# A. FPGA Basics Manuals

In this practical course, Field-Programmable Gate Array (FPGA) is used as the experimental platform, which means to creation of a hardware description for the FPGA and writing software for the created hardware. For experiments, toolchain from Altera Corporation is chosen, which are Quartus II for hardware development and the Nios2SBT for software development. The tutorial is specific for Altera toolchain.

## A.1. Tool Chain Installation

### A.1.1. Installation

Please note that this installation description applies to installing the Altera Design Suite v12 on Windows7 64bit only. To install the toolchain, one can either use the Altera Installer or our pre-downloaded package. Details of the two approaches are show below:

• Altera Installer:

To install all the required software, Altera Corporation provides a free download package on its website, called "Altera Design Suite (Altera Installer)". This is an installer package, which downloads all selected installation components during the installation process from the Internet. To start the installation, execute the installer package as an administrative user (right-click and select "run as administrator"). This step is very important, as the USB-Blaster will not work if you install the software as a low-privileged user.

Select products Select the software products	you want to install.				
	Products			Install Size	Download Size
		ubscription Editio	n (includes Nios II EDS)		
A CO		II software (64-b		810 M	122 M
All	Device Factoria			-	-
	Quartus II W	/eb Edition (Free	) (includes Nios II EDS)	4.5 G	1.3 G
	A V Device Fa	amilies		240 M	226 M
	Arria	II GX		472 M	456 M
/////	Cyclo			29 M	25 M
		ne III/III LS		213 M	
	Cyclo			240 M	
		ne IV GX		179 M	
> Introduction	Cyclo			541 M	
		cy Families		1.5 M	
	MAX			3.8 M	
> License	MAX	-	<i>(</i> - )	4.2 M	
		tera Starter Editio	on (Free)	3.4 G	
> Installer Setup	ModelSim-Alt	tera Edition		3.4 G	
	DSP Builder			254 M	51 M
> Select Destination	Description				
Coloris Decidentes					
> Select Products	Includes Cyclone I	V E device suppo	rt		
					~
> Summary					
		Install:	Download:		
> Installation	Space Required:	4.5 G	1.3 G	Se	elect/Deselect All
	Space Available:	88 G	88 G		
	Space Available:	000	00 0		

Figure 1: Component selection in the Altera Installer

• Pre-downloaded package:

Go to directory *tools/altera-installation*, double click file 12.1sp1\_243\_quartus\_free\_windows.exe, the installation will start as previously described.

Altera offers a paid (Quartus II Subscription Edition) and a free (Quartus II Web Edition) version of its Quartus development suite. The free version includes most of the features for beginners or even professionals. One striking difference is the missing Intellectual Property (IP) license files for several more complex components. These components, which are not fully licensed to free edition users include, but are not limited to NiosII/s or NiosII/f processors (the NiosII/e processor does not require a license), the DDR Random Access Memory (RAM) controller or the Triple Speed Ethernet (TSE) Ethernet Media Access Control (MAC) component.

## A.1.2. Tips

- 1. If running Quartus II v12.0 on a 64bit machine, one might still have difficulties finding the USB Blaster hardware in the Programmer tool. Altera does not ship the 64bit JTAG-Server with the QuartusII Web-Edition. To resolve this issue, install the standalone Programmer Software from https://www.altera.com/download/software/prog-software, which contains a 64bit Programmer and Joint Test Action Group (JTAG)-Server (update: as of Quartus II v12.1, there is no need to separately install the standalone Programmer).
- 2. The installation package does include the driver for the USB Blaster device. The device driver, however, does not automatically installed during the installation of the Altera toolchain. To install the device drive, go to the "Device Manager" and reinstall the drive, as shown in Fig. 2. When reinstalling the driver, locate the driver at directory: c:\altera\12.1sp1\quartus\drivers\usb\_blaster.

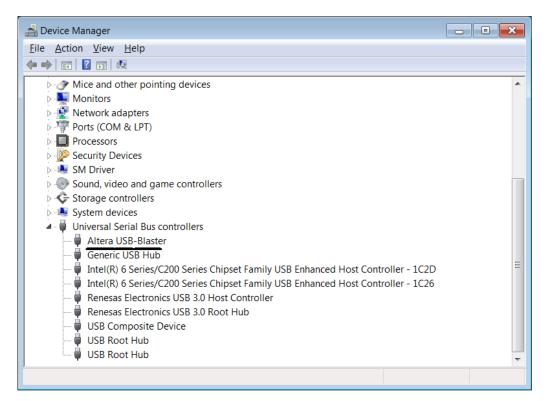


Figure 2: Altera USB Blaster

## A.2. Hello-World SoC

### A.2.1. Prerequisite

- Successfully finish Section A.1.
- The Hello-World package, which includes the hardware description files, e.g., .sof and .sopcinfo, and the software source code.

### A.2.2. Download Hardware Design to FPGA

Now that you have created a hardware description for your FPGA and have successfully generated/compiled it, you can proceed to program your design. Programming your hardware design onto the FPGA can be done from the console or by using the *Programmer* User Interface (UI).

Programming the FPGA chip will keep the hardware description in the chip until the next programming or power loss. This means that the chip has to be programmed again each time the development board is turned off. To avoid reprogramming upon each power cycle, the FPGA chip can be set to an automatic programming mode. In this mode, the FPGA will take the hardware description from a flash memory and program itself upon start. But there is a restriction to this automatic programming mode: It can only be used, if all the components used in the current hardware design are properly licensed (e.g., the TSE Ethernet MAC component is not licensed in Quartus II Web Edition). See section ?? for a short summary of the differences between Quartus versions.

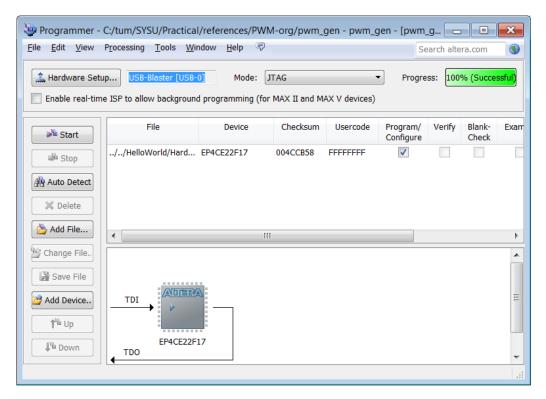


Figure 3: Main window of Programmer

One can start the Programmer manually or even fire it up from your Quartus instance (see figure 3 for a screenshot of the Programmer main window). To do so, open your hardware design project and select "Tools/Programmer" from the menu bar. If you start Programmer from Quartus, the hardware design you want to program should already be preselected. Otherwise, you can use the *Add File...* button to add your *.sof* of *.pof* file. If your FPGA chip is not

selected yet, you can use the *Add Device...* button to select your FPGA chip. Make sure to select the correct chip number, otherwise you risk damaging your hardware.

🔮 Quart	tus II	
1		_stuff/web_server_improved_restored/DE2_115_WEB_SERVER_time_limited.sof the OpenCore Plus feature that will not work after the hardware evaluation ne details.
		ОК

Figure 4: Time limitation warning of Programmer

If you are using a non-licensed component in your hardware design, you can still use that component, but you can only use it while your FPGA is connected to your Computer running Programmer. A message will appear, which reminds you of that situation (the message might look like figure 4).

Hardware Settings JTAG Settings Select a programming hardware setup hardware setup applies only to the cu	to use when prog		s. This programming
Currently selected hardware: USB-I	Blaster [USB-0]		•
Available hardware items			
Hardware	Server	Port	Add Hardware
USB-Blaster	Local	USB-0	Remove Hardware
			Close

Figure 5: Hardware setup window of Programmer

Now you need to set your USB Blaster connection to the FPGA chip. Simply click the Hardware Setup... button and a window like 5 will pop up. If you are using USB Blaster for programming (on DE2-115 boards, you are using USB Blaster), your USB Blaster will show up in the list of Available hardware items. Select it from the Currently selected hardware dropdown list, close the Hardware Setup window and you should be ready to start programming. The Start button should be enabled, if you have added a .sof of .pof file, if you have selected a FPGA device and if your USB Blaster is selected. Start programming by clicking that button. The progress bar should show Success after programming finished successfully. This means that your hardware design was programmed onto the FPGA chip. If you are using non-licensed (time-limited) components, a popup window will appear after successful programming (see figure 4).

Leave this window open and keep the FPGA chip connected to your computer until you have finished using your hardware design.

Sometimes you could have issues with your USB Blaster hardware detection. If your USB Blaster is not selected (from your previous programming), try to select it again in the Hardware Setup... window. If it does not show up in the list of Available hardware items, check your physical connection to your development board. Make sure the USB cable is connected properly to your computer and the FPGA hardware. Also make sure, that your FPGA hardware is connected to a power source and the power switch is turned on. If the USB programmer is still not shown in the list, close the hardware selection window and open it again (you may need to repeat this procedure several times). In fact, simply retrying the failed operation helps in many cases for the Programmer (not only the issues, which are listed here). Another way to get the Programmer working is to select Auto Detect or turning your development board off and on again. If your USB Blaster did never show up in the list of available hardware components, there might be a driver issue on your system. This could be, because you ran the installation process as a non-privileged user. Also make sure, that your USB Blaster is listed in Device Manager (on Windows machines).

Once the download is complete, the *Progress* bar of Programmer window, located at the top-right of Fig. 3, the will indicated 100% (Successful).

#### A.2.3. Setting up the BSP

After finishing all hardware related aspects of this introduction into the DE2-115 FPGA, we are ready to start working on the software. Typically, NiosII applications are developed using the Nios2SBT (NiosII Software Build Tools), a modified version of the eclipse Integrated Development Environment (IDE). NiosII applications usually consist of two parts:

- The NiosII Board Support Package (BSP), a software library and runtime environment, which is customized to a Central Processing Unit (CPU) and a hardware design.
- The NiosII application, which is the program implementing the functionality required by the user and relying on functions provided by the BSP

Start the Nios2SBT and select a workspace if you have not done so yet. To create a new NiosII BSP, select "File/New/Nios II Board Support Package" (Fig. 6), which will open the BSP creation wizard (see Fig. 7). Choose a name for the BSP project and select the path to the *.sopcinfo* file, which has been generated for your NiosII system. The *.sopcfile* should be located in the QuartusII project folder. You can change the location of the BSP (be default, it is saved in a subfolder of the QuartusII project folder). Select the CPU for which the BSP shall be created (in case you have more than one CPU in your NiosII system). A BSP can only be generated for one CPU at once. If you have more CPUs in one hardware design, you have to create a BSP for each processor. Confirm your selection by pressing the *Finish* button of the BSP creation wizard. Your BSP project will appear in the *Project Explorer* of the Nios2SBT.

#### A.2.4. Setting up the application project

Now we are creating the actual NiosII application project, which uses the BSP, that we just created. Choose "File/New/Nios II Application" (Fig. 8) to open the *Nios II Application wizard* (see figure 9). Enter a name for your application project and select the BSP for your project (the BSP that you created in section A.2.3). Create the project by selecting the *Finish* button. The project appears in the *Project Explorer*, as shown in Fig. 10. Basically, we just create two projects, one for the BSP and the other for the application.

Note that although the application project has been created, there is no application-specific source code within the project. The next step is to create the application source code. To input ΠП



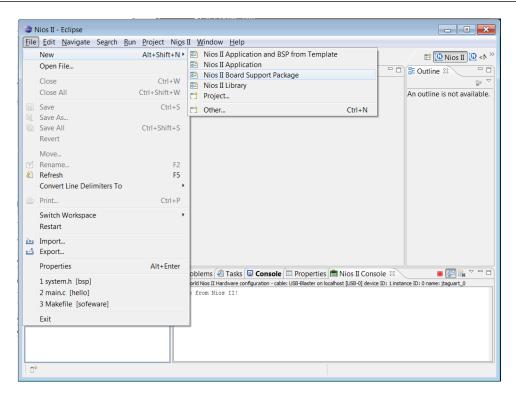


Figure 6: BSP creation wizard Location

Nios II Board Support Package	×
Nios II Board Support Package	
Create a new Nios II Software Build Tools board support package project	
Project name: 1-bsp	
SOPC Information File name: C:\tum\SYSU\Practical\2-LED\Hardware\DE0_NANO_SOPC.sop	
Vse default location	
Location: C:\tum\SYSU\Practical\2-LED\Hardware\software\1-bsp	
CPU: cpu 🗸	
BSP type: Altera HAL	
BSP type version: default	
Additional arguments:	
	*
Command:tive path	
? Finish Cancel	

Figure 7: BSP creation wizard



	Nios II - Eclipse			
<u>F</u> ile	<u>E</u> dit <u>N</u> avigate Se <u>a</u> rch <u>R</u> un <u>P</u> roject Ni	s II	<u>W</u> indow <u>H</u> elp	
	New Alt+Shift+M	•	Nios II Application and BSP from Template	🗈 🔯 Nios II 🔯 <n th="" ≫<=""></n>
	Open File		Nios II Application	□ □ BE Outline 🛛 □
	Close Ctrl+V		Nios II Board Support Package	
	Close All Ctrl+Shift+V		Nios II Library Project	An outline is not available.
	Save Ctrl+	2	Other Ctrl+N	
	Save As Save All Ctrl+Shift+	Г		
4G	Revert			
	Move			
Ľ	Rename F.			
\$	Refresh F			
	Convert Line Delimiters To	<u>}</u>		
	Print Ctrl+			
	Switch Workspace	•		
	Restart			
2 2	Import Export			
	Properties Alt+Ente		lems 🙆 Tasks 💷 Console 💷 Properties 🔚 Nios II Cons	sole 🛛 📕 🕞 🕞 🖓 🖓 🖓
	1 system.h [bsp]		I NIOS II Hardware configuration - cable: USB-Blaster on localhost [USB-0] devi	
	2 main.c [hello]		rom Nios II!	
	3 Makefile [sofeware]			
	Exit			
I∎	0 items selected			

Figure 8: Application creation wizard from Eclipse menu

Nios II Application	- • ×
Nios II Application	
Create a new Nios II Software Build Tools application project	
Project name: 4	
BSP location: C:\tum\SYSU\Practical\test-bench\2-bsp	
	Create
Use default location	
Location: C:\tum\SYSU\Practical\test-bench\4	
Additional arguments:	
	* *
Command: IV Use relative path	
? Finish	Cancel

Figure 9: Application creation wizard

пΠ



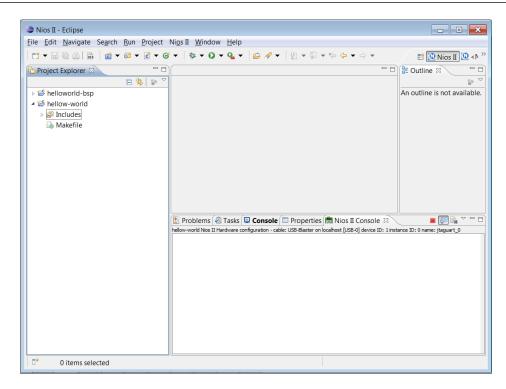


Figure 10: System with BSP and application projects

the application source code into the project, there are two approaches, i.e., using the source-code wizard and manual creation.

New Source	File	
Source File Create a new s	ource file.	
Source folder: Source file:	quartus_demo main.c	Browse
Template:	Default C source template 🔹	Configure
?	Finish	Cancel

Figure 11: New file wizard

#### Source-Code Wizard

To add a new source file to the application project, right-click on the application project and select "New/Source File". In the *New Source File* window (see figure 11), enter a name for the Source File and click *Finish* to create your file. Every project has to contain a main function, which is the entry point to your application. Start by creating this function and adding code to it. Listing 1 shows how a basic main function could look like. It demonstrates the output of text to the console by using:

```
1 printf(''Hello from Nios II!\n'');
```

and controls the LEDs, that we connected via a Parallel I/O (PIO) component by using:

```
1 IOWR_ALTERA_AVALON_PIO_DATA(PIO_LED_BASE, count & 0x01);
```

Congratulations, you created your first NiosII software application.

```
#include <stdio.h>
1
   #include <system.h>
2
   #include "altera_avalon_pio_regs.h"
3
4
5
   int main() {
     printf("Hello from Nios II!\n");
6
7
     int count = 0;
8
     int delay;
     while (1) {
9
        IOWR_ALTERA_AVALON_PIO_DATA(LED_PIO_BASE, count & 0x01);
10
11
        delay = 0;
        while (delay < 2000000) {
12
13
          delay++;
       }
14
        count++;
15
     }
16
17
     return 0;
   }
18
```

Listing 1: Example implementation of main function

#### Manual Creation

In the case that there is existing source code, one convenient way is to manually import the source files into the project. Basically, one can just copy the existing source files directly to the directory where the generated project is located. Then use the reflash key to generate the corresponding Makefile, as shown in Fig. 12.

#### A.2.5. Executing NiosII projects on the target hardware

You can execute your NiosII project from the Nios2SBT. To execute the application, right-click on the application project in the *Project Explorer* and choose "Run As/Nios II Hardware". This will compile your application, create a downloadable image, download the software image and execute it on the target hardware. Before executing, please make sure that you programmed the target hardware with the correct hardware design (see section ??).

Another option is to debug your application. To start debugging, right-click the project and select "Debug As/Nios II Hardware". This will execute the same steps as running the software and additionally start Nios2SBT's debugger.

If a window similar to figure 17 appears, your Target Connection setup may be incorrect. To setup the target connection, open the *Target Connection* tab and click the *Refresh Connections* button. Select a *Processor* to execute your target on and a *Byte Stream Device*, through which

TUM-LS6





😂 Nios II - Eclipse						
<u>F</u> ile <u>E</u> dit <u>N</u> aviga	te	Se <u>a</u> rch <u>R</u> un <u>P</u> roject Ni <u>o</u> s II	<u>W</u> indow <u>H</u> elp			
	<b></b>	₫ • 63 • 6 • 6 •	🌣 🕶 🖸 🕶 🂁 🖉 🎾	<i>🖗</i> 🔻		🗈 🙋 Nios II 🙆 < N 🏁
Project Explorer	· 23				- 8	🗄 Outline 🛛 📃 🗆
		🕒 😫 🔓 🍸				<u>6</u> 9 ▽
🛛 🖉 helloworld-b	osp	[DE0_NANO_SOPC]				An outline is not available.
🔺 😂 hellow-work	- I					
Includes		New		+		
Image: bello_wc		Go Into				
🗋 Makefile		Open in New Window				
		Сору	Ctrl+	-C		
	Ê	Paste	Ctrl+	۰V		
	×	Delete	Dele	te		
	<u>_</u>	Remove from Context	Ctrl+Alt+Shift+Dov	vn		
		Source		•		
		Move				
		Rename		F2		
	è	Import				
	4	Export				
		Build Project				
		Clean Project			🗆 Properties 🔚 Nios II Console 🙁	
	8	Refresh		F5	: USB-Blaster on localhost [USB-0] device ID: 1 insta	
		Close Project				
		Close Unrelated Projects				
		Build Configurations		•		
		Make Targets		•		
		Index		•		
		Show in Remote Systems view				
Ҏ 😂 hello		Convert To				
		Dura A.a				

Figure 12: Reflesh the application project to import source code

😂 Nios II - Eclips		▲		
Eile Edit Navig	1 × 2	Paste Delete Remove from Context Source Move Rename Import Export	Ctrl+V Delete Ctrl+Alt+Shift+Down F2	<ul> <li>Image: Image: Im</li></ul>
b I hello_w Makefil		Build Project		
Lo Makeni		Clean Project Refresh Close Project Close Unrelated Projects	F5	
		Build Configurations Make Targets Index	> >	
	*	Show in Remote Systems view Convert To Run As Debug As Profile As Team Compare With Restore from Local History Nios II Run C/C++ Code Analysis Update Linked Resources	,	Properties ■ Nios II Console
I <sup>¢</sup> 😂 hel		Properties	Alt+Enter	

Figure 13: Compile the application

text output will be transferred (see 18). Confirm your selection with Apply and start debugging the software with Debug.

This finishes the tutorial on how to run your first NiosII Application.



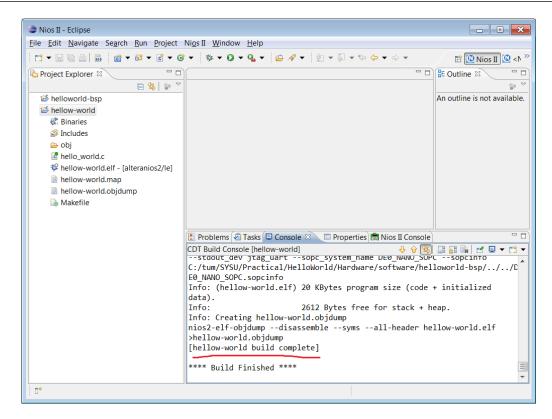


Figure 14: The result of the application compilation

😂 Nios II - Eclips		•		
<u>F</u> ile <u>E</u> dit <u>N</u> avig	Ê	Paste	Ctrl+V	
i 📑 🕶 🗔 🕞 🗠	×		Delete	
:	<u>®_</u>	Remove from Context	Ctrl+Alt+Shift+Down	
C Project Explor		Source	•	
		Move		
b 😂 helloworld		Rename	F2	An outline is not available.
▲ 😂 hellow-wo	2	Import		
▷ 🐝 Binaries ▷ 📦 Includes	⊿	Export		
Includes		Build Project		
⊳ 🛃 hello w		Clean Project		
⊳ 🕸 hellow-	8	Refresh	F5	
hellow-		Close Project		
hellow-		Close Unrelated Projects		
🚡 Makefil		Build Configurations	•	
		Make Targets	+	
		Index	*	2 Properties 🛱 Nios II Console
		Show in Remote Systems view		
		Convert To		Image: Opc_system_name_DE0_NANU_SUPCSOPCINTO
		Run As	+	▲ 1 Lauterbach ISS oworld-bsp///C
		Debug As	•	2 Local C/C++ Application     initialized
		Profile As	+	3 NIOS II Hardware
		Team	•	4 Nios II ModelSim ap.
		Compare With	•	Run Configurations
		Restore from Local History		emblesymsall-header hellow-world.elf
		Nios II	+	te]
	*	Run C/C++ Code Analysis		
		Update Linked Resources		
Ļ		Properties	Alt+Enter	<b>•</b>
Ҏ 😂 hel		<b>*</b>		

Figure 15: Run the application on the FPGA



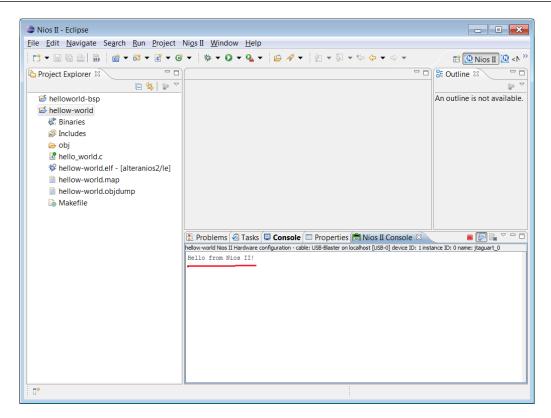


Figure 16: The result of the application execution

Debug Configurations	× ×
Create, manage, and run configurations           Image: Target Connection]: No Nios II target connection path	s were located. Check connections and that a Nios II.sof is downloaded.
Ype filter text         Ype filter text         C C/C++ Application ←         C C/C++ Attach to Ay         C C/C++ Remote App         Project LF file name:         Project LF file name:         Project LF file name:         Project LF file name:         E Launch Group         Image:         I	et Connection % Debugger & Source Common blah C:Usersmartinblah,blah,elf esystem ELF file
?	Debug Close

Figure 17: Error indicating wrong target connection path

### A.2.6. Tips

- Enable the *enable\_small\_driver* option, click the *Generate* button, and then the *Exit* button.
- No white space is allowed for all the names of the files and projects.



eate, manage, and run con Select one of the available targ	-						Ń
ype filter text C C/C++ Application ▲ C C/C++ Attach to Au	Name: blah Nios II Hardware configure Project II Target Connection Connections Processors:		Source	<u>C</u> ommon			,
C/C++ Postmorter	Cable	Device	Device ID	Instance ID	Name	Architecture	Refresh Connections
<ul> <li>C/C++ Remote App Launch Group     </li> <li>Nios II Hardware</li> </ul>	USB-Blaster on localhost [ USB-Blaster on localhost [	EP3C120	1	1 0	nios2_1 nios2_0	Nios2:3 Nios2:3	Resolve Names
blah Nios II Harc	Byte Stream Devices:						System ID Properties
blinky_a_la_Kai l	Cable	Device	Device ID	Instance ID	Name	Version	]
blinky_dual_cpu ≣ blinky_dual_cpu	USB-Blaster on localhost [ USB-Blaster on localhost [			1	jtaguart_1 jtaguart 0		
Blinky-manual-ı de2-115 Nios II I m hello_sd_3july_ri	Disable 'Nios II Console' view Quartus Project File name: < Using defaul	t .sopcinfo & .jdi f	iles extracted f	rom ELF >			
hello_sd_july02 l	System ID checks						
hello_world_0 N	Ignore mismatched system ID Ignore mismatched system timestamp						
IwIP_NIOS_II_Exa	Download						
wIP_NIOS_II_Exa	Download ELF to selected target system	em					
RTOSDemo_nio RTOSDemo_SBT	Start processor Reset the selected target system						
sdram_gang_10j 👻							
ter matched 32 of 32 items							ApplyRevert
]							

Figure 18: Choosing the target connection

😂 Nios II - Eclip	4			
File Edit Navi	1 ×	Paste Delete Remove from Context Source Move	Ctrl+V Delete Ctrl+Alt+Shift+Down	•
Include	2	Rename Import Export	F2	An outline is not available.
🚡 Makefi	2	Build Project Clean Project Refresh Close Project Close Unrelated Projects	F5	
		Build Configurations Make Targets Index	) } }	
		Show in Remote Systems view Convert To Run As Debug As Profile As Team Compare With	> > > >	Properties I Nios II Console ⊠ ■ []
×	¥	Restore from Local History Nios II Run C/C++ Code Analysis Update Linked Resources	,	BSP Editor K Nios II Command Shell Flash Programmer
		Properties	Alt+Enter	

Figure 19: BSP editor

# A.3. Self-Booting

This section describes how to create a self-booting design such that the FPGA board can boot up the designed system from flash memory once power up.



Generate E<u>x</u>it

1		TUM	I-LS6		Robotics
Rie Edit Tools Help	or - C:\tum\SYSU\Practical\HelloWorl		are\helloworld-bsp\settings.bsp		
Module Name	Module Class Name	Module Version	Driver Name	Driver Version	Enable
cpu jtag_uart onchip_memory2 pio led	altera_nios2 altera_avalon_jtag_uart altera_avalon_onchip_memory2 altera_avalon_pio	10.1 10.1 10.1 10.1 10.1	altera_nios2_hal_driver altera_avalon_jtag_uart_driver none altera avalon pio driver	default default none default	
Information Problems	Tracessing				
	Processing eption stack memory region name" set to "onchip	memory?"			
Loading drivers from e     Mapped module: "cpu"     Mapped module: "pio_     Mapped module: "jiag,     Mapped module: "jiag,     Finished loading driver	nsemble report. to use the default driver version. ed" to use the default driver version. 				
	rom settings file. Builder system info file "\\DE0_NANO_SOPC.so jtag_uart_driver.enable_small_driver" set to "true		file]"		

Figure 20: Edit the bsp options to enable small driver

#### A.3.1. Hardware Support

First of all, the designed System-on-a-Chip (SoC) needs hardware support for writing the EPCS flash. In the case that the corresponding hardware is not included in the design, this section guides you to integrate the hardware. Detailed steps are shown in the following figures.

stem Contents System Generation	Target	Clock Settings					
	Device Family: Cyclone IV E	Name	Source		MHz		
k epcs 🗙	Device runny, Cyclone IV L	clk 0	External		50.0		Add
roject 			External		laste		Remo
External Memory Interfaces	Use Conn Name	Description	Clock Base	End	IRQ 1	Tags	
Flash Interfaces     e EPCS/EPC0x1 S	✓ onchip_memory	On-Chip Memory (RAM or ROM)	[ck1]				
	s1	Avalon Memory Mapped Slave Nios II Processor	clk_0 = 0x4	0x67ff			
	instruction_ma		[CIK] CIK 0				
	data_master	Avalon Memory Mapped Master	[ck]	IRQ 0	IRQ 31		
	jtag_debug_m		[ck] 🖹 0x8	000 008			
	I jtag_uart_0	JTAG UART Avalon Memory Mapped Slave	[ck] clk 0 = 0x9				
	→ avalon_itag_s	System ID Peripheral	clk_0 = 0x9 [ck]	020 0x9027	-19		
	→ control_slave	Avalon Memory Mapped Slave	clk_0 = 0x9	0x902f			
	✓ □ pwm_gen_0	pwm_gen_v1	0				
	avalon_slave_	0 Avalon Memory Mapped Slave	clk_0 = 0x9	0x901f			
III V	Remove Edt	I Address M	lap Fitters	Filter: Default			
	components can be edited through the Co						
		ill no longer auto-assigns unique control registe	r value. This option will alway	s be turned on with default value	set to 0.		
	HDL_MODULE automatically set to pwm_	gen					
110: onchip_memory: Memory Wi	Il be initialized from onchip_memory.hex						

Figure 21: Open the design from the SOPC builder. From the "Component Library" at the top left of the figure, search EPCS component. Select the component and "Add" to the design (the red arrow).

🖳 EPCS Serial Flash Controller - epcs_flash_controller_0
EPCS Serial Flash Controller
Parameter Settings
EPCS Serial Flash Controller
Automatically select dedicated active serial interface, if supported
use dedicated active serial interface
Description
The EPCS Serial Flash Controller core enables Nios II systems to access an Altera EPCS serial configuration device connected to the FPGA.
Altera provides device drivers for the Nios II processor which allows you to read and write the EPCS from Nios II software.
You can disable dedicated Active Serial (AS) interface pins and use regular I/O pins instead, in devices that support dedicated AS interface. When you choose to user regular I/O pins or when you pick a device that does not support dedicated AS interface, the EPCS Serial Flash Controller signals are exported to the top level design.
Cancel Einish

Figure 22: After clicking the "Add" button, this figure will show up. Just click the "Finish" button.

After the operation at Fig. 22, one will go back to the main panel of the SOPC builder, similar to Fig. 21. Now double click the nios entry, which lead to Fig. 23. One needs to change the "Reset Vector" of the Nios processor to the EPCS controller.

After the generation, there should be four extra pins in the design, i.e.,data0, dclk, sce, sdo. These are dual-purpose pins. From the menu "Assignment->device-> device and pin options" (Fig. 26) to see if these pins are used as regular I/O. If not, change them into regular I/O (Fig. 27). Otherwise, there will be synthesize errors.

If it is not possible to change them in the GUI, please add following tcl script into project setting file with extension of .qsf file.

```
1 set_global_assignment -name RESERVED_FLASH_NCE_AFTER_CONFIGURATION
2 ''USE AS REGULAR IO''
3 set_global_assignment -name RESERVED_DATAO_AFTER_CONFIGURATION ''USE
4 AS REGULAR IO''
5 set_global_assignment -name RESERVED_DATA1_AFTER_CONFIGURATION ''USE
6 AS REGULAR IO''
7 set_global_assignment -name RESERVED_DCLK_AFTER_CONFIGURATION ''USE AS
8 REGULAR IO''
```

#### Listing 2: Pin Options

The next step is for pin assignment, i.e., to assign the signals of the Nios core to physical pins of the chip. Similar to defining the pin options, one can just copy the code in Listing ?? to the .qsf file. The result of this operation is shown in Fig. 28.

```
    set_location_assignment PIN_H2 -to EPCS_DATA0
    set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to EPCS_DATA0
```

```
_3 set_location_assignment PIN_H1 -to EPCS_DCLK
```

Nios II Processo	r - niosii					
👬 Nio	s II Proce	ssor			About Documentation	
arameter						
Settings						
Core Nios II Ca Core Nios II	aches and Memory Inter	faces > Advanced Featu	res > MMU and MPU S	ettings > JTAG Debug Modul	e $ ightarrow$ Custom Instructions $ ightarrow$	
Select a Nios II core:						
elect a NIOS II COre:	Nios II/e	○Nios II/s	ONios Ⅱ/f	1		
Nios II	32-bit	32-bit	32-bit			
Selector Guide		Instruction Cache	Instruction Cache			
Family: Cyclone IV E		Branch Prediction Hardware Multiply	Branch Prediction Hardware Multiply			
f <sub>system:</sub> 50,0 MHz		Hardware Divide	Hardware Divide			
cpuid: 0			Barrel Shifter Data Cache			
			Dynamic Branch Predictio	n		
erformance at 50,0 MH;	z Up to 8 DMIPS	Up to 32 DMIPS	Up to 57 DMIPS			
.ogic Usage	600-700 LEs	1200-1400 LEs	1400-1800 LEs			
lemory Usage	Two M9Ks (or equiv.)	Two M9Ks + cache	Three M9Ks + cache			
ardware Multiply:		Hardware Divide				
		Hardware Divide				
	2					
	nory: epcs_flash_controller_	0 Voffset: 0x0	0×00	008000		
xception Vector: Mem	ory: onchip_memory	▼ Offset: 0x20	0x000	04020		
Include MMU						
nly include the MMU wh	ien using an operating syste	m that explicitly supports an MMU				
ast TLB Miss Exception			Offset: 0x0			
			UND			
Include MPU						
				Can	cel < Back Next > Einish	
				Cali		

Figure 23: Changing the reset to EPCS controller.

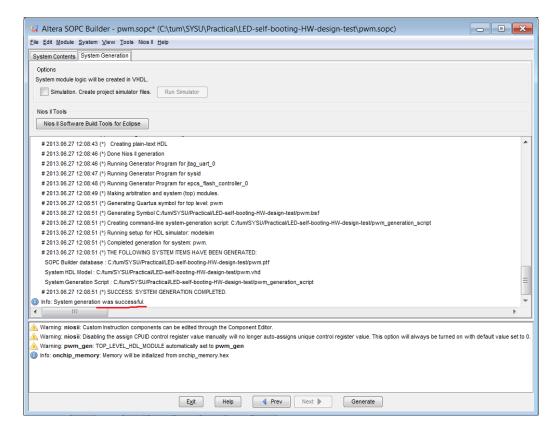
em Contents System Generation												
ponent Library	Targ	get		Clock S	ettings							
epcs 🗙	Devi	ce Family:	Cyclone IV E 🗢	Name			Source			MHz		Add
oject - 1 New Component prary Memories and Memory Controller				clk_0			External			50,0		Remov
External Memory Interfaces	Use	Conn	Name		Description	Clock	Base		End	IRQ	Tags	
<ul> <li>Flash Interfaces</li> <li> <ul> <li>EPCS/EPC0x1 S</li> </ul> </li> </ul>			onchip_mem	ory	On-Chip Memory (RAM or ROM)	[clk1]						
EPOSIEPOUXTS		$\overline{}$	s1		Avalon Memory Mapped Slave	clk_0	= 0x4000		0x67ff			
	<b>V</b>		🗆 niosii		Nios II Processor	[clk]						
			instruction_m	ister	Avalon Memory Mapped Master	clk_0						
			data_master itag debug m	a dula	Avalon Memory Mapped Master Avalon Memory Mapped Slave	[ck]		IRQ C	0x8fff	IRQ 31		
	V		⊟ itag uart 0	oquie	JTAG UART	[ck]	= 0x8800		OXOLLI			
			avalon itag s	lave	Avalon Memory Mapped Slave	clk 0	0x9020		0x9027	ā		
			E sysid		System D Peripheral	[ck]				Ĭ		
		ШÞ	control_slave		Avalon Memory Mapped Slave	clk_0	<b>⊯</b> 0x9028		0x902f			
	<b>V</b>		□ pwm_gen_0		pwm_gen_v1	0						
			avalon_slave		Avalon Memory Mapped Slave	clk_0	n 0x9000		0x901f			
	V	Ш.			EPCS Serial Flash Controller	[clk]						
			epcs_control_	port	Avaion Memory Mapped Slave	clk_0	<b>₫</b> 0x0800		0x0fff	F-1		
ew Edt 🜵 Add	×	Remove	Edit	<b>Z</b>	Address I	1ap	<sup>9</sup> Filters Filte	r: Default				
farning: niosii: Custom Instruction of	ompone	nts can be	edited through the C	omponent	Editor.							
(arning: niosii: Disabling the assign	CPUID o	ontrol regi	ster value manually w	il no long	er auto-assigns unique control registe	or value. This op	tion will always be	turned on with de	efault value set to (	).		
(arning: pwm_gen: TOP_LEVEL_H	DL_MOD	ULE auton	natically set to pwm_	gen								
fo: onchip_memory: Memory will	be initiali	ized from (	onchip_memory.hex									

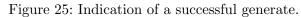
Figure 24: Regenerate the design.

```
4 set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to EPCS_DCLK
5 set_location_assignment PIN_D2 -to EPCS_NCSO
6 set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to EPCS_NCSO
7 set_location_assignment PIN_C1 -to EPCS_ASDO
8 set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to EPCS_ASDO
```

```
Listing 3: Pin Assignment
```

ПΠ





• •
•
•
only 🔒
only 🗎
_
er 9-bit 🔺
-
+
•
*
• 4

Figure 26: Pin options.

General	Dual-Purpose Pins					
Configuration Programming Files Unused Pins Dual-Purpose Pins Capacitive Loading Board Trace Model	Specify how dual-purpose pins should be used after device configuration is complete. The default settings for each pin depend on the current configuration scheme selected in the Configuration tak which is: Active Serial Note: For HardCopy, these settings apply to the FPGA prototype device.					
I/O Timing Voltage	Name	Value				
Pin Placement	DCLK	Use as programming pin				
Error Detection CRC	Data[0]	As input tri-stated				
CvP Settings Partial Reconfiguration	Data[1]/ASDO	As input tri-stated				
	Data[72]	Use as regular I/O				
	FLASH_nCE/nCSO	As input tri-stated				
	Other Active Parallel pins	Use as regular I/O				
	nCEO	Use as programming pin				
		Use as programming pin				
	Description:	Use as regular I/O				
		in should be used when the device is operating in user mode after . The nCEO pin can be reserved as dedicated nCEO programming pin or				
		Reset				

Figure 27: Pin options.

Quartus II 32-bit - C:/tum/SYSU/Practical/LED		· · · · · ·					- 6
jle Edit View Project Assignments Processing	Toole						Search altera.com
🗋 🥁 😹 🕺 🕹 🛍 🗳 🖓 🖓 pwm_g	en	- 😥 💝 🖌 🗑	💿 🕨 🤣 🖄 🖄	🖇 🤹 🛸	👗 🥹 🖤		
roject Navigator 부용×	13	pwm_top.bdf	🖸 🔶		pwm_gen.qsf		
ntity		អ & ក់ គេ គេ ៣ ៣ ៣ ៤	🗠 🐹 🔛 🖉 🕷	🖃 🗆 🖬	5		
Cyclone IV E: EP4CE22F17C6	125	set global assignment -na				C LOWORD-41255" - rection	id auto gignaltan 0
🛛 📰 pwm_top 🐴	126	set global assignment - na					
፼ sld_hub:auto_hub	127	set global assignment - na					
Id_signaltap:auto_signaltap_0	128	set location assignment E		DIDIC_ROOTON	and one on the	burnerout section_id	adco_bignatcap_o
> × pwm:inst	129	set global assignment -na		rin			
> K dock_pll:inst1	130	set global assignment -na			OFF		
	131	set global assignment - na			011		
	132	set global assignment -na			TED CONFIGURATION	"USE AS PECULAR TO"	
	133	set global assignment -na					
	134	set global assignment -na					
	135	set global assignment - na					
	136	set global assignment - na					
	137	set global assignment - na					
	138	set global assignment - na					
III •	139	set global assignment -na					
📤 Hierarchy 📄 Files 🥜 Design Units 4 🕨	140	set global assignment - na					
	141	set instance assignment -					
asks ₽₽×	142	bee_instance_assignment	induce interiore	Diditioni 100	o_pareirein co (	becoment rop	
w: Compilation   Customize	143						
j compilation	144	# -EPCS					
Task	145						
▲ ► Compile Design	146	set location assignment B	TN H2 -to EPCS D	OATAO			
Analysis & Synthesis	147	set instance assignment			L" to EPCS DATA0		
Fitter (Place & Route)	148	set location assignment F					
<ul> <li>Assembler (Generate programming file)</li> </ul>	149	set instance assignment			L" TO -EPCS DOLK		
FineQuest Timing Analysis	150	set location assignment B					
EDA Netlist Writer	151	set instance assignment			L" to EPCS NCSO		
Program Device (Open Programmer)	152	set location assignment B					
<ul> <li>Tregram berice (open tregrammer)</li> </ul>	153	set instance assignment -			L" to EPCS ASDO		
	154		_		_		
	•	III					
All 🗳 🛆 🖓 😹 🔻 < <search>&gt;</search>		~					
Type ID Message							
		1					
					nitial value expre	ession for variable "arg_	_length" because variable was
		gen_0" for hierarchy "pwm:i					
		gen" for hierarchy "pwm:ins					
		d_control_slave_arbitrator"					
📗 🛕 10541 VHDL Signal Declara	ation w	warning at pwm.whd(2025): u	sed implicit defa	ult value fo	or signal "niosii	data_master_read_data va	alid_sysid_control_slave" beca
🛭 🔉 🛕 12125 Using design file :	sysid.	whd, which is not specified	as a design file	for the cu	rent project, but	contains definitions for	or 2 design units and 1 entiti
g ⊳ 🛕 12125 Using design file :	-				1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1		-

Figure 28: Pin options.



		<u>T</u> ools <u>W</u> indow							Search altera.com	
	Report		₽ <b>₽</b> ×			Top View -	Wire Bond			
	Report not available				Cv	clone IV F - I	EP4CE22F17	C6		
1					-,			••		
					• 1	2 3 4 5 5 7 8	9 10 11 12 13 14 15	16		
J					A 🖉			A A		
Ī	Tasks		₽ <i>₽</i> ×		B (0					
Ì	Run Analysis and	d Elaboration								
	4 🚞 Early Pin Plannin				E 🚺		00007000	<u> </u>		
	Early Pin Plan		-							
	_ ,		*		HG	ORAN	VVXXXXX	Z H		
ŀ	•		4		J (0			E 1		
	Groups		₽ <i>₽</i> ×		LG		XXXXXXXX			
	Named: *		<b>•</b>		CELINC M			M Coracian		
	Node Name	Direction	Locati		N Q			O P		
		Direction	LUCau							
					R Q					
	< <new group="">&gt;</new>				T A					
	< <new group="">&gt;</new>						9 10 11 12 13 14 15			
	< <new group="">&gt;</new>		4							
	< <new group="">&gt;</new>	🗞 Edit: 🗶 🗸	4				9 10 11 12 13 14 15		Filter: Pins: all	
	< <new group="">&gt;</new>	🗱 Edit: 🗶 🛷 Direction	Location	I/O Bank			9 10 11 12 13 14 15		Filter: Pins: all	. 5
	< <new group="">&gt;</new>		Location PIN_R8	3	T A	Fitter Location	9 10 11 12 13 14 15 	16		
	< <new group="">&gt;</new>	Direction Input Output	Location PIN_R8 PIN_A2	3	VREF Group B3_N0 B8_N0	Fitter Location	9 10 11 12 13 14 15 	16	Durrent Strength 8mA (default) 8mA (default)	2 (d
	< <new group="">&gt;</new>	Direction Input Output Output	Location PIN_R8 PIN_A2 PIN_C1	3 8 1	T 10 1 VREF Group B3_N0 B8_N0 B1_N0	Fitter Location	IVO Standard     I.5 V (default)	16	Current Strength 8mA (default) 8mA (default) 8mA (default)	2 (d
	< <new group="">&gt;      III     Named: *     Named: *     Node Name     CLS Som     EN_A     Sec SADO     EPCS_DATA0</new>	Direction Input Output Output Unknown	Location PIN_R8 PIN_A2 PIN_C1 PIN_H2	3 8 1 1	T 10 1 VREF Group B3_N0 B3_N0 B1_N0 B1_N0	Fitter Location	I/O Standard I/O Standard 2.5 V (default) 3.3-V LVTTL	16	Current Strength 8mA (default) 8mA (default) 8mA (default) 8mA (default)	2 (d
	< <new group="">&gt;</new>	Direction Input Output Output	Location PIN_R8 PIN_R2 PIN_C1 PIN_C1 PIN_H2 PIN_H1	3 8 1	T 0 VREF Group B3_N0 B8_N0 B1_N0 B1_N0 B1_N0	Fitter Location	IVO Standard     I.5 V (default)	16	Current Strength 8mA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default)	2 (0
	< <new group="">&gt; Named: * Node Name C.C.S.SOM Sec.S.SOC PCS_DATA0 PCS_DATA0 PCS_DATA0 PCS_DATA0 PCS_DATA0 PCS_DCLK PCS_DCLK</new>	Direction Input Output Output Unknown	PIN_R8 PIN_A2 PIN_C1 PIN_H2 PIN_H1 PIN_H1 PIN_D2	3 8 1 1 1 1 1	T 10 1 VREF Group B3_N0 B3_N0 B1_N0 B1_N0	Fitter Location PIN_R8 PIN_A2 PIN_C1	I/O Standard 2.5 V (default) 3.3 V LVTTL 3.3 V LVTTL 3.3 V LVTTL 3.3 V LVTTL	16	Current Strength SmA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default)	2 (0
	< <new group="">&gt;</new>	Direction Input Output Output Unknown Unknown	Location PIN_R8 PIN_R2 PIN_C1 PIN_C1 PIN_H2 PIN_H1	3 8 1 1 1 1	T 0 VREF Group B3_N0 B8_N0 B1_N0 B1_N0 B1_N0	Fitter Location	I/O Standard 2.5 V (default) 3.3 V LVTTL 3.3 V LVTTL 3.3 V LVTTL 3.3 V LVTTL 2.5 V (default)	16	Current Strength 8mA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default)	2 (d
	< <new group="">&gt; Named: * Named: * Node Name Node Name CL, SOM CM, EPCS ASDO EPCS_DATA0 EPCS_DCLK EPCS_NCSO Reset M. attera_reserved_tcl</new>	Direction Input Output Output Unknown Unknown Unknown Input k Input	PIN_R8 PIN_A2 PIN_C1 PIN_H2 PIN_H1 PIN_H1 PIN_D2	3 8 1 1 1 1 1	T 0 VREF Group B3_N0 B1_N0 B1_N0 B1_N0 B1_N0 B1_N0	Fitter Location PIN_R8 PIN_A2 PIN_C1	I/O Standard 2.5 V (default) 3.3 V LVTTL 3.3 V LVTTL 3.3 V LVTTL 3.3 V LVTTL	16	Current Strength SmA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default)	2 (d
	< <new group="">&gt; Mamed: * Named: * Node Name CLK_50M EPCS_ASDO EPCS_DATA0 EPCS_DCLK EPCS_NCSO Reset</new>	Direction Input Output Output Unknown Unknown Unknown Input k Input	PIN_R8 PIN_A2 PIN_C1 PIN_H2 PIN_H1 PIN_H1 PIN_D2	3 8 1 1 1 1 1	T 0 VREF Group B3_N0 B1_N0 B1_N0 B1_N0 B1_N0 B1_N0	Fitter Location PIN_R8 PIN_A2 PIN_15	I/O Standard 2.5 V (default) 3.3 V LVTTL 3.3 V LVTTL 3.3 V LVTTL 3.3 V LVTTL 2.5 V (default)	16	Current Strength 8mA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default)	2 (d
	< <new group="">&gt; Named: * Named: * Node Name Node Name CL, SOM CM, EPCS ASDO EPCS_DATA0 EPCS_DCLK EPCS_NCSO Reset M. attera_reserved_tcl</new>	Direction Input Output Output Unknown Unknown Unknown Input k Input i Input	PIN_R8 PIN_A2 PIN_C1 PIN_H2 PIN_H1 PIN_H1 PIN_D2	3 8 1 1 1 1 1	T 0 VREF Group B3_N0 B1_N0 B1_N0 B1_N0 B1_N0 B1_N0	Fitter Location PIN_R8 PIN_C1 PIN_J15 PIN_H3	I/O Standard 2.5 V (default) 3.3 V LVTTL 3.3 V LVTTL 3.3 V LVTTL 2.5 V (default) 2.5 V (default) 2.5 V (default) 2.5 V (default) 2.5 V (default) 2.5 V (default) 2.5 V (default)	16	Current Strength 8mA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default)	2 (d 2 (d
	< <new group="">&gt;      III     Named: *     Node Name     CL_SOM     CL_SOM     EPCS_ASDO     EPCS_DATA0     EPCS_DATA0     EPCS_DATA0     EPCS_DATA     EPCS_DATA     EPCS_DATA     Atera_reserved_tcl     altera_reserved_tcl     altera_reserved_tcl</new>	Direction Input Output Output Unknown Unknown Input i Input o Output Output Unknown Output Ou	PIN_R8 PIN_A2 PIN_C1 PIN_H2 PIN_H1 PIN_H1 PIN_D2	3 8 1 1 1 1 1	T 0 VREF Group B3_N0 B1_N0 B1_N0 B1_N0 B1_N0 B1_N0	Fitter Location PIN_R8 PIN_A2 PIN_C1 PIN_13 PIN_H3 PIN_H3	1/0 Standard 2.5 V (default) 3.3-V LVTTL 3.3-V LVTTL 3.3-V LVTTL 2.5 V (default) 2.5 V (default) 2.5 V (default) 2.5 V (default) 2.5 V (default)	16	Surrent Strength 8mA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default)	2 (d 2 (d
	< <new group="">&gt; Named: * Named: * Node Name CLX_SOM EPCS_ASDO EPCS_DATA0 EPCS_ASDO EPCS_DCLK EPCS_NCSO Reset altera_reserved_td 3d altera_reserved_td 3d altera_reserved_td</new>	Direction Input Output Output Unknown Unknown Input i Input o Output Output Unknown Output Ou	PIN_R8 PIN_A2 PIN_C1 PIN_H2 PIN_H1 PIN_H1 PIN_D2	3 8 1 1 1 1 1	T 0 VREF Group B3_N0 B1_N0 B1_N0 B1_N0 B1_N0 B1_N0	PIN_H3           PIN_H3           PIN_H3           PIN_H4	I/O Standard 2.5 V (default) 2.5 V (default) 3.3 V LVTTL 3.3 V LVTTL 3.3 V LVTTL 2.5 V (default) 2.5 V (default) 2.5 V (default) 2.5 V (default) 2.5 V (default) 2.5 V (default)	16	Current Strength 8mA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default) 8mA (default)	2 (d 2 (d
	< <new group="">&gt; III Named: * Node Name CLX 50M EPCS_DATA0 EPCS_DATA0 EPCS_DATA0 EPCS_DATA0 EPCS_DATA0 EPCS_DATA0 EPCS_DATA0 EPCS_DATA0 EPCS_DATA0 EPCS_DATA0 EPCS_DATA0 EPCS_DATA0 EPCS_DATA0 EPCS_DATA0 EPCS_DATA0 EPCS_DATA0 EPCS_DATA0 EPCS_DATA0 EPCS_DATA0 EPCS_DCLK EPCS</new>	Direction Input Output Output Unknown Unknown Input k Input i Input o Output is Input is Input is Input	Location PIN_R8 PIN_A2 PIN_C1 PIN_H2 PIN_H1 PIN_D2 PIN_J15 PIN_B12	3 8 1 1 1 1 5 7	T (Å T VREF Group B3_N0 B1_N0 B1_N0 B1_N0 B1_N0 B1_N0 B1_N0 B1_N0 B1_N0 B1_N0 B1_N0 B1_N0 B1_N0	PIN_115           PIN_115           PIN_15           PIN_15           PIN_15           PIN_15           PIN_15           PIN_15	I/O Standard 2.5 V (default) 2.5 V (default) 3.3 V LVTTL 3.3 V LVTTL 3.3 V LVTTL 2.5 V (default) 2.5 V (default)	16	Current Strength SmA (default) SmA (default) SmA (default) SmA (default) SmA (default) SmA (default) SmA (default) SmA (default) SmA (default) SmA (default)	2 (d 2 (d 2 (d 2 (d
	< <new group="">&gt; Named: * Named: * Node Name CLK_50M SEPCS_ASDO EPCS_ASDO EPCS_ASDO EPCS_NCSO Reset altera_reserved_td altera_reserved_td altera_reserved_td altera_reserved_td altera_reserved_td served_reserved_td served_reserved_td altera_reserved_td served_reserved_td altera_reserved_td served_reserved_reser</new>	Direction Input Output Unknown Unknown Input Input Input Output Unknown Input Input Input Output Input Output Input Output	Location PIN_R8 PIN_A2 PIN_C1 PIN_H2 PIN_H2 PIN_H1 PIN_D2 PIN_J15	3 8 1 1 1 1 5	T (A T T BB_NO BB_NO BL_NO B1_NO B1_NO B1_NO B1_NO B1_NO	Fitter Location PIN_R8 PIN_A2 PIN_C1 PIN_H3 PIN_H4 PIN_J4 PIN_J5	I/O Standard 2.5 V (default) 3.3 V LVTTL 3.3 V LVTTL 3.3 V LVTTL 2.5 V (default) 2.5 V (default) 2.5 V (default) 2.5 V (default) 2.5 V (default) 2.5 V (default) 2.5 V (default)	16	Current Strength SmA (default) SmA (default)	2 (d 2 (d 2 (d 2 (d 2 (d 2 (d 2 (d 2 (d

Figure 29: A view of the overall pin assignment of the design.



## A.3.2. Writing the EPCS flash

😂 Nios	▲		_ • • ×
<u>F</u> ile <u>E</u>	Paste	Ctrl+V	
1 📑 🖵 🏅	Celete	Delete	📁 🛷 👻 🗄 👻 🖓 🖛 🏷 🗣 🗇 🔹 📑 🏶 Nios II De 🚺
	Remove from Context	Ctrl+Alt+Shift+Down	
🕒 Proj	Source	•	
	Move		
	Rename	F2	
	Import		
	Export		
<u></u>	Build Project		
î 1	Clean Project		
💼 s 🍭	Refresh	F5	
<u></u>	Close Project		
i≣ €	Close Unrelated Projects		
	Build Configurations	•	
	Make Targets	•	
P 95	Index	•	
	Show in Remote Systems view		
Þ	Convert To		Console 🛛 🔲 Properties 🛛 🤚 🖓 🛐 💷 🚮 🐂 🗹 🕊 🗂 🗝 🗂
Þ	Run As	•	/stem name DE0 Nano SUPCsopcinto
5	Debug As	•	/lectures/Practikum/7.1-Accelerometer/hardware/softwar
	Profile As	•	Nano SOPC.sopcinfo
	Team	+	KBytes program size (code + initialized data).
	Compare With	+	589 KBytes free for stack + heap.
Þ 😂 7	Restore from Local History		.objdump
💼 i	Nios II	•	BSP Editor
	Run C/C++ Code Analysis		Nios II Command Shell
î >	Update Linked Resources		Flash Programmer
	Properties	Alt+Enter	
	-		

Figure 30: Start the EPCS flash programmer from the NIOS II Eclipse IDE.

🛓 Nios II Flash Progr	ammer	- • ×
File Options Tools Help		
New		
Open	ne:	
Save	ie:	
Save As	th:	
Exit Strg+Q	n:	Connections
Nios II Flash Program	mer	
This tool is the GUI version	n of the command line tool: nios2-flash-programmer-generate.	
It parses a SOPC Informat	ion File or Board Support Package (BSP) Settings File, and dynamically creates a tab for each flash device found	
Each Flash tab allows mu	tiple files to be added to a list for eventual conversion to flash and programming to its target flash device.	
	tem from the top level menu bar to create a new flash settings project, Lusing the File->Open menu item.	
Information Problems Pro	ressing	
	S	tart Exit

Figure 31: First set a new download.



🍰 New Flash Programmer	Settings File	×
Get flash programmer system det	-	
Get flash programmer system det	ails from SOPC Information File	
BSP Settings File name:		]
SOPC Information File name:	$\label{eq:linear} \verb C:\tum\tu9\svn-TUM\ectures\Practikum\7-Volatile\Hardware\cpu0.sopcinformation and the second second$	
Master CPU name:	cpu_0 💌	
Flash memory:		
	OK Can	:el

Figure 32: Start the EPCS flash programmer from the NIOS II Eclipse IDE.

💩 Nios II Flash Programmer			
File Options Tools Help			
Target hardware information			
BSP Settings File name:			
SOPC Information File name: C:\tum\tu9\svn-TUM\ectures\Practikum\7.1-Accelerom	eter\hardware\DE0_Nano_SOPC.	sopcinfo	
CPU to program flash: cpu			
Hardware connection:			Connections
Flash: epcs			
Base address:         0x5000000         Memory span:         0x800           Master CPU:         cpu         .zip file system offset in BSP;			J.
Files for flash conversion:			
File Name	Conversion Type	Flash Offset	Add
File generation command:  File programming command:  File programming command:  File for the Decklored Dec			Properties  Properties
Information Problems Processing			
No Nios II target connection paths were located. Check connections and that a Nios II No files have been added for conversion to target flash files.	.sof is downloaded.		
			Start Exit

Figure 33: Select the hardware and software images to download.



🏝 Nios II Flash Programmer						
File Options Tools Help						
Target hardware information						
BSP Settings File name:						
SOPC Information File name: C:\tum\tu9\svn-TUM\ectures\Practikum\7.1-Accelerometer\hardware\PE0_Nano_SOPC.sopcinfo						
CPU to program flash: cpu						
Hardware connection: USB-Blaster on localhost [USB-0] Device: EP3C	Connection: USB-Blaster on localhost [USB-0] Device: EP3C25[EP4CE22@1 Device ID: 1 CPU Instance ID: 0 CPU Connections					
Flash: epcs						
Base address: 0x5000000 Memory span: 0x800						
Master CPU: cpu .zip file system offset in BSP:						
Files for flash conversion:						
File Name	Conversion Type	Flash Offset	Add			
$\verb C:\tum\tu9\svn-TUM\lectures\Practikum\7.1-Accelerometer\hardwar$	SOF	0	Remove			
C:\tum\tu9\svn-TUM\lectures\Fractikum\7.1-Accelerometer\hardwar ELF <no offset=""></no>						
File generation command:						
sof2flashinput="C:/tum/tu9/svn-TUM/lectures/Practikum/7.1-Accele			Properties			
output="C:/tum/tu9/svn-TUM/lectures/Practikum/7.1-Accelerometer/hardware/flash/DE0_Nano_epcs.flash"						
· · · · · · · · · · · · · · · · · · ·						
File programming command:						
nio32-flash-programmer						
"C:/tum/tu9/svn-TUM/lectures/Practikum/7.1-Accelerometer/hardware/flash/DEO_Nano_epcs.flash"						
Information Problems Processing						
A Obtaining system ID information for selected processor						
Connected system ID hash not found on target at expected base address. Connected system timestamp not found on target at expected base address.						
Connected system unescaling not round on target at expected uses address.     The expected CPU name does not match the selected target CPU name.						
		Star	t Exit			

Figure 34: The GUI after selecting the images.