# Building an Embedded Processor System on Xilinx NEXYS3 FPGA and Profiling an Application: A Tutorial

## **Introduction:**

Modern FPGA's are equipped with a lot of resources that allow them to hold large digital systems on a single chip. FPGA vendors provide tools that allow the designer to build embedded systems on efficiently on FPGAs. One of the new concepts that you will learn in this experiments is the system-on-chip approach (SoC). In this methodology, a complete micro-processor system is implemented on a single chip. In our experiment we will use an FPGA board that is equipped with an FPGA from Xilinx a leading FPGA vendor. Xilinx provides a tool for building an embedded SoC on its FPGAs, and we are going to use this tool in this experiment. This tool is called Xilinx Embedded Development Kit (EDK). The EDK allows the designer to build the processor system based on an embedded processor from Xilinx called MicroBlaze. The tool provides a C/C++ compiler for that processor and an IDE based on Eclipse framework. In this tutorial we will build a processor system based on MicroBlaze using the EDK and run this system on Nexys 3 FPGA board.

# **Objectives:**

- 1. Demonstrate the concept of SoC.
- 2. Familiar students with embedded soft processor systems on FPGA.
- 3. Build a soft processor system that will perform simple operations.
- 4. Profile the Application.

### **Equipment and Tools**

- 1. Nexys 3 FPGA board, use two USB cables:
  - a. Connect the first to the USB PROG port.
  - b. Connect the second to the **UART port**.
- 2. Xilinx Embedded Development Kit (XPS, SDK). This is tested on revision 13.4 successfully.

# **Detailed Steps**

# **Part 1:** Building the hardware system on FPGA

In previous labs you learned about the Xilinx ISE design flow and how to use it to map a design using VHDL on FPGA. In this tutorial we will use the Xilinx EDK to build a micro-processor system and write a simple program for that processor to perform simple I/O operations. The first part of

the tutorial attempts to build the hardware system. The EDK is composed of two software components: i) Xilinx Platform Studio (XPS) which is used to build and configure the soft processor system on the FPGA. ii) Xilinx Software Development Kit (SDK), which is the IDE for software development. Perform the following steps to build the hardware system.

**Step 1:** We start by building the hardware using the XPS. Start XPS using the program menu as shown in Figure 1.



#### Figure 1. Starting EDK (Step 1)

When the XPS starts the main window will show the options shown in **Figure 2**. The "Getting Started" part gives you the options to create a new project (empty or based on a specific board), or open an old project that was created before.

**Step 2:** Select the option "Create New Project Base System Builder" which will allow you to build an embedded processor system based on the specification of specific board (In our case Nexys 3 board). The "New Project Wizard" will appear as shown in **Figure 3**.



Figure 2. XPS Window (Step 2)

New Project	
Project File	C:\Projects\EDK_Projects\mydesign\system.xmp Browse
Select an In	terconnect Type
🔘 AXI	System
AX fut xilir	is an interface standard recently adopted by Xilinx as the standard interface used for all current and ure versions of Xilinx IP and tool flows. Details on AXI can be found in the AXI Reference Guide on IX.com.
PLB	System
PLE Vir1 mig do	is the legacy bus standard used by Xilinx that supports current FPGA families, including Spartan6 and ex6. PLB IP will not support newer FPGA families, so is not recommend for new designs that may rate to future FPGA families. Details on PLB can be found in the PLBv46 Interface Simplifications cument on xilinx.com.
Select Existi	ng .bsb Settings File(saved from previous session)
Select Existi	ng .bsb Settings File(saved from previous session) Browse
Select Existi	ng .bsb Settings File(saved from previous session) Browse Peripheral Repository Search Path
Select Existi	ng .bsb Settings File(saved from previous session)  Peripheral Repository Search Path Browse Browse

Figure 3. Building a New Design (Step 3)

**Step 3:** In the new project wizard window of **Figure 3**, select the location you want to save the files of your project. The project will include so many files in the future so try to keep this location specific and separate from other files so that you can refer back to it easily. Create a new folder on the hard drive and name it "Projects" to store all your projects, then create another folder called "EDK\_Projects" to store all projects created using EDK. After that create a folder for this new project and name it for example "mydesign". The project file (usually system.xmp) will be stored in that location (c:\Projects\EDK\_Projects\mydesign\system.xmp) as shown in **Figure 3**.

The second option in this window is to select the interconnect type. The EDK supports two types of bus interface AXI, and PLB. Both are standard bus topology with various specifications. The differences between the two standards are beyond the scope of this article. For this experiment we will use the second option "PLB System".

When done press "OK". This will start the "Base System Builder" tool that will help us build and configure the hardware system as shown in **Figure 4**.



Figure 4. Base System Builder - Welcome Screen (Step 4)

**Step 4:** The welcome screen shown in **Figure 4**, is the first screen in the Base System Builder wizard that will guide you to build your system. In this screen select **"I would like to create a new design"** and then press next. This will let you create a new design.

**Step 5:** The second screen of the wizard is shown in **Figure 5**. In this screen we select the board that we want to use to build the system. The XPS allow you to build a design based on a specific board or create a custom design that is generic. Select the first option **"I would like to create a system for the following development board"**. This will allow you to select a specific board. The board is specified by three options:

- Board Vendor: this is the manufacturer of the board. Select **"Digilnet"** which is the company that builds the Nexys 3 board.
- Board Name: this is the name of the board we want to use. Select "Nexys 3 Board".
- Board Version: the board version. Select version "B".

Welcon	ne Bo	oard Sy	vstem	Processor	Peripheral	Cache	Summary
Board Selection Select a target deve	elopment board.						
Board							
I would like to c	reate a system for	the following develop	oment board				
Board Vendor	Digilent						
Board Name	Nexys 3 Board						
Board Revision	В						
) I would like to a	reate a system for	a custom board					
loard Information							
rchitecture		Device		Package	Spee	d Grade	
partan6	~	xc6slx16	-	csg324	-3		
Use Stepping			a d'enned	- History			~
anat Delavity Acto	us Lileb						
eservolancy Ace	ne raight						
lated Information							
endor's Contact Inf	formation	load Website					
he Nexys3 board fr 28Mbit Numonyx P D/100Mbps speed, LEDs	eatures a Xilinx Spa arallel PCM Flash m HID Host for USB r	artan-6 XC6SLX16-3C emory, 128Mbit Num nouse, keyboard, US	SG324C FPGA dev onyx N25Q 128(x B-UART bridge for	vice, 512Mb or 256 4) SPI PCM Flash m the serial port, 8	Mb (x16) CDRAM compone lemory, SMSC LAN8710 PH slide switches, 5 pushbutto	nt for use with Spa / with MII interface ns in Gamepad con	artan-6, t for use with figuration and
More Info					< Back	Next >	Cancel

Figure 5. Board Selection (Step 5)

The Nexys 3 board is equipped with Spartan 6 FPGA chip which is a mid-size FPGA from Xilinx. The board has several peripherals that can be used with the FPGA to perform several functions (See **Figure 5**):

- 512 MB CDRAM.
- 128 Mbit Flash Memory.
- 8 Switches and 8 LEDS.
- 5 Push Buttons.
- Ethernet Physical Interface for base 10/100 networking.
- USB Host for Keyboard and Mouse
- Serial Port

These peripherals allow you to build a small computer system on the board which we are going to do in this experiment (and the following ones).

When done selecting, press "Next" to move to the next screen.

**Step 6:** The following step is to select the system type. This is shown in **Figure 6.** The XPS allows us to choose between two types of system architecture:

- Single Processor System: This is the common system that you studied in many courses. The
  system is composed on one processor connected to several peripherals using a single common
  bus. The advantage of such system is the simplicity of the design and simplicity of
  programming such systems. The main disadvantage is that all peripherals are connected
  through a common bus, which will reduce the speed for fast peripherals to match the slower
  ones. This type of systems is good for applications that do not require high speed of data
  communication.
- Dual Processor System: This system architecture is composed of two processors; each is connected to a separate bus. One processor will be used for high speed peripherals and the other is used with lower speed peripherals. This increases the performance of the system compared to the single processor system. However, writing software for this system is more complex as it requires synchronization between the two processors.

Select **"Single Processor System"**. This will start building a single processor system. We choose to build a single processor system as this is simple to develop. However, you can later add more processors if required.



Figure 6. Select System Architecture (Step 6)

ờ Base System Builder						?×
Welcome	Board	System	Processor	Peripheral	Cache	Summary
Processor Configuration Configure the processor(s).	1					
Configure the processor(s). Reference Clock Frequency Processor 1 Configuration Processor Type System Clock Frequency Local Memory Debug Interface IT Enable Floating Point	/ 100,00 MicroBlaze 66.67 [64 K3 [On-Chip HW Debug] Unit	Module				MHz MHz MHz
More Info				< Back	Next >	Cancel

When done press "Next" to move to the next screen shown in Figure 7.

Figure 7. Processor Configuration (Step 7)

**Step 7:** The processor configuration screen is shown in **Figure 7.** As stated earlier, Xilinx provides a processor called MicroBlaze that can be implemented on its FPGAs. In this screen we configure the processor as follows:

- Processor Type: There is only one selection which is MicroBlaze.
- System Clock Frequency: This is the processor and common bus reference clock frequency. Most of the modern FPGA's are equipped with clock modules that are able of generating higher frequencies from a single fixed frequency. For example, Nexys 3 board has on board clock of 50 MHz, however, you can use higher frequencies using the FPGA clock modules. Select the desired frequency up to 83 MHz Select **"66.67 MHz".**
- Local Memory: The FPGA contains several memory blocks up to 128 KB. This is different from the on board memories. You can connect up to 64KB of these block memories to the MicroBlaze as a local processor memory. You can connect more memory if required through the local bus. Select **"64KB"** for this tutorial.
- Leave all other options and then press "Next" to move to the next screen.

ripheral Configuration add a peripheral, drag it from vailable Peripherals	n the "Available Pe	eripherals" to the pro	ocessor peripheral list. To	change a core parameter,	click on the periphe	ral.
Peripheral Names			Processor 1 (MicroBla:	ze) Peripherals		Select All
<ul> <li>Devices</li> <li> ETHERNET</li> <li>Internal Peripherals</li> <li>Imb_bram_if_cntlr</li> <li>- xps_bram_if_cntlr</li> <li>- xps_timebase_wdt</li> <li>- xps_timer</li> </ul>		Add > < Remove	Core DIP_Switches_8Bits Core Use Interrupt Ethernet_Lite Core: xps_ethei LEDs_8Bits Core: xps_gnic Micron_RAM Core: xps_mch, PS2_Mouse_Keybo Core: xps_mch PS2_Mouse_Keybo Core: xps_gnic RS222_Uat_1 Core: xps_gnic RS22_Uat_1 Core: mb_brar	netlite _emc ard s te, Baud Rate: 9600, Da n_if_cntIr n_if_cntIr	Parameter xps_gpio	

Figure 8. Peripheral Configuration (Step 8 - 1)

**Step 8:** After configuring the processor we are ready to configure the peripherals (Input/Output devices). The peripheral configuration screen is shown in Figure 8. In this screen a list of all the available peripherals is presented. From this list the designer can choose which peripheral to be connected to the processor. The screen is divided into two lists. The right-hand side list shows the peripherals that are not connected to the processor, while the left-hand side list shows all the peripherals that are currently connected to the processor. By default several peripherals will be connected to the processor as shown in **Figure 8**. In this project we do not need many of these devices so we will remove many of them as shown in **Figure 9**. Remove the following devices:

- PS2\_Mouse\_Keyboard
- Micron\_RAM
- Numonyx\_RCM
- Ethernetlite

To remove a device, select the device then press "Remove".

This leaves the Switches, Leds, Push Buttons, and Local Memory devices connected to the processor.



Figure 9. Peripheral Configuration (Step 8 - 2)

To add a device to the processor connection, select the required device and click add. We will add the xps\_timer since it is <u>required for profiling</u> as seen in **Figure 10.** Make sure to enable interrupts.

Welcome Board	System	Processor	Peripheral	Cache	Sumr
ripheral Configuration					
add a peripheral, drag it from the "Availab	le Peripherals" to the pro	ocessor peripheral list. To o	change a core parameter	, click on the periph	neral.
railable Peripherals					
eripheral Names		Processor 1 (MicroBlaz	e) Peripherals		Select All
IO Devices     Ethermote Lite		Core		Parameter	
ETHERNET		DIP Switches 8Bits			
PS2_Mouse_Keyboard		Core: xps_gpio			
- Micron_RAM		LEDs_8Bits			
Numonyx_PCM		Push Buttons 4Bits			
Internal Peripherals		Core: xps gpio			
xps_bram_if_cnttr		RS232_Uart_1			
xps_timebase_wdt		Core: xps_uartit	e, Baud Rate: 9600, Dat	a B	
xps_timer		Core: Imb. bram	if onth		
		ilmb cntir	ji_cria		
		Core: Imb_bram_	if_cntlr		
		xps_timer_0			
		Core		xps_timer	
	Add >	Count width		32	
	(A)	Configure Mode		Two timers a	are present
	< Remove	Use Interrupt		I <b>v</b>	
		4			

Figure 10. Adding the xps\_timer (Step 8 - 3)

Now we selected all the required devices. We have some input/output devices connected to the processor. After completing the hardware, we will write several software applications for the sake of profiling.

Press "Next" to move to the next screen.

**Step 9:** Cache configuration screen is shown in **Figure 10**. If we have more than one memory types connected to the processor, we can use cache memory to speed up memory access. As we only have local memory connected to the processor, no cache memory can be configures. We will leave this screen unchanged and press **"Next"**.

Base System Builder						8 ×
Welcome	Board	System	Processor	Peripheral	Cache	Summar
Cache Configuration Select cache size and cache r	nemory for processo	·(s).				
Processor 1 (MicroBlaze) C	ache					
There is no cacheable men	nory for this processo	и				
More Info				< Back	Next >	Cancel
				L		

Figure 10. Cache Configuration (Step 9)

**Step 10:** Now <u>we are done</u> configuring the hardware. The last screen of the "Base System Builder" is shown in Figure 11. This screen displays a summary of the system being built. Two types of information are displayed; system components and file allocation. The components list displays the name of each component and the address associated with it. The address assigned to each peripheral is unique and it is used by the processor to locate a specific device and communicate with it. For example, the device DIP\_Switches\_8Bits (the 8 switches on the board) is assigned the address 0x81440000 which means that the processor will use this address to read the value of the switches as digital input.

	Board	System	Processor	Peripheral	Cache	Summa
Summary						
Below is the summary of th	ne system vou are cri	eating				
below is the building of t	ie o yoteni you ure o.	laongr				
/stem Summary						
Core Name	Instance Name	Base Address	High Address			
Processor 1	microblaze_0					
- xps_gpio	DIP_Switches_8Bits	0x81440000	0x8144FFFF			
- xps_gpio	LEDs_8Bits	0x81420000	0x8142FFFF			
- xps_gpio	Push_Buttons_4Bits	0x81400000	0x8140FFFF			
- xps_uartite	RS232_Uart_1	0x84000000	0x0000EEEE			
Imb_bram_if_cottr	ilmb_cottr	0x00000000	0x0000FFFF			
vos timer	vns timer 0	0x83C00000	0x83C0EEEE			
xpa_and	xps_uner_o	0x0500000	0X05001111			
Lambon						
Location						
- Overall						
Dula0 DereenalFile	a Dian EDCA Draiget	n EDK Bacad Drains	talTistacial disustan umo			
D:\a0-PersonalFile	s-D/z0-FPGA_Project	s EDKBasedProjec	ts\Tutorial-4\system.xmp			
<ul> <li>D: \a0-PersonalFile</li> <li>D: \a0-PersonalFile</li> <li>D: \a0-PersonalFile</li> </ul>	s-D\z0-FPGA_Project s-D\z0-FPGA_Project	s\EDKBasedProjec s\EDKBasedProjec	ts\Tutorial-4\system.xmp ts\Tutorial-4\system.mhs	£		
<ul> <li>D: \a0-PersonalFile</li> <li>D: \a0-PersonalFile</li> <li>D: \a0-PersonalFile</li> <li>D: \a0-PersonalFile</li> </ul>	s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project	s EDKBasedProjec s EDKBasedProjec s EDKBasedProjec	ts\Tutorial-4\system.xmp ts\Tutorial-4\system.mhs ts\Tutorial-4\data\system.uc	f		
<ul> <li>D: \a0-PersonalFile</li> <li>D: \a0-PersonalFile</li> <li>D: \a0-PersonalFile</li> <li>D: \a0-PersonalFile</li> <li>D: \a0-PersonalFile</li> </ul>	s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project	s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec	ts\Tutorial-4\system.xmp ts\Tutorial-4\system.mhs ts\Tutorial-4\data\system.uc ts\Tutorial-4\etc\fast_runtim ts\Tutorial-4\etc\fast_runtim	f e.opt		
<ul> <li>D:\a0-PersonalFile</li> <li>D:\a0-PersonalFile</li> <li>D:\a0-PersonalFile</li> <li>D:\a0-PersonalFile</li> <li>D:\a0-PersonalFile</li> </ul>	s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project	s\EDKBasedProjec s\EDKBasedProjec s\EDKBasedProjec s\EDKBasedProjec s\EDKBasedProjec	ts\Tutorial-4\system.xmp ts\Tutorial-4\system.mhs ts\Tutorial-4\data\system.uc ts\Tutorial-4\data\system.uc ts\Tutorial-4\etc\download.o ts\Tutorial-4\etc\download.ou	f e.opt md		
<ul> <li>D:\a0-PersonalFile</li> <li>D:\a0-PersonalFile</li> <li>D:\a0-PersonalFile</li> <li>D:\a0-PersonalFile</li> <li>D:\a0-PersonalFile</li> <li>D:\a0-PersonalFile</li> <li>D:\a0-PersonalFile</li> </ul>	s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project	s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec	ts\Tutorial-4\system.xmp ts\Tutorial-4\system.mhs ts\Tutorial-4\data\system.uc ts\Tutorial-4\etc\fast_runtim ts\Tutorial-4\etc\download.c ts\Tutorial-4\etc\bitgen.ut	f e.opt md		
D:\a0-PersonalFile     D:\a0-PersonalFile     D:\a0-PersonalFile     D:\a0-PersonalFile     D:\a0-PersonalFile     D:\a0-PersonalFile     D:\a0-PersonalFile	s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project	s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec	ts\Tutorial-4\system.xmp ts\Tutorial-4\system.mhs ts\Tutorial-4\system.uc ts\Tutorial-4\data\system.uc ts\Tutorial-4\etc\fast_runtim ts\Tutorial-4\etc\bitgen.ut	f e.opt md		
D: \a0-PersonalFile D: \a0-PersonalFile D: \a0-PersonalFile D: \a0-PersonalFile D: \a0-PersonalFile D: \a0-PersonalFile	s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project	s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec	ts\Tutorial-4\system.xmp ts\Tutorial-4\system.mhs ts\Tutorial-4\system.uc ts\Tutorial-4\etc\fast_runtim ts\Tutorial-4\etc\download.c ts\Tutorial-4\etc\bitgen.ut	f e.opt md		
D: \a0-PersonalFile	s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project	s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec	ts \Tutorial-4\system.xmp ts \Tutorial-4\system.mhs ts \Tutorial-4\system.uc ts \Tutorial-4\ctafs_system.uc ts \Tutorial-4\ctafs_t_runtim ts \Tutorial-4\ctc\bitgen.ut ts \Tutorial-4\ctc\bitgen.ut	f e.opt md		
<ul> <li>D: \a0-PersonalFile</li> </ul>	s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project s-D\z0-FPGA_Project	s (EDKBasedProjec s (EDKBasedProjec s (EDKBasedProjec s (EDKBasedProjec s (EDKBasedProjec s (EDKBasedProjec	ts \Tutorial-4\system.xmp ts \Tutorial-4\system.mhs ts \Tutorial-4\system.mhs ts \Tutorial-4\etc\fast_runtim ts \Tutorial-4\etc\fast_runtim ts \Tutorial-4\etc\bitgen.ut	f e.opt∶ md		
<ul> <li>D: \a0-PersonalFile</li> </ul>	s-D't0-FPGA_Project s-D't0-FPGA_Project s-D't0-FPGA_Project s-D't0-FPGA_Project s-D't0-FPGA_Project s-D't0-FPGA_Project	s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec	ts\Tutorial-4\system.xmp ts\Tutorial-4\system.nit Is\Tutorial-4\signata \system.uic Is\Tutorial-4\stata \system.uic Is\Tutorial-4\stata \system.uic Is\Tutorial-4\stata \system.uic Is\Tutorial-4\stata \system.uic Is\Tutorial-4\stata \system.uic	f e.opt md		
D: \p0-PersonalFile D: \p0-PersonalFile D: \p0-PersonalFile D: \p0-PersonalFile D: \p0-PersonalFile D: \p0-PersonalFile	s-D'x0-FPGA_Project s-D'x0-FPGA_Project s-D'x0-FPGA_Project s-D'x0-FPGA_Project s-D'x0-FPGA_Project s-D'x0-FPGA_Project	s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec s \EDKBasedProjec	ts/Tutorial-4%ystem.xmp ts/Tutorial-4%ystem.kms IS/Tutorial-4%tata/system.uc ts/Tutorial-4%tct/fast_runtim ts/Tutorial-4%tct/fast_runtim ts/Tutorial-4%tct/solwindac. ts/Tutorial-4%tct/bitgen.ut	f e.opt md		
D: \a0-PersonaFile D: \a0-PersonaFile D: \a0-PersonaFile D: \a0-PersonaFile D: \a0-PersonaFile D: \a0-PersonaFile D: \a0-PersonaFile	s-Dto/FPGA_Project s-Dto/FPGA_Project s-Dto/FPGA_Project s-Dto/FPGA_Project s-Dto/FPGA_Project s-Dto/FPGA_Project	s (EDKBasedProjec s (EDKBasedProjec s (EDKBasedProjec s (EDKBasedProjec s (EDKBasedProjec s (EDKBasedProjec	ts \Tutorial-4\system.xmp ts \Tutorial-4\system.xmp ts \Tutorial-4\stats \system.uc ts \Tutorial-4\stats \system.uc ts \Tutorial-4\stats \system.uc ts \Tutorial-4\stats \system.un ts \Tutorial-4\stats \system.ut	f e.opt md		
D:\u0-PersonalFile D:\u0-PersonalFile D:\u0-PersonalFile D:\u0-PersonalFile D:\u0-PersonalFile D:\u0-PersonalFile	s-Dtx0-FPGA_Project s-Dtx0-FPGA_Project s-Dtx0-FPGA_Project s-Dtx0-FPGA_Project s-Dtx0-FPGA_Project s-Dtx0-FPGA_Project	s (EDKBasedProjec s) EDKBasedProjec EDKBasedProjec s) EDKBasedProjec s) EDKBasedProjec	ts/Tutorial-Hsystem.xmp ts/Tutorial-Hsystem.uts ts/Tutorial-Hsystem.uts ts/Tutorial-Hstc/Hsst.gvrfmt ts/Tutorial-Hstc/Jownload.c ts/Tutorial-Hstc/Johnson.ts/ ts/Tutorial-Hstc/Johnson.ts/	f e.opt md		
- D: \u0-PersonalFile D: \u0-PersonalFile D: \u0-PersonalFile D: \u0-PersonalFile D: \u0-PersonalFile D: \u0-PersonalFile	s-Diz0-FPGA_Project s-Diz0-FPGA_Project s-Diz0-FPGA_Project s-Diz0-FPGA_Project s-Diz0-FPGA_Project s-Diz0-FPGA_Project	s (EDKBasedProjec s) EDKBasedProjec EDKBasedProjec s) EDKBasedProjec s) EDKBasedProjec s) EDKBasedProjec	ts/Tutorial-4kpystem.xmp Is/Tutorial-4kpstem.mis Is/Tutorial-4kpats/pstem.us Is/Tutorial-4kpats/pstem.us Is/Tutorial-4kpt/bast/ustra Is/Tutorial-4kpt/bitgen.ut	f e.opt md		
– D: №0-PersonalFile D: №0-PersonalFile D: №0-PersonalFile D: №0-PersonalFile D: №0-PersonalFile D: №0-PersonalFile	s-D/xD-FPGA_Project S-D/xD-FPGA_Project S-D/xD-FPGA_Project S-D/xD+FPGA_Project S-D/xD+FFGA_Project S-D/xD+FFGA_Project	s (EDKBasedProjec S EDKBasedProjec S EDKBasedProjec S EDKBasedProjec S EDKBasedProjec s (EDKBasedProjec	ts/Tutorial-4kpstem.ntp St/Tutorial-4kpstem.ntp ts/Tutorial-4kpstem.ntp ts/Tutorial-4kpst/set_runatm ts/Tutorial-4kpt/set_runatm ts/Tutorial-4kpt/blogen.ut	f e.opt md		
D: 10-PersonalFile D: 10-PersonalFile D: 10-PersonalFile D: 10-PersonalFile D: 10-PersonalFile D: 10-PersonalFile	s-DVD-FFGA_Project s-DVD-FFGA_Project s-DVD-FFGA_Project s-DVD-FFGA_Project s-DVD-FFGA_Project s-DVD-FFGA_Project	s (EDKBasedProjec S EDKBasedProjec S EDKBasedProjec S EDKBasedProjec S EDKBasedProjec S EDKBasedProjec	Its/Tutorial=44system.xmp Its/Tutorial=44setsem.mts Its/Tutorial=44sets/bastem. Its/Tutorial=44cts/bastem. Its/Tutorial=4fetc/bastem.ut Its/Tutorial=4fetc/bitgen.ut	f e.opt md		
D : Ja 0-PersonalFile     Save Base System Build	s-Dyo-FFGA_Project 5-Dyo-FFGA_Project 5-Dyo-FFGA_Project 5-Dyo-FFGA_Project 5-Dyo-FFGA_Project 5-Dyo-FFGA_Project 5-Dyo-FFGA_Project	s EDKBaseProjec EDKBaseProjec s EDKBaseProjec s EDKBaseProjec s EDKBaseProjec s EDKBaseProjec	Is Türbini-Heystem.Anp El Türbini-Heystem.Mis Is Türbini-Heystem.Mis Is Türbini-Heyte (Host, Fast, Tuntim Is Türbini-Heyte (Hosmland, G Is Türbini-Heyte (Hosmland, G	f. 8.opt nd		
D140 PersonalFile	er ("bsb) Settings File FRGA_Project	s EDKBasedProjec s EDKBasedProjec s EDKBasedProjec s EDKBasedProjec a EDKBasedProjec	Is'Illionial-Hystemhtp Stillionial-Hystem.mts Is'Illionial-Hystem.mts Is'Illionial-Hystes.te Is'Illionial-Hystes.te Is'Illionial-Hystes.te Is'Illionial-Hysteb Is'Illionial-Hysteb Ister.te Is'Illionial-Hysteb.sb	f e.opt md		
Di 40 Personali ile	s=Dyo=PFGA_Project S=Dyo=PFGA_Project S=Dyo=PFGA_Project S=Dyo=PFGA_Project S=Dyo=PFGA_Project S=Dyo=PFGA_Project S=Dyo=PFGA_ProjectSEDK	s (EDKBaseProjec EDKBaseProjec s (EDKBaseProjec s (EDKBaseProjec EDKBaseProjec s (EDKBaseProjec s (EDKBaseProjec	Is'Illuorial-Hsystem.xmp Is'Illuorial-Hsystem.mis Is'Illuorial-Hsystem.mis Is'Illuorial-Hsita Lystem.us Is'Illuorial-Hsita Lynum Is'Illuorial-Hsita Lynum Is'Illuorial-Hsit	f . opt: md		

Figure 11. System Summary (Step 10)

The file list includes all the files that are created by wizard to define the project. The wizard generates six files.

- System.xmp: this is the project file that is used by XPS to open the project for future use.
- System.mhs: this is the hardware description file. This is a text file that describes the hardware components of the system.
- System.ucf: user constraints file. This file defines the relation between the system input/output pins and the actual FPGA pins.
- Fast\_runtime.opt, download.cmd, bitgen.ut: these three files are used build the bit file that is later downloaded to the FPGA to configure it to do the function of the system.

This is the final screen of the wizard; clicking **"Finish"** will end the wizard and generate the required files on the folder specified at the beginning of the wizard. Click **"Finish"** and then check the folder **"C:\Projects\EDK\_Projects\mydesign"** it should look as shown in **Figure 13**.

_xps	05/03/2012 10:05	File folder
🧯 data	27/02/2012 12:57	File folder
etc	28/02/2012 3:00 PM	File folder
implementation	27/02/2012 1:05 AM	File folder
pcores	27/02/2012 12:00	File folder
clock_generator_0	27/02/2012 12:57	Text Document
system.bsb	27/02/2012 1:05 AM	BSB File
system	05/03/2012 10:05	Text Document
] system.make	05/03/2012 10:05	MAKE File
system	27/02/2012 1:05 AM	MHS File
system	27/02/2012 1:05 AM	Xilinx Platform Stu
] system_incl.make	05/03/2012 10:05	MAKE File
XpsGuiSessionLock	05/03/2012 10:05	File

Figure 12. Directory Structure

aug         Image: Constraint of the constraint of t		Name - dmb - dmb - amb - mb_pb - B microblaze_0	Bus Name	IP Type 1 imb_v10 1 imb_v10 1 pb_v46	19 Version 2.00.b 2.00.b 1.05.a	bus interface Hite's □ By Connected □ Dy Connected □ Dy Bus Standard □ Dy LMB	THUS S
Image         Image         Image           ription         IP Version         IP           E DCK Install         Image         Image           Image         Analog         Image           Image         Conture         Image           Image         Conture         Image           Image         Conture         Image		Name dmb mb_pb mb_pb B_microblaze_0	Bus Name	1P Type 1 mb_v10 1 mb_v10 1 pb_v46	19 Version 2.00.b 2.00.b 1.05.a	By Connected     Unconnected     By Bus Standard     VIM6	e man e
ription IP Version IP E EDK Install 3 Analog 3 Bus and 4 Conck, R 5 Communi		dimb dimb dimb dimb		1 imb_v10 1 imb_v10 1 pb_v46	2.00.b 2.00.b 1.05.a	Unconnected By Bus Standard UN6	
<ul> <li>EDK Install</li> <li>Analog</li> <li>Bus and</li> <li>Clock, R</li> <li>Communi</li> <li>Communi</li> </ul>	Ref.	- imb - mb_pib . ⊞ microblaze_0		1 imb_v10	2.00.b 1.05.a	By Bus Standard     VILMB	
Analog     Bus and     Clock, R     Comunit     Communit	1 4 4 4 F F	mb_pib microblaze_0		pb_v46	1.05.a	- VIMB	
Bus and     Clock, R     Communi     Communi		_ I microblaze_0					
<ul> <li>Elock, R</li> <li>Communi</li> <li>Communi</li> </ul>		1771 Joseph Annuary		T microoiaze	8.20.b	PLBV46	
Communi     Communi		- mo_oram		😭 bram_block	1.00.a	H M Xinx Pont Io Pont	
e communa		. ⊕ dimb_cntir		1 imb_bram_if	. 3.00.b	DIVI BOCAN	
A CAM and		_ imb_cntir		🙀 Imb_bram_if	3.00.b	- PIXTL MEDERLIKS3	
E Debug		mdm_0		🙀 mdm	2.00.b	PIXIL MBTRACE2	
+ FPGA Re		Aps_intc_0		🙀 xps_intc	2.01.a	XIL_MEMORY_CHANNEL	
General		DIP_Switche		Tr xps_gpio	2.00.a	🖻 🖻 By Interface Type	
IO Modules		E LEDs_88its		1 xps_gpio	2.00.a	Slaves	
<ol> <li>Interpro</li> </ol>	1 S	Push_Button		🙀 xps_gpio	2.00.a	Masters	
Hemory		E xps_tmer_0		📩 xps_timer	1.02.a	Master Slaves	
E PCI	1	# RS232_Uart_1		1 xps_uartite	1.02.a	Monitors Al Taxaeta	
<ul> <li>Perpher</li> <li>Research</li> </ul>		- dock_genera		tock_gener	4.03.a	V Initiators	
2. Lititu	1	proc_sys_res_		proc_sys_re.	3.00.a	E monte a	
Verification		100 C					
roject Local							
N - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 -							
	1	<u></u>				]	
<u>,                                     </u>	Legend			contraction and a second	the second second		
	Master Slave M	aster/Slave Flarge	at Cinitiator Co	Innected OUncon	nected M Monitor		
ch IP Catalog: Clear	Superreded	ase (paid)	se (eval)	acal Pre Produ	ction %2beta setDevelopment		
Project & IP Catalog	Design Su	mary III	& Graphica	Design View	System Assembly View		
<i>h</i> .							10.0
							10
	i Corva I i Corva I i Do Noduks i Dirgo i	ti Donda Donda Herpy H	Core al.     Core al.	Borea de      Borea de	i Borna I. i Borna I. i D'Abdes i Drobe Borno I. i Borna I. i Drobe Borno I. i Borna I. i Drobe Borna I. i Drobe B	Bergend     B	Conserved     Conserved

When the "Base System Builder" is done the XPS window should look like Figure 13.

Figure 13. Xilinx Platform Studio - System Assembly View

**Step 11:** The XPS window (Figure 13) opens at the **"System Assembly View"** which shows the system components and their interconnection. The system assembly view has three tabs:

- **Bus Interface**: this shows the bus interconnection between components. As you can see there are three main buses:
  - ilmb : instruction local memory bus, used to connect the processor to the code memory. Code memory is the memory that holds the programs code (instructions).
  - o dlmb: data local memory bus, used to connect the processor to the data memory.
     Data memory is the memory that holds the program data.
  - mplb: MicroBlaze peripheral local bus (plb), used to connect the processor to all the other peripherals.
- **Ports**: this tab lists all the ports of each peripheral/processor and displays its connection. In this project we will not need to modify this.

• Addresses: this tab lists the addresses assigned to each peripheral. You can modify these addresses through this tab. But for this project we will not do any address modification.

In the "**Bus Interface**" tab you can modify the system by adding more peripherals/processors, removing peripherals, or changing the configuration of any peripheral.

**Step 12:** The last step to build the hardware is to generate the bit-file that is used to configure the FPGA. XPS generates all the necessary files to do so. Each component of the system is defined in VHDL (or other hardware language) and the tool will compile all these files to build a single design file to be downloaded on the board. This process is composed of several tasks:

- **Synthesis**: Will compile all hardware description language (HDL) files of the system and check for any errors. Synthesis will convert HDL into logical blocks.
- Mapping: Transforms the logical blocks generated from synthesis into FPGA blocks.
- **Placement**: Specifies each component of the system and its location on the FPGA.
- Routing: Connects all components of the system on the FPGA.
- Generate bit-file: Produces a single configuration file to be downloaded on the FPGA.

The navigator tool-bar is displayed on the left hand side of the XPS window as shown in **Figure 14.** 

#### Figure 14. Flow Navigator

In the navigator toolbar under the implementation flow you will find a button labeled "Generate BitStream". Click this button to start the hardware building process which will take some time to perform all the tasks listed before. A lot of information will be displayed in the "Console" window in the bottom of the XPS window. Watch the progress of the work until the "Console" window displays the message shown in Figure 15. This message states that the tool has created the file "System.bit" which contains all the required configuration bits. At this point we are done creating the hardware.

▲ WARNING: PhysDesignRules: 2410 - This design is using one or more 9K Block RAMs	<u>^</u>
(RAMB8BWER). 9K Block RAM initialization data, both user defined and	
default, may be incorrect and should not be used. For more information,	
please reference Xilinx Answer Record 39999.	
DRC detected 0 errors and 25 warnings. Please see the previously displayed	
individual error or warning messages for more details.	
Creating bit map	
Saving bit stream in "system.bit".	
Bitstream generation is complete.	
Done!	C100
	-
< III	•
Console A Warnings S Errors	

#### Figure 15 Hardware Build Done

# Part 2: Profiling

In the first part of the tutorial we were able to build the hardware part of the system. In the second part we will write a simple C application and compile it for MicroBlaze processor that we built in part one. The process of building software application and running it is composed of the following tasks:

- Create a software workspace
- Create a board specification project to link the hardware to the software platform
- Create C project and add a source file to it
- Compile the C project to create ELF (embedded executable file)
- Merge the ELF file to the bit-stream created in part 1 (system.bit) to generate a complete bit file that contains both hardware and software (download.bit).
- Download the final bit file (download.bit) to the FPGA to run the application on the processor system.

**Step 1:** We perform all these tasks using another tool called Xilinx Software Development Kit (SDK). We start the SDK using the button **"Export Design"** in the Navigation toolbar. This button copies the bit-file (system.bit) and some other files necessary to link the hardware to a directory called **"SDK\SDK\_Export"** under the projects folder. It will ask if you want to start the SDK. Click **"Export Design"** which starts the window shown in **Figure 16**. In which you have two options; **"Export Only"** to just copy the hardware file (used when the SDK is already running) or **"Export & Launch SDK"** which will copy the files and starts the SDK; <u>click this button</u> to start the SDK.

Export to SDK / Launch SDK	? ×
This dialog allows you to export hardware platform information to be used in SDK.	
Include bitstream and BMM file	
(XPS will regenerate bitstream if necessary, and it may take some time to finish.)	
Directory location for hardware description files	
SDK\SDK_Export	
Evenert Only Evenert & Launch SDK Concol	

Figure 16. Export to SDK (Step 1)

**Step 2:** When the SDK starts the work space launcher window shown in **Figure 17**. The workspace as a software environment which is used to collect several projects in one entity. As the SDK is based on eclipse platform, those who are familiar with eclipse will find it easy to deal with the SDK. Change the workspace path as shown in Figure 17.

Select a w	orkspace		
Xilinx SDK st Choose a we	ores your projects in a folder called a workspace. orkspace folder to use for this session.		
Workspace:	C:\Projects\EDK_Projects\mydesign\SDK\Workspace	• [	Browse
🔲 Use this a	s the d <mark>e</mark> fault and do not ask again		
		ОК	Cancel

Figure 17. Workspace Launcher (Step 2)

**Step 3:** The SDK main window will start then as shown in **Figure 18**. The SDK IDE development area is divided into several areas. The left-hand side is the Project Explorer window which shows one project **"mydesign\_hw\_platform"** which represents the hardware created in part one. You will notice it includes the file **"system.bit"**. The middle area is the code area. By default the file **"system.xml"** which lists the components of the system and information about each of them.



Figure 18. SDK Main Window (Step 3)

**Step 4:** The next step is to create the Xilinx Board Support Package. In the project explorer window, select File -> new. A new project window will appear as shown in **Figure 22**. A list of projects that can be created is shown. Select "Xilinx Board Support Package" which will enable you to create a standalone board support package (no operating system). As seen in Figure 23.

New Proje	t	
Select a w	zard	
Wizards:		
type filter te	d	
🖄 Xilin	x Board Support Package	
Xilin Xilin	< C Project	
Xilin Xilin	x Hardware Platform Specification	
👂 📴 Gen	eral	
▷ 🔁 C/C	**	
p 🖉 Aun		
1	C Back Next >	Einich Cancel
U	N DOLK INEXE >	Cancer

Figure 19. New Project (Step 4)

🔞 New Board Support Pack	age Project	
Xilinx Board Support F	ackage Project	n-h
Create a Board Support Packa	ge.	
Project name: standalone_t	sp_0	
Use default location		
Location: D:\a0-PersonalFile	s-D\z0-FPGA_Projects\EDKBasedProjects\Tutorial-4\SDK	workspace'st Browse
Choose file system	default 💌	
Tarattindena		
Hardware Platform: Tutors	al-4 hw platform	•
CPU: microb	laze_0	
Board Support Package OS		
xilkernel standalone	Standalone is a simple, low-level software layer. It pr processor features such as caches, interrupts and ex features of a hosted environment, such as standard abort and exit.	ovides access to basic ceptions as well as the basic input and output, profiling,
(?)		Finish Cancel

Figure 20. Xilinx Board Support Package Project (Continue .. Step 4)

A new window will appear "Board Support Package Settings" which will enable you to set certain parameters as seen in **Figure 24**.

ntrol various settir	igs of your Board Support P	Package.		
Overview standalone drivers opu	standalone_bsp_0 OS Type: standalo OS Version: 3.03.a Target Hardware	vie I	Standalore is a simple, two level software layer. It provides access to basic processor interrupts and exceptions as well as the basic features of a hosted environment, such profiling, abort and ext.	features such as caches, as standard input and outp
	Hardware Specificati Processor: Supported Libraries Check the box next	on: D: (a0-Personal <sup>®</sup> microblaze_0 to the libraries you	ee-org/on-mode_modects/luctonamigue.yourspace/luctonamigue.yourspace/luctonamig_mw	gator on the left.
	Hardware Specificate Processor: Supported Libraries Check the box next	to the libraries you y	ees uigu +rou, yrojecti pusaakorojecti juutona - pus, wonapaoti juutona - juw, want included in your Board Support Package. You can configure the library in the navi	gator on the left.
	Hardware Specificati Processor: Supported Libraries Check the box next Name http://40	to the libraries you v	ere of bit mount induction in your Board Support Package. You can configure the Board you the new Description.	gator on the left.
	Hardware Specificati Processor: Supported Libraries Check the box next Name Mp140 stifatfs	to the libraries you v Version LOO.a LOO.a LOO.a	ere of bit mount grant purposes purposes of the second sec	igator on the left.
	Hardware Specificatis Processor: Supported Libraries Check the box next Name hip140 xiffatfs xiffatfs xiffath	to the libraries you v Version 1.00.a 3.00.a	ere of Dir Hnshurg Tegette European of Control (Unional – Guine your good (Unional – guine) want included in your Board Support Redage, You can configure the Board you the new Description July 17 Corp Stack (Branzy IndP VI.4.0, Xinnx adapter VI Provide read/write routines to access Res stored on a F Xinn Fach brancy for brank(XOC Comparing parallel fach	igator on the left.
	Hardware Specificati Processor: - Supported Ubraries Check the box next Name Imp140 xifatfs xifatfs xifath xiliaf	to the libraries you a Version 1.00.a 1.00.a 3.00.a 2.04.a	ere of bit mount in your Board Support Package. You can configure the Brary in the navi Description Multi TOD/IP Stack Brary: IndP v1.4.6, Xilon: adapter v1 Provider and print in Configure to the State of the Tourist and State Xilon: Pachi Ibrary: for IndIAVAD CPT compliant parallel fash Xilon: Pachet and Stard Real Leary	igator on the left.

Figure 21. Xilinx Board Support Package Settings (Step 4)

Highlight standalone in the "Board Support Package Settings" and change the value for "enable\_sw\_intrusive\_profiling" to true. Under "enable\_sw\_intrusive\_profiling" you will find a setting for "profile\_timer". Change the value from "none" to xps\_timer\_0 as seen in **Figure 25.** 

⊡ drivers	Name stdin stdout	Value RS232 Uart 1	Default	Type	1020 800
фu	stdin stdout	RS232 Uart 1			Description
	stdout		none	peripheral	stdin peripheral
		RS232_Uart_1	none	peripheral	stdout peripheral
	enable_sw_intrusive_profiling	true	false	boolean	Enable S/W Intrusive Profiling on
	profile_timer	xps_timer_0	none	peripheral	Specify the Timer to use for Prof

Figure 22. Xilinx Board Support Package standalone Settings (Step 4)

The final step is to highlight "cpu" under "drivers" and change the "extra\_compiler\_flage" from –g to "-g –pg" as seen in Figure 26.

Overview	Configuration for driver: cpu				
🖻 drivers	Name	Value	Default	Type	Description
cpu	compiler	mb-gcc	mb-gcc	string	Compiler used to compile both BSI
	archiver	mb-ar	mb-ar	string	Archiver used to archive libraries
	compiler_flags	-02 -c	-02 -c	string	Compiler flags used in BSP and lib
	extra_compiler_flags	-g -pg	-9	string	Extra compiler flags used in BSP a

Figure 23. Xilinx Board Support Package cpu Settings (Step 4)

**Step 5:** The following step is that we create a C project using the SDK. In the project explorer window, select File -> New->Project. A new project window will appear. A list of projects that can be created is shown. Select **"Xilinx C Project"** which will start a Wizard to create a simple C program. Select **"Xilinx C Project"** then click **"Next"** to continue.

**Step 6:** The second screen of the New Project Wizard is shown in **Figure 27**. This screen allows you to select a template for the project you want from a predefined list. Select "Hellow World" which is a template for a simple application that will send some text through the serial port. We will modify this application later. Select "Hello World", keep everything else unchanged, and then press next.

lew Xilinx C Pro	oject make application proje	ect. Choose from one of the sample applications.	F
Project name: hel	llo_world_0		
🔽 Use default loc	ation		
Location: C:\Proj	ects\EDK_Projects\myv	design\SDK\Workspace\hello_world_0	Browse
Choose f	file system: default	-)	
Target Hardware			
Hardware Platfor	m: mydesign hw plat	form	+
Processor	microblaze 0		
	[microsite_o		
Select Project Ter	mplate	Description	
Empty Applicatio	on	Let's say 'Hello World' in C.	
Helio World IwiP Echo Server Memory Tests Peripheral Tests SREC Bootloader Xilkernel POSIX 1	Threads Demo		×
3)		< Back Next > Finish	Cancel

Figure 24. New Project (Step 5)

**Step 7:** A new screen of the New Xilinx C Project will appear as shown in **Figure 24**. This screen allows you to either "Create a new Board Support Package project" or to "Target an existing Board Support Package". We will choose "Target an existing Board Support Package" which is the standalone\_bsp\_p (OS:standalone) which we created in the previous step. **Press Finish**.

New Project	
New Xilinx C Project	-6
Create a managed make application project. Choose from one of the sample applications.	1
C Create a new Board Support Package project	
The template provided by application 'Helio World' will be used to configure the project.	
Project name: hello_world_bsp_0	
🔽 Use default location	
Location: D: \a0-PersonalFiles-D\z0-FPGA_Projects\EDKBasedProjects\Tutorial-4\SDK\workspace\helk	Browse
Choose file system: default 💌	

Figure 25. Target an existing Board Support Package (Step 7)

Step 8: Expand the folder "src" in the "hello\_world\_0" project. You will notice several C files as seen in Figure 29. Double click the file "hello\_world.c" to open it. The code should look like Figure 30. The code is a simple C application that will print the word "Hello World" on the output device.

		5	\ ∑
Image: State of the state	tep	8)	
<pre>#include <stdio.h> #include "platform.h"</stdio.h></pre>			
<pre>void print(char *str);</pre>			
<pre>int main() {</pre>			
<pre>init_platform();</pre>			
<pre>print("Hello World\n\r");</pre>			
<pre>cleanup_platform();</pre>			
<pre>return 0; }</pre>			

Figure 30. Initial Hello World "Source Code" (Step 8)

**Step 9:** Modify the source code with your application that you need to profile. When done save the file. Note that once you are done, the SDK will start compiling your code automatically and generate the output files (in this case it is the download.bit file). To display any messages you will need to set the terminal as will be explained in the Appendix at the end.

**Step 10:** Highlight the "hello\_world\_0" with your mouse and right click. This will show a new menu. Zoom on "C/C++ Build Setting". This will produce a new screen "C Code Properties" as seen in Figure 31.

Properties for hello_world_0		
	Settings	
<ul> <li>B. Resultice</li> <li>Builders</li> <li>C. C. ++ Build</li> <li>C. C. ++ Build</li> <li>Build van Vochme</li> <li>Build van Vochme</li> <li>Logging</li> <li>Settingel</li> <li>Teel Chan Editor</li> <li>Bild Van Betragel</li> <li>Teel Chan Editor</li> <li>Run (Debug Settingel</li> </ul>	Configuration: Debug [Active]           Tord Setting:         Build Artifical:         Binary Parsens         Enor Parsens           Image: Strateging and Strateging	Manage Configurators
	Г	Restore Defaults Apply
(?)		OK Cancel

Figure 31. Properties for hello\_world\_0 (Step 10)

**Step 11:** Highlight profiling and enable it as seen in Figure 32. A message "Finished building: hello\_world\_0.elf.elfcheck" will be produced in the console.





**Step 12:** Again highlight "hello\_world\_0" and right click with the mouse. The same menu that appeared in Step 10 will appear again. This time choose "generate linker script". A new window with the title "Generate linker Scripts" will appear as seen in Figure 33. Change both the Heap Size and Stack Size to 3000. Press "Generate". This will increase the size of Heap and Stack to allow a medium sized code to be profiled properly. A message might appear "Linker Script Already Exists" and will ask you to overwrite the file! Press "Yes". Again a message "Finished building: hello\_world\_0.elf.elfcheck" will appear on the console.

Curbud Settings     Advanced       Projects: Inde_work1_0     Output Settings       N_Pojects: Inde_work1_0     Output Settings       N_Pojects: IDX08xedProjects\11.tornid=45CX/workspace?ields\u00efunctions     Imb_onts_dhe_onte       Place Data Settions in:     Imb_onts_dhe_onte       Place Heap and Stack in:     Imb_onts_dhe_onte       Place Heap and Imb_onte     Imb_onts_dhe_onte       Place Heap and Imb_onte     Imb_onts_dhe_onte       Imb_onts_dhe_onte     Imb_onts_dhe_onte       Imb_onts_dhe_onte     Imb_onts_dhe_onte	enerate linker script						9
Output Settings     Advanced       Project. In Mol_world_0     Output Settings       A_Project. In Mol_world_Stacking     Place Code Sections in:       Modify project SUturnia-HSDR/workspace/hello_world_0/lsrc/lsorpt.M     Browne       Place Data Sections in:     Imb_ontir_dmb_ontir       Place Data Stack in:     Imb_ontir       Imb_ontir_dmb_ontir     Imb_ontir       Place Data Stack in:     Imb_ontir       Imb_ontir_dmb_ontir     Imb_ontir	ontrol your application's memory r	nap.					
A_Project EXRAISeProject STUTuoria-HSDK/workspace/jeels_workd_Ofsrc/isorpti.dl     Browne       Medify project build settings as follows:     Imb_orth_dhb_orth       Yeace Heata and Stack Int     Imb_orth_dhb_orth       Yeace Heata and Yeace Heata and Yeace Heata and Yeace Heata     Imb_orth_dhb_orth       Yeace Heata And Yeace Heata     Imb_orth_dhb_orth <th>Dutput Settings roject: hello_world_0 Dutput Script:</th> <th></th> <th></th> <th></th> <th>Basic Advanced Place Code Sections in:</th> <th>imb_cntir_dimb_cntir</th> <th><u> </u></th>	Dutput Settings roject: hello_world_0 Dutput Script:				Basic Advanced Place Code Sections in:	imb_cntir_dimb_cntir	<u> </u>
Modify project build settings as follows:     Image: Stack int Im	A_Projects\EDKBasedProjects\Tu	torial-4\SDK\workspace\he	ello_world_0\src\lscript.ld	Browse	Place Data Sections in:	imb_cntir_dimb_cntir	•
Set generated script on all project build configurations         Heap Size:         [0000]           Hardware Memory Map.         Size         ~2.93 KB           Memory         Base Address         Size           Jamb_onth_dis_onth         0x00000000         64 KB	lodify project build settings as fol	lows:			Place Heap and Stack in:	imb_cntir_dimb_cntir	7
Handware Memory Map         Stack Size         ~2.93 KB           Memory         Base Address         Size            Imb_ontin_dimb_ontin         0x00000000         64 KB	Set generated script on all projec	t build configurations		•	Heap Size:	B000	
Memory Base Address Size inb_cntir_dinb_cntir 0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	lardware Memory Map				Stack Size:	~2.93 KB	
imb_cntir_dimb_cntir 0x0000000 64X8	Memory	Base Address	Size				
	imb_ontir_dimb_ontir	0x00000000	64 KB				

Figure 33. Generate Linker Script (Step 12)

**Step 13:** Make sure the FPGA is powered on. Select the menu item "Xilinx Tools" from the menu bar and then select "Program FPGA". The "Program FPGA" window of Figure 34 will appear asking you to select the executable (ELF) file to be used.

Beside "microblaze\_0" select the second option

"..\mydesign\SDK\Workspace\hello\_world\_0\Debug\hello\_world\_0.elf", then press "Program". This will start the programming process on the board.

📴 Program FPG/		2
Program FPG	A	-
Specify the bitstre	am and the ELF files that reside in BRAM memory	-0-0
Hardware Configu	ration	
Hardware Specific	ation: D:\a0-PersonalFiles-D\z0-FPGA_Projects\EDKBasedProjects\Tutorial-4\SDK\workspace\Tutorial-4_hw_platform\system	n.xml
Bitstream: D:\a	D-PersonalFiles-D\z0-FPGA_Projects\EDKBasedProjects\Tutorial-4\SDK\workspace\Tutorial-4_hw_platform\system.bit	Browse
BMM File: D:\a	0-PersonalFiles-D/z0-FPGA_Projects\EDKBasedProjects\Tutorial-4\SDK\workspace\Tutorial-4_hw_platform\system_bd.bmm	Browse
Software Configu	ation	
Processor	ELF File to Initialize in Block RAM	
microblaze_0	4\SDK\workspace\hello_world_0\Debug\hello_world_0.elf	
-		
?	Program	Cancel

Figure 34. Program FPGA (Step 13)

- **Step 14:** Replace the Hello World Code with the program you want to profile.
- **Step 15:** Point the mouse on "Run" and activate "Run Configurations". A new window will appear as shown in Figure 35.



Figure 35. Run Configurations (Step 15)

**Step 16:** Double click on "Xilinx C/C++ ELF". The previous window (Figure 35) will modify to a new window as seen in Figure 36.

		and the second se		
eate, manage, and run co	onfigurations			
9 🗈 🗶 😑 🔆 🔹	Name: hello_world_0 Debug			
ype filter text	Main 👋 Device Initialization 😽	STDIO Connection	Debugger Options	Common
C/C++ Application	C/C++ Application:			
Launch Group	Debug\hello_world_0.elf		Search Project	Browse
Remote ARM Linux Applicati	Project:			
Xiinx C/C++ELF	hello_world_0			Browse
	Build (if required) before launching			
	Build configuration: Debug			•
	C Enable auto build	C Disable auto build		
	Use workspace settings	Cardina Manhaman	- Carteria and	
	Connect process input & output to a to	erminal.	seturius	
	Connect process input & output to a te	connare vvorsase	<u>Stution</u>	

Figure 36. Create, Manage and Run Configurations (Step 16)

**Step 17:** Click on "Profile Options" and a new view of the previous window will appear as seen in Figure 37. Make sure you enable Profiling and change the scratch memory address that collects profile data to 0x1500 (or some value that does not overwrite your code, heap and stack!!)

Run Configurations	×
Create, manage, and run co	figurations
Image: Second	Name: helo_world_0 Debug  Man & Debug STDIO Connection & Profile Options @ Debugger Options Common  Fable Profiling  Profiling Options Sampling Frequency (Hz): 10000 Histogram Bin Size (worlds): 4 Scratch memory address to collect profile data: 0x1500
Filter matched 6 of 6 items	Apply Revert
•	Run Close

Figure 37. Create, Manage and Run Configurations (Step 17)

**Step 18:** The system will execute your code and inform you that the profile results are saved and a gmon.out file is produced as seen in Figure 38.



Figure 38. Create, Manage and Run Configurations (Step 18)



**Step 19:** As seen in Figure 39 a gmon.out file is under Debug.

Figure 39. Create, Manage and Run Configurations (Step 19)

- **Step 20:** Double click on the gmon.out and if a new window appears press "yes"
- **Step 21:** The results of profiling will be displayed as seen in Figure 40.

g\gmon.out rogram file: :/a0-PersonalFiles-D g/hello_world_0.elf .6 bytes per bucket,	/z0-FPGA_Proj	ects/EDKBasedProje	cts/Tutorial-4/SDK/wor	kspace/hello_world_0/
Name (loc 🔺	Samples	Calls	Time/Call	%Time
Summary	2637			100.0%
CallFunBig	1433	1	143.300ms	54.34%
🕀 CallFunMedi	717	1	71.700ms	27.19%
CallFunSmal	358	1	35.800ms	13.58%
	0	2	Ons	0.0%
XIntc_Regis	0	1	Ons	0.0%
XIntc_SetIn	1	1	99.999us	0.04%
XUartLite_S	128	67	191.44us	4.85%
🗄 deanup_pla	0	1	Ons	0.0%
It disable_cacl	0	1	Ons	0.0%
	0	1	Ons	0.0%
init_platform     ini	0	1	Ons	0.0%
init_uart     initur     initur     init_uart     inituart     initur     ini	0	1	Ons	0.0%
🕀 main	0	0		0.0%
microblaze_	0	0		0.0%
+ outbyte	0	0		0.0%

Figure 40. Profile Results (Step 21)

# **Appendix A (Terminal Connection)**

Your C code might contain several printf or xil\_printf statements to print certain information on the screen. In order for you to display information you will need to follow the instructions below:

1. Highlight the Terminal icon at the bottom of the SDK screen and then press on the settings icon as seen in Figure 41.

N 88 🗖 🚛	🧶 - 🔛 - 🗶 🗖 🗋
	× № □ I <sub>1</sub>

Figure 41. Terminal Connection

 A new window will appear as seen in Figure 42. Make sure you choose Serial for the "Connection Type" and the appropriate port available. You might also want to set the Baud Rate and other settings.

Connection Typ	e:	1
Serial	4	
Settings:		-
Port:	COM4	•
Baud Rate:	9600	•
Data Bits:	8	•
Stop Bits:	1	•
Parity:	None	-
Flow Control:	None	•
Timeout (sec):	5	

Figure 41. Terminal Connection

3. Press "OK" and this will allow you to see all output on the terminal in addition to the console.

# **Notes:**