YILDIZ TECHNICAL UNIVERSITY
FACULTY OF ELECTRICS - ELECTRONICS
COMPUTER ENGINEERING DEPARTMENT

SENIOR PROJECT

FAST DATA TRANSFERRING VIA NETWORK USING CELL BE PROCESSOR

Project Supervisor: Yrd. Doç. Dr. Sırma YAVUZ

Project Group

03011504 Aytunç BEKEN

04011065 Burak ÇAKIL

İstanbul, 2008

CONTENTS

ABBREVIATION LIST

MTMP	Multi-Threaded Multi-Ported
CELL BE	Cell Broadband Engine
PPE	Power Processing Element
SPE	Synergistic Procesing Element
EIB	Element Interconnect Bus
IP	Internet Protocol
MFC	Memory Flow Controller
SPU	Synergistic Processing Unit
PPU	Power Processing Unit
SP	Single Precision
DP	Double Precision
API	Application Programming Interface
GUI	Graphical User Interface
DMA	Direct Memory Access

FIGURE LIST

TABLE LIST

PREFACE

As the developers, we expect this project to be a perfect solution for everyone that needs massive data transfer, especially business users. We also foresee that this project will be a great progress in Information Technology. We would like to thank to everyone who contributed to this project, foremost, to our project supervisor Asst. Prof. Sırma Yavuz and all the other instructors in the Computer Engineering Department and Server Systems Operations Manager of IBM Turkey, Serkan Şahin. We also would like to thank Turhan Karadeniz and Onur Can Ulusel for their technical support on Cell BE Processor, and Arda Durukan for IBM Turkey University Relations for his additional support.

ÖZET

Proje çalışması olarak Cell BE işlemcisinin sağladığı teknik özelliklerden faydalanarak ağ üzerinden hızlı ve güvenli veri transferi gerçekleştirilmesi amaçlanmaktadır. Proje kapsamında Multi-port Bağlantı ve Cell BE işlemcisinin getirdiği gelişmiş multi-threading desteğinden faydalanılacaktır. Gerçekleştirilen uygulama ile, büyük boyulardaki dosyalar, yüksek çözünürlüklü görüntü ve ses gibi verilerin, yüksek hızlarda ve güvenli şekilde iki istemci bilgisayar arasında transferi sağlanacaktır. Uygulama, gerek şifreleme gerekse yönetilebilirlik açısından en optimum çözümleri sunmaktadır. Bunlarında arasında, veri gönderiminde kullanılacak olan RSA Şifreleme Algoritması gibi modüller bulunmaktadır.

ABSTRACT

The main goal of this project is to implement a fast and secure data transfer via network by using the technical features of the Cell BE processor. In the project, multi-port connection and multi-threading technologies will be taken advantage of. The implementation of the project will make possible to move large amounts of data like, high definition graphics and sound, between two peers (computers) . The application offers optimum solutions for both security and manageability. These solutions include the RSA algorithm which will be used for encryption.

1. INTRODUCTION

As technology evolves, Internet connection speed raises day by day and neccessity of stream transfering becomes more important. Today, thousands of computers are transferring streams like video, sound, file, text over the internet or over the network and some of these compters are serving as big servers of companies,banks,governments, etc. These servers, mainframes transfer huge amount of streams over the network for the purpose of backing up data and transfering from one place to another. On these processes there are some important considerations like speed and security. This project brings a new solution to these considerations.

This project uses a new algorithm which is developed by the project group for transmitting massive streams on high speed and on secure environment. The algorithm is based on multi-threaded stream transmission on mutli-ported peer-to-peer connection and parallel programming. Further information about the algorithm can be found on Chapter 1-1 System Architecture. Results of the researches showed that, the most appropriate hardware for this project on current conditions is the Cell BE Proccessor.

Cell BE Proccessor is a microprocessor architecture jointly developed by Sony Computer Entertainment, Toshiba, and IBM. Cell combines a general-purpose Power Architecture core of modest performance with streamlined coprocessing elements (cores) which greatly accelerate multimedia and vector processing applications, as well as many other forms of dedicated computation. This architecture shows great performance on MT-MP Stream Transmission algorithm. Further information can be found on Chapter 1-3 Cell Architecture.

On this Project, parallel programming techniques are used for implementation of multi-threaded programming. The RapidMind Multi-core Development Platform[2] enables software engineers develop manageable, single-threaded applications that leverage the full potential of multi-core processors from AMD® and Intel®. Seamlessly take advantage of the application acceleration available from GPUs and the Cell Broadband Engine™. Further information can be found on Chapter 1-2, and Chapter 3-1.

1.1. System Architecture

In this chapter detailed information about the algorithm will be given. As mentioned before, algorithm is based on multi-threaded multi-ported stream transmission. The aim of the algorithm is to put the stream into piecss with independent threads and send the stream simultaneously to receiver. The receiver takes the streams by independent threads. All the theads runnig on both sides, divide streams into pieces sized equal to the buffer size which is going to be used globally between both sides. After one thread takes a piece of stream it sends it to the receiver immediately.

On the recevier side, threads wait for the streams. After one thread receives a stream, it writes the stream on a temporary location. After the whole transmission is finished, receiver side assembles all stream pieces together. Graphical description of algorithm can be seem on the Figure 1-1. This approach was later modified to be able to support multiple file transmission. In this algorithm, threads are created equal to the port count specified as well, but each thread handles one file at a time. When a thread finishes sending a file, it immediately starts sending another file in the queue, if there is any. This change is also emphasized in Chapter 4 – Progress of The Project

1.1.1. System Main Modules

In this chapter, main modules of the algorithm is detailed.

1.1.1.1 Program Main Control

Program main module has several functions. One of them is the entrance of the program. In the program logic, user must run the program as root. Another main function controls the program logic, which takes commands from user and parses them. After correct parsing, function calls neccessary other functions to execute the command.

There are also other functions that control buffer size management, port management, username and password management.

1.1.1.2 Multi Threaded Streaming Module

This module creates threads equal to the number of available ports that is given by the user, and it opens connections. The main purpose of the module is to read the files from the file list, and assign each file to a thread. Assigned files then are read to buffer for transmission.

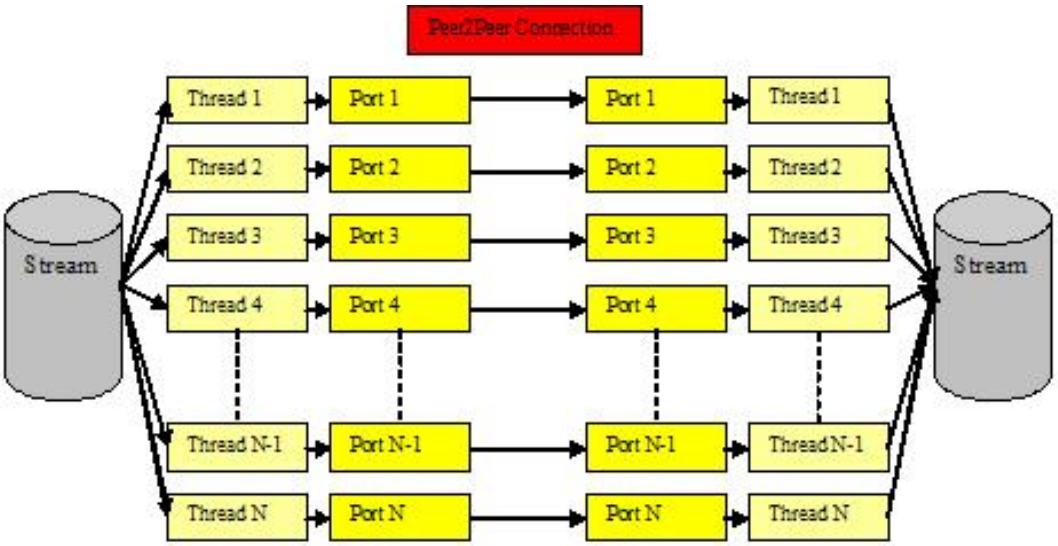


Figure 1-1 System Architecture

1.2. Cell BE Processor

Cell BE (Broadband Engine) Processor is a microprocessor developed by IBM, Toshiba and Sony Corporations. Cell combines a general-purpose Power Architecture core of modest performance with streamlined coprocessing elements which greatly accelerate multimedia and vector processing applications, as well as many other forms of dedicated computation [1].

Cell BE processor architecture consists of 9 cores. One of the cores is named Power Processing Element (PPE) which controls the other 8 cores . These cores are named Synergistic Power Elements (SPE). Each SPE work seperately from each other, which means that up to 8 simultaneous threads could run in a Cell BE processor. This applies if a single thread is running on each SPE.

The part that connects these 8 SPE and the PPE is called Element Interconnect Bus (EIB), this bus also connects these elements with the memory controller, and the I/O ports. The bandwidth of the EIB can theoretically increase up to 204.8 GB/s

An undetailed version of the Cell BE architecture can bee seen in Figure 1-2 below.

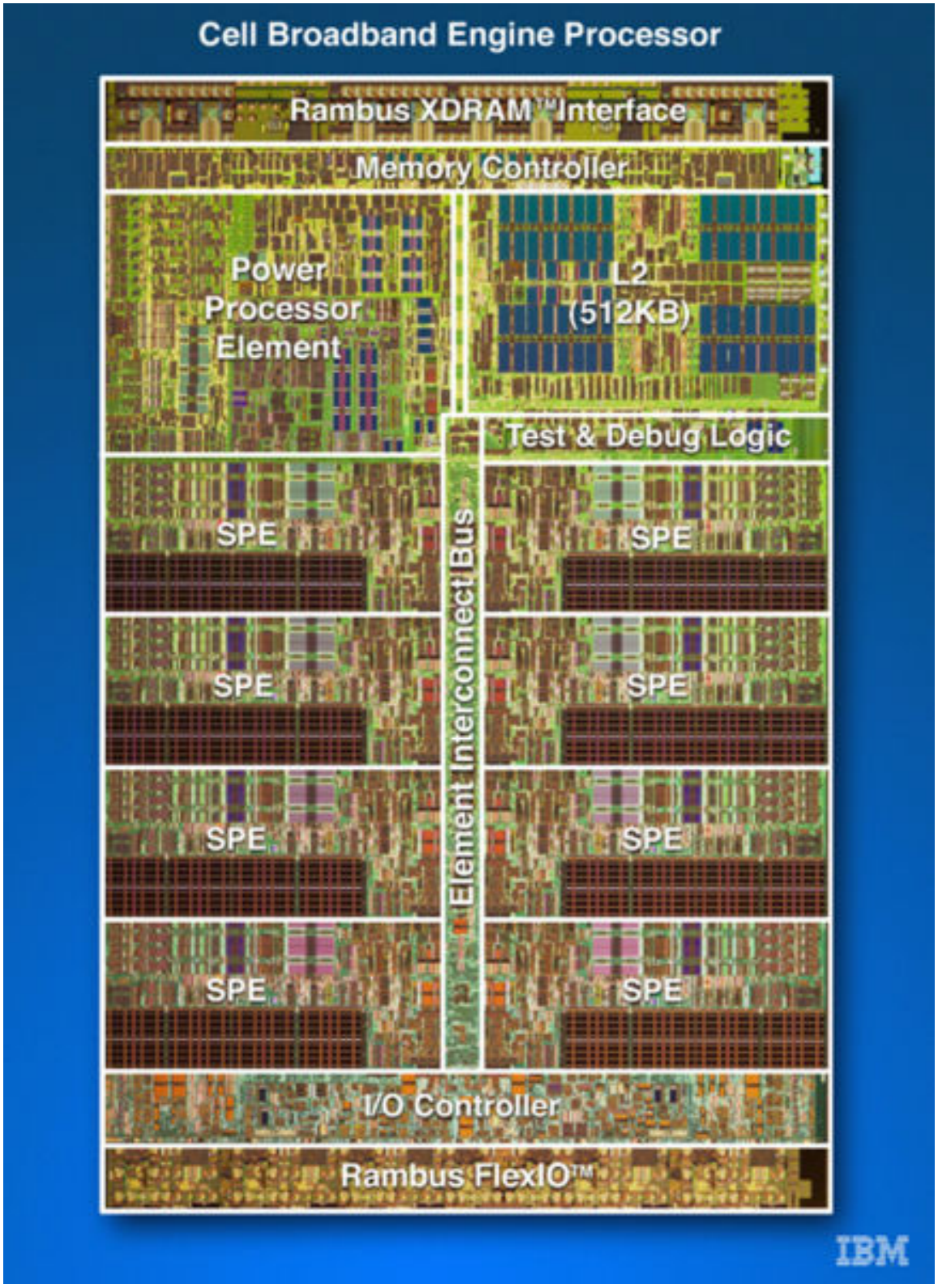


Figure 1-2 Cell BE Architecture

1.3. RapidMind Multi-Core Development Platform

Software developers are using RapidMind today to create manageable, single-threaded applications that leverage the full potential of multi-core processors from AMD® and Intel®. In addition, RapidMind allows developers to seamlessly take advantage of the application acceleration available from GPUs and the Cell Broadband Engine™ [2].

1.3.1. Impact of multi-core on software development

Multi-core processors offer tremendous performance gains, but few applications take full advantage of this new technology because of the significant complexity of parallelizing across the multiple cores.

Applications that are not multi-core enabled will suffer a performance decrease as it will only run on a single core, and will not scale as the number of cores increases.
While efforts to multi-thread an application may take advantage of multiple cores, these projects are ambitious, time-consuming and error-prone. Multi-threaded applications are harder to develop and test, which requires a level of development expertise which is difficult to find [3] . Software organizations are all too aware of the real fear of releasing an unstable solution that quickly fails in the field.

1.3.2. Gain a competitive advantage with RapidMind

Multi-core processing presents an opportunity for software organizations to gain a competitive advantage. The award-winning RapidMind Multi-core Development Platform simplifies the development of parallel applications, reducing the cost and timelines of software development when compared to multi-threaded projects, and greatly improves the likelihood of project success.

1.3.3. RapidMind benefits

Improve application performance by over 10 times
More quickly build and deliver multi-core capable applications
Leverage multi-core using existing development expertise
Use your existing development practices, tools and compilers
Automatically scale your application to an increasing number of cores

1.3.4. Productivity

Software developers focus their skills on the application and not the underlying processor.

1.3.5. Performance

Resulting application fully leverages the potential of the processor and all its cores.

1.3.6. Portability

Applications are hardware independent and will automatically scale to additional cores and future multi-core processors.

1.3.7. Using RapidMind

Unlike typical multi-threading approaches, RapidMind is a development and runtime platform that enables single-threaded, manageable applications that fully leverage multi-core processors. With RapidMind, developers continue to write code in standard C++ and use their existing skills, tools and processes and the RapidMind platform then “parallelizes” across multiple cores.

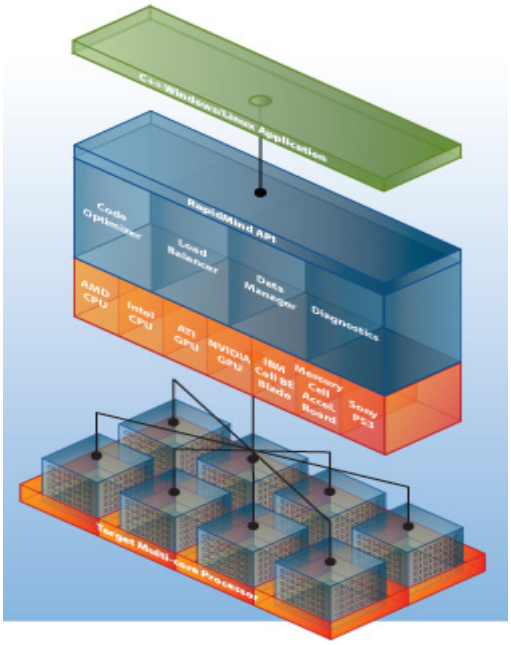


Figure 1-3 RapidMind System Diagram

1.3.8. API

Intuitive, integrates with C++, and requires no new tools, compilers or workflow

1.3.9. Platform

- Code Optimizer analyzes and optimizes computations to remove overhead
- Load Balancer plans and synchronizes work to keep all cores fully utilized
- Data Manager reduces data bottlenecks
- Diagnostics detects and reports performance bottlenecks

2. FEASIBLITY ANALYSIS

2.1. Technical Feasibility

Technical Feasibility can be observed in two different concepts: software and hardware.

2.1.1. Software Used In The Project

The programming language the project is written in was chosen as C++ due to being a fast and reliable language. It also meets the project's needs by its advanced port and thread controlling ability. Last but not least, C++ is currenlty the best programming language amongst the languages that can be used for programming Cell BE processors.
For the purpose of passing the project into multi-core availability phase, a software named RapidMind is used. RapidMind easily transforms the code into a form that can be used to program a Cell BE processor,like adding multi-core capabilities, without the need of additional coding.

Linux was chosen as the operating system because of its suitability for programming environment. The C++ libraries included in Linux enables port and thread programming, making the coding process much easier. Additionally, RapidMind software is compatible with Linux operating system.

As development environment, Eclipse Development Platform was chosen, because of its easy-to-use and user-friendly interface.

As being used in Linux, these softwares are entirely open-source therefore they are free to use and distribute.

2.1.2. Hardware Used In The Project

The hardware used in the project consists of 2 client machines that both have Cell BE processors, such as PlayStation3 game consoles, which cost approximately 850 YTL each. Additionally, a cross conected cat-5 ethernet cable is used for connecting the clients together.

2.2. Economical Feasibility

The total cost of this project comes from the hardware used, and labor. The cost of labor is estimated as assumed that 2 people have developed the project. The Approximate total cost is listed below.

2x Sony PlayStation3 Game Console	1700 YTL
1x Cat5 Ethernet Cable	6 YTL
2x Labor Cost 484 Work Hours 30$ Per Hour	29040 YTL
Total Cost	30746 YTL

Table 1-1: Total Cost Distribution

2.3. Legal Feasibility

This application have no legal disorders since it is programmed on fully open-source software, which are protected by GPL (General Public License). The use of this application is not against any kind of law.

2.4. Gantt Chart

The Gantt Chart of the project can be seen on Figure 2-1 below.

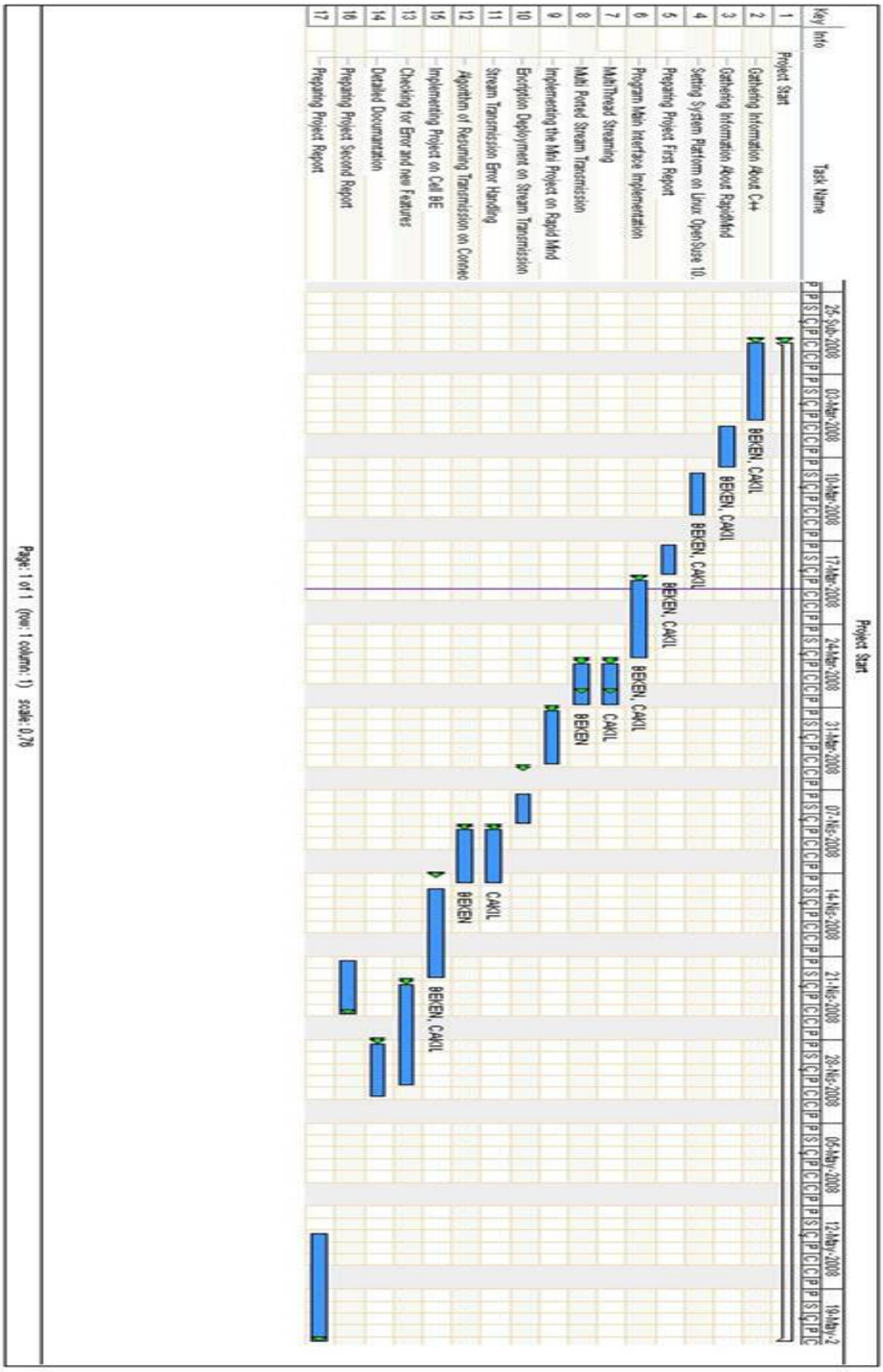


Figure 2-1 Gantt Chart of The Project

3. IN-DEPTH ANALYSIS

3.1. Detailed Architecture of Cell BE Processor

As previously specified, Cell BE Processor has 1 main core called Power Processing Element (PPE) and 8 subcores called Synergistic Processing Elements (SPEs) The PPE has an architecture of an 64-bit IBM PowerPC running on a frequency of 3.2 GHz. The PPE has a 32 KB L1 cache and a 512 KB L2 cache. L1 cache is located inside the Power Processing Unit (PPU) which is the main core of the processor and executes the operations. L1 cache conducts the communication between the PPU and L2 cache, while L2 cache does the same thing between the PPU and the EIB (Element Interconnect Bus). Internal architecture of the PPE can be seen on Figure 3-1.

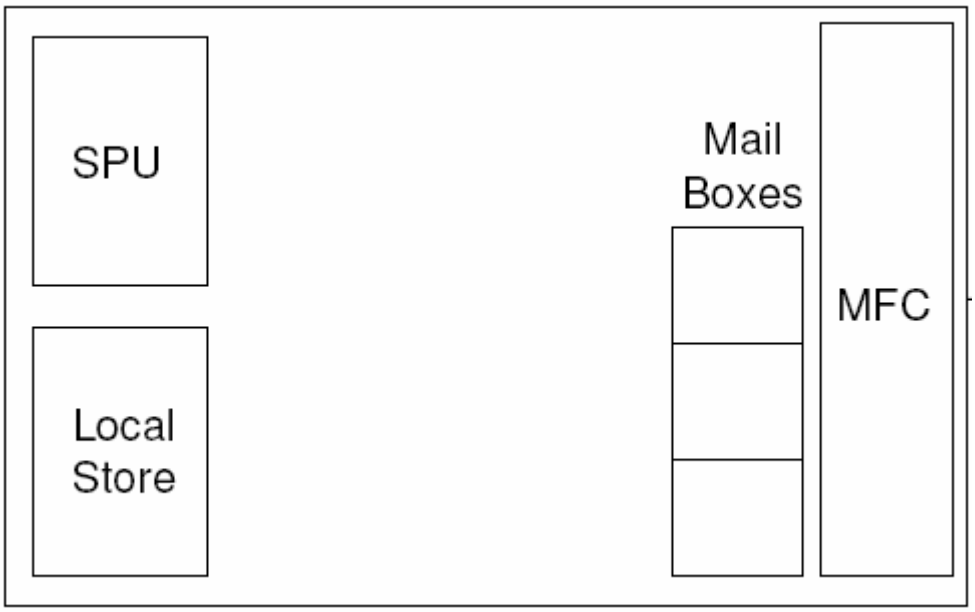


Figure 3-1 Internal Architecture of an SPE

The SPEs are total of 8 single-threaded co-processors which operate independently from each other. However, the SPEs are not able to work on their own; they are obligated to wait for commands from the PPE. Working simultaneously, the SPEs form the real power of the processor. Each SPE has a local store of 512 Kbytes acting as a memory for local operations. These local stores can be adressed both locally and globally. The SPEs also support external storages with a maximum size of 4 Gbytes. They also have a Memory Flow

Controller (MFC) which controls data flow between the SPEs and the EIB, thus the PPE. Internal architecture of the SPE can be seen on Figure 3-2.

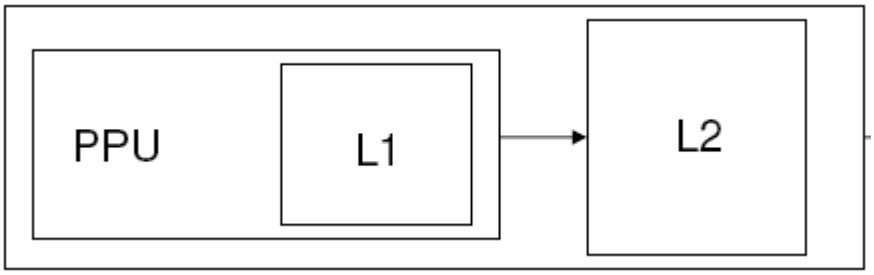


Figure 3-2 Internal Architecture of a PPE

The EIB is the communication bus between The PPE, SPEs and the Memory Controller Units. This unit works on a speed of 96 bytes per clock cycle to be able to address all 8 of the SPEs at the same time. Maximum bandwidth of the EIB is over 200 Gbytes per second.

Looking overall, Cell BE is a processor which consists of 241 Million transistors and has an operation capability of 200 GFlops with Single Precision (SP) and 20 GFlops with Double Precision (DP). It can reach to a top frequency of over 4 GHz. The complete in-depth architecture of the Cell BE can be seen on Figure 3-3, the performance summary table in terms of operation types can be seen on Table 3-1 and a performance comparison chart can be seen on Figure 3-4.

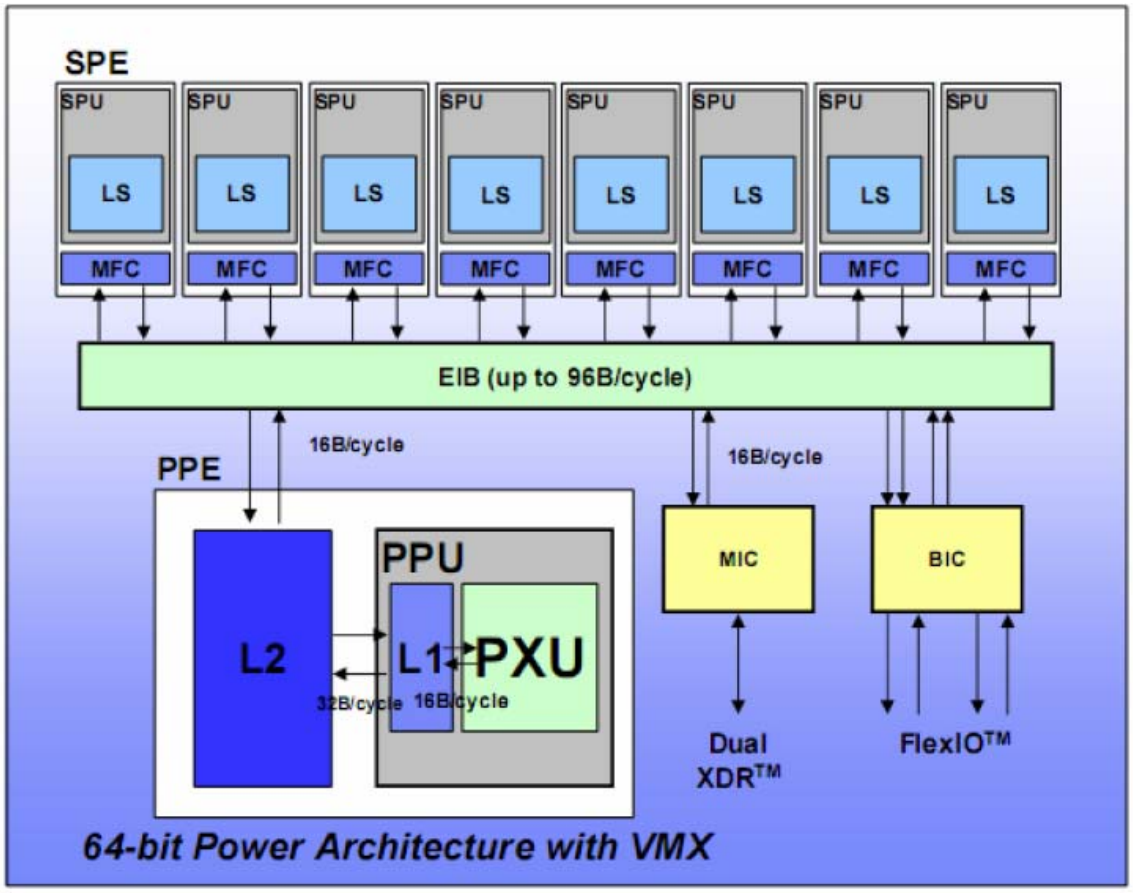


Figure 3-3 In-Depth Architecture of Cell BE Processor

Type	Algorithm	3 GHz GPP	3 GHz BE	BE Perf Advantage
HPC	Matrix Multiplication (S.P.)	25 Gflops	190 GFlops (8SPEs)	8x
	Linpack (S.P.)	18 GFlops (IA32)	150 GFlops (BE)	8x
	Linpack (D.P.)	6 GFlops (IA32)	12 GFLops (BE)	2x
bioinformatic	smith-waterman	570 Mcups (IA32)	420 Mcups (per SPE)	6x
graphics	transform-light	160 MVPS (G5/VMX)	240 MVPS (per SPE)	12x
	TRE	1.6 fps (G5/VMX)	24 fps (BE)	15x
security	AES	1.1 Gbps (IA32)	2Gbps (per SPE)	14x
	TDES	0.12 Gbps (IA32)	0.16 Gbps (per SPE)	10x
	MD-5	2.68 Gbps (IA32)	2.3 Gbps (per SPE)	6x
	SHA-1	0.85 Gbps (IA32)	1.98 Gbps (per SPE)	18x
communication	EEMBC	501 Telemark (1.4GHz mpc7447)	770 Telemark (per SPE)	12x
video processing	mpeg2 decoder (sdtv)	200 fps (IA32)	290 fps (per SPE)	12x

Table 3-1 Performance Table of Cell BE

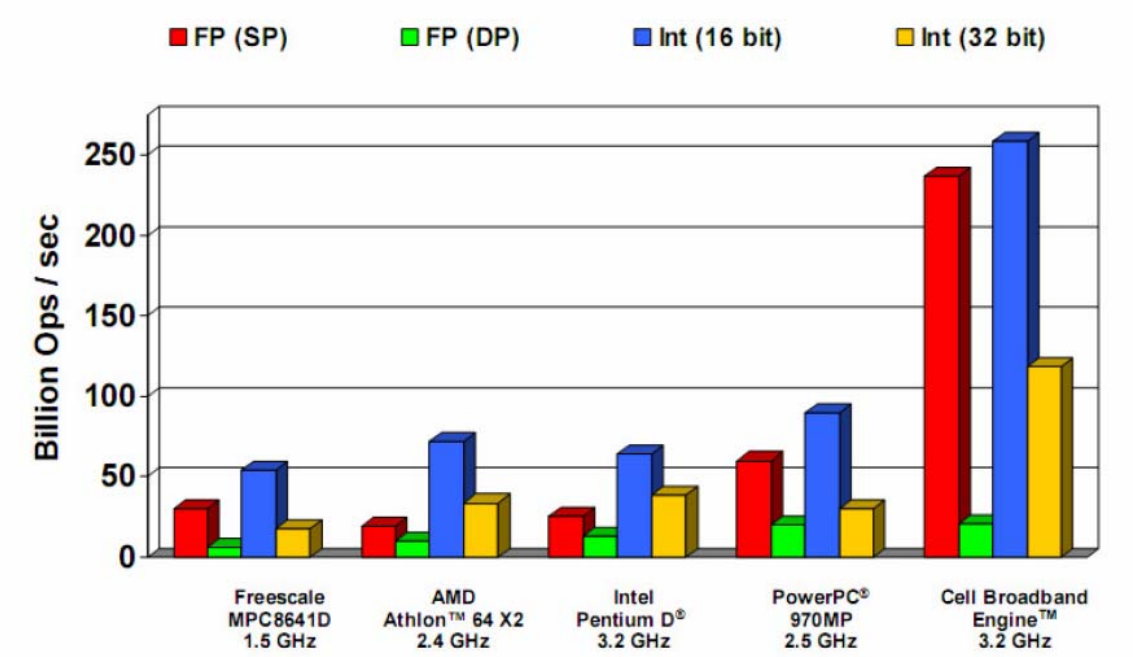


Figure 3-4 Performance Comparison of the Cell BE Processor

3.2. User commands of the Program

The user interface of the application has several input commands which enables the user to manipulate the configuration, such as username, password, packet (buffer) size, ports to be used, data to be sent, etc. Usage of all these commands are listed in this chapter.

3.2.1. Program Initialization

The Initialization of the program is auto-secured by the internal user authorization mechanism of Linux operating system. Therefore, only users who has the rank of root can access the program. After logging in as root, the application can be initialized by typing

./Cell

in the terminal. The program needs **ncurses** library to be able to run. For more detail on ncurses library, refer to Chapter 3-3.

3.2.2. Set Commands

Set keyword is used to change username, password ,buffer size, maximum and minimum buffer size. This command gets parameters, the first of which is to specify the

configuration to be set, and if neccessary the second is to specify the new value of the configuration. Set operations writes new configurations to config file of the program. Avaliable set commands are as follows:

set –b

This command sets the buffer size of the program which is going to be used on transmisson on both sides. This commnad lists the avaliable buffer size values and enables the user to make a choice among them. But the value of the chosen buffer size must be between the local maximum and minimum buffer size values.

set –maxb

This command sets the local maximum buffer size. This value is important for remote side for the transmission. Remote buffer size value must be between the max and min values of the receiver side.

set –minb

This command sets the local minimum buffer size of the system.

set –u <username>

This command changes the local username of the system. As mentioned before, username is a must to enter the program. The parameter <username> is set as the new username of the program.

set –p

This command sets the new local password of the system. User must enter the old password, then enter the new password twice. If the new passwords match, the new password is written to system.

set –maxpc <maxportcount>

This command sets the maximum number of ports that is going to be used on transmission.

set –minpc <minportcount>

This command sets the minimum number of ports that is going to be used on transmission.

3.2.3. Get Commands

Get keyword is used to get configuration and system values. Get command takes one argument that specifies the options. Command prints the value to the screen and do not change the value of the option.

get –b

This command prints the buffer size value on screen.

get –maxb

This command prints the local maximum buffer size value on screen.

get –minb

This command prints the local minimum buffer size value on screen

get –maxpc

This command prints the local maximum port number on screen.

get –minpc

This command prints the local minimum port number on screen

get –p

This command lists the avaliable ports which are added by user and will be used on transmission.

get –u

This command prints the local username of the program on screen.

3.2.4. Add Command

Add keyword is used for adding ports to the system configuration. These ports is neccessary for transmission. The number of ports that is added must be between local maximum and minimum port count.

add –p <portnum>

This command add the port to the port list. If the number of ports reaches the maximum port number then no add operation is done.

add –p <start_port_number>-<finish_port_number>

This command add the ports within the given interval. If the number of ports reaches the maximum port number then no add operation is done.

3.2.5. Rem Command

Rem keyword is used for removing ports from the port list.

rem –p <port_number>

This command removes the given port number from the port list and writes changes to configuration file.

rem –p –all

This command flushes the port list of the system.

3.2.6. Send Command

Send command is used to start the transmission and send a file with. Send command takes 2 arguments which are neccessary for the transmission to be initiated.

send –u <username> -ip <remote_ip_address>

The parameter username is the value of remote side. This authentication is neccessary for the transmission to start. The remote_ip_address is the IP address of the remote system. When this command is entered, the system asks for the remote password. The transmission is started just after the password is entered correctly. The list of files to be sent is read from the file “files.zen” located within the same directory.

3.2.7. Exit Command

This command is neccessary for exiting program.

3.2.8. Reset Command

This command resets the configuration file to default settings.

3.3. Ncurses Library

Ncurses library is a programming library providing an API, allowing the programmer to write text user interfaces ina a terminal- independent manner. It's a toolkit for developing "GUI-like" apps which run under a terminal emulator. It also optimizes screen changes, in order to reduce the latency experienced when using remote shells [4].

Ncurses library functions are extremely useful, enabling the programmer to navigate the terminal cursor freely, thus printing anything to or getting input from anywhere on the terminal screen. In this way, the terminal console can be used like a GUI screen.

The programmer can also create sub-windows, which he can use independently from other windows or the console itself. The border and text styles, attributes, colors etc. can be adjusted as well.

One of the best features of ncurses library is the ability to use the mouse. All the mouse events like click, double click, move, release button can be captured, and a handler function can be triggered with these events.

Additionally, features like menus, forms, panels can be created and functionalized easily with the ncurses library. The user interface of the console version of a well-known application, Yast, is programmed using the ncurses library. A screenshot of this application can be seen on figure 3-5.

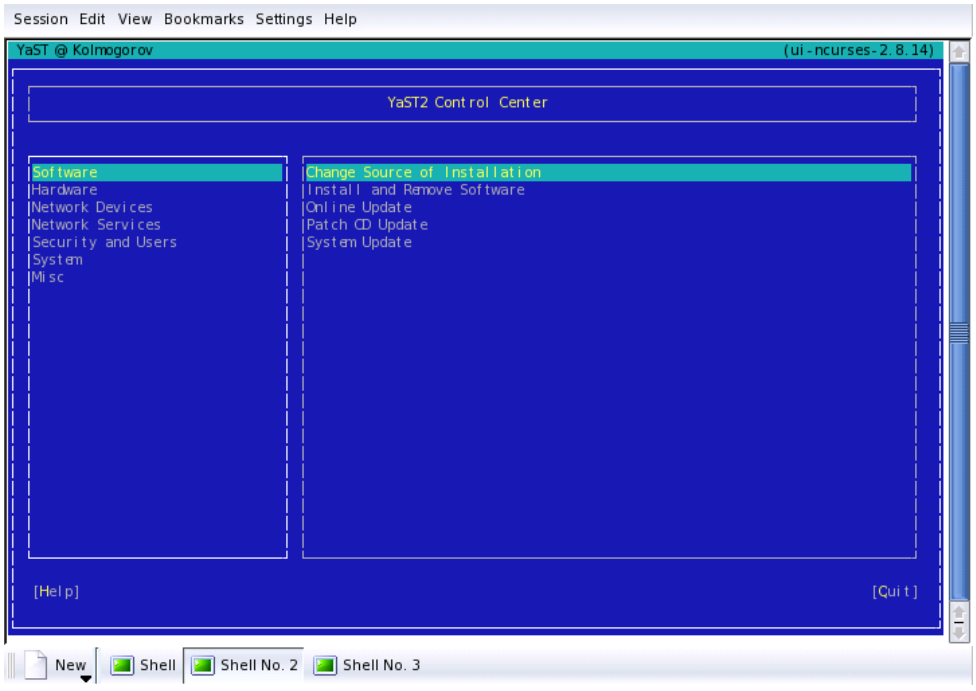


Figure 3-5: A Screenshot of Yast: An Ncurses Application

3.4. SPE Programming

The architecture of Cell BE Processor allows the developer to program each SPE to do its own job, which brings an advantage over RapidMind during development process. With this technique, despite descending to a lower level, programming SPEs becomes easier, so that the programmer can feel he has full control over the hardware. This method uses standart C/C++ language, plus an SPE library which brings along the functions to program each SPE seperately.

A serious matter which should be taken into consideration is that the SPEs does not have access to main memory and they use their own memory blocks called 'Local Stores' each of which has only 256 KB of memory. Because of this, SPEs can not use buffers of considerable sizes. This also reduces performance in file transmission, as well as file access (read/write). There was nothing to do for the buffer size at the final stage of the project, but for file access, the SPEs has to use the main memory. Thanks to the Cell architecture, there is a way. This way is called Direct Memory Access (DMA). With DMA, SPEs can gain access to main memory, without the obligation to communicate with the main processor, PPE. Memory read and write operations can be done with a few C functions.

Another problem encountered was that, since the SPEs run seperate hardware threads, they all try to access the same resource at the same time. This results in corruptions in data transmission like, the receiver can not receive the file completely, buffer reads the wrong part of the file, etc. To solve this serious problem, a global mutex (mutual exclusion) logic was implemented. This logic locks the access to the resource when a SPE reaches and reads from / writes to it. The other SPEs have to wait till the SPE in action finishes its job with the resource. The other processes can be done completely parallelized, without the SPEs being forced to wait each other. This mutex logic was used in several parts of the program, to ensure that the working threads don't cause a starvation or deadlock.

4. PROGRESS OF THE PROJECT

4.1. Implementation Changes

During the development period, the content of the project went through some changes. These changes are specified briefly below.

There has been some serious developments on visual interface. The project now uses the ncurses library to make the interface more user-friendly. There are a few screenshots showing the user interface in Appendix 1 to 5.

The plan to use an algorithm like RSA or AES on data encryption has changed due to the performance issues. After some research, It is turned out to be more feasible to encrypt the session and the hand-shaking packages only instead of encrypting all of the data packages to be sent to the receiver side. Thus, the hand-shaking packages are encrypted with RSA algorithm [5].

Single file transmission has been upgraded to multiple file transmission. In the current version, user-specified number of files can be transmitted at the same time. Each created thread handles single file transmission and once one file transmission is finished, the idle thread passes on to the next file at the queue.

The authentication mechanism to start the program has been removed and the Linux user authorization method is implemented. Thus, only root users have the authority to access the application.

RapidMind Development Platform was abandoned due to data type incompatibilities. Instead, a simple RSA application was developed to test the features of RapidMind, and a near 8x speed improvement was achieved on a Cell BE processor.

4.2. Performance Benchmarks (Without Parallelization)

In this chapter, a benchmark research of the program is made to compare transmission results in terms of speed, by changing buffer size and port count. The comparison table is below. As a notice, these results are taken **before** the application is parallelized with RapidMind.

	Total File Size	
Buffer Size	1GB (4 Ports)	1GB (2 Ports)
512 KB	88 s	87 s
1 MB	87 s	88 s
2 MB	88 s	88 s
4 MB	88 s	88 s
8 MB	89 s	88 s
SFTP (Secure File Transfer Protocol)	1st try: 149 s 2nd try: 102 s 3rd try: 89 s 4th try: 103 s 5th try: 89 s Average Time: **106.4 s**	

Table 4-1 Performance Comparison On a x86 Processor

4.3. Performance Benchmarks (After Parallelization)

After the Cell BE migration, the stats had a slight change. As specified in the Conclusion part, the reason for this is that the Cell BE processor architecture turned out to be not as suitable as dual core high speed processors, since the SPEs are designed to do complex mathematical calculations rather than doing I/O operations. But the power of the algorithm makes this application still far in front of SFTP. The stats can be seen on Table 4-2. The reason of the question mark at the bottom row is that a single 4 GB file could not be created using Cell BE Processor.

	Cell BE Transfer	SFTP
1 GB (1 Port)	96 secs	130 secs
1 GB (4 Ports)	91 secs	130 secs
2 GB (2 Ports)	184 secs	254 secs
4 GB (4 Ports)	369 secs	?

Table 4-2 Performance Comparison on Cell BE Processor

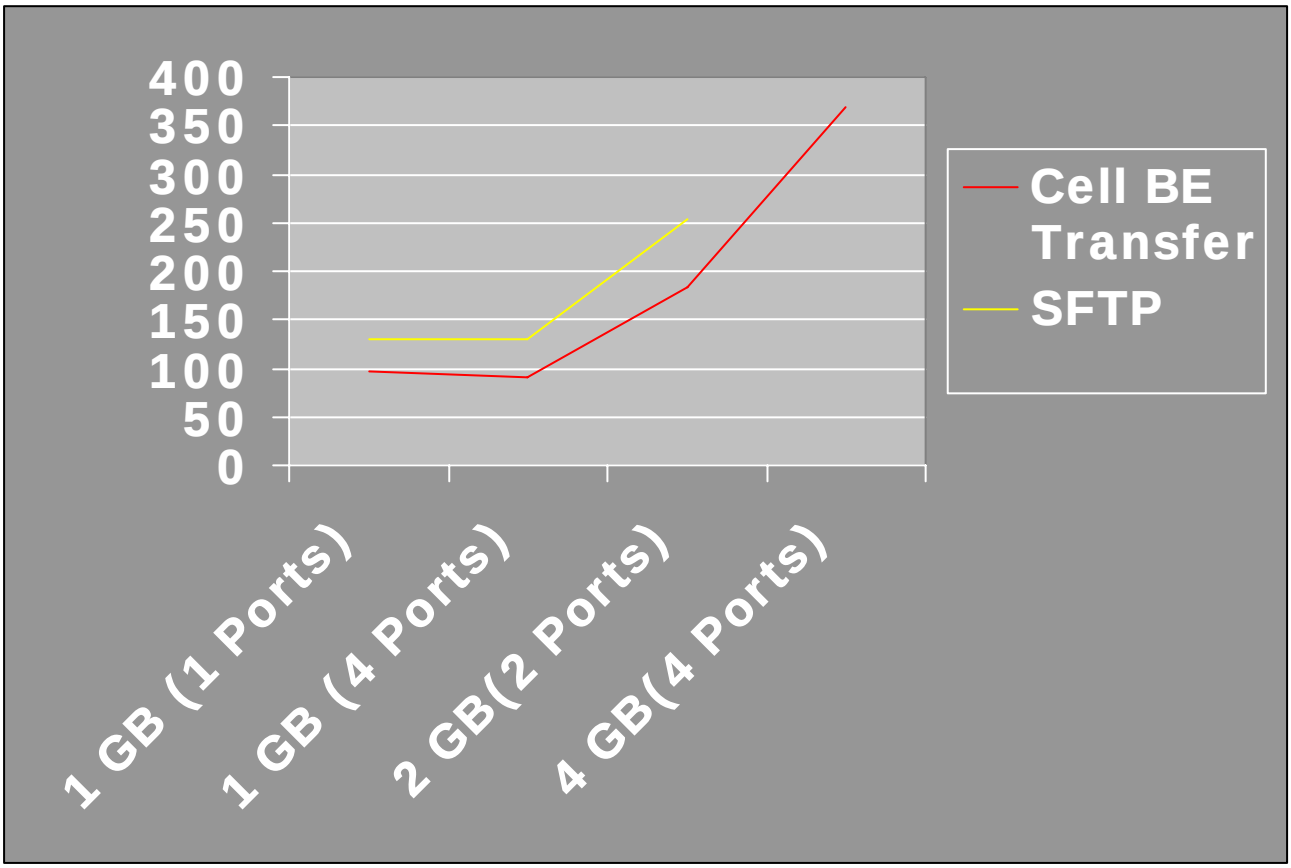


Figure 4-1 Performance Comparison as Graphics

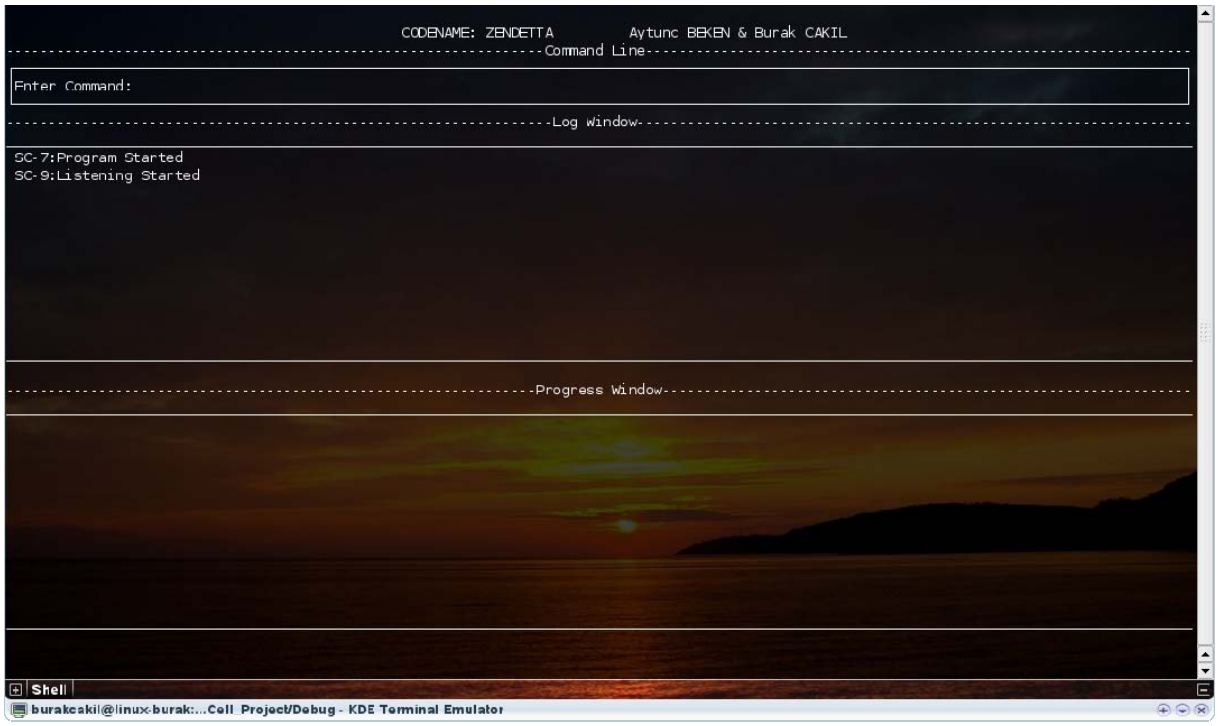
CONCLUSION

The project made it clear that with the maximum use of the existing transmission capabilities, the transmission speed could be increased. This also can help on maintaining the transmission speed on a stable level. But a disadvantage of this is that the user will not be able to use the network between peers efficiently while the transmission is resuming. If he does, he will either have to sacrifice efficiency from the application, or the other applications he uses.

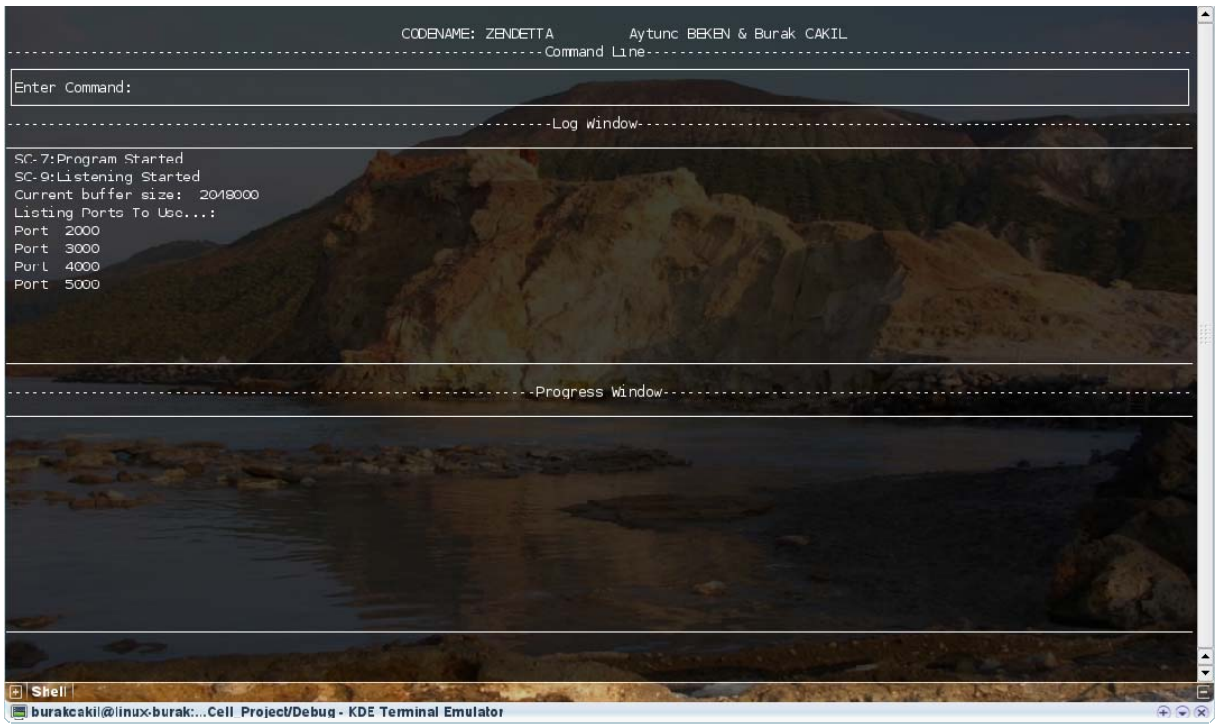
It was also seen that Cell BE processor architecture is not a perfect match for file transmission, since the results from PCs are far better, and local stores of the SPEs have too little memory to construct a useful buffer. But with Cell programming, an application based on mathematical calculations can be much faster on Cell BE processors.

APPENDICES

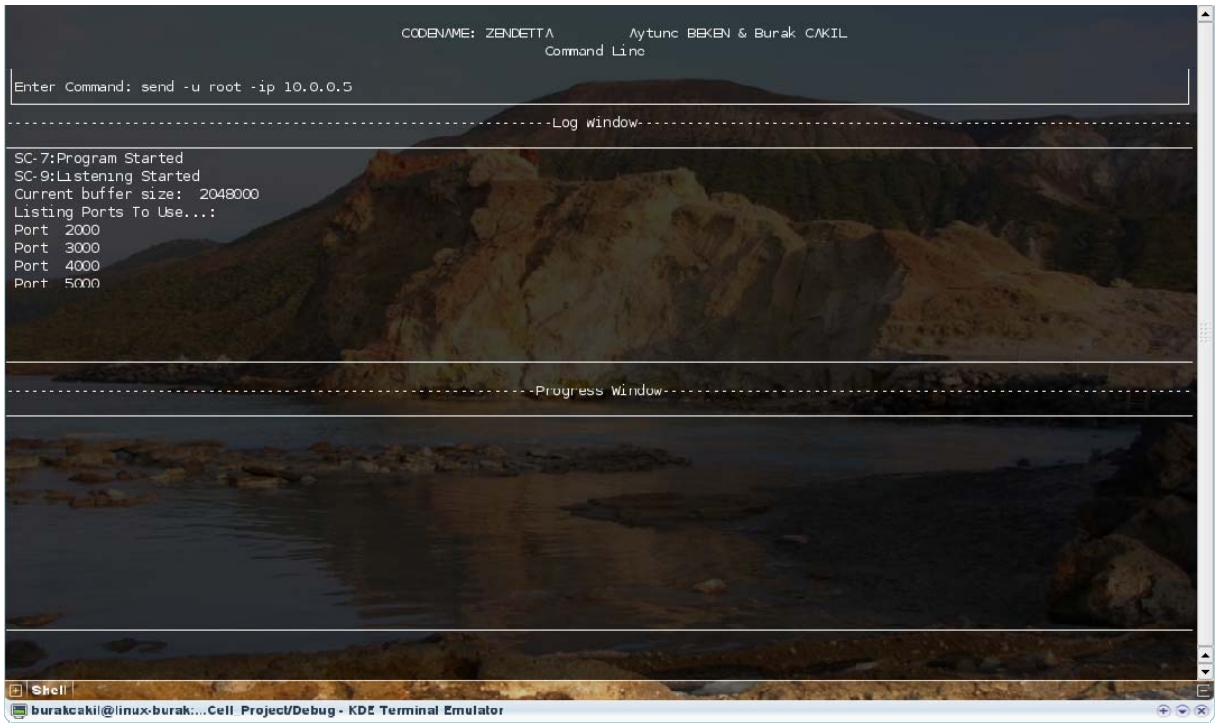
Appendix – 1: Screenshot– Program Start



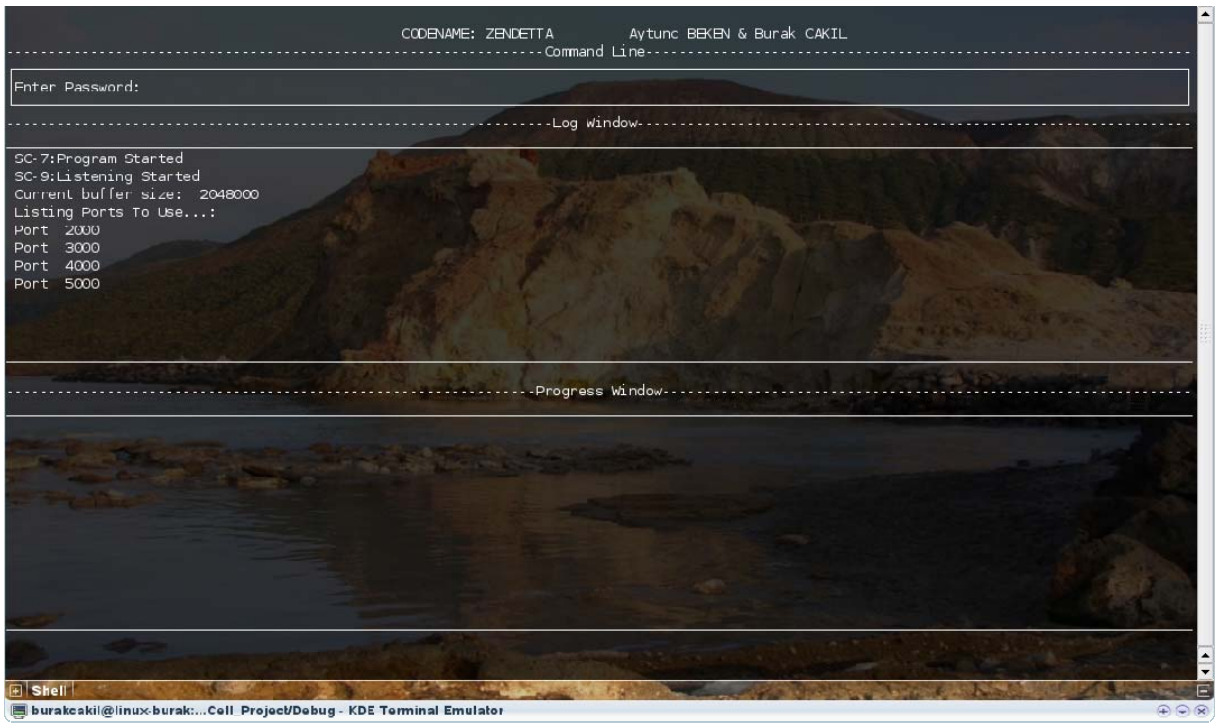
Appendix - 2: Screenshot – Listing buffer size and ports to be used



Appendix–3: Screenshot–Entering the command to initialize transmission



Appendix – 4: Screenshot – Asking for password before transmission



Appendix – 5: Screenshot – After the transmission finished

REFERENCES

[1] J. A. Kahle, M. N. Day, H. P. Hofstee, C. R. Johns, T. R. Maeurer, and D. Shippy, " Introduction to the Cell multiprocessor" , IBM Journal of Research and Development, September 2005.

[2] Monteyne, M., "Rapidmind Multi-Core Development Platform", White Paper, Rapidmind Inc., February 2008.

[3] Stokes, J. "Introduction to Multithreading, Superthreading and Hyperthreading", Ars Technica Journal, October 2002.

[4] Ncurses Library, http://www.gnu.org/software/ncurses.

[5] Mollin Richard A., RSA and Public-Key Cryptography, Chapman & Hall/CRC, 1st ed., November 2002.

[6] Postel, J. , Reynolds, J., "File Transfer Protocol (FTP)",1985.