SEMINAR REPORT

RECONFIGURABLE PROCESSOR ARCHITECTURE

Submitted By

DEEPAK J PUTHUKKADEN

Roll No. 20

EC7 B

ACKNOWLEDGEMENT

I wish to thank **Dr. V.P.Devassia**, Principal, Govt. Model Engineering College, for giving me the opportunity to conduct a detailed study and analysis of my seminar. I wish to thank **Mr. Jayachandran. E.S**, Associate Professor, Head of the Department of Electronics Engineering, for his support and faith in me for the completion of this seminar. I express my profound gratitude to **Ms. Shiji T.P**, Associate Professor, the coordinator of Electronics & Communications for allowing me to proceed with this seminar. I take this opportunity to express my profound gratitude and deep regards to my guide **Mr. Rashid M.E**, Senior Lecturer, Department of Electronics Engineering, for his exemplary guidance, monitoring and constant encouragement throughout the course of this seminar. I would also like to thank all the teachers of the Department of Electronics for their whole-hearted support.

LIST OF ABBREVIATIONS

RP	Reconfigurable Processor
FPGA	Field Programmable Gate array
ASIC	Application Specific Integrated circuit
CPU	Central Processing Unit
ROM	Read Only Memory
PLA	Programmable Logic Array
PAL	Programmable Array Logic
RAM	Random Access Memory
uP	Microprocessor
TI	Texas Instruments
GE	General Electric Company

Index

ABSTRACT

Reconfigurable processors combine the speed of application specific integrated circuits (ASIC) and the universality of classical digital processors by means of adaptability to the currently executed code. There is a great number of varieties in today's reconfigurable processor architectures. Processor architectures can be specified in many ways. Architecture specifications are usually used during the architecture design phase and in software tools such as compilers and simulators. The aim of this seminar is to propose various comparison criteria for reconfigurable processor architectures and to give an overview of the specification methods for them.

Reconfigurable processors (RP) combine the speed of application specific integrated circuits (ASIC) and the universality of classical digital processors. ASIC circuits are the fastest way of processing digital data because they are pre-tailored for known functions. Classical processors spend a lot of time in fetching and decoding relatively small set of basic instructions. The main difference between classical processors and reconfigurable processors is that RPs try to adjust their internal structure to the currently executed code. Classical processors have fixed, unchangeable structure while all RPs have some changeable, adjustable components. These changeable components are called reconfigurable fabric (RF).

Chapter 1

INTRODUCTION

The present day requirements of electronic circuits or processor are speed and flexibility. Speed of a circuit can be increased by designing the circuit for the task it has to perform. By building custom or dedicated circuits for the operation to be performed, the performance of an electronic circuit can be increased drastically. Flexibility increases the ease and the reusability of a circuit. More the flexible, more easily can a circuit be used for different purposes. For imparting flexibility to a circuit, it should contain more general purpose components. These components can do more number of tasks using the same circuit. But this can cause a reduction in the performance of the circuit. This is because the general purpose components in the circuit doesn't deliver the performance shown by their dedicated counterparts.

The above problem can be solved by a circuitry that combines the flexibility of general purpose circuits as well as the performance of dedicated circuits. This is made possible by Reconfigurable Processors. Reconfigurable Processors (RP) or Reconfigurable Computing. Reconfigurable processor is a computer architecture combining some of the flexibility of software with the high performance of hardware by processing with very flexible high speed computing fabric. RPs have a reconfigurable fabric that can be programmed to be made any custom or dedicated circuit. This will enable the RP to deliver the same performance delivered by dedicated circuits. Once the required task is done, the fabric can then be re-programmed again to another circuit and thus is highly flexible.

Modern day RPs have found applications in areas where flexibility and speed are primary concern. These include signal processing, image processing, data encryption and decryption, avionic circuits, military applications, computer aided design etc.

Chapter 2

HISTORY

In 1960, Gerald Estrin, the well-known American computer scientist, proposed the concept of a computer make of a standard processor and an array of reconfigurable hardware. In his model the main processor would control the behaviour of the reconfigurable hardware while the latter would be tailored to perform a specific task, such as image processing or pattern matching as quickly as a dedicated piece of hardware. Once the task was done, the hardware could be adjusted to do some other task. This resulted in a hybrid computer structure combining the flexibility of software with the speed of hardware.

2.1 ROM as PLD

The first devices that were used to create arbitrary combinational logics where ROMs. Consider a ROM with m inputs (the address lines) and n outputs (the data lines). When used as a memory, the ROM contains 2m words of n bits each. The advantage of using a ROM is that any conceivable function of all possible combinations of the m inputs can be made to appear at any of the n outputs, making this the most general-purpose combinational logic device available for m input pins and n output pins.

When EPROMs and EEPROMs where available this introduced the concept of programmability and re-programmability. But ROMs were still not dedicated hardware and it had the following de-merits:

- They are usually much slower than dedicated logic circuits
- They consume more power
- They are often more expensive than programmable logic, especially if high speed is required
- The cannot be used stand alone for sequential circuits as most of the ROMs lack input or output registers

2.2 Early Programmable Logic

In 1969, Motorola offered the XC157, a mask-programmed gate array with 12 gates and 30 uncommitted input/output pins. In 1970, Texas Instruments (TI) developed a mask-programmable IC based on the IBM read-only associative memory. This device, the TMS2000, was programmed by altering the metal layer during the production of the IC. TI coined the term programmable logic array (PLA) for this device.

In 1971, General Electric Company (GE) was developing a programmable logic device based on the new UV erasable PROM technology. The GE device was the first erasable PLD ever developed. In 1973 National Semiconductor introduced a mask-programmable PLA device - DM7575. This was more popular than the TI part but cost of making the metal mask limited its use. The device is significant because it was the basis for the Field Programmable Logic Array (FPLA) produced by Signetics in 1975, the 82S100.

In 1974 GE entered into an agreement with Monolithic Memories to develop a mask-programmable logic device incorporating the GE innovations. The device was named the 'Programmable Associative Logic Array' or PALA. The MMI 5760 was completed in 1976 and could implement multilevel or sequential circuits of over 100 gates. The device was supported by a GE design environment where Boolean equations would be converted to mask patterns for configuring the device. Unfortunately, due to reasons, the part was never brought to market.

MMI introduced a breakthrough device in 1978, the programmable array logic (PAL). The architecture was simpler than that of Signetics FPLA, because it omitted the programmable OR array. This made the parts faster, smaller and cheaper. The PALASM design software (PAL assembler) converted the engineers' Boolean equations into the fuse pattern required to program the part. The PAL devices were soon second-sourced by National Semiconductor, Texas Instruments and AMD.

An innovation of the PAL was the Generic Array Logic device (GAL), invented by Lattice Semiconductor in 1985. This device has the same logical properties as the PAL but can be erased and reprogrammed. The GAL is very useful in the prototyping stage of a design, when any bugs in the logic can be corrected by reprogramming. GALs are programmed and

reprogrammed using a PAL programmer, or by using the in-circuit programming technique on supporting chips.

PALs and GALs are available only in small sizes, equivalent to a few hundred logic gates. For bigger logic circuits, complex PLDs or CPLDs can be used. These contain the equivalent of several PALs linked by programmable interconnections, all in one integrated circuit. CPLDs can replace thousands, or even hundreds of thousands, of logic gates. Some CPLDs are programmed using a PAL programmer, but this method becomes inconvenient for devices with hundreds of pins. A second method of programming is to solder the device to its printed circuit board, then feed it with a serial data stream from a personal computer. Some manufacturers use JTAG to program CPLD's in-circuit.

2.3 Field Programmable Gate Array

While PALs were busy developing into GALs and CPLDs, a separate stream of development was happening. This type of device is based on gate array technology and is called the Field Programmable Gate Array (FPGA). Early examples of FPGAs are the 82S100 array, and 82S105 sequencer, by Signetics, introduced in the late 1970s. The 82S100 was an array of AND terms. The 82S105 also had flip flop functions.

FPGAs use a grid of logic gates, and once stored, the data doesn't change, similar to that of an ordinary gate array. The term "field-programmable" means the device is programmed by the customer, not the manufacturer.

FPGAs are usually programmed after being soldered down to the circuit board, in a manner similar to that of larger CPLDs. In most of the larger FPGAs, the configuration is volatile and must be re-loaded into the device whenever power is applied or different functionality is required. Configuration is typically stored in a configuration PROM or EEPROM. EEPROM versions may be in-system programmable (typically via JTAG).

The difference between FPGAs and CPLDs is that FPGAs are internally based on Look-up tables (LUTs) whereas CPLDs form the logic functions with sea-of-gates (e.g. sum of products). CPLDs are meant for simpler designs while FPGAs are meant for more complex

designs. In general, CPLDs are a good choice for wide combinational logic applications, whereas FPGAs are more suitable for large state machines (i.e. microprocessors).

Altera was founded in 1983 and delivered the industry's first reprogrammable logic device in 1984 – the EP300 - which featured a quartz window in the package that allowed users to shine an ultra-violet lamp on the die to erase the EPROM cells that held the device configuration. Xilinx co-founders Ross Freeman and Bernard Vonderschmitt invented the first commercially viable field-programmable gate array in 1985 – the XC2064. The XC2064 had programmable gates and programmable interconnects between gates, the beginnings of a new technology and market. The XC2064 had 64 configurable logic blocks (CLBs), with two three-input lookup tables (LUTs).

The 1990s were an explosive period of time for FPGAs, both in sophistication and the volume of production. In the early 1990s, FPGAs were primarily used in telecommunications and networking. By the end of the decade, FPGAs found their way into consumer, automotive, and industrial applications.

2.4 Present day developments

In 2002, Cypress began shipping commercial quantities of their RP called Programmable System-on-Chip (PSoC). A PSoC integrated circuit is composed of a core, configurable analogue and digital blocks, and programmable routing and interconnect.

In 2002, Chameleon Systems stated shipping their Chameleon chips. These RPs could be reconfigured at very high speeds of 20us. Thus they can be used switched between different applications, swiftly and hence without considerable performance reduction. This was a major advantage over traditional FPGAs, as reconfiguring the whole FPGA was a time consuming process, which usually involved resetting the whole FPGA and loading the new circuit configuration into the FPGA.

In the 2000s, both Altera and Xilinx stated shipping FPGAs that supported partial and dynamic reconfiguration on the go. This enabled the FPGA to change its circuitry fast without introducing much delay or hassle and hence made different circuit implementations on the same chip to work very efficiently and fast.

In 2010, Xilinx introduced the first All Programmable System on a Chip branded Zynq-7000 that fused features of an ARM high-end microcontroller with a 28 nm FPGA fabric to make it easier for embedded designers to use. The extensible processing platform enables system architects and embedded software developers to apply a combination of serial and parallel processing system designs. The high level of integration helps to reduce power consumption and dissipation. An alternate approach to using hard-macro processors is to make use of soft processor cores that are implemented within the FPGA logic. Nios II, MicroBlaze and Mico32 are examples of popular softcore processors.

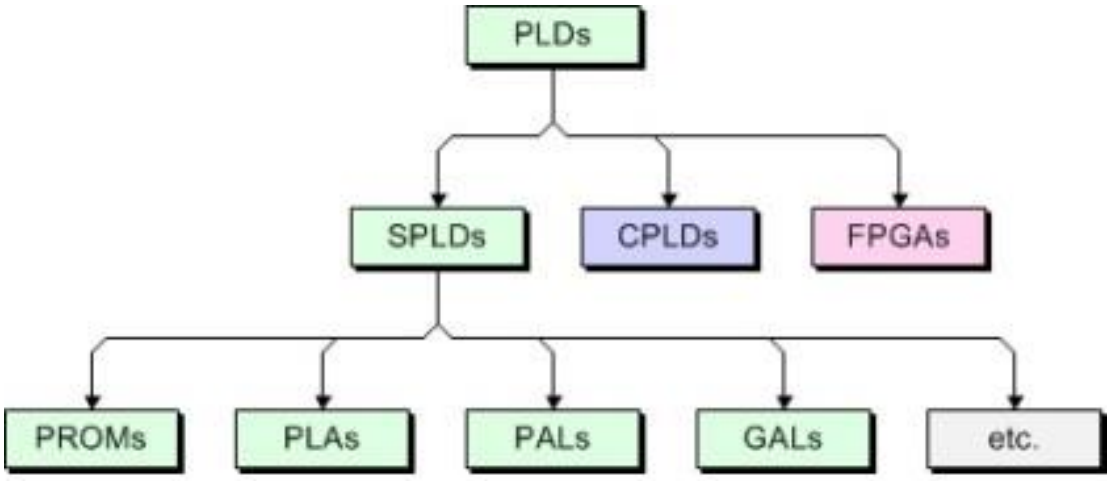


Fig 2.1: Classification of Programmable Devices

Chapter 3
ARCHITECTURE

Reconfigurable processors nowadays have incorporated many architectures. There are different types of RPs that are present in the market now. Examples are FPGA, chameleon chips, PSoC etc. Each variety of RP has the same principle of reconfigurable hardware that is implemented using different architectures.

There are basically 5 different architectural components that we will be covering in this chapter. They are:

3.1 Implementation spectrum
3.2 System level architecture
3.3 Granularity
3.4 Programmable logic elements
3.5 Reconfiguration models

These are the five major architectural concepts that vary with different RPs.

3.1 Implementation Spectrum

In the area of computer architecture, designers are faced with the trade-off between flexibility and performance. The architectural choices span a wide spectrum, with general-purpose processors (GPPs) and application-specific integrated circuits (ASICs) at opposite ends. General purpose processors performs a variety of function using the same instructions. Application specific integrated circuits on the other had perform only the function for which they are designed for. Fig. 1 presents a schematic overview of the speed/flexibility trade-off.

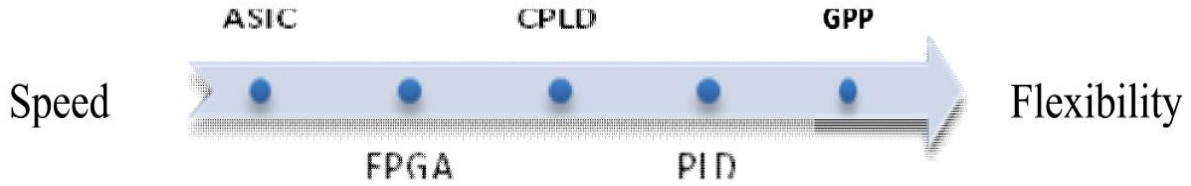


Fig 3.1: Speed/Flexibility comparison

The application specific integration circuit (ASIC) stands at the end of spectrum. It is designed for one specific task therefore there is no need for an instruction set. ASICs are dedicated hardware devices that are tuned to a very small number of applications or even to just one task. For a given task, ASICs achieve a higher performance, require less silicon area, and are less power consuming than instruction-level programmable processors. However, they lack in flexibility. Thus ASICs gives high performance at cost of inflexibility. The general-purpose processor (GPP) can be found at another end of spectrum. A GPP is very flexible, as it can execute any function. Each function can be assembled by various instructions supported by the GPP. However, fetching instructions from memory and decoding those costs (a lot of) time whereas GPP are very slow.

The best solution is to use programmable circuits such as PLD, CPLD and FPGAs. There are some reasons that encourage using programmable circuits. First, the implementations of complex digital functions are simple, second, test and analysis of circuits are very easy and fast. Third, inexpensive fabrications for few productions and the most important reason are the accordance with functional requirements and the ability to change the design over again by user.

3.1.1 PLD

Programmable logic devices (PLD) are integrated circuits with internal logic gates that are connected together through fuses. A process that is called programming defines the functionality of the chip. ROMs (Read Only Memories), PALs (Programmable array logic) and PLAs (Programmable Logic Arrays) are examples of PLDs. The main difference between these devices is the position of the fuses and the fixed connection between gates. Inside each PLD is

a set of fully connected macrocells. These macrocells are typically comprised of some amount of combinatorial logic (AND and OR gates, for example) and a flip-flop. A PLA consists of two levels of logic gates: a programmable " wired " AND - plane followed by a programmable "wired" OR - plane. A PLA is structured. So that any of its inputs (or their complements) can be AND'ed together in the AND - plane.

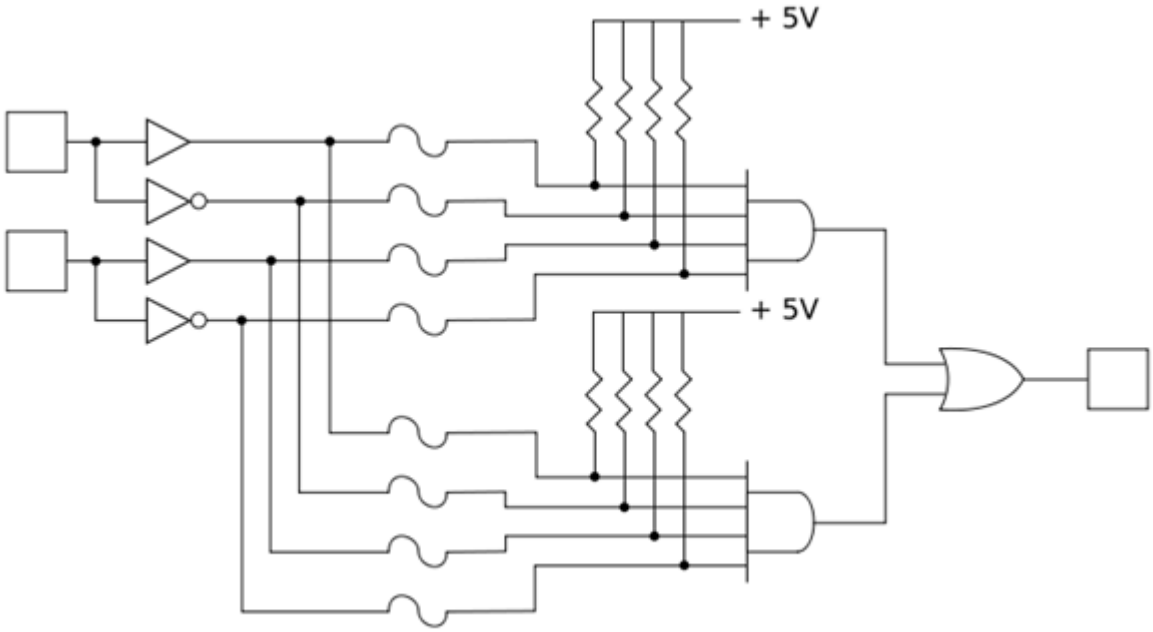


Fig 3.2: A simplified programmable logic device

3.1.2 CPLD

For large logic circuits, complex programmable logic devices (CPLD) can be used. A CPLD consists of a set of interconnection network. The connection between the input/output blocks and the macro cells and those between macro cells and macro cells can be made through the programmable interconnection network.

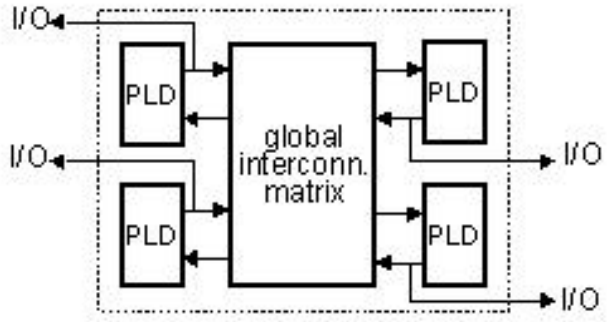


Fig 3.3: CPLD

3.1.3 FPGA

A FPGA is a programmable device consisting, like the CPLDs, of three main parts. A set of programmable logic cells also called logic blocks or configurable logic blocks, a programmable interconnection network and a set of input and output cells around the device. A function to be implemented in FPGA is partitioned in modules, each of which can be implemented in a logic block. The logic blocks are then connected together using the programmable interconnection

In practical view point, there are two differences between FPGA and CPLD; first, the FPGAs with thousand gate capacity have more facilities instead of CPLDs to design more complex and huge digital systems. Second, FPGAs use more programmable switches for FPGA's interconnection blocks that have more delay. Therefore FPGAs have more delays instead of CPLDs and PALs. Also FPGAs are implementable by using computer aided design (CAD) tools such as VHSIC (Very High Speed Integrated Circuit) Hardware Design Language (VHDL).

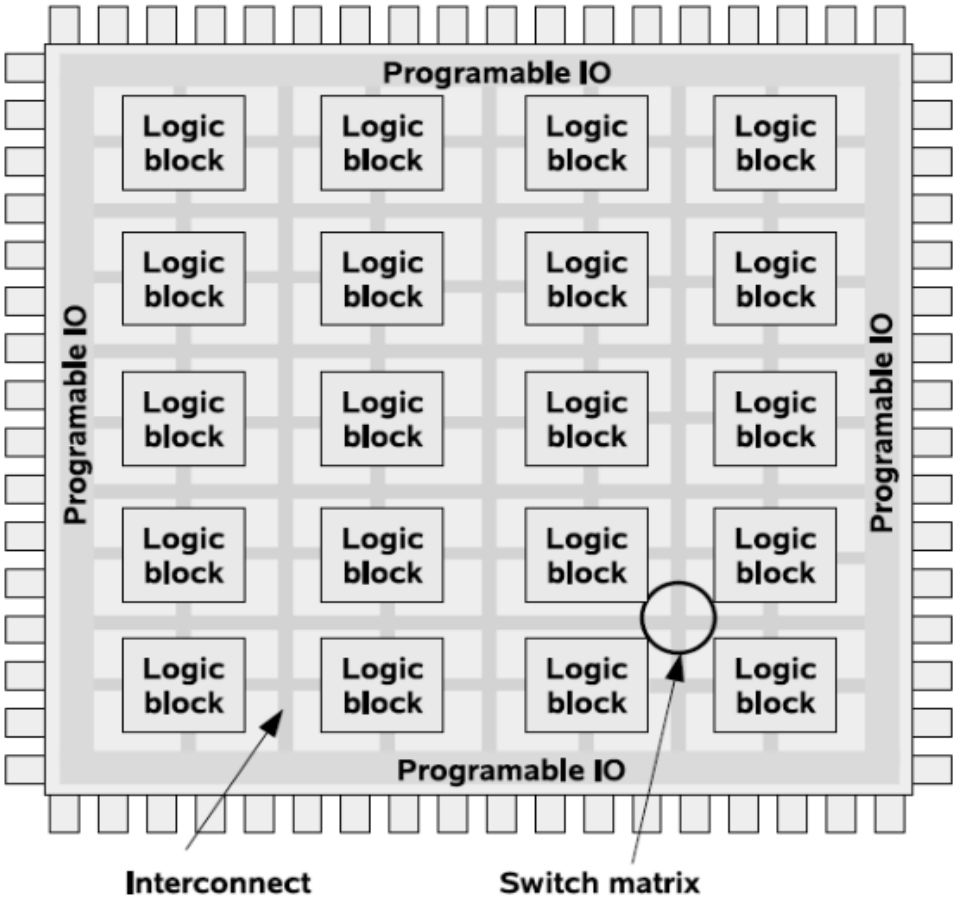


Fig 3.4: Basic FPGA Architecture

3.2 System Level Architecture

A reconfigurable system typically consists of one or more processors, one or more reconfigurable fabrics, and one or more memories. Reconfigurable systems are often classified according to the degree of coupling between the reconfigurable fabric and the CPU. Computer scientists K. Compton and S. Hauck present the four classifications:

3.2.1 External Stand-Alone Processing Unit

In this type of coupling, the reconfigurable fabric is in the form of one or more stand-alone devices. The existing input and output mechanisms of the processor are used to communicate with the reconfigurable fabric. In this configuration, the data transfer between the fabric and the processor is relatively slow, so this architecture only makes sense for applications in which a significant amount of processing can be done by the fabric without processor intervention. Emulation systems like circuit emulation often take on this sort of architecture

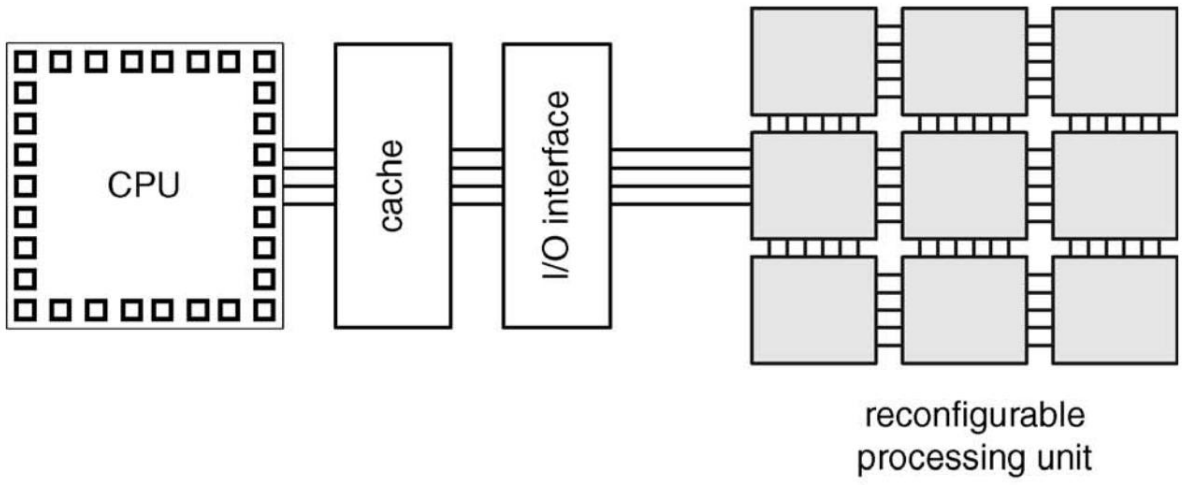


Fig 3.5: External Stand-Alone Processing Unit

3.2.2 Attached Processing Unit

In this type of coupling the reconfigurable fabric is coupled or interfaced with the processor using the system bus. This in turn increases the communication speed of the processor and the reconfiguration fabric. This type of coupling is used in applications where processing power demanding functions are to be carried out. Consider the example of video processing. The reconfigurable fabric in this case will be reconfigured to process video while the processor will be executing other tasks. Since the reconfigurable fabric is configured for video processing, it will process the video signals very efficiently and with speed.

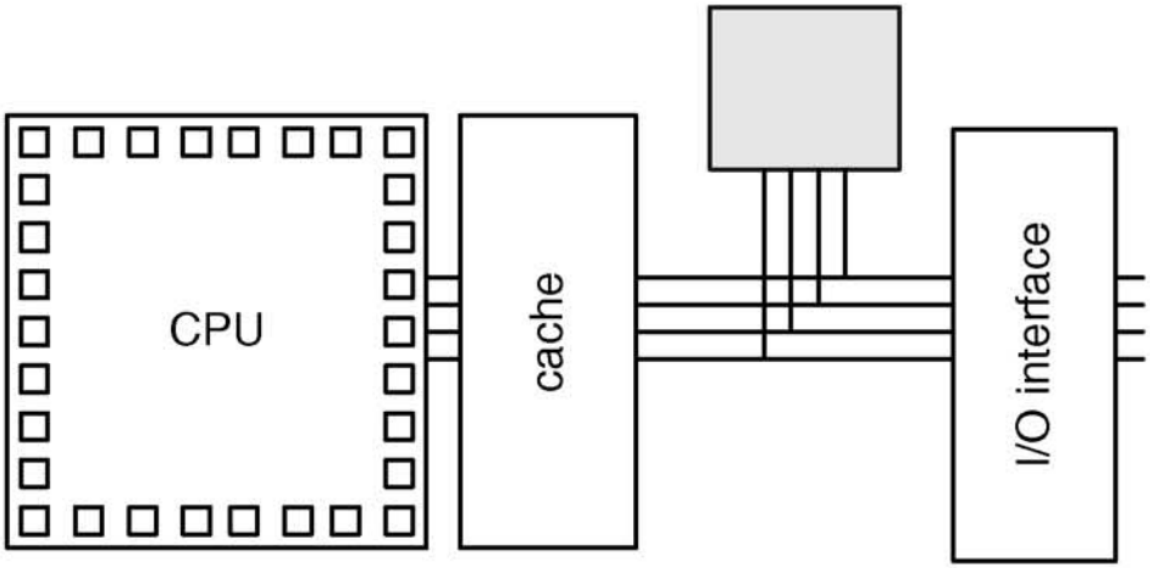


Fig 3.6: Attached processing unit

3.2.3 Co-Processor

This type of coupling involves interfacing the reconfigurable fabric directly with the processor. The processor and the reconfigurable fabric still remains two physically independent units. In this setup the reconfigurable fabric acts as a co-processor to the main processor. As the communication in this case is direct between the processor and the reconfigurable fabric, the cost of communication delay is lower. This type of coupling is done when the processor requires a fast and dedicated co-processor to do some specific calculations or computations. Example say floating point or cryptographic computations.

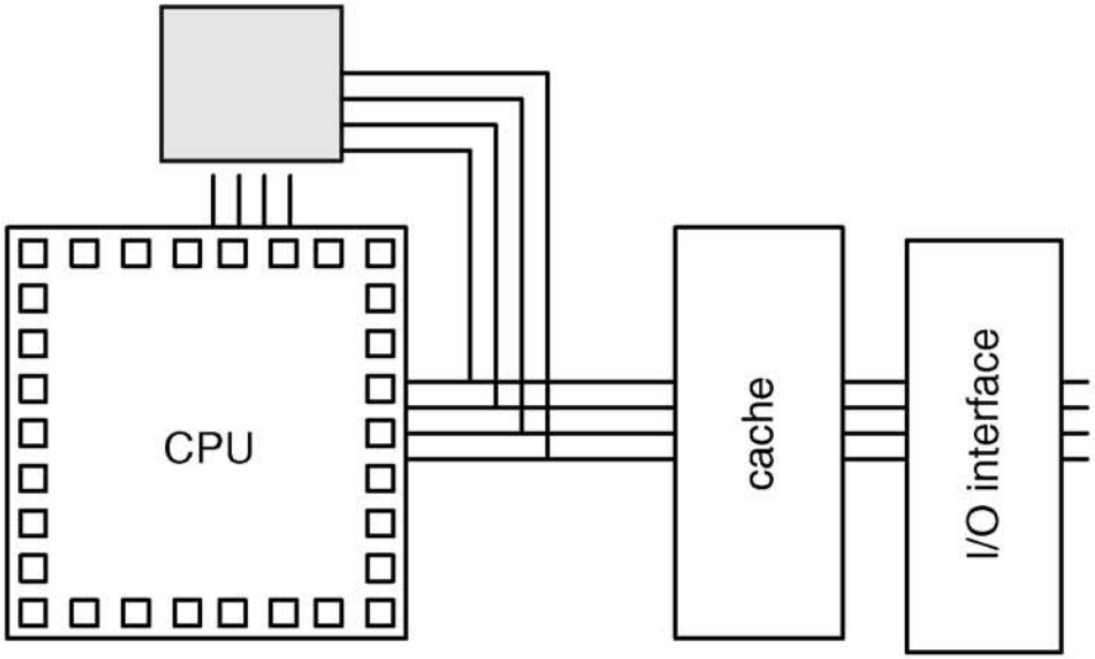


Fig 3.7: Co-processor

3.2.4 Reconfigurable Functional Unit

In this type of coupling, the reconfigurable fabric is embedded in the same dye as that of the processor. That is both the processor and the reconfigurable fabric are present in the same physical unit. Here the processor and the reconfigurable fabric are very tightly coupled. This kind of setup is used when the reconfigurable fabric is used as a part of the processor itself to execute custom instructions.

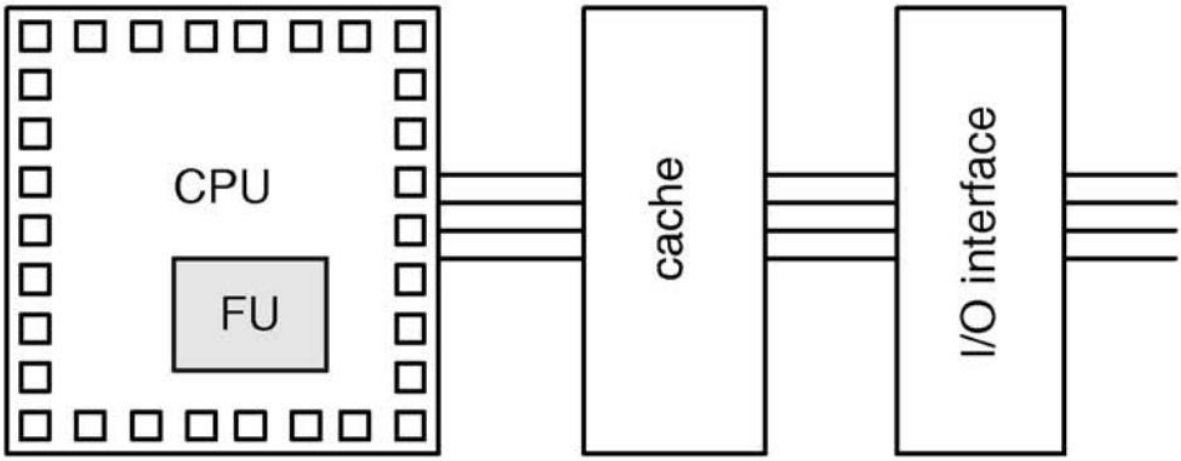


Fig 3.8: Reconfigurable functional unit

3.2.5 Processor Embedded In Reconfigurable Fabric

The last type of coupling is the processor embedded in the reconfigurable fabric. This processor can be a hard core or soft core implementation using the resources of the reconfigurable fabric itself. This type of coupling is used in applications that require some amount of general purpose hardware for increased flexibility or execution of random instructions.

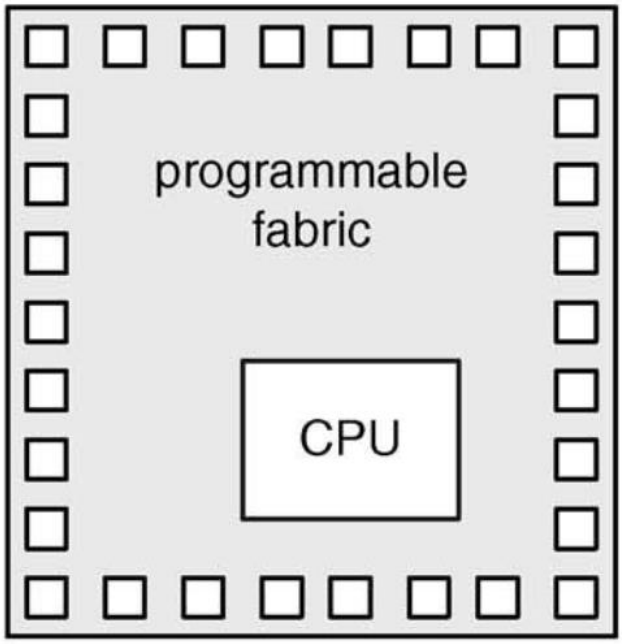


Fig 3.9: Processor embedded in reconfigurable fabric

3.3 Granularity

Granularity is defined as the word length of the data being processed. The logic block inside the reconfigurable processor is defined by its internal structure and granularity. The structure defines the different kinds of logic that can be implemented in the block, while the granularity defines the maximum word length of the implemented functions. The functionality of the logic block is obtained by controlling the connectivity of some basic logic gates or by using LUTs and has a direct impact on the routing resources. As the functional capability increases, the amount of logic that can be packed into it increases.

A collection of CLBs, known as logic cluster, is described with the following four parameters the size of (number of inputs to) the LUT, the number of CLBs in a cluster, the number of inputs to the cluster for use as inputs by the LUTs and the number of clock inputs to a cluster (for use by the registers). Thus, the size and complexity of the basic computing blocks is referred to as the block granularity. All the reconfigurable platforms based on their granularity are distinguished into two types:

3.3.1 Fine-grained architecture

3.3.2 Coarse-grained architecture

3.3.1 Fine-grained Architecture

Fine grained architectures use very small data lengths, one bit wide or a few bits wide. They usually process the data using simple circuits like look up tables, logic gates, multiplexers, adders etc. Normal FPGAs use fine grained architecture. The advantage of fine grained architecture is that it is highly flexible as the data manipulation is just one bit. Also it is very fast because the delay time associated with individual components in the circuit is very small. One disadvantage of fine-grained architecture is that, when the data to be processed is very large, bit by bit processing of the data becomes impractical.

Consider the simple circuit shown in fig 3.10. It shows a logic block. There are 4 inputs to the logic block. This is an example of a simple fine grained architecture.

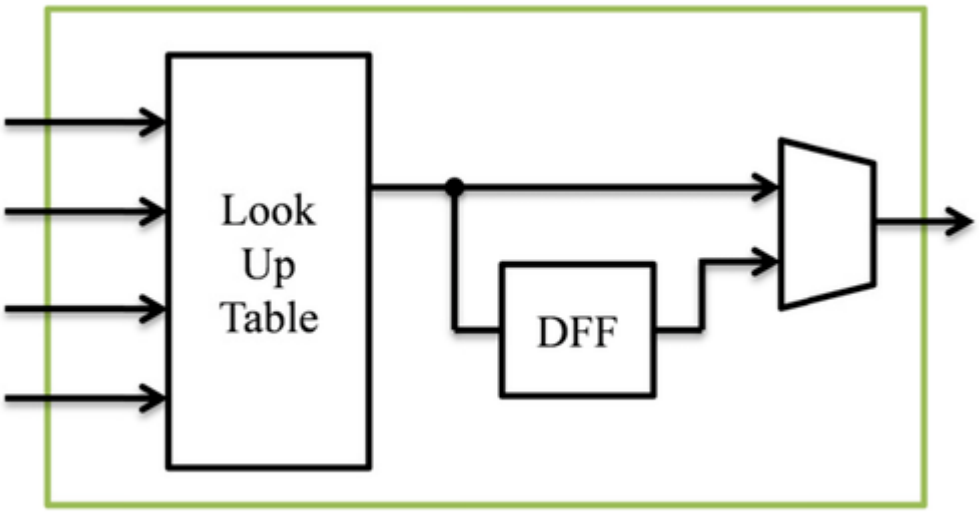
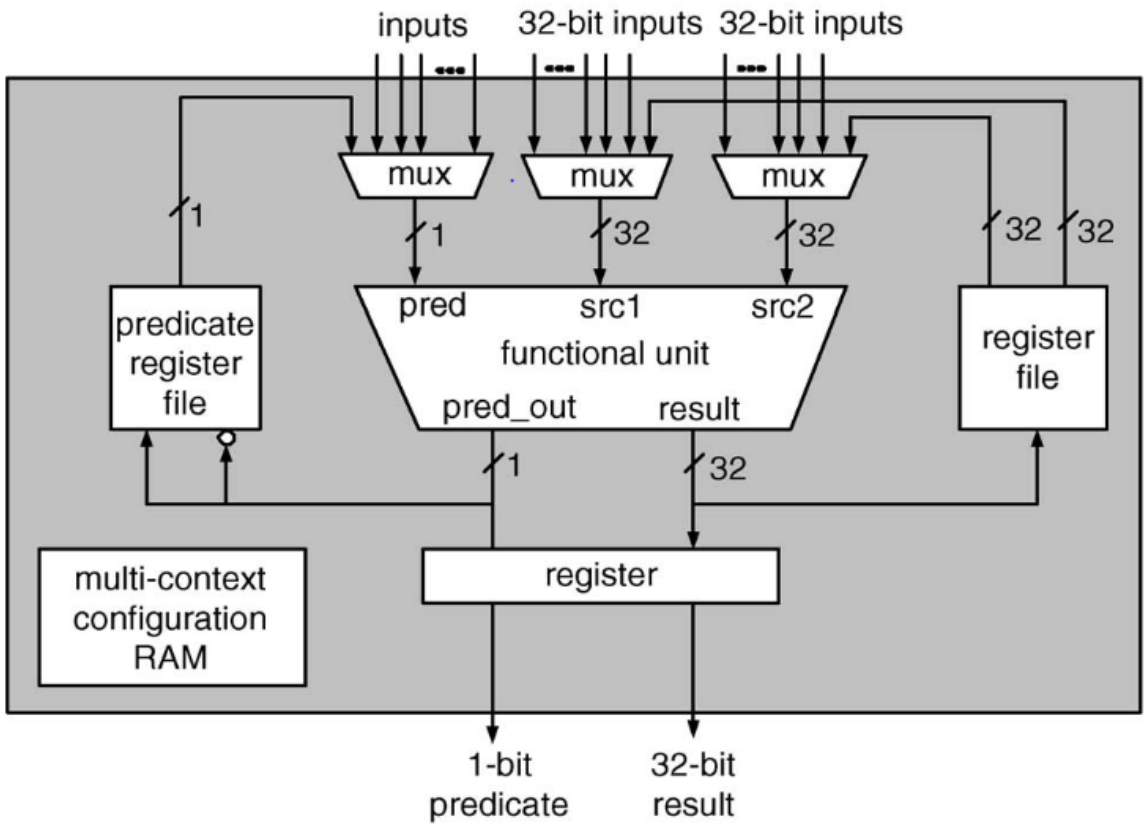


Fig 3.10: Fine-grained logic block

3.3.2 Coarse-grained Architecture

Coarse-grained architectures use very large data lengths. They process data that are many bits wide. Many of the modern FPGAs have started using coarse-grained architecture in their FPGAs. The advantage of coarse-grained architecture is that since it is processing very large data lengths, the overall throughput of the processor becomes very high. Also this can be used for processing power demanding tasks like Digital image processing, digital signal processing etc.

Consider the example of the figure shown below. It shows the Architecture for Dynamic Reconfigurable Embedded System (ADRES). In the figure you can see that the individual components that are present in the logic block are those that processes very large data length inputs.



Fib 3.11: ADRES

A number of reconfigurable systems use a granularity of logic block that we categorize as medium-grained. Medium-grained logic blocks may be used to implement datapath circuits of varying bit widths, similar to the fine-grained structures. However, with the ability to perform more complex operations of a greater number of inputs, this type of structure can be used efficiently to implement a wider variety of operations.

3.4 Programmable Logic Elements

Programmable logic elements/cells are used to implement Boolean equations with more number of inputs. The complication of the individual cells is dependent on the number of inputs to the programmable logic cell. There generally two kinds of programmable logic cells used in reconfigurable systems. They are:

3.4.1 Look up Table (LUT)

3.4.2 Configurable logic blocks (CLB)

They are used w.r.t the complexity and the number of inputs that each logic cell has to handle.

3.4.1 Look Up Table (LUT)

A look-up table (LUT) is a group of memory cells, which contain all the possible results of a given function for a given set of input values. The input values are fed into the look up table and the corresponding output is given at the output terminal. A LUT can be considered as a ROM with N-inputs. In modern RPs, LUT is achieved by using a mux. The inputs are fed as the select lines of the mux while the corresponding output values are fed as input into the mux. The LUT can compute any function of n inputs by simply programming the lookup table with the truth table of the function we want to implement. As shown in the fig 3.12 if we wanted to implement a full adder with our 3-input LUT (often referred to as a 3-LUT), we would assign values to the lookup table memory such that the pattern of select bits chooses the correct row's "answer". The circuit is shown in fig 3.13

The output values will be located in SRAM memory bits. Therefore complicated functions with a large number of inputs can be implemented by combining several lookup tables together. As Fig. shown, in some applications we need to store the output bit of LUT, the D-Flip flop will attach at the end. A D flip-flop (DFF) is included to provide state-holding elements. This DFF can be bypassed when not needed, by selecting the appropriate multiplexer input behind the flip-flop.

A	B	C_{in}	C_{out}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Fig 3.12: Truth table of Full adder

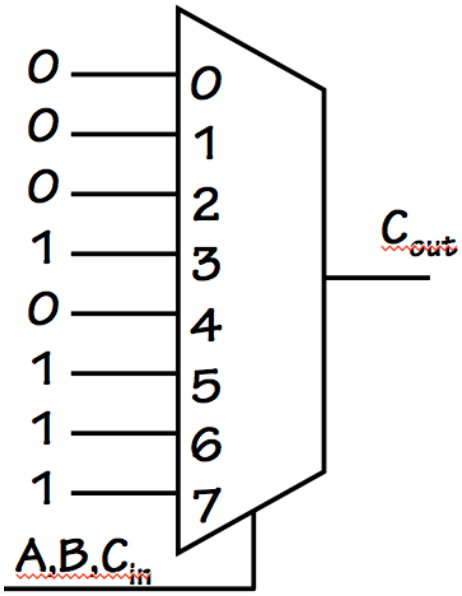


Fig 3.13: Full adder using 3 input MUX

3.4.2 Configurable Logic Blocks (CLB)

Configurable logic blocks are used when more number of inputs are to be processed or more complex processing is to be done. Most of the modern FPGAs use CLBs instead of just LUTs. This is because using CLBs a vast amount of functions can be implemented with ease.

In general, a logic block consists of a few logical cells (called ALM, LE, Slice etc.). A typical cell consists of a 4-input LUT, a Full adder (FA) and a D-type flip-flop, as shown in fig 3.14. The LUTs are in this figure split into two 3-input LUTs. In normal mode those are combined into a 4-input LUT through the left mux. In arithmetic mode, their outputs are fed to the FA. The selection of mode is programmed into the middle multiplexer. The output can be either synchronous or asynchronous, depending on the programming of the mux to the right, in the figure example. In practice, entire or parts of the FA are put as functions into the LUTs in order to save space.

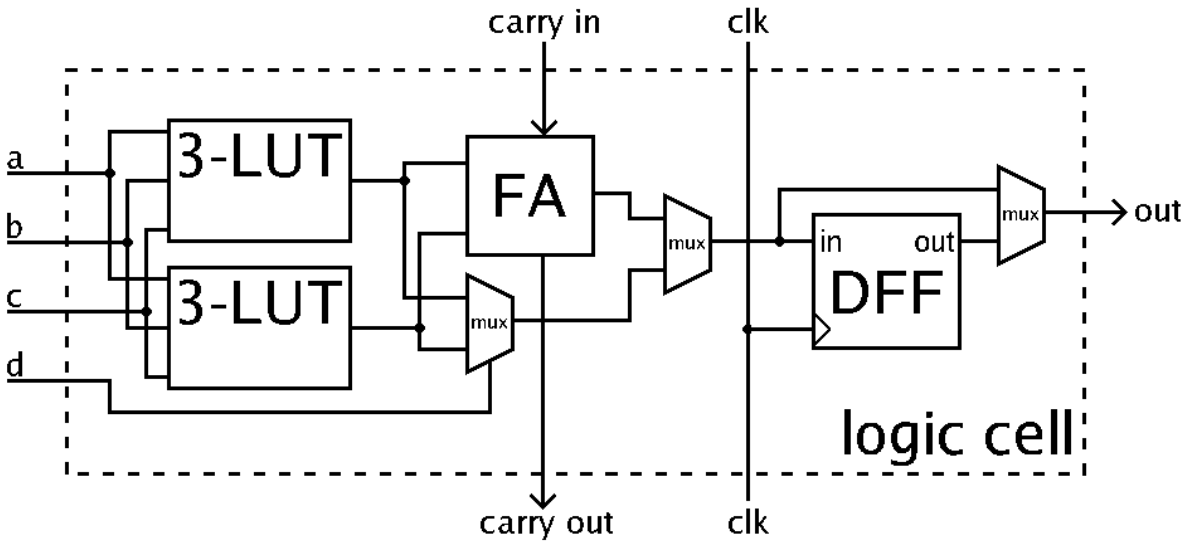


Fig 3.13 A logic cell

Logic blocks typically contain a few LEs/Slices. ALMs and Slices usually contain 2 or 4 structures similar to the example figure, with some shared signals.

3.5 Reconfiguration Models

Reconfiguration is the process of altering the circuit design in a Reconfigurable Processor. It is by reconfiguration, we invoke the property of hardware reusability and flexibility in a reconfigurable processor. Reconfiguration can be done in three modes.

3.5.1 Static reconfiguration

3.5.2 Partial reconfiguration

3.5.3 Dynamic reconfiguration

The basis of difference in the above three modes is the time at which the reconfigurable processors can be reconfigured.

3.5.1 Static Reconfiguration

Static reconfiguration (often referred as compile time reconfiguration) is the simplest and most common approach for implementing applications with reconfigurable logic. Static reconfiguration involves hardware changes at a relatively slow rate. It is a static implementation strategy where each application consists of one configuration. The distinctive feature of this configuration is that it consists of a single system-wide configuration. Prior to commencing an operation, the reconfigurable resources are loaded with their respective configurations. Once operation commences, the reconfigurable resources will remain in this configuration throughout the operation of the application. Thus hardware resources remain static for the life of the design whereas static reconfiguration allocates logic for the duration of an application.

If the design has to be changed, the processor has to be reconfigured again. Reconfiguration in this case is a time consuming process and hence this kind of reconfiguration model is only use when the design of the processor is to never or occasionally be changed. This model of reconfiguration is used in olden and present day entry-level FPGAs.

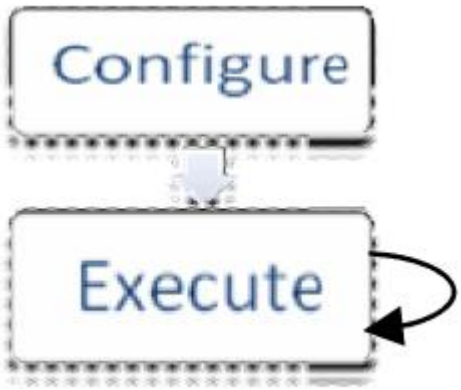


Fig 3.14: Static reconfiguration model

3.5.2 Partial Reconfiguration

In some cases, configurations do not occupy the full reconfigurable hardware, or only a part of a configuration requires modification. In both of these situations a partial reconfiguration of the reconfigurable resources is desired, rather than the full reconfiguration supported by the serial architectures mentioned above. When configurations do not require the entire area available within the array, a number of different configurations may be loaded into otherwise unused areas of the hardware. Partially runtime reconfigurable architectures can allow for complete reconfiguration, or may require a full column of configuration information to be reconfigured at once.

3.5.3 Dynamic Reconfiguration

The dynamic allocation scheme re-allocates hardware at run-time. This is an advanced technique that some people regard as a flexible realization of the time/space trade-off. It can increase system performance by using highly optimized circuits that are loaded and unloaded dynamically during the operation of the system .In this way system flexibility is maintained and functional density is increased.

The dynamic reconfiguration has two main design problems. The first is to divide the algorithm into time-exclusive segments that do not need to run concurrently. The second problem is to co-ordinate the behaviour between different configurations, i.e. the management of transmission of intermediate results from one configuration to the next.

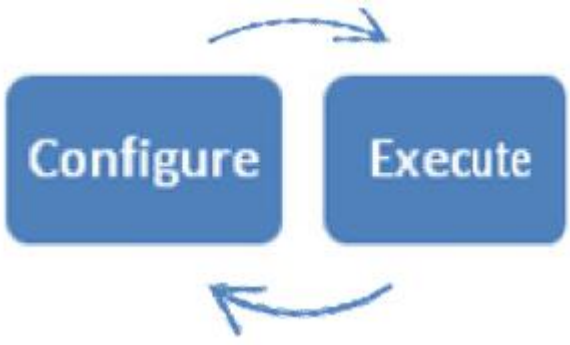


Fig 3.15: Dynamic reconfiguration model

Chapter 4

ADVANTAGES AND LIMITAIONS

4.1 Advantages of Reconfigurable Processors

- Reconfigurable processors provide a mix of both the speed of custom electronic circuits and the flexibility offered by general purpose processors
- Processing efficiency is very high as the circuitry is specially designed for the application it is performing
- Power consumption is very low as there are no unwanted logic or other unused elements in the circuit that can result in idle power consumption
- Hardware reusability is supported
- Reduced developmental and implementation costs
- Reduced risk of hardware errors which can otherwise happen in ASICs

4.2 Limitation of Reconfigurable Processors

- Semantic gap between algorithms and circuits is still a major obstacle
- Capacity of the various commonly available reconfigurable processors available today post an hindrance to the implementation of very large circuits
- Speed is a factor since most the of the reconfigurable processors run at considerably lower clock speed than the general purpose processor available today
- The learning curve of reconfigurable processor designing and implementation is very steep compared to the software development curve associated with general purpose processors
- The developmental tools associated with reconfigurable processor development are still in development stages
- Inertia to change from general purpose processors to hardware implementation is very high due to the already invested amount in the software development field

Chapter 5

APPLICATIONS

Reconfigurable processors find a wide range of applications. Though the topic wasn't considered much into consideration during earlier periods, nowadays there are many applications that are done using reconfigurable processors. Some of them are given below:

- Very fast cryptography – encryption and decryption throughputs can be increased drastically using reconfigurable processors
- High speed signal processing – Signal processing both video and audio, esp. video processing, requires a very powerful processor. This can be done easily by using reconfigurable processors
- Numeric and scientific computing – Reconfigurable processors can be used for efficient high speed scientific computing like DNA structure analysis, protein structure analysis, mathematical calculations
- Market analysis and pattern matching – Reconfigurable processors can be used for stock market pattern analysis that can be useful for stock market trading and other financial activities
- Networking – Networking is an area where heterogeneous data is to be handled. Nowadays, this is done by using dedicated hardware such as routers, switches, bridges etc. each performing different functions. Using reconfigurable processors all these different hardware can be made into a single hardware that performs all these functions

Chapter 6

CONCLUSION

Reconfigurable processors is the next big thing in the hardware domain. Due to its flexibility, performance and efficiency the reconfigurable processors has a very good space of development. The main advantage of reconfigurable processor is that, the development is similar to software development but the result is hardware circuit development. The fact that custom hardware can perform much better than general purpose processors, reconfigurable processors are constantly being incorporated into use in different areas. .Reconfigurable processors are finding applications in new fields like supercomputing, multi parallelism array processors, advanced flight simulation and fluid mechanics, 3-D rendering. In the near future we can expect even our personal computers to be equipped with reconfigurable processors instead of general purpose processors.