ABSTRACT

Construction and Validation of a Reconfigurable Computer Cluster

Willis Scott Troy, M.S.E.C.E.

Advisor: Russell W. Duren, Ph.D.

Cluster computing networks multiple computers (nodes) to exploit their parallel processing power. By pooling resources of multiple computers, computations can be decomposed and allocated across nodes with partial solutions collected and combined to form a complete solution (scatter and gather). By augmenting cluster nodes with reconfigurable hardware, systems can be configured with assistive devices that improve their computational performance; thus reducing computation time and increasing computational flexibility. This thesis evaluates a cluster of 16 Virtex II Pro Development boards that were integrated as an experimental cluster. The well-known 3DES algorithm was used to measure the runtime of multiple partitioned datasets (1 to 16 partitions) to quantify the execution speedup over a varying number of nodes. The results showed that performance can be improved with hardware acceleration, although there is complex interplay between node communication and file I/O timing that impacts the magnitude of the speedup.

Construction and Validation of a Reconfigurable Computer Cluster

by

Willis S. Troy, B.S.

A Thesis

Approved by the Department of Electrical and Computer Engineering

Kwang Y. Lee, Ph.D., Chairperson

Submitted to the Graduate Faculty of
Baylor University in Partial Fulfillment of the
Requirements for the Degree
of
Master of Science in Electrical and Computer Engineering

Approved by the Thesis Committee

Russell W. Duren, Ph.D., Chairperson

Steven R. Eisenbarth, Ph.D.

David B. Sturgill, Ph.D.

Accepted by the Graduate School
August 2009

J. Larry Lyon, Ph.D., Dean

Page bearing signatures is kept on file in the Graduate School.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

3DES - Triple Data Encryption Standard

BRAM - Block Random Access Memory

COTS - Commercial off the Shelf

CPU - Central Processing Unit

DDR SDRAM - Double Data Rate Synchronous Dynamic Random Access Memory

DHCP - Dynamic Host Configuration Protocol

EDA - Electronic Design Automation

EDK - Embedded Development Kit

FPGA - Field Programmable Gate Array

HDL - Hardware Description Language

HLL - High Level Language

HPC - High Performance Computing

IAT - Impulse Accelerated Technology

ICAP - Internal Configuration Access Port

JTAG - Joint Test Action Group

MPI - Message Passing Interface

NFS - Network File System

OS - Operating System

RAM - Random Access Memory

RCC - Reconfigurable Computer Cluster

RFS - Root File System

SATA - Serial Advanced Technology Attachment

SYSACE -System ACE

UART - universal asynchronous receiver/transmitter

VHDL - VHSIC Hardware Description Language

VHSIC - Very-High-Speed Integrated Circuits

XPS - Xilinx Platform Studio

XUP - Xilinx University Program

ACKNOWLEDGMENTS

I would like to thank Dr. Duren for giving me the opportunity to pursue research avenues I enjoy and bending my mind in ways I thought unbendable. I would like to thank Dr. Eisenbarth for his guidance and his willingness to always help me. I would like to thank Dr. Sturgill for tolerating me in his courses despite the fact I was wholly unprepared, and yet genuinely seemed interested in teaching me. I would like to thank Spenser Gilland, whose help with the OS allowed this thesis come to fruition and allowed me to make a friend in the process. And lastly, I would like to thank my friends and family who tolerated my bickering and odd hours due to this thesis.

DEDICATION

To my family
for believing in me when I could hardly believe in myself

CHAPTER ONE

Introduction

Importance

Cluster computing refers to the technique of networking multiple computers to exploit their parallel processing power. By pooling the resources of multiple computers, computations can be allocated to systems and solved independently. Upon completion these systems can report to the partial solutions to a master node (system that distributed data) for further instruction. Clusters are typically comprised of commercial off-the-shelve (COTS) components such as prefabricated computers. While these systems offer remarkable performance, they lack flexibility and sub-optimal for many tasks. By augmenting these computer systems with reconfigurable hardware, such as contained in FPGAs, we can tailor the system hardware for specific functionality, and hence improve performance. For most applications, hardware accelerators run two to three magnitudes faster than equivalent software implementations. Thus, a reconfigurable computing cluster has the capability to reduce computation time and grant greater computational flexibility.

Research Objectives

The primary objective of this thesis was to investigate the performance characteristics of a reconfigurable computing cluster. The cluster adhered to both the cluster computing and reconfigurable computing paradigms. Particular to cluster computing, the cluster must be scalable, affordable, and applicable to a wide range of computational tasks. The reconfigurable paradigm calls for hardware flexibility with high performance. For research purposes, the cluster must allow porting of existing well-known applications, remain non-proprietary, and have a minimal learning curve.

Thesis Organization

The background of reconfigurable computing and computing clusters will be covered in Chapter 2. Chapter 3 will discuss the hardware implementation- in particular the board's Xilinx setup and the cluster's overall construction. Chapter 4 will then cover the operating system ported to the XUP Virtex II Pro boards, and lessons learned concerning particular operating systems. Chapter 5 will discuss the applications that are used to generate quantifiable execution data. Chapter 6 will cover the findings from the test applications implemented on the cluster. And finally, Chapter 7 will discuss the findings, and conclude with recommendations for further research.

CHAPTER TWO

Background on Reconfigurable Computing Clusters

Aforementioned in Chapter 1, a reconfigurable computing cluster is the mix of two distinct computing paradigms: reconfigurable computing and cluster computing. The mix of these two computing paradigms calls for scalability, affordability, hardware flexibility, and high performance.

Cluster Computing

Cluster computing is the utilization of multiple processing elements to gain the advantage of distributed computing. Nodes within a computer cluster are generally interconnected via Ethernet to allow communication across the system. Cluster computers have a strong presence in academia due to their availability and affordability.

Computer Cluster Example

A classic cluster computer example is the "Stone Souper Computer", Oak Ridge National Laboratories' first Beowulf cluster. The cluster was constructed from 126 surplus machines after their proposal for funding was denied. The end product was a heterogeneous cluster used to "produce maps of regions of ecological similarity within the 48 continuous US states" [1]. The cluster is pictured in figure 2. While the "Stone Souper Computer" was built mainly because of its affordability, there are many other clusters built for performance alone, various examples can be found at http://www.top500.org/.

Figure 1: Stone Souper Computer [1]

Communication

Cluster computing's divide and conquer approach to problem solving requires communication amongst the nodes. These communications are referred to as the scatter and gather process, in which data is sent from a single source to multiple destinations (scatter) and multiple sources send data to a single destination (gather). The scatter gather process predominantly utilizes the "single instruction multiple data" (SIMD) technique. SIMD has each node perform the same instruction on received data, thus partitioning the overall task. However, the "multiple instruction multiple data" (MIMD) technique can also be deployed on clusters by having nodes perform unique instructions on received data.

The effective standard for communication has become MPI (Message Passing Interface). This protocol is language independent and has three implementations: MPICH [2], LAM/MPI [3], and Open MPI [4]. For our purposes we use LAM/MPI within C/C++. This allows us to develop MPI communication within C/C++ programs.

Reconfigurable Computer Clusters

A reconfigurable computing cluster (RCC) is a cluster that uses reconfigurable components. Implementations can vary from a cluster of CPUs that interface to reconfigurable logic to a cluster of FPGAs. Our cluster will be of the latter form and will be predominately FPGAs.

Reconfigurable Computer Cluster Example

Cray's XD1 is a high performance computer comprised of twelve 64-bit AMD Opteron processors, six or twelve RapidArray processors, zero or six FPGA application acceleration processors, and one management processor. Each node is comprised of a "two-way or four-way Opteron SMP (Symmetric Multiprocessing) and its associated memory, one or two RapidArray processors, and an optional application acceleration processor." The application processors are FPGAs that act as coprocessors to the Opterons. The FPGAs are based off the Virtex II Pro and are included to accelerate tasks that are repetitive or computationally intensive [5]. Assuming that the optional FPGAs are included in the system, the XD1 is a RCC.

Reconfigurable Computing

Reconfigurable computing is the exploitation of reconfigurable logic to gain the advantages of hardware acceleration. Generally computation can be accomplished by either hardware elements or software programs or a combination. In most computing systems, hardware is fixed and only software is modifiable. In a reconfigurable system both the hardware and software may be configured to offer optimal solutions to computing tasks.

Hardware is configured or “programmed” by an Electronic Design Automation (EDA) toolset that generates a physical implementation from hardware description code. The toolset is typically proprietary and provided by the reconfigurable component manufacturer- in this case Xilinx’s EDK suite 9.1i. The hardware code is written in a HDL (Hardware Description Language), which is either VHDL (VSHIC Hardware Description Language) or Verilog. However, HLL (High Level Languages) to HDL compilers have become popular in hardware development in an effort to enable developers’ strong backgrounds in languages such as C, C++, and Java. Impulse C [6], Join Java [7], and System C [8] are examples of such language-compiler combinations that utilize HLLs for hardware development. These development tools convert HLL to HDL for the developer, *a la* black box methods. The generated HDL can then be used by the EDA toolsets to implement the target component. While HLL based compilers reduce complex designs to a HLL, they lack the degree of specification and optimization available in handcrafted HDL code.

IMPULSE C was utilized to develop applications for the experimental cluster. IMPULSE C is a C-to-HDL compiler developed by IMPULSE Accelerated Technology. IMPULSE C was used over a conventional HDL because the project required a complex peripheral to adequately validate the cluster’s success. IMPULSE C was used over other HLL-to-HDL languages because previous research [9] at Baylor had concluded that IMPULSE C is a simple and inexpensive tool to use in application development. The specifics of the application are covered in Chapter 5.

Generally, the reconfigurable components in a reconfigurable computer are Field Programmable Gate Arrays (FPGAs). A FPGA typically contains reconfigurable logic blocks, a microprocessor(s), RAM, and I/O blocks. The reconfigurable logic is the EDA toolset's target for the HDL physical realization. The reconfigurable components used were embedded in a Virtex II Pro development board produced by Xilinx. Sixteen of these boards comprised the nodes of the experimental cluster. The Virtex II Pro substrate contains two IBM PowerPC processors, 136 eighteen-bit multipliers, 136 BRAMs, and one XC2VP30 FPGA with approximately three million logic gates. The development board features a 2 GB DDR SDRAM DIMM slot and supports several I/O configurations e.g. RS-232, 10/100 Ethernet, JTAG, and SATA. Figure 1 depicts the Virtex II Pro development board with labeled I/O [10]. This research does not utilize all of the board's resources. The particulars of the boards' setup are discussed more in depth in Chapter 3.

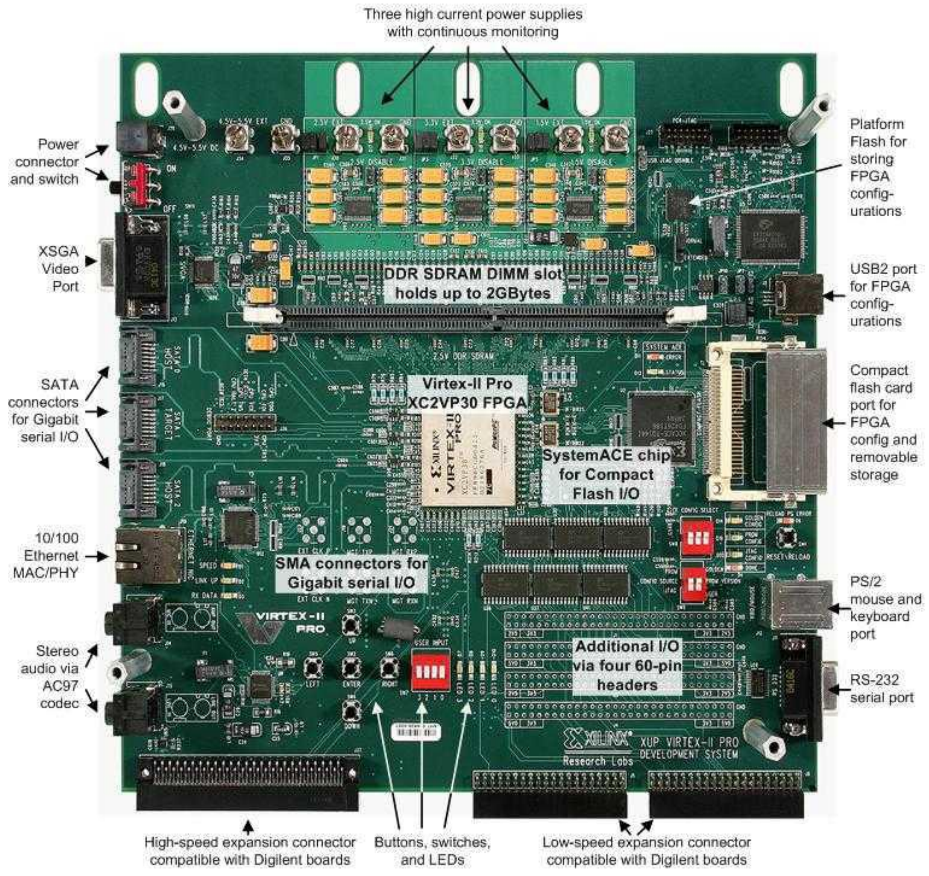

Three high current power supplies with continuous monitoring
Power connector and switch
XSGA Video Port
SATA connectors for Gigabit serial I/O
10/100 Ethernet MAC/PHY
Stereo audio via AC97 codec
Platform Flash for storing FPGA config-urations
USB2 port for FPGA config-urations
Compact flash card port for FPGA config and removable storage
PS/2 mouse and keyboard port
RS-232 serial port
DDR SDRAM DIMM slot holds up to 2GBytes
Virtex-II Pro XC2VP30 FPGA
SystemACE chip for Compact Flash I/O
SMA connectors for Gigabit serial I/O
Additional I/O via four 60-pin headers
High-speed expansion connector compatible with Digilent boards
Buttons, switches, and LEDs
Low-speed expansion connector compatible with Digilent boards

Figure 2: Virtex II Pro Development Board [10]

CHAPTER THREE

Design and Implementation of Hardware Base

Reconfigurable Elements

As mentioned in Chapter 2, the Virtex II Pro Development board is our reconfigurable component. These boards are available from Digilent, Inc. [11]. The structure of the Virtex II Pro substrate is described in Chapter 2; however this research only utilizes 1 of the 2 PowerPC 405 processors. Although the DIMM slot is capable of holding 2 GB of RAM for this research the slot contained a single 256 MB RAM DIMM. Of the I/O capabilities only the 10/100 Mb Ethernet was utilized and connected to a centralized switch/router. The RS232 port was incorporated in the design for system level debugging, and the JTAG interface was used hardware debugging. SYSACE interfaced with a Compact Flash (CF) card to initialize the board's setup at boot, as well as containing a RW OS image. (The multi-gigabit transceivers, video, audio, SATA and PS2 were not used.) The multipliers, BRAMs and logic gates are used where needed as dictated by the functional requirements of each hardware accelerator. The Xilinx Platform Studio tool-chain's configuration of the base system is given in Appendix A.

Multi-Board Setup

The cluster consists of:

1. 16 - XUP V2P boards (http://www.xilinx.com/univ/xupv2p.html),
2. Mini-ITX VIA-x86 form factor system (running Debian 5.0),
3. PROSAFE® 24 port Gigabit Ethernet switch 10/100/1000 Mbps, Model JGS524,
4. A custom 10 AMP 5 VDC power supply,
5. A custom rack to support all the hardware,
6. Off the shelf CAT-5 Ethernet cables

These components can be clearly seen in Figure 3, and a diagram is listed below in Figure 4 for convenience.

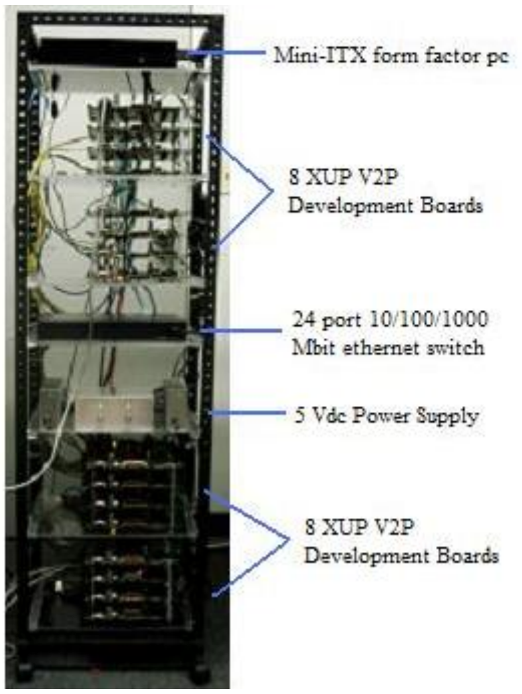

Mini-ITX form factor pc
8 XUP V2P Development Boards
24 port 10/100/1000 Mbit ethernet switch
5 Vdc Power Supply
8 XUP V2P Development Boards

Figure 3: 16 Node Cluster Rack Mounted Physical System

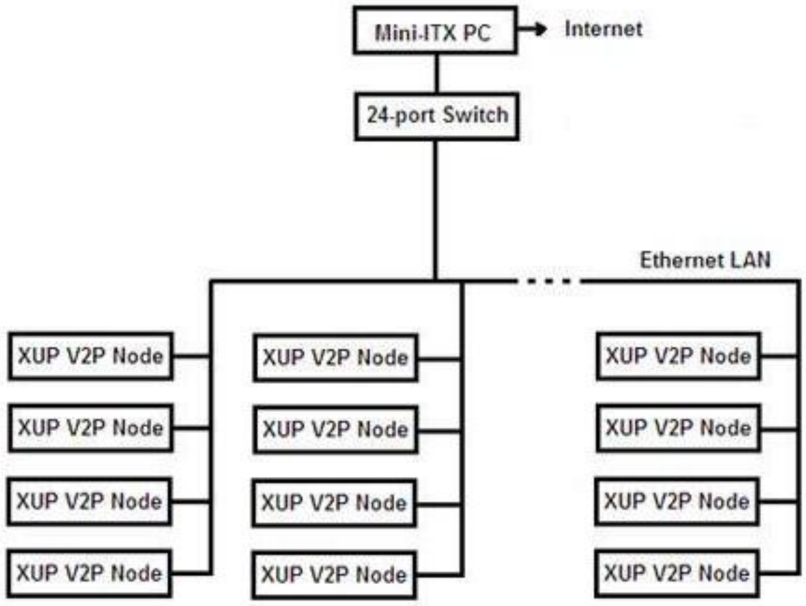


Figure 4: 16 Node Cluster Logical Design

The boards are physically arranged in groups of four. This allows boards to be easily replaced if one fails. The Mini-ITX acts as a Network File System (NFS) to easily transfer files to the CF card in each development board. The Mini-ITX has two network interfaces- one, a 10/100 Mb port is connected to the switch and the second 10 Mb port is connected to the internet. The system also served as a network address (DHCP) server which provided local IP addresses during boot-up of each node. Each node has the RS232 port facing forward for easy access to the node's serial debugging channel.

Power

The cluster is connected to the power mains through a surge protector, which allows the entire cluster to be turned off and on at once. Although 5 VDC is supplied to each node, an onboard power regulator reduces the voltage level to 3.3 VDC.

Custom Power supply

A custom power supply was constructed to supply the 5 volts to the boards. This was done to reduce the number of AC plugs needed to connect the system to the AC mains, and increase aesthetics. The power supply consists of 2 Lambda HK150A-5/A single output switched regulator supplies, with supply side noise filters. Figure 5 is a diagram of the custom power supply.

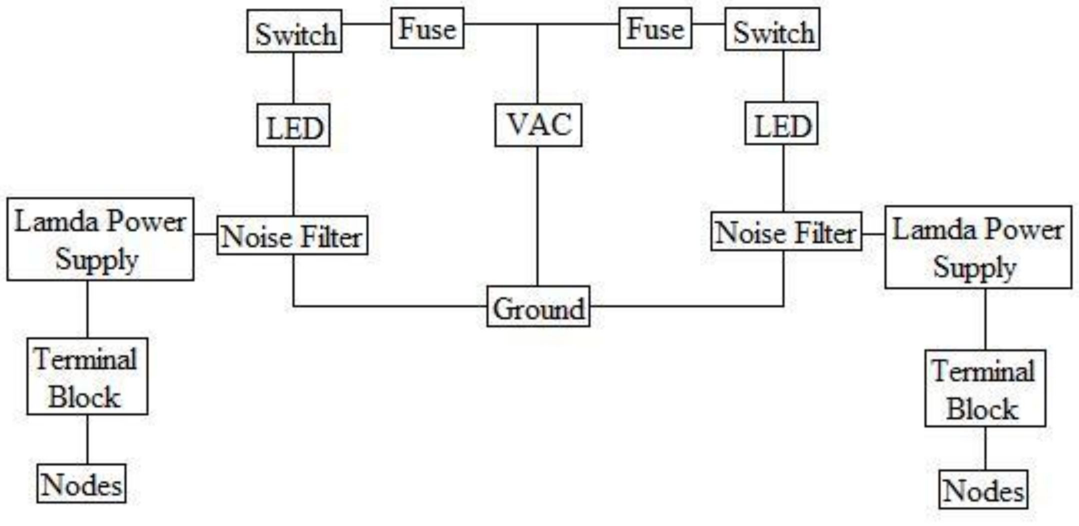
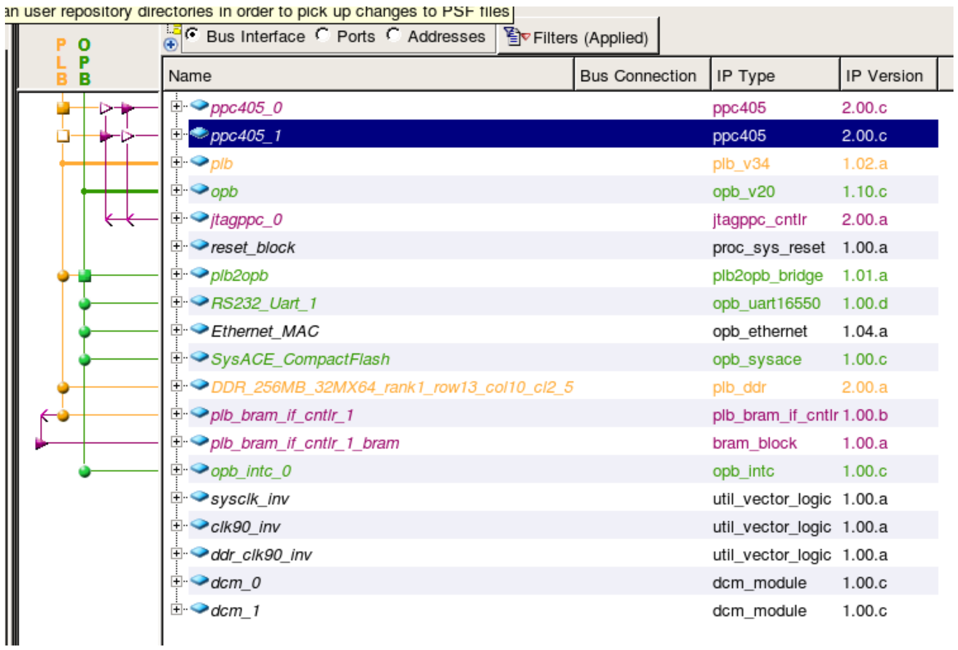


Figure 5: Power Supply Diagram

Base System Creation in XPS

Each node possesses a basic framework of common peripherals. These peripherals include the SYSACE, the DDR controller, the UART, and the Ethernet. The SYSACE provides boot capability by copying the OS image into RAM from the CF. The executable image of developed applications was loaded onto the CF card and loaded into memory for execution by a multicast MPI command (mpirun) from the master node (Mini-ITX system). The DDR controller allows BUS access to RAM; the processor and peripherals are connected via this BUS which allows the flow of data/programs stored in RAM to be accessed by the processor and peripherals. The UART is included as a console port accessible by the OS to allow OS monitored system debugging and system messages during boot-up. An Ethernet controller is included to provide high speed communication between each node and the master node via the switch. However, much of the functionality of the board is left unused, but can be incorporated in expanded research endeavors. For example, the boards have two processors, but this research utilized only a single processor. Step-by-step instructions for the base system creation are included in Appendix A. The system assembly view window should appear as in figure 6.


an user repository directories in order to pick up changes to PSF files
(✓ Bus Interface) Ports Addresses
Filters (Applied)
PLB
OPB

Name	Bus Connection	IP Type	IP Version
ppc405_0		ppc405	2.00.c
ppc405_1		ppc405	2.00.c
plb		plb_v34	1.02.a
opb		opb_v20	1.10.c
jtagppc_0		jtagppc_cntlr	2.00.a
reset_block		proc_sys_reset	1.00.a
plb2opb		plb2opb_bridge	1.01.a
RS232_Uart_1		opb_uart16550	1.00.d
Ethernet_MAC		opb_ethernet	1.04.a
SysACE_CompactFlash		opb_sysace	1.00.c
DDR_256MB_32MX64_rank1_row13_col10_cl2_5		plb_ddr	2.00.a
plb_bram_if_cntlr_1		plb_bram_if_cntlr	1.00.b
plb_bram_if_cntlr_1_bram		bram_block	1.00.a
opb_intc_0		opb_intc	1.00.c
sysclk_inv		util_vector_logic	1.00.a
clk90_inv		util_vector_logic	1.00.a
ddr_clk90_inv		util_vector_logic	1.00.a
dcm_0		dcm_module	1.00.c
dcm_1		dcm_module	1.00.c

Figure 6: System Assembly View of Base System

CHAPTER FOUR

Design and Implementation of Operating System

Due to the complexity involved in developing networking and cluster software, an operating system was needed on the XUPV2P. Versions of Linux and QNX have been ported to the PPC 405. Linux (Debian 5.0 and MonteVista), and QNX 2.1 support the PPC processors inside the XUPV2P. Linux versions are typically free, open source, and commonly used in clusters. QNX is a real-time microkernel architecture OS designed specifically for embedded devices, small binary sizes, and low resource usage. From a educational perspective, QNX is the better OS for this research because it provides better timing support, however, Linux was more malleable to our situation and allowed us to quickly develop a workable solution. (Note: QNX was closed source during our attempts but has recently been open-sourced).

In embedded software development, there are usually two computers involved. The machine that is used to build the basic system components such as the kernel and file system is called the cross-development host and the machine on which the software runs is called the target. In all of the attempts, an Intel X86 based host was used to develop PowerPC software.

Multiple attempts were made at configuring these operating systems to the Virtex II Pro, but were often thwarted due to complexity and inexperience with software development. QNX and MonteVista Linux were configured using Windows XP as the cross-development host. The Debian port was developed using Redhat Linux as the cross development host. In generally, the Windows cross development environment was found to be less stable and slower than a Linux cross-development environment. In addition, the multiple versions of CYGWIN on the Windows cross-development host which needed to be installed for the development environment and the Xilinx tools generated registry conflicts.

QNX

The initial attempt at configuring an operating system focused on QNX [12]. QNX was chosen because professor Eisenbarth had used it on previous projects. QNX is a POSIX compliant mircokernel RTOS. A RTOS would be an ideal OS for a reconfigurable computer cluster because of its deterministic nature in relation to I/O. The configuration was derived from a working implementation for Xilix's ML403 development board. The ML403 utilizes the same hard processor as the XUPV2P, so only peripheral changes were needed. The new configuration simply matched the address range of peripherals to those of the QNX's ML403 project. From here, the hardware and libraries were generated to build the SYSACE file. While QNX was successfully up and running on a ram disk, the configuration was not ideal and had very little flexibility. As new peripherals were added; the absence of functional device drivers and correct configuration information made device driver debugging difficult. In addition the development environment CYGWIN interface crashed under windows. CYGWIN is a program that allows emulation of a POSIX like environment in Windows. Also, a writable file system could not be successfully configured. Ultimately the decision was made to move on to a new OS when development stalled and it was determined QNX lacked the functionality needed. Besides the setup lacking the ability to add components, QNX at the time was not free to download or open source. It was also determined that QNX did not seem to support SSH, a secure protocol for communication between networked systems; because the eventual goal was to connect this system to the internet for remote use it was decided a different OS was a better option. In addition, QNX networking (Qnet), a transparent networking protocol, does not work across machines differing in endianness, thus Qnet can't be used between the XUP V2P boards and the Mini-ITX. Because of the inability to configure QNX easily with the functionality desired; it was decided to default to a Linux distribution as Linux is the typical OS of choice for clusters.

MonteVista

MonteVista [13] is a Linux distribution that has been successfully ported to the XUP-V2P [14] boards by others, but this project was not similarly successful. In hindsight, difficulties included using a Windows XP as development platform and not fully understanding the cross-compilation process. The instructions in reference 13 were attempted under the CYGWIN environment; whereas the original document used the Gentoo PowerPC-based OS as the development platform. Because Gentoo is a Linux distribution that runs on the PowerPC their instructions did not require a cross-compiler toolset, a mistake this researcher failed to realize. Most build issues occurred because the CYGWIN environment lacked the necessary applications to reproduce the build. The inability of the CYGWIN environment to correctly support the development process is not surprising in hindsight because the CYGWIN environment supports a subset of Linux system applications. However, MonteVista can likely be successfully configured for the boards using the same method in the following section. Gentoo is not necessary to configure the OS for these boards; it just greatly simplifies the cross-compilation process.

DEBIAN

Our current system utilizes the Linux kernel from the Xilinx GIT repository. This is a 2.6.28 Linux kernel, upon which a Debian 5.0 root file system was created. Debian [15] is known as a very robust and secure server distribution with vast quantities of precompiled software and support for several processor architectures. Debian was chosen because of its extensive online software repository system and solid PowerPC support. The online software repository system allows binary images of programs to be quickly downloaded without worrying about software dependencies. Debian was chosen because it was the quickest path to a root file system with the networking and the applications needed. The Debian target configuration was hosted on an x86 machine running Redhat Linux with version 9.1i of Xilinx EDK and a cross-compiler built using the crosstools script [16]. While Xilinx EDK 10.1i is the latest version that supports the XUP V2P, this research determined that the memory controller did not load the OS's kernel correctly into RAM and prevented the OS from booting. This issue was resolved by using EDK 9.1i which utilizes a different memory controller in the XUP V2P board support package.

A lot of the preparatory work has already been completed by Xilinx. Specifically, the device tree generator and the Xilinx port of the Linux kernel can be downloaded from the git.xilinx.com website. Using these two projects it was possible to generate the Open Firmware device-tree files and a Linux kernel for the XUPV2P.

The device tree generator is used to provide the Linux kernel with an Open Firmware device tree file. Using this file, the Linux kernel is able to find and utilize the various peripherals in the system including the custom peripheral for 3DES acceleration. The device tree generator allows us to completely change the address map of the system and still boot Linux.

After placing the device tree in the Linux kernel source code, it is time to configure the kernel. A common configuration from the Xilinx ML405 was used as the base and some small additional tweaks were made. After building the Linux kernel, the kernel and BIT file were installed for testing using the JTAG interface.

Building Linux is a process of trial and error. It took well over 50 build and test cycles to decide on the final configuration. The development environment must be agile and allow quick reconfigurations and rebuilds. Using the JTAG interface and a well-designed development environment described at http://baylor-recomp.wikidot.com/, it was easy generate new images.

The JTAG interface was used primarily to quickly download new bitstream logic maps, OS images, and debug the kernel; however for the final implementation an ACE file was used to store the boot image. The Xilinx ACE system is essentially a bootloader for the Xilinx FPGAs. A specifically formatted Compact Flash (CF) card allowed the Xilinx system to find and install the ACE file located on the first partition of the Compact Flash card. The ACE file is a wrapper around both the bitstream logic maps and the kernel image. The first partition of the Compact Flash card is 128MB in length and contains this ACE file. The second partition is 1.9 GB in length and contains a Debian root file system.

The root file system on the second partition was created using debootstrap. This program creates an entire Debian root file system utilizing an internet connection to the Debian repositories. The Debian repositories are a collection of software which represent the Debian distribution. Debootstrap with its “--foreign” option allows the program to be run in 2-stages. The first stage is performed on the development host and requires Internet access and the second stage occurs on the target and requires no Internet access. Utilizing, the open-source Linux kernel from the Xilinx GIT repository and the Debian 5.0 distribution it was possible to quickly create a Linux system for the Xilinx boards. The steps to configure the OS to the XUPV2P are covered in Appendix B.

CHAPTER FIVE

Applications

Three applications were developed to run on the cluster after construction. Two applications developed for the system were 3DES implementations; one that did not use the logic resources and one that did. These two applications were used to determine the runtime difference between the hardware implementation and the software implementation. The third application is small program written to measure the communication overhead for file I/O and network communication between nodes. 3DES was chosen for implementation because of the algorithm's inherent parallelizability and the ability to generate a 3DES hardware implementation.

3DES Encryption Algorithm

3DES is an encryption algorithm included in IMPULSE Technology's IMPULSE C tutorial. The purpose of the tutorial was to demonstrate the speedup of hardware over software. The project is included with CoDeveloper 2.10 [6] and can be found in the CoDeveloper subdirectory under /Examples/Xilinx/VirtexIIPro/3DES. The example also corresponds to Chapter 8 in the IMPULSE C book "Practical FPGA Programming in C" [17]. However, the IMPULSE C implementation does not use an OS or parallelization. The algorithm encrypts an input stream eight characters (64 bit) at a time; this granularity makes it an algorithm that is easily parallelized. The granularity of the 3DES algorithm combined with the hardware implementation makes this a prime candidate for cluster evaluation.

3DES IP Core, Standalone Version

The 3DES IP core was generated and implemented on a standalone system (no OS present) to gain familiarity with IMPULSE C. The IP core was designed to connect to the PLB Bus and generates the hardware code in VHDL. IMPULSE C also generates application code (EDK C) to test the device once implemented on the XUPV2P, which gives information about the hardware interface needed for a device driver. Since the standalone application code communicates directly with the hardware without the presence of a device driver, the application must be aware of the hardware interface details. These details will be addressed in greater detail in the device driver section below. Appendix D shows how to create a standalone system, add the 3DES core, and add the IMPULSE C application code. It is also worth noting that page 160 from the IMPULSE C book claims that the hardware implementation is 10.6X faster than the software version. However, the results show the speedup is only 5.1X, as indicated by the final screen shot in Appendix D.

3DES Implementations

Two 3DES implementations were used for the system; one that is completely written in C++ and one that utilizes the configurable logic of the development systems with an interface written in C++. Both implementations use LAM-MPI, a communication protocol common among distributed computer clusters. The two implementations allow each algorithm’s runtime to be quantified and compared, thus is can be determined whether utilizing hardware is faster than software alone.

Software Based Implementation

The IMPULSE C project supplies C source code for the 3DES encryption algorithm. The C code was written by Phil Karn and is publicly available. To verify the code, a separate C++ implementation was developed that encrypted the 8 hexadecimal numbers from the application code within the XPS project. Once this code was validated, MPI was used to scatter the input data and gather the computational results. Node 0 was used to scatter the data. Node 0 transmits the first nth chunk of data to node 1 (if available) and subsequent chunks to higher numbered nodes until node 0 gets the last nth chunk of data. If the data available to node 0 is not divisible by 8 then the data is zero padded. Node 1 gathers all the partial solutions and writes them to a file since node 1 received the first chunk of data. If node 0 is the only node in use, node 0 reads and writes out all the data. The code is compiled using the command mpic++. To run the code use "mpirun –np <number of nodes used> <program> <file input><file output> 0". For testing, use "time" prior to the program call to display the lapsed time, e.g. "time mpirun –np <number of nodes used> <program><file input> <file output> 0". Appendix E contains the code for the 3DES software implementation.

Hardware Based Implementation

Applications that utilize the reconfigurable logic have three parts require development; the hardware structure, a device driver to interface the hardware structure with the operating system, and the application software.

Hardware. The hardware is the developed logic integrated into the system. This is referred to as a custom IP (imported peripheral) and can be developed via an assortment of tools. As previously mentioned, the IP was generated through IMPULSE C.

Redhat Linux was used to host the base system design; however CoDeveloper 2.1 was hosted on a Windows XP machine. The CoDeveloper 3.0 distribution for Redhat Linux does not contain the 3DES project, so it had to be moved from CoDeveloper 2.1 to CoDeveloper 3.0. Also, IMPULSE C on Linux machines is command-line based and does not possess a GUI. After the device is generated by IMPULSE C adding the IP takes a rescan of the user repositories, wiring the new IP to the PLB, and giving the added device an address. These steps are basic and are the same as the implemented standalone version in Appendix D; however Appendix F contains the method for simplicity as well as documenting the command line steps.

Device Driver. A character device driver was written to provide an interface with the hardware. This allows the device to be accessed as a file in the /dev directory tree (as /dev/des). The IOCTL template was utilized to generate the interface between the application code and the device. The resulting device driver contains 4 function calls that interface with the device to write the key, write data, initiate read, and read the data.

The development of the device driver required a basic understanding of the 3DES IP Core I/O. Fortunately, the standalone version implemented above gave insight into the communication with the device. Using the application code generated by IMPULSE C, one can work backwards and prune the code to determine the required read and write operations on the device. Much of the test application code was frivolous because the code implements two versions of the 3DES algorithm- one that is hardware based, and one that is software based. First, the code that is used for the software implementation was removed – such as the entire des_c function in des_sw.c code file. Next, the code that uses the opb_timer was removed, which was an obvious decision since it will not be used the device driver implementation. Overall, the code was reduced until only the interface C code to the IP Core was left. From here, IMPULSE C specific commands were replaced with the EDK C equivalents- this is accomplished by backtracking through the IMPULSE C include files. For example, HW_STREAM denotes a pointer to a register space. After these steps are done, the hardware I/O is evident and the hardware interface can be easily developed. The reduced application code only requires that the EDK defined data types be converted to a C format, e.g. Xuint32 is u32. From here a framework for an IOCTL device driver can be written.

Because the hardware was created from an existing project implementation and there was little detailed understanding of device driver functionality, debugging the device driver interface was difficult. The most significant problem was determining which functions forced the device to "close" when a task completion was signaled. For example, if the device had encrypted 8 bytes and sent a signal of completion, the device was no longer readable. While the device does not require notification of completed encryption, it does require notification that the encryption key is done writing to the device. This means that when the key is written to hardware it cannot be changed unless the hardware is reset. The device driver code and the reduced application code is listed in Appendix G.

User Application. The user application that uses the device driver is written in C++ and utilizes MPI. This program performs all file I/O and is similar to the software implementation. The data is scattered and gathered exactly the same as the software implementation but differs in that the algorithm is implemented in hardware. The code is compiled using the command mpic++. To run the code use “mpirun –np <number of nodes used> <program> <file input><file output>”. For testing purposes, the “time” system command is invoked prior to the program call to display the lapsed time, e.g. “time mpirun –np <number of nodes used> <program><file input> <file output>”. Appendix I contains the user application code.

Communication and File I/O Overhead Implementation

The third program was written to quantify the communication and file I/O overhead inherent in the system. This program calculates the overhead by scattering and gathering a test file among nodes which each in turn immediately sends this data back to the gathering node. Ultimately, this program reads a given file and outputs the same file under a given name. Just like the 3DES implementations, it does not calculate timing within the program but is called with the system call “time” prior to the program call, e.g. “time mpirun –np <number of nodes used> <program> <file input> <file output>”. Appendix J contains the code.

Runtime

From the standalone 3DES implementation, where the hardware implementation ran 5.1X faster than the software implementation, it is clear that the MPI 3DES hardware implementation should run faster. However, communication and file overhead will impact the speed of both implementations. In addition, the device driver will slow the hardware implementation runtime execution. These obstacles make it clear that the hardware implementation may not run as originally hoped. The runtimes will be discussed in Chapter 6.

CHAPTER SIX

Findings and Discussions

This chapter discusses the runtimes of the developed 3DES and communication overhead applications. Recorded runtimes should clearly demonstrate which 3DES implementation is faster, which will indicate whether reconfigurable logic is viable for our specific application. The runtime for the non-3DES application will give an estimate of the communication and file I/O overhead for the 3DES implementations. Testing is split into two phases: the first phase uses 15 nodes and varies the input file for encryption and the second phase uses a 40 MB file for encryption but varies the number of nodes used. Phase one will demonstrate which algorithm is faster when varying the size of the input data file, and phase two will determine the speedup when more nodes are utilized by each application. Each application was run 20 times so that the impact of timing variations could be reduced. There are 12 unencrypted input file sizes such that each implementation was run 240 times for phase one, totaling 720 runs. Phase two uses 15 nodes so each implementation is executed a total of 300 times, totaling 900 runs. The unencrypted input file sizes can be broken up into 2 distinct groups: a small file size range and a large file size range [table 1].

Table 1: File Sizes

Small File Sizes	14 KB	27 KB	54 KB	108 KB	216 KB	433 KB	866 KB
Large File Sizes	10 MB	20 MB	40 MB	80 MB	160 MB		

Table 2: Phase One Results Using 15 Boards

File Size (MB)	Average Runtime 3DES Without Hardware (sec)	Variance of 3DES Without Hardware (sec)	Average Runtime 3DES With Hardware (sec)	Variance of 3DES With Hardware (sec)	Average Runtime Overhead (sec)	Variance of Overhead (sec)
.014	4.63	0.90	4.98	1.10	4.67	1.29
.027	4.81	1.17	4.63	0.87	4.96	1.15
.054	5.14	1.05	4.76	1.22	5.17	1.04
.108	5.51	1.34	4.97	0.97	4.79	0.91
.216	6.06	1.30	5.69	1.14	5.85	1.26
.433	6.92	1.58	6.14	1.21	6.10	1.10
.866	7.71	1.85	9.07	1.14	7.52	2.05
10	51.92	2.97	45.23	2.84	27.83	1.50
20	98.97	3.98	83.32	3.25	58.88	3.17
40	191.20	3.71	162.10	2.57	114.95	2.84
80	373.61	3.32	320.73	2.57	218.56	5.22
160	745.56	4.92	630.90	2.62	429.81	5.10

Table 3: Phase Two Results, 40 MB Unencrypted Input File Size

Number of Boards	Average Runtime 3DES Without Hardware (sec)	Variance of 3DES Without Hardware (sec)	Average Runtime 3DES with Hardware (sec)	Variance of 3DES With Hardware (sec)	Average Runtime Overhead (sec)	Variance of Overhead (sec)
1	847.28	3.33	425.58	12.16	136.41	2.42
2	426.27	2.81	240.37	1.86	127.17	2.25
3	378.54	2.76	215.83	2.70	120.31	2.98
4	319.40	2.51	207.11	4.67	119.63	2.38
5	284.40	1.82	202.35	2.41	116.56	3.16
6	260.39	1.78	192.46	2.90	115.75	3.87
7	243.13	1.95	184.70	2.49	115.43	3.00
8	230.52	2.56	179.10	2.82	115.11	3.02
9	220.77	2.65	176.03	3.00	115.60	3.60
10	213.13	3.24	172.49	3.36	115.00	1.71
11	206.91	2.02	169.20	2.65	114.48	2.73
12	201.82	3.03	165.93	1.84	113.85	2.12
13	199.67	8.17	164.31	2.61	115.18	2.70
14	193.94	2.49	163.12	2.51	114.01	3.78
15	191.20	3.71	162.10	2.57	114.33	2.38

Tables 2 and 3 make it apparent that that the 3DES application that utilizes the reconfigurable logic runs faster. However, the hardware implementation does not run as theoretically possible. A pure hardware implementation runs 10.6 times faster than a pure software implementation on a single board [17, pg160]; hence this research expected better performance across a parallel implementation. Unfortunately, the communication and file I/O overhead has a significant impact on runtime. Figure 7 displays phase one runtimes on large files sizes. The graph clearly shows that communication and file I/O overhead comprises more than half of runtime (purple line, non-3DES).

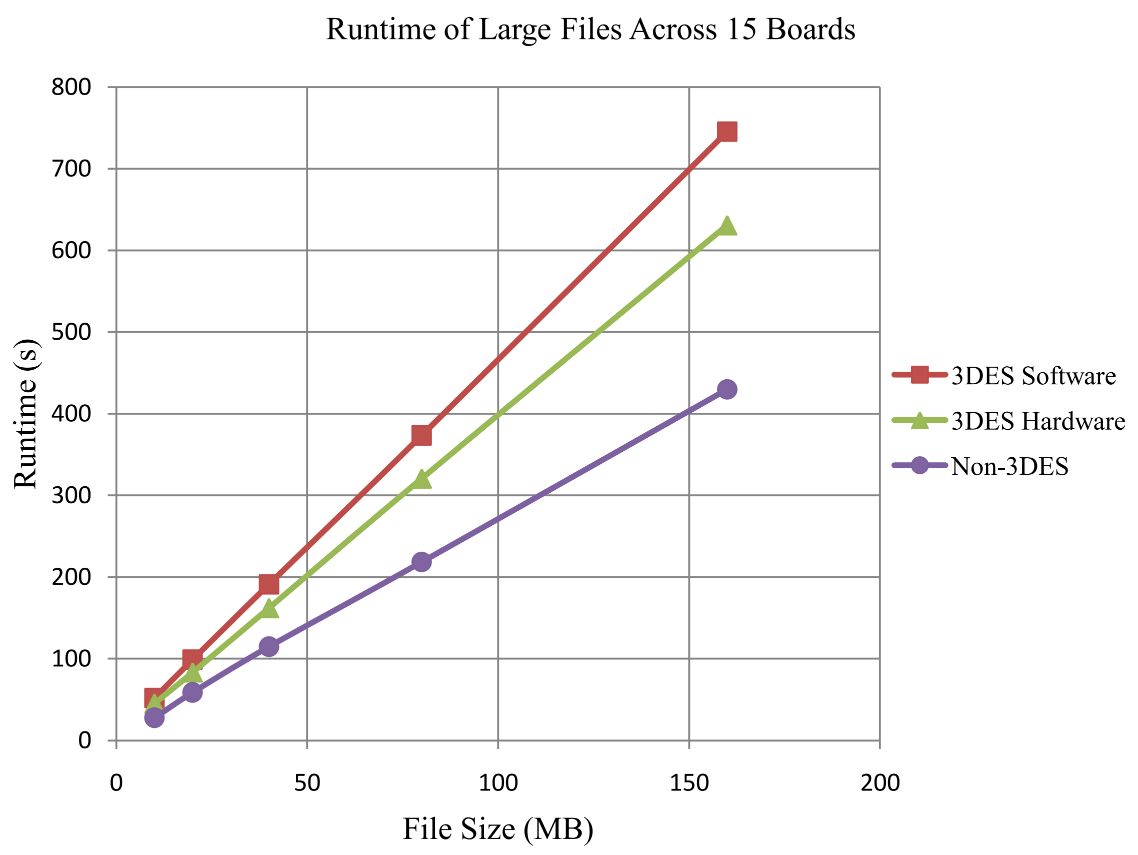


Figure 7: Runtime Versus Unencrypted Input File Size

In an ideal setting communication would be instantaneous, meaning that using 4 boards versus 1 board would result in a 4x speedup. Phase two shows that this type of speedup does not occur when increasing the number of boards, but begins to level off. The nearly constant value of the non-3DES timing (purple line) implies that speedup is limited by file I/O. Initially, network communication was thought to limit runtime, however the non-3DES data for a single node (left most purple data point) in Figure 8 shows a runtime consistent with mutiple nodes.

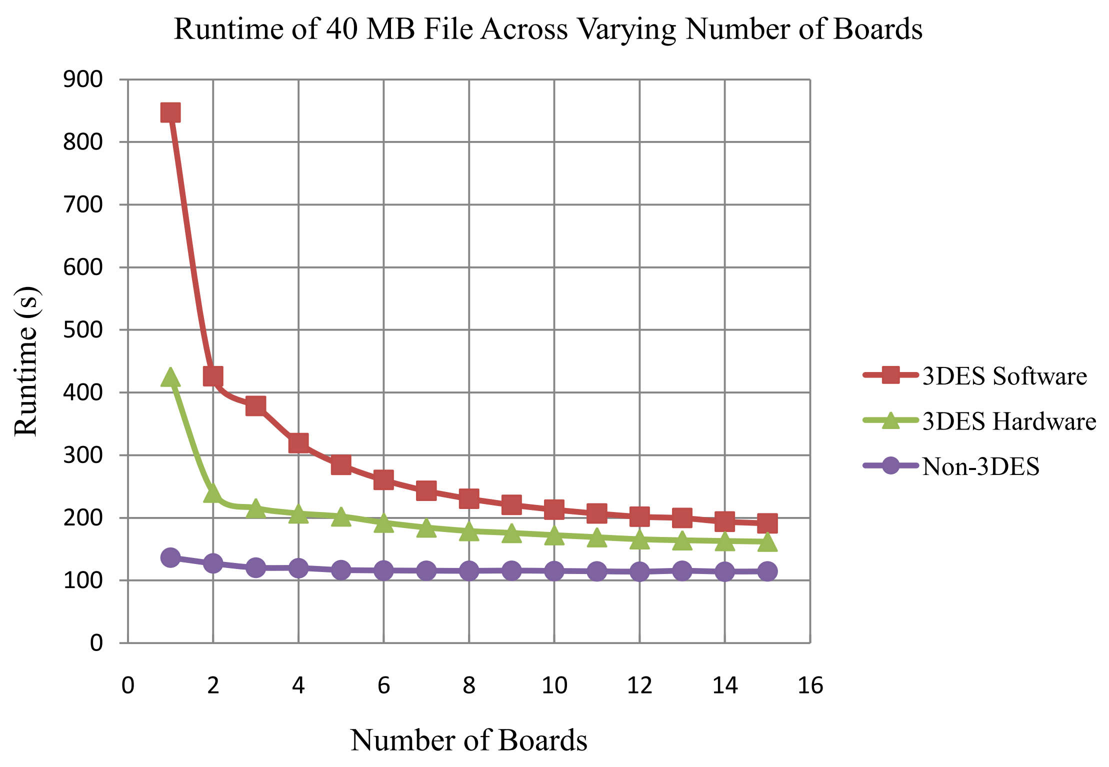


Figure 8: Runtime Versus Nodes Used

CHAPTER SEVEN

Conclusions and Final Recommendations

Lessons Learned

File I/O was the limiting factor for application speedup; this was the result of the CF's read and write time being non-trivial in application execution. File I/O overhead can be reduced by 2 methods. First, the unencrypted data and encrypted solution can be stored in RAM. This method would prevent any read and write calls to the CF within the application, thus circumventing the current file I/O overhead for testing. However, this requires a separate application to collect the solution from RAM if the encrypted data needs to be stored. The second method is to use a network storage device with faster read and write times for file I/O; this allows the unencrypted data file and solution to be stored. Unfortunately, this method may introduce non-trivial communication overhead because of network congestion.

Conclusions

Overall, the construction of Baylor's Reconfigurable Computer Cluster was a qualified success. In reference to the outlined goals of Chapter 1, the only goal not met was that pertaining to the minimal "learning curve" usage metric. Currently, for an application developer to fully utilize the cluster he/she must develop an expertise in the use of Xilinx tools, C++, MPI, and device driver construction. The system clearly meets the other goals.

The reconfigurable logic alone has shown speed up over its software counterpart via IMPULSE C. This thesis' implementations of the 3DES algorithms confirm that using the reconfigurable logic results in improved computational speedup. Unfortunately, the 3DES algorithm's runtime was over twice that of the theoretical speed from the IMPULSE C results [17, pg 160]. However, the current hardware implementation is not the fastest possible. IMPULSE C has a faster hardware version that has a speedup of 425 over the IMPULSE C created software [17, pg 207]; unfortunately this version could not be implemented because of time constraints. Despite the results, there exist algorithms for this cluster that would most likely show greater speedup than the 3DES algorithm implementation.

Recommendations

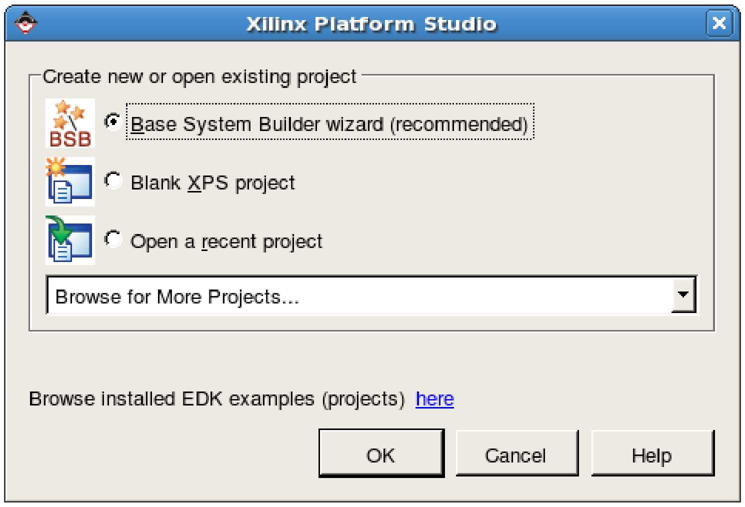
The first recommendation would be to implement a faster technique for file I/O such as a RAM-disk or network based file storage; otherwise application runtime will continue to be limited by file I/O overhead. The second recommendation would be to dynamically load designed logic onto the board. This can be done via ICAP, and is supported by the kernel from the Xilinx GIT tree. This would reduce the time it takes to configure and boot the cluster. The third recommendation is to upgrade the nodes in the cluster to Virtex 5 development boards. The XUP V2P boards are currently legacy products and support is being discontinued. New boards would offer new features for further research.

APPENDICES

APPENDIX A

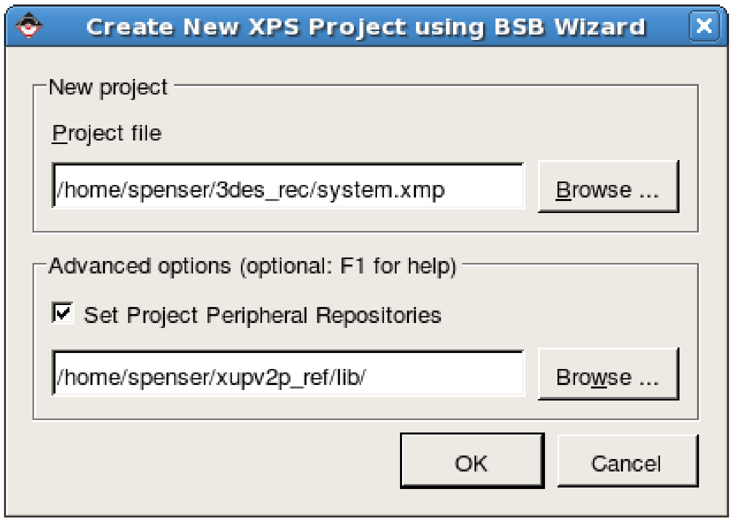
XPS Creation of Hardware Base

This appendix gives step by step instructions in creating the hardware base of our boards in XPS. The instructions assume you are using EDK 9.1i and have the board support package downloaded (http://www.digilentinc.com/Data/Products/XUPV2P/EDK-XUP-V2ProPack.zip). Screenshots are given instead of explicit instructions, which should be easy enough to follow for those familiar with XPS.


Xilinx Platform Studio
Create new or open existing project
BSB
Base System Builder wizard (recommended)
Blank XPS project
Open a recent project
Browse for More Projects...
Browse installed EDK examples (projects) here
OK
Cancel
Help

Base System Builder wizard (recommended) [✓]

When you start XPS you are given the option to create a new project or open an existing one. You will want to create a new project, as depicted above.



Create New XPS Project using BSB Wizard

New project

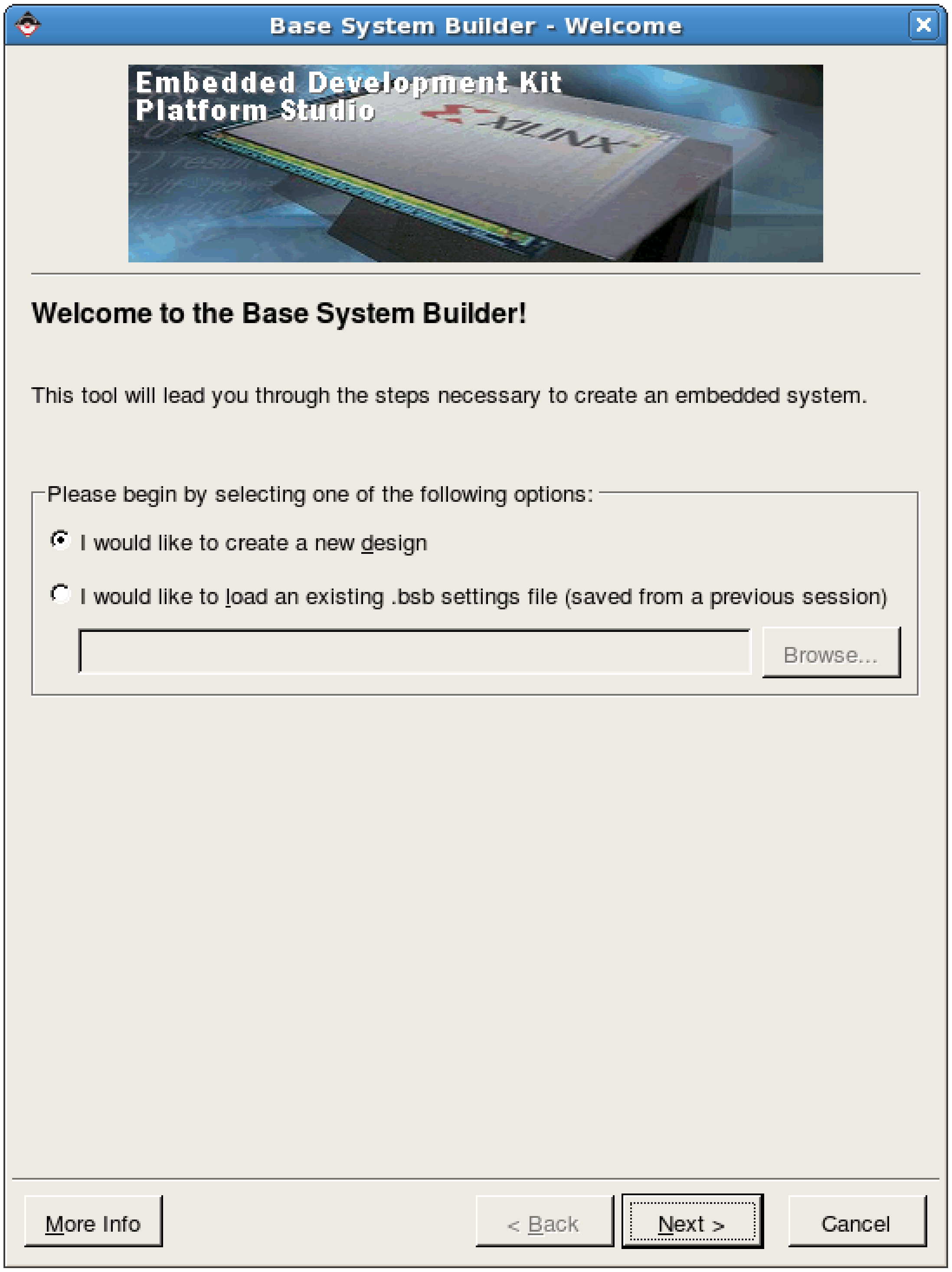
Project file: /home/spenser/3des_rec/system.xmp

Advanced options (optional: F1 for help)

[✓] Set Project Peripheral Repositories

/home/spenser/xupv2p_ref/lib/

You can name the project anything you choose, but make sure you are pointing to where ever you board support package is as shown above.



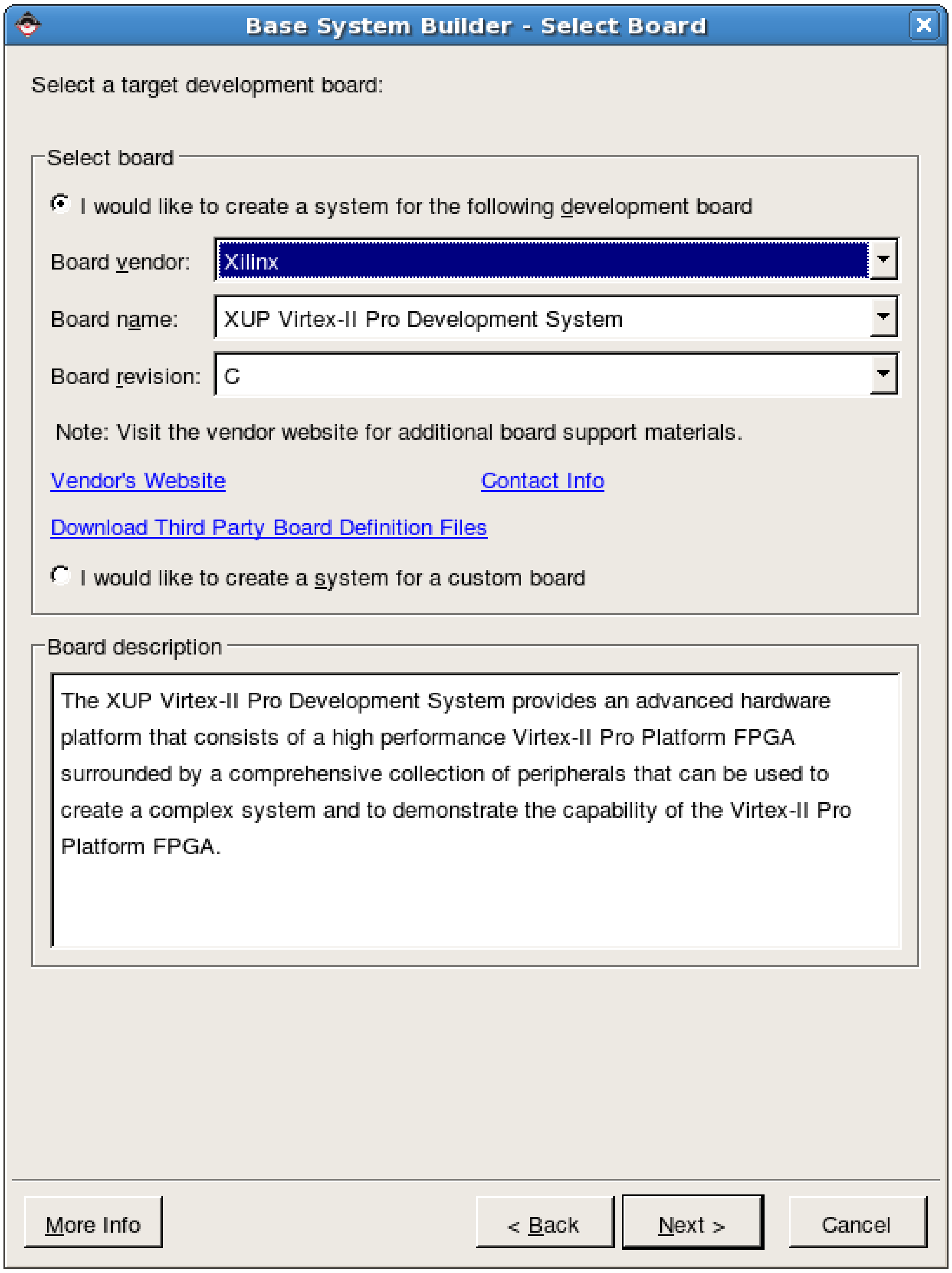
Welcome to the Base System Builder!

This tool will lead you through the steps necessary to create an embedded system.

Please begin by selecting one of the following options:

[✓ I would like to create a new design]

I would like to load an existing .bsb settings file (saved from a previous session)


Base System Builder - Select Board

Select a target development board:

Select board

[✓ I would like to create a system for the following development board]

Board vendor: Xilinx

Board name: XUP Virtex-II Pro Development System

Board revision: C

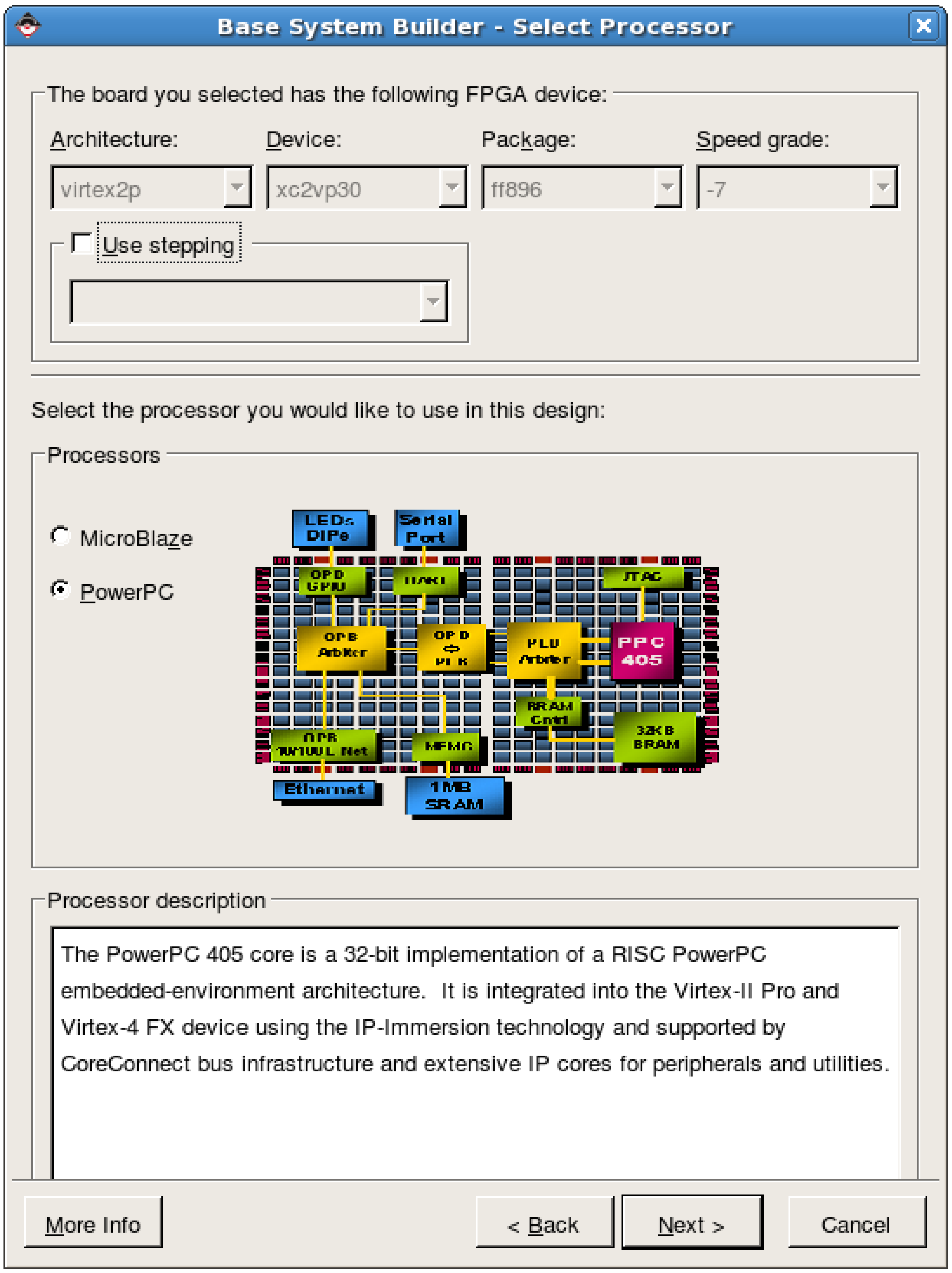
Note: Visit the vendor website for additional board support materials.

Vendor's Website
Contact Info
Download Third Party Board Definition Files

I would like to create a system for a custom board

Board description

The XUP Virtex-II Pro Development System provides an advanced hardware platform that consists of a high performance Virtex-II Pro Platform FPGA surrounded by a comprehensive collection of peripherals that can be used to create a complex system and to demonstrate the capability of the Virtex-II Pro Platform FPGA.



Base System Builder - Select Processor

The board you selected has the following FPGA device:

Architecture:	Device:	Package:	Speed grade:
virtex2p	xc2vp30	ff896	-7

☐ Use stepping

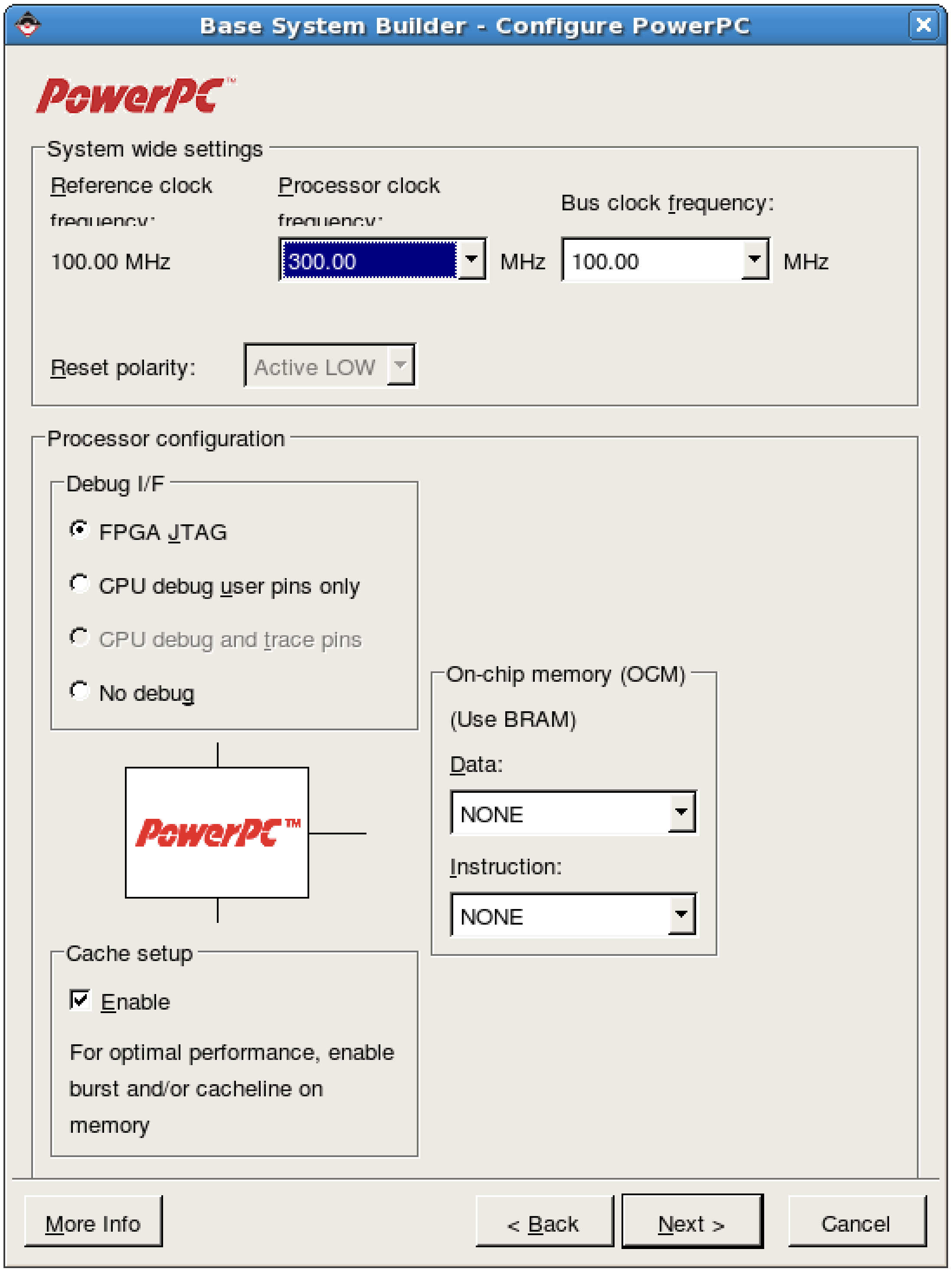
Select the processor you would like to use in this design:

Processors

MicroBlaze [✓ PowerPC]

Processor description

The PowerPC 405 core is a 32-bit implementation of a RISC PowerPC embedded-environment architecture. It is integrated into the Virtex-II Pro and Virtex-4 FX device using the IP-Immersion technology and supported by CoreConnect bus infrastructure and extensive IP cores for peripherals and utilities.



Base System Builder - Configure PowerPC

System wide settings

Reference clock frequency: 100.00 MHz

Processor clock frequency: 300.00 MHz

Bus clock frequency: 100.00 MHz

Reset polarity: Active LOW

Processor configuration

Debug I/F

- [✓ FPGA JTAG]
- CPU debug user pins only
- CPU debug and trace pins
- No debug

On-chip memory (OCM) (Use BRAM)

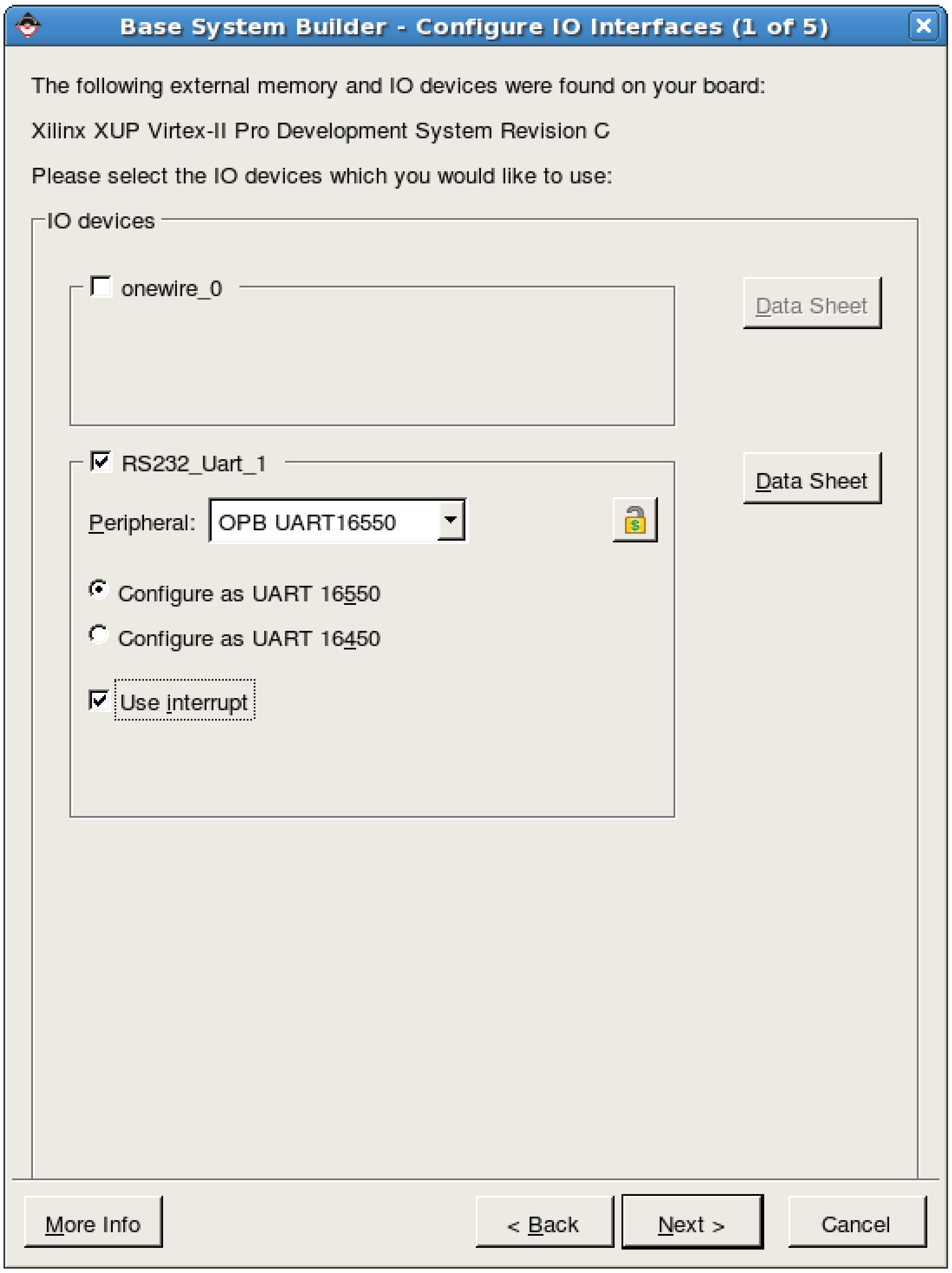
Data: NONE

Instruction: NONE

Cache setup

- [✓] Enable

For optimal performance, enable burst and/or cacheline on memory



The following external memory and IO devices were found on your board:

Xilinx XUP Virtex-II Pro Development System Revision C

Please select the IO devices which you would like to use:

IO devices

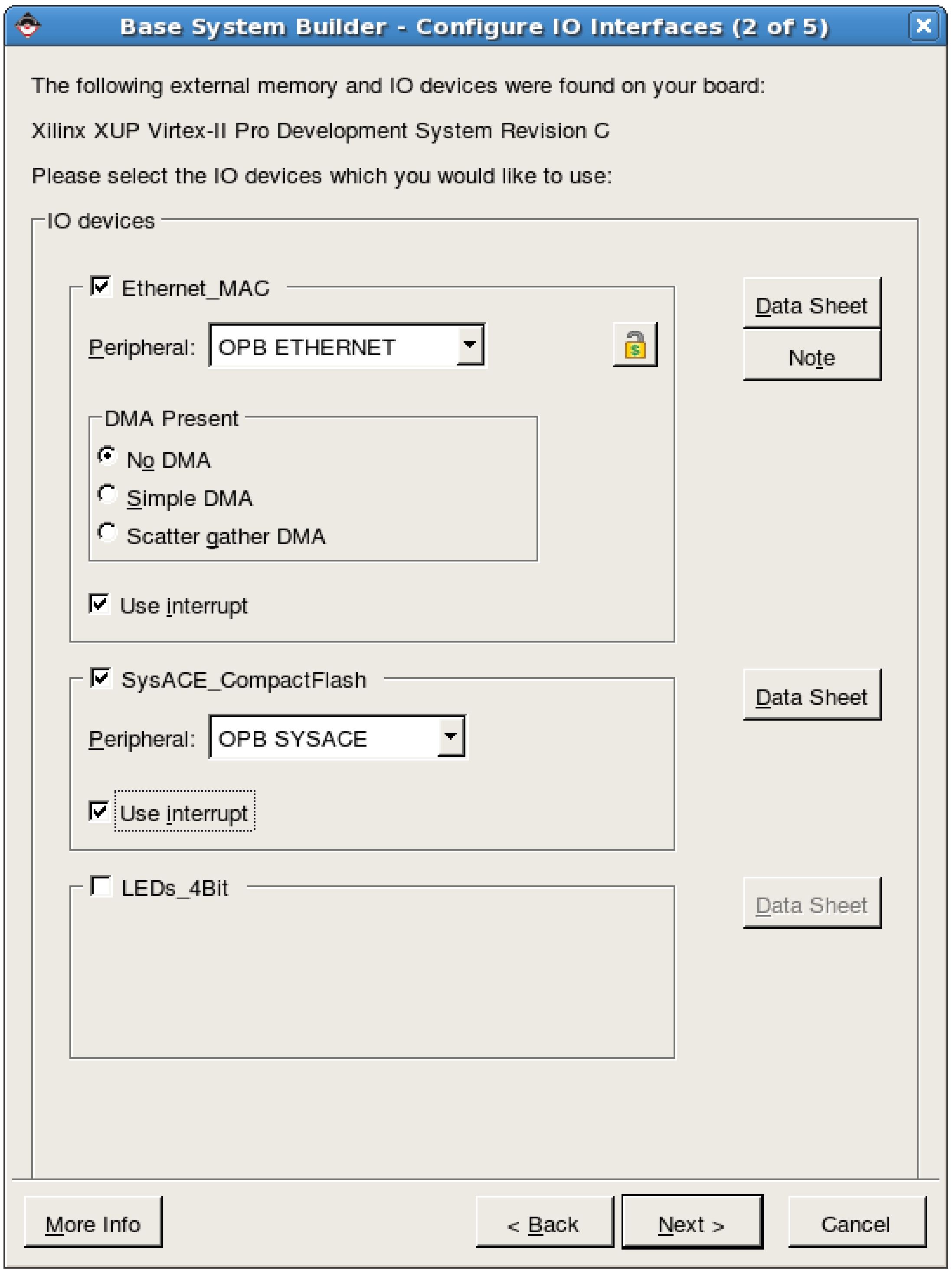
☐ onewire_0

☑ RS232_Uart_1

Peripheral: OPB UART16550

[✓ Configure as UART 16550] Configure as UART 16450

☑ Use interrupt



The following external memory and IO devices were found on your board:

Xilinx XUP Virtex-II Pro Development System Revision C

Please select the IO devices which you would like to use:

IO devices

[✓] Ethernet_MAC

Peripheral: OPB ETHERNET

DMA Present

[✓ No DMA] Simple DMA Scatter gather DMA

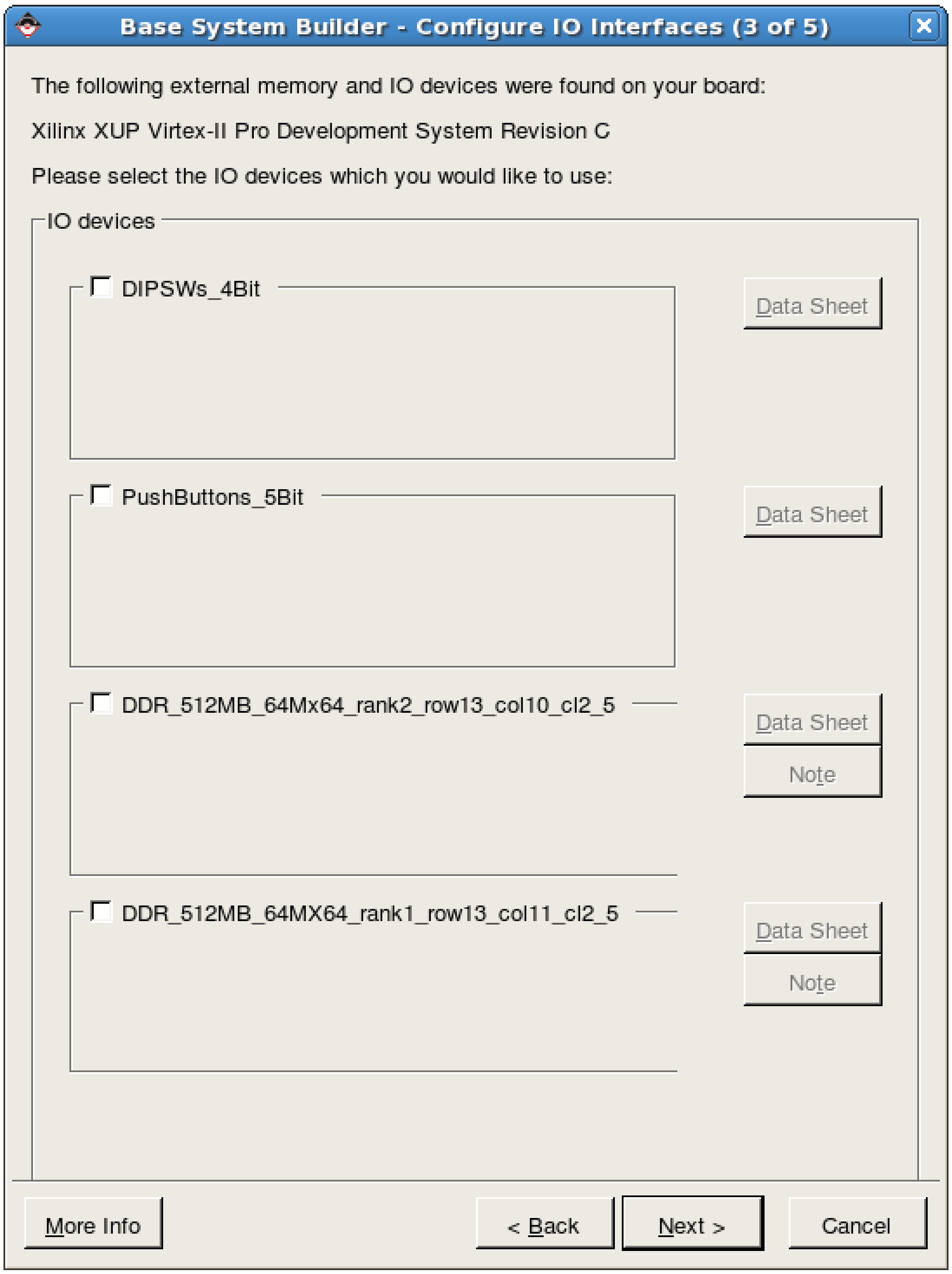
[✓] Use interrupt

[✓] SysACE_CompactFlash

Peripheral: OPB SYSACE

[✓] Use interrupt

[ ] LEDs_4Bit



Base System Builder - Configure IO Interfaces (3 of 5)

The following external memory and IO devices were found on your board:

Xilinx XUP Virtex-II Pro Development System Revision C

Please select the IO devices which you would like to use:

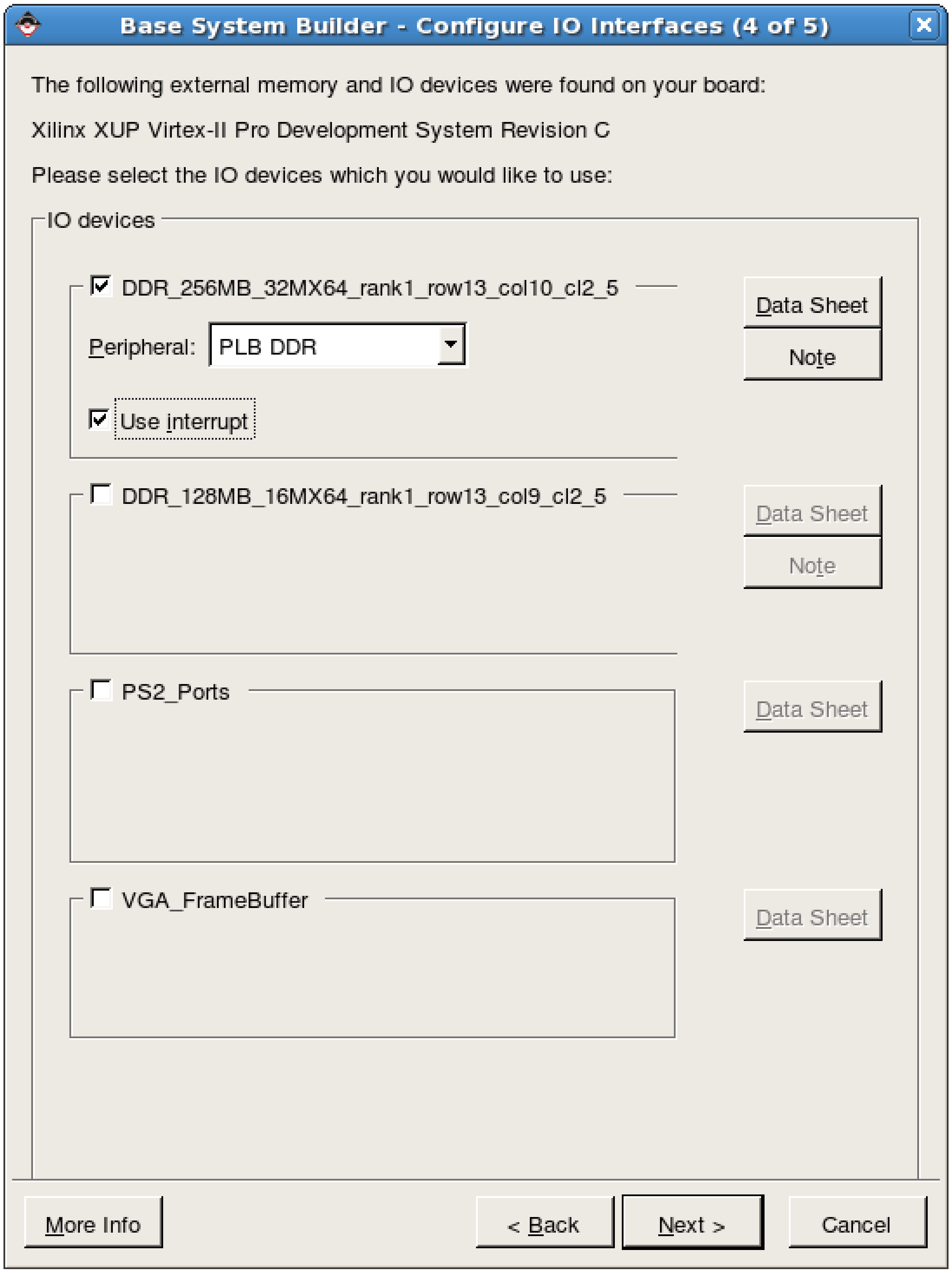
IO devices

☐ DIPSWs_4Bit

☐ PushButtons_5Bit

☐ DDR_512MB_64Mx64_rank2_row13_col10_cl2_5

☐ DDR_512MB_64MX64_rank1_row13_col11_cl2_5


Base System Builder - Configure IO Interfaces (4 of 5)

The following external memory and IO devices were found on your board:

Xilinx XUP Virtex-II Pro Development System Revision C

Please select the IO devices which you would like to use:

IO devices

[✓] DDR_256MB_32MX64_rank1_row13_col10_cl2_5

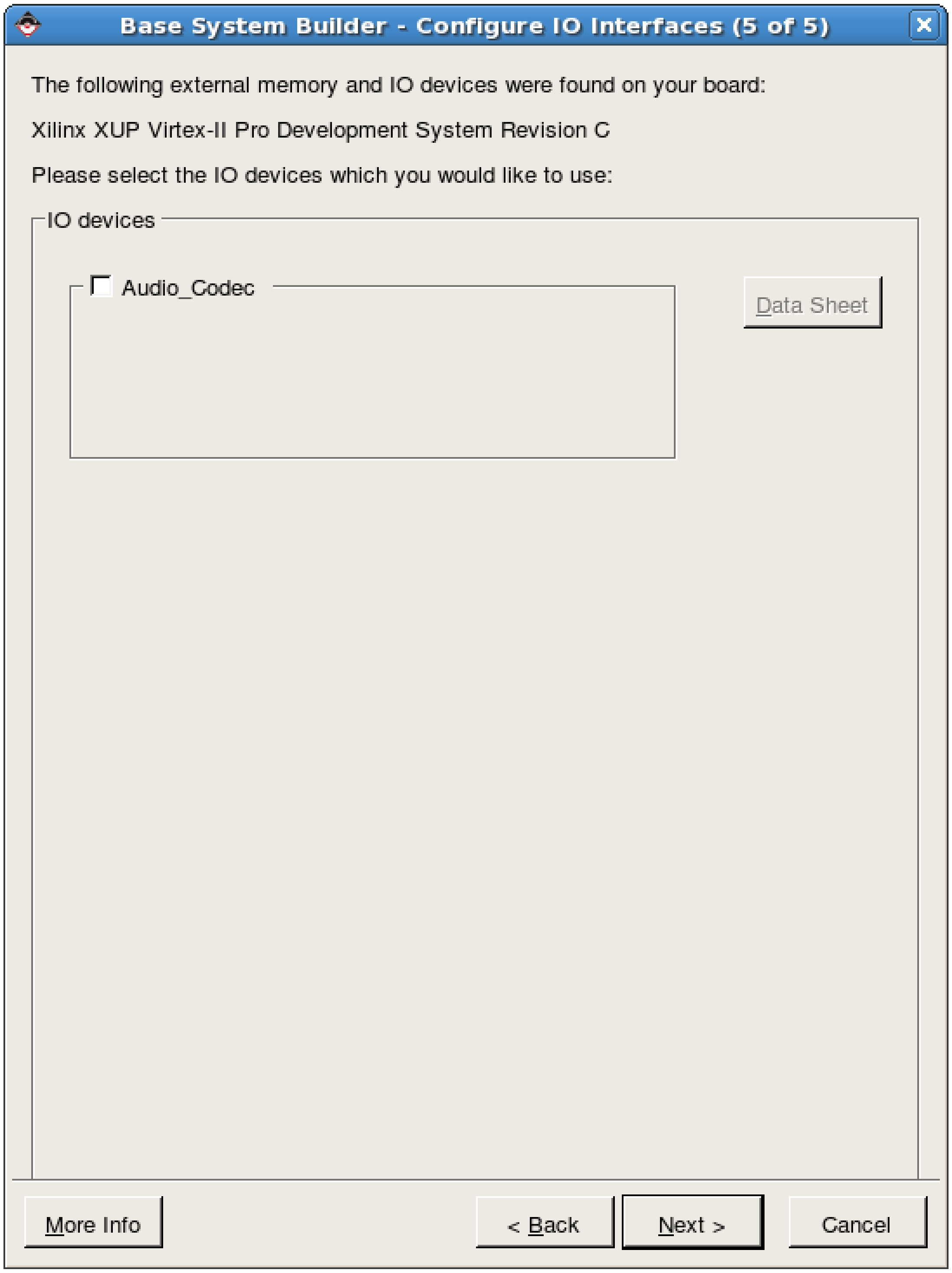
Peripheral: PLB DDR

[✓] Use interrupt

[ ] DDR_128MB_16MX64_rank1_row13_col9_cl2_5

[ ] PS2_Ports

[ ] VGA_FrameBuffer



Base System Builder - Configure IO Interfaces (5 of 5)

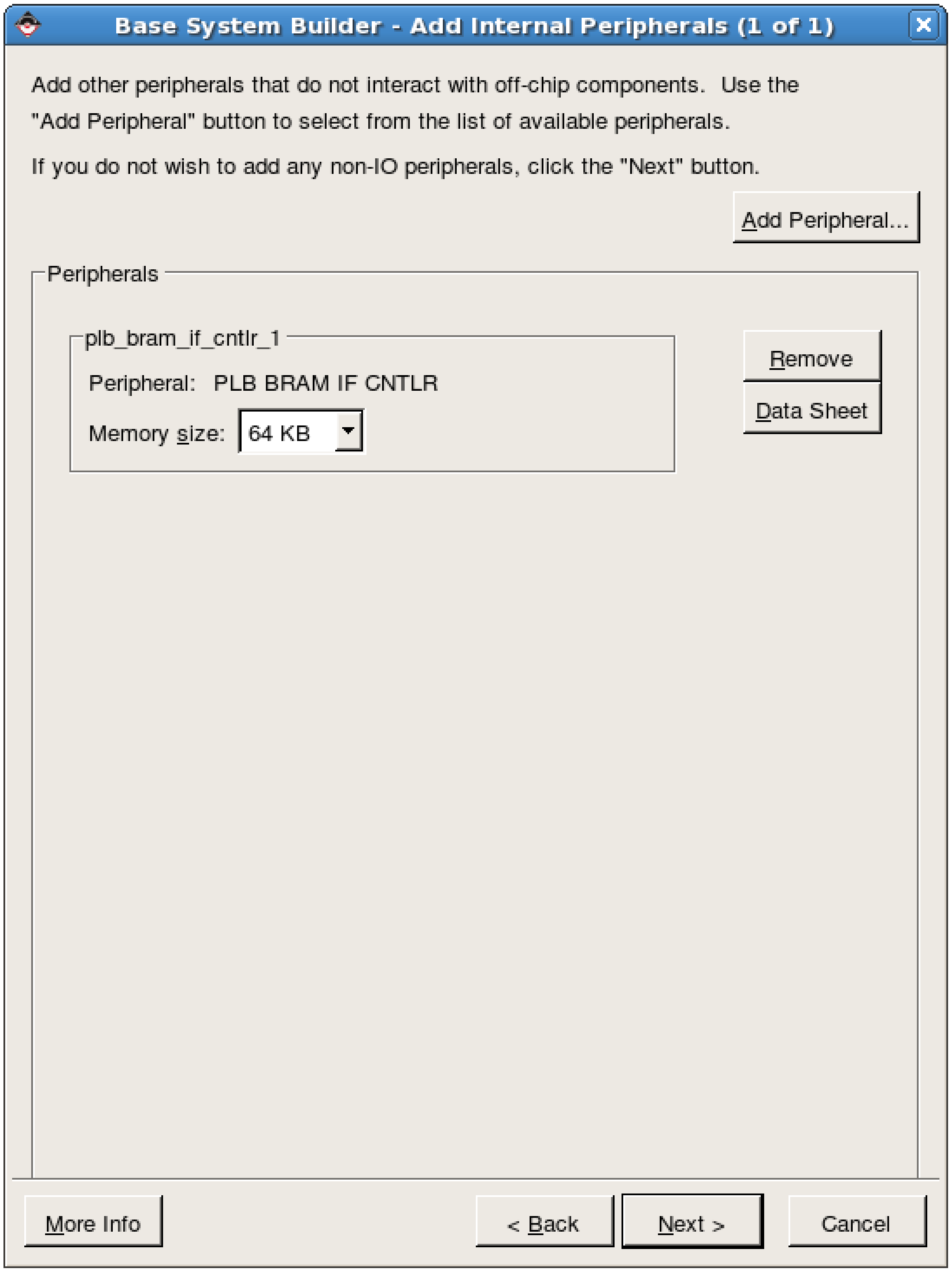
The following external memory and IO devices were found on your board:

Xilinx XUP Virtex-II Pro Development System Revision C

Please select the IO devices which you would like to use:

IO devices

☐ Audio_Codec


Base System Builder - Add Internal Peripherals (1 of 1)

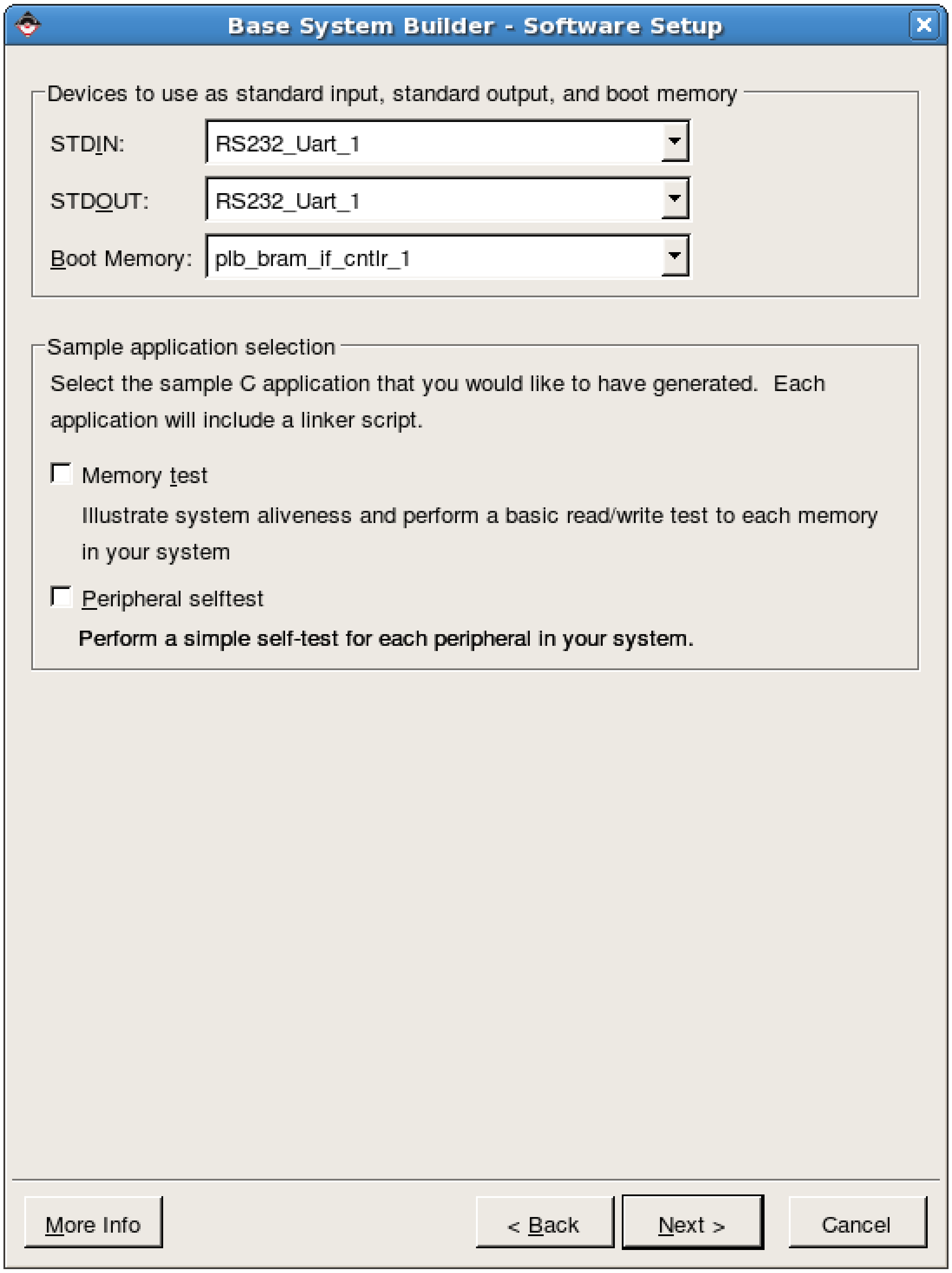
Add other peripherals that do not interact with off-chip components. Use the "Add Peripheral" button to select from the list of available peripherals.

If you do not wish to add any non-IO peripherals, click the "Next" button.

plb_bram_if_cntlr_1

Peripheral: PLB BRAM IF CNTLR

Memory size: 64 KB


Base System Builder - Software Setup

Devices to use as standard input, standard output, and boot memory

STDIN: RS232_Uart_1

STDOUT: RS232_Uart_1

Boot Memory: plb_bram_if_cntlr_1

Sample application selection

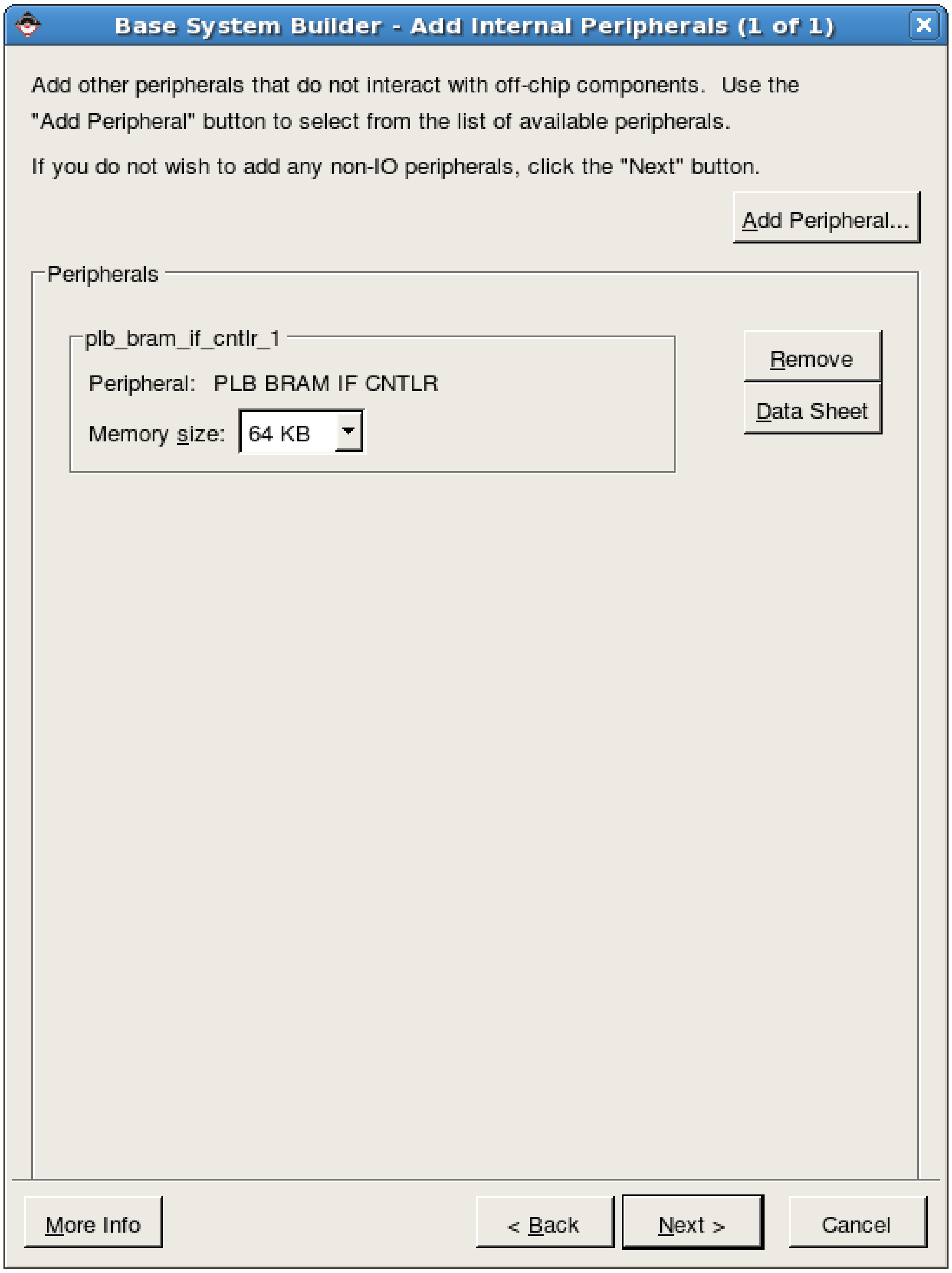
Select the sample C application that you would like to have generated. Each application will include a linker script.

☐ Memory test

Illustrate system aliveness and perform a basic read/write test to each memory in your system

☐ Peripheral selftest

Perform a simple self-test for each peripheral in your system.

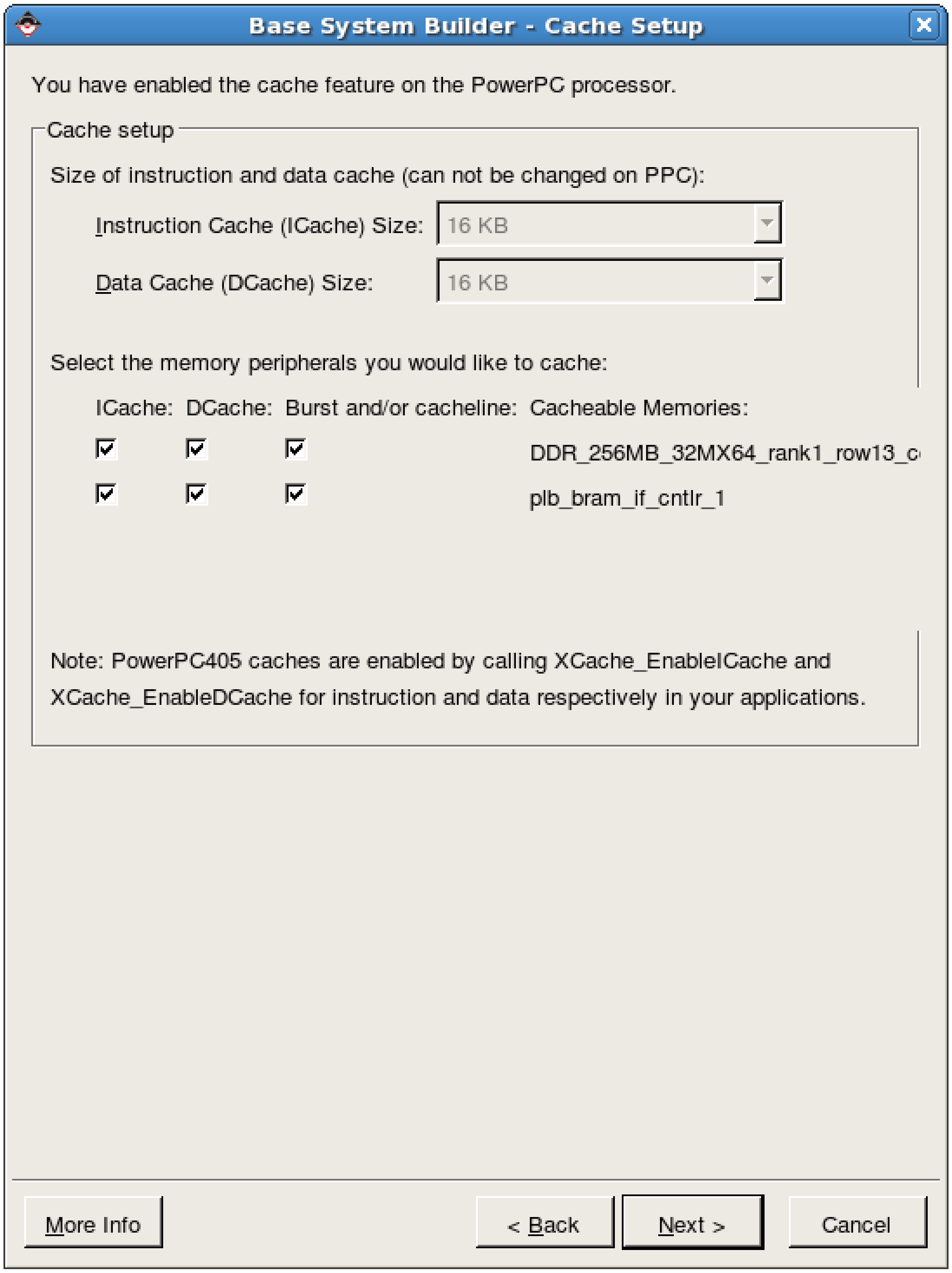

Base System Builder - Add Internal Peripherals (1 of 1)

Add other peripherals that do not interact with off-chip components. Use the "Add Peripheral" button to select from the list of available peripherals.

If you do not wish to add any non-IO peripherals, click the "Next" button.

Peripheral: PLB BRAM IF CNTLR

Memory size: 64 KB


Base System Builder - Cache Setup

You have enabled the cache feature on the PowerPC processor.

Cache setup

Size of instruction and data cache (can not be changed on PPC):

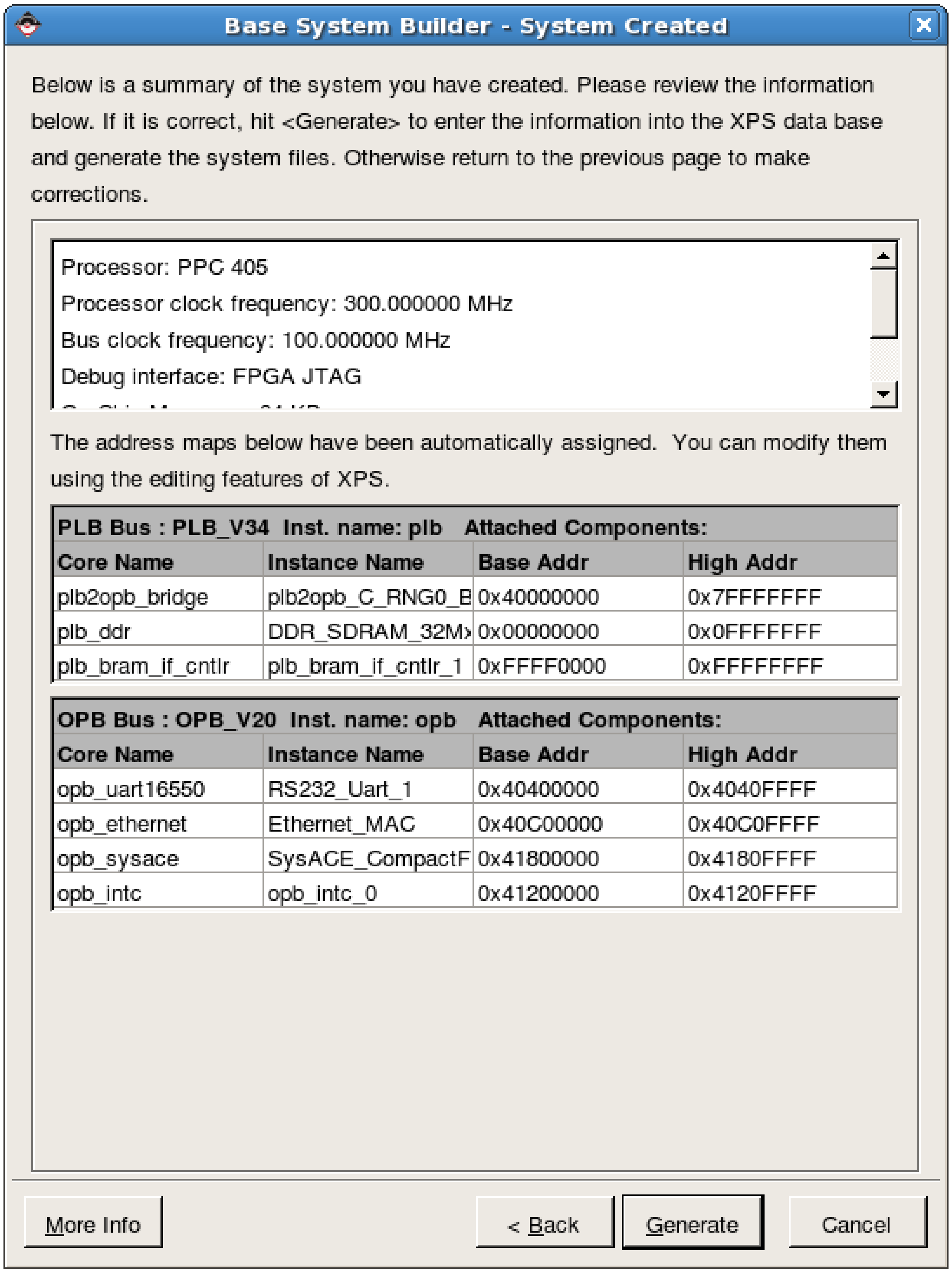
Instruction Cache (ICache) Size: 16 KB

Data Cache (DCache) Size: 16 KB

Select the memory peripherals you would like to cache:

ICache:	DCache:	Burst and/or cacheline:	Cacheable Memories:
✓	✓	✓	DDR_256MB_32MX64_rank1_row13_c
✓	✓	✓	plb_bram_if_cntlr_1

Note: PowerPC405 caches are enabled by calling XCache_EnableICache and XCache_EnableDCache for instruction and data respectively in your applications.



Below is a summary of the system you have created. Please review the information below. If it is correct, hit <Generate> to enter the information into the XPS data base and generate the system files. Otherwise return to the previous page to make corrections.

Processor: PPC 405
Processor clock frequency: 300.000000 MHz
Bus clock frequency: 100.000000 MHz
Debug interface: FPGA JTAG

The address maps below have been automatically assigned. You can modify them using the editing features of XPS.

PLB Bus : PLB_V34 Inst. name: plb Attached Components:			
Core Name	**Instance Name**	**Base Addr**	**High Addr**
plb2opb_bridge	plb2opb_C_RNG0_B	0x40000000	0x7FFFFFFF
plb_ddr	DDR_SDRAM_32M)	0x00000000	0x0FFFFFFF
plb_bram_if_cntlr	plb_bram_if_cntlr_1	0xFFFF0000	0xFFFFFFFF

OPB Bus : OPB_V20 Inst. name: opb Attached Components:			
Core Name	**Instance Name**	**Base Addr**	**High Addr**
opb_uart16550	RS232_Uart_1	0x40400000	0x4040FFFF
opb_ethernet	Ethernet_MAC	0x40C00000	0x40C0FFFF
opb_sysace	SysACE_CompactF	0x41800000	0x4180FFFF
opb_intc	opb_intc_0	0x41200000	0x4120FFFF

This concludes the construction of the hardware base in XPS.

APPENDIX B

Porting the OS to the XUP V2P

This appendix gives step by step instructions in porting Linux to the Virtex II Pro boards. This appendix assumes that you have already completed Appendix A, that you are using a x86 PC running Red Hat 5.2, and are using EDK 9.1i.

Installing the Cross Compiler

Step 1: Get the development tools necessary for the Cross Compiler.

In a terminal window type:

```
sudo yum groupinstall 'Development Tools'
```

Step 2: Now we need to setup a directory for the cross compiler to install into the default is /opt/crosstool and you should replace USER_NAME with your actual username.

In a terminal window type:

```
sudo mkdir /opt/crosstool/  # create the crosstool directory
sudo chown -r USER_NAME /opt/crosstool
chmod u+rwx /opt/crosstool/
chmod a+rx /opt/crosstool/
```

Step 3: Download and Install. These commands will download, unpack the crosstool script, and start the build process for a powerpc-405 cross compiler.

In a terminal window type:

```
wget http://www.kegel.com/crosstool/crosstool-0.43.tar.gz
tar xzf crosstool-0.43.tar.gz
cd crosstool-0.43
./demo-powerpc-405.sh
```

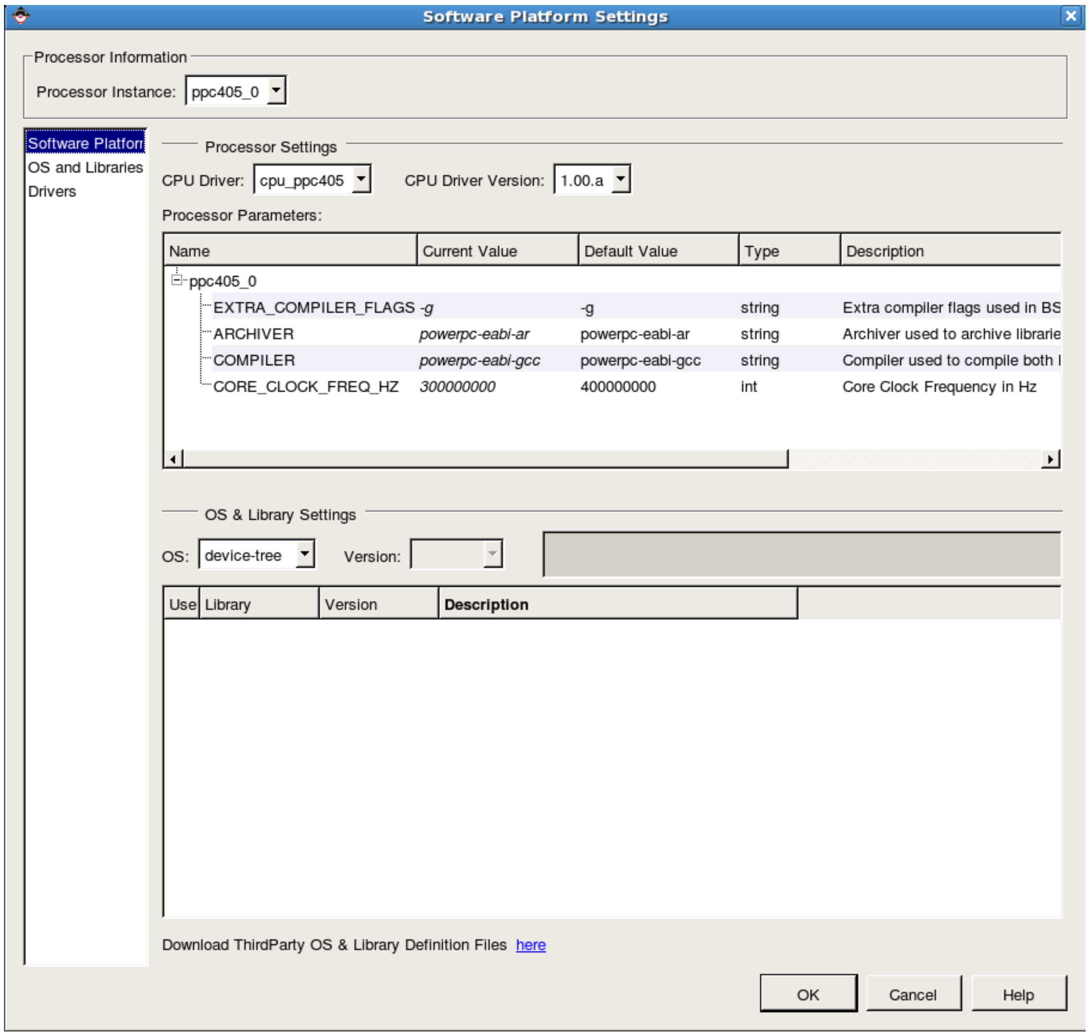
Step 4: Make it easier to use. Right now in order to use the cross compiler you'd have type an insanely long path name. However, by adding two lines to your ~/.bashrc you can cross-compile just as easy as native compiling.

In a terminal window type:

```
vi  ~/.bashrc
# PowerPC Cross Compiler aliases
export PATH=$PATH:/opt/crosstool/gcc-4.1.0-glibc-2.3.6/powerpc-405-linux-gnu/bin
alias ppckmake="make ARCH=powerpc CROSS_COMPILE=powerpc-405-linux-gnu-"
alias ppcmake="make CC=powerpc-405-linux-gnu-gcc"
```

Incorporating the OS with the system

Here you will need to go back to XPS and open your project. You need to go to the software settings and change the data to match the following screen shots.



Software Platform Settings

Processor Information

Processor Instance: ppc405_0

Software Platform
OS and Libraries
Drivers

Processor Settings

CPU Driver: cpu_ppc405 CPU Driver Version: 1.00.a

Processor Parameters:

Name	Current Value	Default Value	Type	Description
ppc405_0				
EXTRA_COMPILER_FLAGS	-g	-g	string	Extra compiler flags used in BS
ARCHIVER	powerpc-eabi-ar	powerpc-eabi-ar	string	Archiver used to archive librarie
COMPILER	powerpc-eabi-gcc	powerpc-eabi-gcc	string	Compiler used to compile both l
CORE_CLOCK_FREQ_HZ	300000000	400000000	int	Core Clock Frequency in Hz

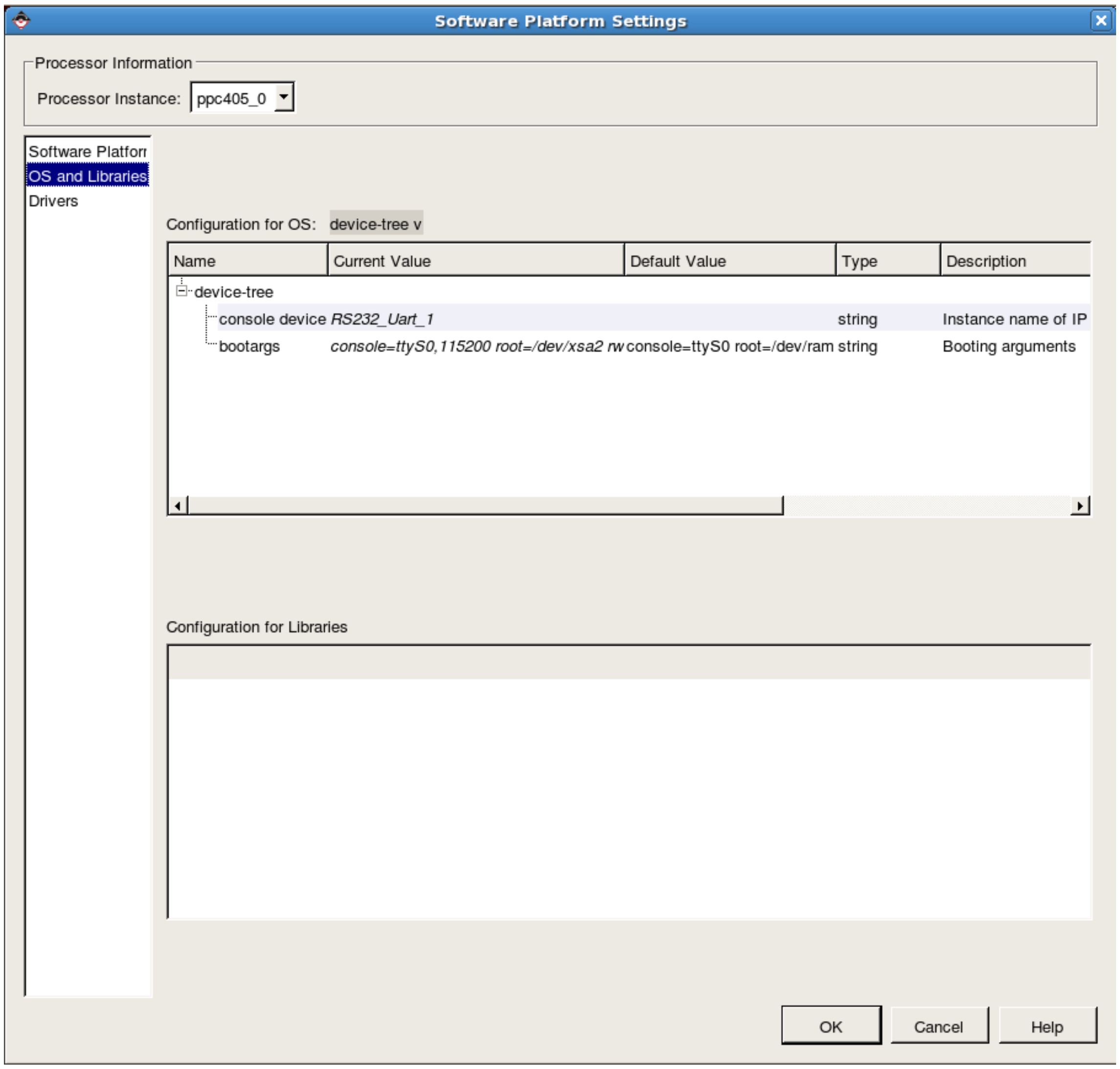
OS & Library Settings

OS: device-tree Version:

Use	Library	Version	Description

Download ThirdParty OS & Library Definition Files here

OK Cancel Help



Processor Instance: ppc405_0

Configuration for OS: device-tree v

Name	Current Value	Default Value	Type	Description
device-tree				
console device	RS232_Uart_1		string	Instance name of IP
bootargs	console=ttyS0,115200 root=/dev/xsa2 rw	console=ttyS0 root=/dev/ram	string	Booting arguments

Notice the our bootargs are set to console=/dev/ttyS0,115200 root=/dev/xsa2 rw <— this little “rw” addition will save you tons of headache when you are trying to understand why you can’t write to the disk.

Here are two tables that describe the bootargs.

device type	console text
xps_uart16550	console=ttyS*N*
xps_uartlite	console=ttyUL*N*
N is 0 for the first device of the type. *N* is 1 for the second device of the type etc.	

root filesystem location	bootarg options
ramdisk	root=/dev/ram
compact flash disk	root=/dev/xsaN (where *N* is the partition number of the root file system.)
nfs	root=/dev/nfs nfsroot=<nfs server>:<nfs share>,tcp (e.g. root=/dev/nfs nfsroot=192.168.1.1:/nfsroots/development,tcp)

Git and Setup Linux

We will be using the open-source Linux version from Xilinx. Change to your project directory.

In a terminal window type:

```
cd <project-directory>
git clone git://git.xilinx.com/linux-2.6-xlnx.git
cd linux-2.6-xlnx/arch/powerpc/boot/dts
ln -s ../../../../../ppc405_0/libsrc/device-tree/xilinx.dts virtex405-xupv2p.dts
cd <project-directory>
ln -s linux-2.6-xlnx/arch/powerpc/boot/simpleImage.virtex405-xupv2p.elf .
```

Generate a bit file

We now need to generate a bitfile and download it to the board.

Generate libraries and BSPs.

In EDK click

Hardware -> Download Bitstream

Software -> Generate Libraries and BSPs

Creating a Debian RFS on Compact Flash

Creating a Debian RFS on a CF card is a two stage process. The first stage must be done on a standard x86 workstation but the second stage must be completed on the board itself.

On the card we will need two partitions one for the ACE file and one where we will place the RFS. We have mostly 2 GB CF cards so they were partitioned them as follows.

Stage One:

Partition sizes are:

1 128 MB Ace File

2 1.9 GB Debian Root File System

Partitioning using parted, mkdosfs and mkfs.ext2

The command below partitions the drive as described above.

```
parted -s $CF_DEV rm 1 # Remove old partitions
parted -s $CF_DEV rm 2
parted -s $CF_DEV mkpart primary fat16 0 128 # Create ACE Partition
parted -s $CF_DEV mkpart primary ext2 128 2048 # Create RFS Partition
mkdosfs -F 16 -R1 $CF_PART1
mkfs.ext2 $CF_PART2
```

Bootstrap using Debootstrap

In this next part will bootstrap a basic Debian system. Make sure you have internet access before attempting this part.

The first step is to mount your RFS partition. I choose to mount it to /mnt because it is common place to mount file systems.

```
mount $CF_PART2 /mnt
```

Next using debootstrap generate an absolutely minimal Debian Etch RFS on the CF card.

```
debootstrap --arch powerpc --foreign etch /mnt http://ftp.debian.org/debian
```

Stage Two: Finish Bootstrap

At this point you have a bare minimal RFS for Debian. In order to make this a fully featured RFS you will need to boot the board using XMD or the System ACE.

There is only one special condition for booting this minimal RFS.

In Software->Software Platform And Settings select the OS and Libraries option and then append /bin/bash to the bootargs.

Once booted run this command to finish the bootstraping procedure

./debootstrap/debootstrap —second-stage

You now have a basic Debian RFS you can add more packages with apt-get and task-sel.

I recommend running task-sel standard to install some basic necessities.

Create a Linux Kernel

Go into the linux-2.6-xlnx directory and run the following within a terminal:

```
ppckmake 40x/virtex4_defconfig #
ppckmake menuconfig
ppckmake simpleImage.virtex405-xupv2p
```

The particulars of our menuconfig can be found in Appendix C.

Testing on JTAG

Start XMD and load the kernel into memory using JTAG.

```
xmd
connect ppc hw
dow simpleImage.virtex405-xupv2p.elf
run
```

On the serial you should see Linux start to boot. From here you can create an ACE file to boot off of.

Create ACE File

In the Xilinx command shell type:

```
xmd –tcl genace.tcl –opt genace.opt
```

Where genace.opt contains:

-jprog

```
-board xupv2p

-target ppc_hw

-hw implementation/download.bit

-elf simpleImage.virtex405-xupv2p.elf

-ace system.ace
```

APPENDIX C

The Menuconfig Options

This appendix has our Menuconfig options, which we need to specify kernel abilities.

```
#
# Automatically generated make config: don't edit
# Linux kernel version: 2.6.28
# Wed Apr 29 14:31:28 2009
#
# CONFIG_PPC64 is not set

#
# Processor support
#
# CONFIG_6xx is not set
# CONFIG_PPC_85xx is not set
# CONFIG_PPC_8xx is not set
CONFIG_40x=y
# CONFIG_44x is not set
# CONFIG_E200 is not set
CONFIG_4xx=y
CONFIG_PPC_MMU_NOHASH=y
# CONFIG_PPC_MM_SLICES is not set
CONFIG_NOT_COHERENT_CACHE=y
CONFIG_PPC32=y
CONFIG_WORD_SIZE=32
# CONFIG_ARCH_PHYS_ADDR_T_64BIT is not set
CONFIG_MMU=y
CONFIG_GENERIC_CMOS_UPDATE=y
CONFIG_GENERIC_TIME=y
CONFIG_GENERIC_TIME_VSYSCALL=y
CONFIG_GENERIC_CLOCKEVENTS=y
CONFIG_GENERIC_HARDIRQS=y
```

```
# CONFIG_HAVE_SETUP_PER_CPU_AREA is not set
CONFIG_IRQ_PER_CPU=y
CONFIG_STACKTRACE_SUPPORT=y
CONFIG_HAVE_LATENCYTOP_SUPPORT=y
CONFIG_LOCKDEP_SUPPORT=y
CONFIG_RWSEM_XCHGADD_ALGORITHM=y
CONFIG_ARCH_HAS_ILOG2_U32=y
CONFIG_GENERIC_HWEIGHT=y
CONFIG_GENERIC_CALIBRATE_DELAY=y
CONFIG_GENERIC_FIND_NEXT_BIT=y
# CONFIG_ARCH_NO_VIRT_TO_BUS is not set
CONFIG_PPC=y
CONFIG_EARLY_PRINTK=y
CONFIG_GENERIC_NVRAM=y
CONFIG_SCHED_OMIT_FRAME_POINTER=y
CONFIG_ARCH_MAY_HAVE_PC_FDC=y
CONFIG_PPC_OF=y
CONFIG_OF=y
CONFIG_PPC_UDBG_16550=y
# CONFIG_GENERIC_TBSYNC is not set
CONFIG_AUDIT_ARCH=y
CONFIG_GENERIC_BUG=y
# CONFIG_DEFAULT_UIMAGE is not set
CONFIG_PPC_DCR_NATIVE=y
CONFIG_PPC_DCR_MMIO=y
CONFIG_PPC_DCR=y
CONFIG_DEFCONFIG_LIST="/lib/modules/$UNAME_RELEASE/.config"

#
# General setup
#
# CONFIG_EXPERIMENTAL is not set
CONFIG_BROKEN_ON_SMP=y
CONFIG_LOCK_KERNEL=y
CONFIG_INIT_ENV_ARG_LIMIT=32
CONFIG_LOCALVERSION=""
# CONFIG_LOCALVERSION_AUTO is not set
CONFIG_SWAP=y
CONFIG_SYSVIPC=y
CONFIG_SYSVIPC_SYSCTL=y
```

```
# CONFIG_BSD_PROCESS_ACCT is not set
# CONFIG_TASKSTATS is not set
# CONFIG_AUDIT is not set
CONFIG_IKCONFIG=y
CONFIG_IKCONFIG_PROC=y
CONFIG_LOG_BUF_SHIFT=14
# CONFIG_CGROUPS is not set
CONFIG_SYSFS_DEPRECATED=y
CONFIG_SYSFS_DEPRECATED_V2=y
# CONFIG_RELAY is not set
CONFIG_NAMESPACES=y
# CONFIG_UTS_NS is not set
# CONFIG_IPC_NS is not set
# CONFIG_BLK_DEV_INITRD is not set
# CONFIG_CC_OPTIMIZE_FOR_SIZE is not set
CONFIG_SYSCTL=y
# CONFIG_EMBEDDED is not set
CONFIG_SYSCTL_SYSCALL=y
CONFIG_KALLSYMS=y
# CONFIG_KALLSYMS_EXTRA_PASS is not set
CONFIG_HOTPLUG=y
CONFIG_PRINTK=y
CONFIG_BUG=y
CONFIG_ELF_CORE=y
CONFIG_COMPAT_BRK=y
CONFIG_BASE_FULL=y
CONFIG_FUTEX=y
CONFIG_ANON_INODES=y
CONFIG_EPOLL=y
CONFIG_SIGNALFD=y
CONFIG_TIMERFD=y
CONFIG_EVENTFD=y
CONFIG_SHMEM=y
CONFIG_AIO=y
CONFIG_VM_EVENT_COUNTERS=y
CONFIG_SLAB=y
# CONFIG_SLUB is not set
# CONFIG_SLOB is not set
# CONFIG_PROFILING is not set
CONFIG_HAVE_OPROFILE=y
```

```
# CONFIG_KPROBES is not set
CONFIG_HAVE_EFFICIENT_UNALIGNED_ACCESS=y
CONFIG_HAVE_IOREMAP_PROT=y
CONFIG_HAVE_KPROBES=y
CONFIG_HAVE_KRETPROBES=y
CONFIG_HAVE_ARCH_TRACEHOOK=y
# CONFIG_HAVE_GENERIC_DMA_COHERENT is not set
CONFIG_SLABINFO=y
CONFIG_RT_MUTEXES=y
# CONFIG_TINY_SHMEM is not set
CONFIG_BASE_SMALL=0
CONFIG_MODULES=y
# CONFIG_MODULE_FORCE_LOAD is not set
CONFIG_MODULE_UNLOAD=y
# CONFIG_MODVERSIONS is not set
# CONFIG_MODULE_SRCVERSION_ALL is not set
CONFIG_KMOD=y
CONFIG_BLOCK=y
# CONFIG_LBD is not set
# CONFIG_BLK_DEV_IO_TRACE is not set
# CONFIG_LSF is not set
# CONFIG_BLK_DEV_INTEGRITY is not set

#
# IO Schedulers
#
CONFIG_IOSCHED_NOOP=y
CONFIG_IOSCHED_AS=y
CONFIG_IOSCHED_DEADLINE=y
CONFIG_IOSCHED_CFQ=y
# CONFIG_DEFAULT_AS is not set
# CONFIG_DEFAULT_DEADLINE is not set
CONFIG_DEFAULT_CFQ=y
# CONFIG_DEFAULT_NOOP is not set
CONFIG_DEFAULT_IOSCHED="cfq"
CONFIG_CLASSIC_RCU=y
# CONFIG_FREEZER is not set

#
# Platform support
```

```
#
# CONFIG_PPC_CELL is not set
# CONFIG_PPC_CELL_NATIVE is not set
# CONFIG_PQ2ADS is not set
# CONFIG_PPC4xx_GPIO is not set
# CONFIG_ACADIA is not set
# CONFIG_EP405 is not set
# CONFIG_HCU4 is not set
# CONFIG_KILAUEA is not set
# CONFIG_MAKALU is not set
# CONFIG_WALNUT is not set
CONFIG_XILINX_VIRTEX_GENERIC_BOARD=y
# CONFIG_PPC40x_SIMPLE is not set
CONFIG_XILINX_VIRTEX_II_PRO=y
CONFIG_XILINX_VIRTEX_4_FX=y
CONFIG_IBM405_ERR77=y
CONFIG_IBM405_ERR51=y
# CONFIG_IPIC is not set
# CONFIG_MPIC is not set
# CONFIG_MPIC_WEIRD is not set
# CONFIG_PPC_I8259 is not set
# CONFIG_PPC_RTAS is not set
# CONFIG_MMIO_NVRAM is not set
# CONFIG_PPC_MPC106 is not set
# CONFIG_PPC_970_NAP is not set
# CONFIG_PPC_INDIRECT_IO is not set
# CONFIG_GENERIC_IOMAP is not set
# CONFIG_CPU_FREQ is not set
# CONFIG_FSL_ULI1575 is not set
CONFIG_XILINX_VIRTEX=y

#
# Kernel options
#
# CONFIG_HIGHMEM is not set
# CONFIG_NO_HZ is not set
# CONFIG_HIGH_RES_TIMERS is not set
CONFIG_GENERIC_CLOCKEVENTS_BUILD=y
# CONFIG_HZ_100 is not set
CONFIG_HZ_250=y
```

```
# CONFIG_HZ_300 is not set
# CONFIG_HZ_1000 is not set
CONFIG_HZ=250
# CONFIG_SCHED_HRTICK is not set
# CONFIG_PREEMPT_NONE is not set
# CONFIG_PREEMPT_VOLUNTARY is not set
CONFIG_PREEMPT=y
# CONFIG_PREEMPT_RCU is not set
CONFIG_BINFMT_ELF=y
# CONFIG_CORE_DUMP_DEFAULT_ELF_HEADERS is not set
# CONFIG_HAVE_AOUT is not set
# CONFIG_BINFMT_MISC is not set
CONFIG_MATH_EMULATION=y
# CONFIG_IOMMU_HELPER is not set
CONFIG_PPC_NEED_DMA_SYNC_OPS=y
CONFIG_ARCH_ENABLE_MEMORY_HOTPLUG=y
CONFIG_ARCH_HAS_WALK_MEMORY=y
CONFIG_ARCH_ENABLE_MEMORY_HOTREMOVE=y
CONFIG_ARCH_FLATMEM_ENABLE=y
CONFIG_ARCH_POPULATES_NODE_MAP=y
CONFIG_FLATMEM=y
CONFIG_FLAT_NODE_MEM_MAP=y
CONFIG_PAGEFLAGS_EXTENDED=y
CONFIG_SPLIT_PTLOCK_CPUS=4
CONFIG_MIGRATION=y
# CONFIG_RESOURCES_64BIT is not set
# CONFIG_PHYS_ADDR_T_64BIT is not set
CONFIG_ZONE_DMA_FLAG=1
CONFIG_BOUNCE=y
CONFIG_VIRT_TO_BUS=y
CONFIG_UNEVICTABLE_LRU=y
CONFIG_PPC_4K_PAGES=y
# CONFIG_PPC_16K_PAGES is not set
# CONFIG_PPC_64K_PAGES is not set
CONFIG_FORCE_MAX_ZONEORDER=11
CONFIG_PROC_DEVICETREE=y
CONFIG_CMDLINE_BOOL=y
CONFIG_CMDLINE=""
CONFIG_EXTRA_TARGETS="simpleImage.virtex405-xupv2p"
# CONFIG_PM is not set
```

```
CONFIG_SECCOMP=y
# CONFIG_COMPRESSED_DEVICE_TREE is not set
CONFIG_ISA_DMA_API=y

#
# Bus options
#
CONFIG_ZONE_DMA=y
CONFIG_4xx_SOC=y
CONFIG_PPC_PCI_CHOICE=y
# CONFIG_PCI is not set
# CONFIG_PCI_DOMAINS is not set
# CONFIG_PCI_SYSCALL is not set
# CONFIG_ARCH_SUPPORTS_MSI is not set
# CONFIG_PCCARD is not set
# CONFIG_HAS_RAPIDIO is not set

#
# Advanced setup
#
# CONFIG_ADVANCED_OPTIONS is not set

#
# Default settings for advanced configuration options are used
#
CONFIG_LOWMEM_SIZE=0x30000000
CONFIG_PAGE_OFFSET=0xc0000000
CONFIG_KERNEL_START=0xc0000000
CONFIG_PHYSICAL_START=0x00000000
CONFIG_TASK_SIZE=0xc0000000
CONFIG_CONSISTENT_START=0xff100000
CONFIG_CONSISTENT_SIZE=0x00200000
CONFIG_NET=y

#
# Networking options
#
CONFIG_COMPAT_NET_DEV_OPS=y
CONFIG_PACKET=y
# CONFIG_PACKET_MMAP is not set
```

```
CONFIG_UNIX=y
CONFIG_XFRM=y
# CONFIG_XFRM_USER is not set
# CONFIG_NET_KEY is not set
CONFIG_INET=y
CONFIG_IP_MULTICAST=y
# CONFIG_IP_ADVANCED_ROUTER is not set
CONFIG_IP_FIB_HASH=y
CONFIG_IP_PNP=y
CONFIG_IP_PNP_DHCP=y
CONFIG_IP_PNP_BOOTP=y
# CONFIG_IP_PNP_RARP is not set
# CONFIG_NET_IPIP is not set
# CONFIG_NET_IPGRE is not set
# CONFIG_IP_MROUTE is not set
# CONFIG_SYN_COOKIES is not set
# CONFIG_INET_AH is not set
# CONFIG_INET_ESP is not set
# CONFIG_INET_IPCOMP is not set
# CONFIG_INET_XFRM_TUNNEL is not set
CONFIG_INET_TUNNEL=y
CONFIG_INET_XFRM_MODE_TRANSPORT=y
CONFIG_INET_XFRM_MODE_TUNNEL=y
CONFIG_INET_XFRM_MODE_BEET=y
# CONFIG_INET_LRO is not set
CONFIG_INET_DIAG=y
CONFIG_INET_TCP_DIAG=y
# CONFIG_TCP_CONG_ADVANCED is not set
CONFIG_TCP_CONG_CUBIC=y
CONFIG_DEFAULT_TCP_CONG="cubic"
CONFIG_IPV6=y
# CONFIG_IPV6_PRIVACY is not set
# CONFIG_IPV6_ROUTER_PREF is not set
# CONFIG_INET6_AH is not set
# CONFIG_INET6_ESP is not set
# CONFIG_INET6_IPCOMP is not set
# CONFIG_INET6_XFRM_TUNNEL is not set
# CONFIG_INET6_TUNNEL is not set
CONFIG_INET6_XFRM_MODE_TRANSPORT=y
CONFIG_INET6_XFRM_MODE_TUNNEL=y
```

```
CONFIG_INET6_XFRM_MODE_BEET=y
CONFIG_IPV6_SIT=y
CONFIG_IPV6_NDISC_NODETYPE=y
# CONFIG_IPV6_TUNNEL is not set
# CONFIG_NETWORK_SECMARK is not set
CONFIG_NETFILTER=y
# CONFIG_NETFILTER_DEBUG is not set
CONFIG_NETFILTER_ADVANCED=y

#
# Core Netfilter Configuration
#
# CONFIG_NETFILTER_NETLINK_QUEUE is not set
# CONFIG_NETFILTER_NETLINK_LOG is not set
# CONFIG_NF_CONNTRACK is not set
CONFIG_NETFILTER_XTABLES=y
# CONFIG_NETFILTER_XT_TARGET_CLASSIFY is not set
# CONFIG_NETFILTER_XT_TARGET_DSCP is not set
# CONFIG_NETFILTER_XT_TARGET_MARK is not set
# CONFIG_NETFILTER_XT_TARGET_NFLOG is not set
# CONFIG_NETFILTER_XT_TARGET_NFQUEUE is not set
# CONFIG_NETFILTER_XT_TARGET_RATEEST is not set
# CONFIG_NETFILTER_XT_TARGET_TCPMSS is not set
# CONFIG_NETFILTER_XT_MATCH_COMMENT is not set
# CONFIG_NETFILTER_XT_MATCH_DCCP is not set
# CONFIG_NETFILTER_XT_MATCH_DSCP is not set
# CONFIG_NETFILTER_XT_MATCH_ESP is not set
# CONFIG_NETFILTER_XT_MATCH_HASHLIMIT is not set
# CONFIG_NETFILTER_XT_MATCH_IPRANGE is not set
# CONFIG_NETFILTER_XT_MATCH_LENGTH is not set
# CONFIG_NETFILTER_XT_MATCH_LIMIT is not set
# CONFIG_NETFILTER_XT_MATCH_MAC is not set
# CONFIG_NETFILTER_XT_MATCH_MARK is not set
# CONFIG_NETFILTER_XT_MATCH_MULTIPORT is not set
# CONFIG_NETFILTER_XT_MATCH_OWNER is not set
# CONFIG_NETFILTER_XT_MATCH_POLICY is not set
# CONFIG_NETFILTER_XT_MATCH_PKTTYPE is not set
# CONFIG_NETFILTER_XT_MATCH_QUOTA is not set
# CONFIG_NETFILTER_XT_MATCH_RATEEST is not set
# CONFIG_NETFILTER_XT_MATCH_REALM is not set
# CONFIG_NETFILTER_XT_MATCH_RECENT is not set
# CONFIG_NETFILTER_XT_MATCH_STATISTIC is not set
# CONFIG_NETFILTER_XT_MATCH_STRING is not set
# CONFIG_NETFILTER_XT_MATCH_TCPMSS is not set
# CONFIG_NETFILTER_XT_MATCH_TIME is not set
# CONFIG_NETFILTER_XT_MATCH_U32 is not set
# CONFIG_IP_VS is not set

#
# IP: Netfilter Configuration
#
# CONFIG_NF_DEFRAG_IPV4 is not set
# CONFIG_IP_NF_QUEUE is not set
CONFIG_IP_NF_IPTABLES=y
# CONFIG_IP_NF_MATCH_ADDRTYPE is not set
# CONFIG_IP_NF_MATCH_AH is not set
# CONFIG_IP_NF_MATCH_ECN is not set
# CONFIG_IP_NF_MATCH_TTL is not set
CONFIG_IP_NF_FILTER=y
# CONFIG_IP_NF_TARGET_REJECT is not set
# CONFIG_IP_NF_TARGET_LOG is not set
# CONFIG_IP_NF_TARGET_ULOG is not set
CONFIG_IP_NF_MANGLE=y
# CONFIG_IP_NF_TARGET_ECN is not set
# CONFIG_IP_NF_TARGET_TTL is not set
# CONFIG_IP_NF_RAW is not set
# CONFIG_IP_NF_ARPTABLES is not set

#
# IPv6: Netfilter Configuration
#
# CONFIG_IP6_NF_QUEUE is not set
# CONFIG_IP6_NF_IPTABLES is not set
# CONFIG_ATM is not set
# CONFIG_BRIDGE is not set
# CONFIG_VLAN_8021Q is not set
# CONFIG_DECNET is not set
# CONFIG_LLC2 is not set
# CONFIG_IPX is not set
# CONFIG_ATALK is not set
# CONFIG_NET_SCHED is not set
# CONFIG_DCB is not set

#
# Network testing
#
# CONFIG_NET_PKTGEN is not set
# CONFIG_HAMRADIO is not set
# CONFIG_CAN is not set
# CONFIG_IRDA is not set
# CONFIG_BT is not set
# CONFIG_PHONET is not set
# CONFIG_WIRELESS is not set
# CONFIG_RFKILL is not set

#
# Device Drivers
#

#
# Generic Driver Options
#
CONFIG_UEVENT_HELPER_PATH="/sbin/hotplug"
CONFIG_STANDALONE=y
CONFIG_PREVENT_FIRMWARE_BUILD=y
CONFIG_FW_LOADER=y
CONFIG_FIRMWARE_IN_KERNEL=y
CONFIG_EXTRA_FIRMWARE=""
# CONFIG_SYS_HYPERVISOR is not set
# CONFIG_CONNECTOR is not set
# CONFIG_MTD is not set
CONFIG_OF_DEVICE=y
CONFIG_OF_I2C=y
# CONFIG_PARPORT is not set
CONFIG_BLK_DEV=y
# CONFIG_BLK_DEV_FD is not set
# CONFIG_BLK_DEV_COW_COMMON is not set
CONFIG_BLK_DEV_LOOP=y
# CONFIG_BLK_DEV_CRYPTOLOOP is not set
# CONFIG_BLK_DEV_NBD is not set
```

```
CONFIG_BLK_DEV_RAM=y
CONFIG_BLK_DEV_RAM_COUNT=16
CONFIG_BLK_DEV_RAM_SIZE=8192
# CONFIG_BLK_DEV_XIP is not set
# CONFIG_CDROM_PKTCDVD is not set
# CONFIG_ATA_OVER_ETH is not set
CONFIG_XILINX_SYSACE=y
# CONFIG_XILINX_SYSACE_OLD is not set
# CONFIG_BLK_DEV_HD is not set
CONFIG_MISC_DEVICES=y
# CONFIG_EEPROM_93CX6 is not set
# CONFIG_ENCLOSURE_SERVICES is not set
CONFIG_XILINX_DRIVERS=y
CONFIG_NEED_XILINX_LLDMA=y
CONFIG_NEED_XILINX_IPIF=y
CONFIG_HAVE_IDE=y
# CONFIG_IDE is not set

#
# SCSI device support
#
# CONFIG_RAID_ATTRS is not set
# CONFIG_SCSI is not set
# CONFIG_SCSI_DMA is not set
# CONFIG_SCSI_NETLINK is not set
# CONFIG_ATA is not set
# CONFIG_MD is not set
# CONFIG_MACINTOSH_DRIVERS is not set
CONFIG_NETDEVICES=y
# CONFIG_DUMMY is not set
# CONFIG_BONDING is not set
# CONFIG_EQUALIZER is not set
# CONFIG_TUN is not set
# CONFIG_VETH is not set
# CONFIG_PHYLIB is not set
CONFIG_NET_ETHERNET=y
CONFIG_MII=y
# CONFIG_IBM_NEW_EMAC is not set
# CONFIG_IBM_NEW_EMAC_ZMII is not set
# CONFIG_IBM_NEW_EMAC_RGMII is not set
# CONFIG_IBM_NEW_EMAC_TAH is not set
# CONFIG_IBM_NEW_EMAC_EMAC4 is not set
# CONFIG_IBM_NEW_EMAC_NO_FLOW_CTRL is not set
# CONFIG_IBM_NEW_EMAC_MAL_CLR_ICINTSTAT is not set
# CONFIG_IBM_NEW_EMAC_MAL_COMMON_ERR is not set
# CONFIG_B44 is not set
CONFIG_XILINX_EMAC=y
# CONFIG_XILINX_EMACLITE is not set
CONFIG_NETDEV_1000=y
# CONFIG_XILINX_TEMAC is not set
CONFIG_XILINX_LLTEMAC=y
# CONFIG_XILINX_LLTEMAC_MARVELL_88E1111_RGMII is not set
# CONFIG_XILINX_LLTEMAC_MARVELL_88E1111_GMII is not set
CONFIG_XILINX_LLTEMAC_MARVELL_88E1111_MII=y
# CONFIG_NETDEV_10000 is not set

#
# Wireless LAN
#
# CONFIG_WLAN_PRE80211 is not set
# CONFIG_WLAN_80211 is not set
# CONFIG_IWLWIFI_LEDS is not set
# CONFIG_WAN is not set
# CONFIG_PPP is not set
# CONFIG_SLIP is not set
# CONFIG_NETPOLL is not set
# CONFIG_NET_POLL_CONTROLLER is not set
# CONFIG_ISDN is not set
# CONFIG_PHONE is not set

#
# Input device support
#
CONFIG_INPUT=y
# CONFIG_INPUT_FF_MEMLESS is not set
# CONFIG_INPUT_POLLDEV is not set

#
# Userland interfaces
#
```

```
CONFIG_INPUT_MOUSEDEV=y
CONFIG_INPUT_MOUSEDEV_PSAUX=y
CONFIG_INPUT_MOUSEDEV_SCREEN_X=1024
CONFIG_INPUT_MOUSEDEV_SCREEN_Y=768
# CONFIG_INPUT_JOYDEV is not set
# CONFIG_INPUT_EVDEV is not set
# CONFIG_INPUT_EVBUG is not set

#
# Input Device Drivers
#
CONFIG_INPUT_KEYBOARD=y
CONFIG_KEYBOARD_ATKBD=y
# CONFIG_KEYBOARD_SUNKBD is not set
# CONFIG_KEYBOARD_LKKBD is not set
# CONFIG_KEYBOARD_XTKBD is not set
# CONFIG_KEYBOARD_NEWTON is not set
# CONFIG_KEYBOARD_STOWAWAY is not set
CONFIG_INPUT_MOUSE=y
CONFIG_MOUSE_PS2=y
CONFIG_MOUSE_PS2_ALPS=y
CONFIG_MOUSE_PS2_LOGIPS2PP=y
CONFIG_MOUSE_PS2_SYNAPTICS=y
CONFIG_MOUSE_PS2_LIFEBOOK=y
CONFIG_MOUSE_PS2_TRACKPOINT=y
# CONFIG_MOUSE_PS2_ELANTECH is not set
# CONFIG_MOUSE_PS2_TOUCHKIT is not set
# CONFIG_MOUSE_SERIAL is not set
# CONFIG_MOUSE_VSXXXAA is not set
# CONFIG_INPUT_JOYSTICK is not set
# CONFIG_INPUT_TABLET is not set
# CONFIG_INPUT_TOUCHSCREEN is not set
# CONFIG_INPUT_MISC is not set

#
# Hardware I/O ports
#
CONFIG_SERIO=y
# CONFIG_SERIO_I8042 is not set
CONFIG_SERIO_SERPORT=y
```

```
CONFIG_SERIO_LIBPS2=y
# CONFIG_SERIO_XILINXPS2 is not set
# CONFIG_SERIO_XILINX_XPS_PS2 is not set
# CONFIG_SERIO_RAW is not set
# CONFIG_GAMEPORT is not set

#
# Character devices
#
CONFIG_VT=y
CONFIG_CONSOLE_TRANSLATIONS=y
CONFIG_VT_CONSOLE=y
CONFIG_HW_CONSOLE=y
# CONFIG_VT_HW_CONSOLE_BINDING is not set
CONFIG_DEVKMEM=y
# CONFIG_SERIAL_NONSTANDARD is not set

#
# Serial drivers
#
CONFIG_SERIAL_8250=y
CONFIG_SERIAL_8250_CONSOLE=y
CONFIG_SERIAL_8250_NR_UARTS=4
CONFIG_SERIAL_8250_RUNTIME_UARTS=4
# CONFIG_SERIAL_8250_EXTENDED is not set

#
# Non-8250 serial port support
#
# CONFIG_SERIAL_UARTLITE is not set
CONFIG_SERIAL_CORE=y
CONFIG_SERIAL_CORE_CONSOLE=y
CONFIG_SERIAL_OF_PLATFORM=y
CONFIG_UNIX98_PTYS=y
CONFIG_LEGACY_PTYS=y
CONFIG_LEGACY_PTY_COUNT=256
# CONFIG_IPMI_HANDLER is not set
CONFIG_HW_RANDOM=y
# CONFIG_NVRAM is not set
# CONFIG_GEN_RTC is not set
```

```
CONFIG_XILINX_HWICAP=y
# CONFIG_R3964 is not set
# CONFIG_RAW_DRIVER is not set
CONFIG_I2C=y
CONFIG_I2C_BOARDINFO=y
CONFIG_I2C_CHARDEV=y
CONFIG_I2C_HELPER_AUTO=y

#
# I2C Hardware Bus support
#

#
# I2C system bus drivers (mostly embedded / system-on-chip)
#
# CONFIG_I2C_IBM_IIC is not set
# CONFIG_I2C_MPC is not set
# CONFIG_I2C_SIMTEC is not set

#
# External I2C/SMBus adapter drivers
#
# CONFIG_I2C_PARPORT_LIGHT is not set

#
# Other I2C/SMBus bus drivers
#
# CONFIG_I2C_PCA_PLATFORM is not set

#
# Miscellaneous I2C Chip support
#
# CONFIG_PCF8575 is not set
CONFIG_I2C_DEBUG_CORE=y
CONFIG_I2C_DEBUG_ALGO=y
# CONFIG_I2C_DEBUG_BUS is not set
# CONFIG_I2C_DEBUG_CHIP is not set
# CONFIG_SPI is not set
CONFIG_ARCH_WANT_OPTIONAL_GPIOLIB=y
# CONFIG_GPIOLIB is not set
# CONFIG_W1 is not set
# CONFIG_POWER_SUPPLY is not set
# CONFIG_HWMON is not set
# CONFIG_THERMAL is not set
# CONFIG_THERMAL_HWMON is not set
# CONFIG_WATCHDOG is not set
CONFIG_SSB_POSSIBLE=y

#
# Sonics Silicon Backplane
#
# CONFIG_SSB is not set

#
# Multifunction device drivers
#
# CONFIG_MFD_CORE is not set
# CONFIG_MFD_SM501 is not set
# CONFIG_HTC_PASIC3 is not set
# CONFIG_MFD_TMIO is not set
# CONFIG_PMIC_DA903X is not set
# CONFIG_MFD_WM8400 is not set
# CONFIG_MFD_WM8350_I2C is not set
# CONFIG_REGULATOR is not set

#
# Multimedia devices
#

#
# Multimedia core support
#
# CONFIG_VIDEO_DEV is not set
# CONFIG_DVB_CORE is not set
# CONFIG_VIDEO_MEDIA is not set

#
# Multimedia drivers
#
# CONFIG_DAB is not set
```

```
#
# Graphics support
#
# CONFIG_VGASTATE is not set
# CONFIG_VIDEO_OUTPUT_CONTROL is not set
CONFIG_FB=y
# CONFIG_FIRMWARE_EDID is not set
# CONFIG_FB_DDC is not set
# CONFIG_FB_BOOT_VESA_SUPPORT is not set
CONFIG_FB_CFB_FILLRECT=y
CONFIG_FB_CFB_COPYAREA=y
CONFIG_FB_CFB_IMAGEBLIT=y
# CONFIG_FB_CFB_REV_PIXELS_IN_BYTE is not set
# CONFIG_FB_SYS_FILLRECT is not set
# CONFIG_FB_SYS_COPYAREA is not set
# CONFIG_FB_SYS_IMAGEBLIT is not set
# CONFIG_FB_FOREIGN_ENDIAN is not set
# CONFIG_FB_SYS_FOPS is not set
# CONFIG_FB_SVGALIB is not set
# CONFIG_FB_MACMODES is not set
# CONFIG_FB_BACKLIGHT is not set
# CONFIG_FB_MODE_HELPERS is not set
# CONFIG_FB_TILEBLITTING is not set

#
# Frame buffer hardware drivers
#
# CONFIG_FB_OF is not set
# CONFIG_FB_VGA16 is not set
# CONFIG_FB_S1D13XXX is not set
# CONFIG_FB_IBM_GXT4500 is not set
CONFIG_FB_XILINX=y
# CONFIG_FB_VIRTUAL is not set
# CONFIG_FB_METRONOME is not set
# CONFIG_FB_MB862XX is not set
# CONFIG_BACKLIGHT_LCD_SUPPORT is not set

#
# Display device support
```

```
#
# CONFIG_DISPLAY_SUPPORT is not set

#
# Console display driver support
#
CONFIG_DUMMY_CONSOLE=y
CONFIG_FRAMEBUFFER_CONSOLE=y
# CONFIG_FRAMEBUFFER_CONSOLE_DETECT_PRIMARY is not set
# CONFIG_FRAMEBUFFER_CONSOLE_ROTATION is not set
CONFIG_FONTS=y
CONFIG_FONT_8x8=y
CONFIG_FONT_8x16=y
# CONFIG_FONT_6x11 is not set
# CONFIG_FONT_7x14 is not set
# CONFIG_FONT_PEARL_8x8 is not set
# CONFIG_FONT_ACORN_8x8 is not set
# CONFIG_FONT_MINI_4x6 is not set
# CONFIG_FONT_SUN8x16 is not set
# CONFIG_FONT_SUN12x22 is not set
# CONFIG_FONT_10x18 is not set
CONFIG_LOGO=y
CONFIG_LOGO_LINUX_MONO=y
CONFIG_LOGO_LINUX_VGA16=y
CONFIG_LOGO_LINUX_CLUT224=y
# CONFIG_SOUND is not set
# CONFIG_HID_SUPPORT is not set
# CONFIG_USB_SUPPORT is not set
# CONFIG_MMC is not set
# CONFIG_MEMSTICK is not set
# CONFIG_NEW_LEDS is not set
# CONFIG_ACCESSIBILITY is not set
# CONFIG_RTC_CLASS is not set
# CONFIG_DMADEVICES is not set
CONFIG_XILINX_EDK=y
# CONFIG_XILINX_LLDMA_USE_DCR is not set
# CONFIG_UIO is not set
# CONFIG_STAGING is not set

#
```

```
# File systems
#
CONFIG_EXT2_FS=y
# CONFIG_EXT2_FS_XATTR is not set
# CONFIG_EXT2_FS_XIP is not set
# CONFIG_EXT3_FS is not set
# CONFIG_EXT4_FS is not set
# CONFIG_REISERFS_FS is not set
# CONFIG_JFS_FS is not set
# CONFIG_FS_POSIX_ACL is not set
CONFIG_FILE_LOCKING=y
# CONFIG_XFS_FS is not set
# CONFIG_OCFS2_FS is not set
CONFIG_DNOTIFY=y
CONFIG_INOTIFY=y
CONFIG_INOTIFY_USER=y
# CONFIG_QUOTA is not set
CONFIG_AUTOFS_FS=y
CONFIG_AUTOFS4_FS=y
CONFIG_FUSE_FS=y

#
# CD-ROM/DVD Filesystems
#
# CONFIG_ISO9660_FS is not set
# CONFIG_UDF_FS is not set

#
# DOS/FAT/NT Filesystems
#
CONFIG_FAT_FS=y
CONFIG_MSDOS_FS=y
CONFIG_VFAT_FS=y
CONFIG_FAT_DEFAULT_CODEPAGE=437
CONFIG_FAT_DEFAULT_IOCHARSET="iso8859-1"
# CONFIG_NTFS_FS is not set

#
# Pseudo filesystems
#
```

```
CONFIG_PROC_FS=y
# CONFIG_PROC_KCORE is not set
CONFIG_PROC_SYSCTL=y
CONFIG_PROC_PAGE_MONITOR=y
CONFIG_SYSFS=y
CONFIG_TMPFS=y
# CONFIG_TMPFS_POSIX_ACL is not set
# CONFIG_HUGETLB_PAGE is not set
# CONFIG_CONFIGFS_FS is not set

#
# Miscellaneous filesystems
#
# CONFIG_HFSPLUS_FS is not set
CONFIG_CRAMFS=y
# CONFIG_VXFS_FS is not set
# CONFIG_MINIX_FS is not set
# CONFIG_OMFS_FS is not set
# CONFIG_HPFS_FS is not set
# CONFIG_QNX4FS_FS is not set
CONFIG_ROMFS_FS=y
# CONFIG_SYSV_FS is not set
# CONFIG_UFS_FS is not set
CONFIG_NETWORK_FILESYSTEMS=y
CONFIG_NFS_FS=y
CONFIG_NFS_V3=y
# CONFIG_NFS_V3_ACL is not set
CONFIG_ROOT_NFS=y
CONFIG_NFSD=y
CONFIG_NFSD_V3=y
# CONFIG_NFSD_V3_ACL is not set
CONFIG_LOCKD=y
CONFIG_LOCKD_V4=y
CONFIG_EXPORTFS=y
CONFIG_NFS_COMMON=y
CONFIG_SUNRPC=y
CONFIG_SMB_FS=y
# CONFIG_SMB_NLS_DEFAULT is not set
# CONFIG_CIFS is not set
# CONFIG_NCP_FS is not set
# CONFIG_CODA_FS is not set

#
# Partition Types
#
# CONFIG_PARTITION_ADVANCED is not set
CONFIG_MSDOS_PARTITION=y
CONFIG_NLS=y
CONFIG_NLS_DEFAULT="iso8859-1"
CONFIG_NLS_CODEPAGE_437=y
# CONFIG_NLS_CODEPAGE_737 is not set
# CONFIG_NLS_CODEPAGE_775 is not set
# CONFIG_NLS_CODEPAGE_850 is not set
# CONFIG_NLS_CODEPAGE_852 is not set
# CONFIG_NLS_CODEPAGE_855 is not set
# CONFIG_NLS_CODEPAGE_857 is not set
# CONFIG_NLS_CODEPAGE_860 is not set
# CONFIG_NLS_CODEPAGE_861 is not set
# CONFIG_NLS_CODEPAGE_862 is not set
# CONFIG_NLS_CODEPAGE_863 is not set
# CONFIG_NLS_CODEPAGE_864 is not set
# CONFIG_NLS_CODEPAGE_865 is not set
# CONFIG_NLS_CODEPAGE_866 is not set
# CONFIG_NLS_CODEPAGE_869 is not set
# CONFIG_NLS_CODEPAGE_936 is not set
# CONFIG_NLS_CODEPAGE_950 is not set
# CONFIG_NLS_CODEPAGE_932 is not set
# CONFIG_NLS_CODEPAGE_949 is not set
# CONFIG_NLS_CODEPAGE_874 is not set
# CONFIG_NLS_ISO8859_8 is not set
# CONFIG_NLS_CODEPAGE_1250 is not set
# CONFIG_NLS_CODEPAGE_1251 is not set
CONFIG_NLS_ASCII=y
CONFIG_NLS_ISO8859_1=y
# CONFIG_NLS_ISO8859_2 is not set
# CONFIG_NLS_ISO8859_3 is not set
# CONFIG_NLS_ISO8859_4 is not set
# CONFIG_NLS_ISO8859_5 is not set
# CONFIG_NLS_ISO8859_6 is not set
# CONFIG_NLS_ISO8859_7 is not set
# CONFIG_NLS_ISO8859_9 is not set
# CONFIG_NLS_ISO8859_13 is not set
# CONFIG_NLS_ISO8859_14 is not set
# CONFIG_NLS_ISO8859_15 is not set
# CONFIG_NLS_KOI8_R is not set
# CONFIG_NLS_KOI8_U is not set
CONFIG_NLS_UTF8=y

#
# Library routines
#
CONFIG_BITREVERSE=y
CONFIG_CRC_CCITT=y
# CONFIG_CRC16 is not set
# CONFIG_CRC_T10DIF is not set
# CONFIG_CRC_ITU_T is not set
CONFIG_CRC32=y
# CONFIG_CRC7 is not set
# CONFIG_LIBCRC32C is not set
CONFIG_ZLIB_INFLATE=y
CONFIG_PLIST=y
CONFIG_HAS_IOMEM=y
CONFIG_HAS_IOPORT=y
CONFIG_HAS_DMA=y
CONFIG_HAVE_LMB=y

#
# Kernel hacking
#
# CONFIG_PRINTK_TIME is not set
CONFIG_ENABLE_WARN_DEPRECATED=y
CONFIG_ENABLE_MUST_CHECK=y
CONFIG_FRAME_WARN=1024
# CONFIG_MAGIC_SYSRQ is not set
# CONFIG_UNUSED_SYMBOLS is not set
# CONFIG_DEBUG_FS is not set
# CONFIG_HEADERS_CHECK is not set
# CONFIG_DEBUG_KERNEL is not set
CONFIG_DEBUG_BUGVERBOSE=y
CONFIG_DEBUG_MEMORY_INIT=y
```

```
# CONFIG_RCU_CPU_STALL_DETECTOR is not set
# CONFIG_LATENCYTOP is not set
CONFIG_SYSCTL_SYSCALL_CHECK=y
CONFIG_HAVE_FUNCTION_TRACER=y

#
# Tracers
#
# CONFIG_DYNAMIC_PRINTK_DEBUG is not set
# CONFIG_SAMPLES is not set
CONFIG_HAVE_ARCH_KGDB=y
CONFIG_PRINT_STACK_DEPTH=64
# CONFIG_IRQSTACKS is not set
# CONFIG_PPC_EARLY_DEBUG is not set

#
# Security options
#
# CONFIG_KEYS is not set
# CONFIG_SECURITY is not set
# CONFIG_SECURITYFS is not set
# CONFIG_SECURITY_FILE_CAPABILITIES is not set
CONFIG_CRYPTO=y

#
# Crypto core or helper
#
# CONFIG_CRYPTO_FIPS is not set
# CONFIG_CRYPTO_MANAGER is not set
# CONFIG_CRYPTO_MANAGER2 is not set
# CONFIG_CRYPTO_NULL is not set
# CONFIG_CRYPTO_CRYPTD is not set
# CONFIG_CRYPTO_AUTHENC is not set
# CONFIG_CRYPTO_TEST is not set

#
# Authenticated Encryption with Associated Data
#
# CONFIG_CRYPTO_CCM is not set
# CONFIG_CRYPTO_GCM is not set
# CONFIG_CRYPTO_SEQIV is not set

#
# Block modes
#
# CONFIG_CRYPTO_CBC is not set
# CONFIG_CRYPTO_CTR is not set
# CONFIG_CRYPTO_CTS is not set
# CONFIG_CRYPTO_ECB is not set
# CONFIG_CRYPTO_PCBC is not set

#
# Hash modes
#
# CONFIG_CRYPTO_HMAC is not set

#
# Digest
#
# CONFIG_CRYPTO_CRC32C is not set
# CONFIG_CRYPTO_MD4 is not set
# CONFIG_CRYPTO_MD5 is not set
# CONFIG_CRYPTO_MICHAEL_MIC is not set
# CONFIG_CRYPTO_RMD128 is not set
# CONFIG_CRYPTO_RMD160 is not set
# CONFIG_CRYPTO_RMD256 is not set
# CONFIG_CRYPTO_RMD320 is not set
# CONFIG_CRYPTO_SHA1 is not set
# CONFIG_CRYPTO_SHA256 is not set
# CONFIG_CRYPTO_SHA512 is not set
# CONFIG_CRYPTO_TGR192 is not set
# CONFIG_CRYPTO_WP512 is not set

#
# Ciphers
#
# CONFIG_CRYPTO_AES is not set
# CONFIG_CRYPTO_ANUBIS is not set
# CONFIG_CRYPTO_ARC4 is not set
# CONFIG_CRYPTO_BLOWFISH is not set
# CONFIG_CRYPTO_CAMELLIA is not set
# CONFIG_CRYPTO_CAST5 is not set
# CONFIG_CRYPTO_CAST6 is not set
# CONFIG_CRYPTO_DES is not set
# CONFIG_CRYPTO_FCRYPT is not set
# CONFIG_CRYPTO_KHAZAD is not set
# CONFIG_CRYPTO_SEED is not set
# CONFIG_CRYPTO_SERPENT is not set
# CONFIG_CRYPTO_TEA is not set
# CONFIG_CRYPTO_TWOFISH is not set

#
# Compression
#
# CONFIG_CRYPTO_DEFLATE is not set
# CONFIG_CRYPTO_LZO is not set

#
# Random Number Generation
#
# CONFIG_CRYPTO_ANSI_CPRNG is not set
CONFIG_CRYPTO_HW=y
# CONFIG_PPC_CLOCK is not set
# CONFIG_VIRTUALIZATION is not set
```

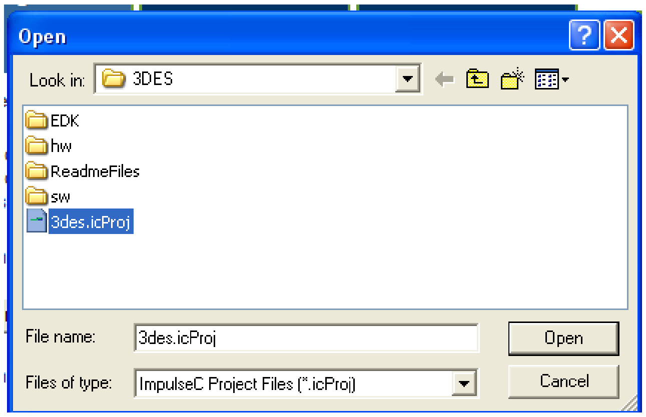
APPENDIX D

Setting 3DES for a Standalone Implementation

This appendix discusses how to get the 3DES core into a generic project. This section assumes you are using EDK 9.1i on a windows machine and have CoDeveloper 2.1 installed.

Step 1: Create IP Core

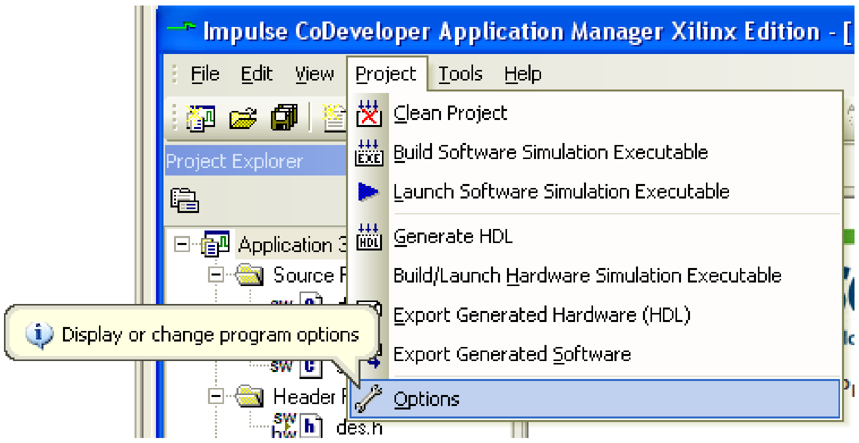
First you will need to open CoDeveloper 2.1. Once open you will need to open the 3DES project for the Virtex II Pro. On my machine this project is located under the CoDeveloper directory at /Examples/Xilinx/VirtexIIPro/3DES.



File name: 3des.icProj

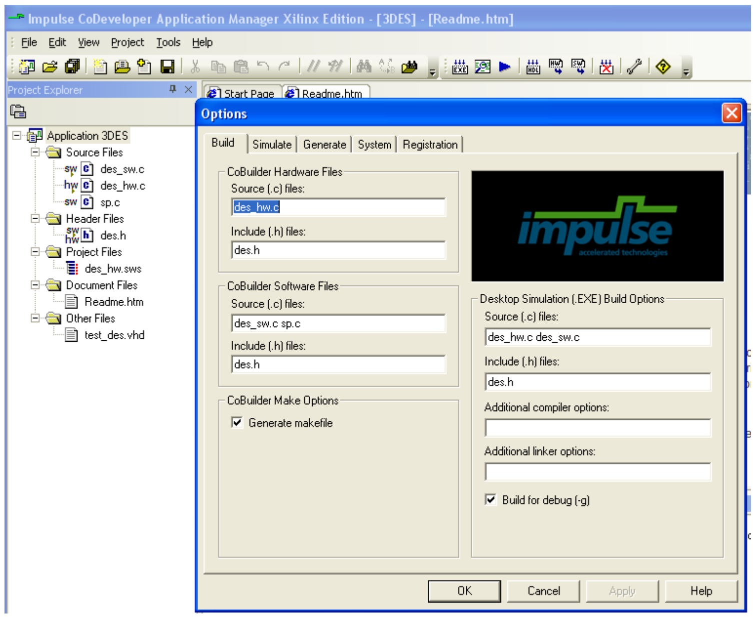
Files of type: ImpulseC Project Files (*.icProj)

Next you will need modify the options of Project, go to Project->Options as indicated below.


Impulse CoDeveloper Application Manager Xilinx Edition - [
File Edit View Project Tools Help
Project Explorer
Application 3
Source F
Header F
des.h
Clean Project
Build Software Simulation Executable
Launch Software Simulation Executable
Generate HDL
Build/Launch Hardware Simulation Executable
Export Generated Hardware (HDL)
Export Generated Software
Options

Display or change program options

Once in the options you will get a screen such as the one below.



CoBuilder Hardware Files
Source (.c) files: des_hw.c
Include (.h) files: des.h

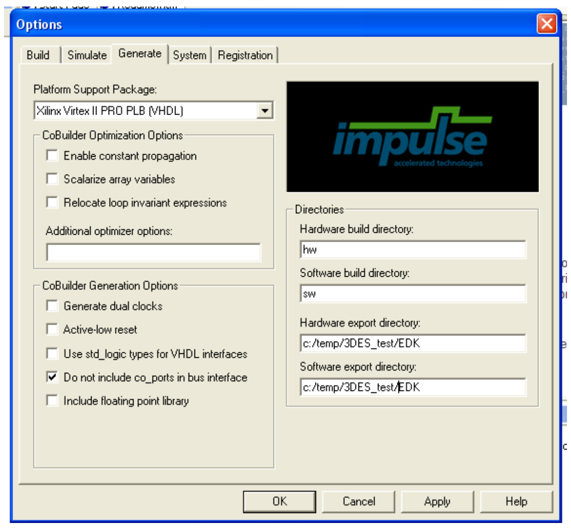
CoBuilder Software Files
Source (.c) files: des_sw.c sp.c
Include (.h) files: des.h

CoBuilder Make Options
[✓] Generate makefile

Desktop Simulation (.EXE) Build Options
Source (.c) files: des_hw.c des_sw.c
Include (.h) files: des.h
Additional compiler options:
Additional linker options:
[✓] Build for debug (-g)

Go to the Generate Tab on the options screen and modify the fields to match those below.

Once done, click OK.


Options

Build | Simulate | Generate | System | Registration

Platform Support Package: Xilinx Virtex II PRO PLB (VHDL)

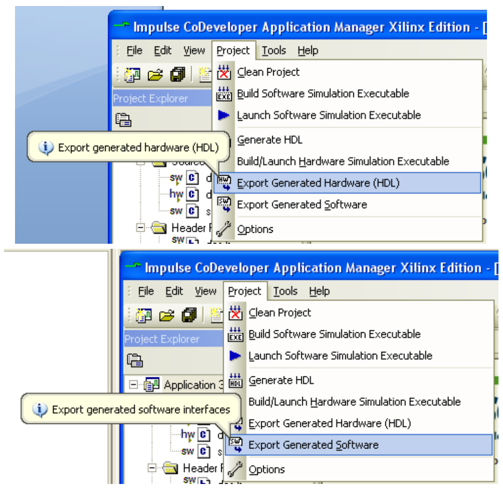
CoBuilder Optimization Options
- [ ] Enable constant propagation
- [ ] Scalarize array variables
- [ ] Relocate loop invariant expressions

Additional optimizer options:

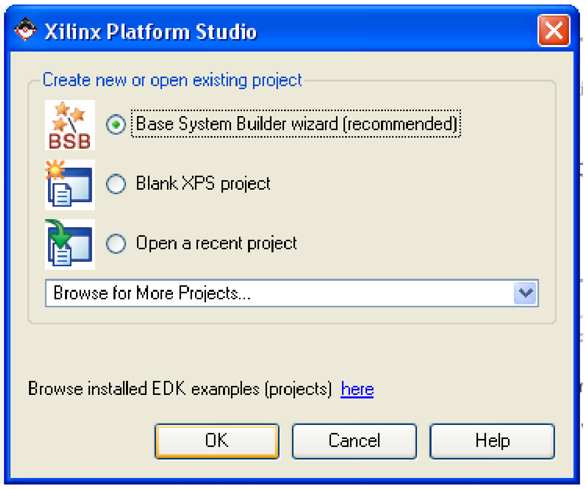
CoBuilder Generation Options
- [ ] Generate dual clocks
- [ ] Active-low reset
- [ ] Use std_logic types for VHDL interfaces
- [x] Do not include co_ports in bus interface
- [ ] Include floating point library

Directories
- Hardware build directory: hw
- Software build directory: sw
- Hardware export directory: c:/temp/3DES_test/EDK
- Software export directory: c:/temp/3DES_test/EDK

Next, you need to export the generated hardware and software, as indicated in the following two screen shots. Exporting the hardware and software will automatically generate the hardware and software if not all ready done so.


Impulse CoDeveloper Application Manager Xilinx Edition
File Edit View Project Tools Help
Project Explorer
Clean Project
Build Software Simulation Executable
Launch Software Simulation Executable
Generate HDL
Build/Launch Hardware Simulation Executable
Export Generated Hardware (HDL)
Export Generated Software
Options
Export generated hardware (HDL)
Export generated software interfaces

Step 2: Create a standalone system. Start XPS and follow the screenshots below.



Xilinx Platform Studio

Create new or open existing project

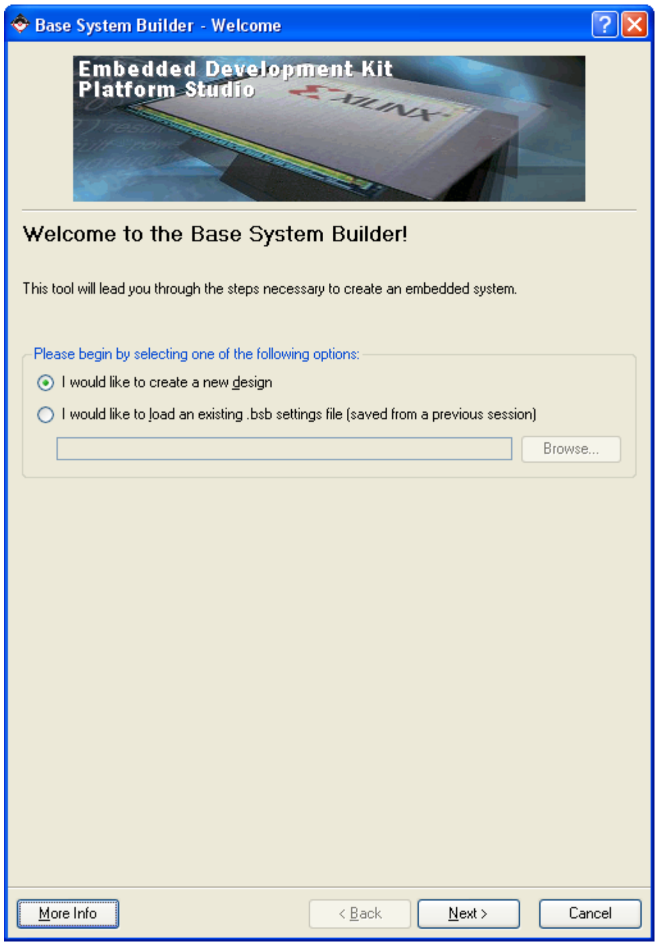
[✓ Base System Builder wizard (recommended)]

Blank XPS project

Open a recent project

Browse for More Projects...

Browse installed EDK examples (projects) here



Base System Builder - Welcome

Embedded Development Kit
Platform Studio

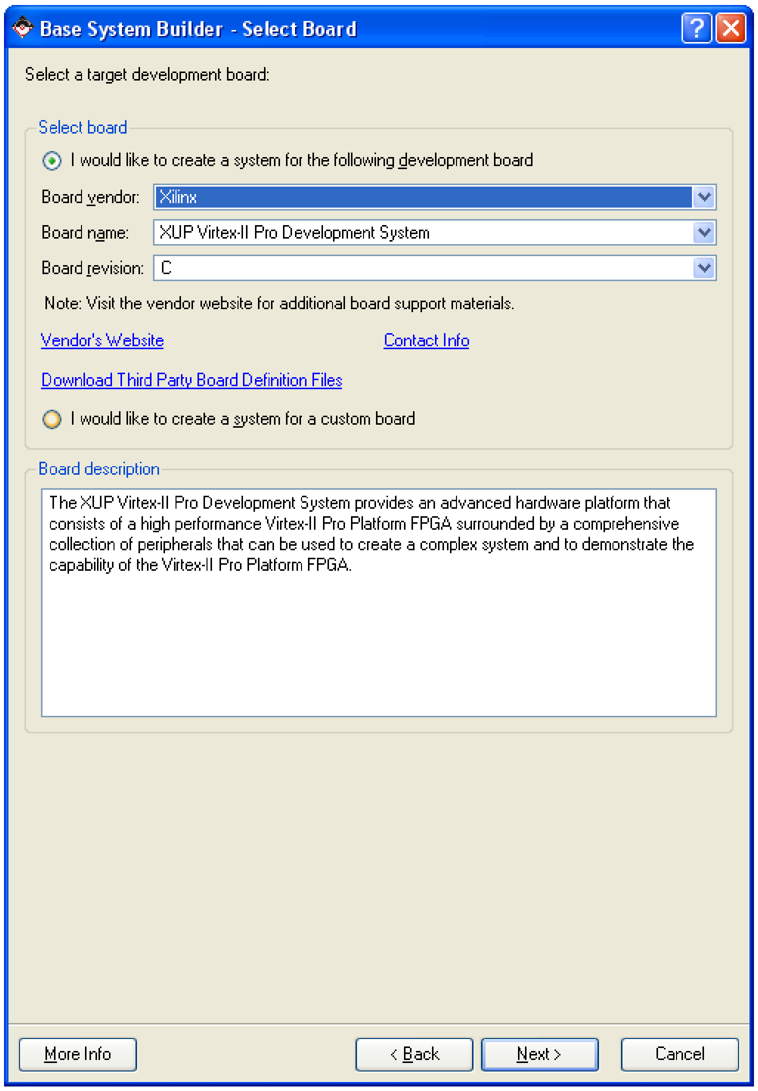
Welcome to the Base System Builder!

This tool will lead you through the steps necessary to create an embedded system.

Please begin by selecting one of the following options:

[✓ I would like to create a new design]

I would like to load an existing .bsb settings file (saved from a previous session)


Base System Builder - Select Board

Select a target development board:

Select board

[✓ I would like to create a system for the following development board]

Board vendor: Xilinx

Board name: XUP Virtex-II Pro Development System

Board revision: C

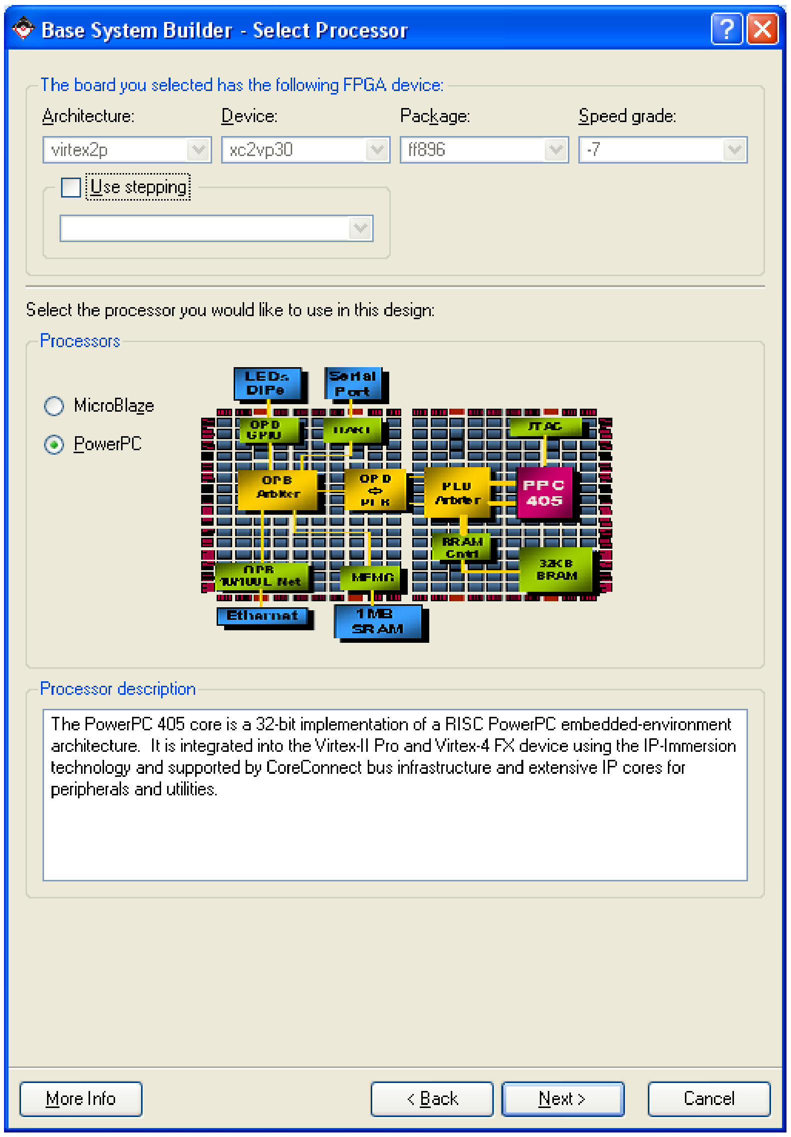
Note: Visit the vendor website for additional board support materials.

Vendor's Website
Contact Info
Download Third Party Board Definition Files

I would like to create a system for a custom board

Board description

The XUP Virtex-II Pro Development System provides an advanced hardware platform that consists of a high performance Virtex-II Pro Platform FPGA surrounded by a comprehensive collection of peripherals that can be used to create a complex system and to demonstrate the capability of the Virtex-II Pro Platform FPGA.



Base System Builder - Select Processor

The board you selected has the following FPGA device:

Architecture:	Device:	Package:	Speed grade:
virtex2p	xc2vp30	ff896	-7

Use stepping [ ]

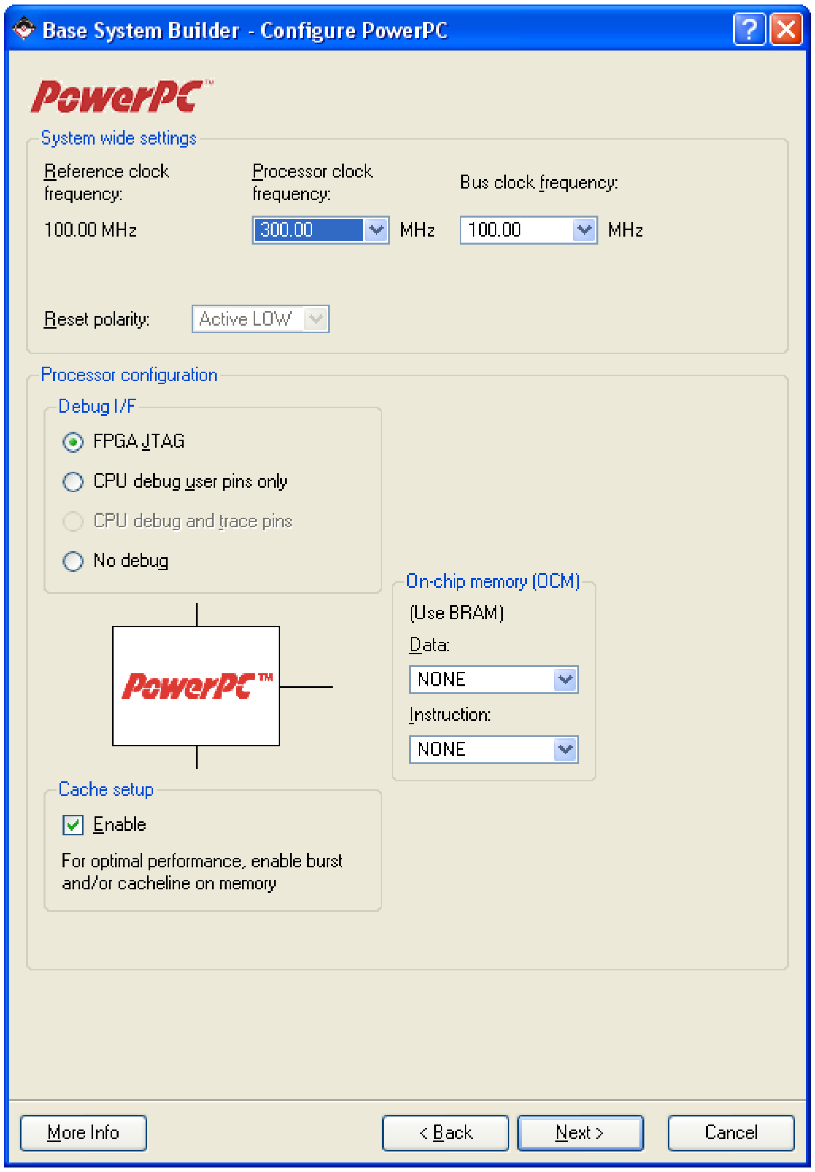
Select the processor you would like to use in this design:

Processors

MicroBlaze [✓ PowerPC]

Processor description

The PowerPC 405 core is a 32-bit implementation of a RISC PowerPC embedded-environment architecture. It is integrated into the Virtex-II Pro and Virtex-4 FX device using the IP-Immersion technology and supported by CoreConnect bus infrastructure and extensive IP cores for peripherals and utilities.



Base System Builder - Configure PowerPC

PowerPC™

System wide settings

Reference clock frequency: 100.00 MHz

Processor clock frequency: 300.00 MHz

Bus clock frequency: 100.00 MHz

Reset polarity: Active LOW

Processor configuration

Debug I/F

[✓ FPGA JTAG]

CPU debug user pins only

CPU debug and trace pins

No debug

On-chip memory (OCM)

(Use BRAM)

Data: NONE

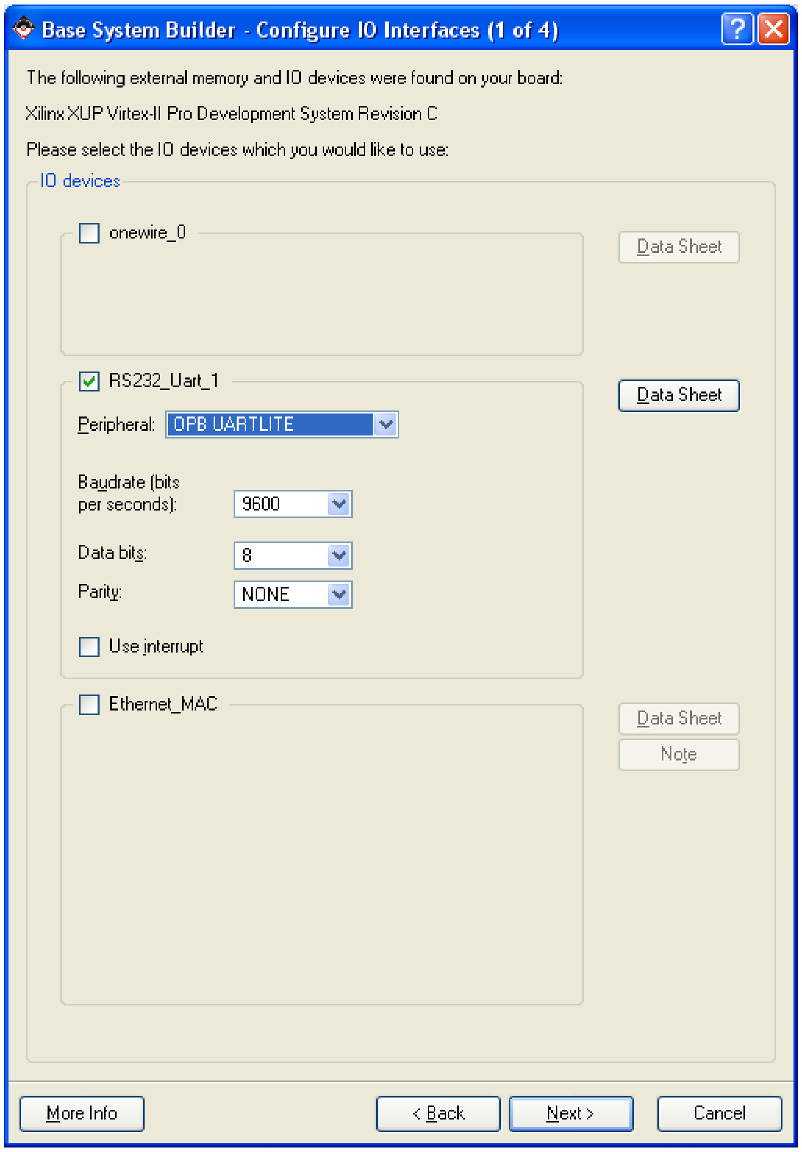
Instruction: NONE

Cache setup

[✓ Enable]

For optimal performance, enable burst and/or cacheline on memory

Note: I used the OPB UARTLITE Peripheral for RS232_Uart_1 because we did not have license at the time for any other RS232 core.


Base System Builder - Configure IO Interfaces (1 of 4)

The following external memory and IO devices were found on your board:

Xilinx XUP Virtex-II Pro Development System Revision C

Please select the IO devices which you would like to use:

IO devices

☐ onewire_0

☑ RS232_Uart_1

Peripheral: OPB UARTLITE

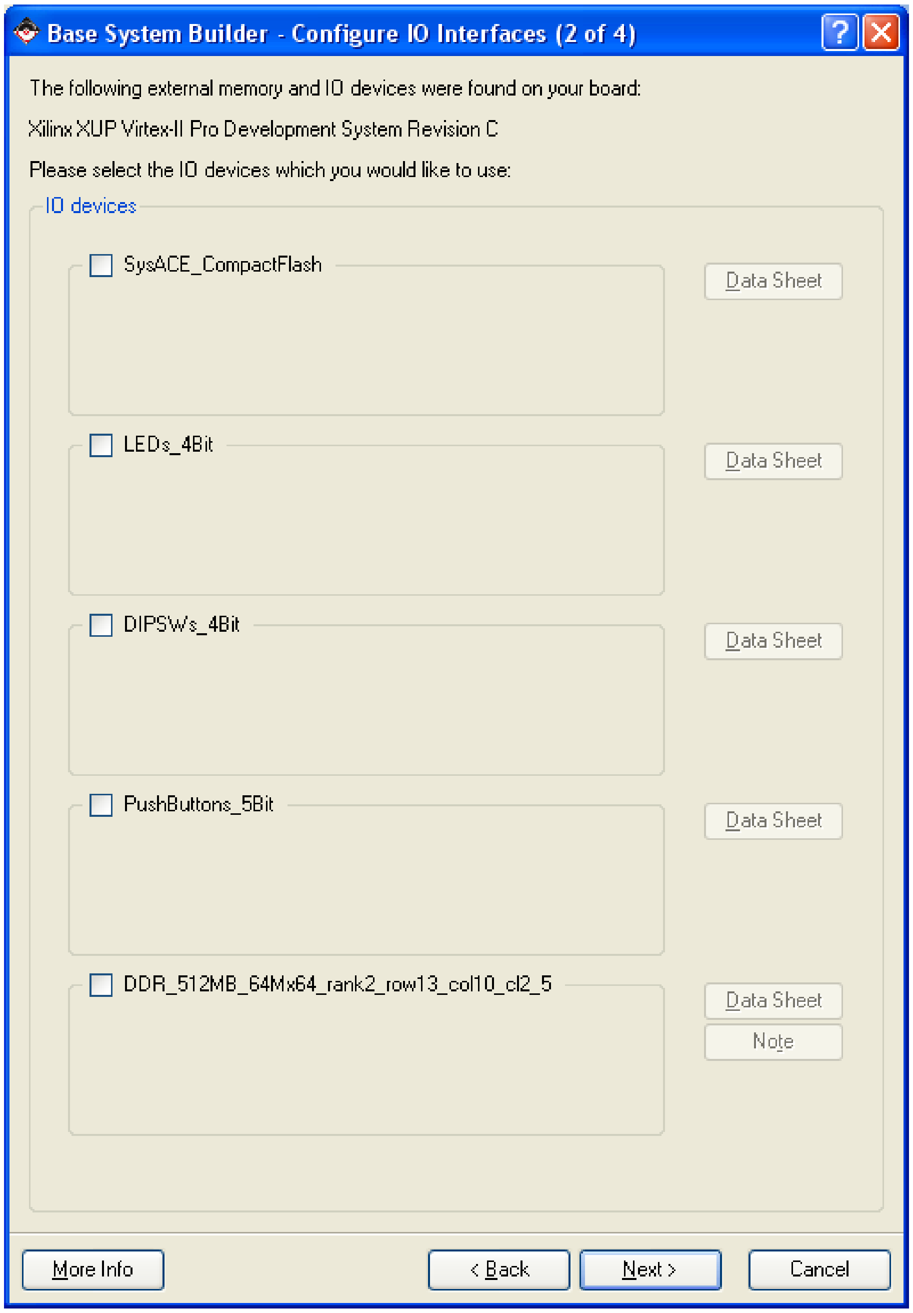
Baudrate (bits per seconds): 9600

Data bits: 8

Parity: NONE

☐ Use interrupt

☐ Ethernet_MAC


Base System Builder - Configure IO Interfaces (2 of 4)

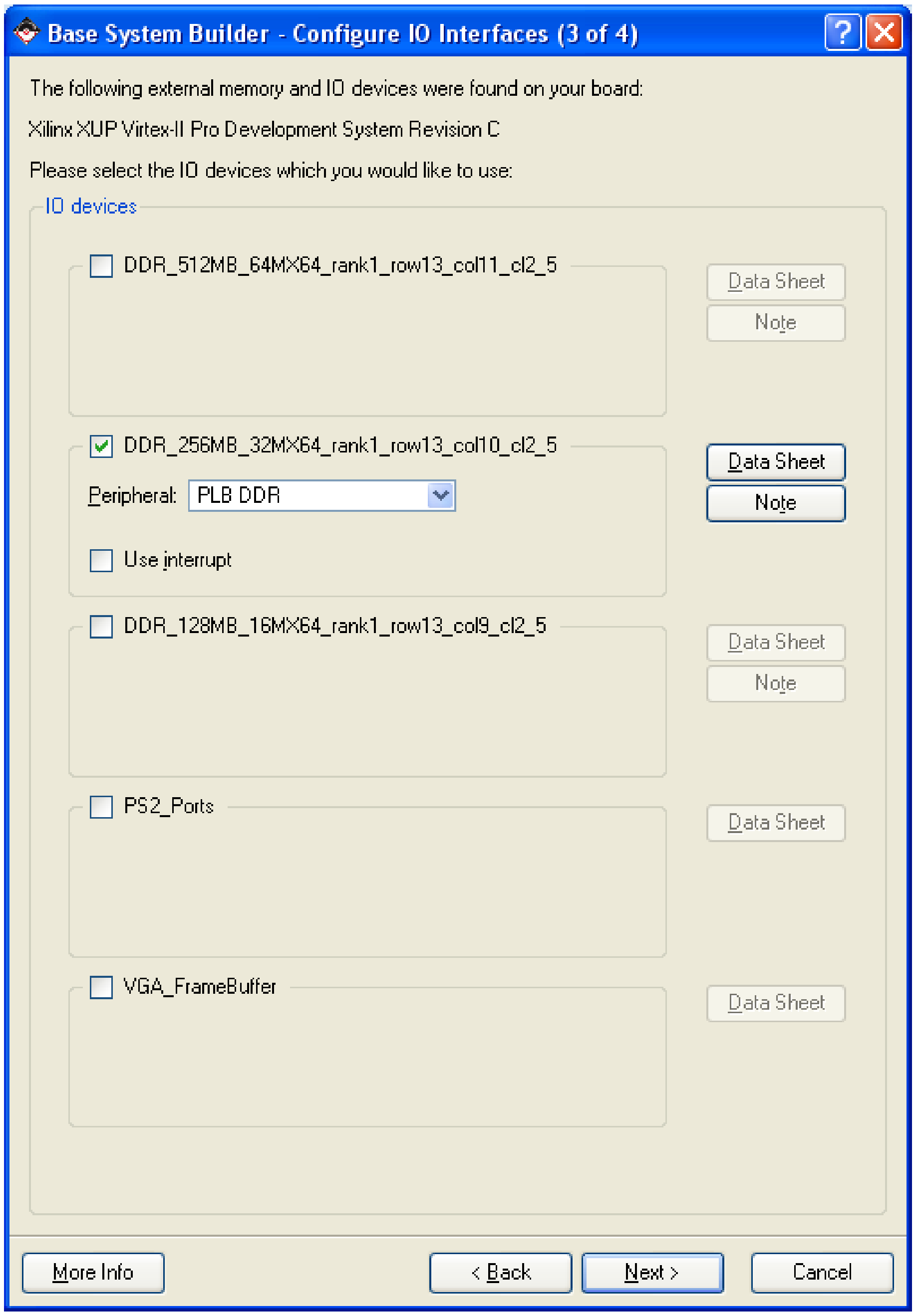
The following external memory and IO devices were found on your board:

Xilinx XUP Virtex-II Pro Development System Revision C

Please select the IO devices which you would like to use:

IO devices

- [ ] SysACE_CompactFlash
- [ ] LEDs_4Bit
- [ ] DIPSWs_4Bit
- [ ] PushButtons_5Bit
- [ ] DDR_512MB_64Mx64_rank2_row13_col10_cl2_5


Base System Builder - Configure IO Interfaces (3 of 4)

The following external memory and IO devices were found on your board:

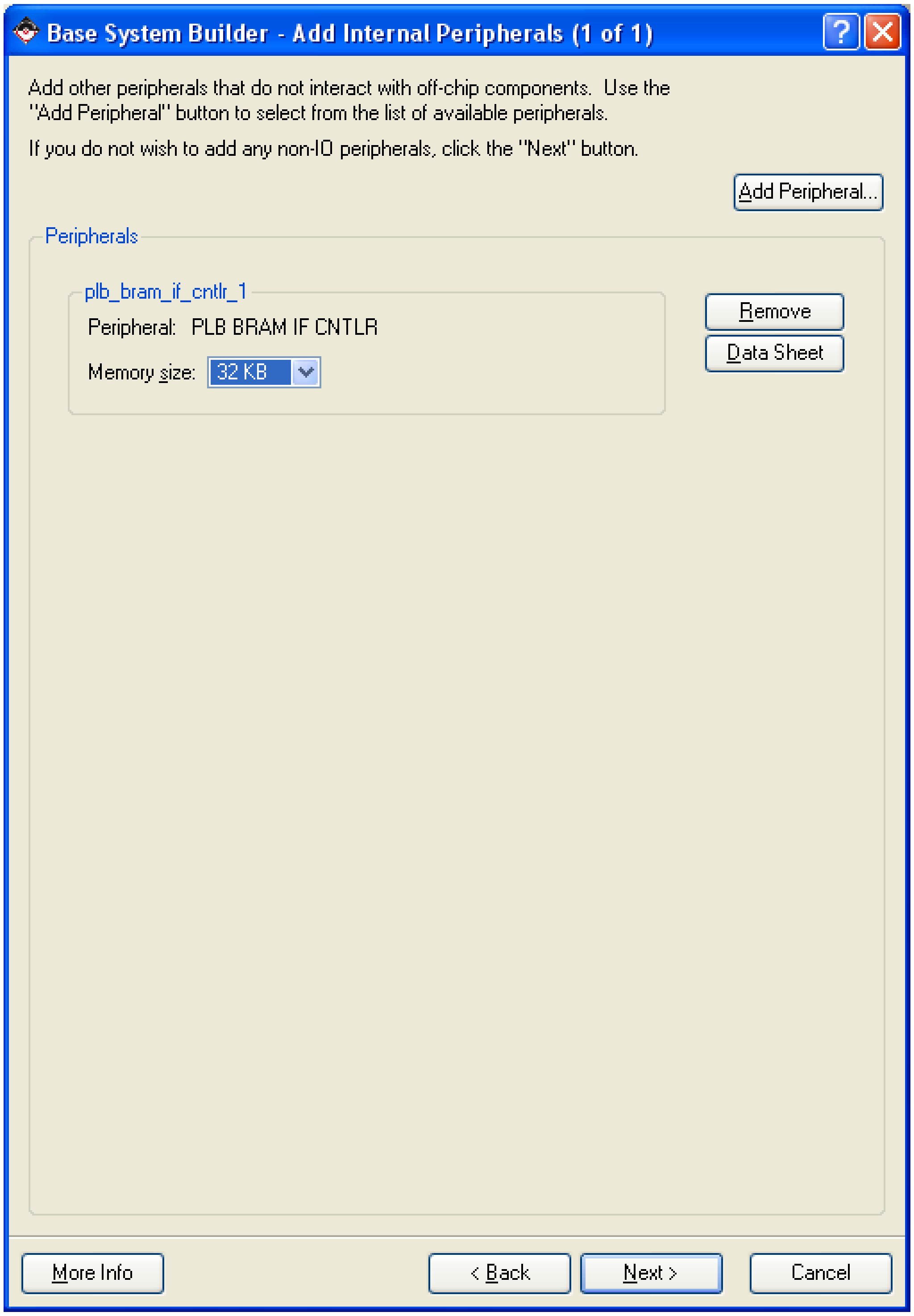
Xilinx XUP Virtex-II Pro Development System Revision C

Please select the IO devices which you would like to use:

IO devices

- [ ] DDR_512MB_64MX64_rank1_row13_col11_cl2_5
- [x] DDR_256MB_32MX64_rank1_row13_col10_cl2_5
  - Peripheral: PLB DDR
  - [ ] Use interrupt
- [ ] DDR_128MB_16MX64_rank1_row13_col9_cl2_5
- [ ] PS2_Ports
- [ ] VGA_FrameBuffer

Note: Here you need to click the Add Peripheral button to add the OPB TIMER Peripheral as needed by the IMPULSE C application code.


Base System Builder - Add Internal Peripherals (1 of 1)

Add other peripherals that do not interact with off-chip components. Use the "Add Peripheral" button to select from the list of available peripherals.

If you do not wish to add any non-IO peripherals, click the "Next" button.

Peripherals

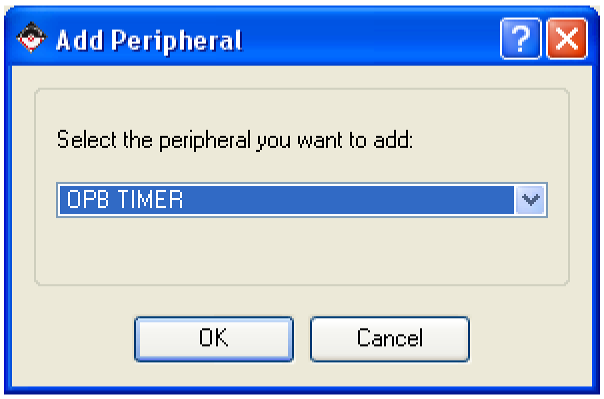
plb_bram_if_cntlr_1

Peripheral: PLB BRAM IF CNTLR

Memory size: 32 KB

If you clicked the Add Peripheral button you should get a window like the one below.

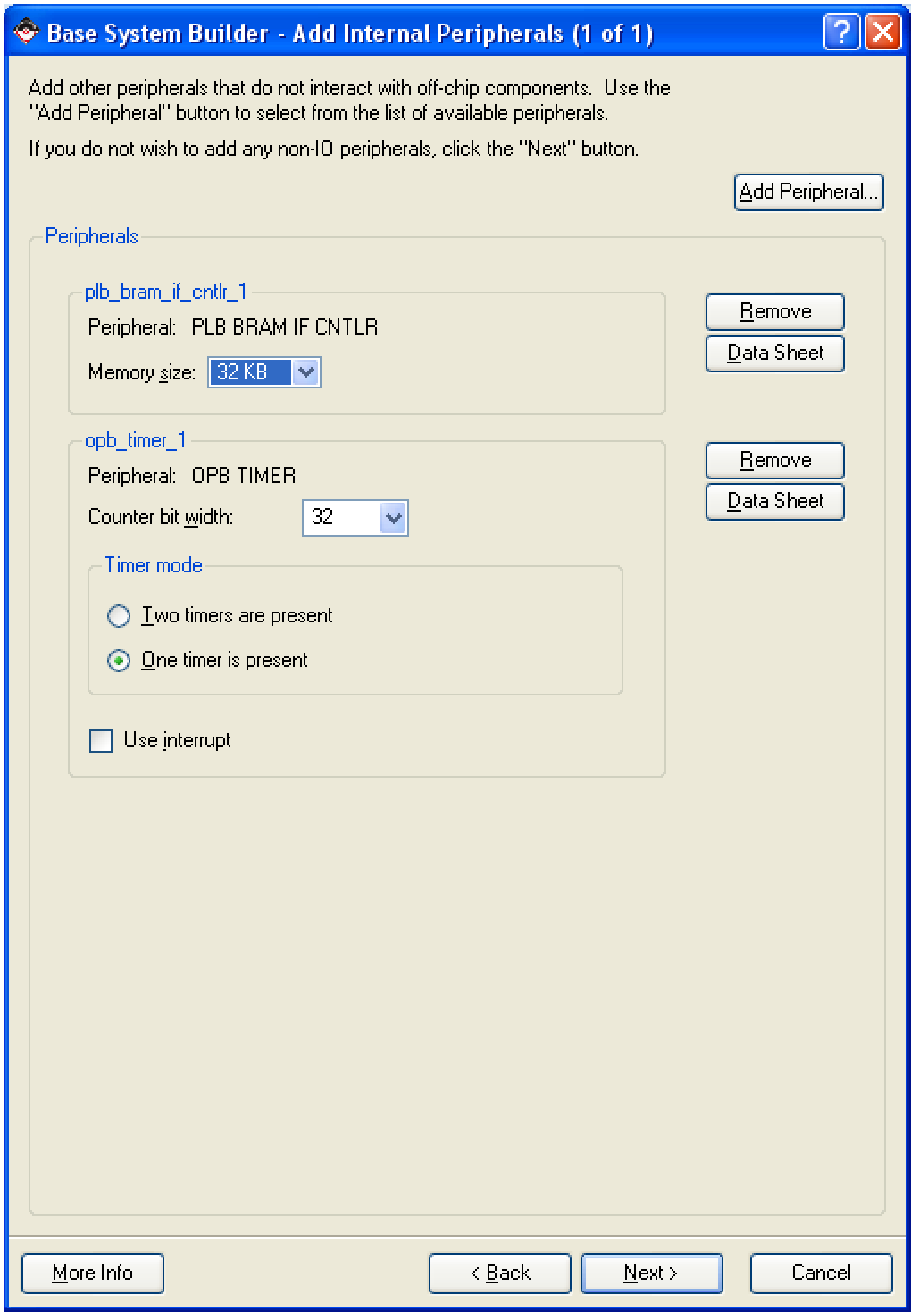
Select the OPB TIMER and click OK.


Add Peripheral

Select the peripheral you want to add:

OPB TIMER

If you successfully added the OPB TIMER, your window should appear as the one below.


Base System Builder - Add Internal Peripherals (1 of 1)

Add other peripherals that do not interact with off-chip components. Use the "Add Peripheral" button to select from the list of available peripherals.

If you do not wish to add any non-IO peripherals, click the "Next" button.

Peripherals

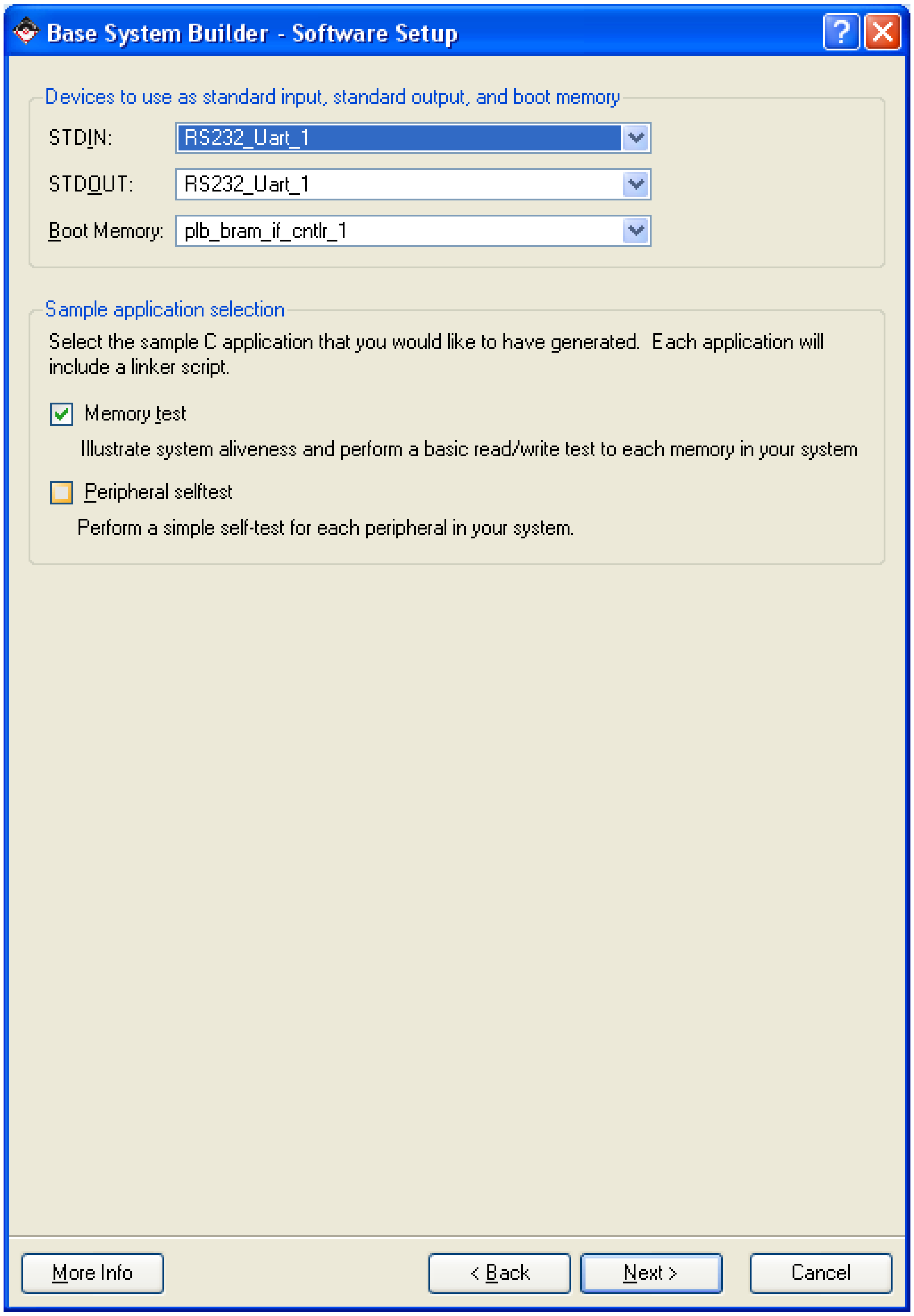
plb_bram_if_cntlr_1
Peripheral: PLB BRAM IF CNTLR
Memory size: 32 KB

opb_timer_1
Peripheral: OPB TIMER
Counter bit width: 32

Timer mode
Two timers are present [✓ One timer is present]

Use interrupt ☐

Note: I kept the memory test under sample application selection so I could use the linker script when I add the application test from IMPULSE C.


Base System Builder - Software Setup

Devices to use as standard input, standard output, and boot memory

STDIN: RS232_Uart_1

STDOUT: RS232_Uart_1

Boot Memory: plb_bram_if_cntlr_1

Sample application selection

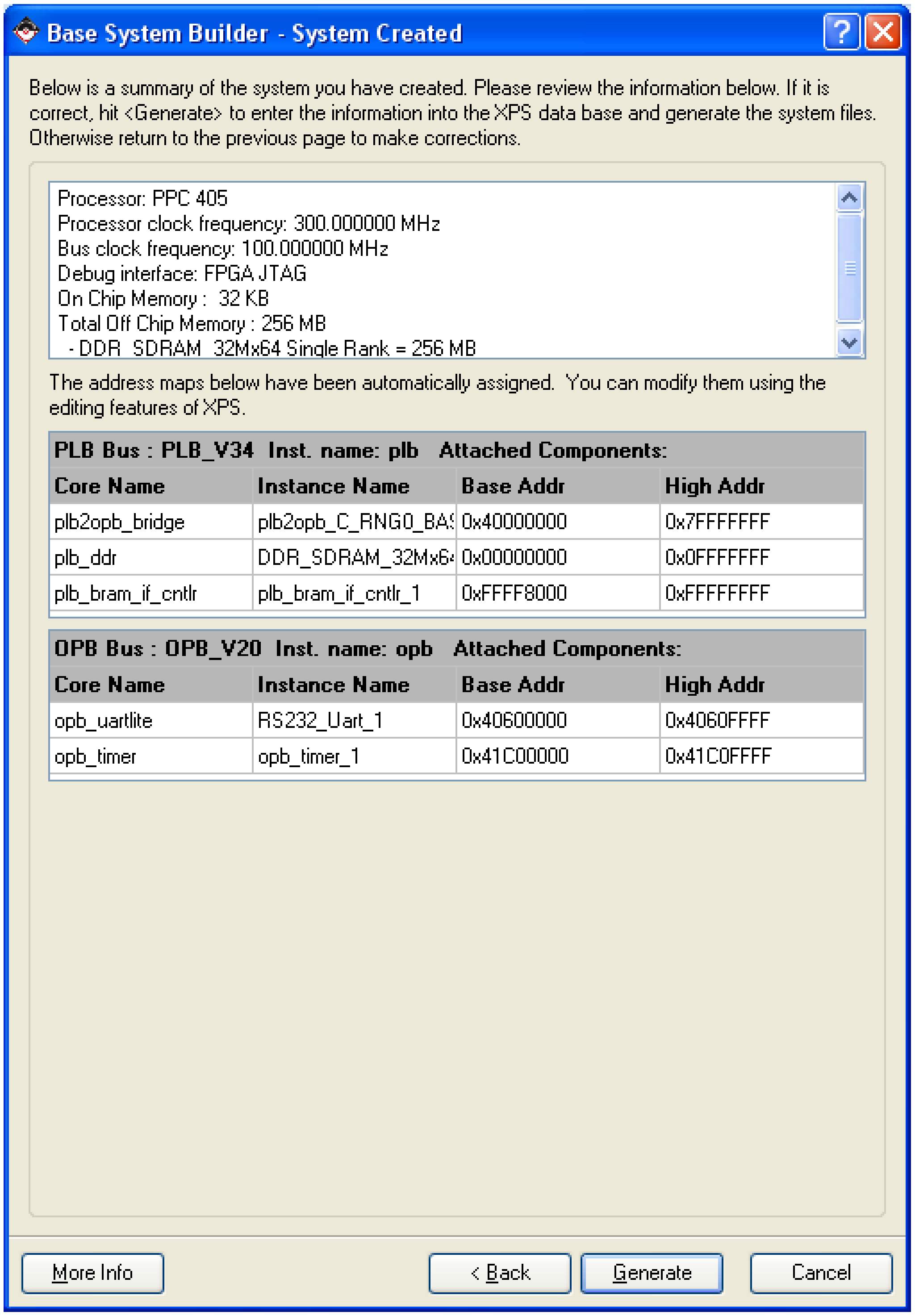
Select the sample C application that you would like to have generated. Each application will include a linker script.

[✓] Memory test

Illustrate system aliveness and perform a basic read/write test to each memory in your system

[ ] Peripheral selftest

Perform a simple self-test for each peripheral in your system.



Below is a summary of the system you have created. Please review the information below. If it is correct, hit <Generate> to enter the information into the XPS data base and generate the system files. Otherwise return to the previous page to make corrections.

```
Processor: PPC 405
Processor clock frequency: 300.000000 MHz
Bus clock frequency: 100.000000 MHz
Debug interface: FPGA JTAG
On Chip Memory :  32 KB
Total Off Chip Memory : 256 MB
 - DDR_SDRAM_32Mx64 Single Rank = 256 MB
```

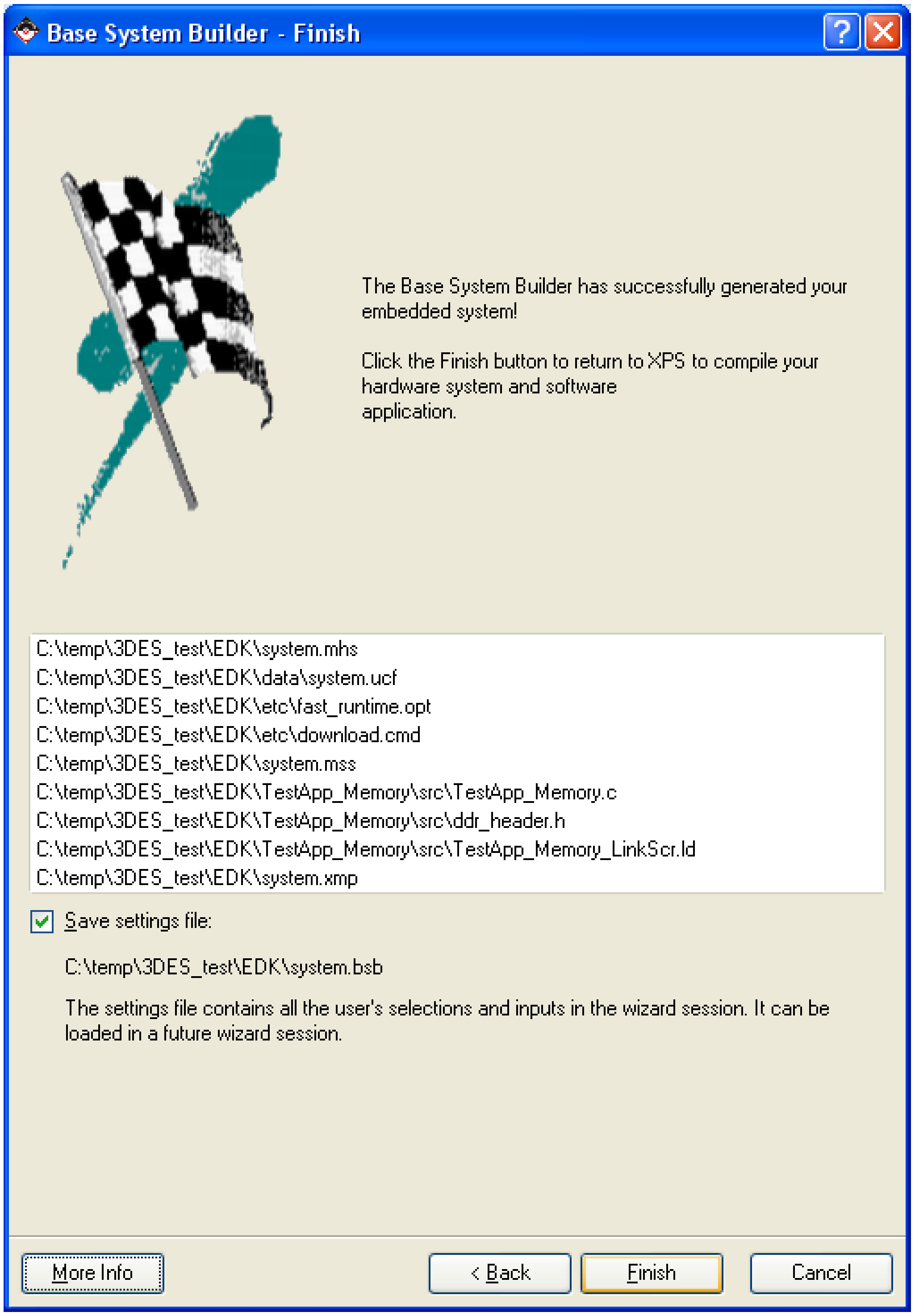
The address maps below have been automatically assigned. You can modify them using the editing features of XPS.

PLB Bus : PLB_V34 Inst. name: plb Attached Components:

Core Name	Instance Name	Base Addr	High Addr
plb2opb_bridge	plb2opb_C_RNG0_BAS	0x40000000	0x7FFFFFFF
plb_ddr	DDR_SDRAM_32Mx64	0x00000000	0x0FFFFFFF
plb_bram_if_cntlr	plb_bram_if_cntlr_1	0xFFFF8000	0xFFFFFFFF

OPB Bus : OPB_V20 Inst. name: opb Attached Components:

Core Name	Instance Name	Base Addr	High Addr
opb_uartlite	RS232_Uart_1	0x40600000	0x4060FFFF
opb_timer	opb_timer_1	0x41C00000	0x41C0FFFF



The Base System Builder has successfully generated your embedded system!

Click the Finish button to return to XPS to compile your hardware system and software application.

```
C:\temp\3DES_test\EDK\system.mhs
C:\temp\3DES_test\EDK\data\system.ucf
C:\temp\3DES_test\EDK\etc\fast_runtime.opt
C:\temp\3DES_test\EDK\etc\download.cmd
C:\temp\3DES_test\EDK\system.mss
C:\temp\3DES_test\EDK\TestApp_Memory\src\TestApp_Memory.c
C:\temp\3DES_test\EDK\TestApp_Memory\src\ddr_header.h
C:\temp\3DES_test\EDK\TestApp_Memory\src\TestApp_Memory_LinkScr.ld
C:\temp\3DES_test\EDK\system.xmp
```

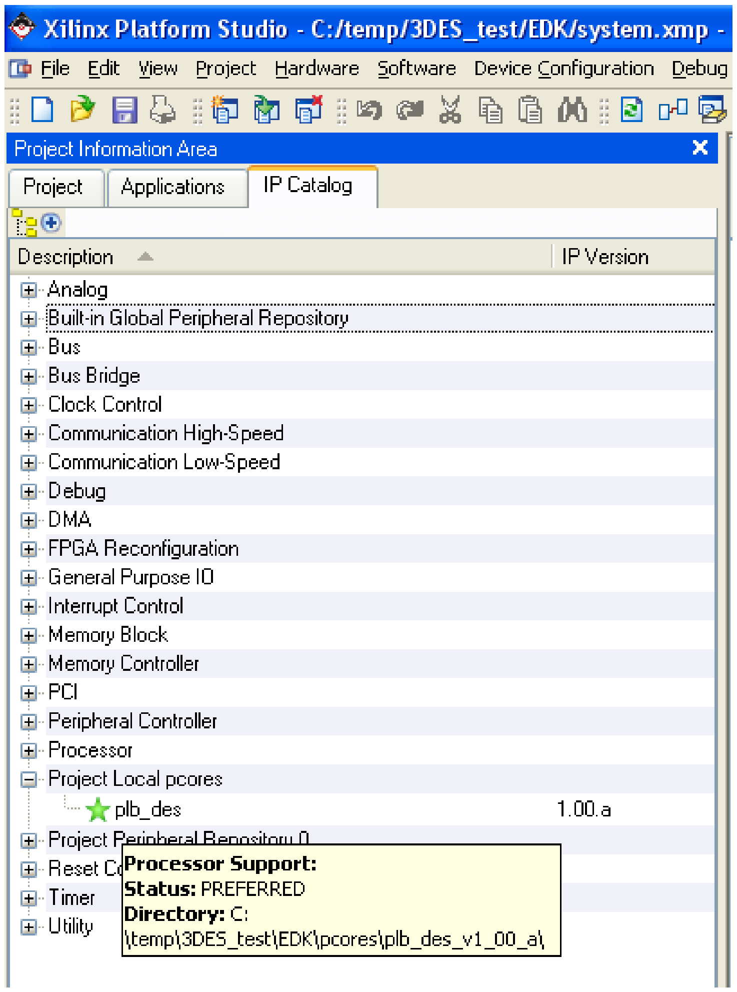
[✓] Save settings file:

C:\temp\3DES_test\EDK\system.bsb

The settings file contains all the user's selections and inputs in the wizard session. It can be loaded in a future wizard session.

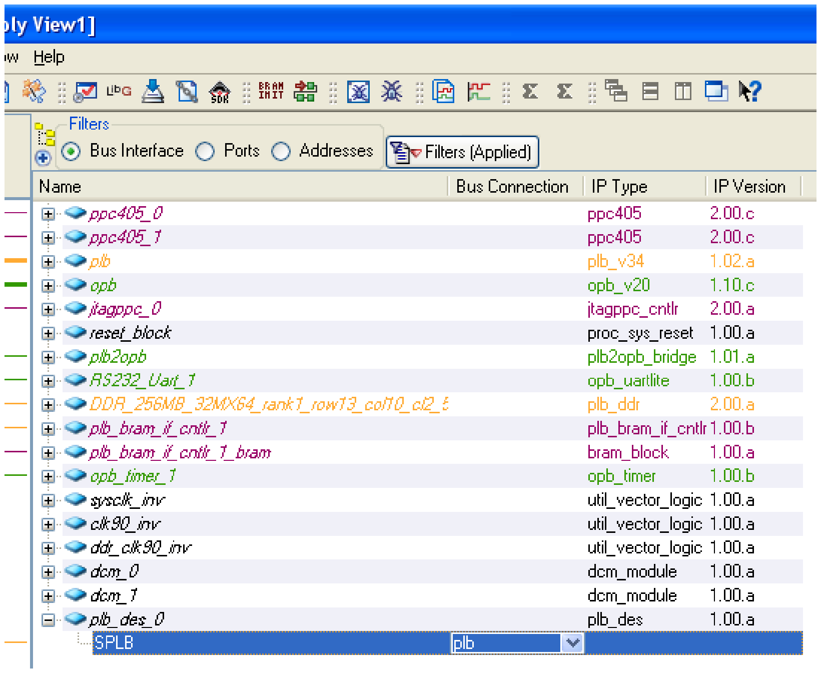
Step 3: Add the 3DES IP Core to the standalone system

To add the 3DES Core to your system you need to go to the IP Catalog and look under "Project Local pcores". Simply double click on plb_des to the IP Core to the system.



Processor Support:
Status: PREFERRED
Directory: C:\temp\3DES_test\EDK\pcores\plb_des_v1_00_a\

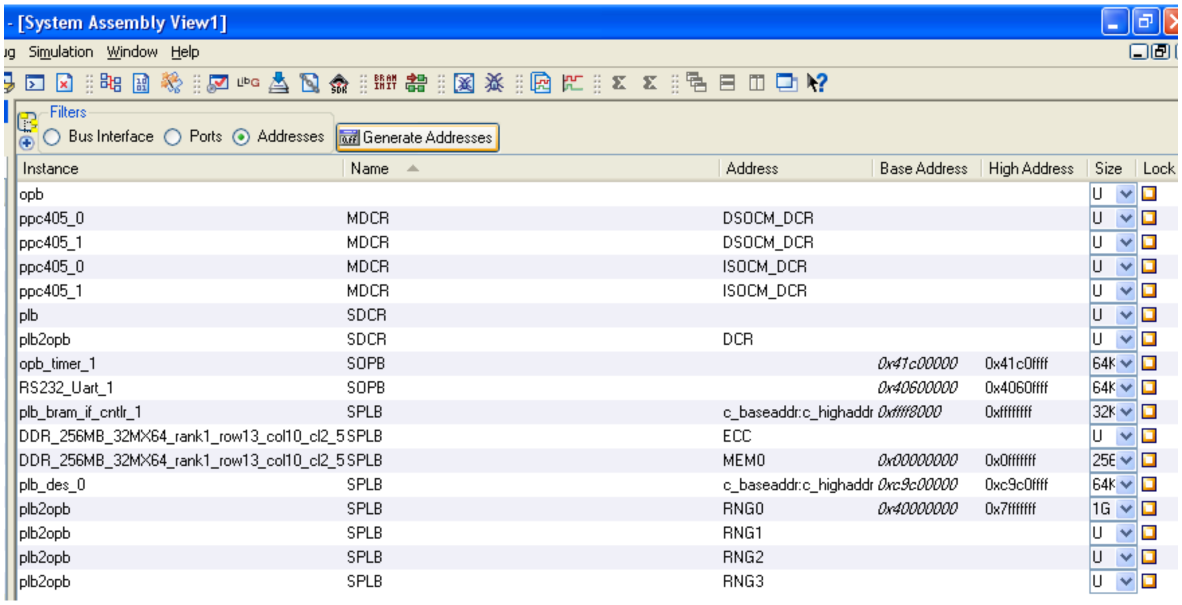
Once you have added the 3DES IP core to the system, you will need to wire it to the PLB bus as indicated below.



[✓ Bus Interface] Ports Addresses — Filters (Applied)

Name	Bus Connection	IP Type	IP Version
ppc405_0		ppc405	2.00.c
ppc405_1		ppc405	2.00.c
plb		plb_v34	1.02.a
opb		opb_v20	1.10.c
jtagppc_0		jtagppc_cntlr	2.00.a
reset_block		proc_sys_reset	1.00.a
plb2opb		plb2opb_bridge	1.01.a
RS232_Uart_1		opb_uartlite	1.00.b
DDR_256MB_32MX64_rank1_row13_col10_cl2_5		plb_ddr	2.00.a
plb_bram_if_cntlr_1		plb_bram_if_cntlr	1.00.b
plb_bram_if_cntlr_1_bram		bram_block	1.00.a
opb_timer_1		opb_timer	1.00.b
sysclk_inv		util_vector_logic	1.00.a
clk90_inv		util_vector_logic	1.00.a
ddr_clk90_inv		util_vector_logic	1.00.a
dcm_0		dcm_module	1.00.a
dcm_1		dcm_module	1.00.a
plb_des_0		plb_des	1.00.a
SPLB	plb		

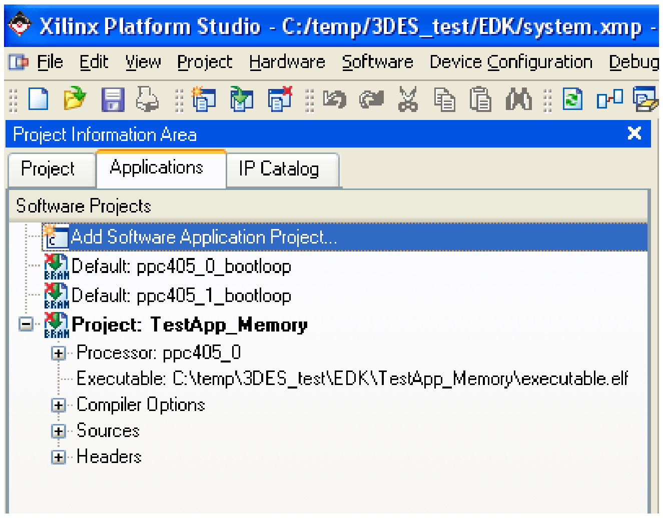
After the IP Core has been wired, you will need to go to the address section in the main window. Once here click Generate Addresses.



Instance	Name	Address	Base Address	High Address	Size	Lock
opb					U	
ppc405_0	MDCR	DSOCM_DCR			U	
ppc405_1	MDCR	DSOCM_DCR			U	
ppc405_0	MDCR	ISOCM_DCR			U	
ppc405_1	MDCR	ISOCM_DCR			U	
plb	SDCR				U	
plb2opb	SDCR	DCR			U	
opb_timer_1	SOPB		*0x41c00000*	0x41c0ffff	64K	
RS232_Uart_1	SOPB		*0x40600000*	0x4060ffff	64K	
plb_bram_if_cntlr_1	SPLB	c_baseaddr:c_highaddr	*0xffff8000*	0xffffffff	32K	
DDR_256MB_32MX64_rank1_row13_col10_cl2_5	SPLB	ECC			U	
DDR_256MB_32MX64_rank1_row13_col10_cl2_5	SPLB	MEM0	*0x00000000*	0x0fffffff	256	
plb_des_0	SPLB	c_baseaddr:c_highaddr	*0xc9c00000*	0xc9c0ffff	64K	
plb2opb	SPLB	RNG0	*0x40000000*	0x7fffffff	1G	
plb2opb	SPLB	RNG1			U	
plb2opb	SPLB	RNG2			U	
plb2opb	SPLB	RNG3			U	

Step 4: Load the generated application software

Now that the IP Core is wired within the system, you need to add the IMPULSE C software application project. Go to the Applications tab under the Project Information Area and double click "Add Software Application Project"

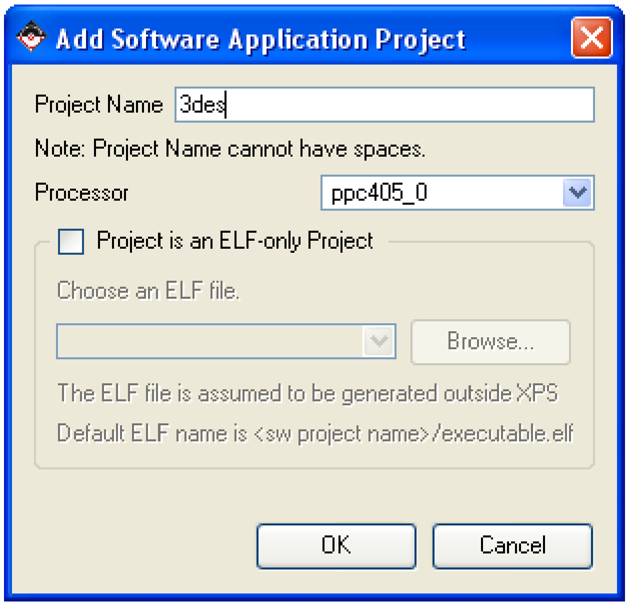


Xilinx Platform Studio - C:/temp/3DES_test/EDK/system.xmp -

File Edit View Project Hardware Software Device Configuration Debug
Project Information Area
Project Applications IP Catalog

Software Projects
- Add Software Application Project...
- Default: ppc405_0_bootloop
- Default: ppc405_1_bootloop
- Project: TestApp_Memory
  - Processor: ppc405_0
  - Executable: C:\temp\3DES_test\EDK\TestApp_Memory\executable.elf
  - Compiler Options
  - Sources
  - Headers

A window should pop up like the one below, where you can choose to name the project name.



Add Software Application Project

Project Name: 3des

Note: Project Name cannot have spaces.

Processor: ppc405_0

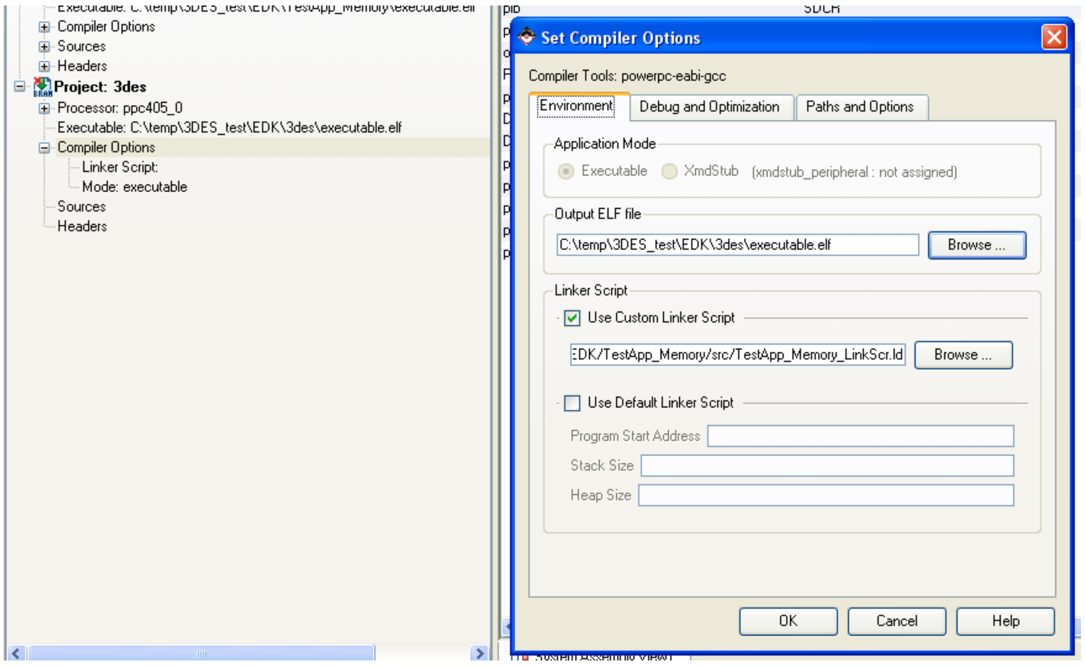
[ ] Project is an ELF-only Project

Choose an ELF file.

The ELF file is assumed to be generated outside XPS

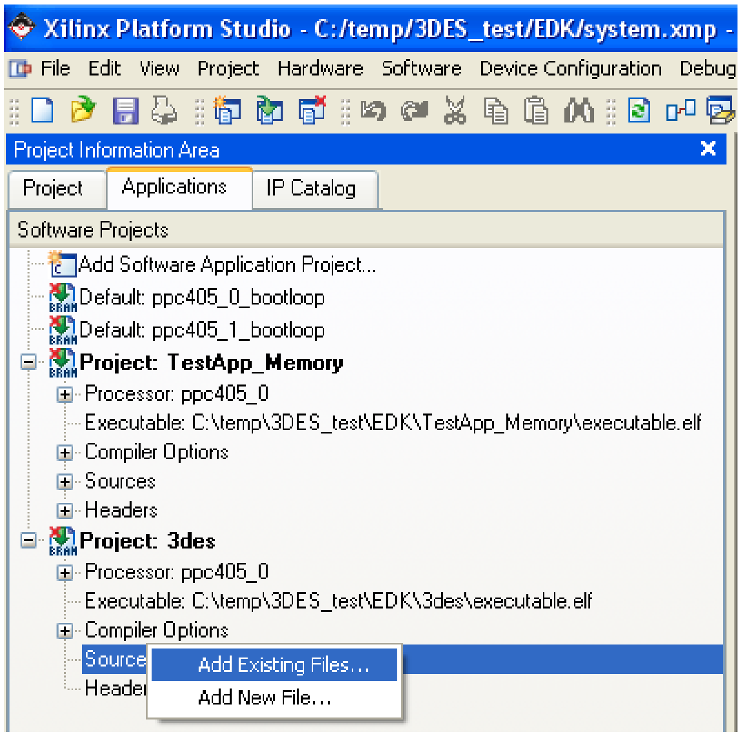
Default ELF name is <sw project name>/executable.elf

Double click the Compiler Options under the newly created project and change the Linker Script so that it points to the TestApp_Memory linker script.

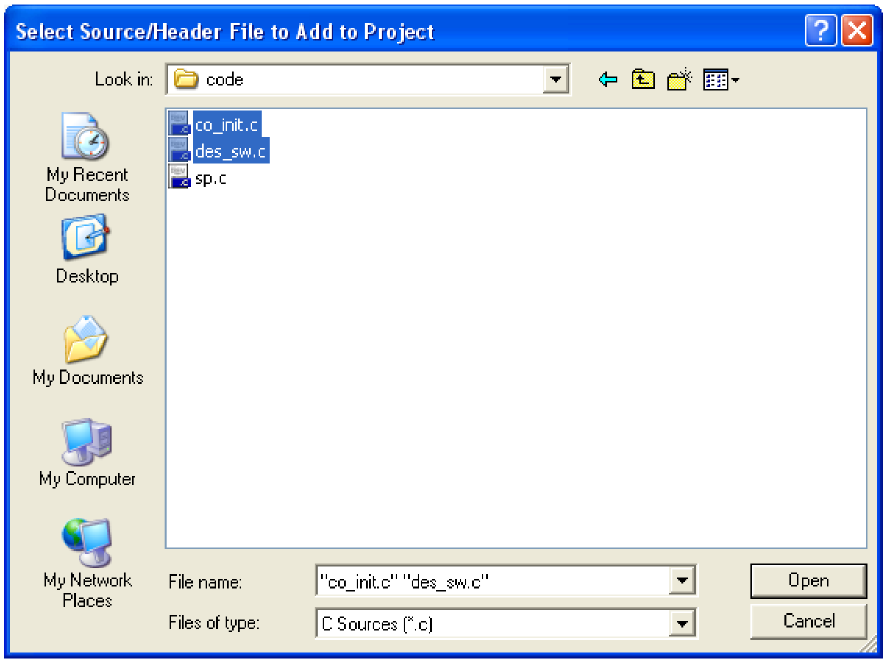

Project: 3des
Processor: ppc405_0
Executable: C:\temp\3DES_test\EDK\3des\executable.elf
Compiler Options
Linker Script:
Mode: executable
Sources
Headers

Set Compiler Options
Compiler Tools: powerpc-eabi-gcc
Environment | Debug and Optimization | Paths and Options
Application Mode: [✓ Executable] XmdStub (xmdstub_peripheral : not assigned)
Output ELF file: C:\temp\3DES_test\EDK\3des\executable.elf
Linker Script
[✓] Use Custom Linker Script: EDK/TestApp_Memory/src/TestApp_Memory_LinkScr.ld
[ ] Use Default Linker Script
Program Start Address:
Stack Size:
Heap Size:

Next, right click the Sources under the newly created software project and select Add Existing files.


Xilinx Platform Studio - C:/temp/3DES_test/EDK/system.xmp -
File Edit View Project Hardware Software Device Configuration Debug
Project Information Area
Project Applications IP Catalog
Software Projects
Add Software Application Project...
Default: ppc405_0_bootloop
Default: ppc405_1_bootloop
Project: TestApp_Memory
Processor: ppc405_0
Executable: C:\temp\3DES_test\EDK\TestApp_Memory\executable.elf
Compiler Options
Sources
Headers
Project: 3des
Processor: ppc405_0
Executable: C:\temp\3DES_test\EDK\3des\executable.elf
Compiler Options
Source
Header
Add Existing Files...
Add New File...

The files you need to add are co_init.c and des.sw.c.



Select Source/Header File to Add to Project

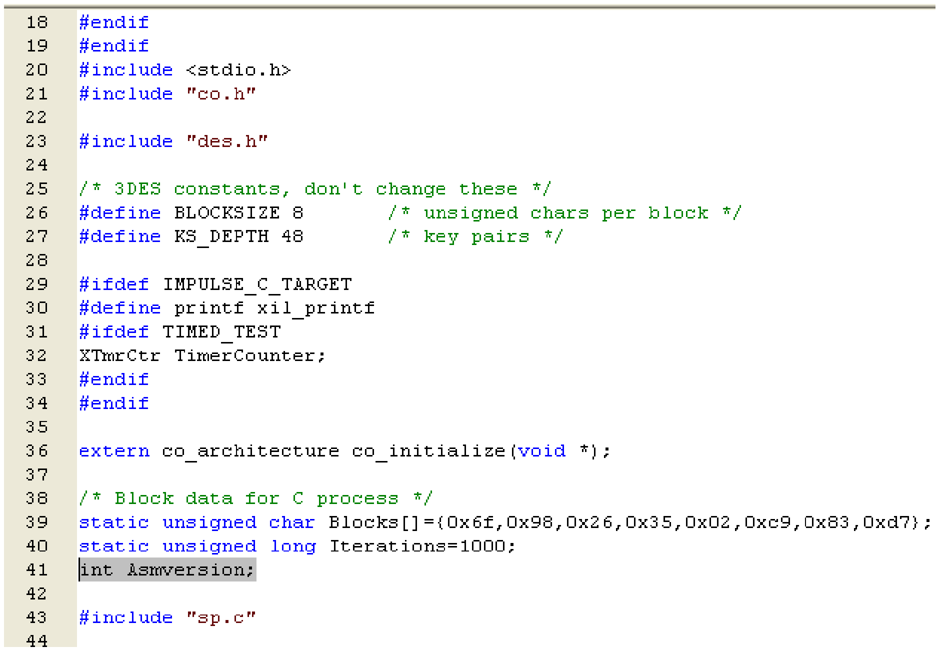
Look in: code

co_init.c
des_sw.c
sp.c

File name: "co_init.c" "des_sw.c"

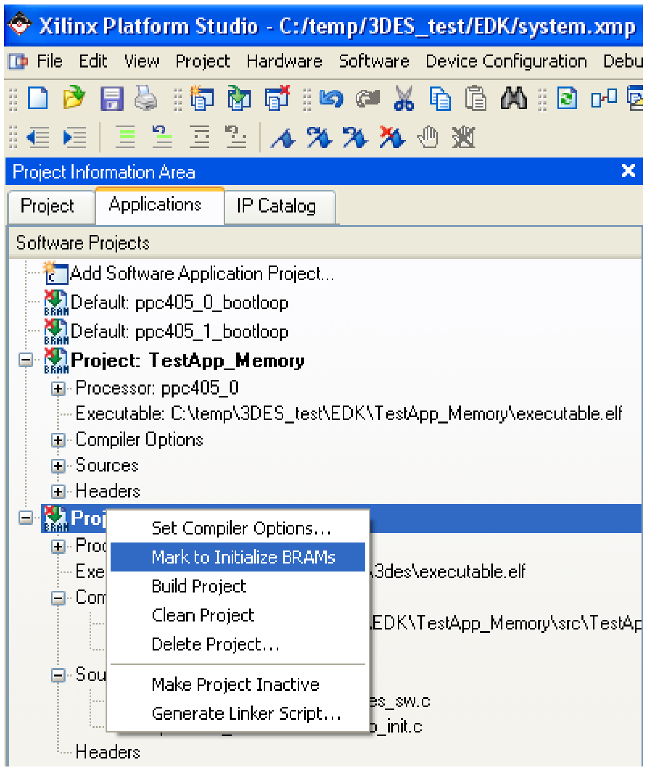
Files of type: C Sources (*.c)

As is, the code will not compile. Des_sw.c must be modified to work, on line 41 modify the like to read "int Asmversion" as shown below.



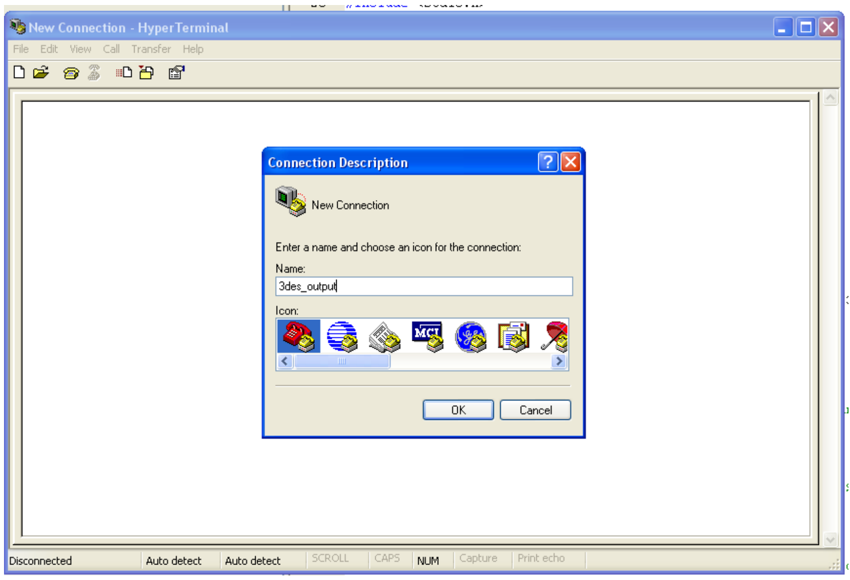
```
18  #endif
19  #endif
20  #include <stdio.h>
21  #include "co.h"
22
23  #include "des.h"
24
25  /* 3DES constants, don't change these */
26  #define BLOCKSIZE 8        /* unsigned chars per block */
27  #define KS_DEPTH 48        /* key pairs */
28
29  #ifdef IMPULSE_C_TARGET
30  #define printf xil_printf
31  #ifdef TIMED_TEST
32  XTmrCtr TimerCounter;
33  #endif
34  #endif
35
36  extern co_architecture co_initialize(void *);
37
38  /* Block data for C process */
39  static unsigned char Blocks[]={0x6f,0x98,0x26,0x35,0x02,0xc9,0x83,0xd7};
40  static unsigned long Iterations=1000;
41  int Asmversion;
42
43  #include "sp.c"
44
```

Now you need to initialize the project. Right click the project's name and select "Mark to Initialize BRAMs". If TestApp_Memory is initialized, de-initialize via the same process.


Xilinx Platform Studio - C:/temp/3DES_test/EDK/system.xmp
File Edit View Project Hardware Software Device Configuration Debu
Project Information Area
Project Applications IP Catalog
Software Projects
Add Software Application Project...
Default: ppc405_0_bootloop
Default: ppc405_1_bootloop
Project: TestApp_Memory
Processor: ppc405_0
Executable: C:\temp\3DES_test\EDK\TestApp_Memory\executable.elf
Compiler Options
Sources
Headers
Set Compiler Options...
Mark to Initialize BRAMs
Build Project
Clean Project
Delete Project...
Make Project Inactive
Generate Linker Script...
\3des\executable.elf
EDK\TestApp_Memory\src\TestAp
es_sw.c
_init.c
Headers

Step 5: Run the Test Application

Before you run the application, make sure HyperTerminal is running and a RS232 cable is connected between the board and computer. To setup HyperTerminal, simply follow the screenshots below.


New Connection - HyperTerminal

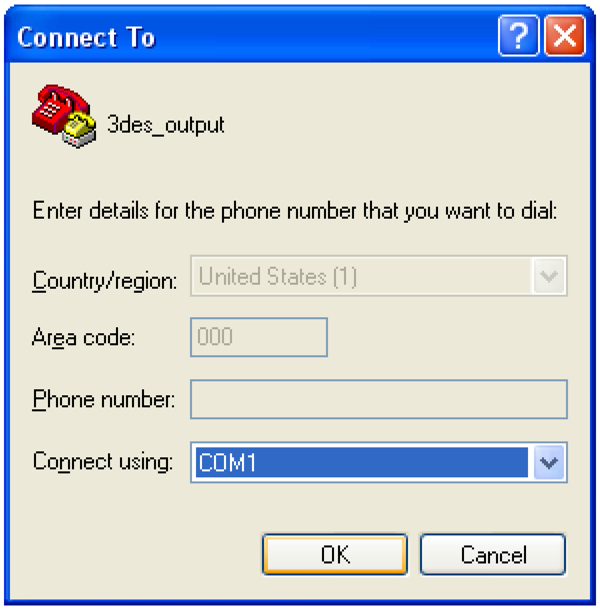
Connection Description

New Connection

Enter a name and choose an icon for the connection:

Name:
3des_output

Icon:


Connect To

3des_output

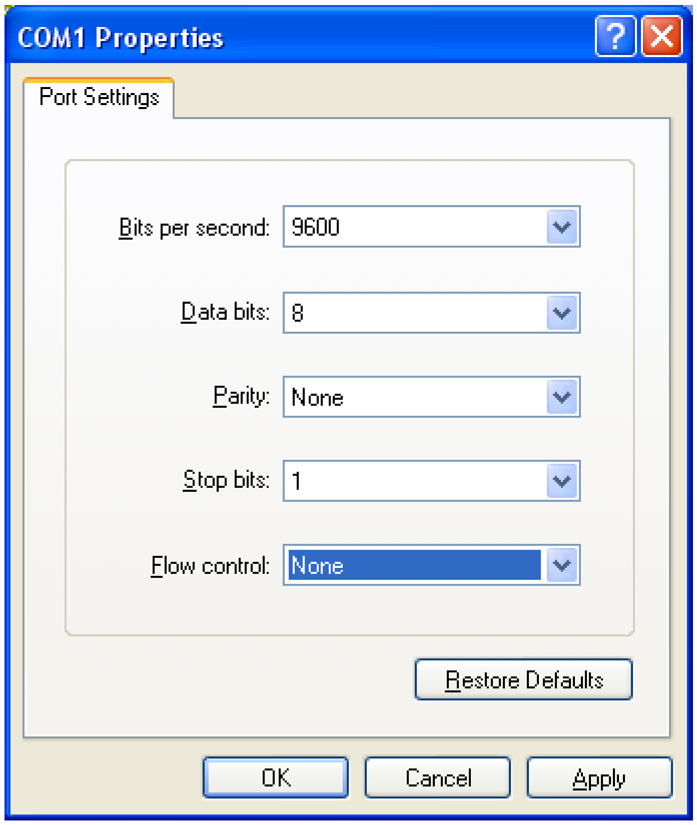
Enter details for the phone number that you want to dial:

Country/region: United States (1)

Area code: 000

Phone number:

Connect using: COM1


COM1 Properties

Port Settings

Bits per second: 9600

Data bits: 8

Parity: None

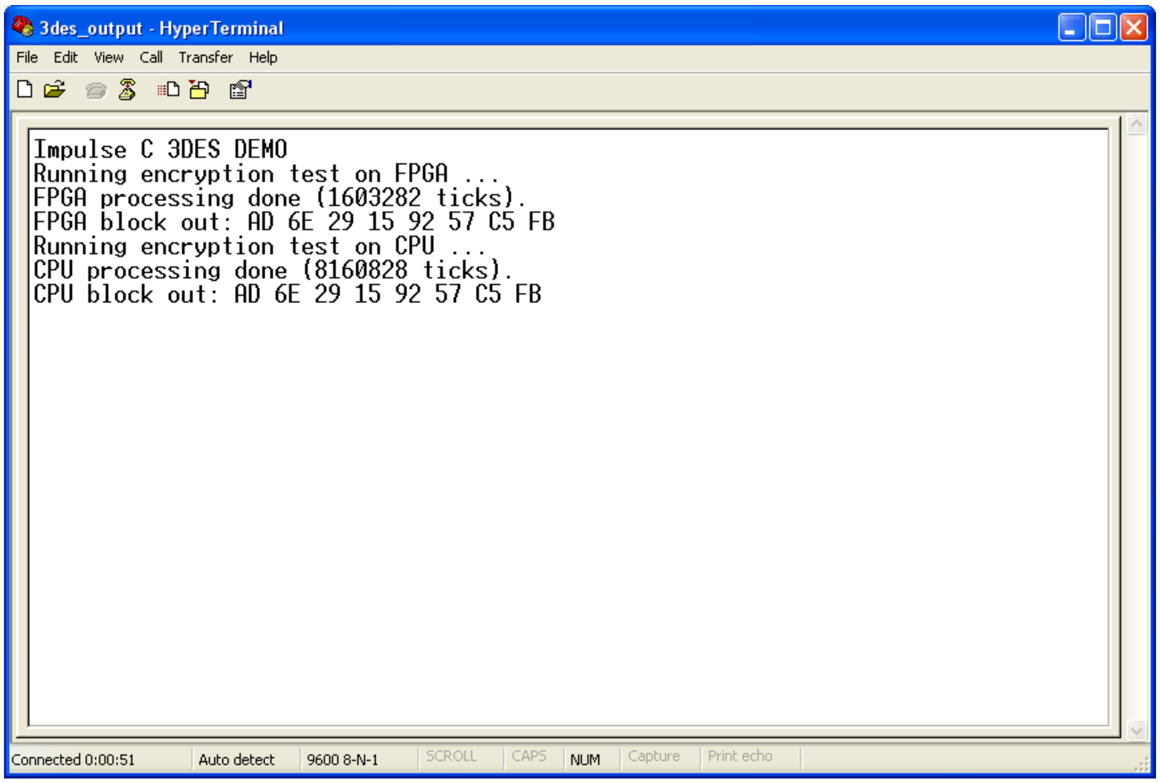
Stop bits: 1

Flow control: None

Once hyperterminal is setup, you can go back to XPS and click download bitstream. This will download the hardware to the board and run the software.

Step 6: Output of Test

Once the bitstream has completed downloading, you should see the following results on the HyperTerminal.


3des_output - HyperTerminal
File Edit View Call Transfer Help

```
Impulse C 3DES DEMO
Running encryption test on FPGA ...
FPGA processing done (1603282 ticks).
FPGA block out: AD 6E 29 15 92 57 C5 FB
Running encryption test on CPU ...
CPU processing done (8160828 ticks).
CPU block out: AD 6E 29 15 92 57 C5 FB
```

APPENDIX E

Software based 3DES

This appendix contains the code that comprises our software implementation of 3DES. The code is comprised of four files: mpi_soft.c, deskey.c, sp.c, and des.h.

mpi_soft.c

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <vector>
#include <string.h>
#include <fstream>
#include <mpi.h>

#include "des.h"
#include "sp.c"
#include "deskey.c"

using namespace std;
int NumBlocks;
vector <unsigned char> Blocks;

/* Keyschedule */
DES3_KS Ks;

#define F(l,r,key){\
   work = ((r >> 4) | (r << 28)) ^ key[0];\
   l ^= Spbox[6][work & 0x3f];\
   l ^= Spbox[4][(work >> 8) & 0x3f];\
   l ^= Spbox[2][(work >> 16) & 0x3f];\
   l ^= Spbox[0][(work >> 24) & 0x3f];\
   work = r ^ key[1];\
   l ^= Spbox[7][work & 0x3f];\
   l ^= Spbox[5][(work >> 8) & 0x3f];\
   l ^= Spbox[3][(work >> 16) & 0x3f];\
   l ^= Spbox[1][(work >> 24) & 0x3f];\
}

void get_data(int rank, int size, string filename)
{
  vector <int> temp;
  int temp_size=0;
  MPI_Status status;
  /* have p0 get data at end to pick up extras */
  if(rank==0)
  {
   int start, end, total;
   int tBlocks;
   int current_rank=1;
   int p0_size=0;
   int c_size=0;
   ifstream infile;
   infile.open(filename.c_str());
   start=infile.tellg();
   infile.seekg(0, ios::end);
   end=infile.tellg();
   infile.seekg(0, ios::beg);
   total=end-start;

   /* take total and right shift by 3, eg divide by 8 */
   tBlocks=total/DES_BLOCKSIZE;
   if(total % DES_BLOCKSIZE)
     tBlocks++; /* tells total number of blocks */

   NumBlocks=tBlocks / size;
   MPI_Bcast(&NumBlocks, 1, MPI_INT,0, MPI_COMM_WORLD);

   temp_size=NumBlocks*DES_BLOCKSIZE;
   NumBlocks=tBlocks-(size-1)*NumBlocks;
   /* Data on p0 will be zero padded to ensure that it
               contains a number of elements divisiable by 8 */
   p0_size=total-temp_size*(size-1);
   Blocks.resize(NumBlocks*DES_BLOCKSIZE,0);
   for(int i=1; i<=size; i++)
   {
     current_rank=i%size; /* gives correct rank to receive from */
     if(current_rank>0)
       c_size=temp_size;
     else
     {
       c_size=p0_size;
     }
     for(int j=0; j<c_size; j++)
     {
       Blocks[j]=infile.get();

     }
     if(current_rank!=0)
       MPI_Send(&Blocks.front(), temp_size, MPI_CHAR, current_rank, 0,
MPI_COMM_WORLD);
   }

  }
  else
  {
   MPI_Bcast(&NumBlocks, 1, MPI_INT, 0, MPI_COMM_WORLD);
   temp_size=NumBlocks*DES_BLOCKSIZE; /* NumBlocks * 8, tells number of
elements */
   Blocks.resize(temp_size);
   MPI_Recv(&Blocks.front(), temp_size,
MPI_CHAR,0,0,MPI_COMM_WORLD,&status);

  }
}

// This is the plain C 3DES process.  It reads the keyschedule and block
// inputs from global variables initialized in the des_producer process
// and does all processing using standard C code.
//
//
void des_c(int rank,int size, string filename2)
{
  int i;
  unsigned int blockCount = 0;
  int current_rank;
  unsigned char block[8];
  unsigned long left,right,work;
  vector <unsigned char> solution;
  vector <unsigned char> temp;
  int write_id;
  MPI_Status status;
  while( blockCount<NumBlocks ) {
    for ( i = 0; i < DES_BLOCKSIZE; i++ ) { /* DES_BLOCKSIZE is always 8 */
      block[i]=Blocks[blockCount * DES_BLOCKSIZE + i];
    }

    // Process the block...
    // Read input block and place in left/right in big-endian order
    //
    left = ((unsigned long)block[0] << 24)
           | ((unsigned long)block[1] << 16)
           | ((unsigned long)block[2] << 8)
           | (unsigned long)block[3];
    right = ((unsigned long)block[4] << 24)
           | ((unsigned long)block[5] << 16)
           | ((unsigned long)block[6] << 8)
           | (unsigned long)block[7];

//   Hoey's clever initial permutation algorithm, from Outerbridge
//   (see Schneier p 478)
//
//   The convention here is the same as Outerbridge: rotate each
//   register left by 1 bit, i.e., so that "left" contains permuted
//   input bits 2, 3, 4, ... 1 and "right" contains 33, 34, 35, ... 32
//   (using origin-1 numbering as in the FIPS). This allows us to avoid
//   one of the two rotates that would otherwise be required in each of
//   the 16 rounds.
//
    work = ((left >> 4) ^ right) & 0x0f0f0f0f;
    right ^= work;
    left ^= work << 4;
    work = ((left >> 16) ^ right) & 0xffff;
    right ^= work;
```

```
left ^= work << 16;
work = ((right >> 2) ^ left) & 0x33333333;
left ^= work;
right ^= (work << 2);
work = ((right >> 8) ^ left) & 0xff00ff;
left ^= work;
right ^= (work << 8);
right = (right << 1) | (right >> 31);
work = (left ^ right) & 0xaaaaaaaa;
left ^= work;
right ^= work;
left = (left << 1) | (left >> 31);

/* First key */
F(left,right,Ks[0]);
F(right,left,Ks[1]);
F(left,right,Ks[2]);
F(right,left,Ks[3]);
F(left,right,Ks[4]);
F(right,left,Ks[5]);
F(left,right,Ks[6]);
F(right,left,Ks[7]);
F(left,right,Ks[8]);
F(right,left,Ks[9]);
F(left,right,Ks[10]);
F(right,left,Ks[11]);
F(left,right,Ks[12]);
F(right,left,Ks[13]);
F(left,right,Ks[14]);
F(right,left,Ks[15]);

/* Second key (must be created in opposite mode to first key) */
F(right,left,Ks[16]);
F(left,right,Ks[17]);
F(right,left,Ks[18]);
F(left,right,Ks[19]);
F(right,left,Ks[20]);
F(left,right,Ks[21]);
F(right,left,Ks[22]);
F(left,right,Ks[23]);
```

```
F(right,left,Ks[24]);
F(left,right,Ks[25]);
F(right,left,Ks[26]);
F(left,right,Ks[27]);
F(right,left,Ks[28]);
F(left,right,Ks[29]);
F(right,left,Ks[30]);
F(left,right,Ks[31]);

/* Third key */
F(left,right,Ks[32]);
F(right,left,Ks[33]);
F(left,right,Ks[34]);
F(right,left,Ks[35]);
F(left,right,Ks[36]);
F(right,left,Ks[37]);
F(left,right,Ks[38]);
F(right,left,Ks[39]);
F(left,right,Ks[40]);
F(right,left,Ks[41]);
F(left,right,Ks[42]);
F(right,left,Ks[43]);
F(left,right,Ks[44]);
F(right,left,Ks[45]);
F(left,right,Ks[46]);
F(right,left,Ks[47]);

/* Inverse permutation, also from Hoey via Outerbridge and Schneier */
right = (right << 31) | (right >> 1);
work = (left ^ right) & 0xaaaaaaaa;
left ^= work;
right ^= work;
left = (left >> 1) | (left  << 31);
work = ((left >> 8) ^ right) & 0xff00ff;
right ^= work;
left ^= work << 8;
work = ((left >> 2) ^ right) & 0x33333333;
right ^= work;
left ^= work << 2;
work = ((right >> 16) ^ left) & 0xffff;
```

```
  left ^= work;
  right ^= work << 16;
  work = ((right >> 4) ^ left) & 0x0f0f0f0f;
  left ^= work;
  right ^= work << 4;

  /* Put the block into the output stream with final swap */
  block[0] = (int) (right >> 24);
  block[1] = (int) (right >> 16);
  block[2] = (int) (right >> 8);
  block[3] = (int) right;
  block[4] = (int) (left >> 24);
  block[5] = (int) (left >> 16);
  block[6] = (int) (left >> 8);
  block[7] = (int) left;

    for (i=0; i<DES_BLOCKSIZE; i++) {
      solution.push_back(block[i]);
    }

  ++blockCount;
}
// if there is more than one processor, the 2nd processor receives date,
// otherwise, p0 keeps all data
if(size>1)
  write_id=1;
else
  write_id=0;


if(rank==write_id)
{
  int size_temp=0;
  ofstream outfile;
  outfile.open(filename2.c_str());
  // receive data and store in a vector
  for(int i=1; i<=size; i++)
  {
    current_rank=i%size; /* gives correct rank to receive from */
```

```
      if(current_rank==rank)
      {
        for(int j=0;j<solution.size();j++)
        {
          outfile.setf(ios::hex,ios::basefield);
          outfile<<solution[j];
        }
      }
      else
      {
        MPI_Recv(&size_temp,1, MPI_INT, current_rank, 0,
MPI_COMM_WORLD,&status);
        temp.resize(size_temp);
        MPI_Recv(&temp.front(),size_temp, MPI_CHAR, current_rank, 0,
MPI_COMM_WORLD, &status);
        for(int j=0; j<size_temp;j++)
        {
          outfile.setf(ios::hex,ios::basefield);
          outfile<<temp[j];
        }
      }
    }

    outfile.close();
  }
  else
  {
    /* send my solution to process 0 */
    int tSize=solution.size();
    MPI_Send(&tSize, 1, MPI_INT, write_id, 0, MPI_COMM_WORLD);
    MPI_Send(&solution.front(),tSize , MPI_CHAR, write_id, 0, MPI_COMM_WORLD);
  }
}

int main(int argc, char *argv[])
{
  int npes, myrank;
  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &npes);
  MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
  int crypt_choice;
  unsigned char * key = (unsigned char *) "Gflk jqo40978J0dmm$%@878"; /* 24 bytes
*/
  string filename, filename2;
  filename =argv[1];
  filename2 = argv[2];
  // 0 for encryption, 1 for decryption
  crypt_choice=atoi(argv[3]);
  des3key(Ks, key, crypt_choice);


  /* Grab data and send out*/
  get_data(myrank, npes, filename);

  /* encryption/decryption step */
  des_c(myrank, npes, filename2);

  MPI_Finalize();
  return(0);
}
```

deskey.c

```
// Copyright(c) 2003-2007 Impulse Accelerated Technologies, Inc.
// All rights reserved.
// www.ImpulseC.com
//
// This source file may be used and distributed without restriction provided
// that this copyright notice is not removed from the file and that any
// derivative work contains this copyright notice.
//
// Portable C code to create DES key schedules from user-provided keys
// This doesn't have to be fast unless you're cracking keys or UNIX
// passwords
//

#include <iostream>
#include <string.h>
#include "des.h"

using namespace std;
/* Key schedule-related tables from FIPS-46 */

/* permuted choice table (key) */
static unsigned char pc1[] = {
        57, 49, 41, 33, 25, 17,  9,
         1, 58, 50, 42, 34, 26, 18,
        10,  2, 59, 51, 43, 35, 27,
        19, 11,  3, 60, 52, 44, 36,

        63, 55, 47, 39, 31, 23, 15,
         7, 62, 54, 46, 38, 30, 22,
        14,  6, 61, 53, 45, 37, 29,
        21, 13,  5, 28, 20, 12,  4
};

/* number left rotations of pc1 */
static unsigned char totrot[] = {
        1,2,4,6,8,10,12,14,15,17,19,21,23,25,27,28
};

/* permuted choice key (table) */
```

```
static unsigned char pc2[] = {
        14, 17, 11, 24,  1,  5,
         3, 28, 15,  6, 21, 10,
        23, 19, 12,  4, 26,  8,
        16,  7, 27, 20, 13,  2,
        41, 52, 31, 37, 47, 55,
        30, 40, 51, 45, 33, 48,
        44, 49, 39, 56, 34, 53,
        46, 42, 50, 36, 29, 32
};

/* End of DES-defined tables */

/* bit 0 is left-most in byte */
static int bytebit[] = {
        0200,0100,040,020,010,04,02,01
};


// Generate key schedule for encryption or decryption
// depending on the value of "decrypt"
//
void deskey(DES_KS k,unsigned char *key,int decrypt)
//DES_KS k;                     /* Key schedule array */
//unsigned char *key;           /* 64 bits (will use only 56) */
//int decrypt;                  /* 0 = encrypt, 1 = decrypt */
{
        unsigned char pc1m[56];         /* place to modify pc1 into */
        unsigned char pcr[56];          /* place to rotate pc1 into */
        register int i,j,l;
        int m;
        unsigned char ks[8];

        for (j=0; j<56; j++) {          /* convert pc1 to bits of key */
                l=pc1[j]-1;             /* integer bit location  */
                m = l & 07;             /* find bit              */
                pc1m[j]=(key[l>>3] &    /* find which key byte l is in */
                        bytebit[m])     /* and which bit of that byte */
                        ? 1 : 0;/* and store 1-bit result */
        }
        for (i=0; i<16; i++) {          /* key chunk for each iteration */
                memset(ks,0,sizeof(ks));        /* Clear key schedule */
                for (j=0; j<56; j++)    /* rotate pc1 the right amount */
                        pcr[j] = pc1m[(l=j+totrot[decrypt? 15-i : i])<(j<28? 28 : 56) ? l:
l-28];
                        /* rotate left and right halves independently */
                for (j=0; j<48; j++){  /* select bits individually */
                        /* check bit that goes to ks[j] */
                        if (pcr[pc2[j]-1]){
                                /* mask it in if it's there */
                                l= j % 6;
                                ks[j/6] |= bytebit[l] >> 2;
                        }
                }
                /* Now convert to packed odd/even interleaved form */
                k[i][0] = ((long)ks[0] << 24)
                 | ((long)ks[2] << 16)
                 | ((long)ks[4] << 8)
                 | ((long)ks[6]);
                k[i][1] = ((long)ks[1] << 24)
                 | ((long)ks[3] << 16)
                 | ((long)ks[5] << 8)
                 | ((long)ks[7]);
        }
}

// Generate key schedule for triple DES in E-D-E (or D-E-D) mode.
//
// The key argument is taken to be 24 bytes. The first 8 bytes are K1
// for the first stage, the second 8 bytes are K2 for the middle stage
// and the third 8 bytes are K3 for the last stage
//
void
des3key(DES3_KS k,unsigned char *key, int decrypt)
{
        if(!decrypt){
                deskey(&k[0],&key[0],0);
                deskey(&k[16],&key[8],1);
                deskey(&k[32],&key[16],0);
        } else {
                deskey(&k[32],&key[0],1);
                deskey(&k[16],&key[8],0);
                deskey(&k[0],&key[16],1);
        }
}
```

sp.c

```
#define SPBOX_X 8
#define SPBOX_Y 64
unsigned long Spbox[8][64] = {
0x01010400,0x00000000,0x00010000,0x01010404,
0x01010004,0x00010404,0x00000004,0x00010000,
0x00000400,0x01010400,0x01010404,0x00000400,
0x01000404,0x01010004,0x01000000,0x00000004,
0x00000404,0x01000400,0x01000400,0x00010400,
0x00010400,0x01010000,0x01010000,0x01000404,
0x00010004,0x01000004,0x01000004,0x00010004,
0x00000000,0x00000404,0x00010404,0x01000000,
0x00010000,0x01010404,0x00000004,0x01010000,
0x01010400,0x01000000,0x01000000,0x00000400,
0x01010004,0x00010000,0x00010400,0x01000004,
0x00000400,0x00000004,0x01000404,0x00010404,
0x01010404,0x00010004,0x01010000,0x01000404,
0x01000004,0x00000404,0x00010404,0x01010400,
0x00000404,0x01000400,0x01000400,0x00000000,
0x00010004,0x00010400,0x00000000,0x01010004,
0x80108020,0x80008000,0x00008000,0x00108020,
0x00100000,0x00000020,0x80100020,0x80008020,
0x80000020,0x80108020,0x80108000,0x80000000,
0x80008000,0x00100000,0x00000020,0x80100020,
0x00108000,0x00100020,0x80008020,0x00000000,
0x80000000,0x00008000,0x00108020,0x80100000,
0x00100020,0x80000020,0x00000000,0x00108000,
0x00008020,0x80108000,0x80100000,0x00008020,
0x00000000,0x00108020,0x80100020,0x00100000,
0x80008020,0x80100000,0x80108000,0x00008000,
0x80100000,0x80008000,0x00000020,0x80108020,
0x00108020,0x00000020,0x00008000,0x80000000,
0x00008020,0x80108000,0x00100000,0x80000020,
0x00100020,0x80008020,0x80000020,0x00100020,
0x00108000,0x00000000,0x80008000,0x00008020,
0x80000000,0x80100020,0x80108020,0x00108000,
0x00000208,0x08020200,0x00000000,0x08020008,
0x08000200,0x00000000,0x00020208,0x08000200,
0x00020008,0x08000008,0x08000008,0x00020000,
0x08020208,0x00020008,0x08020000,0x00000208,
0x08000000,0x00000008,0x08020200,0x00000200,
0x00020200,0x08020000,0x08020008,0x00020208,
0x08000208,0x00020200,0x00020000,0x08000208,
0x00000008,0x08020208,0x00000200,0x08000000,
0x08020200,0x08000000,0x00020008,0x00000208,
0x00020000,0x08020200,0x08000200,0x00000000,
0x00000200,0x00020008,0x08020208,0x08000200,
0x08000008,0x00000200,0x00000000,0x08020008,
0x08000208,0x00020000,0x08000000,0x08020208,
0x00000008,0x00020208,0x00020200,0x08000008,
0x08020000,0x08000208,0x00000208,0x08020000,
0x00020208,0x00000008,0x08020008,0x00020200,
0x00802001,0x00002081,0x00002081,0x00000080,
0x00802080,0x00800081,0x00800001,0x00002001,
0x00000000,0x00802000,0x00802000,0x00802081,
0x00000081,0x00000000,0x00800080,0x00800001,
0x00000001,0x00002000,0x00800000,0x00802001,
0x00000080,0x00800000,0x00002001,0x00002080,
0x00800081,0x00000001,0x00002080,0x00800080,
0x00002000,0x00802080,0x00802081,0x00000081,
0x00800080,0x00800001,0x00802000,0x00802081,
0x00000081,0x00000000,0x00000000,0x00802000,
0x00002080,0x00800080,0x00800081,0x00000001,
0x00802001,0x00002081,0x00002081,0x00000080,
0x00802081,0x00000081,0x00000001,0x00002000,
0x00800001,0x00002001,0x00802080,0x00800081,
0x00002001,0x00002080,0x00800000,0x00802001,
0x00000080,0x00800000,0x00002000,0x00802080,
0x00000100,0x02080100,0x02080000,0x42000100,
0x00080000,0x00000100,0x40000000,0x02080000,
0x40080100,0x00080000,0x02000100,0x40080100,
0x42000100,0x42080000,0x00080100,0x40000000,
0x02000000,0x40080000,0x40080000,0x00000000,
0x40000100,0x42080100,0x42080100,0x02000100,
0x42080000,0x40000100,0x00000000,0x42000000,
0x02080100,0x02000000,0x42000000,0x00080100,
0x00080000,0x42000100,0x00000100,0x02000000,
0x40000000,0x02080000,0x42000100,0x40080100,
0x02000100,0x40000000,0x42080000,0x02080100,
0x40080100,0x00000100,0x02000000,0x42080000,
0x42080100,0x00080100,0x42000000,0x42080100,
0x02080000,0x00000000,0x40080000,0x42000000,
0x00080100,0x02000100,0x40000100,0x00080000,
0x00000000,0x40080000,0x02080100,0x40000100,
0x20000010,0x20400000,0x00004000,0x20404010,
0x20400000,0x00000010,0x20404010,0x00400000,
0x20004000,0x00404010,0x00400000,0x20000010,
0x00400010,0x20004000,0x20000000,0x00004010,
0x00000000,0x00400010,0x20004010,0x00004000,
0x00404000,0x20004010,0x00000010,0x20400010,
0x20400010,0x00000000,0x00404010,0x20404000,
0x00004010,0x00404000,0x20404000,0x20000000,
0x20004000,0x00000010,0x20400010,0x00404000,
0x20404010,0x00400000,0x00004010,0x20000010,
0x00400000,0x20004000,0x20000000,0x00004010,
0x20000010,0x20404010,0x00404000,0x20400000,
0x00404010,0x20404000,0x00000000,0x20400010,
0x00000010,0x00004000,0x20400000,0x00404010,
0x00004000,0x00400010,0x20004010,0x00000000,
0x20404000,0x20000000,0x00400010,0x20004010,
0x00200000,0x04200002,0x04000802,0x00000000,
0x00000800,0x04000802,0x00200802,0x04200800,
0x04200802,0x00200000,0x00000000,0x04000002,
0x00000002,0x04000000,0x04200002,0x00000802,
0x04000800,0x00200802,0x00200002,0x04000800,
0x04000002,0x04200000,0x04200800,0x00200002,
0x04200000,0x00000800,0x00000802,0x04200802,
0x00200800,0x00000002,0x04000000,0x00200800,
0x04000000,0x00200800,0x00200000,0x04000802,
0x04000802,0x04200002,0x04200002,0x00000002,
0x00200002,0x04000000,0x04000800,0x00200000,
0x04200800,0x00000802,0x00200802,0x04200800,
0x00000802,0x04000002,0x04200802,0x04200000,
0x00200800,0x00000000,0x00000002,0x04200802,
0x00000000,0x00200802,0x04200000,0x00000800,
0x04000002,0x04000800,0x00000800,0x00200002,
0x10001040,0x00001000,0x00040000,0x10041040,
0x10000000,0x10001040,0x00000040,0x10000000,
0x00040040,0x10040000,0x10041040,0x00041000,
0x10041000,0x00041040,0x00001000,0x00000040,
0x10040000,0x10000040,0x10001000,0x00001040,
0x00041000,0x00040040,0x10040040,0x10041000,
0x00001040,0x00000000,0x00000000,0x10040040,
0x10000040,0x10001000,0x00041040,0x00040000,
0x00041040,0x00040000,0x10041000,0x00001000,
0x00000040,0x10040040,0x00001000,0x00041040,
0x10001000,0x00000040,0x10000040,0x10040000,
0x10040040,0x10000000,0x00040000,0x10001040,
0x00000000,0x10041040,0x00040040,0x10000040,
0x10040000,0x10001000,0x10001040,0x00000000,
0x10041040,0x00041000,0x00041000,0x00001040,
0x00001040,0x00040040,0x10000000,0x10041000,
};
```

des.h

```
// Copyright(c) 2003-2007 Impulse Accelerated Technologies, Inc.
// All rights reserved.
// www.ImpulseC.com
//
// This source file may be used and distributed without restriction provided
// that this copyright notice is not removed from the file and that any
// derivative work contains this copyright notice.

/* Signal values that indicate which task to do */
#define DES_ENCRYPT   0
#define DES_DECRYPT   1

/* 3DES constants, don't change these */
#define DES_BLOCKSIZE 8           /* unsigned chars per block */
#define DES_KS_DEPTH 48           /* key pairs */
#define SPBOX_X 8
#define SPBOX_Y 64

typedef unsigned long DES_KS[16][2];       /* Single-key DES key schedule */
typedef unsigned long DES3_KS[48][2];      /* Triple-DES key schedule */

/* In deskey.c: */
void deskey(DES_KS,unsigned char *,int);
void des3key(DES3_KS,unsigned char *,int);
```

APPENDIX F

Adding the 3DES IP CORE into the Base Design

This appendix describes how to add the 3DES IP Core to the Xilinx base design. I will assume that the 3DES project has already been copied over to the Redhat machine, or already exists on the Redhat machine.

First open a terminal window and change your directory to where the project is stored.

Next, in the terminal you will need to run “icProj2make.pl 3des.icProj”.

Following, run “Make –f _Makefile build”.

Next you will need to modify the _Makefile.defs so you are able to export the hardware to your target project. Change the *$Option “GenCodeHWExportDir”* line to point to your project directory. For example, *$Option “GenCodeHWExportDir=/home/user/3des_proj”.*

Finally run “Make –f _Makefile export_build” to export the hardware to your project.

Once this is done, simply start XPS and rescan the user repositories and add the new IP Core. This step is just like those displayed in Appendix D.

APPENDIX G

Code for the Device Driver

This appendix contains the code that comprises the device driver for the 3DES IP Core, as well as my reduced application code for XPS. Appendix H contains how to compile this code and load it onto a board.

DEVICE DRIVER CODE

Des.c

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/slab.h>
#include <linux/fs.h>
#include <linux/errno.h>
#include <linux/types.h>
#include <linux/proc_fs.h>
#include <linux/fcntl.h> /* O_ACCMODE */
#include <linux/ioport.h>
#include <asm/system.h> /* cli(), *_flags */
#include <asm/uaccess.h> /* copy_from/to_user */
#include <asm/io.h> /* inb, outb */
#include <linux/delay.h>

#include "des.h" /* IOCTL cmds */
#define reg_off_1 0x00000010 /* offset to point to blocks_out_addr  */
#define reg_off_2 0x00000020 /* offset to point to input_stream_addr  */

#if defined __GNUC__
#  define SYNCHRONIZE_IO __asm__ volatile ("eieio")
#elif defined __DCC__
#  define SYNCHRONIZE_IO __asm volatile(" eieio")
#else
#  define SYNCHRONIZE_IO
```

```
#endif

MODULE_LICENSE("Dual BSD/GPL");

static const unsigned phy_add = 0xc9c00000;
/* Physical address as dictated by XPS address */
static const unsigned remapSize = 0x10000;
volatile static unsigned virt_add;


static int des_ioctl(struct inode *inode, /* see include/linux/fs.h */
                     struct file *file, /* ditto */
                     unsigned int ioctl_num, /* number and param for
ioctl */
                     unsigned long ioctl_param);

static struct file_operations des_fops = {
  .ioctl = des_ioctl
};

static int des_major = MAJOR_NUM;

static int des_init(void)
{
  int result;
  result = register_chrdev(des_major, "des", &des_fops);
  virt_add = (unsigned) ioremap(phy_add, remapSize);
  return result;
}

static void des_exit(void)
{
  iounmap((void *)virt_add);
  unregister_chrdev(des_major, "des");
}

static int des_ioctl(struct inode *inode, /* see include/linux/fs.h */
                     struct file *file, /* ditto */
                     unsigned int ioctl_num, /* number and param for
ioctl */
```

```
                    unsigned long ioctl_param)
{
 u32 num32;
 u8 num8;
 u8 result;
 u32 err;

 /*
 * Switch according to the ioctl called
 */
 switch (ioctl_num) {
  case IOCTL_WRITE_CONFIG:
        // Write Encryption information to IP Core
   /*
    * Receive pointer to a u32 value
    */

   copy_from_user(&num32,   (u32 *)ioctl_param,     4);
  *(volatile u32*) virt_add           = num32;
   SYNCHRONIZE_IO;
   break;

  case IOCTL_CLOSE_CONFIG:
        // Tell Hardware we are done sending encryption information
   *(volatile u32*) (virt_add+8)        = 0;
   break;
  case IOCTL_WRITE_BLOCK:  // Write plaintext to hardware
    /*
    * Receive pointer to user buffer to hold u32 value
    */
   copy_from_user(&num8,   (u8 *)ioctl_param,     1);
   *(volatile u32*) (virt_add+reg_off_1)           = num8;
   SYNCHRONIZE_IO;
   break;
  case IOCTL_GET_INPUT:  //Get the encrypted values from the hardware
   while((err=((*(volatile u32 *)(virt_add+reg_off_2+8))&0x03))==0);
         result=*(volatile u32 *)(virt_add+reg_off_2);
   err=(err!=0x01);
   copy_to_user((void *)ioctl_param, (void *)&result, sizeof(u8));
   SYNCHRONIZE_IO;
```

```
      break;

  }

  return 0;
}

module_init(des_init);
module_exit(des_exit);
```

des.h

```
#ifndef des_H
#define des_H

#include <linux/ioctl.h>

#define MAJOR_NUM 76

#define IOCTL_WRITE_CONFIG    _IOW(MAJOR_NUM, 0, u32 *)
#define IOCTL_CLOSE_CONFIG    _IOR(MAJOR_NUM, 1, u32 *)
#define IOCTL_WRITE_BLOCK     _IOR(MAJOR_NUM, 2, u8 *)
#define IOCTL_GET_INPUT       _IOR(MAJOR_NUM, 3, u8 *)

#endif
```

XPS APPLICATION CODE
The application code consists of three files: sp.c, main2.c, and des_def.h

main2.c

```
#include "des_def.h"
#include "sp.c"
//#include "xio.h"
#include <stdio.h>
#define XPAR_PLB_DES_0_BASEADDR 0xC9C00000 //can ignore xparameters.h
#define config_out_addr XPAR_PLB_DES_0_BASEADDR+0
#define blocks_out_addr XPAR_PLB_DES_0_BASEADDR+16
#define input_stream_addr XPAR_PLB_DES_0_BASEADDR+32

#if defined __GNUC__
#  define SYNCHRONIZE_IO __asm__ volatile ("eieio")
#elif defined __DCC__
#  define SYNCHRONIZE_IO __asm volatile(" eieio")
#else
#  define SYNCHRONIZE_IO
#endif

typedef unsigned long   Xuint32;    /**< unsigned 32-bit */
#define printf xil_printf
static unsigned char Blocks[]={0x6f,0x98,0x26,0x35,0x02,0xc9,0x83,0xd7};

void des_test()
{
  int i, k;
  unsigned char block[8];
  unsigned char blockElement;
  unsigned long data,err;

  for ( k = 0; k < 2; k++ ) {
    for ( i = 0; i < KS_DEPTH; i++ ) {
      data=Ks[i][k];
                    *(volatile Xuint32 *)(config_out_addr) = data; SYNCHRONIZE_IO;
    }
  }

  for ( i = 0; i < SPBOX_X; i++ ) {
   for ( k = 0; k < SPBOX_Y; k++ ) {
     data=Spbox[i][k];
                   *(volatile Xuint32 *)(config_out_addr) = data; SYNCHRONIZE_IO;

   }
  }
  *(volatile Xuint32 *)(config_out_addr+8) = 0; SYNCHRONIZE_IO;


  for ( k = 0; k < BLOCKSIZE; k++ ) {
    blockElement = Blocks[k];
    *(volatile Xuint32 *)(blocks_out_addr) = blockElement; SYNCHRONIZE_IO;

  }

  for ( k = 0; k < BLOCKSIZE; k++ ) {

    while((err=((*(volatile Xuint32 *)(input_stream_addr+8))&0x03))==0);
            blockElement=*(volatile Xuint32 *)(input_stream_addr);
      err=(err!=0x01);
    block[k]=blockElement;
  }
  *(volatile Xuint32 *)(blocks_out_addr+8) = 0; SYNCHRONIZE_IO;



  printf("FPGA block out:");
  for (i=0; i<BLOCKSIZE; i++) {
    printf(" %02x",block[i]);
  }
  printf("\n\r");
}

int main()
{
  unsigned char * key = (unsigned char *) "Gflk jqo40978J0dmm$%@878"; /* 24 bytes
*/
  des3key(Ks, key, 0);  /* Create a keyschedule for encryption */
  printf("Running NEW encryption test on FPGA ...\n\r");
  des_test();
  return 0;
}
```

sp.c

```
#define SPBOX_X 8
#define SPBOX_Y 64
unsigned long Spbox[8][64] = {
0x01010400,0x00000000,0x00010000,0x01010404,
0x01010004,0x00010404,0x00000004,0x00010000,
0x00000400,0x01010400,0x01010404,0x00000400,
0x01000404,0x01010004,0x01000000,0x00000004,
0x00000404,0x01000400,0x01000400,0x00010400,
0x00010400,0x01010000,0x01010000,0x01000404,
0x00010004,0x01000004,0x01000004,0x00010004,
0x00000000,0x00000404,0x00010404,0x01000000,
0x00010000,0x01010404,0x00000004,0x01010000,
0x01010400,0x01000000,0x01000000,0x00000400,
0x01010004,0x00010000,0x00010400,0x01000004,
0x00000400,0x00000004,0x01000404,0x00010404,
0x01010404,0x00010004,0x01010000,0x01000404,
0x01000004,0x00000404,0x00010404,0x01010400,
0x00000404,0x01000400,0x01000400,0x00000000,
0x00010004,0x00010400,0x00000000,0x01010004,
0x80108020,0x80008000,0x00008000,0x00108020,
0x00100000,0x00000020,0x80100020,0x80008020,
0x80000020,0x80108020,0x80108000,0x80000000,
0x80008000,0x00100000,0x00000020,0x80100020,
0x00108000,0x00100020,0x80008020,0x00000000,
0x80000000,0x00008000,0x00108020,0x80100000,
0x00100020,0x80000020,0x00000000,0x00108000,
0x00008020,0x80108000,0x80100000,0x00008020,
0x00000000,0x00108020,0x80100020,0x00100000,
0x80008020,0x80100000,0x80108000,0x00008000,
0x80100000,0x80008000,0x00000020,0x80108020,
0x00108020,0x00000020,0x00008000,0x80000000,
0x00008020,0x80108000,0x00100000,0x80000020,
0x00100020,0x80008020,0x80000020,0x00100020,
0x00108000,0x00000000,0x80008000,0x00008020,
0x80000000,0x80100020,0x80108020,0x00108000,
0x00000208,0x08020200,0x00000000,0x08020008,
0x08000200,0x00000000,0x00020208,0x08000200,
0x00020008,0x08000008,0x08000008,0x00020000,
0x08020208,0x00020008,0x08020000,0x00000208,
0x08000000,0x00000008,0x08020200,0x00000200,
0x00020200,0x08020000,0x08020008,0x00020208,
0x08000208,0x00020200,0x00020000,0x08000208,
0x00000008,0x08020208,0x00000200,0x08000000,
0x08020200,0x08000000,0x00020008,0x00000208,
0x00020000,0x08020200,0x08000200,0x00000000,
0x00000200,0x00020008,0x08020208,0x08000200,
0x08000008,0x00000200,0x00000000,0x08020008,
0x08000208,0x00020000,0x08000000,0x08020208,
0x00000008,0x00020208,0x00020200,0x08000008,
0x08020000,0x08000208,0x00000208,0x08020000,
0x00020208,0x00000008,0x08020008,0x00020200,
0x00802001,0x00002081,0x00002081,0x00000080,
0x00802080,0x00800081,0x00800001,0x00002001,
0x00000000,0x00802000,0x00802000,0x00802081,
0x00000081,0x00000000,0x00800080,0x00800001,
0x00000001,0x00002000,0x00800000,0x00802001,
0x00000080,0x00800000,0x00002001,0x00002080,
0x00800081,0x00000001,0x00002080,0x00800080,
0x00002000,0x00802080,0x00802081,0x00000081,
0x00800080,0x00800001,0x00802000,0x00802081,
0x00000081,0x00000000,0x00000000,0x00802000,
0x00002080,0x00800080,0x00800081,0x00000001,
0x00802001,0x00002081,0x00002081,0x00000080,
0x00802081,0x00000081,0x00000001,0x00002000,
0x00800001,0x00002001,0x00802080,0x00800081,
0x00002001,0x00002080,0x00800000,0x00802001,
0x00000080,0x00800000,0x00002000,0x00802080,
0x00000100,0x02080100,0x02080000,0x42000100,
0x00080000,0x00000100,0x40000000,0x02080000,
0x40080100,0x00080000,0x02000100,0x40080100,
0x42000100,0x42080000,0x00080100,0x40000000,
0x02000000,0x40080000,0x40080000,0x00000000,
0x40000100,0x42080100,0x42080100,0x02000100,
0x42080000,0x40000100,0x00000000,0x42000000,
0x02080100,0x02000000,0x42000000,0x00080100,
0x00080000,0x42000100,0x00000100,0x02000000,
0x40000000,0x02080000,0x42000100,0x40080100,
0x02000100,0x40000000,0x42080000,0x02080100,
0x40080100,0x00000100,0x02000000,0x42080000,
0x42080100,0x00080100,0x42000000,0x42080100,
0x02080000,0x00000000,0x40080000,0x42000000,
0x00080100,0x02000100,0x40000100,0x00080000,
0x00000000,0x40080000,0x02080100,0x40000100,
0x20000010,0x20400000,0x00004000,0x20404010,
0x20400000,0x00000010,0x20404010,0x00400000,
0x20004000,0x00404010,0x00400000,0x20000010,
0x00400010,0x20004000,0x20000000,0x00004010,
0x00000000,0x00400010,0x20004010,0x00004000,
0x00404000,0x20004010,0x00000010,0x20400010,
0x20400010,0x00000000,0x00404010,0x20404000,
0x00004010,0x00404000,0x20404000,0x20000000,
0x20004000,0x00000010,0x20400010,0x00404000,
0x20404010,0x00400000,0x00004010,0x20000010,
0x00400000,0x20004000,0x20000000,0x00004010,
0x20000010,0x20404010,0x00404000,0x20400000,
0x00404010,0x20404000,0x00000000,0x20400010,
0x00000010,0x00004000,0x20400000,0x00404010,
0x00004000,0x00400010,0x20004010,0x00000000,
0x20404000,0x20000000,0x00400010,0x20004010,
0x00200000,0x04200002,0x04000802,0x00000000,
0x00000800,0x04000802,0x00200802,0x04200800,
0x04200802,0x00200000,0x00000000,0x04000002,
0x00000002,0x04000000,0x04200002,0x00000802,
0x04000800,0x00200802,0x00200002,0x04000800,
0x04000002,0x04200000,0x04200800,0x00200002,
0x04200000,0x00000800,0x00000802,0x04200802,
0x00200800,0x00000002,0x04000000,0x00200800,
0x04000000,0x00200800,0x00200000,0x04000802,
0x04000802,0x04200002,0x04200002,0x00000002,
0x00200002,0x04000000,0x04000800,0x00200000,
0x04200800,0x00000802,0x00200802,0x04200800,
0x00000802,0x04000002,0x04200802,0x04200000,
0x00200800,0x00000000,0x00000002,0x04200802,
0x00000000,0x00200802,0x04200000,0x00000800,
0x04000002,0x04000800,0x00000800,0x00200002,
0x10001040,0x00001000,0x00040000,0x10041040,
0x10000000,0x10001040,0x00000040,0x10000000,
0x00040040,0x10040000,0x10041040,0x00041000,
0x10041000,0x00041040,0x00001000,0x00000040,
0x10040000,0x10000040,0x10001000,0x00001040,
0x00041000,0x00040040,0x10040040,0x10041000,
0x00001040,0x00000000,0x00000000,0x10040040,
0x10000040,0x10001000,0x00041040,0x00040000,
0x00041040,0x00040000,0x10041000,0x00001000,
0x00000040,0x10040040,0x00001000,0x00041040,
0x10001000,0x00000040,0x10000040,0x10040000,
0x10040040,0x10000000,0x00040000,0x10001040,
0x00000000,0x10041040,0x00040040,0x10000040,
0x10040000,0x10001000,0x10001040,0x00000000,
0x10041040,0x00041000,0x00041000,0x00001040,
0x00001040,0x00040040,0x10000000,0x10041000,
};
```

Des_def.h

```
typedef unsigned long DES_KS[16][2];        /* Single-key DES key schedule */
typedef unsigned long DES3_KS[48][2];       /* Triple-DES key schedule */

/* In deskey.c: */
void deskey(DES_KS,unsigned char *,int);
void des3key(DES3_KS,unsigned char *,int);

/* In desport.c, desborl.cas or desgnu.s: */
void des(DES_KS,unsigned char *);
/* In des3port.c, des3borl.cas or des3gnu.s: */
void des3(DES3_KS,unsigned char *);

#define BLOCKSIZE 8       /* unsigned chars per block */
#define KS_DEPTH 48       /* key pairs */
/* Keyschedule */
DES3_KS Ks;

/* permuted choice table (key) */
static unsigned char pc1[] = {
   57, 49, 41, 33, 25, 17,  9,
    1, 58, 50, 42, 34, 26, 18,
   10,  2, 59, 51, 43, 35, 27,
   19, 11,  3, 60, 52, 44, 36,

   63, 55, 47, 39, 31, 23, 15,
    7, 62, 54, 46, 38, 30, 22,
   14,  6, 61, 53, 45, 37, 29,
   21, 13,  5, 28, 20, 12,  4
};

/* number left rotations of pc1 */
static unsigned char totrot[] = {
   1,2,4,6,8,10,12,14,15,17,19,21,23,25,27,28
};

/* permuted choice key (table) */
static unsigned char pc2[] = {
   14, 17, 11, 24,  1,  5,
    3, 28, 15,  6, 21, 10,
    23, 19, 12,  4, 26,  8,
    16,  7, 27, 20, 13,  2,
    41, 52, 31, 37, 47, 55,
    30, 40, 51, 45, 33, 48,
    44, 49, 39, 56, 34, 53,
    46, 42, 50, 36, 29, 32
};

/* End of DES-defined tables */


/* bit 0 is left-most in byte */
static int bytebit[] = {
    0200,0100,040,020,010,04,02,01
};
/// Generate key schedule for encryption or decryption
//  depending on the value of "decrypt"
//
void deskey(k,key,decrypt)
unsigned long k[16][2];        /* Key schedule array */
unsigned char *key;     /* 64 bits (will use only 56) */
int decrypt;            /* 0 = encrypt, 1 = decrypt */
{
    unsigned char pc1m[56];     /* place to modify pc1 into */
    unsigned char pcr[56];      /* place to rotate pc1 into */
    register int i,j,l;
    int m;
    unsigned char ks[8];

    for (j=0; j<56; j++) {      /* convert pc1 to bits of key */
        l=pc1[j]-1;     /* integer bit location  */
        m = l & 07;     /* find bit      */
        pc1m[j]=(key[l>>3] &    /* find which key byte l is in */
            bytebit[m]) /* and which bit of that byte */
            ? 1 : 0;    /* and store 1-bit result */
    }
    for (i=0; i<16; i++) {      /* key chunk for each iteration */
        memset(ks,0,sizeof(ks));    /* Clear key schedule */
        for (j=0; j<56; j++)    /* rotate pc1 the right amount */
            pcr[j] = pc1m[(l=j+totrot[decrypt? 15-i : i])<(j<28? 28 : 56) ? l: l-28];
            /* rotate left and right halves independently */
        for (j=0; j<48; j++){   /* select bits individually */
            /* check bit that goes to ks[j] */
            if (pcr[pc2[j]-1]){
                /* mask it in if it's there */
                l= j % 6;
                ks[j/6] |= bytebit[l] >> 2;
            }
        }
        /* Now convert to packed odd/even interleaved form */
        k[i][0] = ((long)ks[0] << 24)
         | ((long)ks[2] << 16)
         | ((long)ks[4] << 8)
         | ((long)ks[6]);
        k[i][1] = ((long)ks[1] << 24)
         | ((long)ks[3] << 16)
         | ((long)ks[5] << 8)
         | ((long)ks[7]);
//        if(Asmversion){
//            /* The assembler versions pre-shift each subkey 2 bits
//             * so the Spbox indexes are already computed
//             */
//            k[i][0] <<= 2;
//            k[i][1] <<= 2;
//        }
    }
}

//  Generate key schedule for triple DES in E-D-E (or D-E-D) mode.
//
//  The key argument is taken to be 24 bytes. The first 8 bytes are K1
//  for the first stage, the second 8 bytes are K2 for the middle stage
//  and the third 8 bytes are K3 for the last stage
//
void des3key(k,key,decrypt)
unsigned long k[48][2];
unsigned char *key;     /* 192 bits (will use only 168) */
int decrypt;            /* 0 = encrypt, 1 = decrypt */
{
    if(!decrypt){
        deskey(&k[0],&key[0],0);
        deskey(&k[16],&key[8],1);
        deskey(&k[32],&key[16],0);
    } else {
        deskey(&k[32],&key[0],1);
        deskey(&k[16],&key[8],0);
        deskey(&k[0],&key[16],1);
    }
}

#define F(l,r,key){\
    work = ((r >> 4) | (r << 28)) ^ key[0];\
    l ^= Spbox[6][work & 0x3f];\
    l ^= Spbox[4][(work >> 8) & 0x3f];\
    l ^= Spbox[2][(work >> 16) & 0x3f];\
    l ^= Spbox[0][(work >> 24) & 0x3f];\
    work = r ^ key[1];\
    l ^= Spbox[7][work & 0x3f];\
    l ^= Spbox[5][(work >> 8) & 0x3f];\
    l ^= Spbox[3][(work >> 16) & 0x3f];\
    l ^= Spbox[1][(work >> 24) & 0x3f];\
}
```

APPENDIX H

Compiling and Loading the Device Driver

Compiling the Device Driver:

To compile the device driver you need to create a makefile in the directory of the device driver code. The makefile below works for the device driver in Appendix G.

Makefile:

```
obj-m +=des.o

all:
	make -C ../linux-2.6-xlnx/ M=$(PWD) modules

clean:
	make -C ../linux-2.6-xlnx/ M=$(PWD) clean
```

From here you need to open a terminal and go to the directory of the device driver. Once there all you need to do is type: *ppcmake all*

This will generate a file named "des.ko", which is your device driver. You then need to move this to your target machine – the command scp works just fine for this.

Loading the Device Driver

Once the device driver is on the target machine, all you need to do is get on the machine and type:

```
Insmod des.ko
mknod /dev/des c 76 0
```

The two above commands insert the device driver into the system and notes that it is a character device driver that has I/O capability via /dev/des.

APPENDIX I

3DES Implementation That Uses the Hardware

This appendix contains the code that comprises our hardware implementation of 3DES.

The code is comprised of four files: mpi_hard.c, des2.h, sp.c, and des_def.h.

mpi_hard.c

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <errno.h>
#include <fcntl.h> /* open */
#include <unistd.h> /* exit */
#include <sys/ioctl.h> /* ioctl */
#include <vector>
#include <fstream>
#include <string>
#include <iostream>
#include <mpi.h>
int file_desc;
#include "des2.h"
#include "des_def.h"
#include "sp.c"

using namespace std;
int NumBlocks;
vector <unsigned char> Blocks;


/*
 * Functions for the ioctl calls
 */
void ioctl_write_config(int file_desc, uint32_t *param)
{
  int ret_val = ioctl(file_desc, IOCTL_WRITE_CONFIG, param);
  if (ret_val < 0) {
    printf("ioctl_write_config failed:%d\n", ret_val);
    exit(EXIT_FAILURE);
  }
}
void ioctl_close_config(int file_desc, uint32_t *param)
{
     ioctl(file_desc, IOCTL_CLOSE_CONFIG, param);
}

void ioctl_write_block(int file_desc, uint8_t *param)
{
  int ret_val = ioctl(file_desc, IOCTL_WRITE_BLOCK, param);
  if (ret_val < 0) {
    printf("ioctl_write_config failed:%d\n", ret_val);
    exit(EXIT_FAILURE);
  }
}

void ioctl_get_input(int file_desc, uint8_t *result)
{

  int ret_val = ioctl(file_desc, IOCTL_GET_INPUT, result);
  if (ret_val < 0) {
    printf("ioctl_get_values failed:%d\n", ret_val);
    exit(EXIT_FAILURE);
  }
}

void get_data(int rank, int size, string filename)
{
  vector <int> temp;
  int temp_size=0;
  MPI_Status status;
  /* have p0 get data at end to pick up extras */
  if(rank==0)
  {

    int start, end, total;
    int tBlocks;
    int current_rank=1;
    int p0_size=0;
    int c_size=0;
    ifstream infile;
    infile.open(filename.c_str());
    start=infile.tellg();
    infile.seekg(0, ios::end);
    end=infile.tellg();
    infile.seekg(0, ios::beg);
    total=end-start;
    /* take total and right shift by 3, eg divide by 8 */
    tBlocks=total/BLOCKSIZE;
    if(total % BLOCKSIZE)
      tBlocks++; /* tells total number of blocks */
    NumBlocks=tBlocks/size;
    MPI_Bcast(&NumBlocks, 1, MPI_INT,0, MPI_COMM_WORLD);

    temp_size=NumBlocks*BLOCKSIZE;
    NumBlocks=tBlocks-(size-1)*NumBlocks;
    /* Data on p0 will be zero padded to ensure that it
             contains a number of elements divisiable by 8 */
    p0_size=total-temp_size*(size-1);
    Blocks.resize(NumBlocks*BLOCKSIZE,0);
    cout<<"size is "<<NumBlocks*BLOCKSIZE<<endl;
    for(int i=1; i<=size; i++)
    {
      current_rank=i%size; /* gives correct rank to receive from */
      if(current_rank>0)
        c_size=temp_size;
      else
      {
        c_size=p0_size;
      }
      for(int j=0; j<c_size; j++)
      {
        Blocks[j]=infile.get();
      }
      if(current_rank!=0)
        MPI_Send(&Blocks.front(), temp_size, MPI_CHAR, current_rank, 0,
MPI_COMM_WORLD);
```

```
    }
    infile.close();
  }
  else
  {
    //cout<<rank<<" is receieving data "<<endl;
    MPI_Bcast(&NumBlocks, 1, MPI_INT, 0, MPI_COMM_WORLD);
    temp_size=NumBlocks*BLOCKSIZE; /* NumBlocks * 8, tells number of elements */
    Blocks.resize(temp_size);
    MPI_Recv(&Blocks.front(), temp_size,
MPI_CHAR,0,0,MPI_COMM_WORLD,&status);

  }
  //cout<<rank<<" is done "<<endl;
}
void prep_driver()
{
  int i, k;
  uint32_t zero=0;
  uint32_t data;
  for ( k = 0; k < 2; k++ ) {
    for ( i = 0; i < KS_DEPTH; i++ ) {
      data=Ks[i][k];
      ioctl_write_config(file_desc, &data);

    }
  }

  for ( i = 0; i < SPBOX_X; i++ ) {
    for ( k = 0; k < SPBOX_Y; k++ ) {
      data=Spbox[i][k];
      ioctl_write_config(file_desc, &data);

    }
  }
  ioctl_close_config(file_desc, &zero);
}
void des_test(int rank,int size, string filename2)
{
  int i, k;
```

```
unsigned int blockCount = 0;
unsigned char block[8];
unsigned char blockElement;
vector <unsigned char> solution;
vector <unsigned char> temp;
int current_rank;
int write_id;
MPI_Status status;

while( blockCount<NumBlocks ) {
  for ( k = 0; k < BLOCKSIZE; k++ ) {
    blockElement = Blocks[blockCount * BLOCKSIZE + k];
    ioctl_write_block(file_desc,&blockElement);

  }

  for ( k = 0; k < BLOCKSIZE; k++ ) {
    ioctl_get_input(file_desc, &blockElement);
    block[k]=blockElement;
  }

  for (i=0; i<BLOCKSIZE; i++) {
    solution.push_back(block[i]);
  }
  ++blockCount;
}
if(size>1)
  write_id=1;
else
  write_id=0;

if(rank==write_id)
{
  int size_temp=0;
  ofstream outfile;
  outfile.open(filename2.c_str());

  for(int i=1; i<=size; i++)
  {
    current_rank=i%size; /* gives correct rank to receive from */
```

```
      if(current_rank==rank)
      {
        for(int j=0;j<solution.size();j++)
        {
          outfile.setf(ios::hex,ios::basefield);
          outfile<<solution[j];
        }
      }
      else
      {
        MPI_Recv(&size_temp,1, MPI_INT, current_rank, 0,
MPI_COMM_WORLD,&status);
        temp.resize(size_temp);
        MPI_Recv(&temp.front(),size_temp, MPI_CHAR, current_rank, 0,
MPI_COMM_WORLD, &status);
        for(int j=0; j<size_temp;j++)
        {
          outfile.setf(ios::hex,ios::basefield);
          outfile<<temp[j];
        }
      }
    }

    outfile.close();
  }
  else
  {
    /* send my solution to process 0 */
    int tSize=solution.size();
    MPI_Send(&tSize, 1, MPI_INT, write_id, 0, MPI_COMM_WORLD);
    MPI_Send(&solution.front(),tSize , MPI_CHAR, write_id, 0,
MPI_COMM_WORLD);
  }


}

int main(int argc, char *argv[])
{
  int npes, myrank;
  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &npes);
  MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
  string filename, filename2;
  filename=argv[1];
  filename2=argv[2];
  file_desc=open("/dev/des",O_RDWR);
  unsigned char * key = (unsigned char *) "Gflk jqo40978J0dmm$%@878"; /* 24 bytes
*/
  des3key(Ks, key, 0);  /* Create a keyschedule for encryption */
  /* Grab data and send out*/
  prep_driver();  // Send encryption information to device
  get_data(myrank, npes, filename);
  des_test(myrank, npes, filename2);
  close(file_desc);
  MPI_Finalize();
  exit(EXIT_SUCCESS);
}
```

des2.h

```
#ifndef des_H
#define des_H

#include <linux/ioctl.h>

#define MAJOR_NUM 76

/*
 * Set the values of parameters
 */
#define IOCTL_WRITE_CONFIG    _IOW(MAJOR_NUM, 0, uint32_t *)
/*
 * Get the result value
 */
#define IOCTL_CLOSE_CONFIG    _IOR(MAJOR_NUM, 1, uint32_t *)
#define IOCTL_WRITE_BLOCK     _IOR(MAJOR_NUM, 2, uint8_t *)
#define IOCTL_GET_INPUT       _IOR(MAJOR_NUM, 3, uint8_t *)


#endif
```

des_def.h

```
typedef unsigned long DES_KS[16][2];        /* Single-key DES key schedule */
typedef unsigned long DES3_KS[48][2];      /* Triple-DES key schedule */

/* In deskey.c: */
void deskey(DES_KS,unsigned char *,int);
void des3key(DES3_KS,unsigned char *,int);

/* In desport.c, desborl.cas or desgnu.s: */
void des(DES_KS,unsigned char *);
/* In des3port.c, des3borl.cas or des3gnu.s: */
void des3(DES3_KS,unsigned char *);

#define BLOCKSIZE 8       /* unsigned chars per block */
#define KS_DEPTH 48       /* key pairs */
/* Keyschedule */
DES3_KS Ks;

/* permuted choice table (key) */
static unsigned char pc1[] = {
   57, 49, 41, 33, 25, 17,  9,
    1, 58, 50, 42, 34, 26, 18,
   10,  2, 59, 51, 43, 35, 27,
   19, 11,  3, 60, 52, 44, 36,

   63, 55, 47, 39, 31, 23, 15,
    7, 62, 54, 46, 38, 30, 22,
   14,  6, 61, 53, 45, 37, 29,
   21, 13,  5, 28, 20, 12,  4
};

/* number left rotations of pc1 */
static unsigned char totrot[] = {
   1,2,4,6,8,10,12,14,15,17,19,21,23,25,27,28
};

/* permuted choice key (table) */
static unsigned char pc2[] = {
   14, 17, 11, 24,  1,  5,
     3, 28, 15,  6, 21, 10,
    23, 19, 12,  4, 26,  8,
    16,  7, 27, 20, 13,  2,
    41, 52, 31, 37, 47, 55,
    30, 40, 51, 45, 33, 48,
    44, 49, 39, 56, 34, 53,
    46, 42, 50, 36, 29, 32
};

/* End of DES-defined tables */


/* bit 0 is left-most in byte */
static int bytebit[] = {
    0200,0100,040,020,010,04,02,01
};
/// Generate key schedule for encryption or decryption
//  depending on the value of "decrypt"
//
void deskey(DES_KS k,unsigned char *key,int decrypt)
//unsigned long k[16][2];         /* Key schedule array */
//unsigned char *key;     /* 64 bits (will use only 56) */
//int decrypt;            /* 0 = encrypt, 1 = decrypt */
{
    unsigned char pc1m[56];     /* place to modify pc1 into */
    unsigned char pcr[56];      /* place to rotate pc1 into */
    register int i,j,l;
    int m;
    unsigned char ks[8];

    for (j=0; j<56; j++) {      /* convert pc1 to bits of key */
        l=pc1[j]-1;     /* integer bit location  */
        m = l & 07;     /* find bit      */
        pc1m[j]=(key[l>>3] &    /* find which key byte l is in */
            bytebit[m]) /* and which bit of that byte */
            ? 1 : 0;    /* and store 1-bit result */
    }
    for (i=0; i<16; i++) {      /* key chunk for each iteration */
        memset(ks,0,sizeof(ks));    /* Clear key schedule */
        for (j=0; j<56; j++)    /* rotate pc1 the right amount */
```

```
            pcr[j] = pc1m[(l=j+totrot[decrypt? 15-i : i])<(j<28? 28 : 56) ? l: l-28];
            /* rotate left and right halves independently */
        for (j=0; j<48; j++){   /* select bits individually */
            /* check bit that goes to ks[j] */
            if (pcr[pc2[j]-1]){
                /* mask it in if it's there */
                l= j % 6;
                ks[j/6] |= bytebit[l] >> 2;
            }
        }
        /* Now convert to packed odd/even interleaved form */
        k[i][0] = ((long)ks[0] << 24)
         | ((long)ks[2] << 16)
         | ((long)ks[4] << 8)
         | ((long)ks[6]);
        k[i][1] = ((long)ks[1] << 24)
         | ((long)ks[3] << 16)
         | ((long)ks[5] << 8)
         | ((long)ks[7]);
//        if(Asmversion){
//            /* The assembler versions pre-shift each subkey 2 bits
//             * so the Spbox indexes are already computed
//             */
//            k[i][0] <<= 2;
//            k[i][1] <<= 2;
//        }
    }
}

//  Generate key schedule for triple DES in E-D-E (or D-E-D) mode.
//
//  The key argument is taken to be 24 bytes. The first 8 bytes are K1
//  for the first stage, the second 8 bytes are K2 for the middle stage
//  and the third 8 bytes are K3 for the last stage
//
void des3key(DES3_KS k,unsigned char *key, int decrypt)
//unsigned long k[48][2];
//unsigned char *key;     /* 192 bits (will use only 168) */
//int decrypt;            /* 0 = encrypt, 1 = decrypt */
{
    if(!decrypt){
        deskey(&k[0],&key[0],0);
        deskey(&k[16],&key[8],1);
        deskey(&k[32],&key[16],0);
    } else {
        deskey(&k[32],&key[0],1);
        deskey(&k[16],&key[8],0);
        deskey(&k[0],&key[16],1);
    }
}

#define F(l,r,key){\
    work = ((r >> 4) | (r << 28)) ^ key[0];\
    l ^= Spbox[6][work & 0x3f];\
    l ^= Spbox[4][(work >> 8) & 0x3f];\
    l ^= Spbox[2][(work >> 16) & 0x3f];\
    l ^= Spbox[0][(work >> 24) & 0x3f];\
    work = r ^ key[1];\
    l ^= Spbox[7][work & 0x3f];\
    l ^= Spbox[5][(work >> 8) & 0x3f];\
    l ^= Spbox[3][(work >> 16) & 0x3f];\
    l ^= Spbox[1][(work >> 24) & 0x3f];\
}
```

sp.c

```
#define SPBOX_X 8
#define SPBOX_Y 64
unsigned long Spbox[8][64] = {
0x01010400,0x00000000,0x00010000,0x01010404,
0x01010004,0x00010404,0x00000004,0x00010000,
0x00000400,0x01010400,0x01010404,0x00000400,
0x01000404,0x01010004,0x01000000,0x00000004,
0x00000404,0x01000400,0x01000400,0x00010400,
0x00010400,0x01010000,0x01010000,0x01000404,
0x00010004,0x01000004,0x01000004,0x00010004,
0x00000000,0x00000404,0x00010404,0x01000000,
0x00010000,0x01010404,0x00000004,0x01010000,
0x01010400,0x01000000,0x01000000,0x00000400,
0x01010004,0x00010000,0x00010400,0x01000004,
0x00000400,0x00000004,0x01000404,0x00010404,
0x01010404,0x00010004,0x01010000,0x01000404,
0x01000004,0x00000404,0x00010404,0x01010400,
0x00000404,0x01000400,0x01000400,0x00000000,
0x00010004,0x00010400,0x00000000,0x01010004,
0x80108020,0x80008000,0x00008000,0x00108020,
0x00100000,0x00000020,0x80100020,0x80008020,
0x80000020,0x80108020,0x80108000,0x80000000,
0x80008000,0x00100000,0x00000020,0x80100020,
0x00108000,0x00100020,0x80008020,0x00000000,
0x80000000,0x00008000,0x00108020,0x80100000,
0x00100020,0x80000020,0x00000000,0x00108000,
0x00008020,0x80108000,0x80100000,0x00008020,
0x00000000,0x00108020,0x80100020,0x00100000,
0x80008020,0x80100000,0x80108000,0x00008000,
0x80100000,0x80008000,0x00000020,0x80108020,
0x00108020,0x00000020,0x00008000,0x80000000,
0x00008020,0x80108000,0x00100000,0x80000020,
0x00100020,0x80008020,0x80000020,0x00100020,
0x00108000,0x00000000,0x80008000,0x00008020,
0x80000000,0x80100020,0x80108020,0x00108000,
0x00000208,0x08020200,0x00000000,0x08020008,
0x08000200,0x00000000,0x00020208,0x08000200,
0x00020008,0x08000008,0x08000008,0x00020000,
0x08020208,0x00020008,0x08020000,0x00000208,
0x08000000,0x00000008,0x08020200,0x00000200,
0x00020200,0x08020000,0x08020008,0x00020208,
0x08000208,0x00020200,0x00020000,0x08000208,
0x00000008,0x08020208,0x00000200,0x08000000,
0x08020200,0x08000000,0x00020008,0x00000208,
0x00020000,0x08020200,0x08000200,0x00000000,
0x00000200,0x00020008,0x08020208,0x08000200,
0x08000008,0x00000200,0x00000000,0x08020008,
0x08000208,0x00020000,0x08000000,0x08020208,
0x00000008,0x00020208,0x00020200,0x08000008,
0x08020000,0x08000208,0x00000208,0x08020000,
0x00020208,0x00000008,0x08020008,0x00020200,
0x00802001,0x00002081,0x00002081,0x00000080,
0x00802080,0x00800081,0x00800001,0x00002001,
0x00000000,0x00802000,0x00802000,0x00802081,
0x00000081,0x00000000,0x00800080,0x00800001,
0x00000001,0x00002000,0x00800000,0x00802001,
0x00000080,0x00800000,0x00002001,0x00002080,
0x00800081,0x00000001,0x00002080,0x00800080,
0x00002000,0x00802080,0x00802081,0x00000081,
0x00800080,0x00800001,0x00802000,0x00802081,
0x00000081,0x00000000,0x00000000,0x00802000,
0x00002080,0x00800080,0x00800081,0x00000001,
0x00802001,0x00002081,0x00002081,0x00000080,
0x00802081,0x00000081,0x00000001,0x00002000,
0x00800001,0x00002001,0x00802080,0x00800081,
0x00002001,0x00002080,0x00800000,0x00802001,
0x00000080,0x00800000,0x00002000,0x00802080,
0x00000100,0x02080100,0x02080000,0x42000100,
0x00080000,0x00000100,0x40000000,0x02080000,
0x40080100,0x00080000,0x02000100,0x40080100,
0x42000100,0x42080000,0x00080100,0x40000000,
0x02000000,0x40080000,0x40080000,0x00000000,
0x40000100,0x42080100,0x42080100,0x02000100,
0x42080000,0x40000100,0x00000000,0x42000000,
0x02080100,0x02000000,0x42000000,0x00080100,
0x00080000,0x42000100,0x00000100,0x02000000,
0x40000000,0x02080000,0x42000100,0x40080100,
0x02000100,0x40000000,0x42080000,0x02080100,
0x40080100,0x00000100,0x02000000,0x42080000,
0x42080100,0x00080100,0x42000000,0x42080100,
0x02080000,0x00000000,0x40080000,0x42000000,
0x00080100,0x02000100,0x40000100,0x00080000,
0x00000000,0x40080000,0x02080100,0x40000100,
0x20000010,0x20400000,0x00004000,0x20404010,
0x20400000,0x00000010,0x20404010,0x00400000,
0x20004000,0x00404010,0x00400000,0x20000010,
0x00400010,0x20004000,0x20000000,0x00004010,
0x00000000,0x00400010,0x20004010,0x00004000,
0x00404000,0x20004010,0x00000010,0x20400010,
0x20400010,0x00000000,0x00404010,0x20404000,
0x00004010,0x00404000,0x20404000,0x20000000,
0x20004000,0x00000010,0x20400010,0x00404000,
0x20404010,0x00400000,0x00004010,0x20000010,
0x00400000,0x20004000,0x20000000,0x00004010,
0x20000010,0x20404010,0x00404000,0x20400000,
0x00404010,0x20404000,0x00000000,0x20400010,
0x00000010,0x00004000,0x20400000,0x00404010,
0x00004000,0x00400010,0x20004010,0x00000000,
0x20404000,0x20000000,0x00400010,0x20004010,
0x00200000,0x04200002,0x04000802,0x00000000,
0x00000800,0x04000802,0x00200802,0x04200800,
0x04200802,0x00200000,0x00000000,0x04000002,
0x00000002,0x04000000,0x04200002,0x00000802,
0x04000800,0x00200802,0x00200002,0x04000800,
0x04000002,0x04200000,0x04200800,0x00200002,
0x04200000,0x00000800,0x00000802,0x04200802,
0x00200800,0x00000002,0x04000000,0x00200800,
0x04000000,0x00200800,0x00200000,0x04000802,
0x04000802,0x04200002,0x04200002,0x00000002,
0x00200002,0x04000000,0x04000800,0x00200000,
0x04200800,0x00000802,0x00200802,0x04200800,
0x00000802,0x04000002,0x04200802,0x04200000,
0x00200800,0x00000000,0x00000002,0x04200802,
0x00000000,0x00200802,0x04200000,0x00000800,
0x04000002,0x04000800,0x00000800,0x00200002,
0x10001040,0x00001000,0x00040000,0x10041040,
0x10000000,0x10001040,0x00000040,0x10000000,
0x00040040,0x10040000,0x10041040,0x00041000,
0x10041000,0x00041040,0x00001000,0x00000040,
0x10040000,0x10000040,0x10001000,0x00001040,
0x00041000,0x00040040,0x10040040,0x10041000,
0x00001040,0x00000000,0x00000000,0x10040040,
0x10000040,0x10001000,0x00041040,0x00040000,
0x00041040,0x00040000,0x10041000,0x00001000,
0x00000040,0x10040040,0x00001000,0x00041040,
0x10001000,0x00000040,0x10000040,0x10040000,
0x10040040,0x10000000,0x00040000,0x10001040,
0x00000000,0x10041040,0x00040040,0x10000040,
0x10040000,0x10001000,0x10001040,0x00000000,
0x10041040,0x00041000,0x00041000,0x00001040,
0x00001040,0x00040040,0x10000000,0x10041000,
};
```

APPENDIX J

Communication Overhead Code

This appendix contains the code I use to give an estimate of the communication overhead. This appendix only contains 1 file: NOP.c

NOP.c

```
/*
  This program is used to estimate the overhead from  communication alone.
  All computation for 3DES has been removed,  so that a file is read in and
  split to multiple targets.  The targets send the data back after receiving
  the data, and the data is then outputed- so the file is not altered.
*/

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <errno.h>
#include <fcntl.h> /* open */
#include <unistd.h> /* exit */
#include <sys/ioctl.h> /* ioctl */
#include <vector>
#include <fstream>
#include <string>
#include <iostream>
#include <mpi.h>

#define BLOCKSIZE 8       /* unsigned chars per block */
using namespace std;
int NumBlocks;
vector <unsigned char> Blocks;


void get_data(int rank, int size, string filename)
{
  vector <int> temp;
```

```
int temp_size=0;
MPI_Status status;
/* have p0 get data at end to pick up extras */
if(rank==0)
{
  int start, end, total;
  int tBlocks;
  int current_rank=1;
  int p0_size=0;
  int c_size=0;
  ifstream infile;
  infile.open(filename.c_str());
  start=infile.tellg();
  infile.seekg(0, ios::end);
  end=infile.tellg();
  infile.seekg(0, ios::beg);
  total=end-start;
  /* take total and right shift by 3, eg divide by 8 */
  tBlocks=total/BLOCKSIZE;
  if(total % BLOCKSIZE)
    tBlocks++; /* tells total number of blocks */
  NumBlocks=tBlocks/size;
  MPI_Bcast(&NumBlocks, 1, MPI_INT,0, MPI_COMM_WORLD);

  temp_size=NumBlocks*BLOCKSIZE;
  NumBlocks=tBlocks-(size-1)*NumBlocks;

  p0_size=total-temp_size*(size-1);
  Blocks.resize(NumBlocks*BLOCKSIZE,0);
  for(int i=1; i<=size; i++)
  {
    current_rank=i%size; /* gives correct rank to send to */
    if(current_rank>0)
      c_size=temp_size;
    else
    {
      c_size=p0_size;
    }
    for(int j=0; j<c_size; j++)
    {
        Blocks[j]=infile.get();
      }
      if(current_rank!=0)
        MPI_Send(&Blocks.front(), temp_size, MPI_CHAR, current_rank, 0,
MPI_COMM_WORLD);
    }
    infile.close();
  }
  else
  {
    MPI_Bcast(&NumBlocks, 1, MPI_INT, 0, MPI_COMM_WORLD);
    temp_size=NumBlocks*BLOCKSIZE; /* NumBlocks * 8, tells number of elements */
    Blocks.resize(temp_size);
    MPI_Recv(&Blocks.front(), temp_size,
MPI_CHAR,0,0,MPI_COMM_WORLD,&status);
  }
}

void des_test(int rank,int size, string filename2)
{
  int i, k;
  unsigned int blockCount = 0;
  unsigned char block[8];
  unsigned char blockElement;
  vector <unsigned char> temp;
  int current_rank;
  int write_id;
  MPI_Status status;

  /* Calculations would normally go here */
  if(size>1)
    write_id=1;
  else
    write_id=0;

  if(rank==write_id)
  {
    int size_temp=0;
    ofstream outfile;
    outfile.open(filename2.c_str());
    for(int i=1; i<=size; i++)
    {
      current_rank=i%size; /* gives correct rank to receive from */
      if(current_rank==rank)
      {
        for(int j=0;j<Blocks.size();j++)
        {
          outfile<<Blocks[j];
        }
      }
      else
      {
        MPI_Recv(&size_temp,1, MPI_INT, current_rank, 0,
MPI_COMM_WORLD,&status);
        temp.resize(size_temp);
        MPI_Recv(&temp.front(),size_temp, MPI_CHAR, current_rank, 0,
MPI_COMM_WORLD, &status);
        for(int j=0; j<size_temp;j++)
        {
          outfile<<temp[j];
        }
      }
    }
    outfile.close();
  }
  else
  {
    /* send data to process write_id */
    int tSize=Blocks.size();
    MPI_Send(&tSize, 1, MPI_INT, write_id, 0, MPI_COMM_WORLD);
    MPI_Send(&Blocks.front(),tSize , MPI_CHAR, write_id, 0, MPI_COMM_WORLD);
  }
}

int main(int argc, char *argv[])
{
  int npes, myrank;
  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &npes);
```

```
 MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
 string filename, filename2;
 filename=argv[1];
 filename2=argv[2];
 /* Grab data and send out*/
 get_data(myrank, npes, filename);
 des_test(myrank, npes, filename2);
 MPI_Finalize();
 exit(EXIT_SUCCESS);
}
```

APPENDIX K

Data

Test One:
Runtimes for 3DES implementation with Hardware

File Size	14 KB	27 KB	54 KB	108 KB	216 KB	433 KB	866 KB
Test 1	5.629	3.888	3.953	4.153	4.481	5.135	6.897
Test 2	4.211	3.868	4.033	4.122	4.444	5.318	6.749
Test 3	4.711	3.875	4.33	4.835	5.787	5.143	7.246
Test 4	4.202	5.014	3.966	4.122	5.974	10.331	7.379
Test 5	6.805	6.641	7.257	7.238	5.004	6.459	7.599
Test 6	4.376	3.999	4.203	4.644	4.66	5.202	9.317
Test 7	5.459	3.869	3.959	5.153	5.921	5.902	6.812
Test 8	4.886	3.904	4.347	4.805	7.673	6.409	8.038
Test 9	6.785	5.59	6.636	5.487	5.605	5.6	6.855
Test 10	3.888	3.892	3.947	5.8	7.144	6.586	7.849
Test 11	4.898	4.125	3.998	4.158	8.005	6.522	9.19
Test 12	3.821	4.853	5.773	4.714	4.74	5.296	6.83
Test 13	6.817	4.47	6.998	4.554	5.169	6.752	10.68
Test 14	4.521	5.713	4.125	4.883	4.697	7.29	7.117
Test 15	5.273	4.479	4.016	4.125	6.234	5.621	7.712
Test 16	3.824	4.36	3.945	5.23	5.027	6.778	6.7
Test 17	6.975	3.911	7.121	5.899	4.456	5.118	6.593
Test 18	3.815	6.437	3.958	4.121	7.373	5.134	9.585
Test 19	4.534	4.9711	4.501	4.117	6.518	6.61	6.907
Test 20	4.077	4.738	4.183	7.333	4.848	5.575	8.05

File Size	10 MB	20 MB	40 MB	80 MB	160 MB
Test 1	39.62	79.649	163.98	319.639	630.772
Test 2	43.809	81.612	158.467	316.463	640.241
Test 3	43.485	88.702	164.686	322.05	651.952
Test 4	43.816	83.655	163.153	321.159	630.918
Test 5	42.829	87.385	161.408	323.093	628.425
Test 6	45.562	83.025	162.032	318.015	632.511
Test 7	48.454	86.706	159.706	321.369	634.023
Test 8	41.935	79.43	164.831	320.736	630.623
Test 9	45.782	83.955	161.883	317.475	635.869
Test 10	46.887	79.244	157.831	320.081	623.05

Test 11	41.707	81.756	159.769	321.215	627.537
Test 12	47.15	79.677	167.118	321.791	628.31
Test 13	42.914	88.061	160.562	321.837	626.089
Test 14	50.583	89.54	157.834	325.82	633.869
Test 15	47.773	79.846	163.43	319.51	632.804
Test 16	42.981	83.107	164.693	318.896	650.389
Test 17	45.68	83.432	164.338	319.574	627.708
Test 18	47.849	83.874	162.66	319.682	623.1
Test 19	48.75	81.004	163.197	327.449	611.449
Test 20	46.983	82.817	160.434	318.722	618.281

Runtimes for 3DES implementation without Hardware

File Size	14 KB	27 KB	54 KB	108 KB	216 KB	433 KB	866 KB
Test 1	4.096	3.883	4.293	4.191	4.61	6.925	8.74
Test 2	6.043	3.885	4.385	5.516	4.595	5.42	7.062
Test 3	3.83	4.491	3.99	4.184	6.267	6.255	10.763
Test 4	5.345	3.883	4.948	7.3565	8.954	6.711	8.069
Test 5	4.112	5.959	4.88	4.715	5.062	5.425	11.341
Test 6	5.018	3.991	5.452	4.97	5.051	5.897	7.287
Test 7	4.034	4.091	4.25	4.762	6.774	9.813	7.366
Test 8	5.943	5.919	7.303	7.458	4.948	5.614	10.632
Test 9	3.841	7.368	4.966	4.284	6.389	6.377	8.95
Test 10	6.222	4.168	4.301	5.222	6.471	9.196	10.573
Test 11	4.157	3.898	4.241	4.327	6.147	6.289	8.342
Test 12	6.031	4.963	4.189	7.983	5.788	6.013	7.156
Test 13	3.828	7.429	6.456	4.582	4.621	9.484	10.618
Test 14	5.387	3.888	4.509	5.821	8.184	5.953	7.281
Test 15	3.821	5.667	6.23	4.201	7.319	5.746	11.975
Test 16	5.146	3.902	4.882	8.123	5.253	8.844	7.48
Test 17	3.834	5.928	5.218	6.704	6.196	5.915	11.563
Test 18	3.872	4.278	7.427	4.337	8.183	6.229	7.137
Test 19	4.124	4.809	4.55	5.202	4.821	10.068	11.736
Test 20	3.949	3.879	6.253	6.248	5.58	6.228	7.32

File Size	10 MB	20 MB	40 MB	80 MB	160 MB
Test 1	48.376	102.691	192.602	379.271	739.498
Test 2	52.33	105.101	192.815	373.728	741.361
Test 3	53.049	92.723	185.628	373.772	730.137
Test 4	51.421	98.673	193.034	372.583	750.383
Test 5	53.546	94.715	186.954	367.971	749.089
Test 6	51.761	95.59	194.985	379.8	743.541
Test 7	51.374	104.901	188.764	372.461	746.934
Test 8	51.788	99.727	188.951	373.366	741.073

Test 9	52.756	103.824	199.484	375.556	745.648
Test 10	51.9	98.451	193.603	374.733	747.481
Test 11	51.84	101.935	194.489	370.786	746.376
Test 12	57.247	96.22	185.606	376.436	746.647
Test 13	50.338	96.674	190.602	366.609	749.416
Test 14	50.128	99.036	195.148	376.976	745.484
Test 15	47.591	95.949	186.614	373.578	747.677
Test 16	59.282	105.88	192.761	373.659	745.642
Test 17	55.833	93.383	191.816	375.665	747.13
Test 18	47.354	100.247	186.541	369.351	754.232
Test 19	50.741	97.778	191.566	372.643	746.371
Test 20	49.673	95.922	192.023	373.2	747.157

Timing of communication overhead

File Size	14 KB	27 KB	54 KB	108 KB	216 KB	433 KB	866 KB
Test 1	3.819	6.302	5.061	4.053	5.531	5.21	13.767
Test 2	3.83	6.347	6.367	4.374	7.757	7.925	7.472
Test 3	4.295	5.192	3.91	4.007	4.448	5.707	5.527
Test 4	7.45	4.558	4.653	4.247	5.398	7.226	5.729
Test 5	3.888	3.848	5.914	4.234	6.529	6.809	6.657
Test 6	4.372	4.092	6.019	4.609	4.483	5.816	8.39
Test 7	3.929	3.827	4.557	4.571	4.581	7.582	10.369
Test 8	3.808	5.531	4.098	4.726	7.51	5.804	5.582
Test 9	4.453	6.729	6.654	6.774	7.577	5.059	5.684
Test 10	4.346	5.784	4.301	5.492	4.201	7.678	9.077
Test 11	5.121	3.837	3.883	3.992	6.61	5.18	6.067
Test 12	6.981	5.935	5.682	3.992	4.862	4.988	9.334
Test 13	3.807	3.833	7.228	5.249	4.234	7.113	5.843
Test 14	3.797	4.122	6.024	3.993	7.38	4.644	6.407
Test 15	4.068	3.828	5.434	5.701	6.604	5.569	7.865
Test 16	4.211	6.223	6.042	5.518	6.676	7.382	6.271
Test 17	3.801	6.987	4.136	7.031	6.148	6.767	8.242
Test 18	5.543	3.822	4.259	4.872	4.787	4.899	8.686
Test 19	7.909	3.847	3.869	4.248	4.625	6.078	7.301
Test 20	4.045	4.508	5.308	4.035	7.108	4.649	6.226

File Size	10 MB	20 MB	40 MB	80 MB	160 MB
Test 1	25.094	61.056	122.976	238.138	435.551
Test 2	30.096	52.025	110.363	219.721	434.942
Test 3	25.994	63.473	118.307	212.075	436.436
Test 4	26.424	63.372	114.524	217.808	429.201
Test 5	27.309	56.461	114.108	217.582	425.441
Test 6	30.865	58.054	113.832	217.496	425.005

Test 7	27.252	58.693	114.781	218.609	427.25
Test 8	28.601	57.888	114.386	221.067	426.819
Test 9	26.966	57.15	116.679	215.628	426.399
Test 10	28.272	58.066	115.123	216.648	431.126
Test 11	27.326	56.731	115.113	217.652	425.783
Test 12	29.258	60.325	113.582	217.193	435.754
Test 13	26.303	59.741	115.845	219.421	420.957
Test 14	27.425	59.401	116.552	220.254	430.444
Test 15	27.142	55.667	114.438	211.17	439.453
Test 16	28.975	58.552	109.755	218.995	424.632
Test 17	30.189	66.658	115.404	218.026	429.472
Test 18	26.854	59.623	113.114	219.615	434.517
Test 19	28.232	56.283	117.841	218.243	433.172
Test 20	28.113	58.284	112.336	215.765	423.779

Test Two:
Runtimes for 3DES implementation with Hardware

Boards Used	1	2	3	4	5	6	7	8
Test 1	436.764	237.669	213.722	214.028	200.193	189.639	186.394	176.775
Test 2	473.187	241.603	217.276	222.081	202.191	192.839	182.27	179.534
Test 3	430.188	238.589	218.544	204.91	204.685	192.292	188.232	179.415
Test 4	420.006	241.756	216.791	209.807	202.897	194.685	185.421	179.258
Test 5	420.138	244.061	214.881	202.628	203.543	190.156	188.253	181.089
Test 6	417.963	240.652	210.34	203.066	199.976	194.387	182.642	179.414
Test 7	420.623	241.793	217.155	210.962	204.077	193.9	185.404	177.973
Test 8	422.743	241.949	217.269	204.939	203.689	193.542	182.356	180.626
Test 9	417.958	238.075	218.07	205.598	203.328	189.961	181.091	179.55
Test 10	424.008	241.3	216.06	204.968	203.263	188.015	185.133	176.883
Test 11	421.946	237.953	218.665	209.149	195.037	193.97	189.084	184.138
Test 12	424.778	240.532	210.286	204.567	199.699	187.37	187.556	178.007
Test 13	424.884	241.084	214.706	206.598	202.458	194.394	182.445	177.467
Test 14	422.514	240.539	216.409	208.762	204.309	196.542	183.103	183.16
Test 15	420.118	239.938	216.8	204.566	199.76	187.983	181.135	173.84
Test 16	423.164	241.932	218.243	206.407	205.167	193.893	184.305	175.97
Test 17	423.898	241.397	218.364	202.978	201.626	195.34	184.33	181.852
Test 18	414.629	240.382	210.398	208.995	204.133	193.951	187.698	173.791
Test 19	427.383	240.033	216.257	202.884	202.623	189.51	183.052	182.029
Test 20	424.753	236.246	216.333	204.305	204.316	196.814	184.183	181.265

Boards Used	9	10	11	12	13	14	15
Test 1	176.515	169.683	174.073	164.475	164.363	164.055	163.98
Test 2	174.164	173.707	171.617	167.079	166.23	164.634	158.467
Test 3	176.348	167.766	169.028	164.386	167.78	161.363	164.686
Test 4	176.8	176.632	169.466	165.787	161.893	164.573	163.153

Test 5	170.188	167.598	169.465	168.183	163.979	161.023	161.408
Test 6	180.597	173.788	168.57	162.819	164.503	164.981	162.032
Test 7	179.281	171.788	167.656	168.685	161.673	161.826	159.706
Test 8	178.295	169.265	167.668	165.468	165.186	163.645	164.831
Test 9	172.818	174.127	169.395	165.533	159.396	164.952	161.883
Test 10	177.576	167.74	168.612	167.082	167.37	161.539	157.831
Test 11	173.749	177.777	171.678	169.305	164.916	161.744	159.769
Test 12	173.982	174.332	164.78	164.248	170.322	163.897	167.118
Test 13	181.25	169.765	169.453	167.549	164.341	160.982	160.562
Test 14	179.42	175.689	174.938	166.862	162.58	163.082	157.834
Test 15	174.997	174.161	167.231	163.773	163.299	160.654	163.43
Test 16	176.2	171.074	169.456	166.608	161.39	166.676	164.693
Test 17	175.172	174.71	166.03	165.748	163.836	162.957	164.338
Test 18	174.686	169.07	167.783	162.794	167.39	156.464	162.66
Test 19	177.748	178.219	171.955	166.988	161.593	167.023	163.197
Test 20	170.714	172.866	165.224	165.164	164.199	166.278	160.434

Runtimes for 3DES implementation without Hardware

Boards Used	1	2	3	4	5	6	7	8
Test 1	847.804	429.942	381.856	320.019	282.54	262.31	241.184	232.225
Test 2	851.153	430.632	378.3	321.318	283.359	261.395	247.383	232.962
Test 3	845.761	423.807	380.165	322.086	288.78	260.113	244.375	225.077
Test 4	852.386	422.27	381.168	318.845	285.636	260.425	240.935	228.802
Test 5	845.457	424.05	379.168	319.405	286.125	260.832	244.574	229.251
Test 6	842.858	427.58	374.827	318.084	285.847	264.393	242.268	233.232
Test 7	851.849	425.996	377.3	315.306	283.493	259.379	240.017	231.936
Test 8	846.48	426.073	379.345	323.769	282.499	256.644	242.503	232.253
Test 9	848.813	431.173	377.886	319.865	283.672	261.401	241.485	227.335
Test 10	849.488	423.139	383.011	319.749	283.187	257.7	240.33	233.385
Test 11	848.261	422.728	379.738	319.743	283.224	261.368	243.252	231.83
Test 12	845.639	429.3	377.699	320.615	282.268	261.145	243.737	231.813
Test 13	846.268	426.169	375.018	316.851	284.847	260.8	243.88	231.248
Test 14	840.914	422.694	377.969	317.988	283.222	259.254	244.55	233.562
Test 15	851.509	429.828	376.994	315.575	287.966	259.486	244.223	229.557
Test 16	843.438	424.583	370.823	323.284	282.12	257.746	246.705	233.222
Test 17	851.972	428.006	380.217	319.7	284.541	261.253	242.221	227.768
Test 18	844.493	426.014	379.847	318.633	284.716	259.859	243.887	226.186
Test 19	844.562	426.395	381.071	322.323	285.119	260.96	241.763	229.946
Test 20	846.516	424.973	378.38	314.843	284.925	262.395	243.235	228.748

Boards Used	9	10	11	12	13	14	15
Test 1	222.507	216.754	206.949	202.884	199.331	193.06	192.602
Test 2	223.474	210.775	206.943	204.339	193.159	191.735	192.815
Test 3	219.76	207.824	209.999	202.029	198.533	196.964	185.628
Test 4	217.676	216.091	205.337	195.51	202.492	194.544	193.034
Test 5	215.859	212.941	208.743	205.748	200.416	193.631	186.954
Test 6	223.025	215.411	206.879	201.814	196.777	193.238	194.985
Test 7	223.51	215.059	207.631	202.627	193.755	195.725	188.764
Test 8	219.555	214.552	205.013	197.836	197.955	193.014	188.951
Test 9	221.47	214.37	209.393	204.636	200.39	195.533	199.484
Test 10	220.225	210.864	209.946	204.06	192.629	191.345	193.603
Test 11	219.415	214.504	205.39	202.602	192.228	194.665	194.489
Test 12	221.812	213.115	204.914	195.683	217.991	196.28	185.606
Test 13	224.313	216.035	205.726	203.215	198.178	194.754	190.602
Test 14	219.875	205.312	207.943	204.97	196.986	186.566	195.148
Test 15	214.661	213.005	205.716	201.1	199.198	196.919	186.614
Test 16	224.908	214.458	208.663	197.656	193.21	195.793	192.761
Test 17	219.31	214.914	205.571	200.445	197.358	195.212	191.816
Test 18	221.502	206.995	207.989	201.372	225.123	190.378	186.541
Test 19	221.391	216.575	207.602	205.176	202.173	194.923	191.566
Test 20	221.093	213.026	201.832	202.683	195.587	194.448	192.023

Timing of communication overhead

Boards Used	1	2	3	4	5	6	7	8
Test 1	131.359	128.399	120.819	117.876	113.261	114.004	114.777	115.999
Test 2	136.676	127.443	120.151	118.171	118.554	115.397	113.214	113.393
Test 3	136.202	125.628	124.661	123.729	111.431	114.423	115.802	114.083
Test 4	137.756	132.263	119.903	118.895	120.402	110.593	113.346	114.915
Test 5	139.062	126.294	113.593	122.492	117.782	118.164	118.656	115.438
Test 6	136.287	128.655	126.27	115.683	119.28	117.524	116.001	120.1
Test 7	138.886	126.809	116.099	122.344	115.548	117.607	110.373	115.108
Test 8	136.81	129.778	121.832	116.831	115.086	116.15	120.723	111.937
Test 9	130.892	125.665	118.99	119.788	116.074	116.83	114.499	115.916
Test 10	139.138	125.75	120.905	116.537	113.075	112.556	116.801	107.711
Test 11	135.771	127.221	118.896	118.352	117.819	116.781	114.738	119.924
Test 12	136.88	126.083	122.939	122.045	115.411	113.256	119.377	116.13
Test 13	138.24	128.02	118.712	120.565	119.728	108.866	114.104	117.33
Test 14	136.171	123.7	121.6	122.583	115.368	123.601	119.474	112.463
Test 15	133.949	126.098	120.413	117.835	116.51	108.244	113.274	115.738
Test 16	137.606	123.314	123.55	120.498	115.774	122.014	113.518	114.464
Test 17	134.84	131.139	117.377	118.752	111.706	117.005	115.288	116.815
Test 18	140.548	125.29	122.797	122.738	124.607	119.162	113.396	116.218

Test 19	135.378	128.955	117.715	117.317	115.101	115.038	110.732	110.08
Test 20	135.696	126.883	118.983	119.579	118.602	117.773	120.467	118.451

Boards Used	9	10	11	12	13	14	15
Test 1	113.98	115.452	111.489	114.023	115.999	114.667	114.089
Test 2	116.585	119.939	116.899	111.873	115.152	116.218	112.61
Test 3	114.842	112.304	114.758	107.644	117.437	112.156	114.259
Test 4	114.953	113.752	113.322	113.871	113.758	114.158	113.754
Test 5	112.921	115.397	118.004	114.019	115.223	112.408	118.2
Test 6	117.222	114.24	109.46	116.437	116.672	108.947	110.532
Test 7	114.216	115.162	118.294	114.531	112.903	122.307	114.699
Test 8	113.228	114.161	112.659	114.4	107.838	108.285	118.416
Test 9	115.479	115.053	116.958	117.055	122.235	116.179	114.506
Test 10	115.248	112.58	114.479	113.594	114.1	105.579	114.287
Test 11	115.593	115.095	115.559	117.815	115.541	119.85	114.061
Test 12	114.062	113.016	115.1	113.951	115.949	113.441	114.733
Test 13	115.779	115.614	114.544	114.265	116.734	113.117	114.522
Test 14	113.149	114.325	115.492	113.104	114.228	115.919	115.324
Test 15	113.59	115.633	113.962	113.039	114.855	113.813	112.68
Test 16	118.43	115.133	113.391	112.827	115.936	116.703	108.802
Test 17	113.771	116.391	114.74	112.652	117.297	111.321	118.719
Test 18	129.525	117.371	108.915	115.791	112.78	115.289	115.694
Test 19	113.223	115.535	119.348	112.527	113.454	114.317	113.832
Test 20	116.104	113.789	112.28	113.554	115.587	115.427	112.847

BIBLIOGRAPHY

[1] The Stone Souper Computer, http://stonesoup.esd.ornl.gov/

[2] MPICH, http://www.mcs.anl.gov/research/projects/mpi/mpich1/

[3] LAM-MPI, http://www.lam-mpi.org/

[4] Open-MPI, http://www.open-mpi.org/

[5] CRAY, Inc., "Cray XD1™ System Overview"; October 2005.

[6] Impulse Accelerated Technologies, http://www.impulseaccelerated.com/

[7] Join Java, http://joinjava.unisa.edu.au/

[8] System C, http://www.systemc.org/home

[9] Li Shen, C. C., "Evaluating Impulse C and Multiple Parallelism Partitions for a Low-Cost Reconfigurable Computing System," M.S. Thesis, Dept. Eng., Baylor University, Waco, TX, December 2008.

[10] XUP V2P Documentation, http://www.xilinx.com/univ/xupv2p.html

[11] Digilent, Inc., http://www.digilentinc.com/

[12] QNX, http://www.qnx.com

[13] MonteVista, http://www.mvista.com/

[14] Donaldson, J.W., Porting MontaVista Linux to the XUP Virtex-II Pro Development Board M.S. Project, Department of Computer Science, Rochester Institute of Technology, Rochester, NY, August 22, 2006.

[15] Debian, http://www.debian.org/

[16] Crosstool, http://www.kegel.com/crosstool/

[17] D. Pellerin and S. Thibault, *Practical FPGA Programming in C.* Upper Saddle River, NJ: Prentice Hall Professional Technical Reference, 2005.