Role of Reconfigurable Devices in High Performance Computing System

Anant Mittal
UG Research Scholar,
Bharati Vidyapeeth's College of Engineering, New Delhi, INDIA

Sunil Kr. Singh
Associate Professor, CSE Deptt,
Bharati Vidyapeeth's College of Engineering, New Delhi, INDIA

Asmita Goyal
UG Research Scholar,
Bharati Vidyapeeth's College of Engineering, New Delhi, INDIA

ABSTRACT
In modern high-performance embedded system technology, there is a demand of applications with high performance but minimum resource utilization. Due to advancement in technology, developer integrates multiple functionalities into a single chip. Reconfigurable architectures can adapt the behavior of the hardware resources to a specific computation that needs to be performed and also increases its utilization. ASIC are application specific architectures with high performance and limited resource requirement but lacks in flexibility. Reconfigurable computing devices fulfill both needs i.e. flexibility and performance. Advancements in the field of reconfigurable computing hardware and software provide greater design flexibility and reduced cost. In this paper, we emphasize on the role of reconfigurable devices in high performance computing.

Keywords
Reconfigurable Computing, High Performance Computing, Algorithm, FPGA.

1. INTRODUCTION
The process of designing the digital hardware has been changed extraordinarily by the development of very sophisticated field programmable devices (FPDs). Earlier, board level designs included number of SSI (small-scale integration) chips containing various basic gates. But these days every design which is made today consists of mostly high-density devices. Not only devices like processors and memory but also logic circuits such as state machine controllers, counters, registers, and decoders are in need for high-volume systems or in other words, high-density systems. For this, designers need to integrate them into high-density gate arrays. But because of high manufacturing and engineering cost, and long time to produce it, it made them inappropriate to use locally. Due to these reasons, this leads the designers to fabricate field programmable devices (FPDs). The advantages of which are low manufacturing and start up cost, as the cost is low, it has very less financial risk. Most important advantage of FPDs is that the end user programs the device, by which easy design changes can be done and FPDs can be effectively used in various fields such as Reconfigurable Computing. The goal of the new architectures is to reduce the time and to speed up the computations [7].

2. RELATED RESEARCH WORK
Research related area in the field of Reconfigurable Computing is rapidly advancing for scientific and high performance multimedia applications. While today's Field programmable Gate Array (FPGA) technology is more promising for implementing reconfigurable computational systems, their capabilities in certain areas (such as floating point arithmetic) are far better than other technologies. For this reason, efficient system architectures must encompass a heterogeneous mixture of the best technologies. The target systems are built on a heterogeneous computing platform, including configurable hardware, ASIC and general purpose processors and FPGA based upon partial reconfigurable SRAM. The primary difficulty in this approach lies in system design. A designer must now maintain a set of different system architectures, which exist at different times in the system's lifetime, and map these architectures onto the same group of resources. The designers must manage the behavior of the system, determining the operational modes of the system, the rules for transitioning between operational modes, and the functional properties within each operational mode. In addition, the system must make efficient use of the resources, enabling the designer to minimize the envelope of hardware required to support the union of all operational modes. Currently system design tools are insufficient to manage this complexity [3][8].

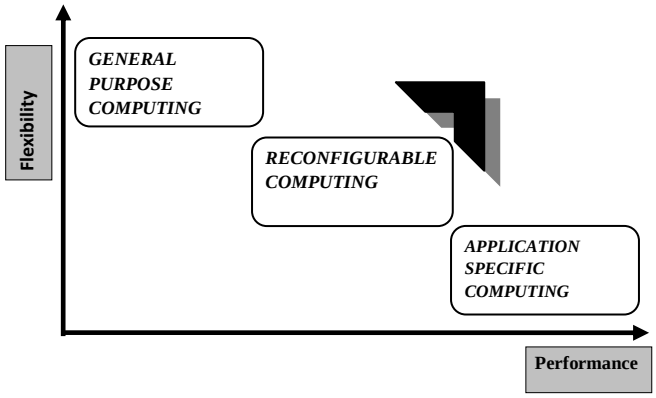
3. HIGH PERFORMANCE HARDWARE SYSTEM
Today a mix of dedicated hardware solutions and programmable devices is found in applications for which no other approach can meet the real-time performance. Many of the systems are characterized by high throughput requirements in the front-end with very regular processing and lower throughputs in the back-end, but with a high degree of data dependency and, therefore, requiring more general-purpose programming. Many applications can be met with programmable signal processors. In these instances, the signal processor is typically large in size with plenty of power, or, conversely, the algorithm complexity is low, permitting its implementation in a single or a few microprocessors. Programmable signal processors, provide a high degree of flexibility since the algorithm techniques are implemented using high-order languages such as C. However, the implementation must be rigorous with a high degree of care to ascertain real-time performance and reliability. Reconfigurable computing, for example, utilizing field programmable gate arrays (FPGAs) achieves higher computing performance in a fixed volume and power when compared to programmable computing systems. This performance improvement comes at the expense of only having flexibility in the implementation if the algorithm techniques can be easily mapped to a fixed set of gates, table look-ups, and Boolean operations. The most demanding applications require most of the computing be implemented in custom hardware to meet capabilities for cases in which trillions of operations per second per unit volume (TOPS/ft3)

and 100s GOPS/W are needed. Today such computing performance demands custom designs and dedicated hardware implemented using application-specific integrated circuits (ASICs)[2][12].

4. RECONFIGURABLE COMPUTING

The two main means on which any processors are characterized are: flexibility and performance. GPPs (General Purpose Processors) are very flexible because they are able to compute any kind of task but they cannot compute in parallel so they are not performance oriented. Also they are not very efficient if the same instruction has to be executed on huge sets of data repetitively. GPPs are flexible because they always adapt to the hardware in order to be executed.

Application Specific Integrated Processors (ASIPs) bring much performance because they are optimized and manufactured specifically for a particular application. The instruction set required for that application can then be built in a chip. Performance is possible here because the hardware is always adapted to the application. So on the basis of performance and flexibility, GPPs and ASIPs can placed as



shown in figure 1.

Figure1. Performance and Flexibility graph

Between the GPPs and the ASIPs are a large numbers of processors depending on their performance and their flexibility. Reconfigurable computing as shown in fig. represents an intermediate approach between the ASIPs and GPPs. It combines a reconfigurable-hardware unit with a software programmable processor. For a given application, at a given time, the spatial structure of the device will be modified such as to use the best computing approach to speed up that application. The structures of reconfigurable devices are changed by modifying all or part of the hardware at compile-time or at run-time, usually by downloading a bit stream into the device. The first chip that could implement logic circuits was PROM in which inputs were address lines and outputs were data lines. PROM is used very rarely because it contained a full decoder for its address inputs, and logic functions rarely require than a few product terms. The first effective device that was fabricated was PLA which consists of two levels of logic gates, a programmable, wired-AND plane followed by a programmable, wired-OR plane as shown in figure 2. PLAs are well suited for implementing SOP forms. The disadvantages of PLA were that they were expensive and had poor speed performance because programmable logic planes were difficult to manufacture and had significant propagation delays. To overcome this, PAL devices were developed that had a programmable, wired-AND plane that fed fixed-OR gates as shown in figure 3. To overcome the lack of generality, PALs come in variants with different number of inputs and outputs and various sizes of OR gates [1][6].

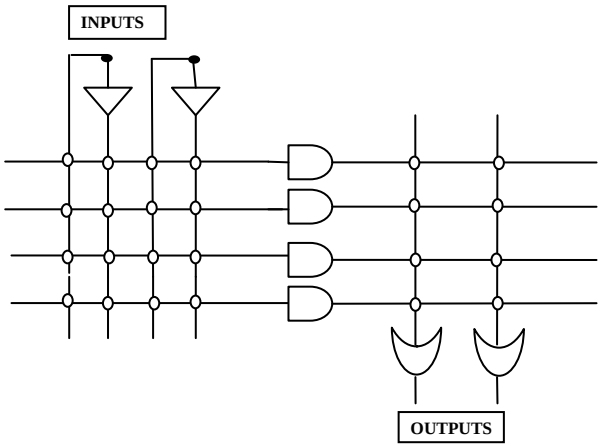


Figure2. PLA-Programmable Logic Array

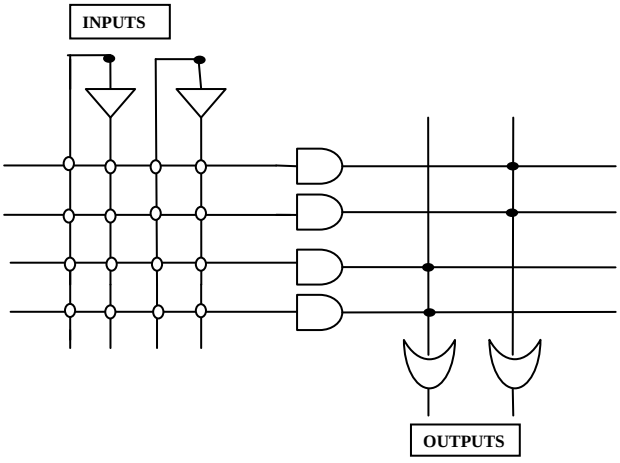


Figure3. PAL-Programmable Array Logic

5. RECONFIGURABLE COMPUTING DEVICES

Reconfigurable devices have both the flexibility of the GPP and the performance of the ASIP. Depending upon the number of gates present in the device and the technology that can be used to program it, PLDs can be classified as shown in figure4. . Progress in reconfiguration has been tremendous in the last two decades due to the introduction of the Field Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs).

5.1 CPLDs and FPGAs

The Complex Programmable Logic Device (CPLD) and the Field Programmable Gate Array are the two new devices that are not only flexible but also have shorter turn-around time.

CPLDs and FPGAs bridge the gap between PALs and Gate Arrays

.

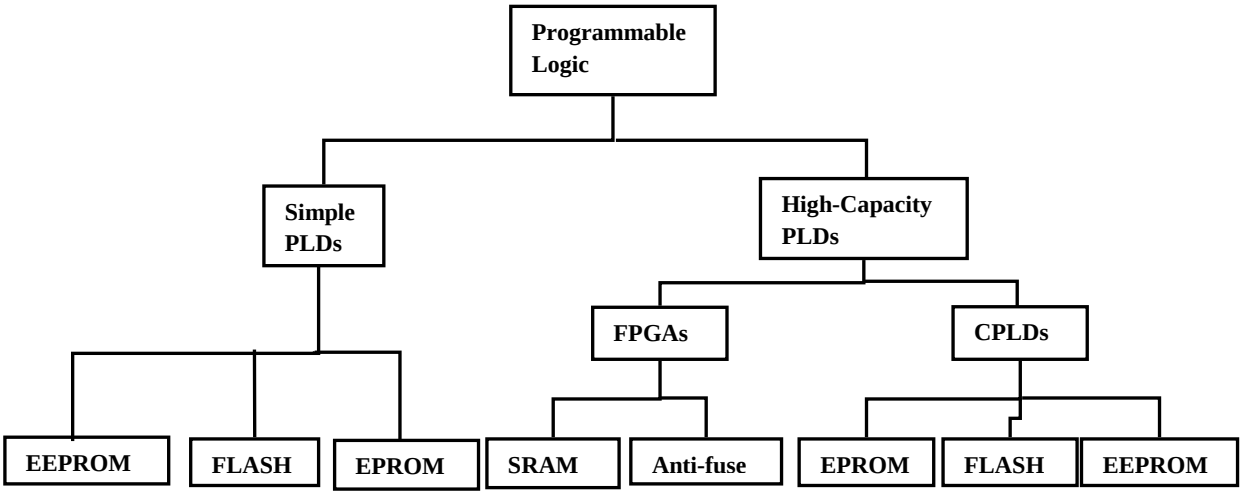


Figure 4. Types of PLDs

5.2 Complex Programmable Logic Devices (CPLDs)

CPLD is basically a single chip in which a number of PALs are embedded and interconnected through a cross point switch. Due to this technology and design it can handle very complex logic.

5.2.1 *CPLD Architecture*

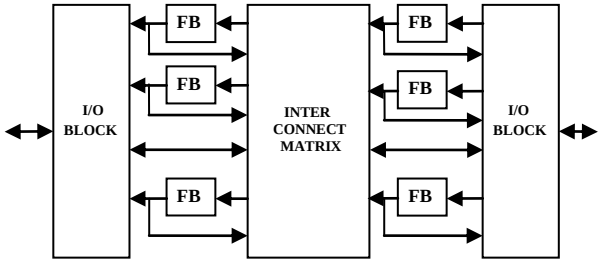


Figure5. CPLD Architecture

Figure 5 shows the internal architecture of a typical CPLD. It consist of function blocks, input/output block, and an interconnect matrix. A function block consists of an AND plane into which input is fed from the function block itself or the input/output block which drives signals to the CPLD device at appropriate voltage and current levels. The outputs received from the AND plane are then ORed together, and are selected via multiplexer. Clocked filpflop can be used to send out the output of the mux and to prevent the delay on a signal at both input/output ends. The CPLD interconnect is a large programmable switch matrix which connects signals from all parts of the device with each other. It provides a flexibility that allows us to implement many complex logical functions as various interconnections are possible. The devices are programmed using different technologies and various programmable elements like Electrically Programmable Read Only Memory (EPROM), Electrically Erasable PROM (EEPROM) and Flash EPROM. The examples of CPLD Families are Altera MAX 7000, MAX 9000 [10] and Xilinx XC9500 families [9] etc.

5.2.2 *CPLD Architecture Issues*

The main issues that should be taken into consideration are how many function blocks are required, range of delays, number of logic gates (AND and OR) needed, how many I/O are reserved for signals like clock, set, reset etc , the additional logic that can be implemented to increase the flexibility, the interconnect capability i.e. the number of combinations of connections can be made, the elements required to program the devices based on the programming technology and the reprogramming capability of the device.

5.3 Field Programmable Gate Arrays (FPGAs)

Field Programmable Gate Arrays are very useful in prototyping ASICs as they are designed very much like ASIC.

5.3.1 *FPGA Architecture*

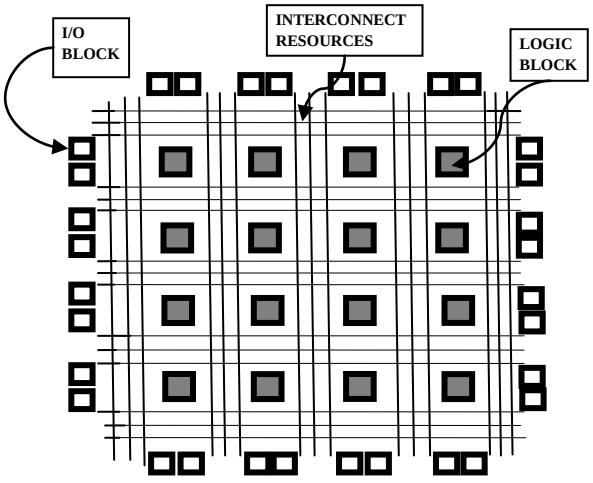


Figure6: FPGA architecture

As shown in figure 6, the architecture consists of configurable logic blocks (CLB) which contains RAM for designing logic functions, flip-flops which act as storage elements, and multiplexers to select the logic. It also consists of configurable I/O blocks which drives signals to the device. The I/O are said to be configurable because the polarity of the output can be programmed. The programmable interconnect has the long lines that connect CLBs with each other through programmable switches. Special long lines, called global clock lines, are designed for fast propagation of signals that are connected to each clocked element in each CLB. There is a clock circuitry that drives clock signals to each logic block and the global clock line. Additional logic resources such as ALUs, memory, and decoders are also present.

Based on the complexity of the logic that can be implemented by CLB, FPGAs are of two types, small grain for simple function and large grain for complex functions. The two basic types of programmable elements for an FPGA are SRAM which uses static RAM bits and anti-fuse in which a certain amount of current during programming of the device causes the two sides of the anti-fuse to connect. SRAM is easily fabricated and is reprogrammable but it is volatile and has large delay whereas anti-fuse is non volatile and has less delay but its fabrication is complex and is non reprogrammable. Example of FPGA families, are Altera FLEX, Xilinx XC4000 and Virtex families based on SRAM and Actel SX and MX etc based on anti fuse [6][9][10].

5.3.2 *FPGA Architecture Issues*

The main issues that should be taken into consideration are that there should be synchronization between the speed of the microprocessor on FPGA and the modern bus, the arrangement of input data in on-board and on-chip memories has a large impact on performance, implementation techniques should be used in order to have a better trade off between performance and space[13].

6. LANGUAGES FOR PROGRAMMING RECONFIGURABLE COMPUTING DEVICES

A hardware description language describes hardware. Designers write code for programming hardware. Synthesis tools translate that code into bit streams that can be downloaded to the reconfigurable hardware. The languages that are used to program FPGAs differ basically in the level of abstraction. Recent trends are to use high-level languages such as C and Handel-C for programming of hardware design and more sophisticated synthesis tools to translate the specifications to hardware. The challenge here is to get efficient hardware designs. Another approach is to use libraries of pre designed components that have already been optimized for a particular family of devices. These components are usually parameterized so that different versions can be used in a wide range of designs. The most common method for specifying an FPGA design is to use an HDL. There are two dominant choices in this field, VHDL and Verilog[5].

7. ALGORITHMIC REQUIREMENTS FOR HIGH PERFORMANCE COMPUTATION

A programmer has to design and rewrite code to make it reconfigurable. So to make sure that the reconfigurable code has a higher performance various algorithmic considerations, timing analyses, and benchmarking experiments are to be followed and taken into regard.

7.1 Algorithmic Requirements

In FPGAs, various requirements of the algorithm are taken to be in consideration like data parallelism which basically means that the data can be reused until all the data can be processed which in turn increases the computational performance. Data parallelism can be implemented without increasing bandwidth and delay by the technique of pipelining which allows overlapping of computations at different stages of same task. Also to enhance the performance concurrency can be done i.e. several operations can be performed simultaneously using different computational nodes[11].

7.2 Input Data Requirements

While programming the reconfigurable codes, certain delays in the computation due to transfer of data from the memory of the CPU to the local memory of the FPGA should be taken care of as it overrides the improvement in the computational performance of the FPGA. Also, the size of the bit, integers or fixed point numbers increases the area of the chip. It also affects the computational speed and efficiency of the circuit. Data must be transmitted at high rates to keep the hardware busy and thus resulting in high performance. For optimal FPGA acceleration, scheduling of data transfers and computation should be done effectively, in order to achieve substantial levels of concurrency and pipelining.

7.3 Optimal Design for Reconfigurable Algorithm

In FPGA, implementation of an algorithm is an excellent choice. For example, consider an application that organizes data elements into specific groups according to their value. In C programming, it is recommended to implement binary search. Whereas in FPGA, algorithm is based on brute force method is considerably more optimum. A computational unit can be programmed inside the FPGA if it is executed inside a loop. A loop has a great impact on the performance of the algorithm as it directly affects the FPGA acceleration but there should be a tradeoff between the number of loops and performance of the code.

There are certain considerations that should be taken care of in the loop structure such as pointers like linked lists should be avoided as they can lead to inconsistency with parallel access. Trigonometric, logarithmic, and transcendental operations should not be used because these operations require a considerable area of the chip [4][13].

8. CONCLUSION

We have reviewed the factors which determine the role of reconfigurable devices in high performance computing. The two main types of reconfigurable devices, field programmable gate arrays (FPGA) and complex programmable logic devices (CPLD), are both extensively used. Each device contributes particular strength in computation. The ability to select the suitable HDL, system design procedure and to test designs, are the major challenges. But a appropriate algorithm, data organization and movement can help to minimize the computational delay in reconfigurable device computing. We expect that above investigated role help advancements of high performance computing with reconfigurable devices to design reconfigurable embedded system and other advanced system.

9. REFERENCES

[1] Christopher Bobda, "Introduction to Reconfigurable Computing", Springer, 2007.

[2] David R. Martinez, Robert A. Bond, M. Michael Vai, "High Performance Embedded Computing Handbook: A Systems Perspective", CRC Press, 2008.

[3] K. Bondalapati and V. Prasanna. "Reconfigurable Computing systems in Proc. IEEE, vol.90, no.7, July 2002, pp.1201-1217.

[4] Marco Lanzagorta, Stephen Bique, Robert Rosenberg, "Introduction to Reconfigurable Supercomputing", Morgan and Claypool, 2010.

[5] Prof. Sunil Kr. Singh , Dr. M. P. S. Bhatia , Dr. Rajni Jindal, "Architectural Modeling for Hardware and Software in Reconfigurable Embedded System" in Int. J. of Recent Trends in Engineering and Technology, Vol. 1, No. 1, Nov 2009

[6] Scott Hauck and Andre DeHon, "Reconfigurable Computing: The Theory and Practice of FPGA based Computation", Morgan Kaufmann, 2008.

[7] Stephen Brown, Jonathan Rose, "FPGA and CPLD architecture: A Tutorial", IEEE Design and Test of Computers, Summer 1996.

[8] Villasenor, J., Mangione—Smith, W., "Configurable Computing", Scientific American, June, 1997.

[9] Xilinx, Inc., http://www.xilinx.com/

[10] Altera Corporation, http://www.altera.com/

[11] Phan C. Vinh, Jonathan P. Bowen, "A Provable Algorithm for Reconfiguration in Embedded Reconfigurable Computing", Proceedings of the 2005 29th Annual IEEE/NASA Software Engineering Workshop (SEW'05)

[12] Dally, W.J. and S. Lacy. 1999." VLSI architecture: past, present, and future", Proceedings of the 20th Anniversary Conference on Advanced Research in VLSI.

[13] Scott Hauck, "The Roles of FPGAs in Reprogrammable Systems", to appear in Proceedings of the IEEE.