Starting A Project With Altera Quartus II And Creating A System With Qsys

fpga@seanwrall.com fpga.seanwrall.com

YouTube

https://www.youtube.com/watch?v=2VL7mHfvoyY

Website

http://fpga.seanwrall.com/lessons/lesson2/

Date Updated: October 16, 2015
Version: 1.0

Documentation Written By Sean W. Rall

Revision History

Last Updated: October 16, 2015

No Updates

Creating A Project in Quartus II

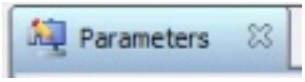
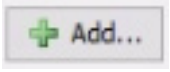
1. Open Quartus II
2. Go to File --> New Project Wizard...
   A. Click next to the introduction screen
   B. For the working directory choose C:/altera/mySystems/system1
   C. Name the project NiosII
   D. Name the top level entity the same as the project name, in this case Nios II
   E. Click next
   F. Choose an Empty project
   G. Click next
   H. Click next to the add files (this is for importing designs into your project, but we don't have any designs)
   I. Click next
   J. Under family choose Cyclone IV E
   K. Filter by the number written on your FPGA, for the DE2-115 it is "ep4ce115f29c7"
   L. Select your FPGA
   M. Click next
   N. Under simulation select none
   O. Click next
   P. Now you have a summary page then click finish
   Q. Now you have the basic project setup

Note: You cannot have any spaces in your file path or file names

Creating a Block Diagram in Quartus II

1. Go to File --> New
   A. Select Block Diagram/Schematic File
   B. Click Ok
2. Save the Block digram file by going to File --> Save As
   A. Make sure you are in your working directory (in this case C:/altera/mySystems/system1)
   B. Save the block diagram the same as what you named your project
      (in this case NiosII.bdf)
   C. Click Save

Creating A Basic System In Qsys

1. Go to Tools --> Qsys (or you can click on this ion  in the task-bar)
2. Clock
   A. Double Click on clk_0 to edit it
      i. Make sure the clock frequency is set to 50000000 Hz
      ii. Exit out of the clock by clicking the X 
   B. Rename the clk_0 to Clock by right clicking on the clock and choosing Rename (alternatively you can select to the clock with a single click and press F2 yon your keyboard to rename)
   C. Export clk_in by double clicking in the associated export column; name this clk
   D. Export clk_in_reset by double clicking in the associated export column; name this reset
3. Adding The Nios II Processor
   E. In the Ip Catalog in the left hand side navigate to Processors and Peripherals --> Embedded Processors --> Nios II Processor
      i. Double clcik on the name of the component to add, or select the component and click add 
   F. A Settings window swill appear
      i. Under main for Nios II core select Nios II/f
      ii. Click Finish
   G. Rename the nios2_gen2_0 by right clicking on name and selecting rename, rename the processor to “nios_processor”
   H. Add the following connections by clicking on the circle to make it a filled in circle under the connections column
      i. Clock.clk connected to nios_processor.clk
      ii. Clock.clk_reset connected to nios_processor.reset
      iii. nios_processor.debug_reset_request connected to nios_processor.reset
      iv. nios_processor.data_master connected to nios_processor.debug_mem_slave
      v. nios_processor.instruction_master connected to nios_processor.debug_mem_slave

4. On Chip Memory
   A. Search in the top of the ip catalog to the left for "on chip" and selest the "On-Chip Memory (RAM or ROM)"
   B. Add this by double clicking it
   C. In the pop up select the following
      i. Type set to RAM (writable)
      ii. Block Type set to AUTO
      iii. Data width set to 32
      iv. For the DE2-115 Total memory size 20400bytes (this is different per development board, consult your boards documentation)
      v. Have checked "initialize memory content"
      vi. Click finish
   D. Rename the component to onchip_memory
   E. Make the following connections
      i. Clock.clk connect to onchip_memory.clk1
      ii. nios_processor.data_master connect to onchip_memory.s1
      iii. nios_processor.instruction_master connect to onchip_memory.s1
      iv. Clock.clk_reset connect to onchip_memory.reset1
      v. nios_processor.debug_reset_request connect to onchip_memory.reset
   F. Set the base address to 0x000
   G. Lock the base address by clicking the lock icon  and making it locked 
   H. In the top navigation go to System --> Assign Base Addresses

It is best practice to set your primary memory (the memeory you plan to run instuctions off to both .data_master and .instruction_master and then set all other memeory just to .data_master

5. Jtag Uart
   A. Search in the top of the ip catalog to the left for "jtag" and select "JTAG UART"
   B. Add this by double clicking it
   C. In the pop up select the following for both Write and Read FIFO
      i. Buffer depth select 64
      ii. irq threshold select 8
      iii. Click finish
   D. Rename the component to jtag_uart
   E. Make the following connections
      i. Clock.clk connect to jtag_uart.clk
      ii. Clock.clk_reset connec to jtag_uart.reset
      iii. nios_processor.data_master connect to jtag_uart.avalon_jtag_slave
      iv. ios_processor.debug_reset_request connect to jtag_uart.reset
   F. Connect the IRQ by clicking on the box in the IRQ column and set it to 8
   G. In the top navigation go to System --> Assign Base Addresses

It is best practice to set the Jtag Uart IRQ to a least priority, you have 32 interrupts 0 being first priority and 32 being least priority

6. System ID or sysID
   A. Search in the top of the ip catalog to the left for "sysid" and select "System ID Peripheral"
   B. Add this by double clicking it
   C. In the pop up create a random hex number as the system ID
      i. Click finish
   D. Rename the component to sysid
   E. Make the following connections
      i. Clock.clk connect to sysid.clk
      ii. Clock.clk_reset connec to sysid.reset
      iii. nios_processor.data_master connect to sysid.control_slave
      iv. ios_processor.debug_reset_request connect to sysid.reset
   F. Connect the IRQ by clicking on the box in the IRQ column and set it to 8
   G. In the top navigation go to System --> Assign Base Addresses

You must have a System ID of something other on 0 for the system to work in the Nios II development tools for eclipse

7. Green LED's PIO
   A. Search in the top of the ip catalog to the left for "pio" and select "PIO (Parallel I/O)"
   B. Add this by double clicking it
   C. In the pop up select the following
      i. Width to 8 bits
      ii. Direction select Output
      iii. Output Port Rest Value 0x00
   D. Rename the component to green_leds
   E. Make the following connections
      i. Clock.clk connect to green_leds.clk
      ii. Clock.clk_reset connec to green_leds.reset
      iii. nios_processor.data_master connect to green_leds.s1
      iv. ios_processor.debug_reset_request connect to green_leds.reset
   F. Export the external connection by double clicking on the export column
      i. Name the export green_leds
   G. In the top navigation go to System --> Assign Base Addresses

8. Save the system
9. In the top navigation go to Generate --> Generate HDL...
   A. Choose Verilog
   B. Uncheck Create timing and resource estimates for third-party EDA synthesis tools
   C. Check Create block symbol file (.bsf)
   D. Select none for create symbol model
   E. Make sure the output path directory is the same as the project directory
   F. Click Generate

Note: Before Generating there must not be any errors or warnings in Qsys

Creating A Basic System In Qsys

The Following is the Qsys System created

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags	Opcode Name
☑		**Clock**	Clock Source							
		clk_in	Clock Input	**clk**	***exported***					
		clk_in_reset	Reset Input	**reset**						
		clk	Clock Output	*Double-click to export*	Clock					
		clk_reset	Reset Output	*Double-click to export*						
☑		**nios_processor**	Nios II Processor							
		clk	Clock Input	*Double-click to export*	**Clock**					
		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]	0x8800	0x8fff			
		custom_instruction_master	Custom Instruction Master	*Double-click to export*						
☑		**onchip_memory**	On-Chip Memory (RAM or ROM)							
		clk1	Clock Input	*Double-click to export*	**Clock**					
		s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk1]	0x0000	0x4faf			
		reset1	Reset Input	*Double-click to export*	[clk1]					
☑		**jtag_uart**	JTAG UART							
		clk	Clock Input	*Double-click to export*	**Clock**					
		reset	Reset Input	*Double-click to export*	[clk]					
		avalon_jtag_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x8000	0x8007			
		irq	Interrupt Sender	*Double-click to export*	[clk]			0		
☑		**green_leds**	PIO (Parallel I/O)							
		clk	Clock Input	*Double-click to export*	**Clock**					
		reset	Reset Input	*Double-click to export*	[clk]					
		s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x8010	0x801f			
		external_connection	Conduit	**green_leds**						
☑		**sysid**	System ID Peripheral							
		clk	Clock Input	*Double-click to export*	**Clock**					
		reset	Reset Input	*Double-click to export*	[clk]					
		control_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x8008	0x800f			