# Operating Systems with their Reconfigurable Architecture

Deepak Kumar[1] and Dipanshu Attaria[2],
[1]*CSE, Dronacharya College of Engineering*
[2]*CSE, GB Pant College of Engineering*

***Abstract*—This paper is basically meant as an introduction to operating systems. Here we will discuss about the main functions of operating systems and all the types of operating systems which exists, as well as possible benefits gained by placing typical software functionality in hardware. In this article, the tasks of operating system are also described. Other than that we will discuss the types of operating systems, programming languages and also the programmable logic devices. This paper also includes some benefits with hardware support for operating systems. Operating systems in hardware are still in the development phase and one version of how an operating system in hardware could look like will be presented, including layers and message passing. And we are also introducing the hardware support in the form of reconfigurable architecture.**

***Index Terms*—Operating system, programmable logic device, FPGA, parallelism, hardware, software**

## I. INTRODUCTION

Demands on computer performance have increased rapidly over the years. Video processing and gaming are among the fields leading the way when pushing computers towards the breaking-point. Computer scientists have put a lot of focus on improving computer hardware; introducing multiple core central processing units (CPUs) and improved cache for example. In order to take the next step towards better performance it is not enough to improve hardware; software improvements are needed to fully take advantage of new technology. Since introducing reconfigurable architecture (RA), proven software-based algorithms known to be resource demanding can be moved down to hardware with enhanced performance. This migration should not only be limited to algorithms but also parts of an operating system (OS). Section 2 presents what an OS is, followed by section 3 that will introduce different types of OS. Section 4 will give a short summary of differences between a high-level programming language (HLPL) and a hardware description language (HDL). Types of programmable logic devices (PLDs) including a example is placed in section 5, followed by section 6 which mentions with few words about increased computer performance. Section 7 presents benefits that are achieved by introducing hardware support for OS. An example of how a hardware operating system (HOS) could look like is presented in section 8. Finally the paper is ended with acknowledgements in section 9, summary with conclusions in section 10 and at last future work in section 11.

## II. WHAT IS AN OPERATING SYSTEM?

A lot of people have come in contact with an OS; some without even knowing it. But even if you are aware of the fact that you are using an OS, it can be difficult to specify exactly what it is. An OS is said to take care of two somewhat unrelated tasks, managing resources and extending the machine [2.2]. These tasks are very important so let us look at them in more detail.

### *A. Managing Resources*

Nowadays computers consist of processors, memory, hard drives, printers etc. and we need a way to manage them. In a computer running multiple applications at the same time, we could easily end up with a mess if every application was granted access to a resource at any given time. This is where the OS gives a helping hand by only allowing an application to access a resource if it is not already in use. The OS usually solves problems like this by implementing queues, in the case of printers, or locking resources, in the case of optical drives. A perfect example of where at most one application can have access to a resource is when using a printer. Imagine having a paper printed with two completely different documents merged together into one. Or imagine burning two image files onto the same digital versatile disc (DVD). How resources in a computer are shared among users or applications can be divided into two groups; in time and in space. When a resource is shared in time, it means that there are several applications or users who want to use a resource at the same time. The only way to solve this problem is to let them take turns using the resource. The order in which this happens is determined by the OS; this is called scheduling. Shortest job first, priority based and round robin are just a few of the scheduling algorithms available. There are situations where applications or users can share the same resource at the same time. For example as soon as an application starts executing, it is given a part of the memory. This means you can have multiple applications in memory at the same time. As most applications today only use a fraction of the memory, it would be a waste of resources to keep only one in memory at a time. This kind of sharing is called sharing in space.

### B. Extending the Machine

The architecture of a computer can be very scary to look at. There are instruction sets, memory, registers, buses etc. For a programmer this can be too much to comprehend, but knowledge required if you are told to do some low level programming. This is where the OS comes in. The OS hides all the low level bits and pieces from the programmer (see Figure 1) and introduces a much simpler view. Access to low level functionality is obtained through system calls provided by the OS.

A Nehalem chip is divided into two broad domains, namely, the "core" and the "un-core". Components in the core domain operate with the same clock frequency as that of the actual computation core. In EOS's case this is 2.8GHz.The un-core domain operates under a different clock frequency. This modular organization reacts one of Nehalem's objectives of being able to consistently implement chips with different levels of computation abilities and power consumption profiles. For instance, a Nehalem chip may have from two to eight cores; one or more high-speed QPI interconnects deferent sizes for L3 caches, as well as, memory sub-systems with different DRAM bandwidths.

Similar partitioning of CMP chip into different clock domains can be found in other processors, such as, in IBM's Power5, 6 and 7, in AMDs multi-core chips and serves very similar purposes. Outside the Nehalem chip, but at close physical proximity, we and the DRAM which is accessible by means of three 8-byte DDR3 channels, each capable to operate at up to 1.333 Giga-transfers/sec. The aggregate nominal main memory bandwidth is 31.992 GB/s per chip or on the average 7.998 GB/s per core. This is a significant Improvement over all previous Intel micro-architectures. The maximum operating frequency of the DDR3 buses is determined by the number of DIMMs in the slots.

## III. TYPES OF OPERATING SYSTEMS

### A. Mainframe Operating Systems

It is not uncommon for a mainframe computer to take up an entire room by itself. The storage capacity of a mainframe computer surpasses personal computers by a wide margin. A mainframe OS focuses on processing as many tasks as possible at once. The above mentioned properties make a mainframe computer suitable for web server duties.

### B. Server Operating System

Server OS run on servers based on either a personal computer, workstation or mainframe. Users share software and hardware by connecting over a network. Services users are usually interested in are print services, file services and web services.

### C. Personal Computer Operating System

To increase performance of a computer, it is getting more and more common nowadays to use more than one processor. This hardware setup needs a special OS. Usually a modified version of the server OS is used with a particular set of features handling the extra complexity involved in communication.

### D. Real Time Operating System

If you have ever come in contact with an OS it is most likely this type. This OS only has to take one user into consideration. They are normally installed on computers located at home, at work, at school etc. You typically use it for writing documents, chatting, checking your e-mail and browsing the Internet. A well-known personal computer OS is Microsoft Windows XP.

### E. Embedded Operating System

In a real-time operating system (RTOS) time is a key factor. Computing data correctly is not enough; it also has to be delivered in time. There are two types of real-time systems. The difference between them is primarily whether or not they accept deadlines to occasionally be missed or not. The airbag example described earlier is a good example of a hard real-time system. A single miss can be the difference between life and death.

### F. Smart Card Operating System

A personal digital assistant (PDA) uses this type of OS. Embedded OS are generally small and the devices using them only provide a limited set of functionality; after all, this is not a general purpose OS. They are designed not to be resource demanding with respect to memory usage and power consumption. It is not uncommon to find an embedded OS in kitchen appliances like microwave ovens.

## IV. PROGRAMMING LANGUAGES

When developing software today, a HLPL is usually used to implement functionality. C, C++ and Java are just a small collection of HLPLs. Hardware used to be static and hard to reconfigure, but with the introduction of programmable hardware a new area of programming languages has derived, HDLs, such as VHDL and Verilog.

A clear difference between HLPLs and HDLs is that HLPL instructions are executed serially, one after another, in a CPU. HDL instructions can instead be executed in parallel thanks to the hardware (see Figure 2).

**Example:**

HLPL(C/C++):

```
X = I + 1;
Y = J – 5;
```

Z = K + 2;

This example will be executed serially as binary code in a CPU.

HDL(VHDL):
X <= I + 1;
Y <= J – 5;
Z <= K + 2;

Since these instructions are translated into gates, instead of binary code, it is possible to execute them in parallel

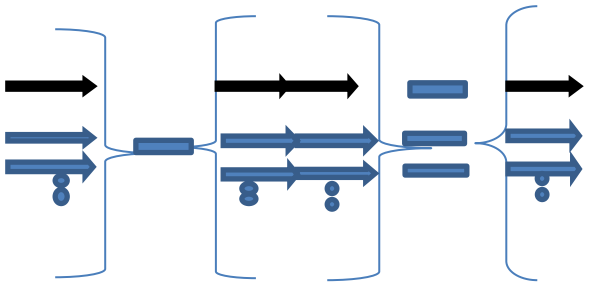

FIG: 2. INSTRUCTIONS PROCESSED IN SOFTWARE (LEFT) AND HARDWARE (RIGHT)

## V. PRGRAMMABLE LOGIC DEVICES

Today many types of programmable logic devices (PLDs) are available which can be used in different fields. In [2] is a list with names of devices that hardware and software can be downloaded to:

• Programmable Logic Device (PLD)
• Simple Programmable Logic Device (SPLD)
• Complex Programmable Logic Device (CPLD)
• Field Programmable Gate Array (FPGA)
• Field Programmable Inter Connect (FPIC)

These PLDs can be configured by downloading new software and hardware before run-time and use the configuration when the device is started. A computing paradigm that tries to combine the flexibility from software with the performance of hardware is called Reconfigurable Computing (RC).

## VI. BENEFITS WITH HARDWARE SUPPORT FOR OPERATING SYSTEMS

In section 4 an example of why functionality in hardware is faster than software was presented. This was only a tiny and easy function but lead to a great improvement in performance. Would it be possible to enlarge this simple example to a more complex one and even move parts from the OS down to hardware? The answer for the question is yes, so the following sections will present benefits that are the result of moving parts from the OS down to hardware.

### *A) Speed:*

One major benefit with hardware is it is much faster than software. Better performance means that operations take less time to execute. Create task, suspend task etc. are a few examples of operations that can be improved

### *B) Real Parallelism:*

In a CPU, instructions are executed one after another in serial as presented before. In a system with several processes it looks like the CPU is actually able to execute several processes at the same time, but this is unfortunately not true. At any time, only one process is able to execute in the CPU.

### *C) Determinism:*

Computer science is far from physiology but the meaning of determinism is translatable into the digital world of ones and zeros. Hardware is by nature deterministic in regard to time and as a result makes it easier to predict.

### *D) Predictability:*

In an RTOS it is important to make correct predictions of worst case execution time (WCET) for a program, to make sure that all tasks meet their deadline when scheduled. These predictions are made easier if the system has deterministic characteristics (which hardware has), to make this a little clearer

. Example 1: CPU –

A CPU is running a small program; there are not so many other programs or interrupts. In small scale, a CPU is therefore predicable. When the CPU is scheduled with many programs, interrupts etc., it gets harder to predict.

### *E) New Functionality:*

This advantage is the result of the other benefits, Hardware offers a new way of implementing algorithms thanks to its speed. Some algorithms that used to take too long to execute in software can utilize the hardware to lower the execution time.

## VII. SUMMARY AND CONCLUSION

The future of OS and higher performance might be the integration of software and hardware. There are a number of benefits that hardware can offer to developers thanks to the introductions of PLDs and HDLs. Similarities between HLPLs and HDLs makes it even easier to translate software into hardware. Developers of real-time systems can use the advantage of determinism that is natural in hardware to make better predictions for WCET and the speed of hardware to perform complex mathematical calculations in less time, where it used to be impossible with the timing constraints. Parts of OS or algorithms can be places in hardware, depending on how much functionality is parallelizable, performance increase in different scales.

## VIII. ACKNOWLEDGEMENTS

We want to thank Lennart Lindh for his inspiration for the subject and explanation of benefits.

## REFRENCES

[1] Tanenbaum Andrew. Modern Operating System (second edition). Prentice Hall. 2001

[2] Lindh, Lennart & Klevin, Tommy. Programmerbara kretsar : Utveckling av inbyggda system. Studentliteratur. 2005

[3] Wikipedia. Reconfigurable computing. <http://en.wikipedia.org/w/index.php?title=Reconfigurable_computing&oldid=318577595> (accessed October 14, 2009).

[4] Nordström, Susanna. Configurable Hardware Support for Single Processor Real-Time Systems. Mälardalen University Press Licentiate Theses. 2008