#### **Chapter 3: Hardware and Software Codesign Flow**

#### 3.1 Introduction to Nios II Soft-Core Processor

1) Introduction to the Altera Nios II Soft Processor:

<system cd>\DE2\_115\_tutorials\tut\_nios2\_introduction.pdf

• *focus*: All of the information in this resource is needed for creating systems and should be read carefully, as familiarity will greatly help students in avoiding time consuming mistakes.

Nios II is an embedded processor architecture designed specifically for Altera's FPGA boards. An example of a Nios II processor system could be found on <u>page 11</u> from Altera's Nios II Processor Reference Handbook. When implementing your board there is three different types of CPU's to choose from which are the NIOS II/fast, NIOS II/standard, and NIOS II/economy. The main differences between the CPU's are the balance between performance and cost.



**NOTE:** This figure taken from Altera's Nios II Processor Reference Handbook: <u>http://www.altera.com/literature/hb/nios2/n2cpu\_nii5v1.pdf</u> page 11

#### Figure 2-1. Nios II Processor Core Block Diagram



**NOTE:** This figure taken from Altera's Nios II Processor Reference Handbook: <u>http://www.altera.com/literature/hb/nios2/n2cpu\_nii5v1.pdf</u> page 18

2) Nios II Hardware Development:

http://www.altera.com/literature/tt/tt\_nios2\_hardware\_tutorial.pdf

• *focus:* This resource is an excellent overview of the basic requirements to creating a system using QSys in Quartus II, instantiating the design in the project files, implementation, and then creating the necessary software.

3) Nios II Processor Reference: <u>http://www.altera.com/literature/hb/nios2/n2cpu\_nii5v1.pdf</u>

• *NOTE:* This resource has a lot of detailed information which is not necessary to complete most projects, but it is good to be familiar with document in the case of troubleshooting.

#### 3.2 Code sign Flow





Figure 1–2 shows the Nios II system development flow between hardware and software. This flow consists of three types of development: hardware design steps, software design steps, and system design steps.

**NOTE:** This figure taken from Altera's Nios II Hardware Development Tutorial: http://www.altera.com/literature/tt/tt nios2 hardware tutorial.pdf

#### 3.3 Overview of System Integration Software SOPC Builder and QSys



**NOTE:** This diagram was taken from Altera's Nios II Software Developer's Handbook, http://www.altera.com/literature/hb/nios2/n2sw\_nii5v2.pdf

# System Integration Software

This software allows the designer to marry hardware and software. In order to use the Nios II soft-core processor, a system must be designed using either SOPC builder or QSys (both are accessed from Quartus II-> Menu -> Tools). QSys is a newer version of SOPC builder and it is encouraged that students begin with QSys. This development tool primarily generates the .sopcinfo file which is used in Nios II SBT for Eclipse to create the software project to run on top of the FPGA design, utilizing the Nios II soft-core processor.

After creating a system to suit the students' project needs, "Generation" (synonymous to "Compilation") automatically creates the necessary hardware files for low-level abstraction. A main niosII module is created in this process, which is instantiated from the top-level hardware file. This process is described as *System Integration* 

Although much of the reading presented here applies to SOPC Builder, the information applies also to QSys and an effort should be made to use QSys in place of SOPC Builder.

- Introduction to the Altera SOPC Builder: <system cd>\DE2\_115\_tutorials\tut\_sopc\_introduction\_verilog.pdf
- 2) QSys System Design: <u>http://www.altera.com/literature/tt/tt\_qsys\_intro.pdf</u>

• QSys main reference page: <u>http://www.altera.com/products/software/quartus-</u> ii/subscription-edition/gsys/qts-gsys.html?GSA pos=10&WT.oss r=1&WT.oss=qSys

3) SOPC Builder User Guide: <u>http://www.altera.com/literature/ug/ug\_sopc\_builder.pdf</u>

## **3.4 Introduction to Nios II SBT for Eclipse**

Eclipse allows the user to use the software that was executed by a Nios II processorbased system in an FPGA. The user can configure the FPGA on the development board with the pre-generated Nios II standard hardware system by downloading the FPGA configuration file to the board.

1) Nios II Software Developer's Handbook:

http://www.altera.com/literature/hb/nios2/n2sw\_nii5v2.pdf

**NOTE**: Link is placed here for reference, but is not necessary for review in this stage.

#### Binary Adder Tutorial Using Nios II {Includes video and written instructions}

A link to the video describing the Binary Adder Tutorial:

http://www.youtube.com/watch?v=307zpB2Y\_UA&feature=channel\_video\_title

http://www.youtube.com/watch?v=6ATrkRqraJY&feature=channel\_video\_title

The major steps were:

- 1) Create hardware system in system builder
- 2) Build new system in QSys system
- 3) Instantiate the Nios II module in top level entity
- 4) Add IP variation file
- 5) Adjust .sdc
- 6) Place design on FPGA

7) Develop Software in Nios II SBT for Eclipse.

#### Hardware:

- Clock
- Red LEDs
- Switches
- 7 segment Hex
- LCD

## **NIOS II Binary Adder**

## Step 1: System Builder

- 1) Open DE2\_115\_tools->DE2\_115\_system\_builder to find DE2\_115\_SystemBuilder.exe
- 2) Name the project under Project Name: Binary\_Adder\_Nios



3) Check all Components that you will be using: in this Tutorial we are using CLOCK, LEDx27, 7-Segementx8, Switchx18, and of course the LCD.

erasic DE2-115 System Builder V 1.0.0	Statement of the local division of the local	ALC: NO.	
		System Configuration Project Name:	
		Binary_Adder	
DE2-115 FPGA BO			
Load Setting	Generate	IR Receiver     GPIO Header     IO Voltage: [3.3 V (         Prefix Name:         None         HSMC         IO Voltage: [2.5 V (         Prefix Name:	Cefault)
Save Setting	Exit	None	•

- 4) Click Generate
- 5) Create a directory for your project and then click save



6) To open this project open the .qpf file

# Step 2: Building Qsys System

- 1) Open Qsys under tools tab
- Start by adding a Nios II Processor Core: Under "Component Library"-> Processors -> Nios II Processor -> Add
  - a. Select "Nios II/s"
  - b. Set "Hardware multiplication type" = "None"
  - c. Disable "Hardware divide"
  - d. "Finish"
  - e. Rename Nios to "cpu"
- 3) On-Chip Memory: Under "Component Library"-> Memories and Memory Controllers -> On-Chip -> On-Chip Memory (RAM or ROM)-> Click "Add"
  - a. Block Type list = "Auto"
  - b. Total Memory size = "204800" to specify 2KB of memory
  - c. Do not change any other default settings.
  - d. "Finish"
  - e. Under the "System Contents" tab, right-click the on-chip memory and rename as "onchip\_mem"

- 4) JTAG UART: Component Library -> Interface Protocols -> Serial -> JTAG UART -> Add
  - a. Do not change any default settings
  - b. Rename to "jtag\_uart"
- 5) Interval Timer: Component Library -> Peripherals -> Microcontroller Peripherals -> Interval Timer-> Add
  - a. Under "Hardware Options" set "Presets" to "Full-Featured"
  - b. Do not change any other default settings
  - c. Rename to "sys\_clk\_timer"
- 6) System ID Peripheral: Component Library-> Peripherals -> Debug and Performance -> System ID Peripheral-> Add
  - a. Do not change any default settings
  - b. Rename as "sysid"
- 7) PIO's: Component Library-> Peripherals -> Microcontroller Peripherals -> PIO -> Add
  - a. Under "Basic Settings" enter the value of "4" for the box labeled "Width"
  - b. Do not change any other default settings
  - c. Finish
  - d. Rename as "pio\_led"
  - e. For this example us two "pio\_led"
  - f. Repeat these steps for two "pio\_sw" with 4 bits of width and change to input.
  - g. Repeat these steps for pio\_hex0 through 7 with widths of 7 bits.
- LCD: Component Library-> Peripherals -> Display-> Character LCD -> Add
   a. Finish
- 9) Go to the "Connections" column and connect the following ports: (Figure Below)
  - a. For all the components connect the clock input and outputs to clock\_50
  - b. For all the components connect the Avalon memory mapped slave to the On-chip memory AMMS.
  - c. Open the **Nios II processor named CPU** and change the reset vector and exception vectors to onchip\_memory2
- 10) Go to the "Export" column and connect the following ports:
  - a. Click on "click to export" on the external connection row to activate connection for all of the led's, switches and 7 segment display.
  - b. Click on "click to export" on the external row for the LCD
- 11) Under Generation click generate
  - a. Save as "Nios"
  - b. Once generation is complete coping code from HDL example

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ
	and the first first the	E clk 0	Clock Source					
		clk in	Clock Input	clk				
	clk_0.clk	clk in reset	Reset Input	reset				
		< clk	Clock Output	Click to export	clk 0	Assign	Dana	
		< clk_reset	Reset Output	Click to export	-	Assign	Dase	
		onchip_memory2	On-Chip Memory (RAM or ROM)	Clock Input		Addres	ses	
		→ clk1	Clock Input	Click to export	clk_0			
1		→ s1	Avalon Memory Mapped Slave	Avalon MMS	[clk1]	0x00040	000 0x00071ff	f
		> reset1	Reset Input	Click to export	[clk1]			
		🗆 сри	Nios II Processor	Reset Input				
		→ clk	Clock Input	Click to export	clk_0			
	∏●┼┼┝───	reset_n	Reset Input	Click to export	[clk]			
		< data_master	Avalon Memory Mapped Master	Click to export	[clk]	I	RQ 0 II	RQ 31
		< instruction_master	Avalon Memory Mapped Master	Click to export	[clk]		_	
		< jtag_debug_module_re	. Reset Output	Click to export	[clk]			<b>- I</b> II
		jtag_debug_module	Avalon Memory Mapped Slave	Click to export	[clk]	0x00080	800 0x00080ff	f
	×	< custom_instruction_m	Custom Instruction Master	Click to export				
		⊟ jtag_uart	JTAG UART	2010-00740 03	100			
		> clk	Clock Input	Click to export	clk_0			0
		> reset	Reset Input	Click to export	[clk]			- 1 4
		avalon_itag_slave	Avalon Memory Mapped Slave	Click to export	[clk]	= 0x00081	0f0 0x000810f	7 ▶—10
			Character LCD					-
		✓ CIK	Ext	ernal Connections	CIK_U			
		reset	Reset input	ci ili connections	[CIK]			
		control_slave	Avaion memory mapped Slave	led external	[Cik]	0x00081	020 00008102	I
			Interval Timer	icd_external				_
		Sys_cik_timer	Clock Input	Click to export	olk 0			
			Reset Input	Click to export	Icik]			1
		> s1	Avalon Memory Manned Slave	Glick to export	[clk]	- 0x00081	000 0x0008101	न र्म्न
		E sysid asys	System ID Peripheral		Lond			-
	•	> clk	Clock Input	Click to export	clk 0			
1		> reset	Reset Input	Click to export	[clk]			
		control_slave	Avalon Memory Mapped Slave	Click to export	[clk]		0f8 0x000810f	f
1		□ pio_led	PIO (Parallel I/O)					
20.000		> clk	Clock Input	Click to export	clk_0			
	∏ ● ┶┼ �	→ reset	Reset Input	Click to export	[clk]			
		→ s1	Avalon Memory Mapped Slave	Glick to export	[clk]		030 0x0008103	f
		external_connection	Conduit Endpoint	pio_led_external_conner	C			
		pio_led2	PIO (Parallel I/O)					
		> clk	Clock Input	Click to export	clk_0			
		> reset	Reset Input	Click to export	[clk]	22	5	
		→ s1	Avalon Memory Mapped Slave	Click to export	[clk]	0x00081	0e0 0x000810e	f
		> external_connection	Conduit Endpoint	pio_led2_external_conne	B			-
		E pio_sw	PIO (Parallel VO)					
		CIK	Clock Input	Click to export	CIK_0			
		reset	Augles Manage Managed Street	Click to export	[CIK]		0.10 0.00000000	7
		evternal connection	Conduit Endpoint	pio sw external connor	CIKJ	= 0x00081	040 0x0008104	-
		External_connection		pio_sw_external_conner				-
			Clock Input	Click to export	clk 0			
			Reset Input	Click to export	[clk]			
		\$ \$1	Avalon Memory Mapped Slave	Click to export	[ck]	0x00081	0c0 0x000810c	F
			The second s	Stren is shallen	19900	0400001	000 1040000100	

# **Step 2: Quartus HDL Connections**

- 1) Add IP Variation File: Menu bar: Assignments -> Settings
  - a. Under "Category" -> "Files" -> (...) Browse -> Choose script files for type to find(\*.tcl, \*.sdc, \*.qip)
  - b. Locate and choose the file nios2/synthesis/nios.qip
  - c. Add to project, click okay and close
- Copy code under structural coding in Quartus (Code located in the Codes folder under Binary\_Adder\_Quartus)
  - a. Notice LCD\_BLON is set to 1'b1;
  - b. Notice LCD\_ON is set to 1'b1;
  - c. Notice all connections in parenthesis



- 3) Compile and Run
  - a. Compile and Run

## Step 3: Develop the Software for Nios II SBT for Eclipse

- 1) This step relies on the .sopcinfo file created when generating the Qsys system
- 2) Open Nios II SBT for Eclipse
  - a) Indicate workspace as your project directory, and create a new file called "Software" and click "Okay"
  - b) Set perspective to Nios II: Menu -> Window -> Open Perspective -> Other -> Nios II
  - c) Menu -> File -> New -> Nios II Application and BSP from Template



- i) Under "Target Hardware Information" select file <directory>\nios.sopcinfo
- ii) Under "Application Project" type "Binary Adder" as "Project Name"
- iii) Under "Project Template" select "helloWorld"
- iv) Click "Finish"

is li Sonware Examples		
ease specify a .sopcinfo file		
Target hardware information SOPC Information File name:		
CPU name:	•	
Application project		
Project name:		
Vise default location		
Project location:		
Project template		
Templates	Template description	
	Helle World prints 'Helle from Nies II' to STDOUT	
Blank Project	Helio wond prints Helio nom Nios 1 to 310001.	A
Blank Project Board Diagnostics Count Binary	This example runs with or without the MicroC/OS-II RTOS	Â
Blank Project Board Diagnostics Count Binary Hello Freestanding	This example runs with or without the MicroC/OS-II RTOS and requires an STDOUT device in your system's hardware.	Î
Blank Project Board Diagnostics Count Binary Hello Freestanding Hello MicroC/OS-II	This example runs with or without the MicroC/OS-II RTOS and requires an STDOUT device in your system's hardware.	
Blank Project Board Diagnostics Count Binary Hello Freestanding Hello MicroC/OS-II Hello World Hello World Small	This example runs with or without the MicroC/OS-II RTOS and requires an STDOUT device in your system's hardware. For details, click Finish to create the project and refer to the readmost be file in the normed directory.	
Blank Project Board Diagnostics Count Binary Hello Freestanding Hello Wirdd Hello World Small Memory Test	This example runs with or without the MicroC/OS-II RTOS and requires an STDOUT device in your system's hardware. For details, click Finish to create the project and refer to the readme.txt file in the project directory.	E
Blank Project Board Diagnostics Count Binary Hello Freestanding Hello MicroC/OS-II Hello World Hello World Small Memory Test Memory Test Small	This example runs with or without the MicroC/OS-IRTOS and requires an STDOUT device in your system's hardware. For details, click finish to create the project and refer to the readme.txt file in the project directory. The BSP for this template is based on the Altera HAL	
Blank Project Board Diagnostics Count Binany Hello Freestanding Hello MicroC/OS-II Hello World Small Memory Test Memory Test Small Simple Socket Server Simple Socket Server	This example runs with or without the MicroCOS-II RTOS and requires an STDOUT device in your system's hardware. For details, click Finish to create the project and refer to the readmetat file in the project directory. The BSP for this template is based on the Altera HAL operating system.	E
Blank Project Board Diagnostics Count Binary Hello Freestanding Hello MicroC/05-II Hello World Small Memory Test Memory Test Small Simple Socket Server Simple Socket Server (RGMII) Web Server	This example runs with or without the MicroC/OS-II RTOS and requires an STDOUT device in your system's hardware. For details, click Finish to create the project and refer to the readmet.tt file in the project directory. The BSP for this template is based on the Altera HAL operating system. For information about how this software example relates to	
Blank Project Board Diagnostics Count Binany Hello Frestanding Hello MicroC/OS-II Hello World Hello World Memory Test Memory Test Memory Test Small Simple Socket Server Web Server Web Server (RGMII)	This example runs with or without the MicroC/OS-II RTOS and requires an STDOUT device in your system's hardware. For details, click Finish to create the project and refer to the readme.txt file in the project directory. The BSP for this template is based on the Altera HAL operating system. For information about how this software example relates to	
Blank Project Board Diagnostics Count Binary Hello Freestanding Hello MicroC/OS-II Hello World Small Memory Test Memory Test Small Simple Socket Server Simple Socket Server Web Server (RGMII) Web Server (RGMII)	This example runs with or without the MicroCOS-II RTOS and requires an STDOUT device in your system's hardware. For details, click Finish to create the project and refer to the readmet.tt file in the project directory. The BSP for this template is based on the Altera HAL operating system. For information about how this software example relates to	

3) Include C++ code (Code located in the Codes folder under Binary\_Adder\_Nios2)

```
#include <stdio.h>
#include <stdlib.h>
#include <stdlib.h>
#include "system.h"
#include "altera_avalon_pio_regs.h"
           void lcd display(int a, int b);
          int main()
                int value, value2;
                };;
                 while(1){
              while(i)(
value= IORD_ALTERA_AVALON_PIO_DATA(PIO_SW_BASE);
value2= IORD_ALTERA_AVALON_PIO_DATA(PIO_SW2_BASE);
IOWR_ALTERA_AVALON_PIO_DATA(PIO_LED_BASE,value);
IOWR_ALTERA_AVALON_PIO_DATA(PIO_LED2_BASE,value2);
              IOWR_ALTERA_AVALON_PIO_DATA(PIO_HEX6_BASE, segments[value%10]);
IOWR_ALTERA_AVALON_PIO_DATA(PIO_HEX7_BASE, segments[value/10]);
              IOWR_ALTERA_AVALON_PIO_DATA(PIO_HEX4_BASE, segments[value2%10]);
IOWR_ALTERA_AVALON_PIO_DATA(PIO_HEX5_BASE, segments[value2/10]);
              IOWR_ALTERA_AVALON_PIO_DATA(PIO_HEX0_BASE, segments[(value+value2)%10]);
IOWR_ALTERA_AVALON_PIO_DATA(PIO_HEX1_BASE, segments[(value+value2)/10]);
              lcd_display(value2,value);
                }
                 //----
              return 0;

void lcd_display(int a, int b){
    FILE *pLCD;
    char text[32];
    sprintf(text, " $-2.2i + $-2.2i = $-2.2i \r", a,b,a+b);

                 pLCD = fopen(LCD_NAME, "w");
                  if(pLCD){
                     fulce.plcD);
folose(pLCD);
           }else{
                 printf("Failed to Display\n");
           }
4) Build project
```

5) Run as Hardware

#### **Nios II System Development Flow**

