Reconfigurable Computing Systems Used To Support Next Generation High Speed Applications

M. Munawwar Iqbal Ch, Muhammad Younus Javed and M. Aqeel Iqbal
Department of Computer Engineering
College of Electrical and Mechanical Engineering
National University of Sciences and Technology (NUST), Islamabad, Pakistan

***Abstract* – Emerging dimensions of scientific computing have changed the structural requirements of the under laying hardware and software resources. The growing scientific applications are demanding for the high speed computing platforms having the capability of the run time architectural updation. The conventional computing approaches using application specific integrated circuits and programmable general purpose processors are still suffering with the lack of architectural modifications for demanded updation and computing speed respectively, during the life of running applications. The reconfigurable computing is becoming the next generation computing approach as it promises the high level integration of hardware speed and software flexibility at one platform. The reconfigurable computing is intended to fill the gape between the high speed computing platforms like application specific integrated circuits and the highly flexible computing platform like programmable general purpose processors. This research paper briefly investigates the roll of reconfigurable computing systems in the emerging scientific applications.**

Keywords - Reconfigurable Processors, Reconfigurable Computing, Reconfigurable Logic, FPGAs, RISPs

I. CONVENTIONAL COMPUTING APPROACHES

In conventional computing paradigm there are two distinct approaches being used for the execution of the algorithms. One conventional approach being used is based on computing using *Application Specific Integrated Circuits (ASICs)* while the second conventional approach being used is based on computing using *General Purpose Processors (GPPs)* as shown in Fig. 1.

A. Computing Using ASICs

The primary conventional computing approach is to use hardwired circuits such as Application Specific Integrated *Circuits (ASICs)* to execute the operations in hardware. A kind of customized hardware is specially designed to support and execute an application. The highly specialization of hardwired circuit will result in fast and efficient execution for the designed task. Also this kind of the dedication approach demonstrates the minimal power consumption overhead as the designers avoid unnecessary contemporary logic modules. However, huge amount of development work is required to achieve extreme efficiency. Also, the hardwired circuit cannot be altered after fabrication. Any changes, modifications, or updates to the circuit require a redesign and re-fabrication of the chip. This is an expensive process in effort, time, and cost for maintaining hardwired technology as shown in Fig. 2.

B. Computing Using GPPs

The secondary conventional computing approach for the execution of the algorithms is fully based on the use of software dependant programmable processors. The concerned algorithm is represented in the form of sequential program codes. The processor executes this program code which is in fact in the form of instructions, to perform the required computation. *Instruction Set architecture (ISA)* interfaces between the program instructions for the algorithm and the executing hardware. Changes in the either end will not affect the functionality of the other side as long as the instruction set architecture specifications were followed. Therefore the changes in software instructions can modify the functionality of an operation without the need to change any part or module of the hardware resources. This gives great flexibility to designers to freely modify the software code. This great flexibility is in tradeoff by large performance overhead. During the process of program or application execution the processor fetches each instruction from memory, decodes its meaning and then performs the operation of the instruction. These overheads result in degraded performance and greater power consumption as a whole execution process as shown in Fig. 2.

While instruction set architecture serves as an interface between the software and the hardware, it also limits the potential growth of the under laying system. After the chip fabrication, any operations to be implemented must be built based on the instruction set architecture specifications. Improvements in the hardware must maintain the full ISA specification, even obsolete ones, to be backward compatible with existing software programs. The software based solution of any algorithm demonstrates a temporal execution flow but the ASIC based solution demonstrates a spatial flow.

II. RECONFIGURABLE COMPUTING

The *reconfigurable computing (RC)* is an emerging engineering paradigm that bridges up this gap of algorithmic flexibility and speed, and hence allowing a processor to be optimized to the task at hand as shown in Fig. 1. So the designer is not forced to achieve a trades-off between the performance gain on a narrow area and the versatility [1]. Reconfigurable computing provides the technology to implement a processing unit that can be dynamically optimized for the algorithm it is implementing. *Field Programmable Gate Arrays (FPGAs)* have an architectural tendency to provide such a kind of innovative computational concept where it can ideally be reused and re-optimized for any number of times for multiple algorithms simply by reprogramming it [1].

The FPGA consists of reconfigurable logic that can be reprogrammed to compute applications ranging from very fine-grained, highly repetitive to coarse-grained, general purpose computational tasks. SRAM-based FPGAs can be used to create processors and co-processors whose internal architectures as well as interconnections can be reconfigured to match the needs of a given applications. Computing circuits built from FPGAs can obtain the true goal of parallel processing. They execute algorithms in-circuit with the inherent parallelism of hardware, while avoiding the instruction fetch and load/store bottlenecks of traditional von Neumann architectures as shown in Fig. 2.

The recent idea of reconfigurable computing applications and system designs has been a subject of research since few decades, but most of the existing projects have demonstrated the potential of connecting one or more commercially available FPGAs to an on shelf existing processor via a standard external bus such as the IO/PCI bus [2]. If reconfigurable computing is really intended to become the computing paradigm of the future applications, then the main parts must be brought closer together. Only a few studies have considered integrating a processor and FPGA core into a single device as shown in the Fig. 3. With the two tailored to cooperate closely with each other; and so there remain important questions about how such a device might be built and programmed, and how it would fit within an existing general-purpose computing framework [2].

Recently it has been observed that the reconfigurable computing is really becoming an important part of the research in the field of computer architectures and software based systems. By placing computationally intensive parts of an algorithm onto the reconfigurable logic, the performance gain of the overall application can be drastically accelerated. This is in fact mainly due to the reason that because the reconfigurable computing combines benefits of both software and ASIC implementations. Like software, the mapped circuit is flexible, and can be changed over the lifetime or life cycle of the system. Similar to an ASIC, reconfigurable systems provide a method to map circuits into hardware, achieving far greater performance than software as a result of bypassing the fetch-decode-execute cycle of traditional microprocessors, and parallel execution of multiple operations. Even due to all of these technological advancements, the reconfigurable computing is still at the depth of microcode stage and hence requiring programmers of the applications to have a considerably deep knowledge about low level circuit design in order to create an application. A large number of the C-based languages exist that map to *Hardware Description Languages (HDLs)*; however those languages still need the programmer to think on scale of low hardware circuits [3]. As seen in software, major productivity gains will not occur unless the level of reasoning about problems changes.

A program implemented on a circuit that takes advantage of the parallelism offered by hardware is much more difficult to create than a sequence of instructions to run on a processor. In order to make gains in productivity, the abstraction level at which engineers and scientists create reconfigurable computing programs further needs to be elevated. Along the way, some optimizations may be given up but that will allow for the development of larger and more complex systems that could not be feasibly created otherwise. Several attempts are being made at doing this through developing higher level languages for programming hardware in a C-like manner. Many of these approaches present a small step forward in raising the level of abstraction but still it requires a significant knowledge of reconfigurable computing programming to create systems.

III. RISPS AND RECONFIGURABLE COMPUTING

Today the embedded systems are composed of many hardware and software based components which are interacting with each other during the execution of the application programs. In fact the balance between these used components will determine the success of the concerned system.

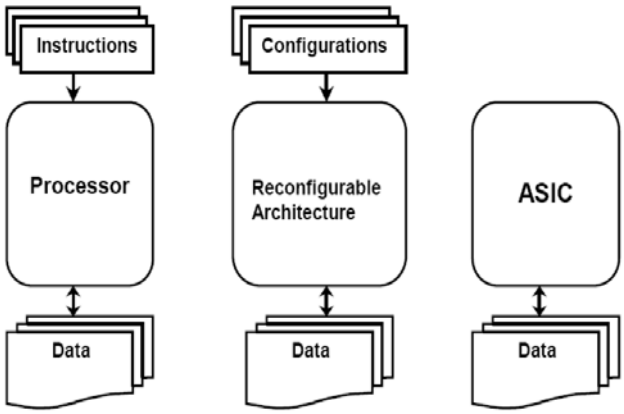

Fig. 1 Computing Systems

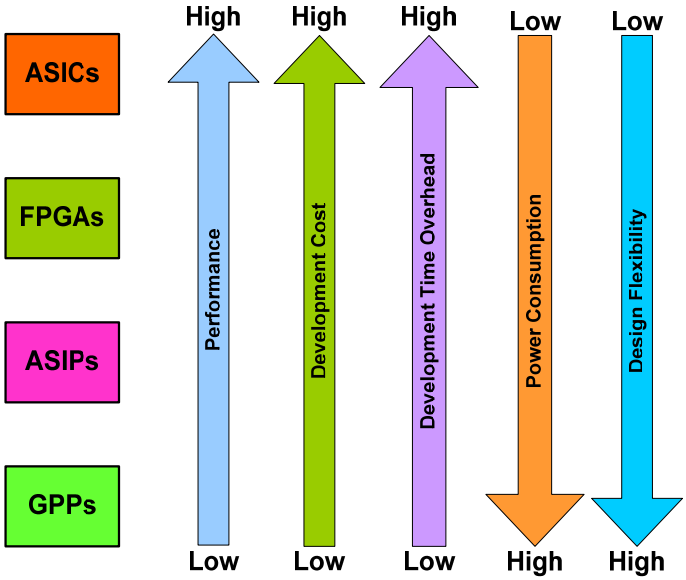

Fig. 2 Characteristics of Computing Systems

Due to the surprising nature of software technology, the software components are extremely easier to modify or update as compared to the hardware based components. Due to this dramatic flexibility nature, the software components being running on programmable general purpose processors provide an easy way to eliminate the bugs, to change the application, to reuse components, to differentiate a product or to reduce the more important time to market. However, when compared to pure hardware based solutions, the software components are slower and consume more power. Hardware components are used when speed and power consumption are very critical in the prescribed task. Unfortunately, hardware components require a lengthy and very expensive design process. Typical hardware components cannot be modified after they have been fabricated. The task of the system designer is to find an adequate balance between these components which interact very closely.

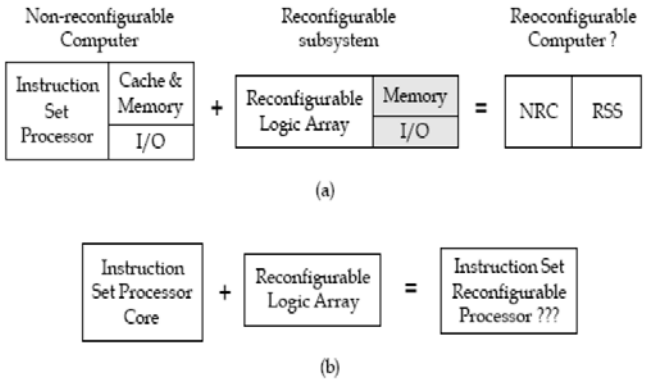

Fig. 3 Evolution of Reconfigurable Systems

The interaction between software and hardware components of a system is very tight and closely coupled, especially in the case of embedded systems, where cost efficient solutions are a must. *Application Specific Instruction Set Processors (ASIPs)* and *Reconfigurable Instruction Set Processors (RISPs)* are most prominent examples of the interaction of the hardware and software components since they contain very specialized hardware and software components interacting very closely in the form of a single integrated chip [4] as shown in Fig. 3. Conventional programmable general purpose processors do not have this type of interaction due to the generic nature of the processors. As the processor becomes more and more specialized, so does the interaction between hardware and software. Designing a system with such a type of processors requires a methodology that encompasses both software and hardware aspects.

A. Computing Using ASIPs.

An ASIP is a kind of the hybrid architecture between a programmable processor and an integrated custom logic. Non critical parts of the running application are implemented using the standard instruction set or hardware core while the critical parts are implemented using a specialized instruction set, typically using one or more specialized functional units that have been customized for the running application [4]. Thus these functional units implement a specialized instruction set architecture. This specialization in fact permits reduced code size, reduced power consumption and higher processing power. The Implementation of an application on an ASIP system involves the design of the customized functional units and the mapping of the software algorithms to these customized units. Both must be done concurrently in order to obtain a good solution, but due to the nature of hardware and software development this is not always possible.

B. Computing Using RISPs.

A RISP consists of an integrated processor core that has been extended with reconfigurable logic as shown in Fig. 3. It is quite similar to an ASIP but instead of having Specialized Functional Units (SFUs), it contains the Reconfigurable Functional Units (RFUs). The reconfigurable functional units in fact provide this adaptation of the processor to the application, while the standard processor core provides software programmability. As a concept; the RISPs execute instructions just as standard processors and ASIPs, though the main difference is that the instruction set of RISPs is divided into two main sets. The first set includes the fixed instruction set, which is implemented in fixed hardware and the second set is reconfigurable instruction set, which is implemented in the reconfigurable logic and can be changed during runtime of the active application [4]. This reconfigurable instruction set is equivalent to the specialized instruction set of the ASIP but with the ability to be modified after the processor has been manufactured.

The cost of designing a processor core is drastically reduced by reusing it in many different applications by running a large number of different software programs. In ASIPs, the cost for every new kind of generation comes from the redesign of their

specialized functional units, which can be quite high as compared to programmable processors. The rapidly emerging prototyping techniques are essential for reducing the cost and time benchmarks required for the redesign. A RISP can be used as a best prototyping device for a family of ASIP processors that share the same basic core having fixed instruction set. In this sense, it can be viewed as a *Field Programmable Gate Array (FPGA)* specialized for ASIP design. In fact the RISPs provide a better prototyping platform than a pure FPGA, for a set of ASIP cores due to the specialized elements which it contains. The RISP already contains data and program memory; one or more register files, control logic and other processor specific elements. These elements are optimized for a processor implementation, which is not the case for FPGA technology. The RISP programming tools can also help in the design of new ASIPs. The required new specialized functional units must be designed for an ASIP. The RISP programming tools should do this automatically. These tools can also be used to instantiate an ASIP, which can be viewed as the non reconfigurable version of a RISP.

The RISPs not only provide the benefits in field of prototyping but in fact, the first models of a product can use a RISP instead of a more expensive ASIP. As the product matures, transition to an ASIP can be done in a convenient manner without taking any design risk. This allows the product to evolve technically without a significant risk and also allows a smaller time to market and hence reduces development cost rapidly [4]. We can observe a quit similar kind of trend in the ASIC and FPGA markets. Also in many cases it is now more cost effective to use FPGAs instead of ASICs. Furthermore, in some scientific user applications, more preferably the RISPs would be the processor of choice. Evolving scientific standards, unknown demanded applications and a broad diversity of algorithms are cases where a fixed solution will eventually fail to deliver the required high performance and computational power. Evolving scientific standards and unknown applications make it very difficult to create specialized hardware for them. Applications with a broad diversity of algorithms require much specialized hardware which may be more expensive than a reconfigurable solution. The RISPs offer this kind of computation platform flexibility that ASIPs lack.

Many features of a typical RISP depend on the availability of good programming tools optimized in order to obtain good results. More importantly, it is necessary to design the processor and these programming tools simultaneously and check regularly that one does not impose any severe restrictions on the other. The basic element of a RISP code generation tool or suite is a high level language compiler that performs the automatic hardware/software partitioning. In some cases, such a type of complex compiler might not be needed or might not be technically feasible. A normal compiler can also be used and then the code is modified to include reconfigurability. The compilation process is quite complex and introduces many new concepts to compiler designers. The initial compilation phases are typical of high level language compilers. The source code is analyzed and an internal representation is obtained, normally in the form of some sort of control and data flow graph. The graph is composed of basic building blocks, hyper building blocks, regions or any other constructs that facilitates compilation. The size of the blocks depends on the technical design of the processor architecture. High level optimizing transformations are also done during these compiling stages of the code.

The Reconfigurable Processors (RPs) are the new types of processor which combines a microprocessor core with reconfigurable logic. The objective of this design approach is to add the benefits from microprocessors and reconfigurable logic both at the same time. The Fig.4 shows the brief comparison of a microprocessor (μP) and a reconfigurable processor (RP) in the form of a table showing different Synergism characteristics of both of them. The reconfigurable logic will provide hardware specialization to the application being under execution [4]. It will also provide quite similar benefits to those offered by the application specific instruction set processors. ASIPs have been integrated with the specialized hardware that accelerates the execution of the applications it was designed for. A reconfigurable processor would have this same benefit but without having to commit the hardware solution into silicon. Reconfigurable processors can be adapted after design, in the same way as that of the programmable processors can adapt to application changes. Reconfigurable instructions set processors are in fact a subset of reconfigurable processors.

	μP	RP(FPGA-based)
Processing Style	Software→Control Flow (von Neumann) Temporal – reuse of fixed hardware	Hardware→Data Flow Spatial – Unfolding parallel operations with changeable hardware
Parallelism Exploited	Coarse-Grain	Fine-Grain
Clocking Rate	Very Fast Saturating Rate	Relatively Slow Increasing Speed
Applications Partitioning	Relatively Easy (S.W./Parallel Programming)	Harder
Commercial Availability	COTS, multipurpose	COTS, multipurpose

Fig. 4 Synergism between μP and RPs

Many systems have been designed with very different characteristics and the research on this type of processors has been quite active since the last few years. The essential enabler of this has been the fast growth of FPGA capacity. There have been many reconfigurable processors, especially before the year 1994. Most of these processors are in fact of the attached processor type. The design of a reconfigurable processor can technically be divided into two main tasks. The first one is the interfacing between the microprocessor and the reconfigurable logic. This includes all the issues related to how data is transferred to and from the reconfigurable logic, as well as synchronization between the two elements. The second task is the design of the reconfigurable logic itself.

Granularity, reconfigurability and interconnections are issues included in this task. The Fig. 5 shows the working mechanism of a typical RISP. This flow chart is showing the different steps being involved during the execution of an application program on a typical RISP.

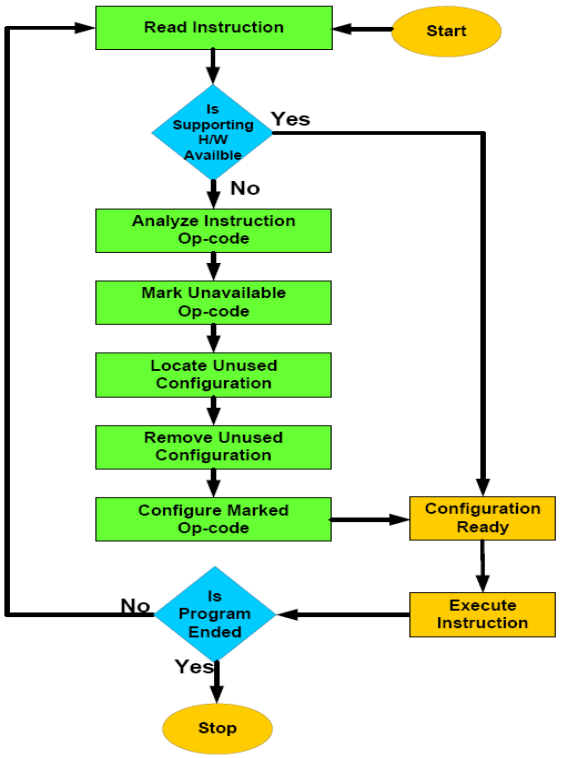

Fig. 5 Working Flow of a Typical RISP

IV. RECONFIGURABLE COMPUTING DEVICES

As a concept the reconfigurable computing has been in existence since the last few decades. Even general-purpose processors use some sort of the same basic ideas, such as reusing computational components for independent computations and using multiplexers to control the routing structures between these components. However, the term *reconfigurable computing*, as it is used in current research, refers to systems incorporating some form of hardware programmability by customizing the hardware being used using a large number of physical control points. These control points can then be changed or updated periodically in order to execute a different class of applications using the same hardware. Reconfigurable computing devices have emerged as most flexible and high performance computing devices in computing systems. We have been observing various examples of computing systems that are fully reconfigurable at the fine grain logic level as well as such devices that are reconfigurable at coarse grain architectural level. In addition we have been seeing the increased use of reconfigurable cores as fundamental computing components in emerging embedded systems micro-processors coupled with a reconfigurable component as well as the application specific integrated circuits coupled with a reconfigurable component. Reconfigurable devices initially emerged as fast prototyping devices for application specific integrated circuits designers. It allowed the designers to compile the application to reconfigurable hardware to determine that the application exhibited the correct functionality. The prototyping removed the costly step of fabricating the application, especially when fabrication yielded a device it exhibited incorrect functionality. Additionally, the rapid prototyping concept has demonstrated the need for intensive simulation to verify correctness.

The power of reconfigurable computing devices lays the immense amount of its inner computational flexibility that it provides. In fact the computational flexibility allows run-time reconfiguration or modification and on-the-fly reorganization of the circuit based on the input parameters. Due to the ability to customize to the input data, many user and scientific applications show drastic speed-ups when implementing them on reconfigurable computing systems. The familiar 90-10 rule asserts that 90% of execution time is consumed by about 10% of a program's code and that 10% generally being inner loops of computational algorithm. Reconfigurable devices excel in those cases where computation represented by a configuration is repeated many times so that the time required to load a configuration can be amortized over a long execution time and/or overlapped with other execution. When all of an application's important loop bodies can be configured to it within the reconfigurable logic or reconfigurable computing machine (one at a time), there would seem to be no need for the overhead of a fully dynamic instruction fetch and issue mechanism and hence allowing the machine to be more leaner and efficient.

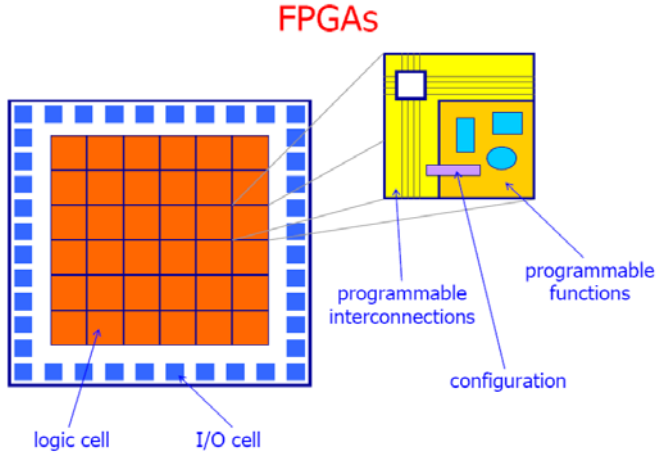

Fig. 6 Basic FPGA Block Diagram

The most recent advances in reconfigurable computing domain are for the most part derived from the technologies developed for FPGAs in the mid-1980s. FPGAs were

originally developed to serve as a hybrid device between *Programmable Array Logics (PALs)* and *Mask Programmable Gate Arrays (MPGAs)*. Like PALs, FPGAs are fully electrically programmable or customizable, meaning that the physical design costs are amortized over multiple application circuit implementations on the same chip and the hardware can be customized nearly instantaneously [5]. Like MPGAs, they can implement very complex computations and algorithms on a single chip, with devices currently in production containing equivalent of millions of logic gates [5]. Because of these features, FPGAs had been primarily viewed as a glue logic replacement and low cost rapid-prototyping vehicles. However the flexibility, capacity and performance of these devices have opened up the completely new avenues in high-performance computing and hence forming the basis of reconfigurable computing [5]. Consider the Fig. 6 and Fig. 7 which are showing the internal architecture of a basic FPGA.

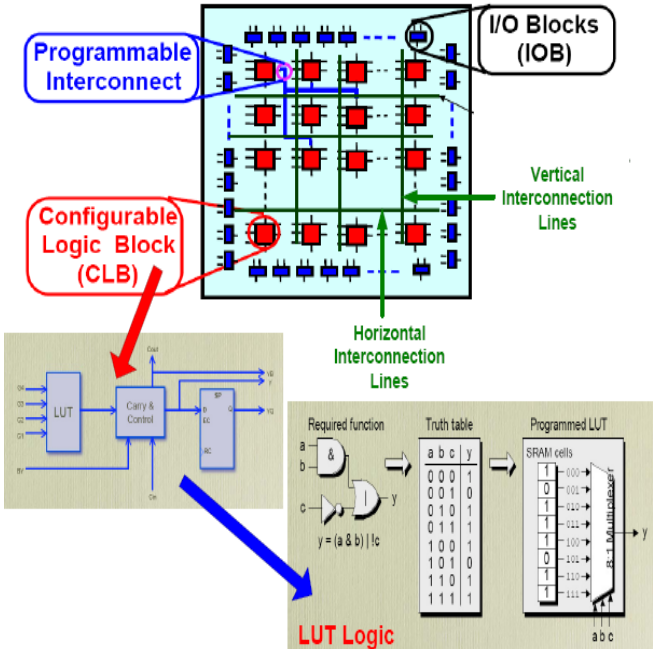

Fig. 7 FPGA Internal Design Concept

A Field Programmable Gate Array (FPGA) is a silicon chip containing an array of configurable logic blocks (CLBs) and a network of programmable routing resources that are spread throughout the device as shown in Fig. 8. Mainly FPGAs have been provided by the companies like Xilinx [6] and Altera Corporations [7]. Unlike an Application Specific Integrated Circuit (ASIC) which can perform a single specific function for the lifetime of the chip; a field programmable gate array can be reprogrammed to perform a different function in a matter of microseconds. FPGAs have been involved in two major fields of computing including one as a prototyping platform and other as a low cost, small volume alternative to ASICs.

The configurable logic blocks are the functional units and are individually quite small. Between the configurable logic blocks there is a reconfigurable routing network, connecting the different configurable logic blocks with one another as shown in Fig. 8. Configurable logic blocks consists of one or two four-way look-up tables (LUTs) taking four bits inputs and generating one bit output as shown in Fig. 9. By filling the look-up tables, configuring the multiplexers and setting all other control bits the behavior of the configurable logic block is defined. There are long and short wires connected to each other in switch matrixes and to the configurable logic blocks through switch nodes. Consider the Fig. 10 which shows the architecture of I/O block of basic FPGA.

V. System Level Coupling

Programmable logic tends to be inefficient at implementing certain types of operations, such as loops and branch control. In order to run an application most efficiently in a reconfigurable computing system, the areas of the active program that cannot be easily mapped to the reconfigurable logic are executed on a host micro-processor. Meanwhile, the areas with a high density of computation that can benefit from implementation in hardware are mapped to the reconfigurable logic [8]. For the systems that use a micro-processor in conjunction with reconfigurable logic, there are several ways in which these two structures can be coupled [8]. Four main options that have been used for the coupling of reconfigurable logic within the computing system have the following characteristics.

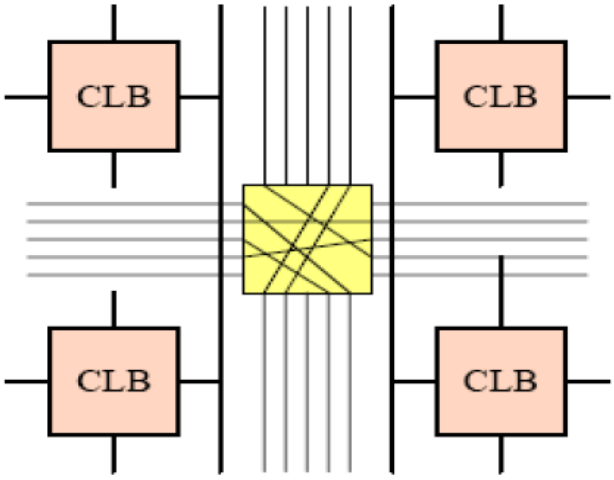

Fig. 8 CLB Interconnections

A. Tightly Coupled Unit Approach.

In first coupling approach the reconfigurable logic hardware can be used solely to provide reconfigurable functional units within a host processor. This allows for a traditional programming environment with the addition of custom instructions that may change over time. Here the

reconfigurable units execute as functional units on the main microprocessor data path, with registers used to hold the input and output operands. This type of coupling approach is used in reconfigurable instruction set processors (RISPs) also known as reconfigurable processors.

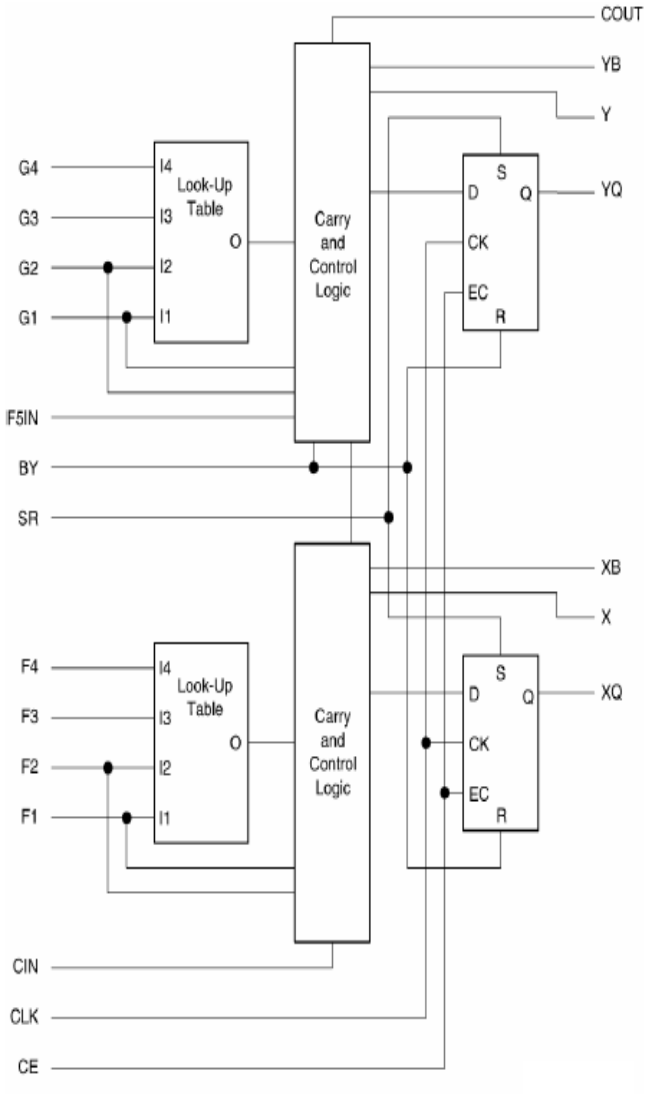

Fig. 9 Basic CLB Internal Architecture

B. *Co-processor Unit Approach.*

In second coupling approach a reconfigurable unit may be used as a coprocessor. A coprocessor is in general larger than a functional unit, and is able to perform computations without the constant supervision of the host processor. Instead, the processor initializes the reconfigurable hardware and either sends the necessary data to the logic or provides information on where this data might be found in memory.

C. *Attached Processor Unit Approach.*

In third coupling approach the attached reconfigurable processing unit behaves as if it is an additional processor in a multiprocessor system. The host processor's data cache is not visible to the attached reconfigurable processing unit. There is, therefore a higher delay in communication between the host processor and the reconfigurable hardware such as when communicating configuration information, input data and results. However, this type of reconfigurable hardware allows for a great deal of computation independence by shifting large chunks of a computation over to the reconfigurable hardware.

D. *Loosely Coupled Unit Approach.*

In fourth coupling approach the most loosely coupled form of reconfigurable hardware is that of an external standalone processing unit. This type of reconfigurable hardware communicates infrequently with a host processor. This model is similar to that of networked workstations, where processing may occur for very long periods of time without a great deal of communication.

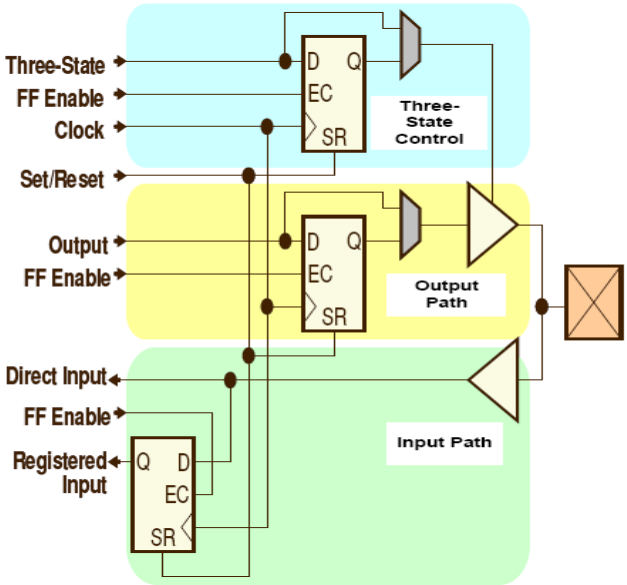

Fig. 10 FPGA I/O Structure

VI. CONCLUSION

The emerging scientific computations are requiring the extended data bandwidth with the dynamic behavior for their absolute execution flows. This is all due to the growing demands of the computing applications. The use of application specific integrated circuits and general purpose processors for

the solution of such type of applications is suffering with the lack of flexibility and speed respectively. The scientists were looking for a new type of computing platform having the tendency of merging the flexibility like general purpose processors and speed like application specific integrated circuits. Reconfigurable computing is now playing a vital role to enhance the computational powers of the existing systems. Field programmable gate arrays have been selected as a device for providing the reconfigurable computing behavior of the systems. The performance of a reconfigurable computing system is greatly dependant on the type of the coupling approaches being used for the integration of the reconfigurable logic with the standard processing unit. Still reconfigurable computing is suffering with many potential drawbacks that can be addressed for future high speed computations.

REFERENCES

[1]. K. Compton and S. Hauck, "Reconfigurable computing: a survey of systems and software," ACM Computing Surveys, vol. 34, no. 2, pp. 171–210, 2002.
[2]. R. Hartenstein, "Trends in reconfigurable logic and reconfigurable computing," in Proceedings of the 9th IEEE International Conference on Electronics, Circuits, and Systems (ICECS '02), pp. 801–808, Dubrovnik, Croatia, September 2002.
[3]. Bondalapati, K., and Prasanna, V.K.: 'Reconfigurable computing systems', Proc. IEEE, 2002, 90, (7), pp. 1201–1217
[4]. Francisco Barat, R. Lauwereins, G. Deconinck, "Reconfigurable Instruction Set Processors from a Hardware/Software Perspective"; IEEE Transactions on Software Engineering, Vol. 28, No.9, pp. 847–862, Sept., 2002.
[5]. P. Garcia, K. Compton, M. Schulte, E. Blem, and W. Fu. An overview of reconfigurable hardware in embedded systems. EURASIP Journal on Embedded Systems, 2006: pp. 1–19, 2006.
[6]. Xilinx Inc., UG071,"Virtex-4 Configuration Guide", v. 1.9, October, 2007.
[7]. Altera Corporation Products. 2008; http://www.altera.com/products/prd-index.html
[8]. Compton K. and Hauck S. An introduction to reconfigurable computing. IEEE Computer, April 2000.

AUTHORS PROFILES

M. Munawwar Iqbal Ch

M. Munawwar Iqbal Ch Received His M.Sc. Degree In Computer Science In 1996 From Quaid-I-Azam University Islamabad, Pakistan. Then He Earned His Ms Degree In Computer Software Engineering In 2005 From National University Of Science And Technology, Nust, Islamabad, Pakistan. Currently He Is Pursuing His Phd Research At Department Of Computer Engineering, College Of E & ME, NUST, Under The Supervision Of Dr Muhammad Younus Javed.

Dr. Muhammad Younus Javed

Dr Muhammad Younus Javed did his PhD (Adaptive Communication Interfaces) from University of Dundee, United Kingdom in 1991 and MS (Disambiguation Systems) from the same university in 1988. He completed BE Electrical Engineering from UET Lahore in 1982. He is serving in the College of Electrical and Mechanical Engineering since 1991 and has taught a number of courses at undergraduate and postgraduate level. He is currently Associate Dean (Academics & Evaluation) at College of E & ME, National University of Sciences & Technology (NUST), Rawalpindi. Five PhD scholars are working under his supervision. He is recipient of ORS Award from the University of Dundee for his outstanding PhD research work. His areas of interest are Digital Image Processing, Operating Systems, Design and Theory of Algorithmics. He has 190 national/ international publications to his credit.

M. Aqeel Iqbal

M. Aqeel Iqbal Is An Assistant Professor In The Department Of Software Engineering, Faculty Of Engineering And Information Technology, Foundation University, Institute Of Engineering And Management Sciences, Rawalpindi, Pakistan. As A Researcher He Has A Deep Affiliation With The College of E & ME, National University Of Sciences And Technology (NUST), Rawalpindi, Pakistan