Technical Article

Interfacing a BeagleBone BLE/Wi-Fi Cape with an Arrow Max10 DECA Board: Part 1

Allie Hopkins

This is a guest blog post by John Tauch, principal engineer with Dallas Logic.

In today's smart device world, designers are faced with the challenge of adding wireless communication interfaces to their products. Whether you need to provide simple monitoring, data backhaul or product reconfiguration, the ability to add wireless device interfaces has become a requirement rather than an option. One technology found in many of today's products is the FPGA (field programmable gate array). FPGA's are becoming more prevalent in many products, but are rarely discussed when the conversation turns to adding smart device connectivity. In today's post, I'll discuss adding the SimpleLink™ multi-standard CC2650 wireless microcontroller (MCU) to your Altera Max10 FPGA Project. Although the target device in this post is Altera's Max10 FPGA, the material is applicable to any Altera FPGA. For the following example we are using Arrow's Max10 DECA board and Arrow's BeagleBone Bluetooth® low energy (BLE) / Wi-Fi® cape plug-in module.

Let's begin with an overview of the BLE / Wi-Fi cape. The CC2650 wireless MCU shown in Figure 1 in GREEN, consists of the CC2650 device, a dedicated CJTAG programming header and a chip antenna. The CC2650 data interface can be accessed using either SPI or I2C. The programming header uses TI's standard pin out, thus allowing the CC2650 wireless MCU to be configured by a number of programming sources including: TI's RF-06, CC2650DK, or IAR's CJTAG programmer. Sixteen of the CC2650 device's 31 IOs are present on the expansion headers. Both the SPI and I2C interfaces are available on the modules P1 header.

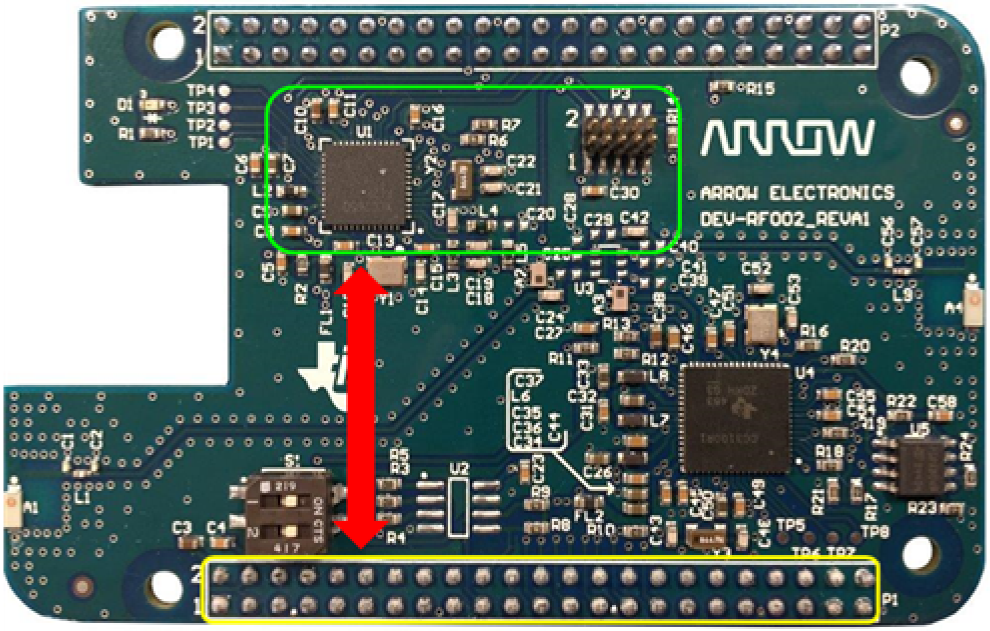


Figure 1. BeagleBone BLE / Wi-Fi Cape

Figure 2 is the interface diagram for the DECA board and BeagleBone BLE/Wi-Fi Cape. For today's post we will focus on the SPI bus circled in RED. This bus is used as the primary communication interface between the Nios II soft processor and the CC2650 wireless MCU.

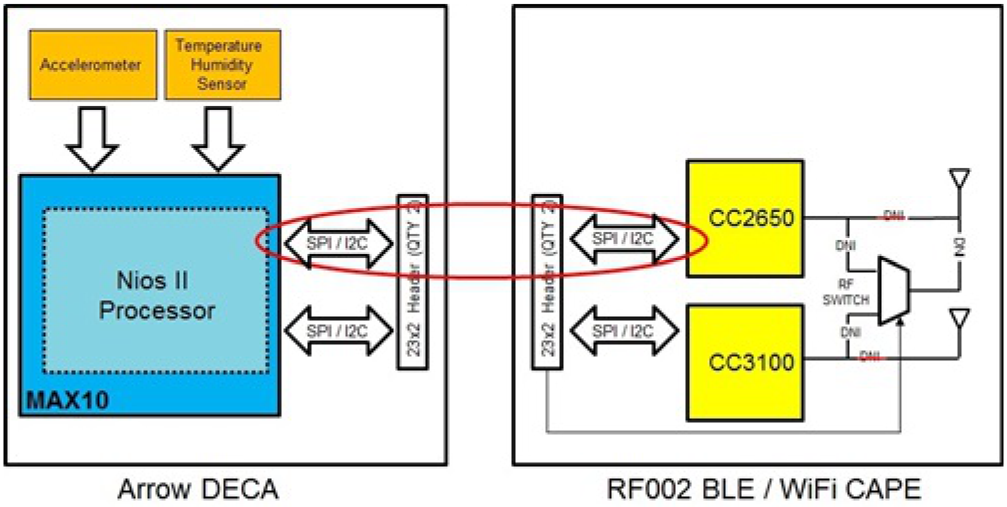
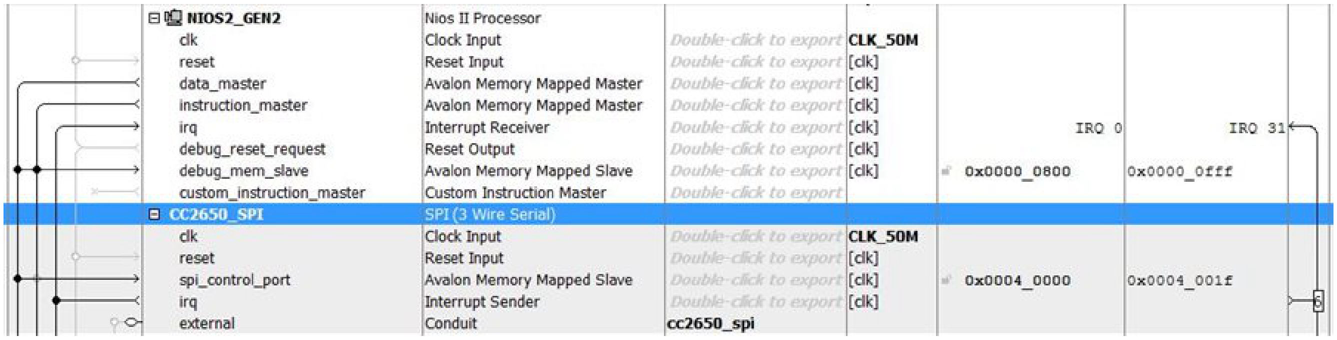


Figure 2. DECA to Cape Connection Diagram

The Nios II processor is synthesized from FPGA logic elements and resides inside the Max10 device. The function of the Nios II is to read the DECA's onboard sensors and then transfer the sensor data to the CC2650 device. Upon receiving the sensor data, the CC2650 will transmit an update to the Bluetooth Smart connected device (smartphone or tablet). The CC2650 wireless MCU is configured to use a 500ms interval timer which sets the sensor update resolution. At every 500ms interval, the CC2650 device will strobe its data request signal. The data request signal is connected to a dedicated interrupt on the Nios II processor. Once interrupted, the Nios II will initiate a SPI data transfer and then request updated data from each of the DECA's sensors. This will insure that updated sensor data is available at every 500ms interval.

For this design, we utilized Altera's Nios II soft processor and a standard Qsys SPI peripheral. The Qsys system integration tool provides a convenient environment for building Nios II based systems. Figure 3 shows a portion of the system contents page which is used to manage the connectivity in a Qsys system.

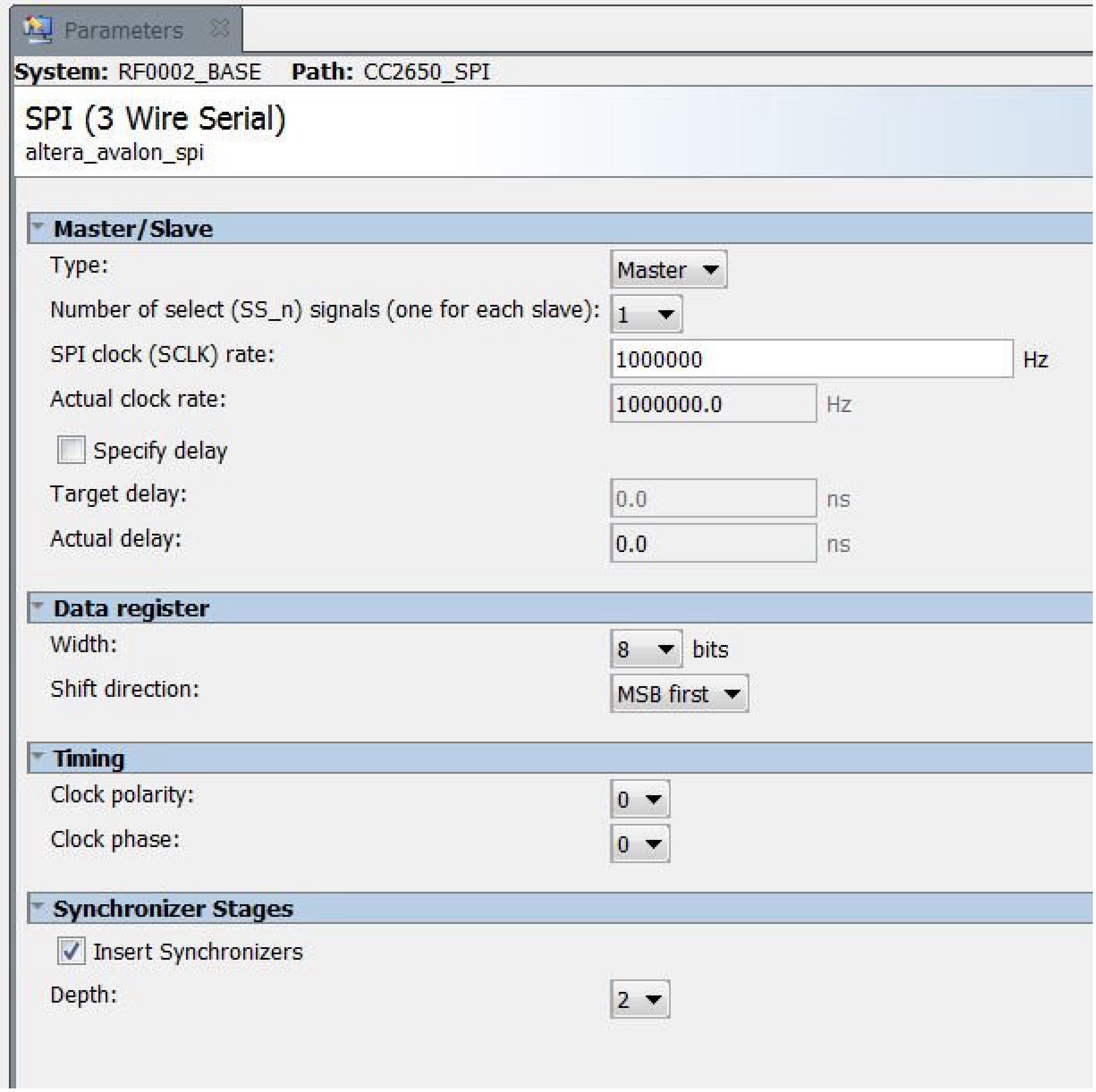
In Figure 3, you can see both the Nios II processor and the SPI peripheral. To connect the peripheral all you need to do is click on the bubble that attaches the SPI peripheral to the Nios II's data_master and connect the SPI peripherals IRQ. The last step is to provide the peripheral with an address location and then configure the SPI peripheral itself.



⊟ NIOS2_GEN2	Nios II Processor					
clk	Clock Input	Double-click to export	CLK_50M			
reset	Reset Input	Double-click to export	[clk]			
data_master	Avalon Memory Mapped Master	Double-click to export	[clk]			
instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]			
irq	Interrupt Receiver	Double-click to export	[clk]	IRQ 0	IRQ 31	
debug_reset_request	Reset Output	Double-click to export	[clk]			
debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_0800	0x0000_0fff	
custom_instruction_master	Custom Instruction Master	Double-click to export				
⊟ CC2650_SPI	SPI (3 Wire Serial)					
clk	Clock Input	Double-click to export	CLK_50M			
reset	Reset Input	Double-click to export	[clk]			
spi_control_port	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0004_0000	0x0004_001f	
irq	Interrupt Sender	Double-click to export	[clk]			6
external	Conduit	cc2650_spi				

Figure 3. Altera's Qsys Peripheral List

Figure 4 shows the SPI peripherals parameter window. The parameter window provides all of the configurable attributes associated with the SPI peripheral, for instance clock rate, data width, shift direction, etc.



Parameters

System: RF0002_BASE **Path:** CC2650_SPI

SPI (3 Wire Serial)

altera_avalon_spi

Master/Slave

Type: Master

Number of select (SS_n) signals (one for each slave): 1

SPI clock (SCLK) rate: 1000000 Hz

Actual clock rate: 1000000.0 Hz

□ Specify delay

Target delay: 0.0 ns

Actual delay: 0.0 ns

Data register

Width: 8 bits

Shift direction: MSB first

Timing

Clock polarity: 0

Clock phase: 0

Synchronizer Stages

☑ Insert Synchronizers

Depth: 2

Figure 4. Altera's Qsys SPI Configuration Window

Once you have completed adding all of your system's peripherals you can generate your Nios II system. The output of the system generation process is an HDL (hardware description language) module that can be instantiated in your FPGA design. You can select either VHDL or Verilog as the HDL output for the system generation process. In addition to the HDL module, the QSYS tool will also generate a nios.sopcinfo file that will be used to create a BSP (board support package) for the Nios II software system.

With the hardware interface completed, all that remains is to add some software to read and write to the SPI peripheral. The Nios II software / firmware environment is an Eclipse based integrated development environment (IDE), so this should be familiar to most embedded designers. In the next blog in this series, we will take a look at the embedded portion of our FPGA design.

Check out the rest of the blogs in this series:

- Interfacing a BeagleBone BLE/Wi-Fi Cape with an Arrow Max10 DECA board: Part 2
- Interfacing a BeagleBone BLE/Wi-Fi Cape with an Arrow Max10 DECA board: Part 3

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265