Designing & Simulating the USB20HR System from Scratch

Tutorial

System Level Solutions, Inc. (USA)
14100 Murphy Avenue
San Martin, CA 95046
(408) 852 - 0067

http://www.slscorp.com

IP Core Version: 2.2
Document Version: 2.3
Document Date: June 2008

IP Usage Note

tt_ipusb20hr_hw_sim_2.3

About this Tutorial

Introduction

This tutorial accompanies all the steps for creating and simulating USB20HR system from scratch. Also it shows how to create, compile, debug, run and simulate a C/C++ program using the Nios II IDE.

Table below shows the revision history of the tutorial.

Version	Date	Description
2.3	May 2008	- Added note for figure 5.2 - Minor modification in chapter 5 - Updated document as per the new directory structure
2.2	March 2008	Changed Table 5.1 Pin assignments for the Nios II Development Kit (2C35)
2.1	February 2008	Combined hardware and simulating the USB20HR system in one tutorial
2.0	November 2007	Updated tutorial as per the USB2.0 IP core version 2.x (USB20HR)
1.2	December 2006	Second Publication of the Designing USB2.0 System from Scratch-Removal of Control Endpoint Registers and Modification of CSR Register of each endpoint.
1.1	May 2006	First Publication of the Tutorial

How to find Information

- The Adobe Acrobat Find feature allows you to search the contents of a PDF file. Use Ctrl + F to open the Find dialog box. Use Shift + Ctrl + N to open to the Go To Page dialog box.
- Thumbnail icons, which provide miniature preview of each page, provide a link to the pages. Links allow you to jump to related information.

How to Contact SLS

For the most up-to-date information about SLS products, go to the SLS worldwide website at http://www.slscorp.com. For additional information about SLS products, consult the source shown below.

Information Type	E-mail
Product literature services, SLS literature services, Nontechnical customer services, Technical support.	support@slscorp.com

Typographic Conventions

The tutorial uses the typographic conventions as shown below.

Visual Cue	Meaning
Bold Type with Initial Capital letters	All headings and Sub heading Titles in a document are displayed in bold type with initial capital letters; Example: **Designing & Compiling, Creating a Quartus II Project etc.**
Bold Type with Italic Letters	All Definitions, Figure and Table Headings are displayed in Italics. Examples: **Figure 1-1. Create a new System Dialog box.**
1. 2.	Numbered steps are used in a list of items, when the sequence of items is important. Such as steps listed in procedure.
•	Bullets are used in a list of items when the sequence of items is not important.
☞	The hand points to special information that requires special attention
⚠ CAUTION	The caution indicates required information that needs special consideration and understanding and should be read prior to starting or continuing with the procedure or process.
⚠ WARNING	The warning indicates information that should be read prior to starting or continuing the procedure or processes.
👣	The feet direct you to more information on a particular topic.

Contents

1. Introduction

This tutorial walks you through the hardware & software development flow. It shows you how to use SOPC Builder and the Quartus II software to create, process and simulate your own USB20HR Device IP core & Nios II system

This tutorial is basically for users who are new to the Nios II processor as well as users who are new to the concept of using embedded systems in FPGA's. This tutorial guides you through the steps necessary to create, compile and simulate a USB20HR IP system design. This simple, single-master Nios II system consist of a Nios II embedded processor and associated system peripherals and interconnections for use with the input & output hardware available on cyclone board.

This tutorial is divided into the following four sections:

- ***'Designing & Compiling'*** - Teaches you how to use SOPC builder to create the Nios II & USB20HR system module in block design file (.bdf) and how to compile the Nios II design using the Quartus II Compiler.
- ***'Programming'*** - Teaches you how to use the Quartus II Programmer and the Byte Blaster / USB Blaster cable to configure the FPGA on the cyclone board, so that the FPGA can be configured with your design whenever power is applied to the board.
- ***'Running the Software on Your Nios II System'*** - Provides the instructions for running software on your Nios II system using the Nios II integrated development environment (IDE).
- ***Simulating the USB20HR Application on Nios II System*** - Provides the steps, which will guide you how to build a system for the Nios II processor and simulate it with ModelSim-Altera 6.1g simulation tool.

Hardware & Software Requirements

The user will require following hardware & software

- A PC running with Win 2000/XP OS
- Nios II embedded processor version 7.2 or higher
- The Quartus II software version 7.2 or higher
- Nios II Development Kit (2C35)
- Altera USB Blaster/Byte Blaster download cable
- USB 2.0 Snap On Board from System Level Solutions (SLS)
- ModelSim -Altera 6.1g

2. Designing & Compiling

To use the instructions in this section, you need to be familiar with the Quartus II software interface, specifically toolbars. Refer to Quartus II help for more information about using the Quartus II software.

Creating a Quartus II Project

Here are the steps to create a new Quartus II project:

1. **Open** the Quartus II.
2. Choose **File>New Project Wizard**.
3. Click **Next**.
4. Select Working Directory of the Project, Name of the project as **‘usb20hr_pi_n2_2c35’** & top-level entity as “**usb20hr_pi_n2_2c35”.** Then click **Next**.
5. Click **Next**.
6. Click **Next**.
7. Select the family as **‘Cyclone II’**. Select Specify device selected in ‘Available device list’ radio button in Target Device option.
8. This design from scratch is given for Nios II Development Kit (2C35), so you have to select the FPGA for cyclone board (which is Cyclone II **EP2C35F672C6**), so under Filters / Speed Grade select **6** Then under Available devices: highlight **‘EP2C35F672C6’**. Click **Next**. If you want to design a system for the board other then **Nios II Development Kit (2C35)** then you will have to select device whichever is there on your board.
9. Click **Finish**.

After creating a new project in Quartus II, its now time to generate a system into SOPC builder.

Start SOPC Builder

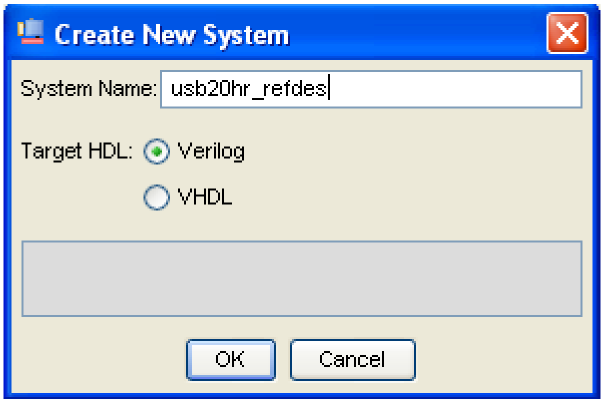
SOPC builder is a software tool that allows you to create a fully functioning, custom-embedded micro controller called the Nios II system module. A complete Nios II system module contains a Nios II embedded processor and its associated peripherals.

To start SOPC builder, perform the following steps:

1. Choose **SOPC Builder** from the **Tools** menu. SOPC Builder displays the **Create New System** dialog box.
2. Type **'usb20hr_refdes'** in the **System Name** box**.** See Figure 2-1.
3. Under HDL Language field specify **Verilog.**

   ☞ SOPC Builder generates plain text Verilog HDL or VHDL for all of its native components depending on the language you choose.

Figure 2-1. Create New System

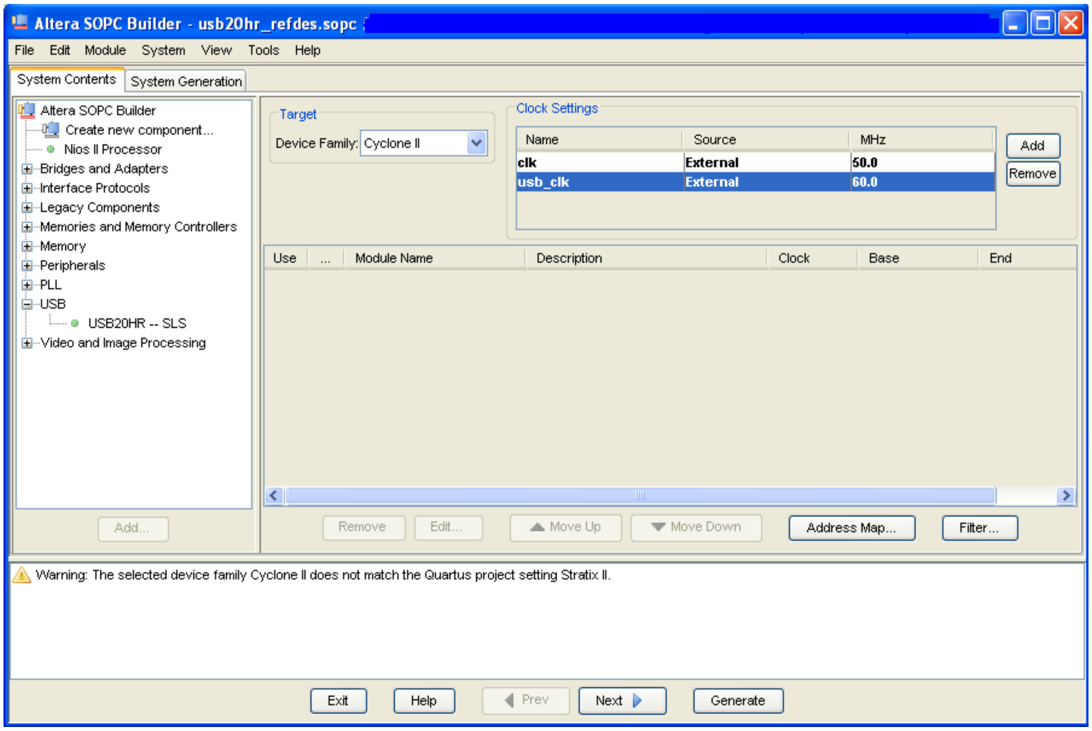


System Name: usb20hr_refdes

Target HDL: [✓ Verilog] VHDL

4. Click **OK.**
5. The Altera **SOPC Builder - usb20hr_refdes** system window appears and the **System Contents tab** is displayed. You are now ready to set the **speed** and add the **Nios II CPU and peripherals** to your system. The components you will be adding are located in the module pool on the left hand side of the System Content tab. See Figure 2-2.

Figure 2-2. SOPC Builder -System Contents Tab



Device Family: Cyclone II

Name	Source	MHz
clk	External	50.0
usb_clk	External	60.0

Warning: The selected device family Cyclone II does not match the Quartus project setting Stratix II.

Define the System in SOPC Builder

After creating project in SOPC builder, now to define the system in SOPC Builder follow the procedure mentioned below:

Specify Target Hardware Settings

The functionality of the SOPC Builder system depends on the hardware on which it will run. Thus, specifying the target board is the first step in creating a system according to your requirement.

1. Select the **System Clock Frequency** as **50 Mhz**.
2. Select the **usb_clk** as a **60Mhz**. This is because the USB20HR IP needs 60 MHz.

Adding CPU & Peripherals

This section describes adding following modules to the SOPC Builder.

- Nios II 32-bit CPU
- JTAG UART
- USB20HR
- Avalon-MM Tristate Bridge
- SSRAM
- On-chip Memory
- DMA
- PLL
- PIO

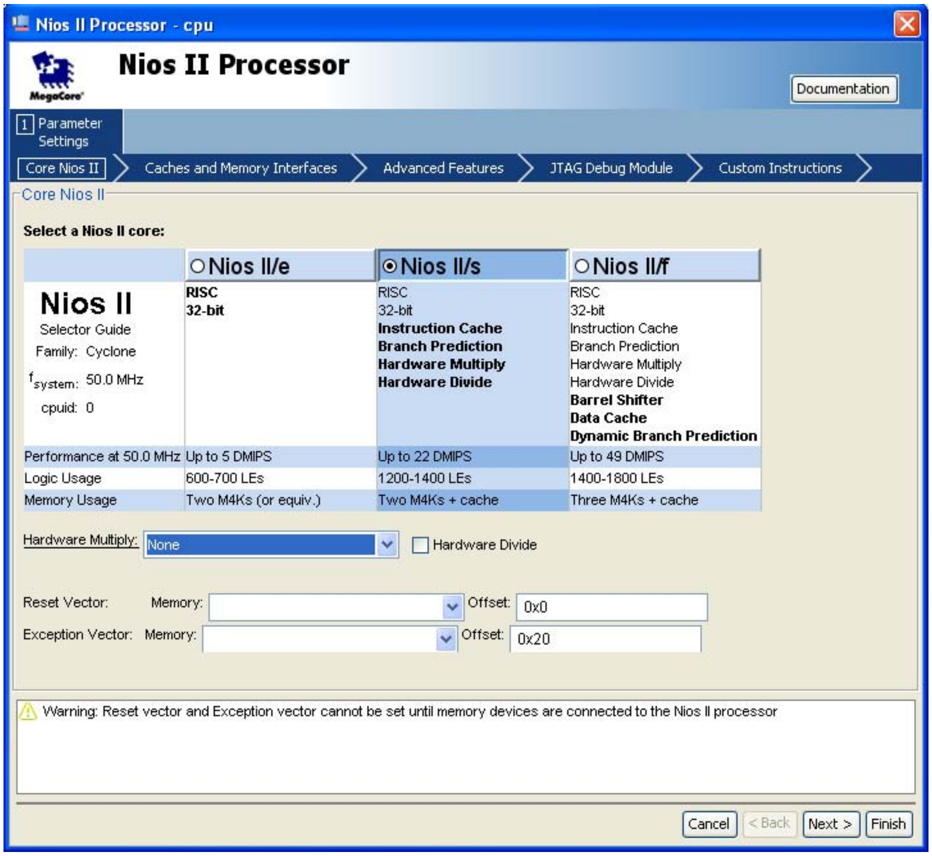
Add the Nios II 32-bit CPU

To add the Nios II 32-bit CPU, named CPU, perform the following steps:

1. Under Avalon Modules, select **Nios II Processor - Altera Corporation**.
2. Double click to **Add.** The Nios II configuration wizard titled **Nios II Processor - cpu displays.**
3. Specify the following options under the **Nios II Core tab**.
   - Select the processor core: **Nios II/s.**
   - Keep Hardware multiply: **None** see Figure 2-3.

☞ You can also select **Nios II/f processor** for better performance.

Figure 2-3. Nios II Processor - CPU Configuration Wizard



Select a Nios II core:

Nios II Selector Guide Family: Cyclone f_{system}: 50.0 MHz cpuid: 0	○ Nios II/e	● Nios II/s	○ Nios II/f
	RISC **32-bit**	RISC 32-bit **Instruction Cache** **Branch Prediction** **Hardware Multiply** **Hardware Divide**	RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide **Barrel Shifter** **Data Cache** **Dynamic Branch Prediction**
Performance at 50.0 MHz	Up to 5 DMIPS	Up to 22 DMIPS	Up to 49 DMIPS
Logic Usage	600-700 LEs	1200-1400 LEs	1400-1800 LEs
Memory Usage	Two M4Ks (or equiv.)	Two M4Ks + cache	Three M4Ks + cache

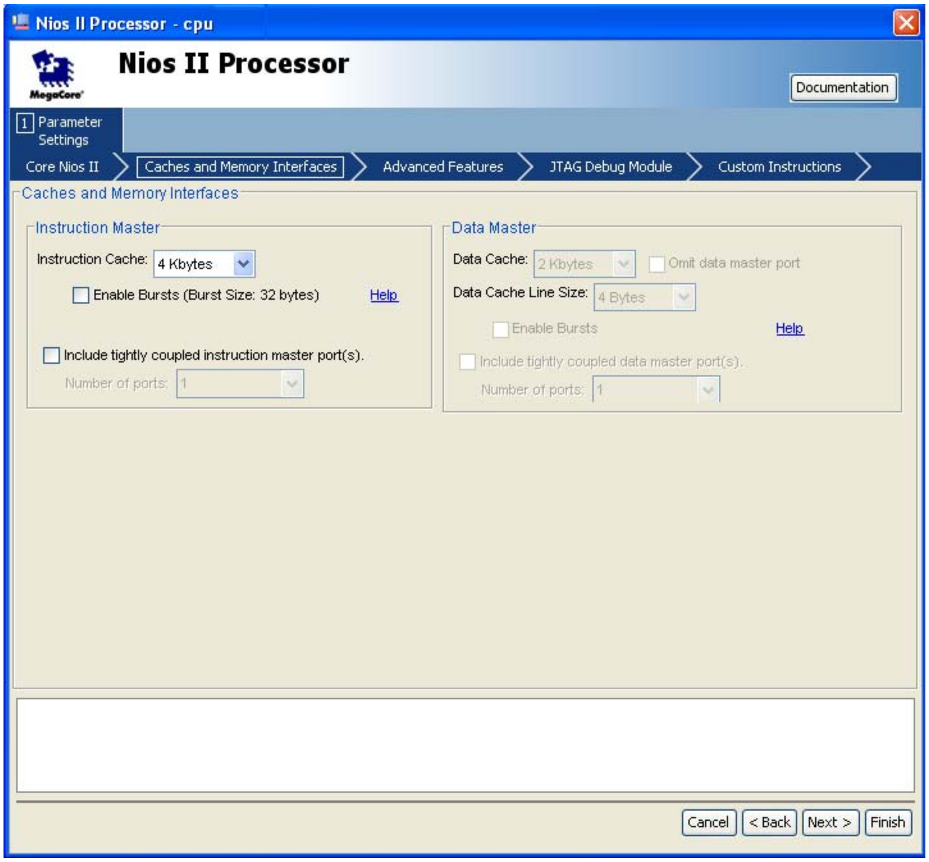
Hardware Multiply: None
☐ Hardware Divide

Reset Vector: Memory: Offset: 0x0
Exception Vector: Memory: Offset: 0x20

Warning: Reset vector and Exception vector cannot be set until memory devices are connected to the Nios II processor

4. Select settings for Nios II core as shown in Figure 2-4.

Figure 2-4. Nios II Core Options - Caches & Tightly Coupled Memories Configuration Wizard



Caches and Memory Interfaces

Instruction Master

Instruction Cache: 4 Kbytes

Enable Bursts (Burst Size: 32 bytes)

Include tightly coupled instruction master port(s).

Number of ports: 1

Data Master

Data Cache: 2 Kbytes

Omit data master port

Data Cache Line Size: 4 Bytes

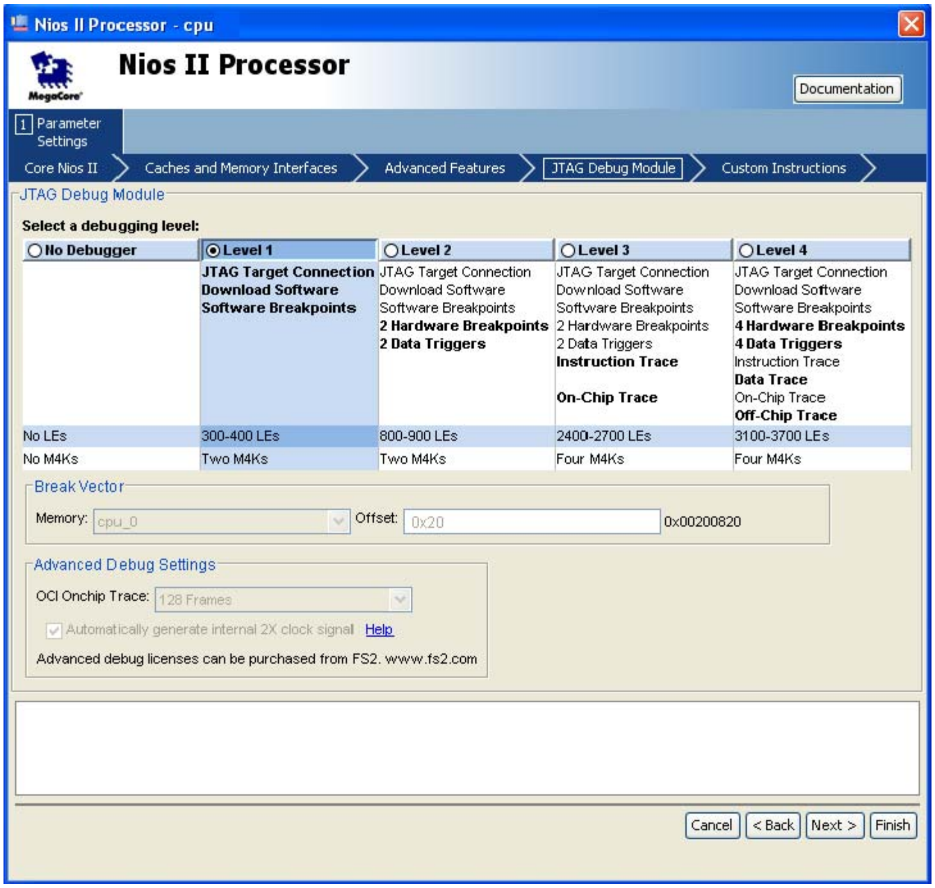
Enable Bursts

Include tightly coupled data master port(s).

Number of ports: 1

5. Click the **JTAG Debug Module** tab and choose the **highlighted** tab as shown in Figure 2-5.

Figure 2-5. Nios II Options - JTAG Debug Module



Nios II Processor - cpu

Nios II Processor

Documentation

1 Parameter Settings

Core Nios II > Caches and Memory Interfaces > Advanced Features > JTAG Debug Module > Custom Instructions

JTAG Debug Module

Select a debugging level:

No Debugger	[✓ Level 1]	Level 2	Level 3	Level 4
	JTAG Target Connection **Download Software** **Software Breakpoints**	JTAG Target Connection Download Software Software Breakpoints **2 Hardware Breakpoints** **2 Data Triggers**	JTAG Target Connection Download Software Software Breakpoints 2 Hardware Breakpoints 2 Data Triggers **Instruction Trace** **On-Chip Trace**	JTAG Target Connection Download Software Software Breakpoints **4 Hardware Breakpoints** **4 Data Triggers** Instruction Trace **Data Trace** On-Chip Trace **Off-Chip Trace**
No LEs	300-400 LEs	800-900 LEs	2400-2700 LEs	3100-3700 LEs
No M4Ks	Two M4Ks	Two M4Ks	Four M4Ks	Four M4Ks

Break Vector

Memory: cpu_0 Offset: 0x20 0x00200820

Advanced Debug Settings

OCI Onchip Trace: 128 Frames

[✓] Automatically generate internal 2X clock signal Help

Advanced debug licenses can be purchased from FS2. www.fs2.com

Cancel | < Back | Next > | Finish

6. Click **Finish** & you are returned to the main SOPC Builder window.

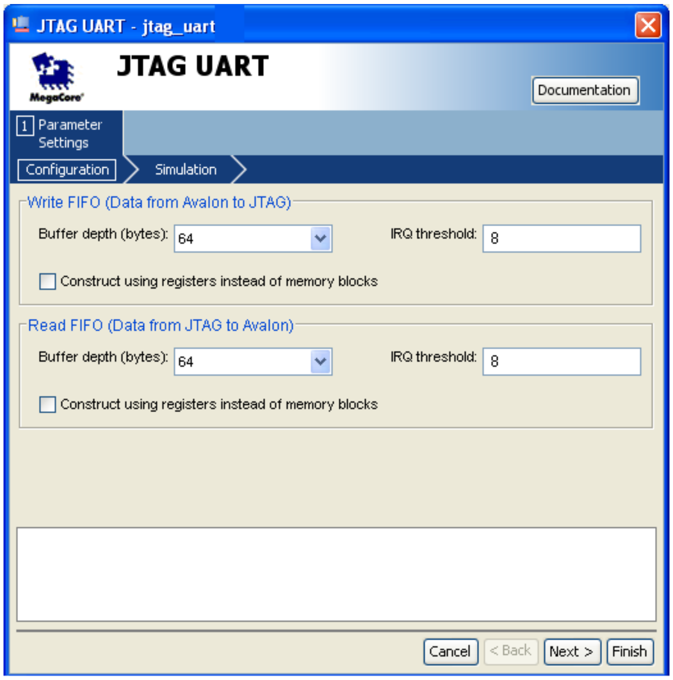
☞ You will see errors in the SOPC Builder message display. However, the issue causing the errors are resolved when the rest of the elements are added to the design. At this stage error messages can be safely ignored.

Add the JTAG UART

This Nios II design includes one JTAG UART interface component, which is added to reduce the number of connections necessary to ‘talk’ to the Nios II. To add this peripheral perform the following steps:

1. Select **Interface Protocol>Serial> JTAG UART** and click **Add.** The **JTAG UART - jtag_uart wizard** displays as shown in Figure 2-6.

Figure 2-6. JTAG UART Configuration Wizard



Write FIFO (Data from Avalon to JTAG)

Buffer depth (bytes): 64
IRQ threshold: 8
Construct using registers instead of memory blocks

Read FIFO (Data from JTAG to Avalon)

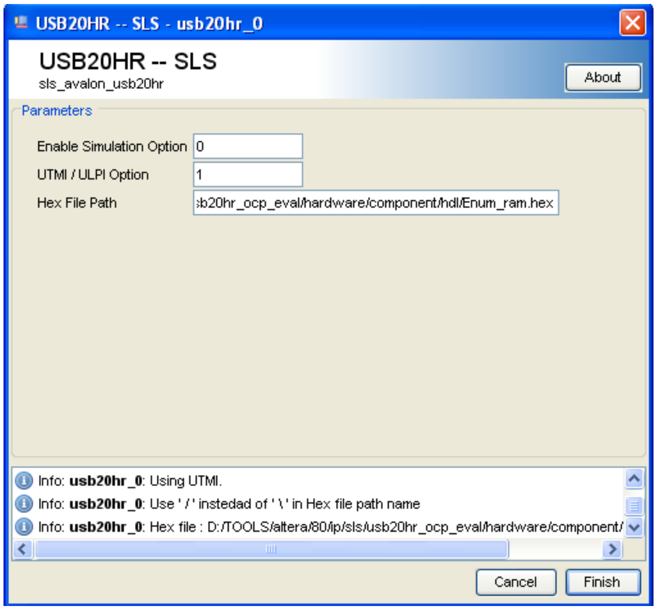
Buffer depth (bytes): 64
IRQ threshold: 8
Construct using registers instead of memory blocks

2. Accept the default options by clicking **Finish**.

Add the USB20HR

To add USB20HR, double click on **USB >USB20HR -- SLS.** IP Megawizard opens. See Figure 2-7.

Figure 2-7. USB20HR IP Tool-bench


USB20HR -- SLS - usb20hr_0

USB20HR -- SLS
sls_avalon_usb20hr

Parameters

Enable Simulation Option: 0
UTMI / ULPI Option: 1
Hex File Path: :b20hr_ocp_eval/hardware/component/hdl/Enum_ram.hex

```
Info: usb20hr_0: Using UTMI.
Info: usb20hr_0: Use ' / ' instedad of ' \ ' in Hex file path name
Info: usb20hr_0: Hex file : D:/TOOLS/altera/80/ip/sls/usb20hr_ocp_eval/hardware/component/
```

1. Under **Enable Simulation Option:**
   - Type **0** while you are Designing the USB20hr System from Scratch.
   - Type **1** while simulating the USB20hr System.
2. For UTMI, type **1** and for ULPI type **0** under **UTMI/ULPI** Option.
3. Under Hex File Path, type the Hex file path. The Hex file is located at *<USB20HR IP installation path>\hardware\component\hdl.*
4. Click **Finish.** You will return to SOPC Builder System content tab window.
5. Select **USB_clk** under **clk** option of USB20HR component in SOPC Builder

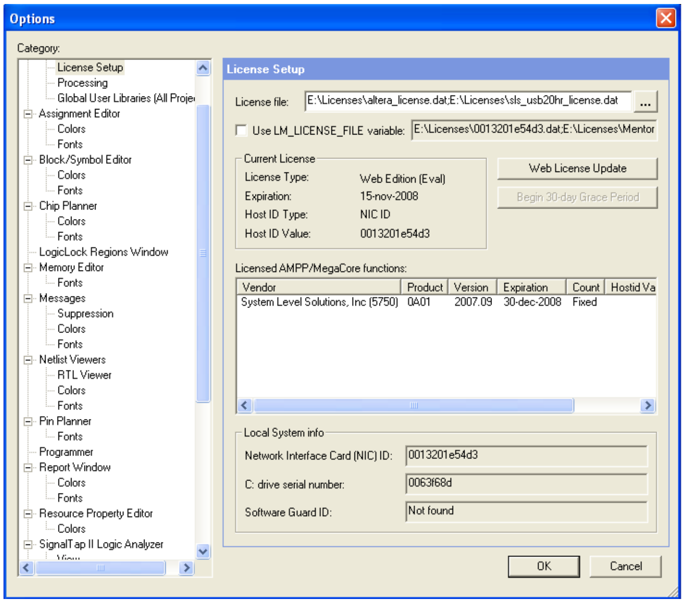
After the IP is added, please note that this IP is encrypted one. To use this IP follow the steps below:

- Open the **Quartus II** > **Tool** > **License Setup…**
- Then under License Setup/ License file edit combo box. Add semicolon & type the **license file path of USB20HR IP**.

☞ Make sure that you have given the license file extension ". dat".

6. Then Click **OK.** & reopen the **Quartus II > Tool > License Setup**… You will see as displayed in Figure 2-8.

Figure 2-8. License Setup



License file: E:\Licenses\altera_license.dat;E:\Licenses\sls_usb20hr_license.dat

Use LM_LICENSE_FILE variable: E:\Licenses\0013201e54d3.dat;E:\Licenses\Mentor

Current License

License Type:	Web Edition (Eval)
Expiration:	15-nov-2008
Host ID Type:	NIC ID
Host ID Value:	0013201e54d3

Licensed AMPP/MegaCore functions:

Vendor	Product	Version	Expiration	Count	Hostid Va
System Level Solutions, Inc (5750)	0A01	2007.09	30-dec-2008	Fixed	

Local System info

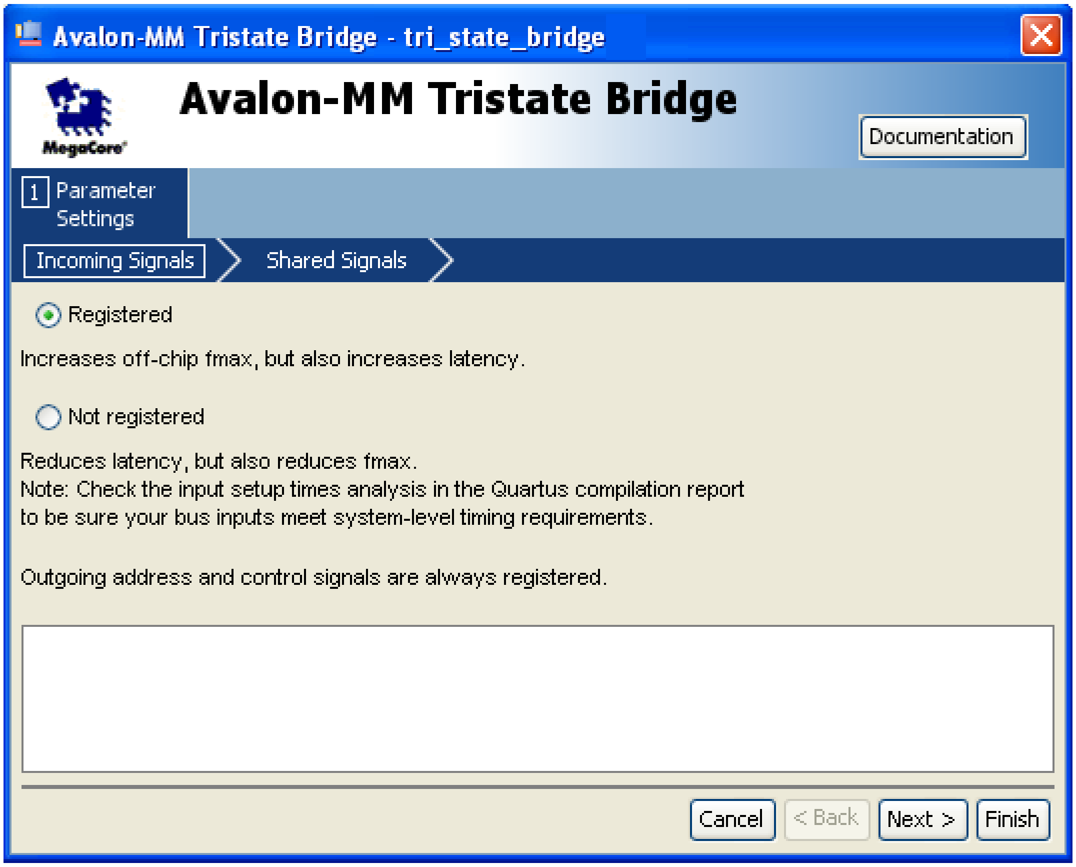
Network Interface Card (NIC) ID:	0013201e54d3
C: drive serial number:	0063f68d
Software Guard ID:	Not found

Add the Avalon-MM Tristate Bridge

For the Nios II system to communicate with memory external to the FPGA on the Nios II Development Kit (2C35), you must add a bridge between the Avalon bus. To add this:

1. Select **Bridges and Adapters>Memory Mapped >** **Avalon-MM Tristate Bridge** and click **Add.** The **Avalon-MM Tristate Bridge - tri_state_bridge** wizard displays. See Figure 2-9.

Figure 2-9. Avalon Tristate Bridge Configuration Wizard- Incoming Signals Tab



(●) Registered

Increases off-chip fmax, but also increases latency.

( ) Not registered

Reduces latency, but also reduces fmax.
Note: Check the input setup times analysis in the Quartus compilation report
to be sure your bus inputs meet system-level timing requirements.

Outgoing address and control signals are always registered.

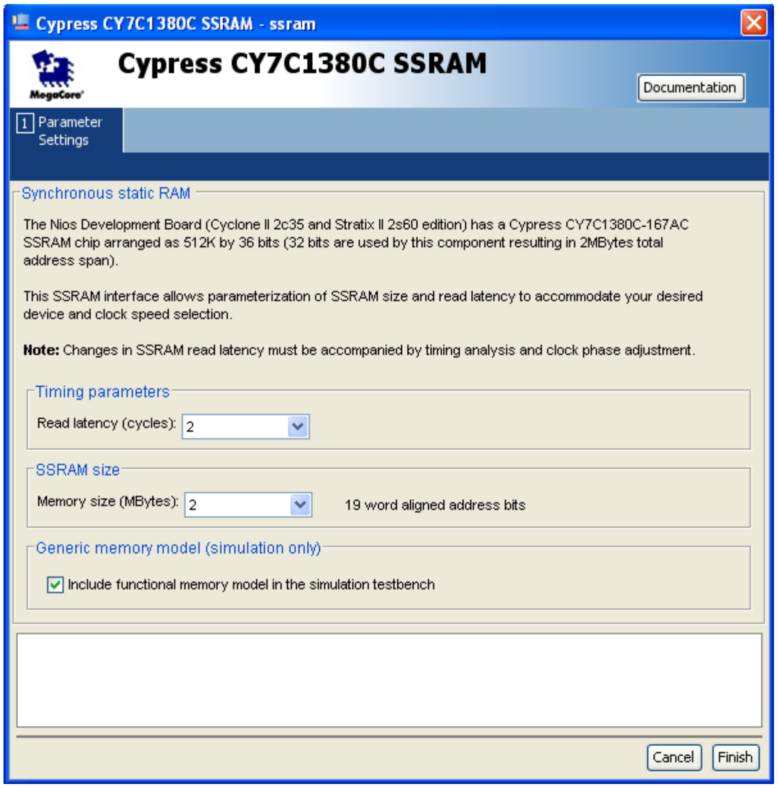
2. Click **Finish.**

Add the SSRAM

Depending on which hardware you are using, user has to select the external SRAM or any memory. To add SRAM perform the following steps:

1. Select **Memories and Memory controller > SRAM>Cypress CY7C1380C SSRAM.** You will see SRAM Configuration wizard as shown in Figure 2-7. Set the following options:
   - Memory Size: **2**
   - Read latency: **2**

Figure 2-10. SSRAM Configuration Wizard



Synchronous static RAM

The Nios Development Board (Cyclone II 2c35 and Stratix II 2s60 edition) has a Cypress CY7C1380C-167AC SSRAM chip arranged as 512K by 36 bits (32 bits are used by this component resulting in 2MBytes total address span).

This SSRAM interface allows parameterization of SSRAM size and read latency to accommodate your desired device and clock speed selection.

Note: Changes in SSRAM read latency must be accompanied by timing analysis and clock phase adjustment.

Timing parameters

Read latency (cycles): 2

SSRAM size

Memory size (MBytes): 2   19 word aligned address bits

Generic memory model (simulation only)

☑ Include functional memory model in the simulation testbench

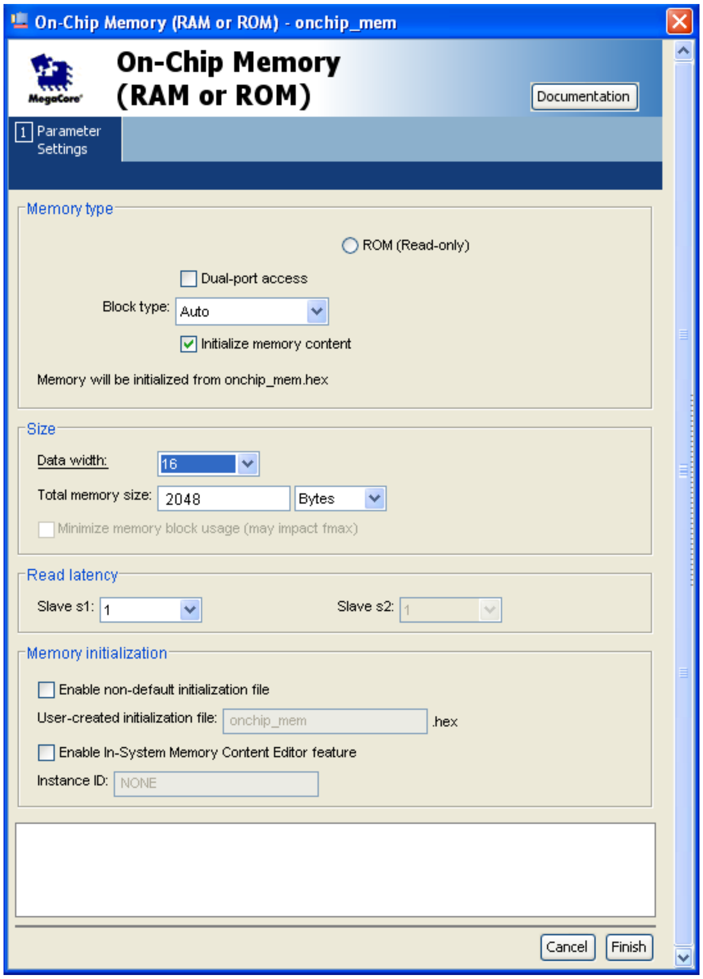
2. Click **Finish.**

Add the On-chip Memory

On-Chip RAM is used as a data storage memory into SOPC builder system. To add on chip memory into SOPC builder system perform following steps.

1. Select **Memories and Memory Controllers -> On -Chip> On -Chip Memory (RAM or ROM)** and then click on **Add**.
2. Change settings as shown in Figure 2-11.

Figure 2-11. On-Chip Memory Configuration Wizard


On-Chip Memory (RAM or ROM) - onchip_mem

On-Chip Memory (RAM or ROM)

Documentation

1 Parameter Settings

Memory type

ROM (Read-only)

☐ Dual-port access

Block type: Auto

☑ Initialize memory content

Memory will be initialized from onchip_mem.hex

Size

Data width: 16

Total memory size: 2048 Bytes

☐ Minimize memory block usage (may impact fmax)

Read latency

Slave s1: 1

Slave s2: 1

Memory initialization

☐ Enable non-default initialization file

User-created initialization file: onchip_mem .hex

☐ Enable In-System Memory Content Editor feature

Instance ID: NONE

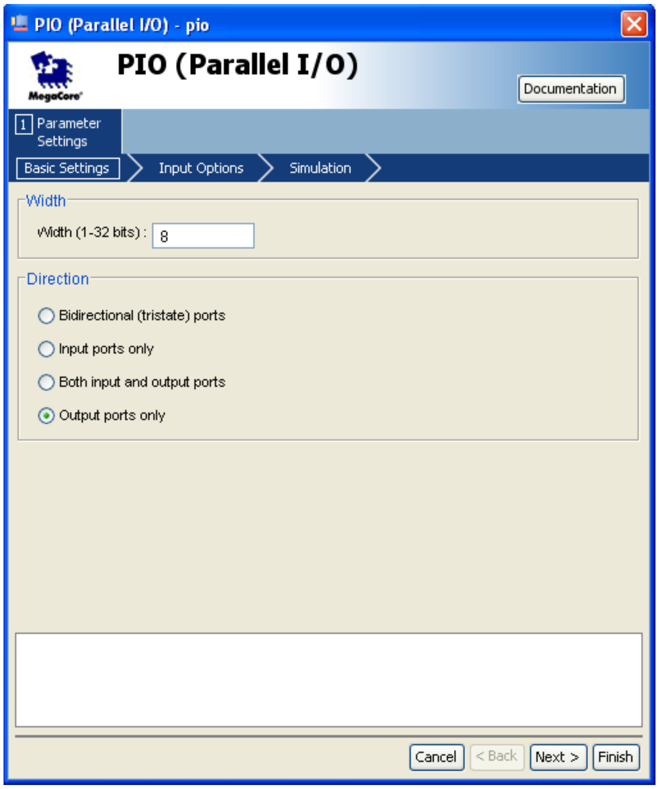
3. Click on **Finish.**

Add the PIO

To provide an interface for the LEDs, add the PIO.

1. Select **PIO**(Parallel I/O) under **Peripherals>Micro Controller Peripheral>PIO** and click **Add.** The **PIO-pio** wizard displays as shown in Figure 2-12.

Figure 2-12. PIO Configuration Wizard


PIO (Parallel I/O) - pio

PIO (Parallel I/O)

Width

Width (1-32 bits) : 8

Direction

( ) Bidirectional (tristate) ports

( ) Input ports only

( ) Both input and output ports

(●) Output ports only

2. Under Basic Settings tab, specify the following options: See
   - Width: **8 bits**
   - Direction: **Output ports** only
3. Click **Finish.** You are returned to the Altera SOPC Builder-nios32 window.

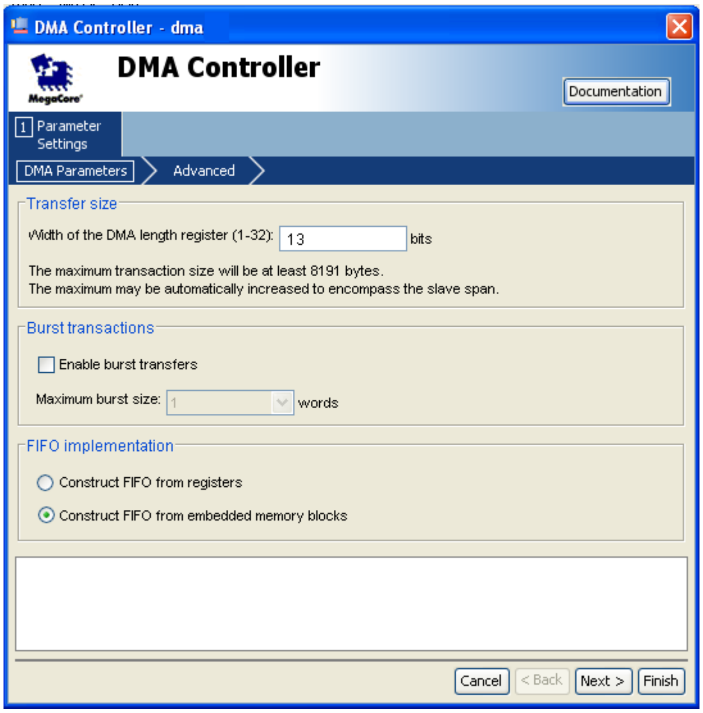
Add the DMA

To access FIFO directly in the core, add DMA. To add DMA follow the steps below:

1. Select **DMA** under **Memories and Memory Controller>DMA> DMA Controller** and click **Add.** The **DMA Controller - dma** wizard displays as shown in Figure 2-13.

2. Under DMA parameter tab, specify following settings:
   - Under Transfer Size, type "**13bit**" in the width of DMA length register box.
   - Under FIFO Implementation, select "**Construct FIFO from embedded memory blocks**".

Figure 2-13. Avalon DMA Controller Configuration Wizard



Transfer size

Width of the DMA length register (1-32): 13 bits

The maximum transaction size will be at least 8191 bytes.
The maximum may be automatically increased to encompass the slave span.

Burst transactions

☐ Enable burst transfers

Maximum burst size: 1 words

FIFO implementation

○ Construct FIFO from registers

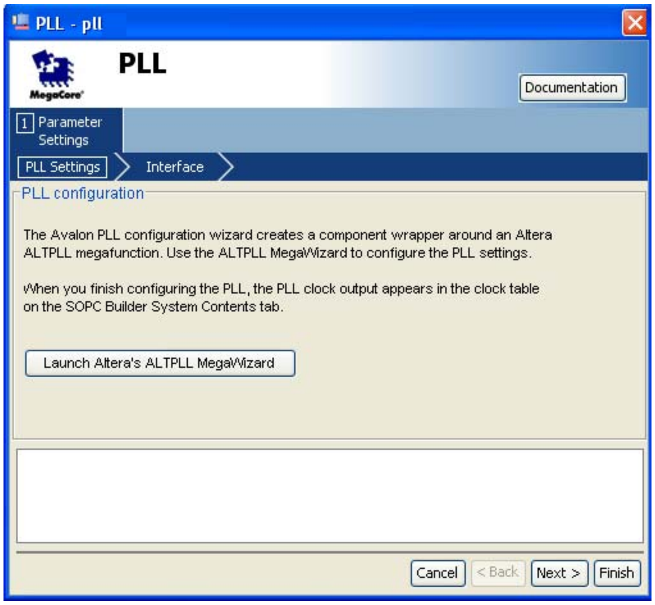
● Construct FIFO from embedded memory blocks

3. Click **Finish**. You will return to SOPC builder system content tab.
4. Connect DMA read and write master with USB20HR and SRAM.
5. Set Clock**: 60 MHZ** (USB2.0 Clock).

Add the PLL

To provide the direct clk to the SSRAM, PLL is used. Select **PLL** under **PLL** and click **Add.** You will see the **PLL Configuration Wizard** as shown in Figure 2-14.

Figure 2-14. PLL Configuration Wizard

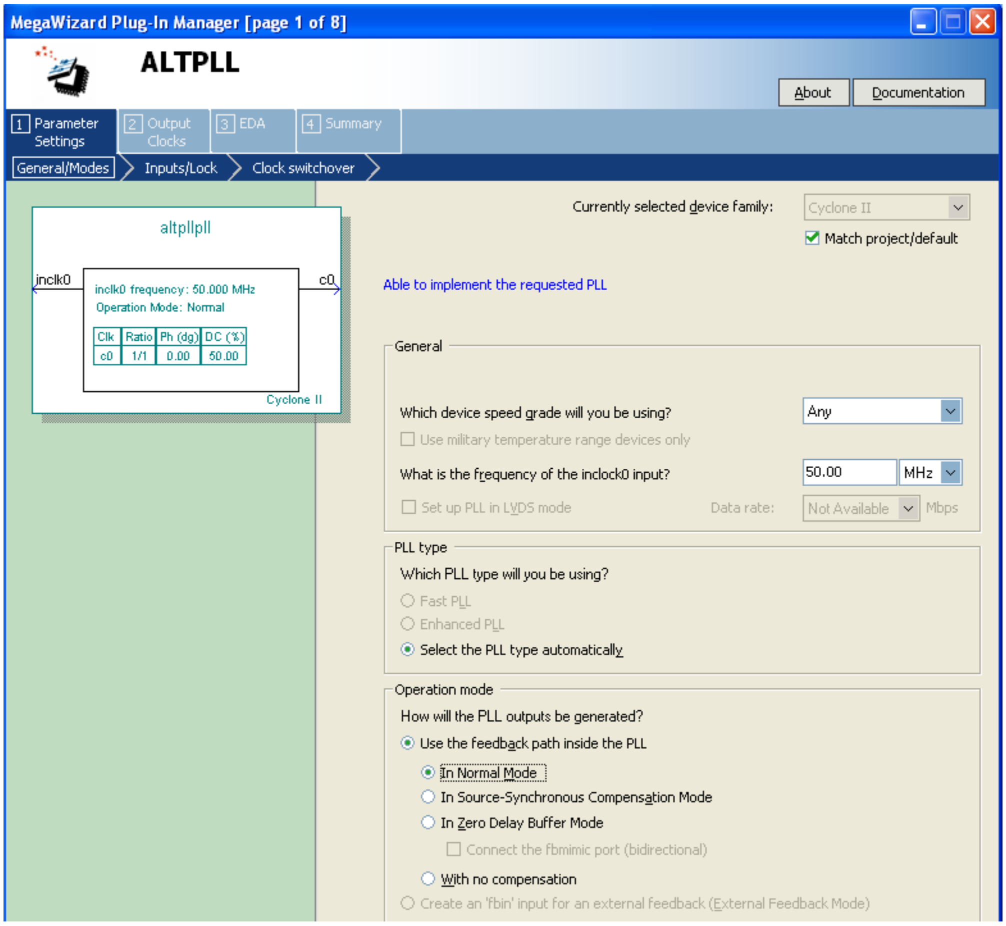


The Avalon PLL configuration wizard creates a component wrapper around an Altera ALTPLL megafunction. Use the ALTPLL MegaWizard to configure the PLL settings.

When you finish configuring the PLL, the PLL clock output appears in the clock table on the SOPC Builder System Contents tab.

1. Click on the **Launch Altera's ALTPLL MegaWizard**. You will see the MegaWizard Plug-In Manager[Page 1] as shown in Figure 2-15.

Figure 2-15.MegaWizard Plug-In Manager [Page 1 of 8]


MegaWizard Plug-In Manager [page 1 of 8]

ALTPLL

About | Documentation

1 Parameter Settings | 2 Output Clocks | 3 EDA | 4 Summary

General/Modes > Inputs/Lock > Clock switchover >

altpllpll

inclk0 — c0

inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	1/1	0.00	50.00

Cyclone II

Currently selected device family: Cyclone II

[✓] Match project/default

Able to implement the requested PLL

General

Which device speed grade will you be using? Any

[ ] Use military temperature range devices only

What is the frequency of the inclock0 input? 50.00 MHz

[ ] Set up PLL in LVDS mode — Data rate: Not Available Mbps

PLL type

Which PLL type will you be using?

( ) Fast PLL
( ) Enhanced PLL
(●) Select the PLL type automatically

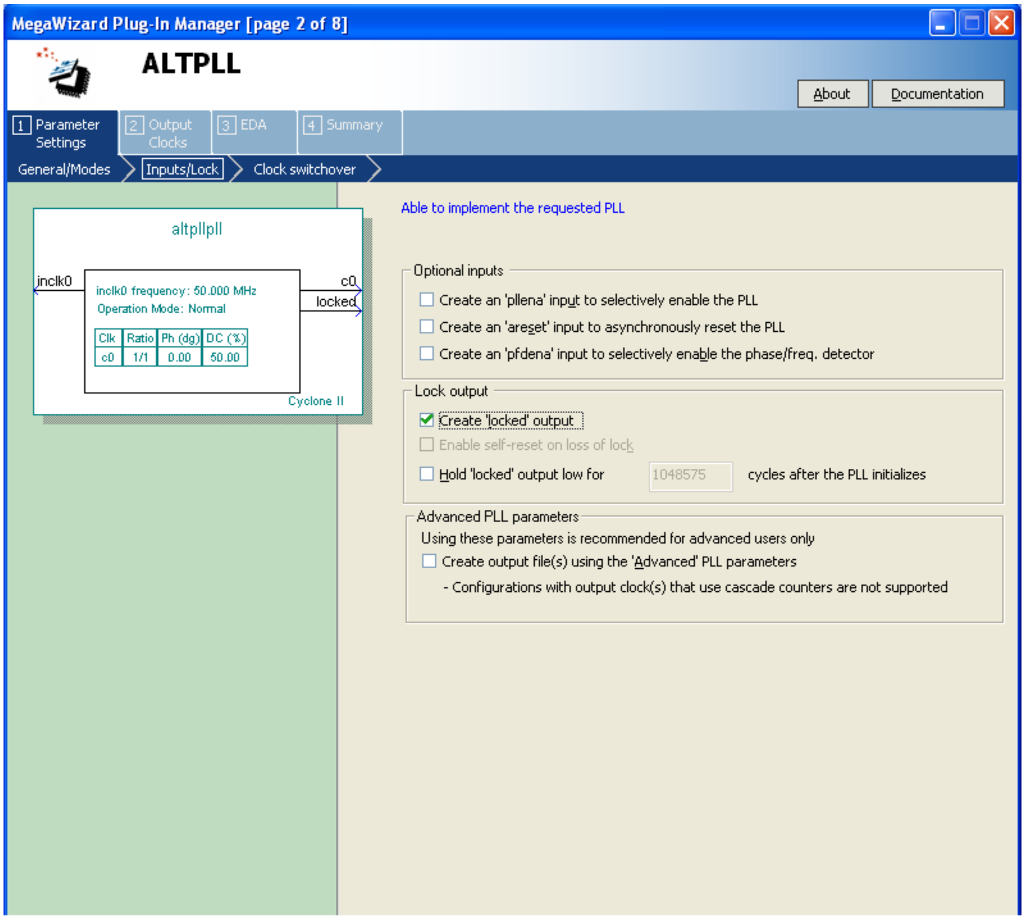
Operation mode

How will the PLL outputs be generated?

(●) Use the feedback path inside the PLL
  (●) In Normal Mode
  ( ) In Source-Synchronous Compensation Mode
  ( ) In Zero Delay Buffer Mode
    [ ] Connect the fbmimic port (bidirectional)
  ( ) With no compensation
( ) Create an 'fbin' input for an external feedback (External Feedback Mode)

2. Type **50** under **What is the frequency of the inclock0 input?**.
3. Keep the default settings as shown in Figure 2-15.
4. Press **Enter.** See Figure 2-17.

Figure 2-16.MegaWizard Plug-In Manager [Page 2 of 8]


MegaWizard Plug-In Manager [page 2 of 8]

ALTPLL

Able to implement the requested PLL

Optional inputs

- ☐ Create an 'pllena' input to selectively enable the PLL
- ☐ Create an 'areset' input to asynchronously reset the PLL
- ☐ Create an 'pfdena' input to selectively enable the phase/freq. detector

Lock output

- ☑ Create 'locked' output
- ☐ Enable self-reset on loss of lock
- ☐ Hold 'locked' output low for 1048575 cycles after the PLL initializes

Advanced PLL parameters

Using these parameters is recommended for advanced users only

- ☐ Create output file(s) using the 'Advanced' PLL parameters

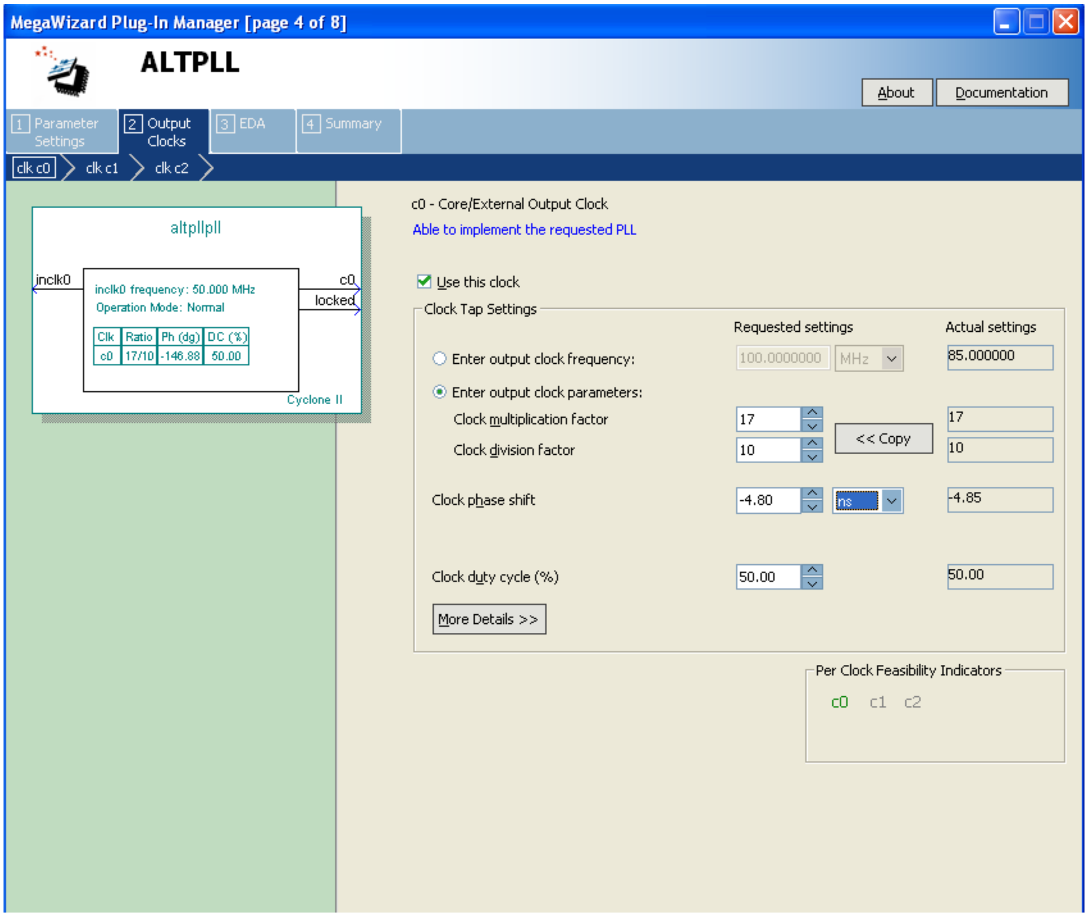
- Configurations with output clock(s) that use cascade counters are not supported

inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	1/1	0.00	50.00

5. Check **Create "locked" output** under **Lock output**.
6. Press **Enter.**
7. Press **Enter.** You will go on to the 4th page of the wizard Figure 2-17.

Figure 2-17.MegaWizard Plug-In Manager [Page 4 of 8]


MegaWizard Plug-In Manager [page 4 of 8]

ALTPLL

inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	17/10	-146.88	50.00

c0 - Core/External Output Clock
Able to implement the requested PLL

[✓] Use this clock

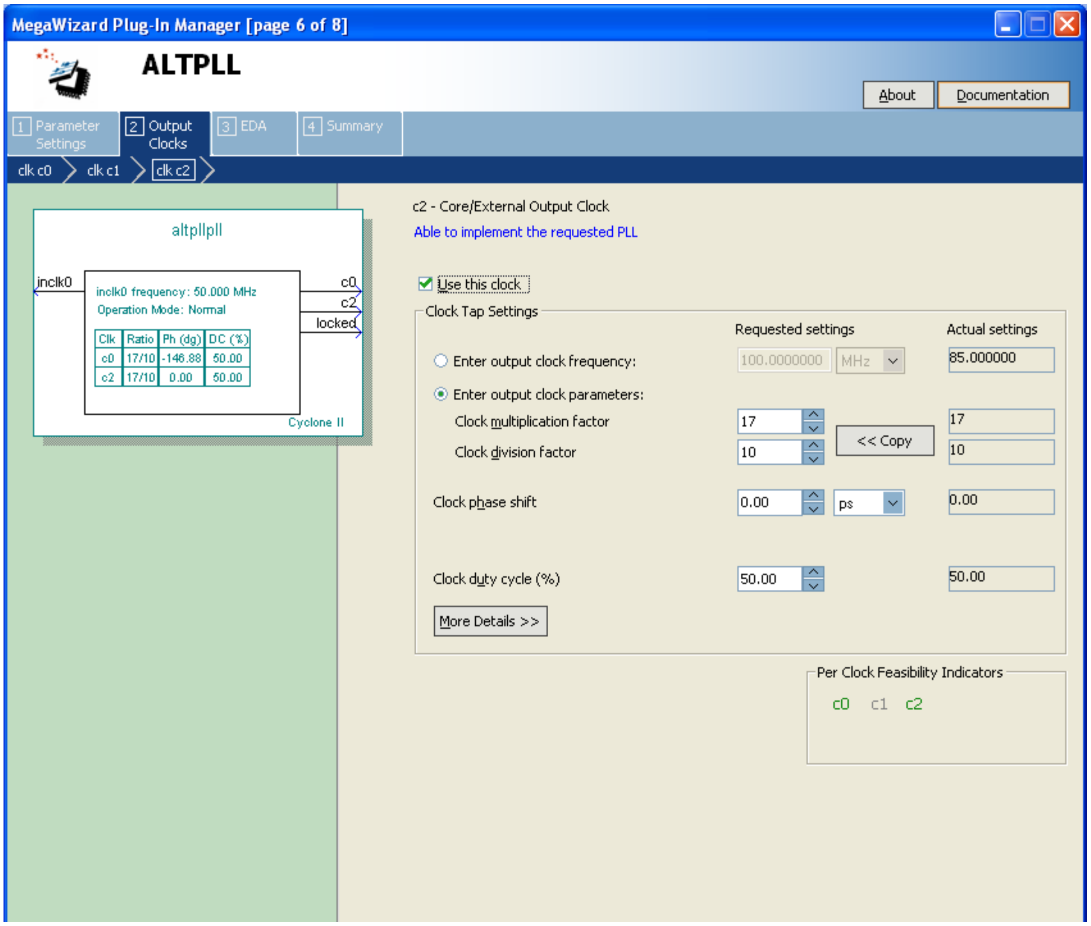
Clock Tap Settings

	Requested settings		Actual settings
○ Enter output clock frequency:	100.0000000	MHz	85.000000
● Enter output clock parameters:			
Clock multiplication factor	17	<< Copy	17
Clock division factor	10		10
Clock phase shift	-4.80	ns	-4.85
Clock duty cycle (%)	50.00		50.00

8. Check **Use this clock** checkbox.
9. Select **Enter output clock parameters:** under **Clock Tap Settings**. Set the following options:
   - Clock multiplication factor: **17**
   - Clock division factor: **10**
   - Clock phase shift: **-4.80** ns
   - Clock duty cycle: **50**

10. Press **Enter.** You will go on to the 6th page of the wizard. See Figure 2-18.

Figure 2-18.MegaWizard Plug-In Manager [Page 6 of 8]



inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	17/10	-146.88	50.00
c2	17/10	0.00	50.00

c2 - Core/External Output Clock
Able to implement the requested PLL

☑ Use this clock

Clock Tap Settings

	Requested settings		Actual settings
○ Enter output clock frequency:	100.0000000	MHz	85.000000
● Enter output clock parameters:			
Clock multiplication factor	17		17
Clock division factor	10		10
Clock phase shift	0.00	ps	0.00
Clock duty cycle (%)	50.00		50.00

Per Clock Feasibility Indicators
c0 c1 c2

11. Check **Use this clock** checkbox.

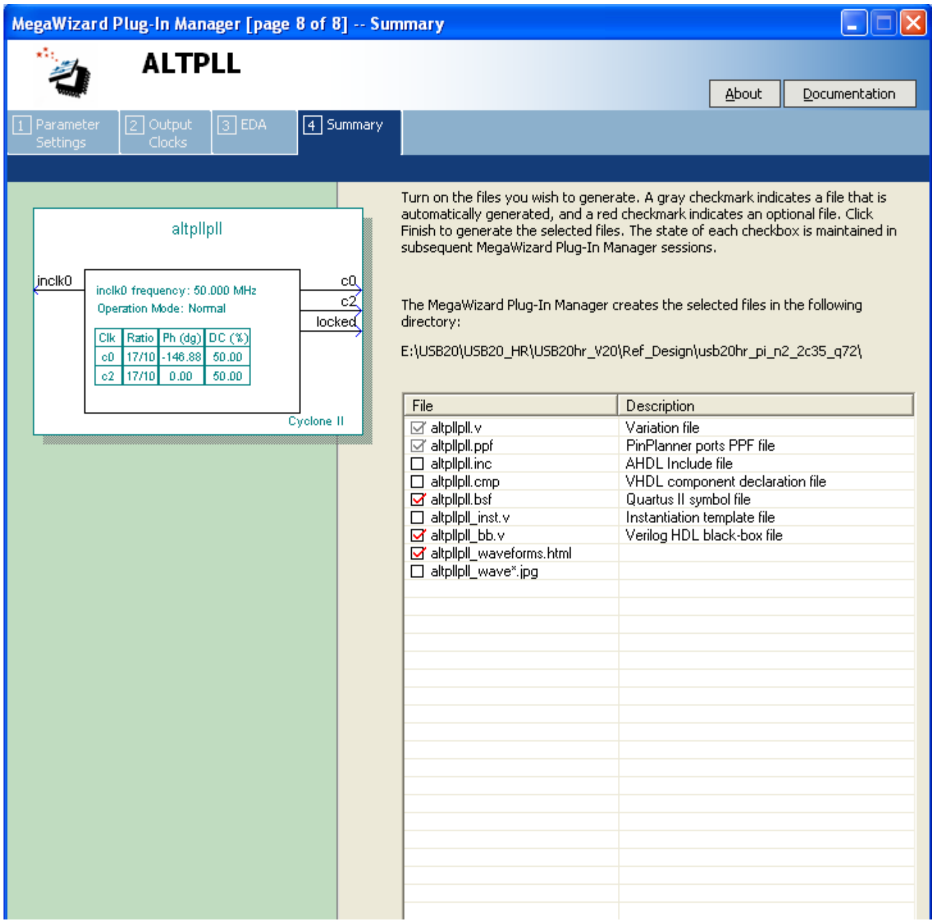
12. Select **Enter output clock parameters:** under **Clock Tap Settings**. Set the following options:

- Clock multiplication factor: **17**
- Clock division factor: **10**
- Clock phase shift: **0.00 ps**

- Clock duty cycle: **50**

13. Press **Enter.** You will go on to the 8th page of the wizard. See Figure 2-19.

Figure 2-19.MegaWizard Plug-In Manager [Page 8 of 8]



inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	17/10	-146.88	50.00
c2	17/10	0.00	50.00

Turn on the files you wish to generate. A gray checkmark indicates a file that is automatically generated, and a red checkmark indicates an optional file. Click Finish to generate the selected files. The state of each checkbox is maintained in subsequent MegaWizard Plug-In Manager sessions.

The MegaWizard Plug-In Manager creates the selected files in the following directory:

E:\USB20\USB20_HR\USB20hr_V20\Ref_Design\usb20hr_pi_n2_2c35_q72\

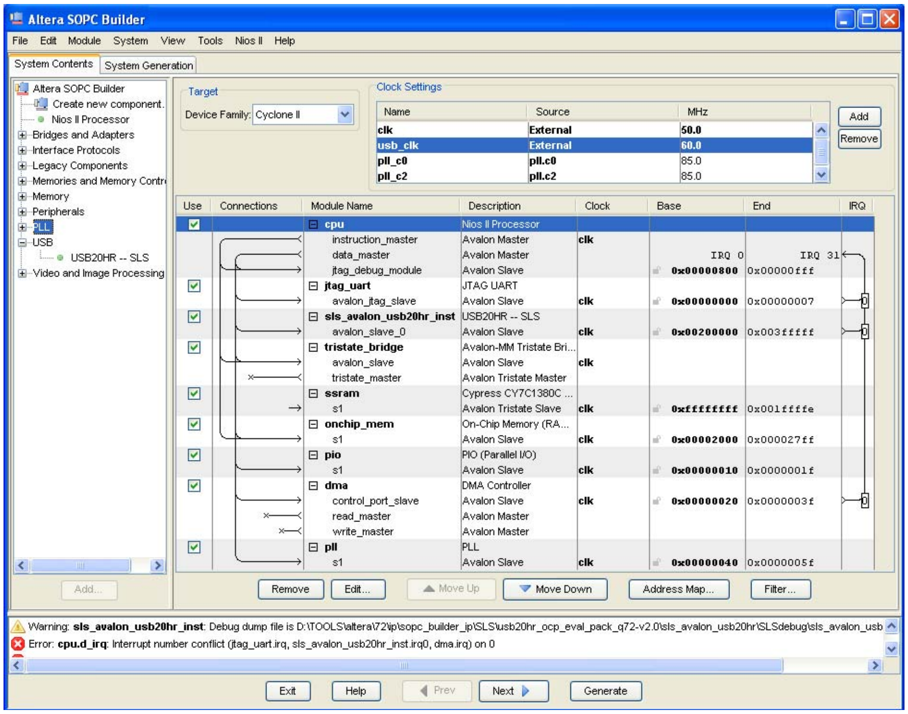
File	Description
✓ altpllpll.v	Variation file
✓ altpllpll.ppf	PinPlanner ports PPF file
☐ altpllpll.inc	AHDL Include file
☐ altpllpll.cmp	VHDL component declaration file
✓ altpllpll.bsf	Quartus II symbol file
☐ altpllpll_inst.v	Instantiation template file
✓ altpllpll_bb.v	Verilog HDL black-box file
✓ altpllpll_waveforms.html	
☐ altpllpll_wave*.jpg	

14. Press **Enter.** You will return to PLL Configuration Wizard. See Figure 2-14.

15. Click on **Finish.**

After adding all components, the SOPC builder system will look like as shown in Figure 2-20.

Figure 2-20.SOPC Builder System After Adding all Components



Device Family: Cyclone II

Clock Settings

Name	Source	MHz
clk	External	50.0
usb_clk	External	60.0
pll_c0	pll.c0	85.0
pll_c2	pll.c2	85.0

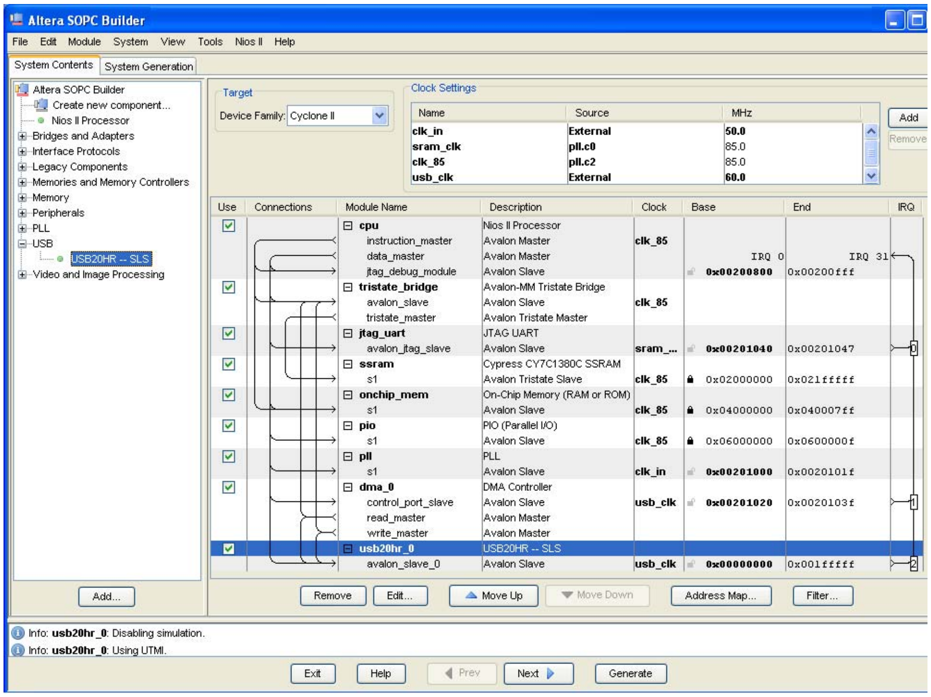
Use	Module Name	Description	Clock	Base	End	IRQ
✓	cpu	Nios II Processor				
	instruction_master	Avalon Master	clk			
	data_master	Avalon Master		IRQ 0	IRQ 31	
	jtag_debug_module	Avalon Slave		0x00000800	0x00000fff	
✓	jtag_uart	JTAG UART				
	avalon_jtag_slave	Avalon Slave	clk	0x00000000	0x00000007	0
✓	sls_avalon_usb20hr_inst	USB20HR -- SLS				
	avalon_slave_0	Avalon Slave	clk	0x00200000	0x003fffff	0
✓	tristate_bridge	Avalon-MM Tristate Bri...				
	avalon_slave	Avalon Slave	clk			
	tristate_master	Avalon Tristate Master				
✓	ssram	Cypress CY7C1380C ...				
	s1	Avalon Tristate Slave	clk	0xffffffff	0x001ffffe	
✓	onchip_mem	On-Chip Memory (RA...				
	s1	Avalon Slave	clk	0x00002000	0x000027ff	
✓	pio	PIO (Parallel I/O)				
	s1	Avalon Slave	clk	0x00000010	0x0000001f	
✓	dma	DMA Controller				
	control_port_slave	Avalon Slave	clk	0x00000020	0x0000003f	0
	read_master	Avalon Master				
	write_master	Avalon Master				
✓	pll	PLL				
	s1	Avalon Slave	clk	0x00000040	0x0000005f	

Warning: **sls_avalon_usb20hr_inst**: Debug dump file is D:\TOOLS\altera\72\ip\sopc_builder_ip\SLS\usb20hr_ocp_eval_pack_q72-v2.0\sls_avalon_usb20hr\SLSdebug\sls_avalon_usb

Error: **cpu.d_irq**: Interrupt number conflict (jtag_uart.irq, sls_avalon_usb20hr_inst.irq0, dma.irq) on 0

Now, arrange all components and provide the clock to each component as shown in Figure 2-21. Also rename the **clocks** under **clock settings** as shown in Figure 2-21.

Figure 2-21.Final View of the SOPC Builder System



Device Family: Cyclone II

Name	Source	MHz
clk_in	External	50.0
sram_clk	pll.c0	85.0
clk_85	pll.c2	85.0
usb_clk	External	60.0

Use	Connections	Module Name	Description	Clock	Base	End	IRQ
✓		cpu	Nios II Processor				
		instruction_master	Avalon Master	clk_85			
		data_master	Avalon Master		IRQ 0	IRQ 31	
		jtag_debug_module	Avalon Slave		0x00200800	0x00200fff	
✓		tristate_bridge	Avalon-MM Tristate Bridge				
		avalon_slave	Avalon Slave	clk_85			
		tristate_master	Avalon Tristate Master				
✓		jtag_uart	JTAG UART				
		avalon_jtag_slave	Avalon Slave	sram_...	0x00201040	0x00201047	0
✓		ssram	Cypress CY7C1380C SSRAM				
		s1	Avalon Tristate Slave	clk_85	0x02000000	0x021fffff	
✓		onchip_mem	On-Chip Memory (RAM or ROM)				
		s1	Avalon Slave	clk_85	0x04000000	0x040007ff	
✓		pio	PIO (Parallel I/O)				
		s1	Avalon Slave	clk_85	0x06000000	0x0600000f	
✓		pll	PLL				
		s1	Avalon Slave	clk_in	0x00201000	0x0020101f	
✓		dma_0	DMA Controller				
		control_port_slave	Avalon Slave	usb_clk	0x00201020	0x0020103f	1
		read_master	Avalon Master				
		write_master	Avalon Master				
✓		usb20hr_0	USB20HR -- SLS				
		avalon_slave_0	Avalon Slave	usb_clk	0x00000000	0x001fffff	2

Info: **usb20hr_0**: Disabling simulation.
Info: **usb20hr_0**: Using UTMI.

Click on the Module name **CPU**. You will return to the **Nios II Processor - CPU Configuration Wizard**. Specify following options:

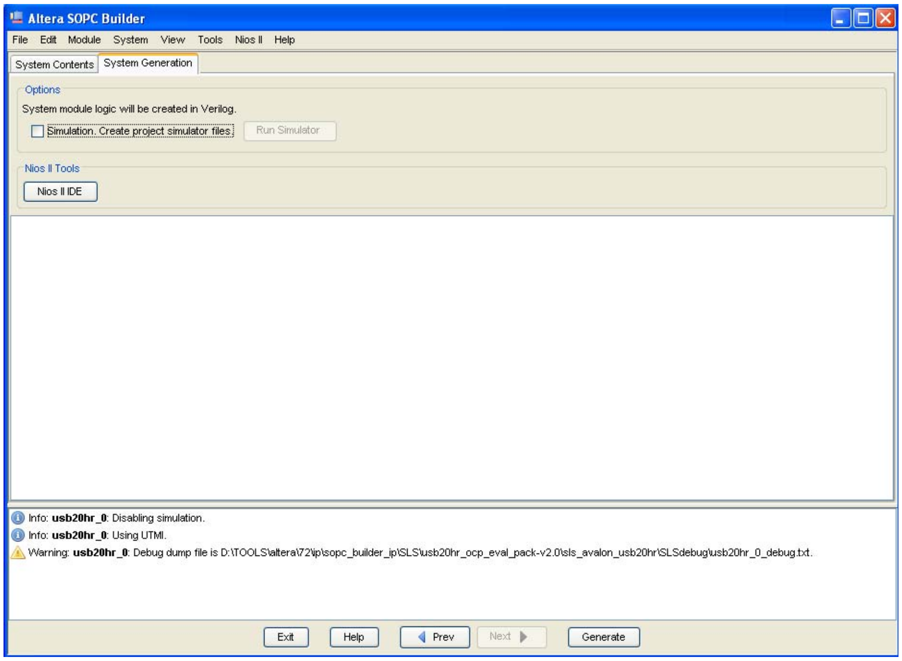
- Reset Vector Memory: **SSRAM**
- Exception Vector Memory: **SSRAM**

Generating the System

To generate the design logic, perform the following steps.

1. Click the **System Generation tab**.
2. Make the following settings under **Options** in **System Generation** tab. See Figure 2-22.
   - **Simulation:**
     – If you are designing the USB20HR system from scratch, uncheck this box.
     – If you are simulating the USB20HR system, check this box.

Figure 2-22.Generating the System



System module logic will be created in Verilog.

```
Info: usb20hr_0: Disabling simulation.
Info: usb20hr_0: Using UTMI.
Warning: usb20hr_0: Debug dump file is D:\TOOLS\altera\72\ip\sopc_builder_ip\SLS\usb20hr_ocp_eval_pack-v2.0\sls_avalon_usb20hr\SLSdebug\usb20hr_0_debug.txt.
```

3. Click **Generate**.
4. When generation is complete, the **SYSTEM GENERATION COMPLETED** message displays.

   ☞ Do NOT exit SOPC Builder at this point. We will return to this window prior to testing the system with software.

Integrate the SOPC Builder System into Quartus II

You have now created your SOPC builder system to evaluate USB20HR IP. Now compile this system with other necessary logic in quartus II software for generating programming file.

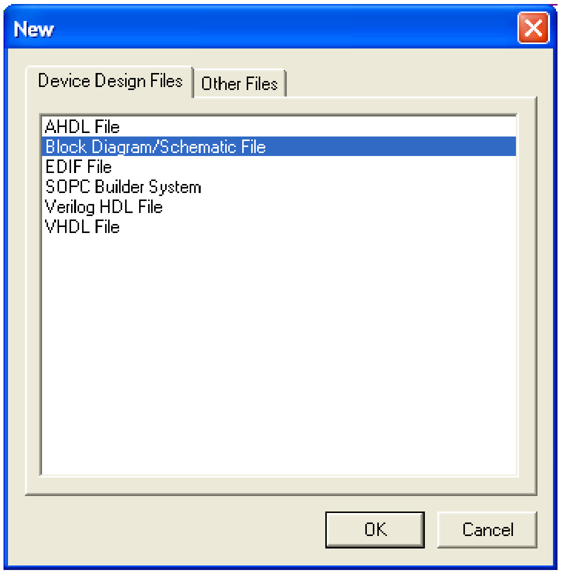
Adding the Quartus II Symbol to the BDF

During generation, SOPC Builder creates a symbol of the System Top, for use in Quartus II. You can add this symbol to your BDF. To add the symbol perform the following steps:

1. Select **File (menu)** > **New**.

2. Under Device Design Files, highlight **Block Diagram/Schematic File**. See Figure 2-23.

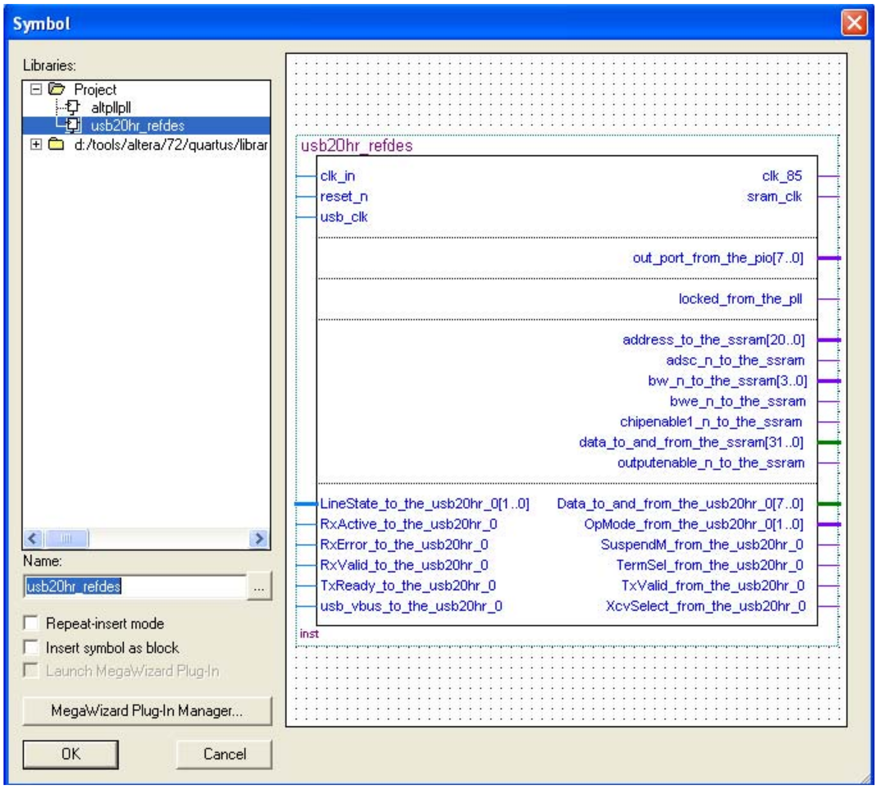
Figure 2-23. New Block Design File



AHDL File
Block Diagram/Schematic File
EDIF File
SOPC Builder System
Verilog HDL File
VHDL File

3. Click **OK**
4. Return to the Quartus II software and double click anywhere inside the **BDF window**. The Symbol dialog box appears.
5. In the Symbol dialog box, Click on **Project** to expand Project directory.
6. Click on the project **usb20hr_refdes.** A large symbol will appear representing the Nios II system module you just created. See Figure 2-24.

Figure 2-24. BDF Dialogbox


Symbol
Libraries:
Project
altpllpll
usb20hr_refdes
d:/tools/altera/72/quartus/librar
Name:
usb20hr_refdes
Repeat-insert mode
Insert symbol as block
Launch MegaWizard Plug-In
MegaWizard Plug-In Manager...
OK
Cancel
usb20hr_refdes
clk_in
reset_n
usb_clk
clk_85
sram_clk
out_port_from_the_pio[7..0]
locked_from_the_pll
address_to_the_ssram[20..0]
adsc_n_to_the_ssram
bw_n_to_the_ssram[3..0]
bwe_n_to_the_ssram
chipenable1_n_to_the_ssram
data_to_and_from_the_ssram[31..0]
outputenable_n_to_the_ssram
LineState_to_the_usb20hr_0[1..0]
RxActive_to_the_usb20hr_0
RxError_to_the_usb20hr_0
RxValid_to_the_usb20hr_0
TxReady_to_the_usb20hr_0
usb_vbus_to_the_usb20hr_0
Data_to_and_from_the_usb20hr_0[7..0]
OpMode_from_the_usb20hr_0[1..0]
SuspendM_from_the_usb20hr_0
TermSel_from_the_usb20hr_0
TxValid_from_the_usb20hr_0
XcvSelect_from_the_usb20hr_0
inst

7. Click **OK**. The Symbol dialog box closes and an outline of the System Top symbol is attached to the pointer.
8. Place the symbol in the block design schematic files.
9. Choose **Save.**
10. Similarly add input, output pins into design schematic file and make a connection of those pins as shown in Figure 2-25. and Figure 2-26.
11. Now assign input output pins for UTMI as shown in Figure 2-25. and for ULPI as shown in Figure 2-26.

☞ If you are designing a system for another board then you will have to do pin assignment according to FPGA IO pins assignment of that board. For pin assignments refer Appendix A. The pin assignments are given for connection of USB Snap On Board on Santa Cruz headers J11,J12,J13.

Figure 2-25.Final BDF for UTMI Interface

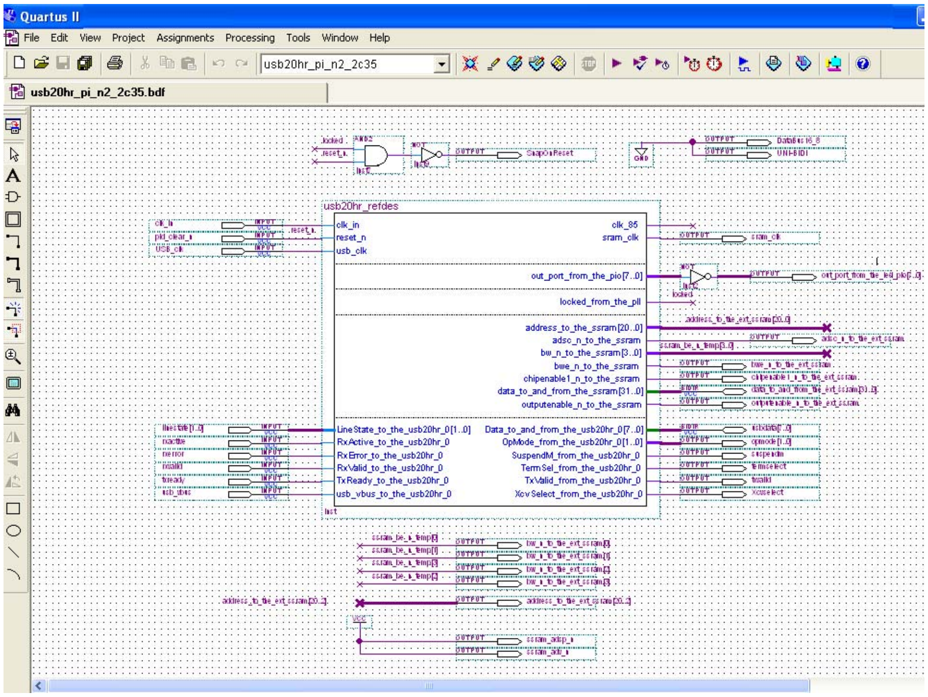
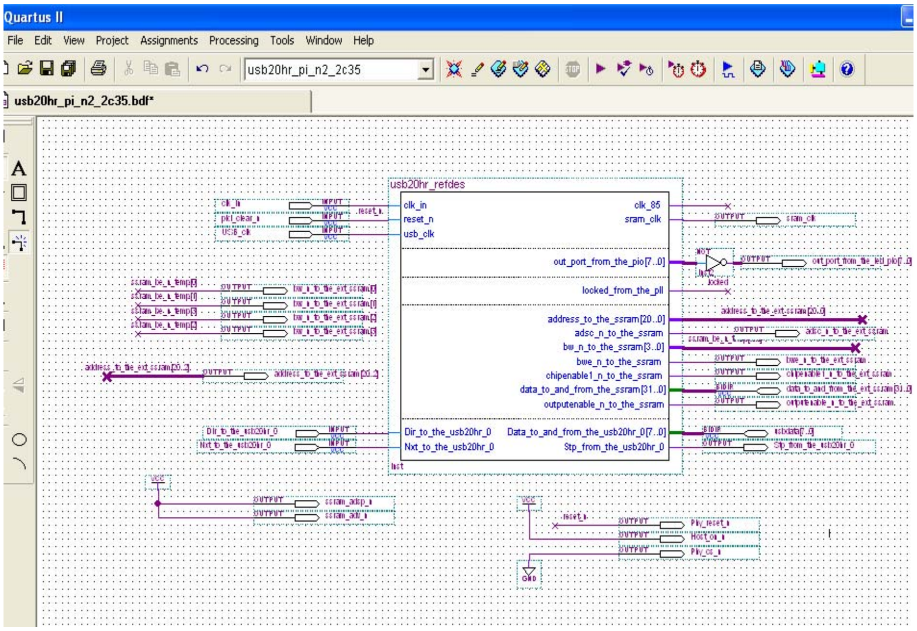

Quartus II
File Edit View Project Assignments Processing Tools Window Help
usb20hr_pi_n2_2c35
usb20hr_pi_n2_2c35.bdf
usb20hr_refdes
clk_in
reset_n
usb_clk
clk_85
sram_clk
out_port_from_the_pio[7..0]
locked_from_the_pll
address_to_the_ssram[20..0]
adsc_n_to_the_ssram
bw_n_to_the_ssram[3..0]
bwe_n_to_the_ssram
chipenable1_n_to_the_ssram
data_to_and_from_the_ssram[31..0]
outputenable_n_to_the_ssram
LineState_to_the_usb20hr_0[1..0]
RxActive_to_the_usb20hr_0
RxError_to_the_usb20hr_0
RxValid_to_the_usb20hr_0
TxReady_to_the_usb20hr_0
usb_vbus_to_the_usb20hr_0
Data_to_and_from_the_usb20hr_0[7..0]
OpMode_from_the_usb20hr_0[1..0]
SuspendM_from_the_usb20hr_0
TermSel_from_the_usb20hr_0
TxValid_from_the_usb20hr_0
XcvSelect_from_the_usb20hr_0
inst

Figure 2-26.Final BDF for ULPI Interface



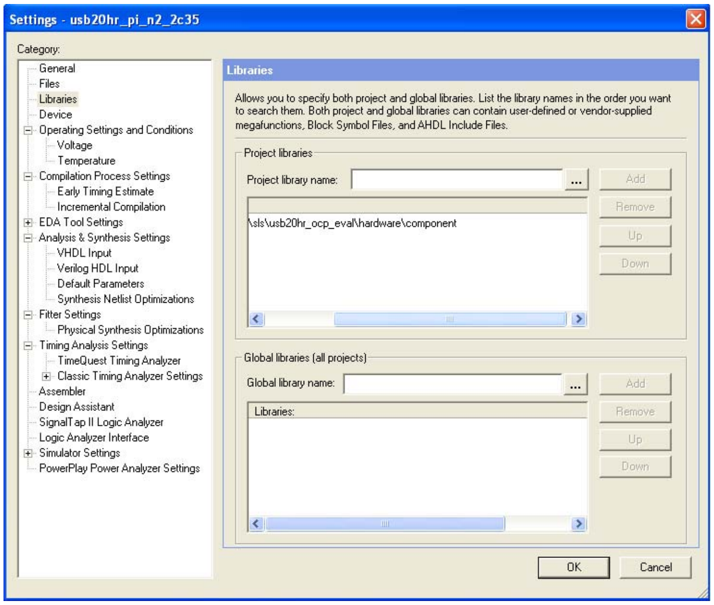
Device Settings

Device settings will depend on board used for reference design. Device setting for this design for Nios II Development Kit (2C35) is given below.

To do device settings follow the steps below:

1. Choose **Assignment>Settings**.
2. Select **Library** under **Category** and add the path of the USB20HR component as shown in Figure 2-27.

Figure 2-27. Library Settings



Allows you to specify both project and global libraries. List the library names in the order you want to search them. Both project and global libraries can contain user-defined or vendor-supplied megafunctions, Block Symbol Files, and AHDL Include Files.

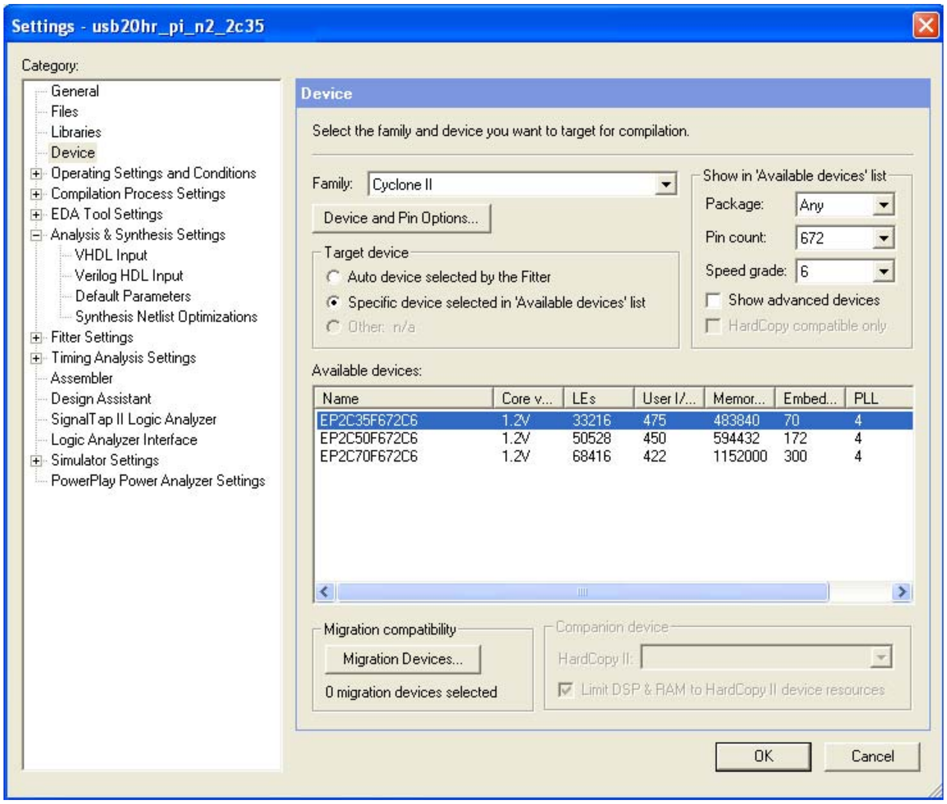
\sls\usb20hr_ocp_eval\hardware\component

3. Under **Category** select **Device**.

4. Under **Device tab** select **Cyclone II** family. See Figure 2-28.

5. Specify following options in the Available Devices list see Figure 2-28.
   - Package: **Any**
   - Pin Count**: 672**
   - Speed grade: **6**
   - Available Devices: **EP2C35F672C6**

Figure 2-28.Device Settings



Settings - usb20hr_pi_n2_2c35

Category:
- General
- Files
- Libraries
- Device
- Operating Settings and Conditions
- Compilation Process Settings
- EDA Tool Settings
- Analysis & Synthesis Settings
  - VHDL Input
  - Verilog HDL Input
  - Default Parameters
  - Synthesis Netlist Optimizations
- Fitter Settings
- Timing Analysis Settings
- Assembler
- Design Assistant
- SignalTap II Logic Analyzer
- Logic Analyzer Interface
- Simulator Settings
- PowerPlay Power Analyzer Settings

Device

Select the family and device you want to target for compilation.

Family: Cyclone II

Device and Pin Options...

Target device
- Auto device selected by the Fitter
- [✓ Specific device selected in 'Available devices' list]
- Other: n/a

Show in 'Available devices' list
- Package: Any
- Pin count: 672
- Speed grade: 6
- Show advanced devices
- HardCopy compatible only

Available devices:

Name	Core v...	LEs	User I/...	Memor...	Embed...	PLL
EP2C35F672C6	1.2V	33216	475	483840	70	4
EP2C50F672C6	1.2V	50528	450	594432	172	4
EP2C70F672C6	1.2V	68416	422	1152000	300	4

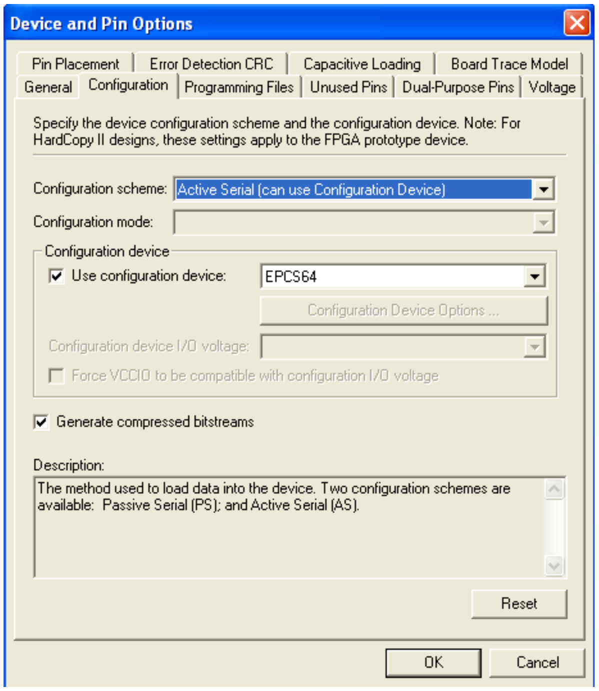
Migration compatibility
Migration Devices...
0 migration devices selected

Companion device
HardCopy II:
Limit DSP & RAM to HardCopy II device resources

OK Cancel

6. Click on **Device & Pin Options.**
7. Specify following options under **Configuration** tab see Figure 2-29.
   - Configuration scheme: **Active Serial (can use Configuration Device)**
   - Configuration device: **EPCS64**
   - Check on **Generate Compressed Bitstream**.

Figure 2-29. Configuration Tab Settings



Specify the device configuration scheme and the configuration device. Note: For HardCopy II designs, these settings apply to the FPGA prototype device.

Configuration scheme: Active Serial (can use Configuration Device)

Configuration mode:

Configuration device

[✓] Use configuration device: EPCS64

Configuration Device Options ...

Configuration device I/O voltage:

[ ] Force VCCIO to be compatible with configuration I/O voltage

[✓] Generate compressed bitstreams

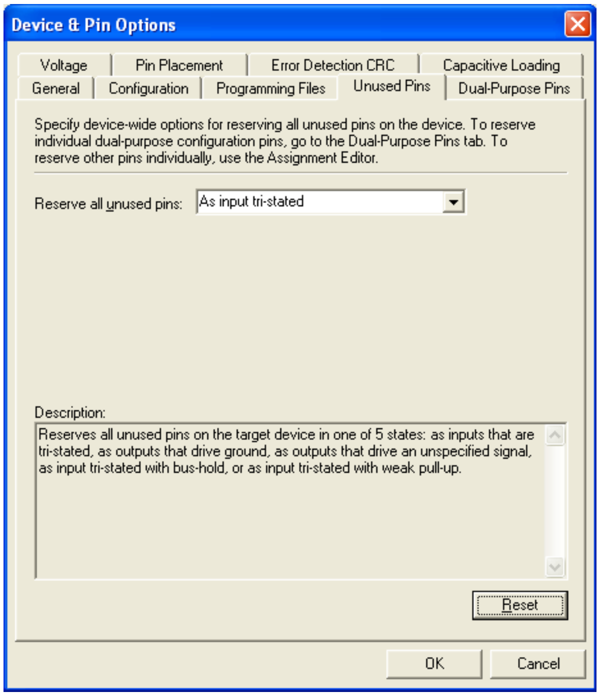
Description:

The method used to load data into the device. Two configuration schemes are available: Passive Serial (PS); and Active Serial (AS).

8. Click on **Unused Pins** Tab.

9. Click on **As inputs, tristated** under **Unused pins** tab**.** See Figure 2-30.

Figure 2-30. Unused Pins Tab Settings



Specify device-wide options for reserving all unused pins on the device. To reserve individual dual-purpose configuration pins, go to the Dual-Purpose Pins tab. To reserve other pins individually, use the Assignment Editor.

Reserve all unused pins: As input tri-stated

Description:

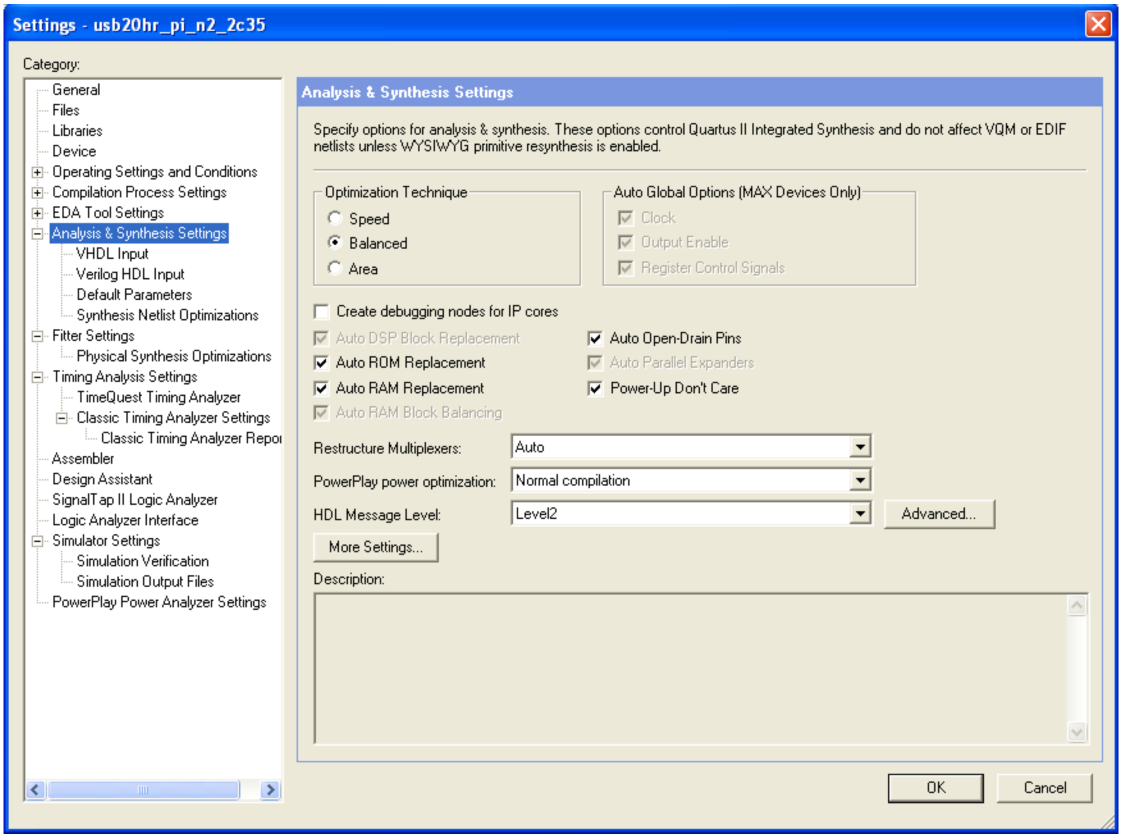
Reserves all unused pins on the target device in one of 5 states: as inputs that are tri-stated, as outputs that drive ground, as outputs that drive an unspecified signal, as input tri-stated with bus-hold, or as input tri-stated with weak pull-up.

10. Remain **Default settings** for all other tabs.

11. Click on **OK**.

12. Click on **Analysis and Synthesis** settings and do settings according to Figure 2-31.

Figure 2-31.Analysis and Synthesis Settings



Settings - usb20hr_pi_n2_2c35

Category:

- General
- Files
- Libraries
- Device
- Operating Settings and Conditions
- Compilation Process Settings
- EDA Tool Settings
- Analysis & Synthesis Settings
  - VHDL Input
  - Verilog HDL Input
  - Default Parameters
  - Synthesis Netlist Optimizations
- Fitter Settings
  - Physical Synthesis Optimizations
- Timing Analysis Settings
  - TimeQuest Timing Analyzer
  - Classic Timing Analyzer Settings
    - Classic Timing Analyzer Repor
- Assembler
- Design Assistant
- SignalTap II Logic Analyzer
- Logic Analyzer Interface
- Simulator Settings
  - Simulation Verification
  - Simulation Output Files
- PowerPlay Power Analyzer Settings

Analysis & Synthesis Settings

Specify options for analysis & synthesis. These options control Quartus II Integrated Synthesis and do not affect VQM or EDIF netlists unless WYSIWYG primitive resynthesis is enabled.

Optimization Technique: Speed [✓ Balanced] Area

Auto Global Options (MAX Devices Only): ✓ Clock, ✓ Output Enable, ✓ Register Control Signals

- ☐ Create debugging nodes for IP cores
- ✓ Auto DSP Block Replacement
- ✓ Auto ROM Replacement
- ✓ Auto RAM Replacement
- ✓ Auto RAM Block Balancing
- ✓ Auto Open-Drain Pins
- ✓ Auto Parallel Expanders
- ✓ Power-Up Don't Care

Restructure Multiplexers: Auto

PowerPlay power optimization: Normal compilation

HDL Message Level: Level2

Compiling the Design

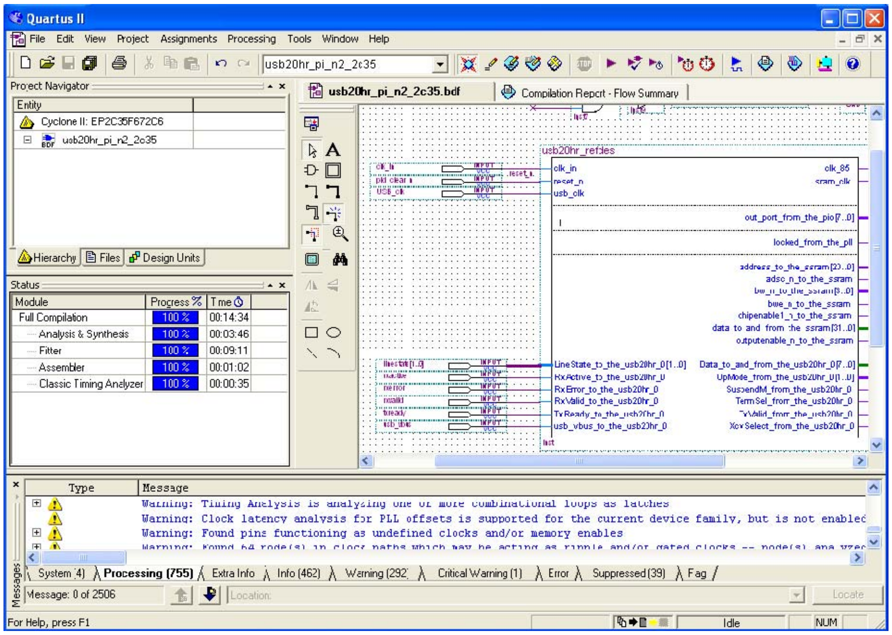
During compilation, the Compiler locates & processes all design and project files, generates messages & reports related to the current compilation, creates the **SRAM object file** (.sof) as well as any optional programming files.

To compile the **usb20hr_pi_n2_2C35** design, follow these steps:

1. Choose **Start Compilation** (Processing menu), or click the **Start Compilation** toolbar button. If you get a message asking if you want to save the changes you made the BDF file, choose **Yes**.

2. When compilation completes, you can view the results in the Status window. See Figure 2-32.

Figure 2-32.Compilation Report Window



Status

Module	Progress %	Time
Full Compilation	100 %	00:14:34
Analysis & Synthesis	100 %	00:03:46
Fitter	100 %	00:09:11
Assembler	100 %	00:01:02
Classic Timing Analyzer	100 %	00:00:35

```
Type    Message
        Warning: Timing Analysis is analyzing one or more combinational loops as latches
        Warning: Clock latency analysis for PLL offsets is supported for the current device family, but is not enabled
        Warning: Found pins functioning as undefined clocks and/or memory enables
        Warning: Found 64 node(s) in clock paths which may be acting as ripple and/or gated clocks -- node(s) analyzed
```

Message: 0 of 2506

3. Programming

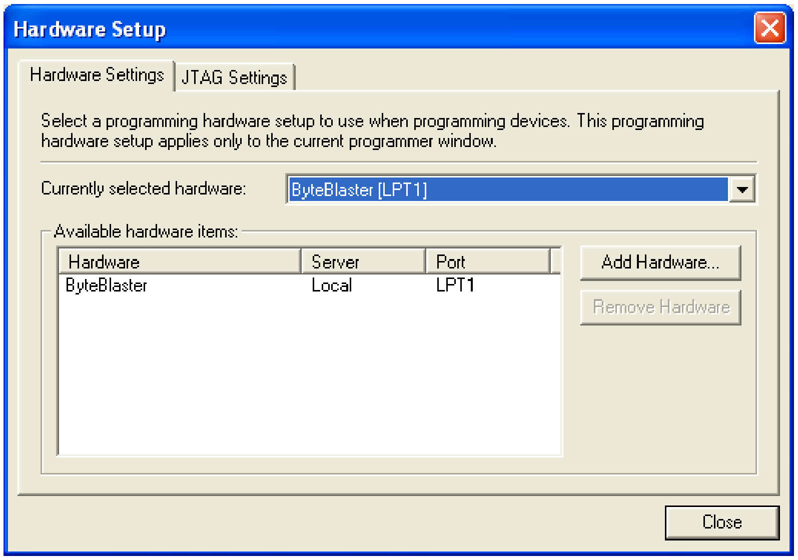
After successful compilation, the Quartus II Compiler generates one or more programming files that the Programmer uses to program or configure a device.

Configure an FPGA

To program your design, follow these steps:

1. Choose **Programmer (Tool menu)**. The Programmer window opens.
2. In the Mode list select **Active Serial** or **JTAG**. If user selects active serial mode, **.pof** appears else if jtag is selected **.sof** appears.
3. Click **Hardware Setup** to configure the programming hardware, see Figure 3-1.

Figure 3-1. Hardware Setup



Hardware Setup

Hardware Settings | JTAG Settings

Select a programming hardware setup to use when programming devices. This programming hardware setup applies only to the current programmer window.

Currently selected hardware: ByteBlaster [LPT1]

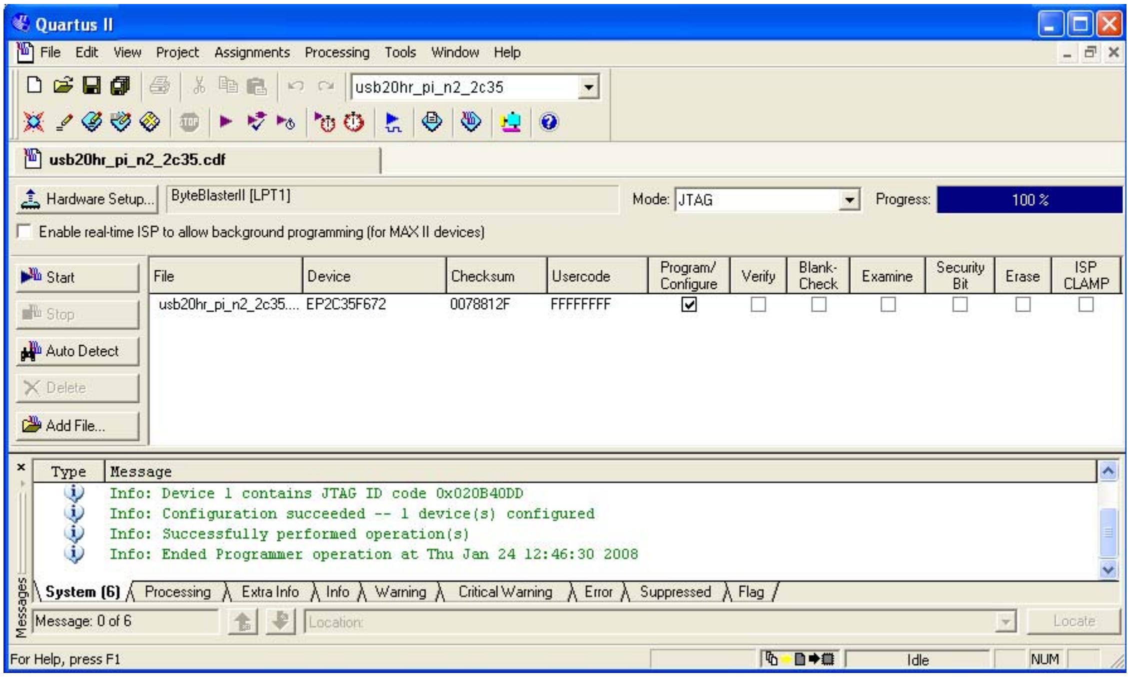
Available hardware items:

Hardware	Server	Port
ByteBlaster	Local	LPT1

4. Under **Hardware Settings** tab select **USB Blaster** or **Byte BlasterII (LPT1)** in the **Currently selected hardware**

5. Click **Close** to exit the Hardware Setup window.
6. In the Programmer window, turn on **Program/Configure**. See Figure 3-2.

Figure 3-2. Programmer Window



ByteBlasterII [LPT1]   Mode: JTAG   Progress: 100 %

☐ Enable real-time ISP to allow background programming (for MAX II devices)

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase	ISP CLAMP
usb20hr_pi_n2_2c35....	EP2C35F672	0078812F	FFFFFFFF	☑	☐	☐	☐	☐	☐	☐

```
Type  Message
      Info: Device 1 contains JTAG ID code 0x020B40DD
      Info: Configuration succeeded -- 1 device(s) configured
      Info: Successfully performed operation(s)
      Info: Ended Programmer operation at Thu Jan 24 12:46:30 2008
```

System (6) | Processing | Extra Info | Info | Warning | Critical Warning | Error | Suppressed | Flag

Message: 0 of 6   Location:

7. Click **Start**.
8. User can see the progress as shown in Figure 3-2.

4. Running the Project in Nios II System

The design that we have created is for the Cyclone II board. We will run a simple test application of port interface on the Cyclone II board using USB Interface. We will be using Nios II Integrated Development Environment (IDE) to run our software on top of Nios II System.

Start **Nios II IDE** by following one of the two options

Start>programs>altera>Nios II Development Kit 7.2>Nios II IDE

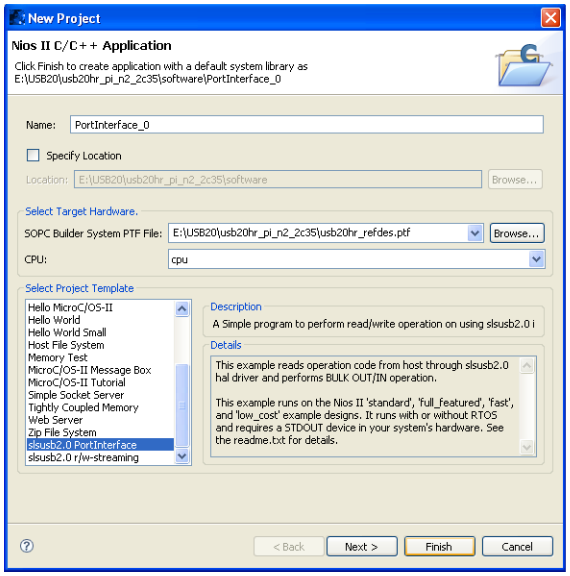
or

Click **System Generation tab** of the SOPC Builder window then click the **Run Nios II IDE button.**

Creating the Project

1. Select **File** > **New C/C++ Application**.
2. Select Project Template **“slsusb2.0 Portinterface”.**
3. In SOPC Builder System option, under **Select Target Hardware** browse your system.ptf **(usb20hr_refdes.ptf)** file. See Figure 4-1.

Figure 4-1. Creating USB Project Sample Application


New Project

Nios II C/C++ Application

Click Finish to create application with a default system library as E:\USB20\usb20hr_pi_n2_2c35\software\PortInterface_0

Name: PortInterface_0

☐ Specify Location

Location: E:\USB20\usb20hr_pi_n2_2c35\software

Select Target Hardware.

SOPC Builder System PTF File: E:\USB20\usb20hr_pi_n2_2c35\usb20hr_refdes.ptf

CPU: cpu

Select Project Template

- Hello MicroC/OS-II
- Hello World
- Hello World Small
- Host File System
- Memory Test
- MicroC/OS-II Message Box
- MicroC/OS-II Tutorial
- Simple Socket Server
- Tightly Coupled Memory
- Web Server
- Zip File System
- slsusb2.0 PortInterface
- slsusb2.0 r/w-streaming

Description

A Simple program to perform read/write operation on using slsusb2.0 i

Details

This example reads operation code from host through slsusb2.0 hal driver and performs BULK OUT/IN operation.

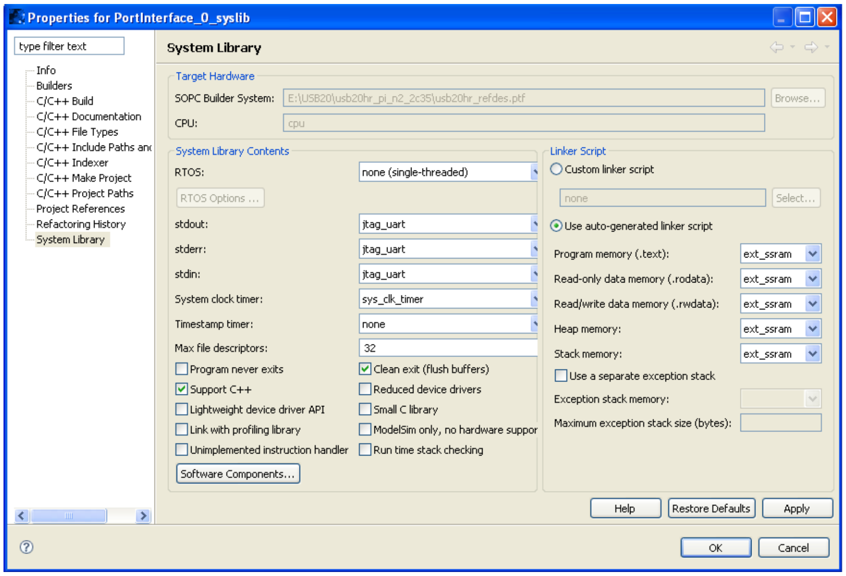
This example runs on the Nios II 'standard', 'full_featured', 'fast', and 'low_cost' example designs. It runs with or without RTOS and requires a STDOUT device in your system's hardware. See the readme.txt for details.

4. This template loads the required **source (.c)** file into the project.
5. After making above settings click **Next** and then **Finish**. It will create the project.

System Library Settings

1. Right click on the created project, under system library Property. Select **ssram** as Program memory, Read-only data memory, Read/write memory, Heap memory, Stack memory. see Figure 4-2.
2. Select **jtag_uart** in stdin, stderr, stdin toolbar. This is a cable through which program will run in Nios II IDE.
3. Select the checkbox option **modelsim only, no hardware support** under system library contents, if you want to perform simulation.

Figure 4-2. System Library Settings


Properties for PortInterface_0_syslib

System Library

Target Hardware

SOPC Builder System: E:\USB20\usb20hr_pi_n2_2c35\usb20hr_refdes.ptf

CPU: cpu

System Library Contents

RTOS: none (single-threaded)

stdout: jtag_uart

stderr: jtag_uart

stdin: jtag_uart

System clock timer: sys_clk_timer

Timestamp timer: none

Max file descriptors: 32

☐ Program never exits ☑ Clean exit (flush buffers)

☑ Support C++ ☐ Reduced device drivers

☐ Lightweight device driver API ☐ Small C library

☐ Link with profiling library ☐ ModelSim only, no hardware suppor

☐ Unimplemented instruction handler ☐ Run time stack checking

Linker Script

○ Custom linker script: none

● Use auto-generated linker script

Program memory (.text): ext_ssram

Read-only data memory (.rodata): ext_ssram

Read/write data memory (.rwdata): ext_ssram

Heap memory: ext_ssram

Stack memory: ext_ssram

☐ Use a separate exception stack

Exception stack memory:

Maximum exception stack size (bytes):

4. For getting more speed user has to do following settings which optimizes the c application core.
   - Click on **C/C++ Build -->Select Tool Settings Tab --> Select General** in Nios II compiler option and select optimization level **-o3.**
   - Right Click on **project** and select **properties** option. Click on **C/C++ Build --> Select Tool Settings Tab --> Select General** in Nios II compiler option and select optimization level **-o3.**

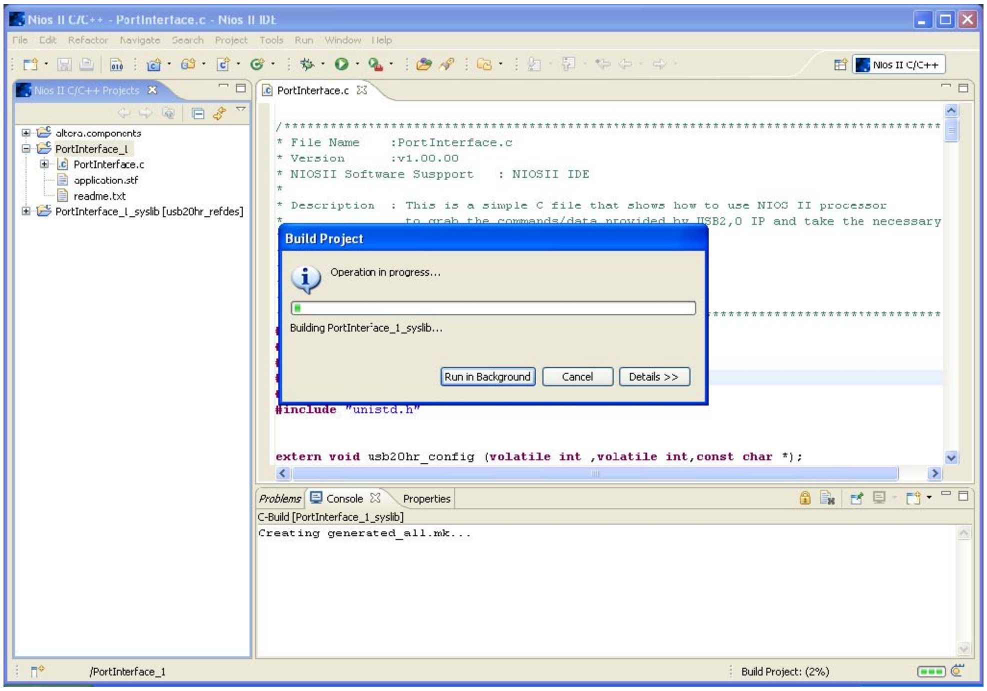
Building and Managing the Project

Compile C Application

To compile C Application, perform the following steps:

1. To build your project, right click on your project in the navigator pane and select **Build Project**. See Figure 4-3..

Figure 4-3. Building the Project


Nios II C/C++ - PortInterface.c - Nios II IDE
File Edit Refactor Navigate Search Project Tools Run Window Help
Nios II C/C++
Nios II C/C++ Projects
altera.components
PortInterface_1
PortInterface.c
application.stf
readme.txt
PortInterface_1_syslib [usb20hr_refdes]
PortInterface.c
Problems Console Properties
/PortInterface_1
Build Project: (2%)

```
/*************************************************************************************
* File Name    :PortInterface.c
* Version      :v1.00.00
* NIOSII Software Susspport   : NIOSII IDE
*
* Description  : This is a simple C file that shows how to use NIOS II processor
*                to grab the commands/data provided by USB2,0 IP and take the necessary
```

```
#include "unistd.h"

extern void usb20hr_config (volatile int ,volatile int,const char *);
```

Build Project

Operation in progress...

Building PortInterface_1_syslib...

Run in Background | Cancel | Details >>

C-Build [PortInterface_1_syslib]

```
Creating generated_all.mk...
```

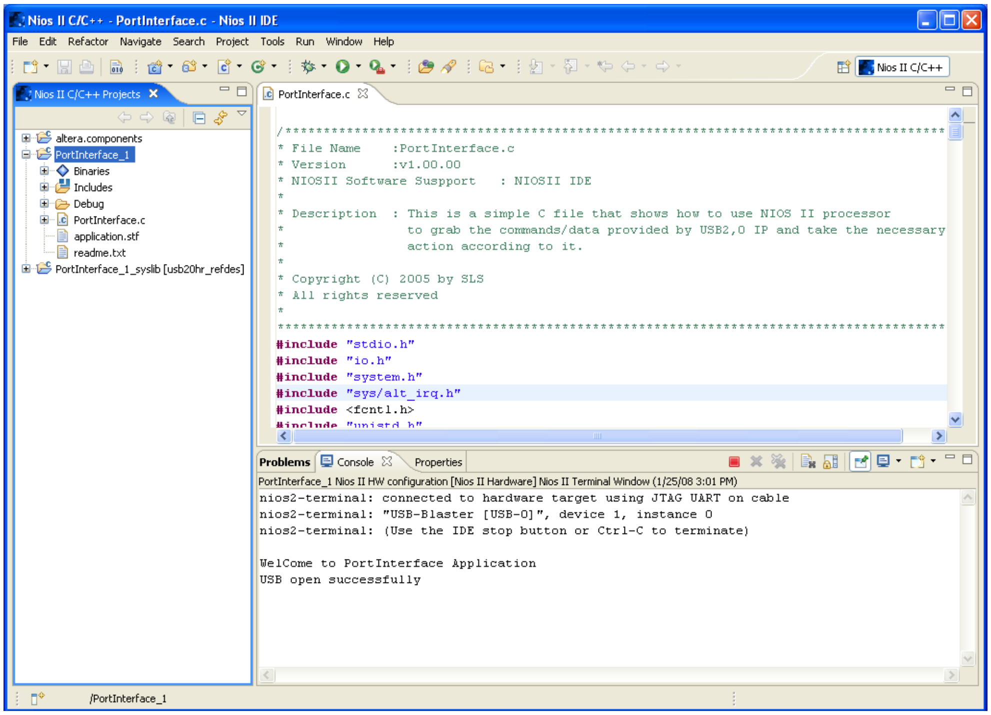
2. When the build process successfully completes, you can run the project. After Successful running the project on hardware you can see window as shown in Figure 4-4.

Running the Program on Nios II System

To run your program on the development board, perform the following steps:

1. Choose **Run > Run in the main Nios II IDE window.**
2. Double Click on **Nios II Hardware (RAM) in the Configurations browser** on the left-hand side. The Run window displays.
3. If you have more than one JTAG cable connection, then you will need to select the **Target Connection tab** and choose the **cable** that is connected to your board from the **JTAG cable pull down menu**.
4. Accept the **Default** and click **Run.**
5. On successful download, you should see as shown in Figure 4-4.below.

Figure 4-4. Program Downloaded Successfully



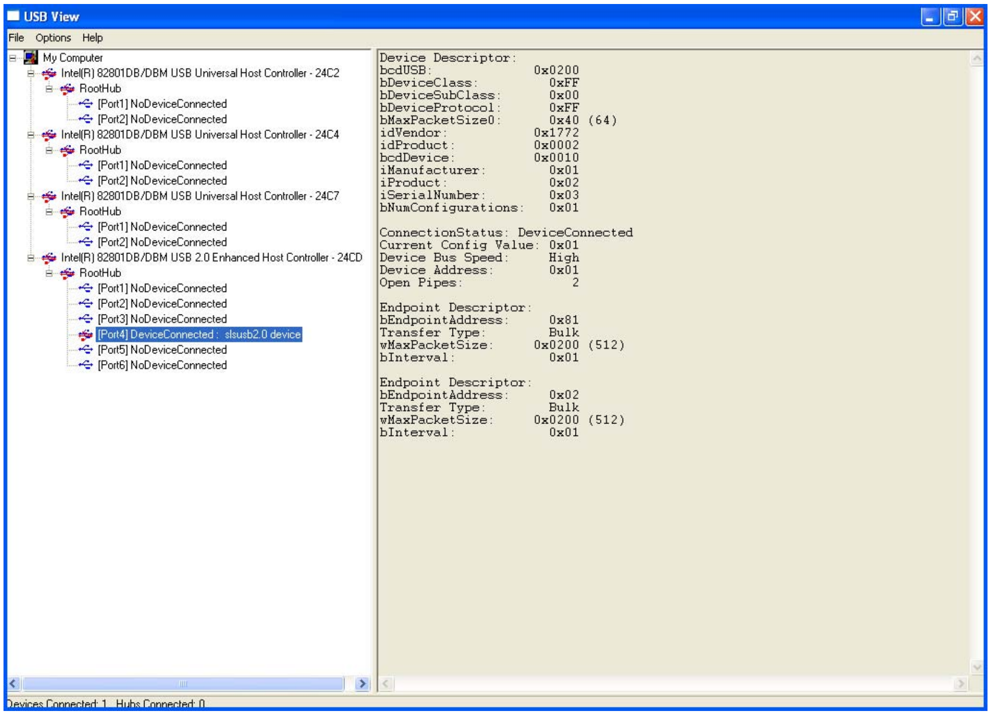
```
/*****************************************************************************
* File Name    :PortInterface.c
* Version      :v1.00.00
* NIOSII Software Susspport    : NIOSII IDE
*
* Description  : This is a simple C file that shows how to use NIOS II processor
*                to grab the commands/data provided by USB2,0 IP and take the necessary
*                action according to it.
*
* Copyright (C) 2005 by SLS
* All rights reserved
*
******************************************************************************
#include "stdio.h"
#include "io.h"
#include "system.h"
#include "sys/alt_irq.h"
#include <fcntl.h>
```

```
PortInterface_1 Nios II HW configuration [Nios II Hardware] Nios II Terminal Window (1/25/08 3:01 PM)
nios2-terminal: connected to hardware target using JTAG UART on cable
nios2-terminal: "USB-Blaster [USB-0]", device 1, instance 0
nios2-terminal: (Use the IDE stop button or Ctrl-C to terminate)

WelCome to PortInterface Application
USB open successfully
```

Device Detection

1. Now connect the **USB cable**, one end at the back of the PC and other to USB B-type connector.
2. If you are using this IP for the first time then you will see **“New Hardware found”** Wizard. Browse for the driver files located at <*USB20HR Installation Path*>**\usb20hr_**<*licensetype*>**\ software\driver\win2k_xp_x32.**
3. Click on SLSUSBView from <*USB20HR Installation Path*>\ **usb20hr_**<*licensetype*>**\software\utilities\usbview.**You will see **the** device connected on port where you connect your USB cable. For Bulk transfer you will see **“Device Connected: slsusb2.0 device”.** See Figure 4-5.

Figure 4-5. SLSUSBView



```
My Computer
  Intel(R) 82801DB/DBM USB Universal Host Controller - 24C2
    RootHub
      [Port1] NoDeviceConnected
      [Port2] NoDeviceConnected
  Intel(R) 82801DB/DBM USB Universal Host Controller - 24C4
    RootHub
      [Port1] NoDeviceConnected
      [Port2] NoDeviceConnected
  Intel(R) 82801DB/DBM USB Universal Host Controller - 24C7
    RootHub
      [Port1] NoDeviceConnected
      [Port2] NoDeviceConnected
  Intel(R) 82801DB/DBM USB 2.0 Enhanced Host Controller - 24CD
    RootHub
      [Port1] NoDeviceConnected
      [Port2] NoDeviceConnected
      [Port3] NoDeviceConnected
      [Port4] DeviceConnected : slsusb2.0 device
      [Port5] NoDeviceConnected
      [Port6] NoDeviceConnected
```

```
Device Descriptor:
bcdUSB:               0x0200
bDeviceClass:           0xFF
bDeviceSubClass:        0x00
bDeviceProtocol:        0xFF
bMaxPacketSize0:        0x40 (64)
idVendor:             0x1772
idProduct:            0x0002
bcdDevice:            0x0010
iManufacturer:          0x01
iProduct:               0x02
iSerialNumber:          0x03
bNumConfigurations:     0x01

ConnectionStatus: DeviceConnected
Current Config Value:  0x01
Device Bus Speed:       High
Device Address:         0x01
Open Pipes:                2

Endpoint Descriptor:
bEndpointAddress:       0x81
Transfer Type:          Bulk
wMaxPacketSize:       0x0200 (512)
bInterval:              0x01

Endpoint Descriptor:
bEndpointAddress:       0x02
Transfer Type:          Bulk
wMaxPacketSize:       0x0200 (512)
bInterval:              0x01
```

```
Devices Connected: 1   Hubs Connected: 0
```

4. Now open the **portinterface** from the following path. *<USB20HR Installation Path>***\usb20hr_***<licensetype>***\software\utilities\portinterface.**

5. For use of portinterface Application, refer **readme.txt** file in *<USB20HR Installation Path>***\usb20hr_***<licensetype>***\software\utilities\portinterface.**

☞ *<USB20HR Installation Path>* is the installation directory. The default installation directory is **c:\altera\<version #>\ip\sls**. *<licensetype>* is the IP Core license type. The licensetype can be **ocp_eval** or **oc_full**.

This is the hardware tutorial you learned for the Nios II Development Kit (2C35). By this way, you can make the USB 2.0 system for any board by adding or removing the components as per your requirement.

5. Simulating the System in Modelsim Using Nios II IDE

We will run a simple test application of port interface on the modelsim to perform simulation of the design. We will be using Nios II Integrated Development Environment (IDE) to run our software on top of Nios II System.

Start **NIOS II IDE by following one of the two options**

Start>programs>altera>Nios II Development Kit 7.2>NiosII IDE

or

Click **System Generation tab** of the SOPC Builder window then click the **Run NiosII IDE button.**

Creating the Project

For creating the project, follow the steps mentioned in "Creating the Project" on page 39 of chapter 4.

System Library Settings

For system library settings, follow the steps mentioned in "System Library Settings" on page 40 of chapter 4.

Building and Managing the Project

For building and managing the project, follow the steps mentioned in "Building and Managing the Project" on page 41 of chapter 4.

After successfully building the system and before running the program, do the following modifications in **usb20hr_refdes.v** file:

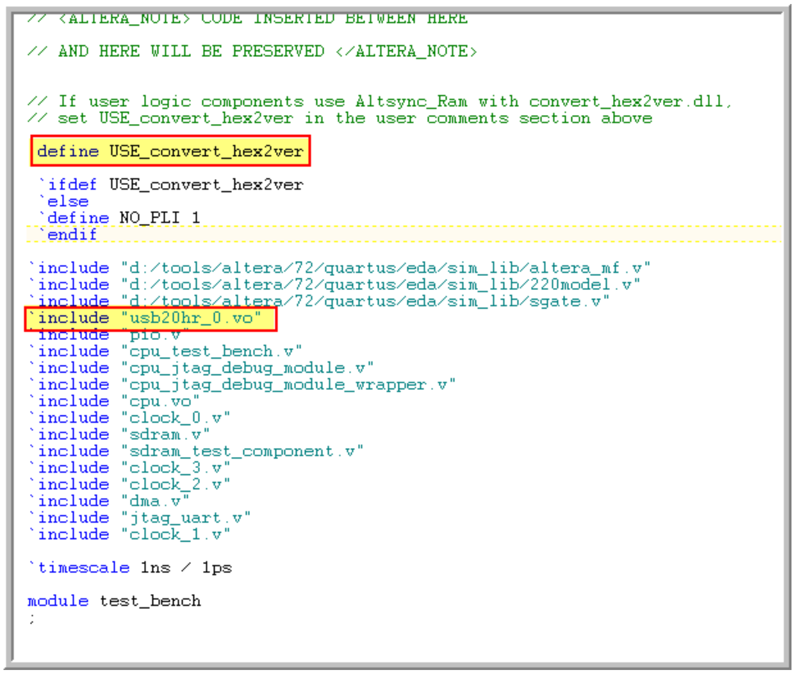
1. The USB20HR component uses Mega wizard RAM for enumeration data and for endpoint buffer memory, so simulation within ModelSim-Altera 6.1g add "**define USE_convert_hex2ver"** in User comment area of **usb20hr_refdes.v** file. See Figure 5-1.
2. Now to perform a simulation of the design you will need ".VO" file for USB20HR component. Select **usb20hr_0.vo** file from the following path:
*<USB20HR IP installation path>***usb20hr**_*<licensetype>***hardware**\

simulation\utmi
and copy and add this file into current project directory. Also include this file into usb20hr_refdes.v file. See Figure 5-1.

☞ If you want to perform a simulation of ULPI interface then select **usb20hr_0.vo** file from the following path:
<*USB20HR IP installation path*>**\usb20hr_**<*licensetype*> **\hardware\simulation\ulpi**

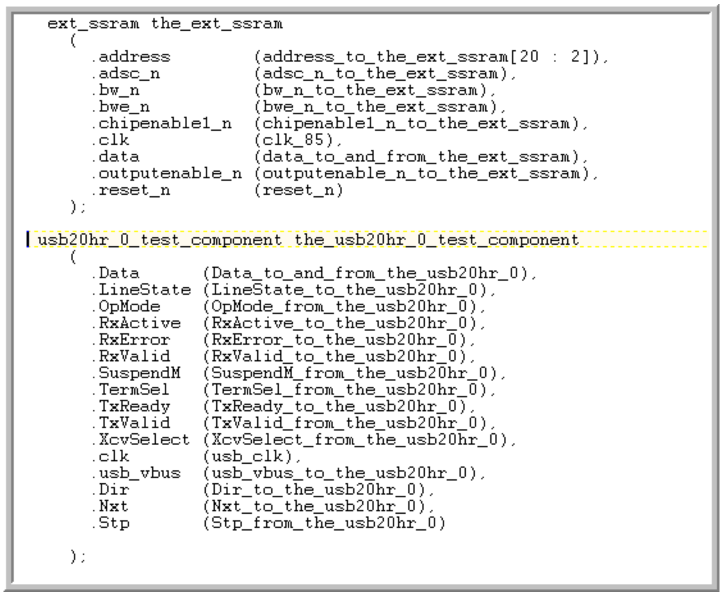
3. Create an instance of **usb20hr_0_test_component** into test bench area of **USB20hr_refdes.v** file. See Figure 5-2.

Figure 5-1. usb20hr_refdes.v file (Part 1)



```
// <ALTERA_NOTE> CODE INSERTED BETWEEN HERE

// AND HERE WILL BE PRESERVED </ALTERA_NOTE>


// If user logic components use Altsync_Ram with convert_hex2ver.dll,
// set USE_convert_hex2ver in the user comments section above

 define USE_convert_hex2ver

 `ifdef USE_convert_hex2ver
 `else
 `define NO_PLI 1
 `endif

`include "d:/tools/altera/72/quartus/eda/sim_lib/altera_mf.v"
`include "d:/tools/altera/72/quartus/eda/sim_lib/220model.v"
`include "d:/tools/altera/72/quartus/eda/sim_lib/sgate.v"
`include "usb20hr_0.vo"
`include "pio.v"
`include "cpu_test_bench.v"
`include "cpu_jtag_debug_module.v"
`include "cpu_jtag_debug_module_wrapper.v"
`include "cpu.vo"
`include "clock_0.v"
`include "sdram.v"
`include "sdram_test_component.v"
`include "clock_3.v"
`include "clock_2.v"
`include "dma.v"
`include "jtag_uart.v"
`include "clock_1.v"

`timescale 1ns / 1ps

module test_bench
;
```

Figure 5-2. usb20hr_refdes.v file (Part 2)



```
ext_ssram the_ext_ssram
  (
    .address        (address_to_the_ext_ssram[20 : 2]),
    .adsc_n         (adsc_n_to_the_ext_ssram),
    .bw_n           (bw_n_to_the_ext_ssram),
    .bwe_n          (bwe_n_to_the_ext_ssram),
    .chipenable1_n  (chipenable1_n_to_the_ext_ssram),
    .clk            (clk_85),
    .data           (data_to_and_from_the_ext_ssram),
    .outputenable_n (outputenable_n_to_the_ext_ssram),
    .reset_n        (reset_n)
  );

usb20hr_0_test_component the_usb20hr_0_test_component
  (
    .Data      (Data_to_and_from_the_usb20hr_0),
    .LineState (LineState_to_the_usb20hr_0),
    .OpMode    (OpMode_from_the_usb20hr_0),
    .RxActive  (RxActive_to_the_usb20hr_0),
    .RxError   (RxError_to_the_usb20hr_0),
    .RxValid   (RxValid_to_the_usb20hr_0),
    .SuspendM  (SuspendM_from_the_usb20hr_0),
    .TermSel   (TermSel_from_the_usb20hr_0),
    .TxReady   (TxReady_to_the_usb20hr_0),
    .TxValid   (TxValid_from_the_usb20hr_0),
    .XcvSelect (XcvSelect_from_the_usb20hr_0),
    .clk       (usb_clk),
    .usb_vbus  (usb_vbus_to_the_usb20hr_0),
    .Dir       (Dir_to_the_usb20hr_0),
    .Nxt       (Nxt_to_the_usb20hr_0),
    .Stp       (Stp_from_the_usb20hr_0)

  );
```

☞

1. If you are using name of the component different then the **usb20hr_0** inside the SOPC builder then accordingly you will have to modify the name inside the port mapping area.
2. Also make sure that the .vo file and its module name have the same name as the USB component name used in the SOPC builder.
3. During UTMI simulation ULPI signals will be floating and similarly during ULPI simulation UTMI signals will be floating.

Running the Program on Nios II System

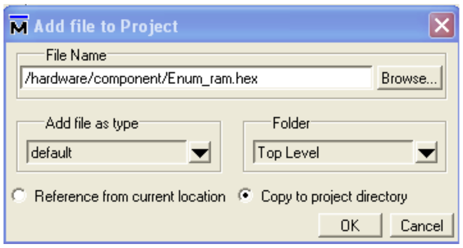
To run your program on Nios II Model Sim, perform the following steps:

1. To run the program **Right click on the project.** Click on the **Run As Nios II Model Sim**
2. You will see that the ModelSim Altera 6.1g will be opened automatically.
3. Right click in Workspace select **Add to Project --> Exiting File.**

4. Browse the **Enum_ram.hex** file from this path *<USB20HR IP Installation Directory>*\**usb20hr_***<licensetype>* **\\hardware\\component.**

5. Click on **Copy to project directory**. See Figure 5-3.

Figure 5-3. Adding Enum_ram.hex file



/hardware/component/Enum_ram.hex

Add file as type: default

Folder: Top Level

Reference from current location [✓ Copy to project directory]

6. Type **'s'** command in the command prompt of the ModelSim-Altera 6.1g. The design gets compiled but will not be loaded into the simulation workspace.

   USB20HR IP uses precompiled library so in order to use precompile part, user need to create a simulation configuration and the precompile library as explained in following two steps.

7. On the left hand side of the ModelSim-Altera 6.1g, in the **"workspace"** part, right click and select **add to "project / simulation configuration**".

8. Select **test_bench** to simulate from **"design /work/"**
   - Under **library** browse to the path where precompiled parts are placed.

     <USB20HR IP installation path>\**usb20hr_***<licensetype>*\\ **hardware\\simulation\\ulpi\\ usb20hr_ulpi_lib_6.1g. or**

     <USB20HR IP installation path>\***usb20hr_****<licensetype>*\\ **hardware\\simulation\\utmi\\usb20hr_utmi_lib_6.1g.**

     Here select **usb20hr_utmi_lib_6.1g** for **UTMI simulation.** (precompile library version should be same as ModelSim-Altera 6.1g version)

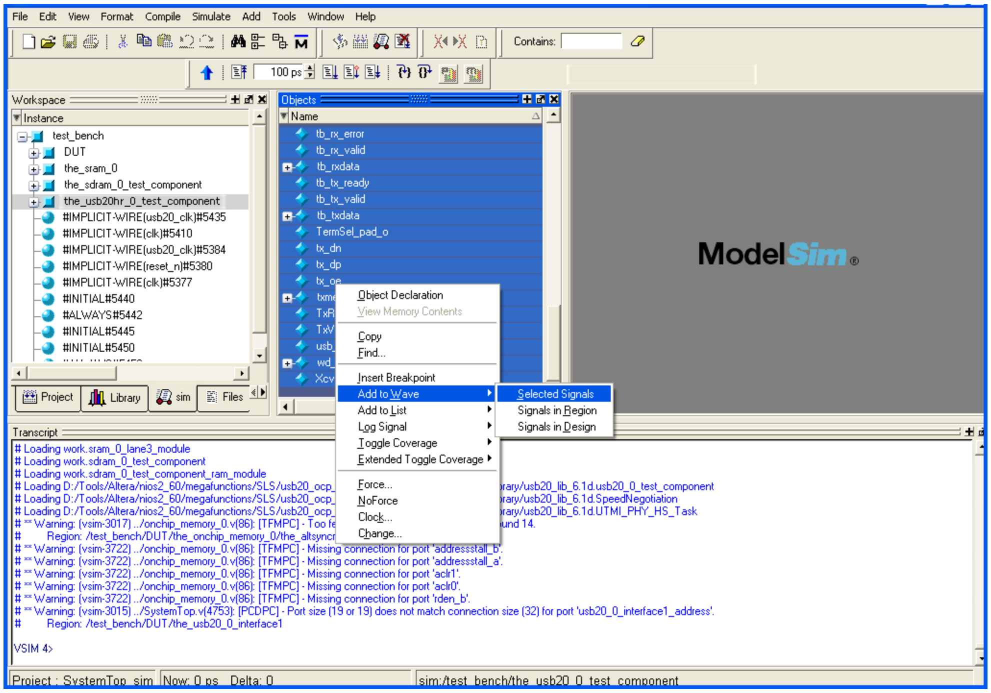
- Under **library** browse to the path where ***altera_mf*** is placed.
  *<ALTERA Installation Path>***ModelTech_ae\altera\Verilog**

☞ If you are simulating for ULPI, select the library as **usb20hr_ulpi_lib_6.1g.**

9. Click **OK**. Now you are ready to start simulation. Double click on the new **generated simulation configuration**, the design gets loaded.

10. Right click on the **instances** you want to simulate and add to the **wave window** as shown in the Figure 5-4.

Figure 5-4. Adding Instance to Wave Window



```
# Loading work.sram_0_lane3_module
# Loading work.sdram_0_test_component
# Loading work.sdram_0_test_component_ram_module
# Loading D:/Tools/Altera/nios2_60/megafunctions/SLS/usb20_ocp_          rary/usb20_lib_6.1d.usb20_0_test_component
# Loading D:/Tools/Altera/nios2_60/megafunctions/SLS/usb20_ocp_          rary/usb20_lib_6.1d.SpeedNegotiation
# Loading D:/Tools/Altera/nios2_60/megafunctions/SLS/usb20_ocp_          rary/usb20_lib_6.1d.UTMI_PHY_HS_Task
# ** Warning: (vsim-3017) ../onchip_memory_0.v(86): [TFMPC] - Too fe          und 14.
#       Region: /test_bench/DUT/the_onchip_memory_0/the_altsyncr
# ** Warning: (vsim-3722) ../onchip_memory_0.v(86): [TFMPC] - Missing connection for port 'addressstall_b'.
# ** Warning: (vsim-3722) ../onchip_memory_0.v(86): [TFMPC] - Missing connection for port 'addressstall_a'.
# ** Warning: (vsim-3722) ../onchip_memory_0.v(86): [TFMPC] - Missing connection for port 'aclr1'.
# ** Warning: (vsim-3722) ../onchip_memory_0.v(86): [TFMPC] - Missing connection for port 'aclr0'.
# ** Warning: (vsim-3722) ../onchip_memory_0.v(86): [TFMPC] - Missing connection for port 'rden_b'.
# ** Warning: (vsim-3015) ../SystemTop.v(4753): [PCDPC] - Port size (19 or 19) does not match connection size (32) for port 'usb20_0_interface1_address'.
#       Region: /test_bench/DUT/the_usb20_0_interface1

VSIM 4>
```

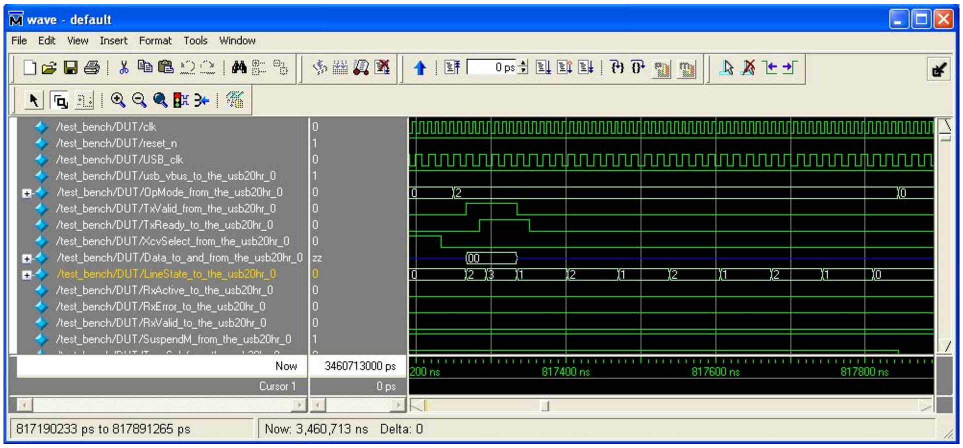
Project : SystemTop_sim | Now: 0 ps Delta: 0 | sim:/test_bench/the_usb20_0_test_component

11. Now type **Run-all** to run the test component.This will take few minutes.

12. You will see different Waveform generated as shown in simulation waveform window. See Figure 5-5. , Figure 5-6., Figure 5-7.

- This behavioral model is for the host. Host resets the device and waits for Chirp K from the device to start speed negotiation. You will see a message on the ModelSim-Altera 6.1g console.
- After that host **sends the chirp sequence**(KJKJKJ) for the speed negotiation with the device in the reset mode. See Figure 5-5.
- Later then, word enumeration process will start. In enumeration process host **grabes the necessary information** about the descriptors from the device for further communication.See Figure 5-6.(host reads 1st descriptor.)
- In **data Transfer mode**(Bulk Out) host transfers 4 byte address (00220000) and 4byte data(11223344) with write operation opcode for port interface application to the device See Figure 5-7.

Figure 5-5. Simulation Waveform Window-showing Sending of Chirp Sequence



```
/test_bench/DUT/clk                          0
/test_bench/DUT/reset_n                      1
/test_bench/DUT/USB_clk                      0
/test_bench/DUT/usb_vbus_to_the_usb20hr_0    1
/test_bench/DUT/OpMode_from_the_usb20hr_0    0
/test_bench/DUT/TxValid_from_the_usb20hr_0   0
/test_bench/DUT/TxReady_to_the_usb20hr_0     0
/test_bench/DUT/XcvSelect_from_the_usb20hr_0 0
/test_bench/DUT/Data_to_and_from_the_usb20hr_0 zz
/test_bench/DUT/LineState_to_the_usb20hr_0   0
/test_bench/DUT/RxActive_to_the_usb20hr_0    0
/test_bench/DUT/RxError_to_the_usb20hr_0     0
/test_bench/DUT/RxValid_to_the_usb20hr_0     0
/test_bench/DUT/SuspendM_from_the_usb20hr_0  1
                                  Now        3460713000 ps
                                  Cursor 1   0 ps
817190233 ps to 817891265 ps    Now: 3,460,713 ns   Delta: 0
```

Figure 5-6. Simulation Waveform Window-Showing reading of 1st descriptor

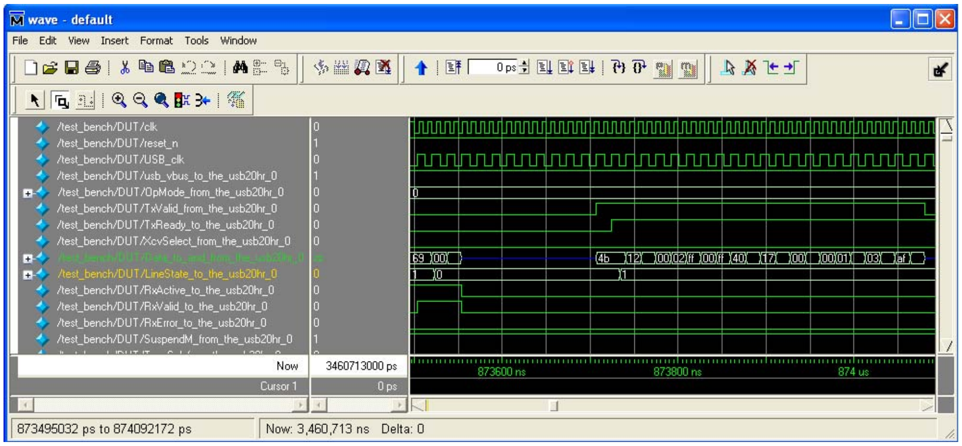
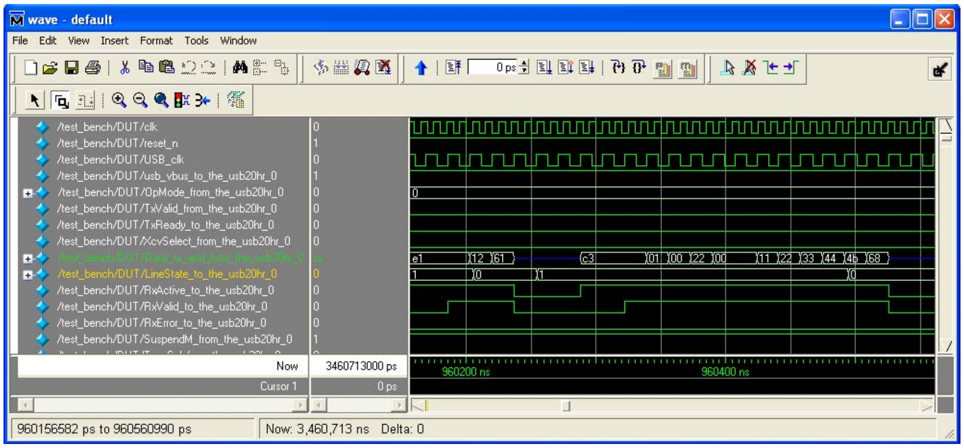

wave - default
File Edit View Insert Format Tools Window
/test_bench/DUT/clk
/test_bench/DUT/reset_n
/test_bench/DUT/USB_clk
/test_bench/DUT/usb_vbus_to_the_usb20hr_0
/test_bench/DUT/OpMode_from_the_usb20hr_0
/test_bench/DUT/TxValid_from_the_usb20hr_0
/test_bench/DUT/TxReady_to_the_usb20hr_0
/test_bench/DUT/XcvSelect_from_the_usb20hr_0
/test_bench/DUT/LineState_to_the_usb20hr_0
/test_bench/DUT/RxActive_to_the_usb20hr_0
/test_bench/DUT/RxValid_to_the_usb20hr_0
/test_bench/DUT/RxError_to_the_usb20hr_0
/test_bench/DUT/SuspendM_from_the_usb20hr_0
Now 3460713000 ps
Cursor 1 0 ps
873600 ns
873800 ns
874 us
873495032 ps to 874092172 ps
Now: 3,460,713 ns Delta: 0

Figure 5-7. Simulation Waveform Window-Data Transfer Mode


wave - default
File Edit View Insert Format Tools Window
/test_bench/DUT/clk
/test_bench/DUT/reset_n
/test_bench/DUT/USB_clk
/test_bench/DUT/usb_vbus_to_the_usb20hr_0
/test_bench/DUT/OpMode_from_the_usb20hr_0
/test_bench/DUT/TxValid_from_the_usb20hr_0
/test_bench/DUT/TxReady_to_the_usb20hr_0
/test_bench/DUT/XcvSelect_from_the_usb20hr_0
/test_bench/DUT/LineState_to_the_usb20hr_0
/test_bench/DUT/RxActive_to_the_usb20hr_0
/test_bench/DUT/RxValid_to_the_usb20hr_0
/test_bench/DUT/RxError_to_the_usb20hr_0
/test_bench/DUT/SuspendM_from_the_usb20hr_0
Now 3460713000 ps
Cursor 1 0 ps
960200 ns
960400 ns
960156582 ps to 960560990 ps
Now: 3,460,713 ns Delta: 0

You have learned here the steps for simulating the system with USB20HR IP, acting as an avalon slave, and Nios II processor acting as an avalon master. Similiarly, you can simulate and verify your own system.

Appendix A: Pin Assignments for Nios II Development Kit (2C35)

Table 5-1 shows the pin assignments for the Nios II Development Kit (2C35).

Table 5-1. Pin Assignments for the Nios II Development Kit (2C35)

FPGA Pin No.	Pin Name
PIN_N26	usb_clk
PIN_B13	clk_in
PIN_AB3	address_to_the_ext_ssram[0]
PIN_AB4	address_to_the_ext_ssram[1]
PIN_G5	address_to_the_ext_ssram[2]
PIN_G6	address_to_the_ext_ssram[3]
PIN_C2	address_to_the_ext_ssram[4]
PIN_C3	address_to_the_ext_ssram[5]
PIN_B2	address_to_the_ext_ssram[6]
PIN_B3	address_to_the_ext_ssram[7]
PIN_L9	address_to_the_ext_ssram[8]
PIN_F7	address_to_the_ext_ssram[9]
PIN_L10	address_to_the_ext_ssram[10]
PIN_J5	address_to_the_ext_ssram[11]
PIN_L4	address_to_the_ext_ssram[12]
PIN_C6	address_to_the_ext_ssram[13]
PIN_A4	address_to_the_ext_ssram[14]
PIN_B4	address_to_the_ext_ssram[15]
PIN_A5	address_to_the_ext_ssram[16]
PIN_B5	address_to_the_ext_ssram[17]
PIN_B6	address_to_the_ext_ssram[18]
PIN_A6	address_to_the_ext_ssram[19]

Table 5-1. Pin Assignments for the Nios II Development Kit (2C35)

FPGA Pin No.	Pin Name
PIN_C4	address_to_the_ext_ssram[20]
PIN_G9	adsc_n_to_the_ext_ssram
PIN_M3	bw_n_to_the_ext_ssram[0]
PIN_M2	bw_n_to_the_ext_ssram[1]
PIN_M4	bw_n_to_the_ext_ssram[2]
PIN_M5	bw_n_to_the_ext_ssram[3]
PIN_C7	chipenable1_n_to_the_ext_ssram
PIN_L2	data_to_and_from_the_ext_ssram[0]
PIN_L3	data_to_and_from_the_ext_ssram[1]
PIN_L7	data_to_and_from_the_ext_ssram[2]
PIN_L6	data_to_and_from_the_ext_ssram[3]
PIN_N9	data_to_and_from_the_ext_ssram[4]
PIN_P9	data_to_and_from_the_ext_ssram[5]
PIN_K1	data_to_and_from_the_ext_ssram[6]
PIN_K2	data_to_and_from_the_ext_ssram[7]
PIN_K4	data_to_and_from_the_ext_ssram[8]
PIN_K3	data_to_and_from_the_ext_ssram[9]
PIN_J2	data_to_and_from_the_ext_ssram[10]
PIN_J1	data_to_and_from_the_ext_ssram[11]
PIN_H2	data_to_and_from_the_ext_ssram[12]
PIN_H1	data_to_and_from_the_ext_ssram[13]
PIN_J3	data_to_and_from_the_ext_ssram[14]
PIN_J4	data_to_and_from_the_ext_ssram[15]
PIN_H3	data_to_and_from_the_ext_ssram[16]
PIN_H4	data_to_and_from_the_ext_ssram[17]
PIN_G1	data_to_and_from_the_ext_ssram[18]
PIN_G2	data_to_and_from_the_ext_ssram[19]
PIN_F2	data_to_and_from_the_ext_ssram[20]

Table 5-1. Pin Assignments for the Nios II Development Kit (2C35)

FPGA Pin No.	Pin Name
PIN_F1	data_to_and_from_the_ext_ssram[21]
PIN_K8	data_to_and_from_the_ext_ssram[22]
PIN_K7	data_to_and_from_the_ext_ssram[23]
PIN_G4	data_to_and_from_the_ext_ssram[24]
PIN_G3	data_to_and_from_the_ext_ssram[25]
PIN_K6	data_to_and_from_the_ext_ssram[26]
PIN_K5	data_to_and_from_the_ext_ssram[27]
PIN_E2	data_to_and_from_the_ext_ssram[28]
PIN_E1	data_to_and_from_the_ext_ssram[29]
PIN_J8	data_to_and_from_the_ext_ssram[30]
PIN_J7	data_to_and_from_the_ext_ssram[31]
PIN_D5	outputenable_n_to_the_ext_ssram
PIN_J9	bwe_n_to_the_ext_ssram
PIN_D7	ssram_adsp_n
PIN_H10	ssram_adv_n
PIN_E5	sram_clk
PIN_C5	pld_clear_n
PIN_AC10	out_port_from_the_led_pio[0]
PIN_W11	out_port_from_the_led_pio[1]
PIN_W12	out_port_from_the_led_pio[2]
PIN_AE8	out_port_from_the_led_pio[3]
PIN_AF8	out_port_from_the_led_pio[4]
PIN_AE7	out_port_from_the_led_pio[5]
PIN_AF7	out_port_from_the_led_pio[6]
PIN_AA11	out_port_from_the_led_pio[7]
For UTMI Interface	
PIN_M21	SnapOnReset
PIN_V24	UNI-BIDI

Table 5-1. Pin Assignments for the Nios II Development Kit (2C35)

FPGA Pin No.	Pin Name
PIN_V26	DataBus16_8
PIN_V23	linestate[0]
PIN_V25	linestate[1]
PIN_H25	rxactive
PIN_J24	rxerror
PIN_H26	rxvalid
PIN_T23	txready
PIN_P17	usb_vbus
PIN_J26	Xcvselect
PIN_R17	txvalid
PIN_K24	termselect
PIN_K19	suspendm
PIN_J25	opmode[0]
PIN_K23	opmode[1]
PIN_E25	usbdata[0]
PIN_F24	usbdata[1]
PIN_F23	usbdata[2]
PIN_J21	usbdata[3]
PIN_J20	usbdata[4]
PIN_F25	usbdata[5]
PIN_F26	usbdata[6]
PIN_N18	usbdata[7]
For ULPI Interface	
PIN_U21	usbdata[0]
PIN_P17	usbdata[1]
PIN_T19	usbdata[2]
PIN_R17	usbdata[3]
PIN_R19	usbdata[4]

Table 5-1. Pin Assignments for the Nios II Development Kit (2C35)

FPGA Pin No.	Pin Name
PIN_U26	usbdata[5]
PIN_T17	usbdata[6]
PIN_T18	usbdata[7]
PIN_V26	phy_cs_n
PIN_V24	Phy_reset_n
PIN_V25	Dir
PIN_V23	Nxt
PIN_U20	Stp