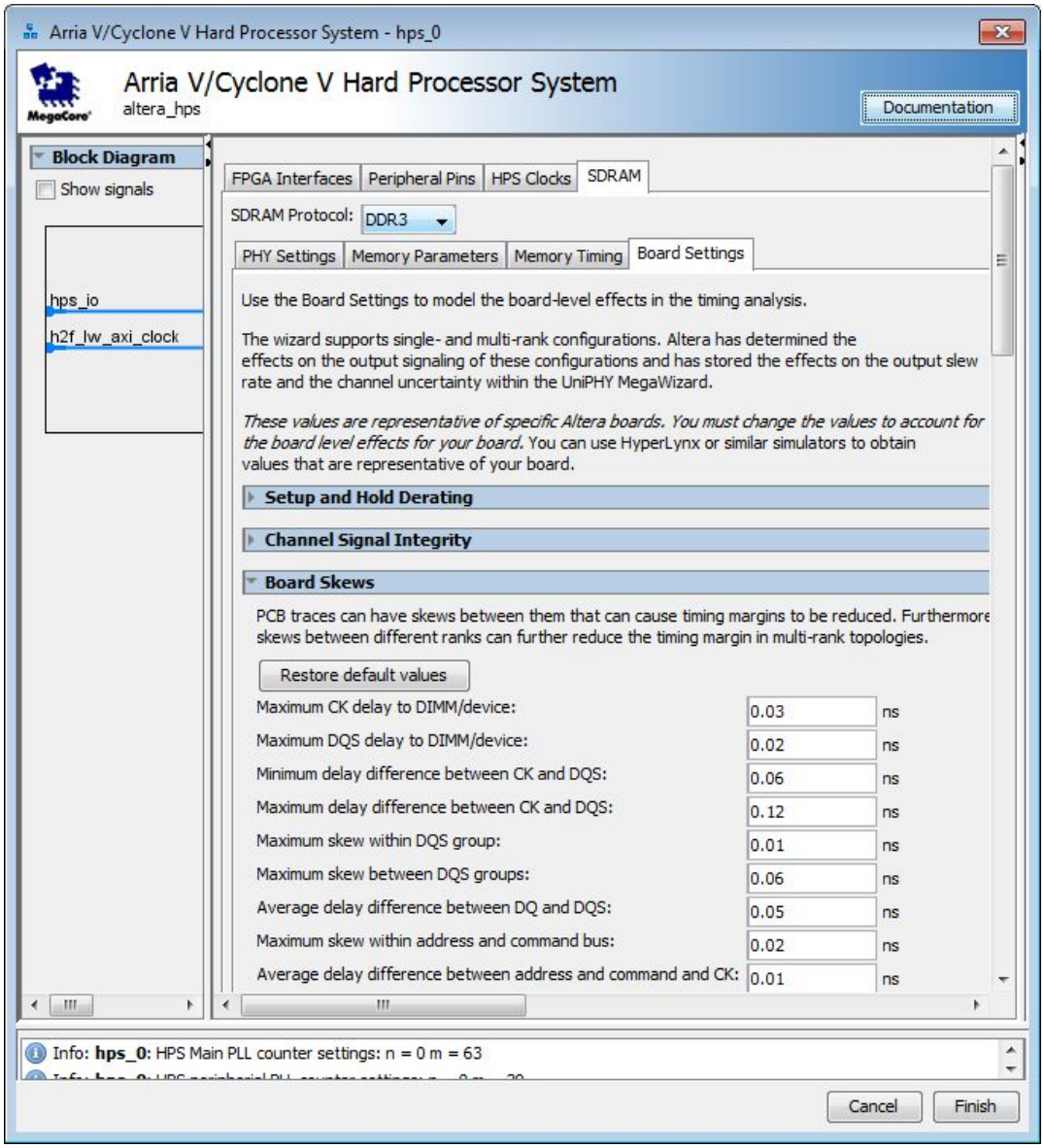

Arria V/Cyclone V Hard Processor System - hps_0
Arria V/Cyclone V Hard Processor System
altera_hps
Documentation
Block
Show
hps_io
h2f_lw
FPGA Interfaces
Peripheral Pins
HPS Clocks
SDRAM
SDRAM Protocol:
DDR3
PHY Settings
Memory Parameters
Memory Timing
Board Settings
Clocks
Memory clock frequency:
400.0
MHz
Use specified frequency instead of calculated frequency
Achieved memory clock frequency:
400.0
MHz
PLL reference clock frequency:
25.0
MHz
Advanced PHY Settings
Supply Voltage:
1.5V DDR3
I/O standard:
SSTL-15
Info: hps_0: HPS Main PLL counter settings: n = 0 m = 63
Info: hps_0: HPS peripherial PLL counter settings: n = 0 m = 39
Cancel
Finish


Arria V/Cyclone V Hard Processor System - hps_0
Arria V/Cyclone V Hard Processor System
altera_hps
Documentation
Block
Show
hps_io
h2f_lw
FPGA Interfaces
Peripheral Pins
HPS Clocks
SDRAM
SDRAM Protocol:
DDR3
PHY Settings
Memory Parameters
Memory Timing
Board Settings
Apply memory parameters from the manufacturer data sheet
Apply device presets from the preset list on the right.
Memory vendor:
Other
Memory format:
Discrete Device
Memory device speed grade:
800.0
MHz
Total interface width:
32
Number of DQS groups:
4
Number of chip select/depth expansion:
1
Number of clocks:
1
Row address width:
15
Column address width:
10
Bank-address width:
3
Enable DM pins
DQS# Enable
Info: hps_0: HPS Main PLL counter settings: n = 0 m = 63
Cancel
Finish


Arria V/Cyclone V Hard Processor System - hps_0
Arria V/Cyclone V Hard Processor System
altera_hps
Documentation
Block Diag
Show signa
hps_io
h2f_lw_axi_
DQS# Enable
Memory Initialization Options
Mirror Addressing: 1 per chip select:
0
Address and command parity
Burst Length:
Burst chop 4 or 8 (on the fly)
Read Burst Type:
Sequential
DLL precharge power down:
DLL off
Memory CAS latency setting:
11
Output drive strength setting:
RZQ/7
ODT Rtt nominal value:
RZQ/4
Auto selfrefresh method:
Manual
Selfrefresh temperature:
Normal
Memory write CAS latency setting:
8
Dynamic ODT (Rtt_WR) value:
RZQ/4
Info: hps_0: HPS Main PLL counter settings: n = 0 m = 63
Info: hps_0: HPS peripherial PLL counter settings: n = 0 m = 39
Cancel
Finish

Under Board Settings -> Setup and Hold Derating and Channel Signal Integrity, leave Altera's default settings checked.

The remaining configuration is as follows:

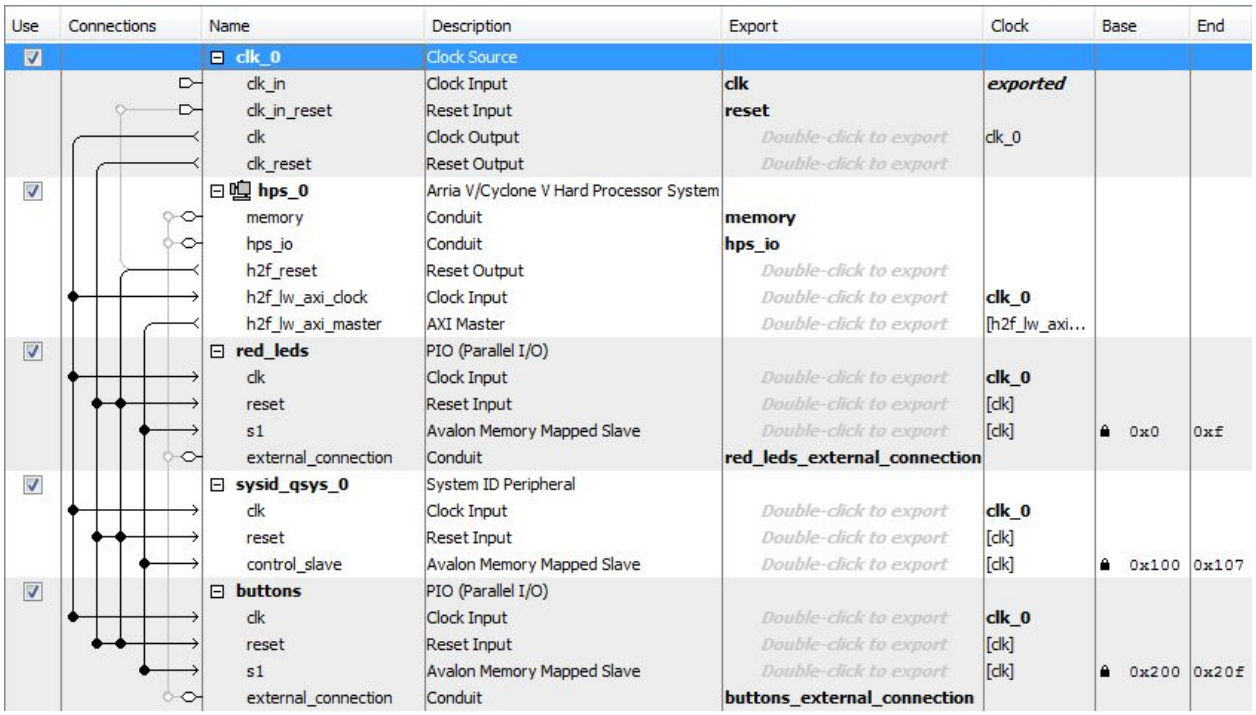


3. PIO (Parallel I/O) for red LEDs
   Set the width to 10 and the direction to output. Leave all other settings as default.

4. PIO (Parallel I/O) for buttons
   Set the width to 4 and the direction to input. Leave all other settings as default.

5. System ID Peripheral
   Use the default configuration

Once these components are added, connect them according to the following image:



Use	Connections	Name	Description	Export	Clock	Base	End
☑		clk_0	Clock Source				
		clk_in	Clock Input	clk	exported		
		clk_in_reset	Reset Input	reset			
		clk	Clock Output	Double-click to export	clk_0		
		clk_reset	Reset Output	Double-click to export			
☑		hps_0	Arria V/Cyclone V Hard Processor System				
		memory	Conduit	memory			
		hps_io	Conduit	hps_io			
		h2f_reset	Reset Output	Double-click to export			
		h2f_lw_axi_clock	Clock Input	Double-click to export	clk_0		
		h2f_lw_axi_master	AXI Master	Double-click to export	[h2f_lw_axi...		
☑		red_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]	🔒 0x0	0xf
		external_connection	Conduit	red_leds_external_connection			
☑		sysid_qsys_0	System ID Peripheral				
		clk	Clock Input	Double-click to export	clk_0		
		reset	Reset Input	Double-click to export	[clk]		
		control_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	🔒 0x100	0x107
☑		buttons	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]	🔒 0x200	0x20f
		external_connection	Conduit	buttons_external_connection			

Note, the base addresses for the red leds and buttons are set at 0x000 and 0x200, respectively. The system ID peripheral base address is 0x100. You may change these to suit your design as long as they are updated in the app.c file (discussed later).

Next, ensure your toplevel VHDL file in Quartus is connected as follows. A template is provided in buttons.vhd for reference.

A. Declare these pins in the entity (in addition to all of the hps signals):

```
-- FPGA side pins
        LEDR                                    : out std_logic_vector(9 downto 0);
-- Button / Key Pins
        KEY_N                                   : in std_logic_vector(3 downto 0)
```

B. Add these ports in the architecture's component declaration

```
buttons_external_connection_export  : in    std_logic_vector(3 downto 0)  := (others => 'X'); -- export
red_leds_external_connection_export : out   std_logic_vector(9 downto 0)                    -- export
```

C. Port map the pins to the ports in the instantiation:
   buttons_external_connection_export => KEY_N,
   red_leds_external_connection_export => LEDR

Note, the reset signal is by default connected to KEY_0, and is also connected via port map of
   reset_reset_n => KEY_N(0),
This current implementation does not allow for use of 1 of the buttons since a reset signal needs to be connected within the FPGA. This is to be updated once a suitable alternative is found.

Ensure that the .qip and .tcl scripts have also been added to the project before compiling. Also be sure to run the appropriate .tcl scripts beforehand. Then you may compile the design.

A zipped archive of a template project is included with this report for your reference.

Software Procedure

To ensure the software can properly communicate with the hardware, specify the base addresses in the app.c file, specifically for the red LEDs and buttons. These address locations can be read/written from/to as follows

```
long ButtonsPressed = alt_read_word(BUTTONS_BASE);
alt_write_word(LEDR_BASE, ButtonsPressed);
```

An example is shown in the attached app.c file.
Since the buttons are active low, 0xf is constantly being written to the LEDs turning them on. Thus once a button is pressed, the value changes and can be handled accordingly. The provided if block provides a skeleton outline for handling different key presses. This skeleton would only handle one button press at a time and could be expanded to handle more cases to handle multiple button presses simultaneously.

References

[1] University of Toronto. *Digital Embedded Systems Lab: Push Buttons.* [Online]. Available:http://www-ug.eecg.utoronto.ca/desl/nios_devices_SoC/dev_pushbuttons.html

Attached Files

app.c: Provides an example main.c and AppTask to detect button presses.
Buttons.qar: Archived Quartus/Qsys project example to implement peripheral buttons.