FPGA : Architecture, Configuration and Design Flow

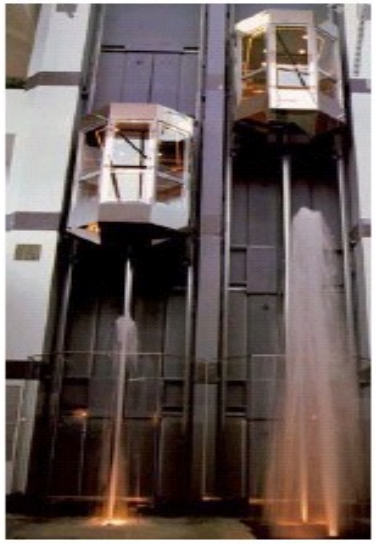
Prof. Amitabha Sinha

Director , School Of IT
West Bengal University of Technology

Electronic Products

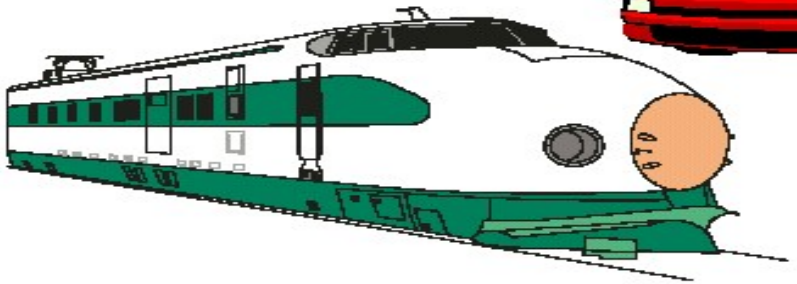
Basic building blocks for most of the electronic products are VLSI devices

Some Applications



Embedded System =

Computers Inside a Product



Range of Applications---Low end to high end

- Digital cellular phones
- Automated inspection
- Vehicle collision avoidance
- Voice over Internet
- Motor control
- Consumer audio
- Voice mail
- Navigation equipment
- Audio production
- Videoconferencing
- Pagers
- Music synthesis, effects
- Satellite communications
- Seismic analysis
- Secure communications
- Tapeless answering machines
- Sonar
- Cordless phones
- Digital cameras
- Modems (POTS, ISDN, cable,...)
- Noise cancellation
- Medical ultrasound
- Patient monitoring
- Radar

Issues Involved in Designing State-of-the-art VLSI devices

- Low Cost
- High speed
- Flexibility
- Low Power consumption

But Why Flexibility?

- To add more features without changing the basic hardware

  ----- Which leads to faster time to market

  -------- reaches the customer with a cost

  effective solution

  because

  long product development cycle can be largely reduced

  thereby reducing the product development cost

How the Flexibility can be achieved?

- By incorporating programmability in the hardware

But Microprocessors/ DSP Processors are Programmable Processors and available off-the self.

Then??????

Microprocessors/DSP Processors are based on Von-Neumann architectural concept where a given application(Program) is stored in the memory in sequential fashion and is executed sequentially!!!!

Contd.

- So, these Von-Neumann Processors fail to exploit the concurrency in the algorithm even though they offer flexibility.

-------- Thus performance is not achieved for many applications especially where high speed is required.

This leads to an alternative solution where,

Speed of the hardware can be achieved retaining the flexibility of the software(programmability).

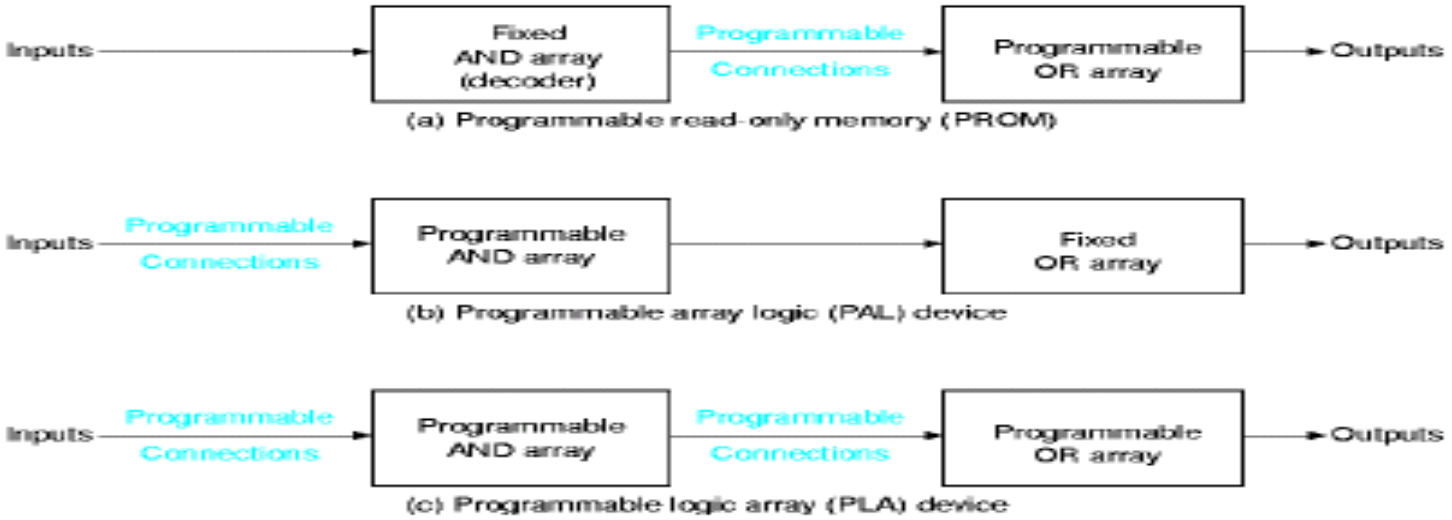
Programmable Hardware

Programmable hardware offers a balance between flexibility and efficiency

because

Potential concurrency in the algorithm can be exploited by directly mapping the algorithm onto architecture

PROGRAMMABLE LOGIC DEVICES



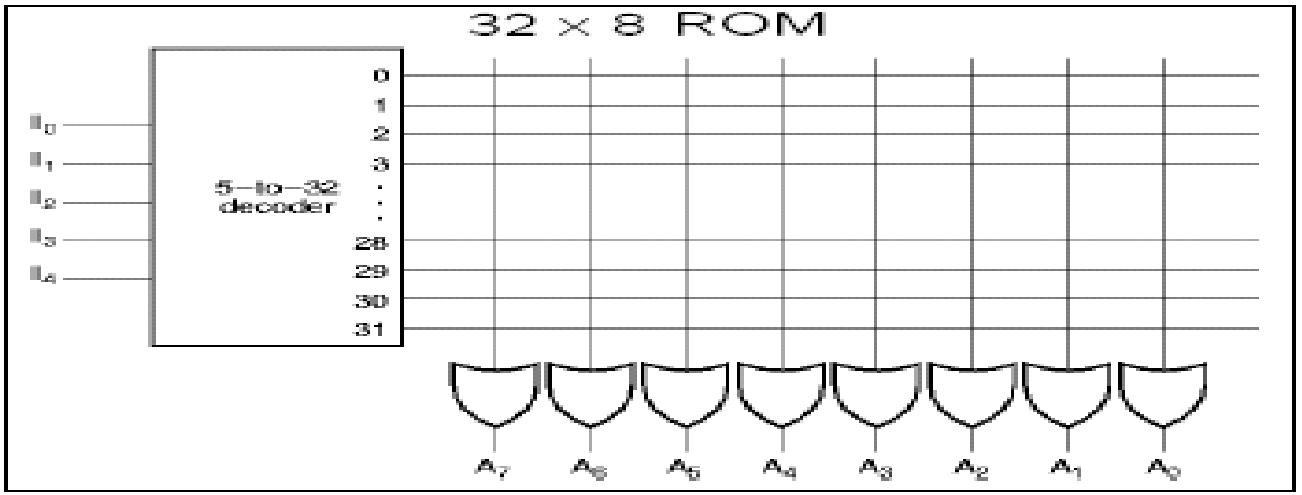
PLDs (combinatorial circuits): ROM, PLA, PAL, CPLD, and FPGA

Store *permanent* binary information (nonvolatile). Can be read only (cannot be altered). Information is specified by designer and *physically inserted* (embedded) into the PLD

Programmable connections are formed by *fuses*, *masks*, or *antifuses* depending on the technology. Irreversible programming

Read-Only Memory

k inputs (address) $\Rightarrow$ $2^k \times n$ ROM $\Rightarrow$ n outputs (data)



- $k \times 2^k$ decoder to decode input address
- n OR gates with 2^k input each
- Decoder output is connected to all n OR gates through fuses
- ROM $\longrightarrow$ $2^k \times n$ programmable connections

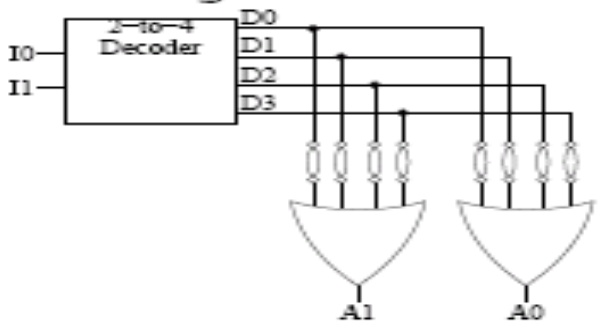
Programming a ROM

Example of 4 × 2 ROM

Truth table

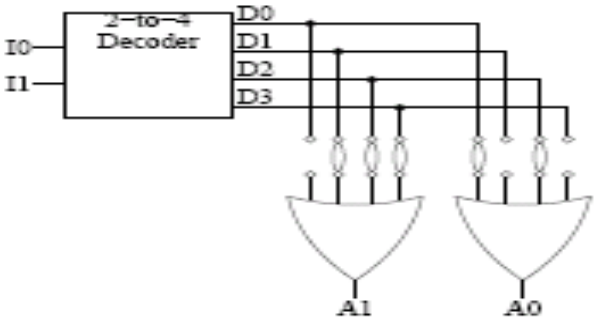
Address		Content	
I_1	I_0	A_1	A_0
0	0	0	1
0	1	1	0
1	0	1	1
1	1	1	0

NonProgrammed ROM

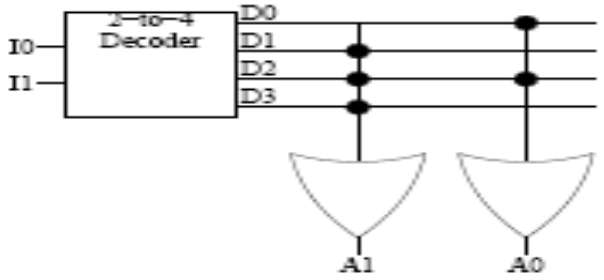


⇓

Programmed ROM



Compact ROM



Truth table → address and content of ROM

Programming → stores truth table in ROM

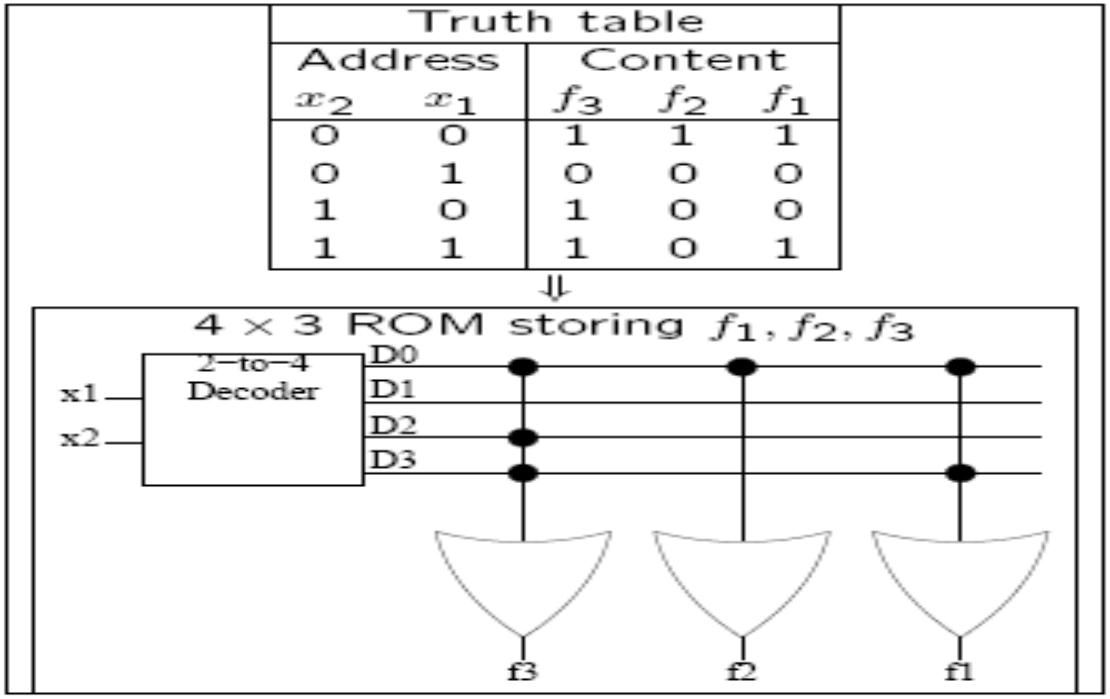
- 0 = Open connection = Fuse blown
- 1 = Closed connection = Fuse intact

Function Synthesis with ROM

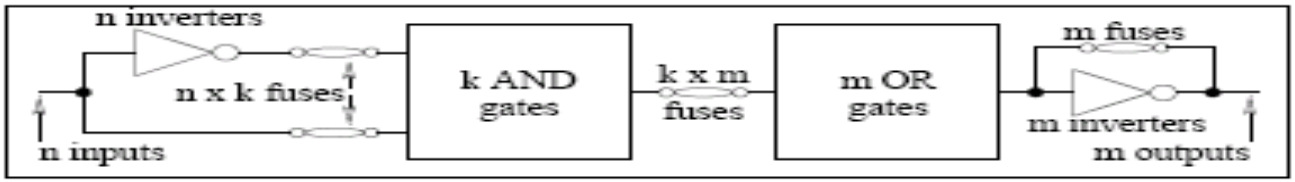
Any set of functions $f_1(x_k, \ldots, x_1)$, $\ldots$, $f_n(x_k, \ldots, x_1)$ can be realized with a $2^k \times n$ ROM

Example: Implement $f_1(x_2, x_1) = \sum m(0, 3)$, $f_2(x_2, x_1) = \overline{x_2 + x_1}$, and $f_3(x_2, x_1) = \prod M(1)$ with a 4×3 ROM

Truth table				
Address		Content		
x_2	x_1	f_3	f_2	f_1
0	0	1	1	1
0	1	0	0	0
1	0	1	0	0
1	1	1	0	1



Programmable Logic Array



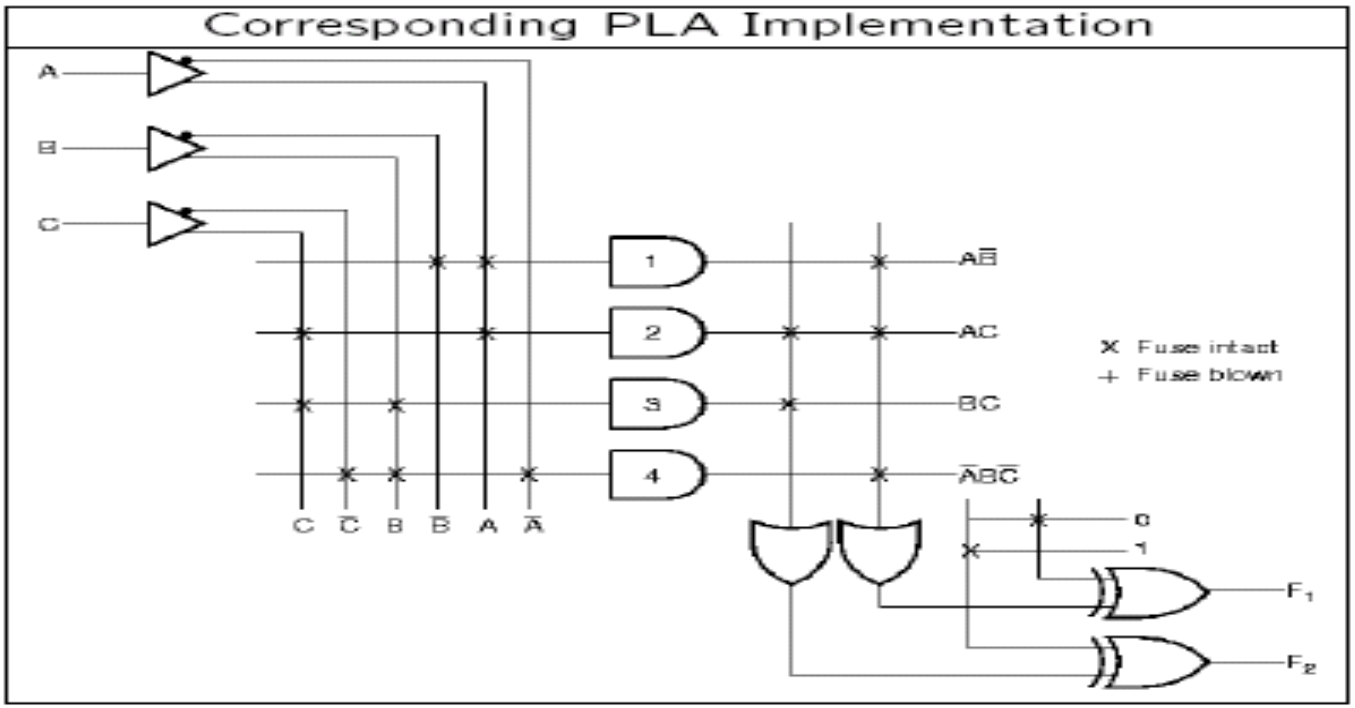
Behave like a ROM but has different structure

- Uses ANDs array instead of decoder to produce product terms of inputs
- Has programmable connections before ANDs, between ANDs and ORs, after ORs. That is $2nk + km + m$ fuses
- More flexible than ROM but more difficult to program
- Logic expressions for content information to be stored in PLA must be obtained fisrt, then minimized, and finally programmed into the PLA using a PLA program table
- PLA program table specifies product terms and sum terms of information that will be stored in PLA

Programming a PLA

PLA Program Table						
		Inputs			Outputs	
Term	Term#	A	B	C	F_1	F_2
$A\bar{B}$	1	1	0	_	1	_
AC	2	1	_	1	1	1
BC	3	_	1	1	_	1
$\bar{A}B\bar{C}$	4	0	1	0	1	_
					T	C

Corresponding PLA Implementation



Function Synthesis with PLA

Any set of functions $f_1(x_1, \ldots, x_n)$, ..., $f_m(x_1, \ldots, x_n)$ can be realized with a PLA

Example Implement $f_1(a, b, c) = \sum m(3, 5, 6, 7)$ and $f_2(a, b, c) = \sum m(0, 2, 4)$ with a PLA

First Simplify $f_1, \bar{f}_1, f_2, \bar{f}_2$, that is

$$f_1(a, b, c) = ab + ac + bc \qquad f_1, f_2 \rightarrow 5 \text{ terms}$$
$$\bar{f}_1(a, b, c) = \bar{a}\bar{b} + \bar{a}\bar{c} + \bar{b}\bar{c} \qquad f_1, \bar{f}_2 \rightarrow 4 \text{ terms}$$
$$f_2(a, b, c) = \bar{a}\bar{c} + \bar{b}\bar{c} \qquad \bar{f}_1, f_2 \rightarrow 3 \text{ terms}$$
$$\bar{f}_2(a, b, c) = ab + c \qquad \bar{f}_1, \bar{f}_2 \rightarrow 5 \text{ terms}$$

Second Select combination of functions that has less terms, that is

$$f_1 = \overline{\bar{f}_1} = \overline{\bar{a}\bar{b} + \bar{a}\bar{c} + \bar{b}\bar{c}}$$
$$f_2(a, b, c) = \bar{a}\bar{c} + \bar{b}\bar{c}$$

Third Construct a PLA program table from selected functions

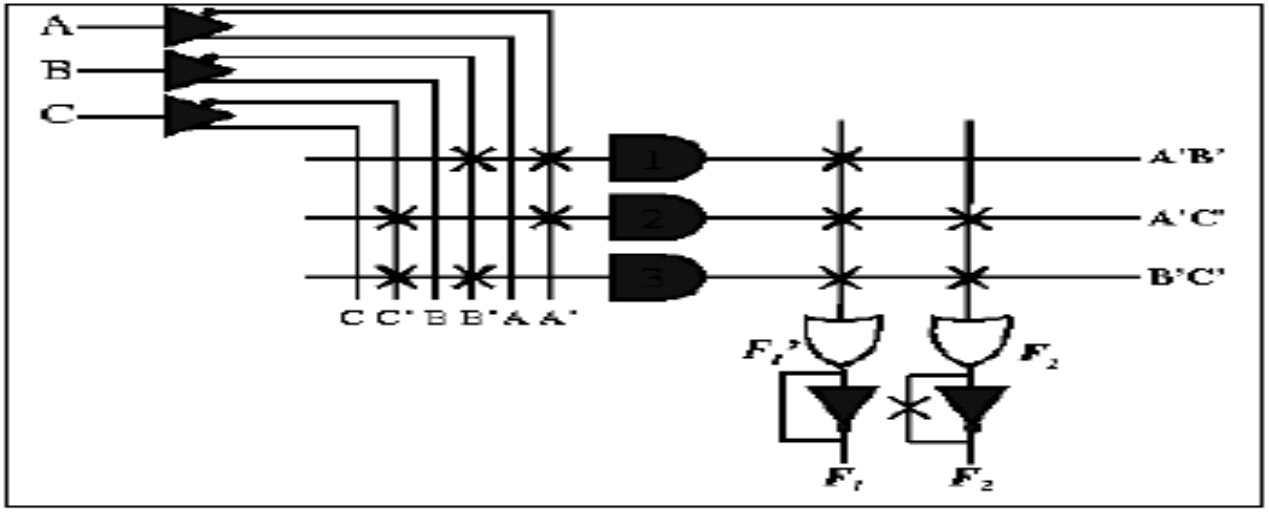
Term	Term#	Inputs a	b	c	Outputs f_1	f_2
$\bar{a}\bar{b}$	1	0	0	_	1	_
$\bar{a}\bar{c}$	2	0	_	0	1	1
$\bar{b}\bar{c}$	3	_	0	0	1	1
					C	T

Function Synthesis with PLA
(continued)

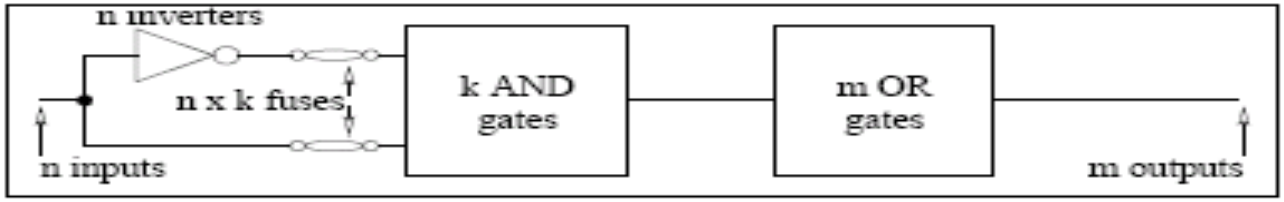
Third Construct a PLA program table from selected functions

Term	Term#	Inputs a	b	c	Outputs f_1	f_2
$\bar{a}\bar{b}$	1	0	0	_	1	_
$\bar{a}\bar{c}$	2	0	_	0	1	1
$\bar{b}\bar{c}$	3	_	0	0	1	1
					C	T

Fourth Construct PLA circuit from PLA program table



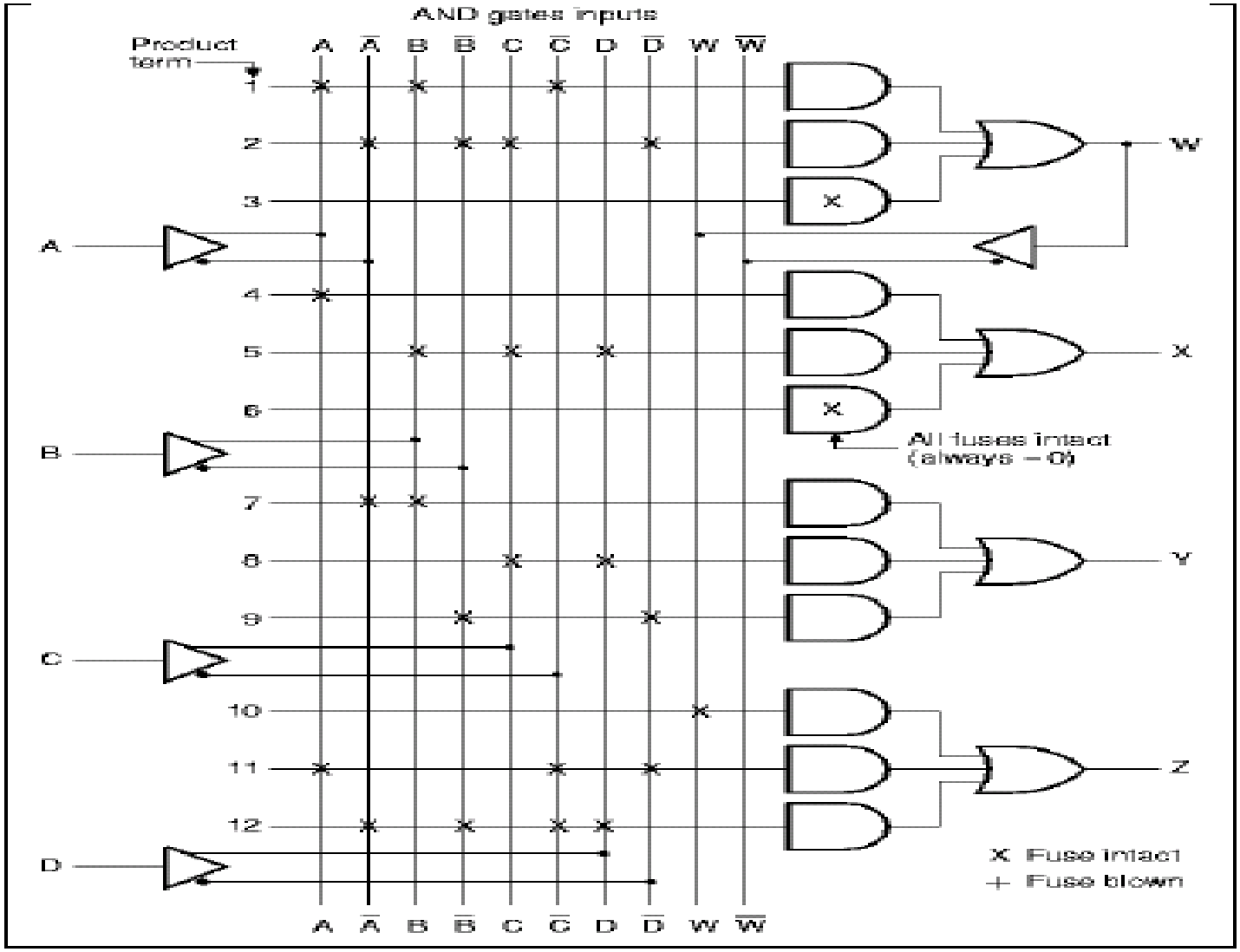
Programmable Array Logic



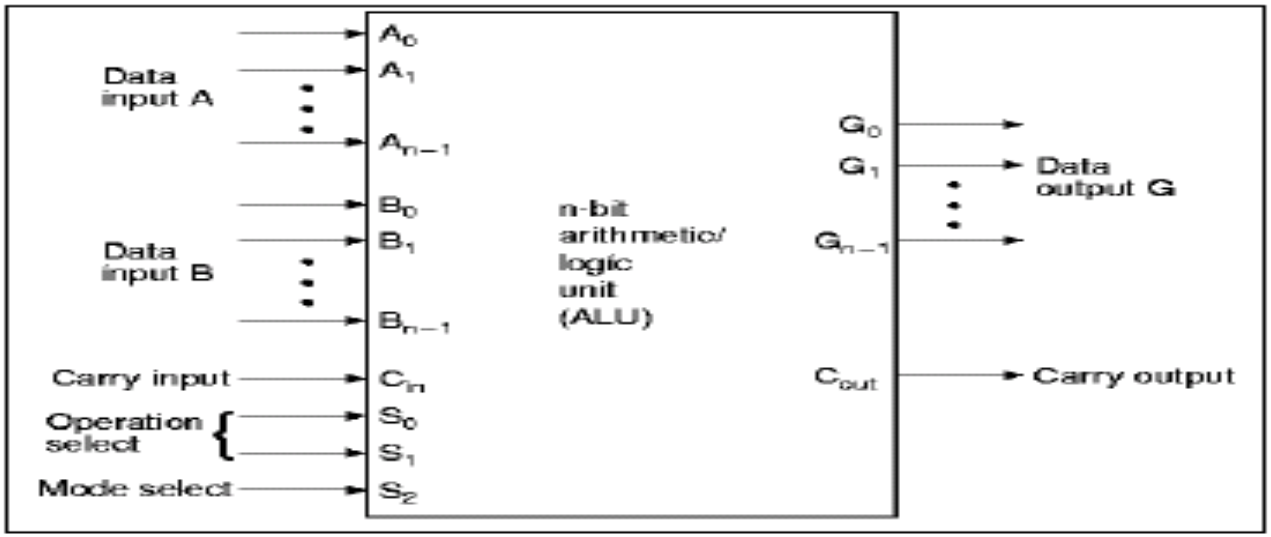
Similar to PLA

- Only the connection inputs to ANDs are programmable
- Easier to program than but not as flexible as PLA
- There are feedback connections
- Logic expressions for content information to be stored in PAL must be obtained fisrt, then minimized, and finally programmed into the PAL using a PAL program table
- PAL program table specifies only product terms of information that will be stored in PAL

Programming a PAL



Arithmetic-Logic Unit



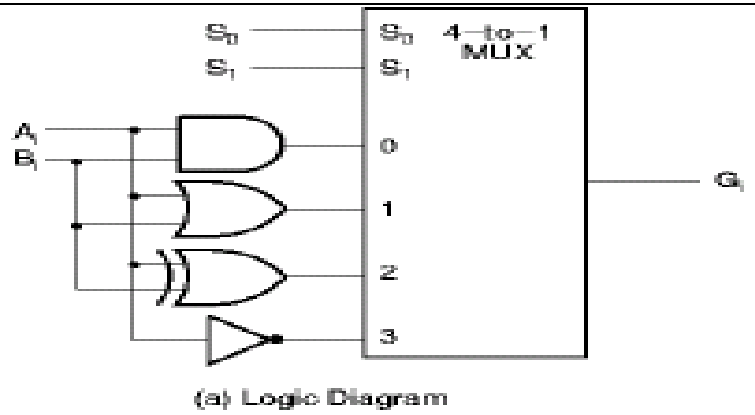
Essential element of the Central Processing Unit

Arithmetic and logic functions on binary words

- n-bit data inputs A and B
- n-bit data output $G = f(A, B)$
- Selection inputs S_0, S_1 select a function f
- Selection input S_2 select an operating mode (arithmetic or logic)

ALU
(continued)

Logic Circuit

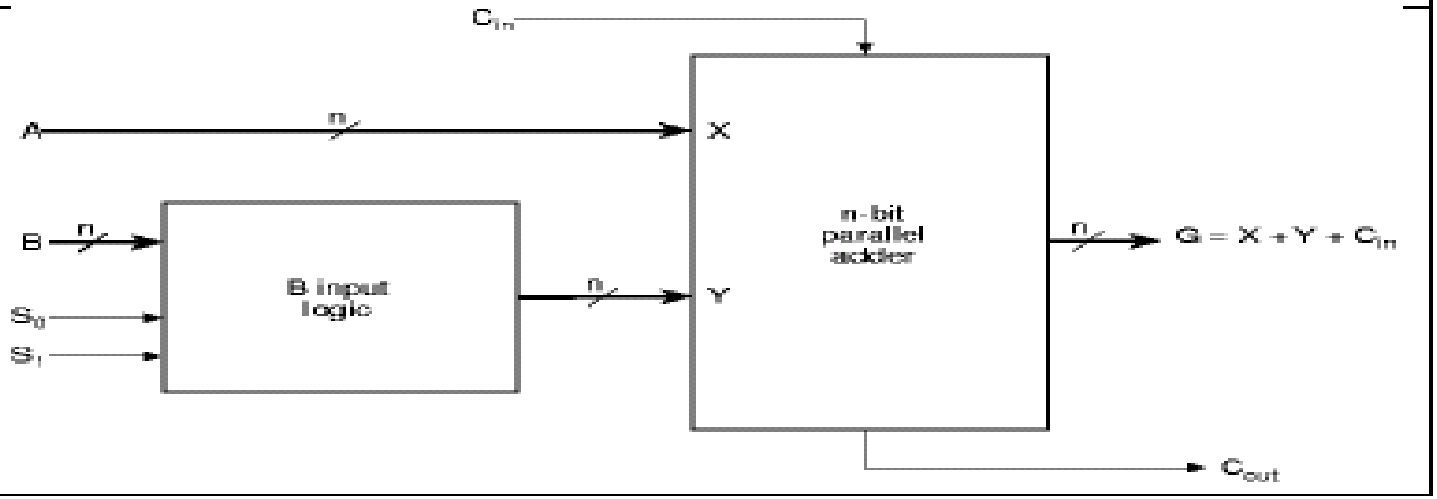


(a) Logic Diagram

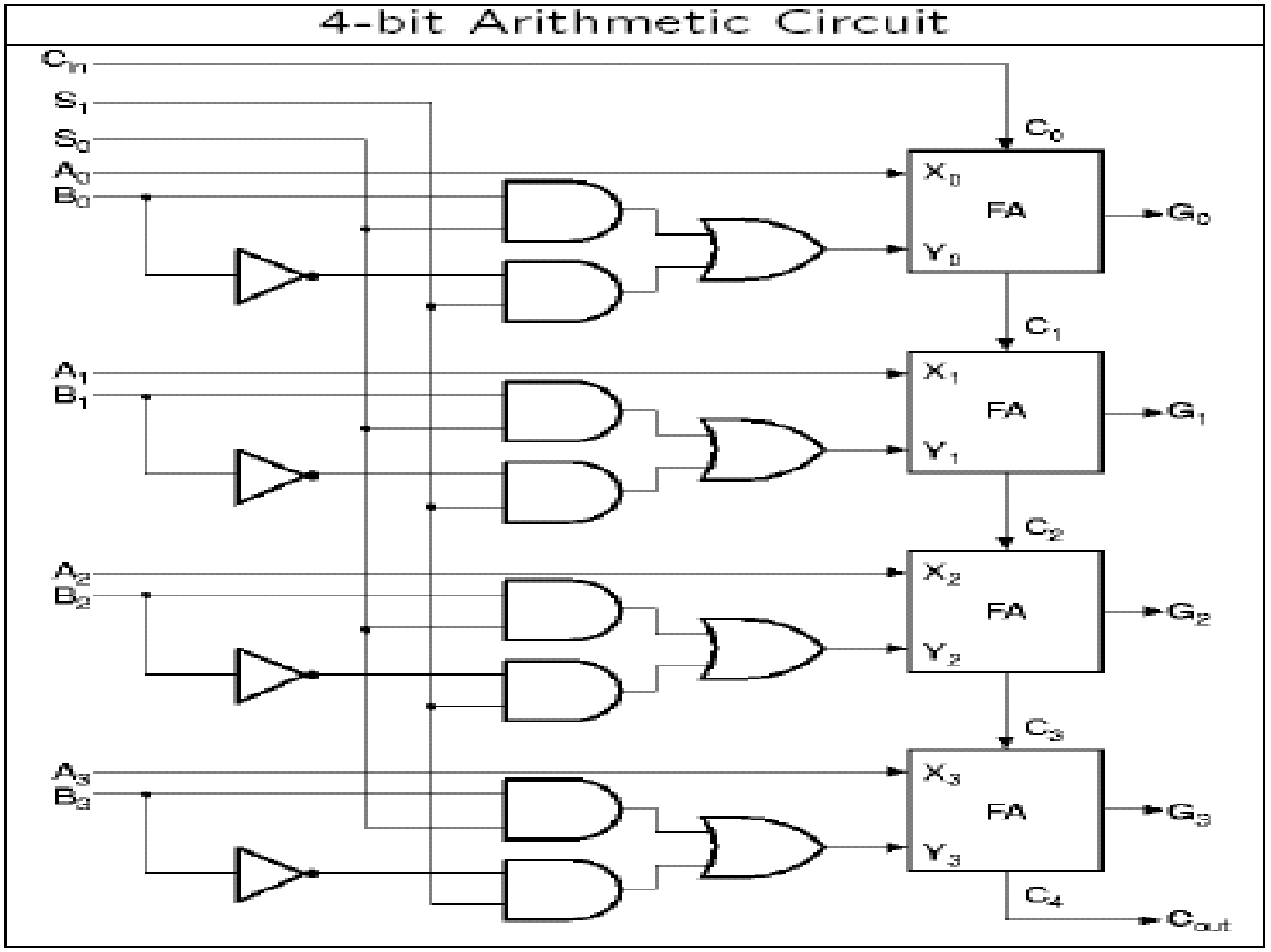
S_1	S_0	Output	Operation
0	0	$G = A \wedge B$	AND
0	1	$G = A \vee B$	OR
1	0	$G = A \oplus B$	XOR
1	1	$G = \overline{A}$	NOT

(b) Function Table

Arithmetic Circuit

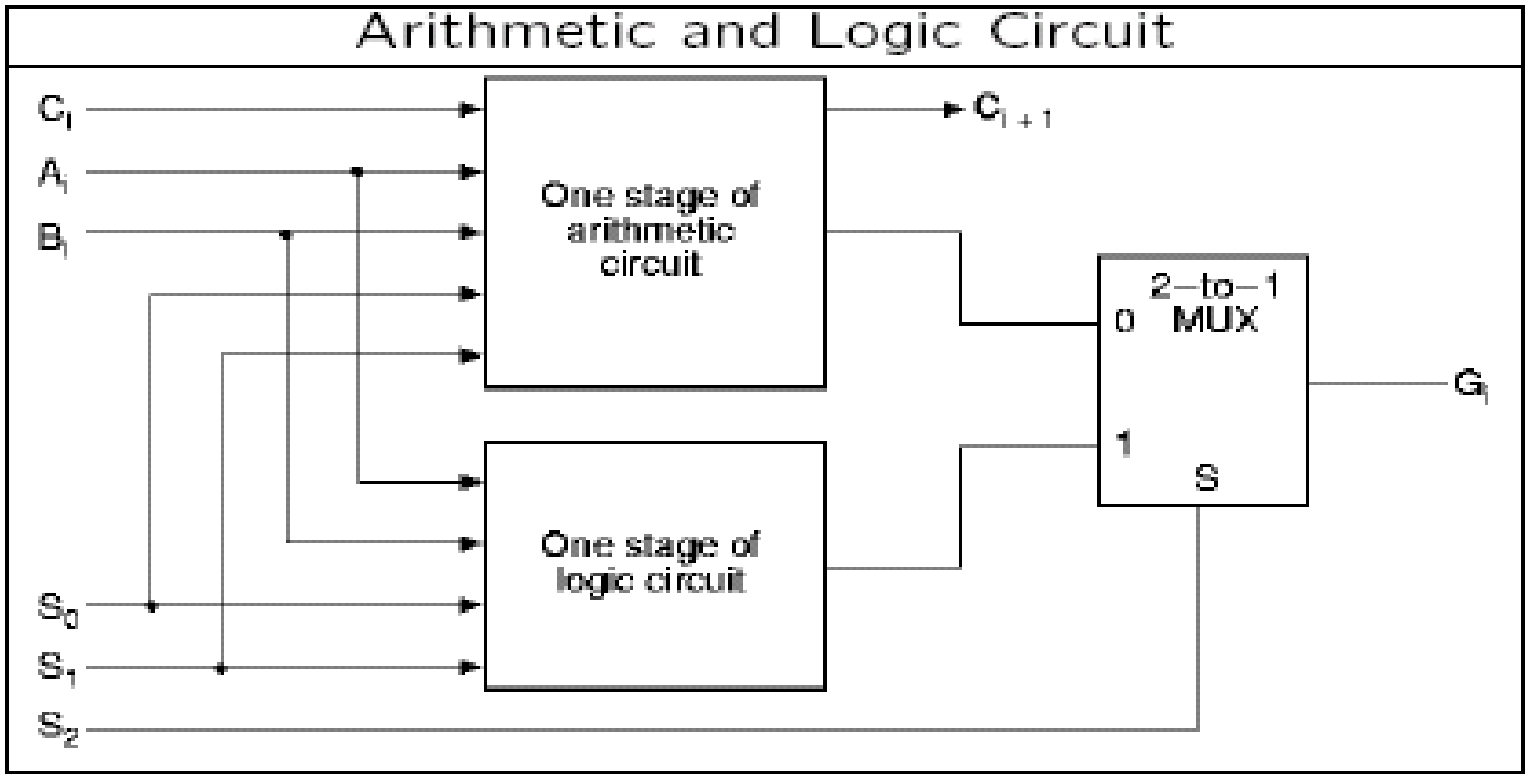


ALU
(continued)



ALU

(continued)

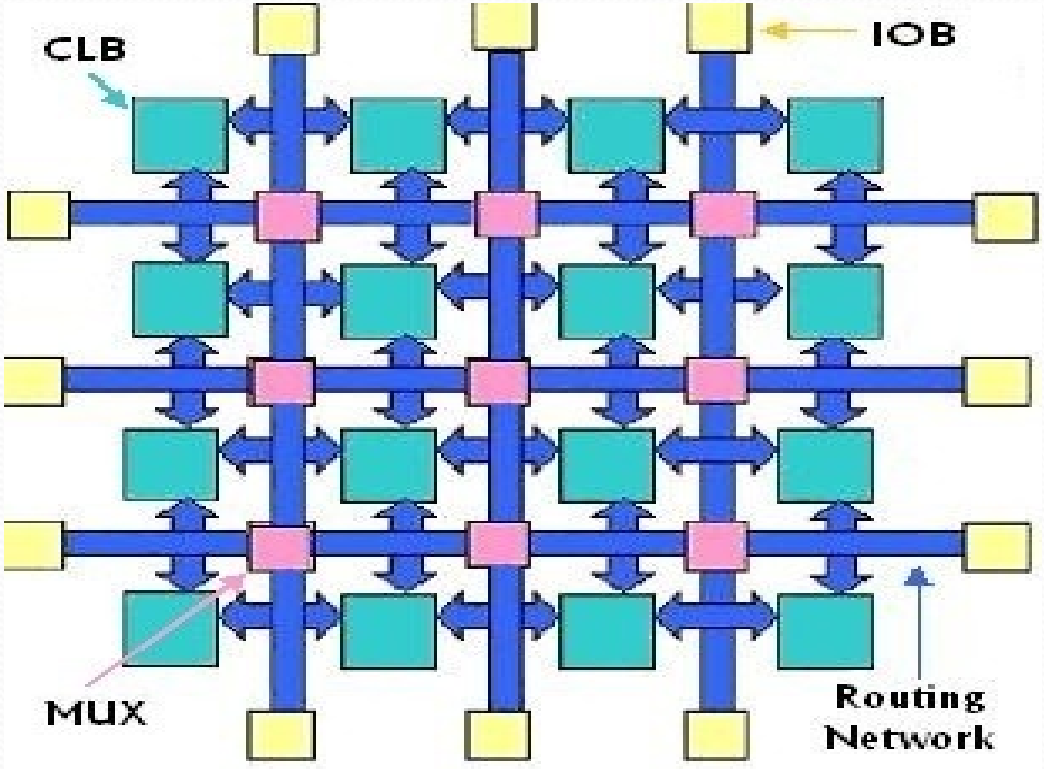


FPGA ARCHITECTURE

- FPGAs consist of three major resources:
- Configurable Logic blocks (CLB)
- Routing blocks (Programmable Interconnect)
- I/O blocks
- Other Resourcs:

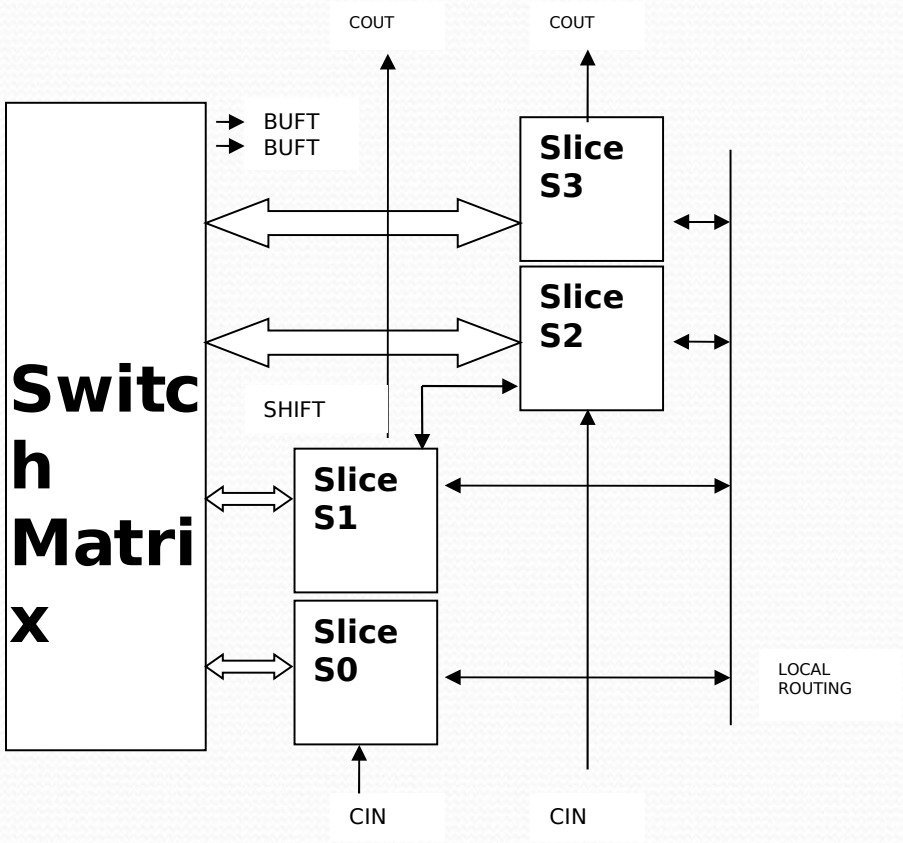
  1.Memory 2.Multipliers 3.Global clock buffers 4.Boundary scan logic

EXAMPLE ARCHITECTURE (XILINX VIRTEX II)



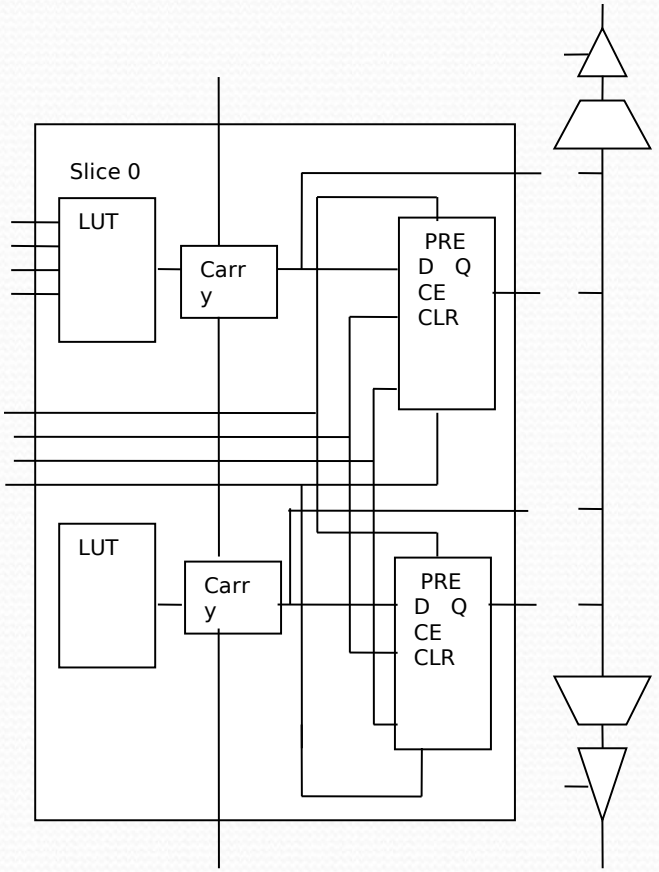
Structure of a CLB

- Each Virtex™-II CLB contains four slices
  - Local routing provides feedback between slices in the same CLB, and it provides routing to neighboring CLBs
  - A switch matrix provides access to general routing resources



Structure of a Slice

- Each slice has four outputs
  - Two registered outputs, two non-registered outputs
  - Two BUFTs associated with each CLB, accessible by all 16 CLB outputs
- Carry logic runs vertically, up only
  - Two independent carry chains per CLB

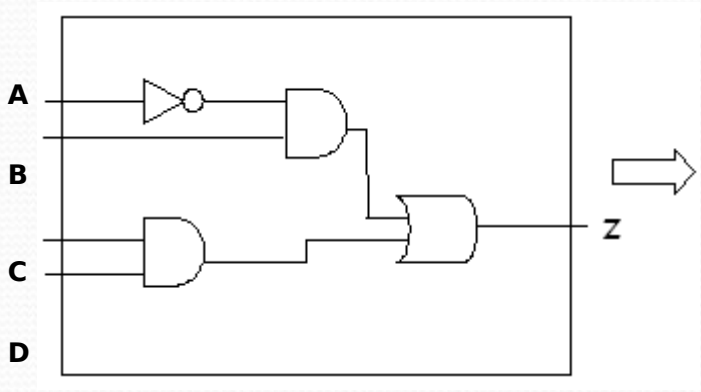


Look Up Table

- A LUT (Lookup table) is memory array
- A 4-input AND gate is replaced by a LUT that has four address inputs and one single bit output with 16 one bit locations
- Location 15 would have a logic value ‘1’ stored, all others would be zero
- LUT’s can be programmed and reprogrammed to change the logical function implemented

Look-Up Tables (LUTs) to implement Combinatorial Logic

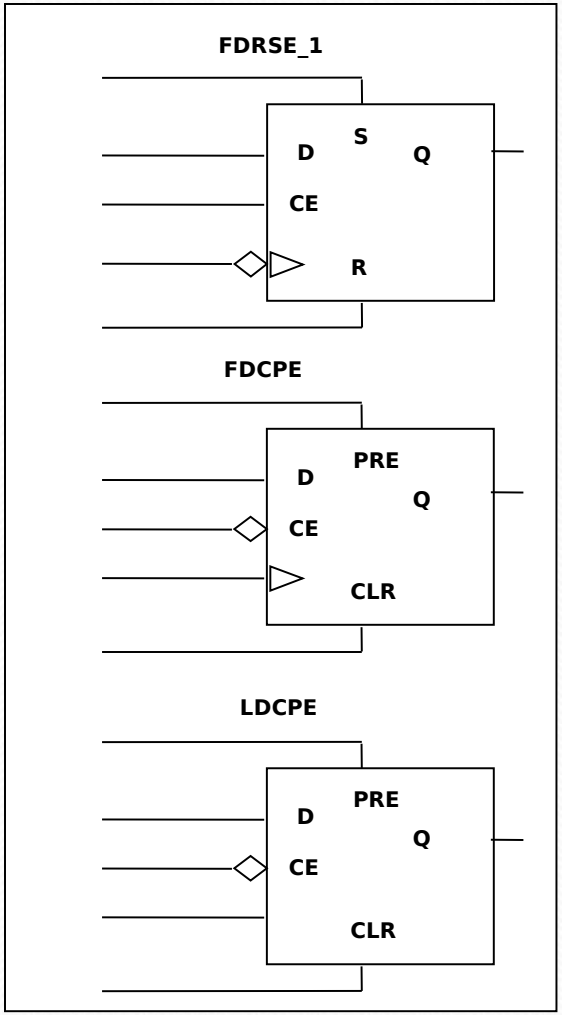
- Combinatorial logic is stored in Look-Up Tables (LUTs)
  - Also called Function Generators (FGs)
  - Capacity is limited by the number of inputs, not by complexity
- Delay through the LUT is constant



A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Flexible Sequential Elements

- Either flip-flops or latches
- Two in each slice; eight in each CLB
- Inputs come from LUTs or from an independent CLB input
- Separate set and reset controls
  - Can be synchronous or asynchronous
- All controls are shared within a slice
  - Control signals can be inverted locally within a slice

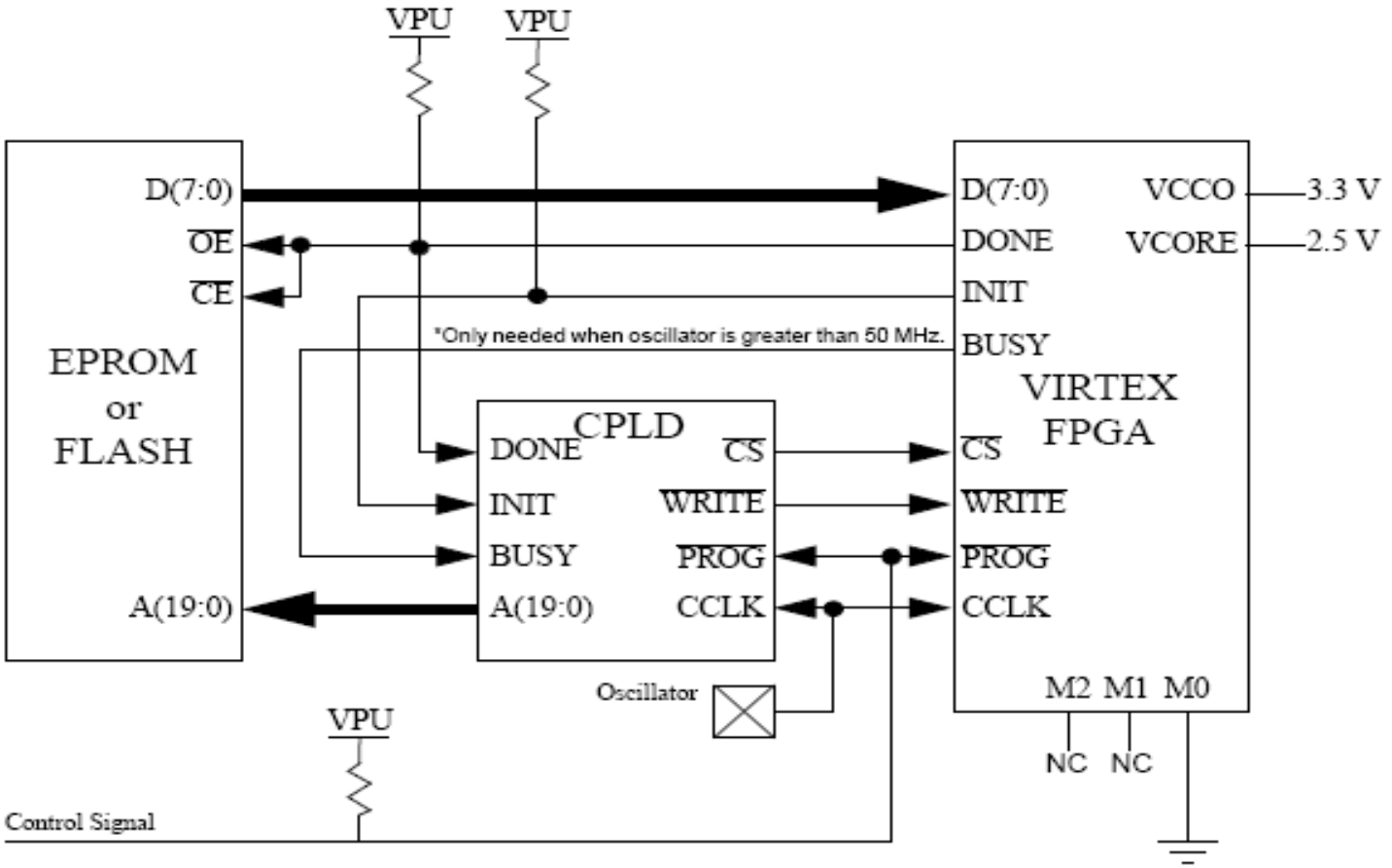


FPGA(Virtex XC9500) Configuration

- Configuration bit-streams are normally stored in EPROMs/EEPROMS/Flash memories because they are available in greater storage capacities than serial PROMs.
- A parallel PROM stores data as an addressed byte which is accessed by an address bus.
- There are different modes to configure the FPGAs.The SelectMAP configuration mode utilizes an 8 bit configuration bus and 7 control signals for synchronization and handshaking of data

Configuration using 'Slave MAP'

Mode



Signal Lines

- **D7...D0 :** The D7 through D0 pins comprise the 8 bit bidirectional data bus.
- **CCLK:** The CCLK is the configuration clock input signal used by the internal configuration logic.
- **PROG:** The PROG input signal resets the internal configuration logic and re-initializes the internal configuration memory.
- **DONE:** The DONE output indicates the completion of configuration and the beginning of the Startup sequence.
- **INIT:**The bidirectional open-drain INIT pin is used to hold off configuration initialization, and it indicates when CRC errors occur in the configuration data.
- **WRITE:** The WRITE input is a write strobe that must be asserted and held throughout the loading of data.
- **BUSY:** The BUSY open-drain output indicates whether the current byte is being loaded or ignored.
- **CS:** The CS input is the Chip Select signal used to enable the FPGA to be sensible of the SelectMAP interface.
- **M2 M1 M0:**
- The MODE pins M<2:0> must be set to indicate which configuration mode will be used. For SelectMAP configuration these must be <110>.

Configuration Process Steps :

1.Reset the Configuration Logic:

If configuration is to follow power-up, or a

recycling of the power source, then the configuration logic will become active already initialized.

However, if the FPGA is to be reconfigured without the recycling of power, then the PROG input must be asserted Low for at least 300ns to reset the configuration logic.

Continued…

2.Time-out Start of Configuration:

After the release of the PROG input, or the powering up of the FPGA, the INIT output stays Low while the internal configuration memory is automatically cleared.

Configuration may begin as soon as INIT has gone High.The FPGA may ignore as much as the first three clock cycles of data; however, a Virtex Bit-Stream is padded with 8 dummy bytes at the beginning of the data stream to account for this.

Continued…

3.Loading Configuration Data:

To begin configuration, the WRITE and CS inputs must be asserted Low and held. Each byte is loaded on the rising edge of CCLK. If BUSY was Low during the CCLK transition then the byte was accepted. If it was High then the byte was ignored and must be reloaded on the next clock cycle.

BUSY is a synchronous signal; Therefore, CCLK can not be suspended until the BUSY is de-asserted. For configuration clocks below 50MHz the BUSY can be ignored entirely, and removed from the interface design.

Continued…

4.End of Configuration:

The DONE output transitions High to indicate the end of configuration and the beginning of the startup sequence. During the startup sequence the D7...D0, WRITE, BUSY, INIT, and CS pins all become user IO. If any of these pins are used by the configured design, then any driving source used only for configuration purposes must be disabled so as not to contend. Depending on the startup options selected in BitGen, completing the startup phase will require as much as 8 CCLK cycles after DONE has transitioned High.

Partial Reconfiguration

1) Partial reconfiguration is the ability to reconfigure select areas of an FPGA anytime after its initial configuration. We can do this while the design is operational and the device is active (known as active partial reconfiguration) or when the device is inactive in shutdown mode (known as static partial reconfiguration).

Xilinx supports two basic styles of partial reconfiguration: module-based and difference-based. Module-based partial reconfiguration uses modular design concepts to reconfigure large blocks of logic. The distinct portions of the design to be reconfigured are known as reconfigurable modules.

Because specific properties and specific layout criteria must be met with respect to a reconfigurable module, any FPGA design intending to use partial reconfiguration must be planned and laid out with that in mind.

Continued…

3) Difference-based partial reconfiguration is a method of making small changes in an FPGA design, such as changing I/O standards, LUT equations, and block RAM content.

4) There are two supported ways to make such design changes: at the front end or the back end. Front-end changes can be done by HDL or schematic. We must re-synthesize and re-implement the design, creating a new placed and routed native circuit description (NCD) file. Backend modifications are made in FPGA Editor, a GUI tool within ISE™ software used to view/edit device layout and routing.

Continued…

5. Partial reconfiguration can be implemented by a basic controller to manage the reconfiguration of an FPGA. This could be in the form of an embedded or external processor.

Continued…

6) Xilinx offers a suite of processor solutions. The PicoBlaze™ and MicroBlaze™ soft-core processors both support the Spartan and Virtex families. The Virtex-II Pro FPGA embodies the hard-processor solution with the integration of an IBM PowerPC™ 405 32-bit RISC processor into the FPGA. Now, with the introduction of the Virtex-4 FX platform FPGA, Xilinx has increased processing power by introducing two PowerPC 405 processor cores in a single device (see Table 1).

Processor	Type of Processor	Supported Devices
PicoBlaze	Soft-Processor Core	Virtex Family Spartan Family CoolRunner-II CPLDs
MicroBlaze	Soft-Processor Core	Virtex Family Spartan Family
PowerPC	Hard Processor	Virtex-II Pro Virtex-4 FX

Table 1 — Xilinx processor solutions and supported devices

Benefits

Increased system performance: Although a portion of the design is being reconfigured, the rest of the system can continue to operate. There is no loss of performance or functionality with unaffected portions of a design – no down time. It also allows for multiple applications on a single FPGA. Configuration latency decreases substantially.

Continued…

3) The ability to change hardware. Xilinx FPGAs can be updated at any time, locally or remotely. Partial reconfiguration allows us to easily support, service, and update hardware in the field.

4) Hardware sharing. Because partial reconfiguration allows to run multiple applications on a single FPGA, hardware sharing is realized. Benefits include reduced device count, reduced power consumption, smaller boards, and overall lower costs.

Continued…

5) Shorter reconfiguration times . Configuration time is directly proportional to the size of the configuration bit stream. Partial reconfiguration allows us to make small modifications without having to reconfigure the entire device. By changing only portions of the bitstream – as opposed to reconfiguring the entire device – the total reconfiguration time is shorter.

Applications

1) Partial reconfiguration is the cornerstone for power-efficient, cost effective software-defined radios (SDRs). Through the JTRS Program, SDRs are becoming a reality for the defense industries as an effective and necessary tool for communication. SDRs satisfy the JTRS standard by having both a software-reprogrammable operating environment and the ability to support multiple channels and networks simultaneously.

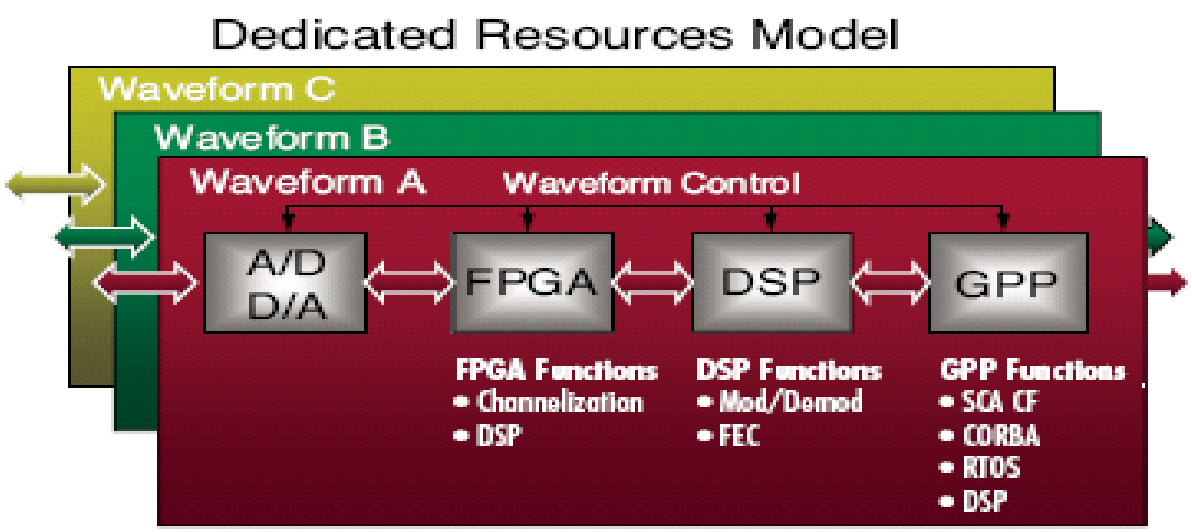


Figure 1 – Dedicated resources model: three-channel SDR modem

2) Figure 1 shows a three-channel SDR modem supporting a Software Communications Architecture Core Framework (SCA CF), as mandated for JTRS. Current implementations of SCA enabled SDR modems with multiple channels require multiple sets of processing resources and a dedicated set of hardware for each channel. The more channels SDR must support, the more dedicated resources needed. This aversely affects space, weight, power consumption, and cost.

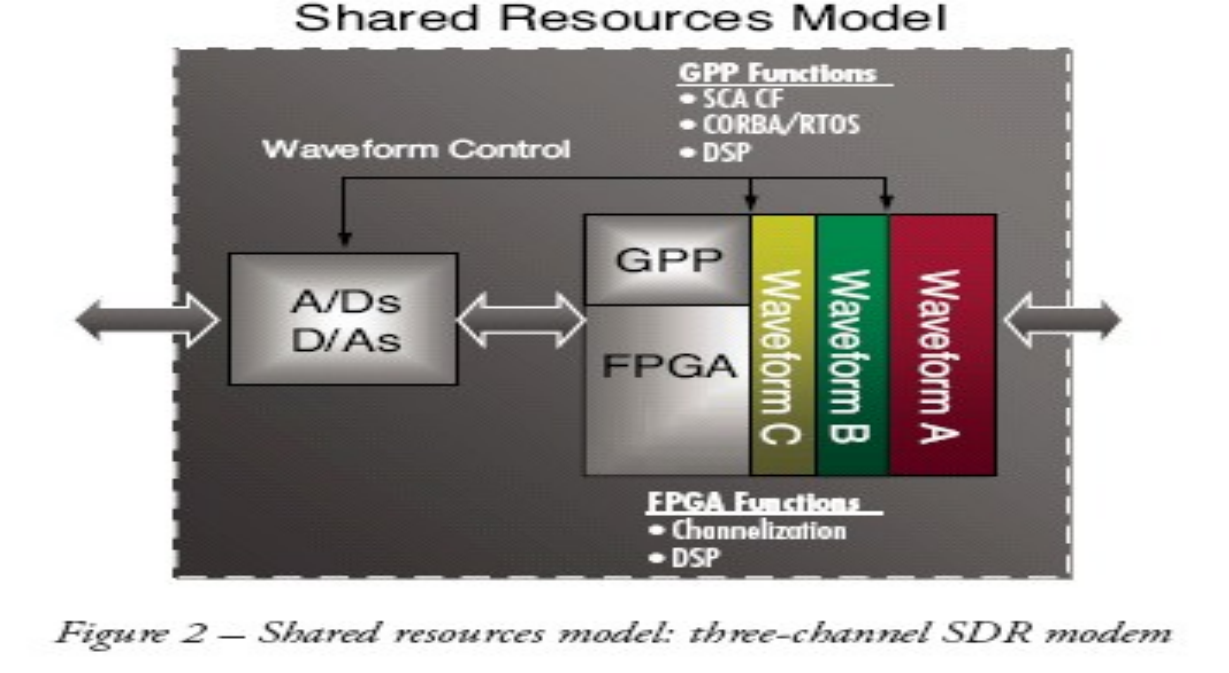


Figure 2 – Shared resources model: three-channel SDR modem

3) With partial reconfiguration, the ability to implement an SDR modem using shared resources is realized, as shown in Figure 2. A shared resources model enabled by partial reconfiguration of an FPGA to support multiple waveforms can be supported by the SCA as mandated by JTRS. FPGA implementations of SDR, with partial reconfiguration, results in effective use of resources, lower power consumption, and extensive cost savings.

Continued…

4) Partial reconfiguration can also be used in many other applications. Another example is in mitigation and recovery from single-event upsets (SEU). In-orbit, space-based, and extra-terrestrial applications have a high probability of experiencing SEUs. By performing partial reconfiguration, in conjunction with read back, a system can detect and repair SEUs in the configuration memory without disrupting its operations or completely reconfiguring the FPGA. (Read back is the process of reading the internal configuration memory data to verify that current configuration data is correct.)

Capabilities & Benefits

The capabilities and benefits offered by partial reconfiguration reach across many industries and applications. Leverage partial reconfiguration by using Xilinx FPGAs in our next design, the only truly partially reconfigurable devices. We can take advantage of any of its benefits, from remote hardware upgrading to on-chip hardware sharing, and give your designs the reconfigurability advantage.

FPGA Design Cycle

- Define a new project and enter the design using VHDL, Verilog or AHDL languages. Design can also be entered using Schematic diagrams that can be translated to any HDL

- Compile and simulate the design. Find and fix timing violations. Get power consumption estimates and perform synthesis

- Download the design to the target FPGA board.

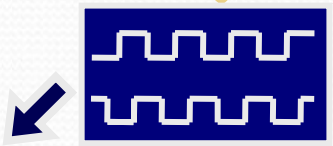
Downloading the Design

- Once the FPGA based design is verified, the program can be downloaded to an FPGA chip using the design tool

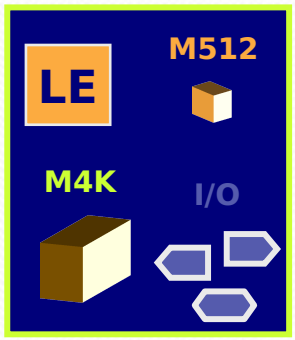
- Designs can be downloaded using parallel port or USB or Ethernet or JTAG port

FPGA Design Flow

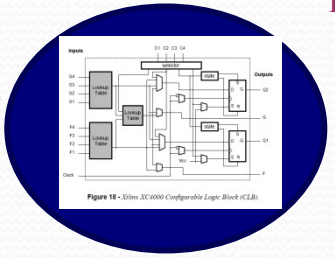
Design Entry/RTL Coding
- Behavioral or Structural Description of Design



RTL Simulation
- Functional Simulation
- Verify Logic Model & Data Flow (No Timing Delays)



Synthesis
- Translate Design into Device Specific Primitives
- Optimization to Meet Required Area & Performance Constraints



Place & Route
- Map Primitives to Specific Locations inside Target Technology with Reference to Area & Performance Constraints
- Specify Routing Resources to Be Used

A new Computing Paradigm

What is Configurable Computing?

- Programming/Configuring highly tuned hardware circuits to provide the functionality needed for specific task

- Program or Structure the hardware to directly implement the native operations required by the applications and also to exploit the concurrency inherent in the computation

Reconfigurable Computing

- It is the manipulation of the logic within a "programmable hardware" device at run time.
- the design of the hardware will change in response to the demands placed upon the system while it is running.

Advantages :

i) Speed of hardware

ii) Flexibility of Software

Advantages of Reconfigurable Computing

- Ability to execute larger hardware designs with fewer gates and to realize the flexibility of software based solutions while retaining the execution speed of more traditional hardware based approach.
- Lower system cost
- Reduced time to market
- Incremental design flow
- Effective hardware utilization

Some of the important features ....

- ON-THE- FLY RECONFIGURIBILITY
- PARTIAL RECONFIGURIBILITY

FPGA – The Basic Building Block for Re-Configurable Computing

FPGA(Field-Programmable Gate Arrays)

- Is a basic building block for Reconfigurable Computing
- Circuits that can be programmed by means of bit-streams that completely specifies the logical functions and connectivity to be implemented
- Traditionally , not been viewed as effective computing device due to their "**High power requirements**", "**Low clock speed**" & "**Long Programming/Configuration time**"

State- of- the-art reconfigurable FPGAs (Xilinx Virtex 2/4) contd.

- There is always a high probability that each CLB may not be available for implementation of a particular function, due to insufficient routing logic and buses
  - 60-80 % Area only for Routing !!!!
  - in effect fewer CLBs will be available for logic implementations than actually present.
  - Depending on the implementation there will always be wastage of the re-configurable fabric present in the FPGA

Therefore, Performance and flexibility is achieved at a very high silicon cost!!!

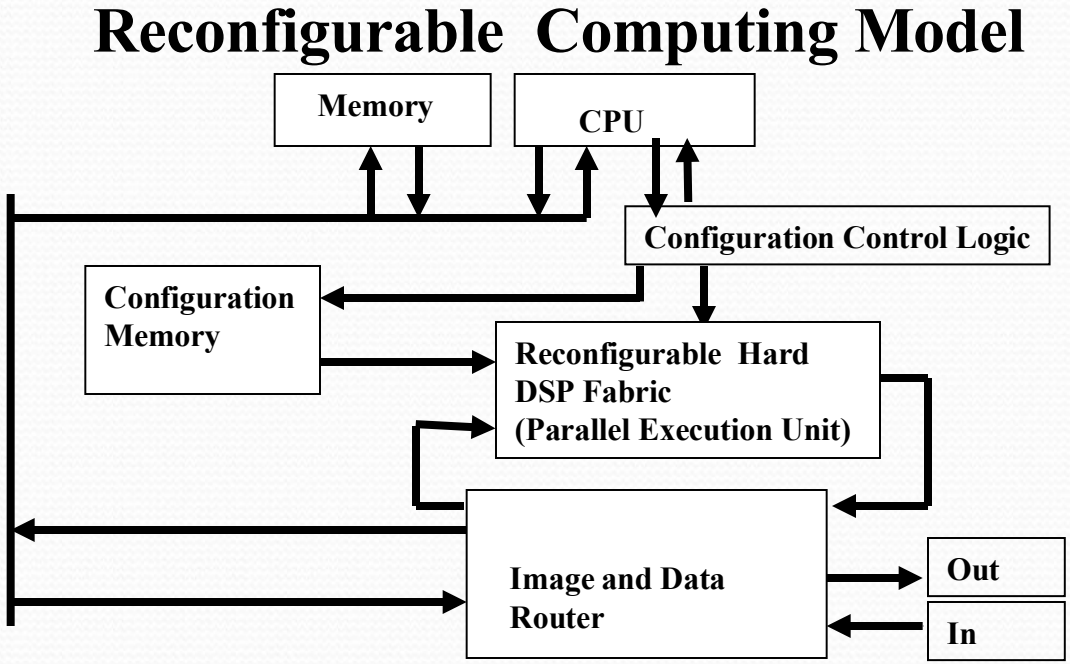
An alternative to conventional DSP Processor/ ASIC /FPGA

- **There is a need to innovate a re-configurable parallel architecture for DSP applications which eliminates the drawbacks of the FPGAs and ASICs by offering**
  - **A balance between flexibility & reconfiguration latency**
  - **Higher performance.**

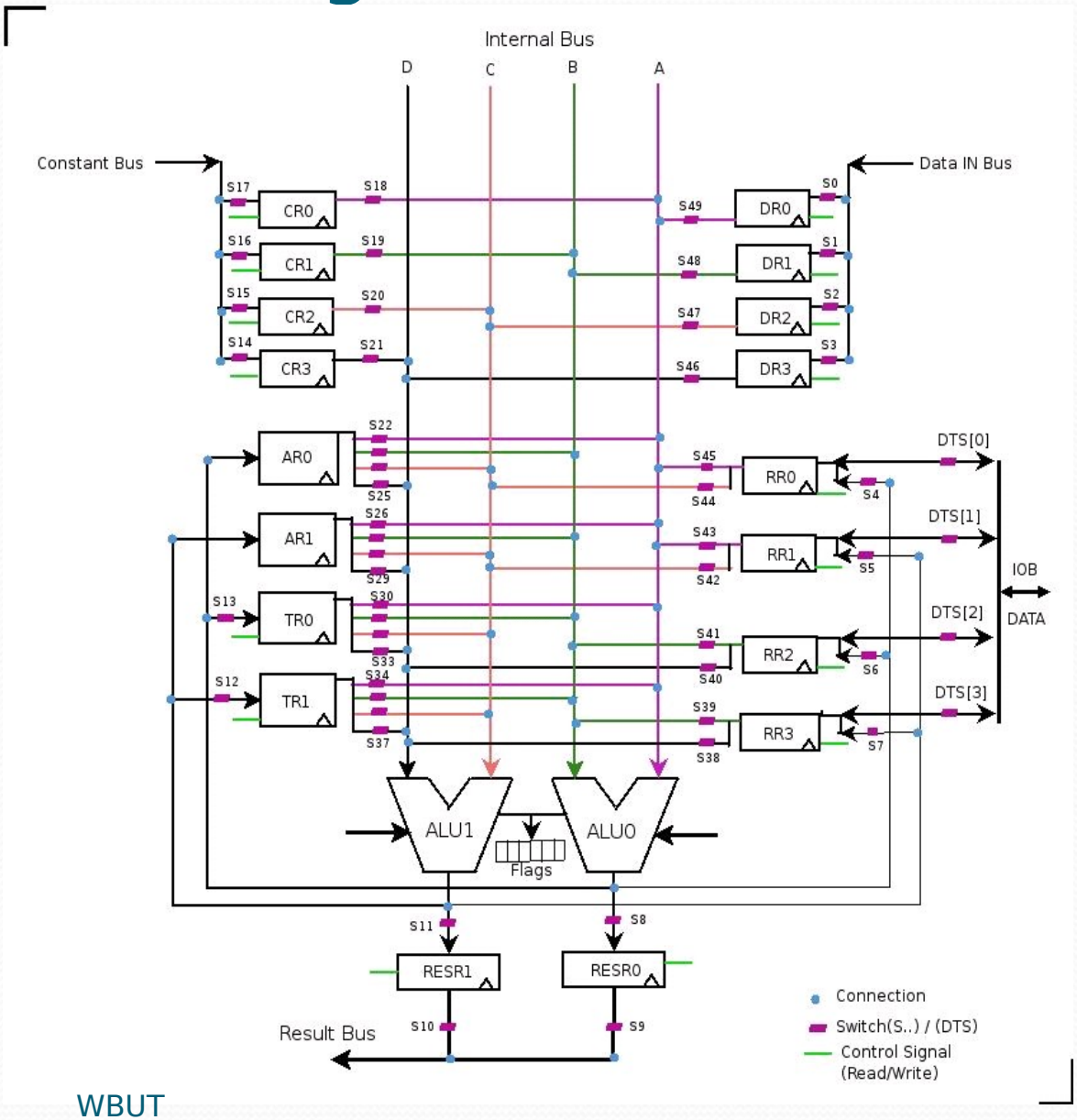
HardDSP™ architecture

[A trade mark of ESP microDesign Inc.]

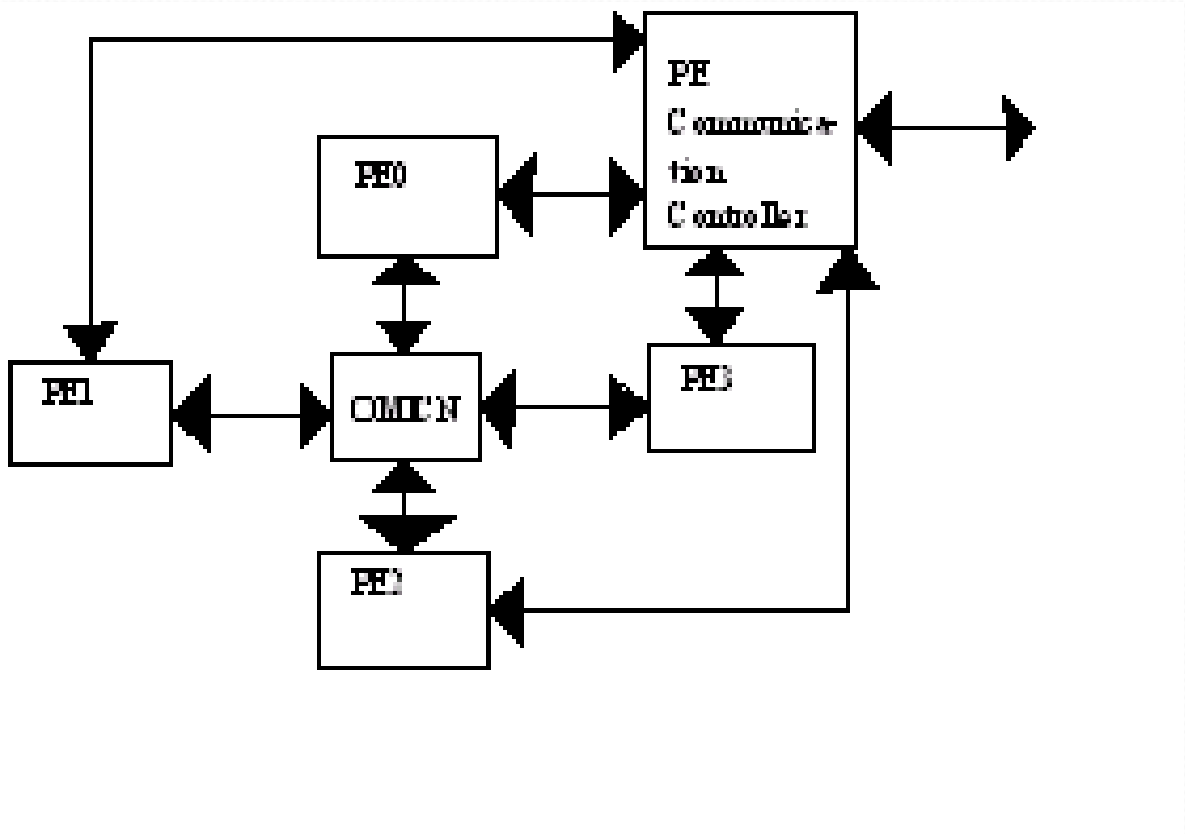
- **Parallel hard-logic DSP under CPU control**
- **Direct hardware logic execution**
- **Parallel execution engines**



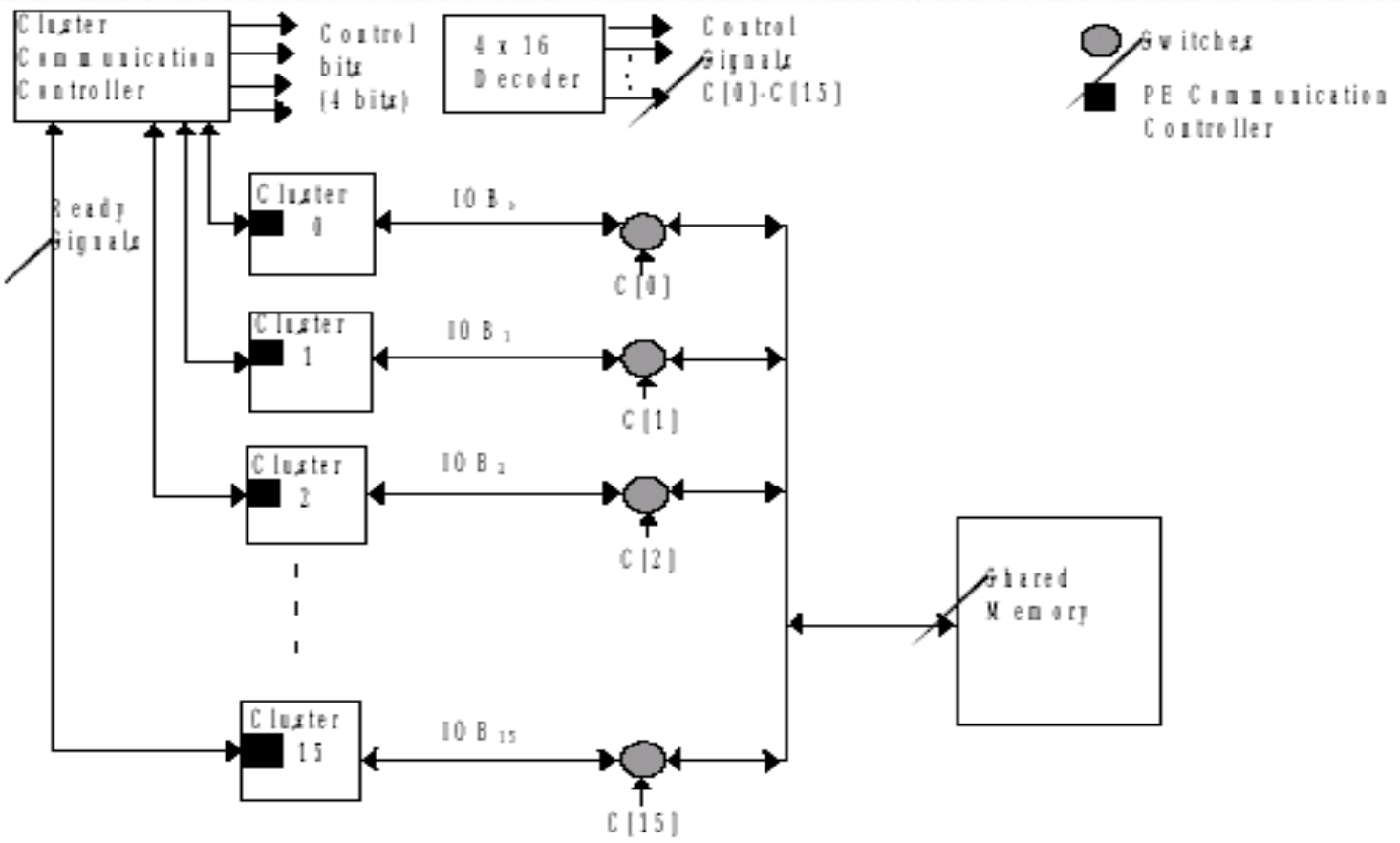
Basic Processing Elements:



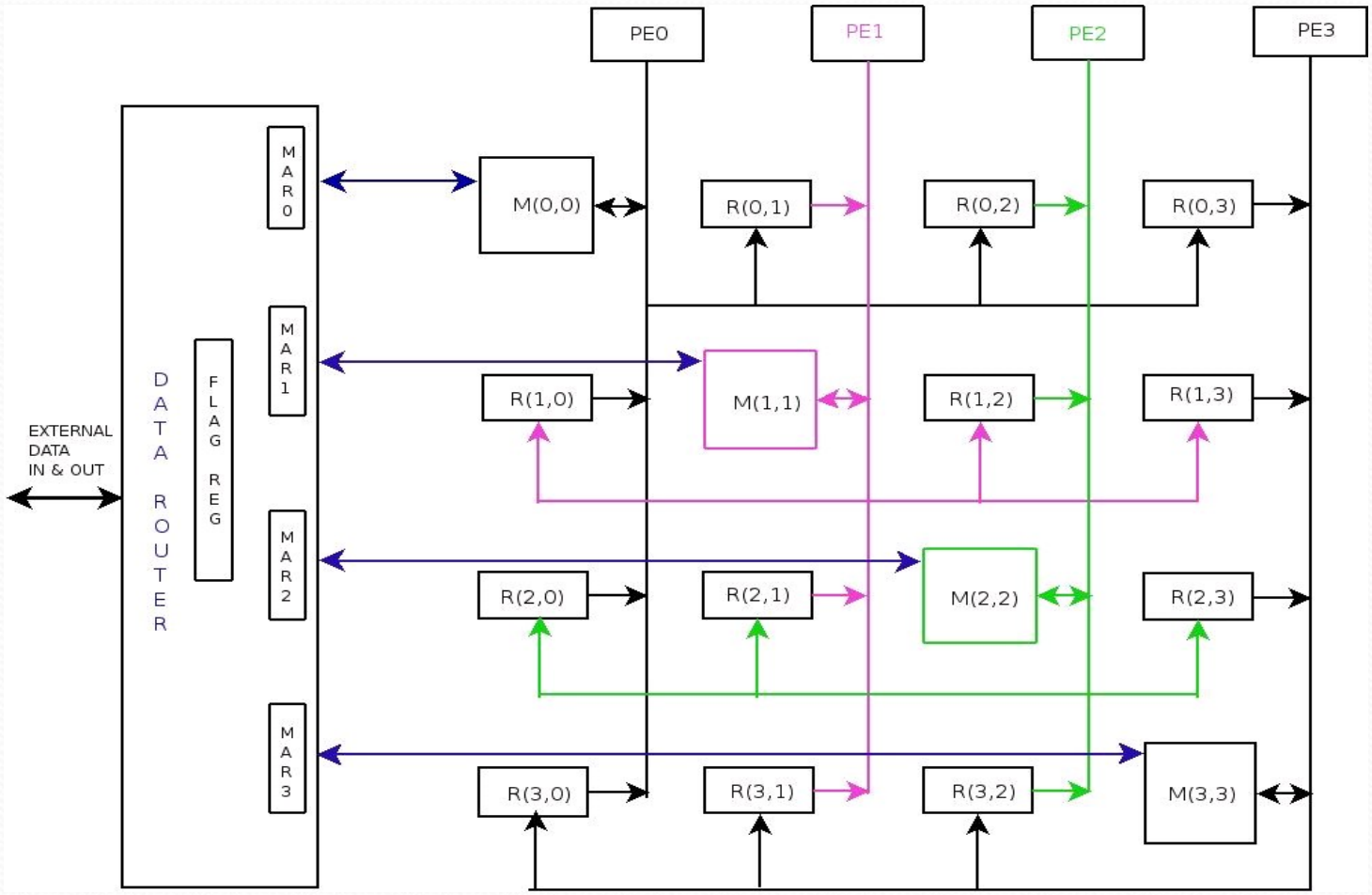
Organization of a Cluster:



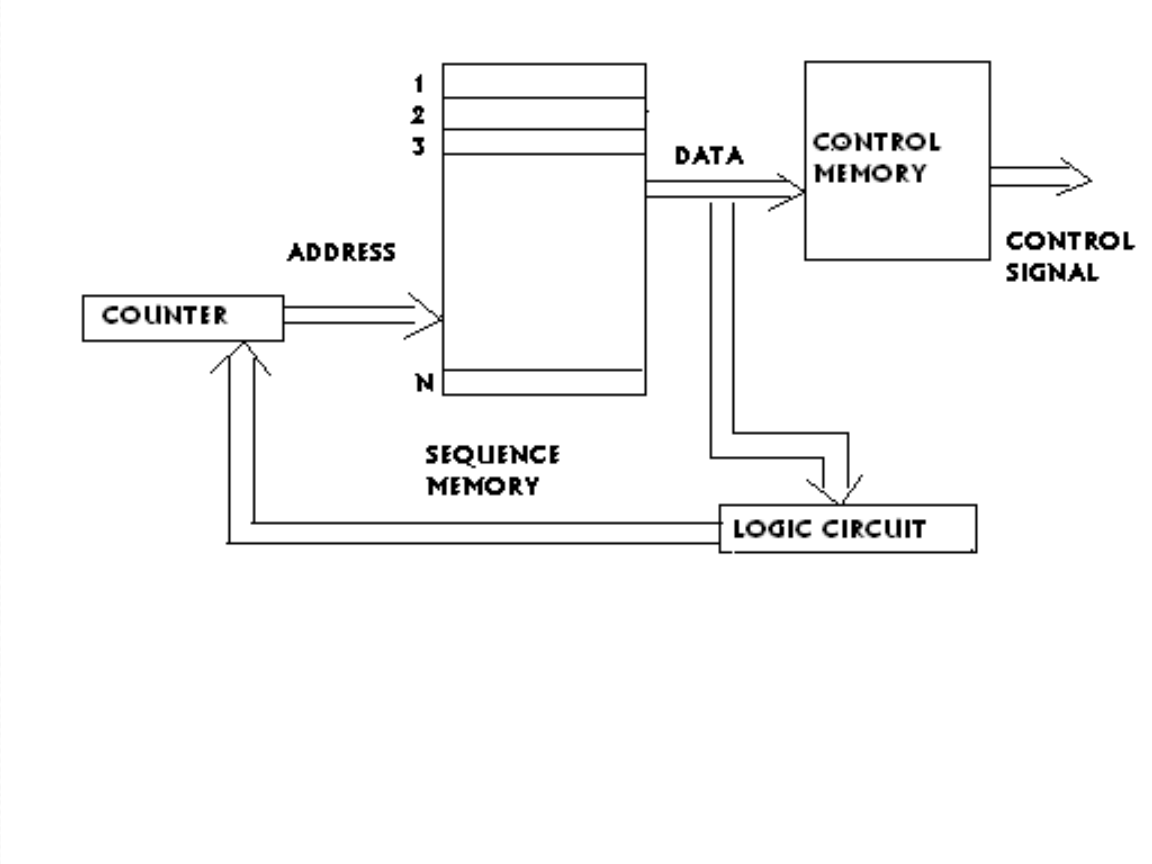
Inter-cluster communication:



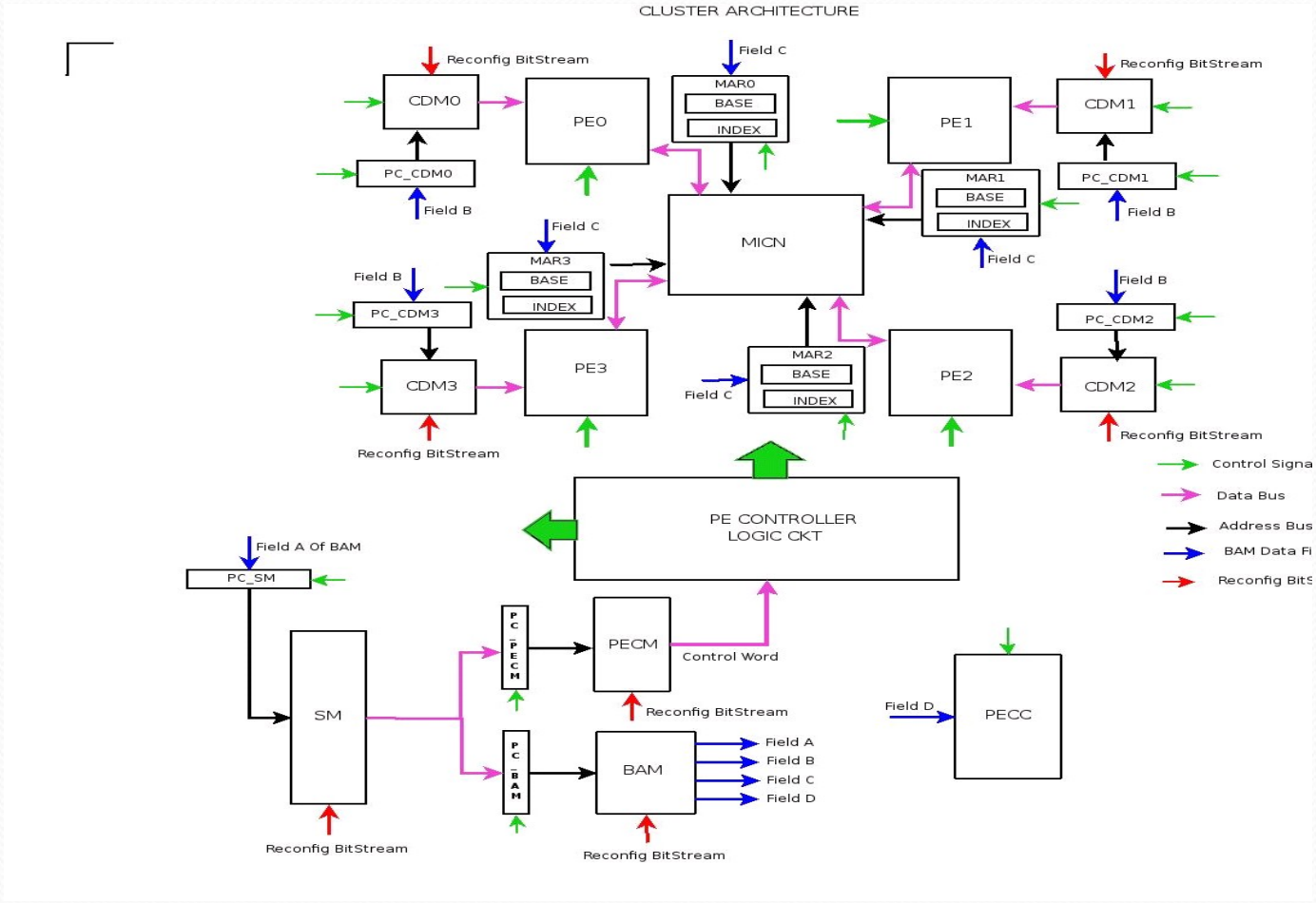
Intra cluster communication:



DISTRIBUTED MICRO CONTROL UNIT:



A cluster with PE controller:



Implementation Platform for Validating the architecture

Xilinx Multimedia Demo board

FPGA: Xilinx Virtex 2V2000

2 Million Gates !!!

Simulation and Debug tools Used:

Xilinx ISE 6.2i

ModelSim XE

CONCLUSION

- A new Reconfig DSP Processor has been Architected which provides:
  - a programmable computational mechanism balanced between ASIC and FPGA
  - enhances re-configurability of the hardware and offers fine grain parallel computation at hardware speed
  - achieves the performance of an ASIC and flexibility of a FPGA.
  - Achievement of fine-grain parallelism due to micro level concurrency
  - Low latency “Dynamic Re-configuration” to any type of parallel/pipelined architecture like SIMD, MIMD , Systolic etc.
  - Achievement of a higher degree of scalability by increasing the number of PEs for realizing complex DSP functions.

REFERENCES

- [1] J. Siegal, et al, "PASM: partition able SIMD/MIMD system for H image processing and pattern recognition", *IEEE Trans. Computer,* vol.C30, no. 12, pp.934-947, Dec 1981.
- [2] B.G. Lee, "A new Algorithm to compute the discrete cosine transforms", IEEE Trans on Acoustics, speech and signal Processing, vol. ASSP-32, pp.1243- 1245, Dec.1984.
- [3] G. Li and B.W. Wah, "The design of optimal systolic arrays," IEEE Trans. On Computers, vol.34, no.1, pp.67-77, Jan.1985.
- [4] Hungwen Li and Quentin F. Stout, "Reconfigurable SIMD Massively parallel Computers", Proc. IEEE, Vol79, no.4, April 1991,pp. 429-443 .
- [5] K. Mayer-Patel and L.A. Rowe, "Exploiting temporal parallelism for software-only video effects processing", Proc. of the sixth ACM international conference on Multimedia", Bristol, U.K., pp.161-169, September, 1998.
- [6] K.K. Parhi, "VLSI Digital signal Processing Systems", A Wiley-Inter science Publication, 1999.
- [7] A. Sinha, et al., "Re-configurable Parallel Architecture for Signal/Image Processing Applications", Proc. Embedded Systems Conference, Stuttgart, Germany, October 9th -11th, 2001.
- [8] Xilinx, "Introduction and overview", Virtex-II Pro Platform FPGAs, March 9th, 2004.
- [9] V. Baumgarte, F. May, A. Nuckel, M. Vorbach and M. Weinhardt, "PACT XPP – A Self Reconfigurable Data Processing Architecture", The Journal of Supercomputing, vol. 26, Issue 2, pp. 167-184, 2003.
- [10] Pavel Sinha, Dharuba Basu , Amitabha Sinha " A novel Architecture of a Reconfigurable Parallel DSP Processor ", Proc. The 3rd Int. IEEE Northwest workshop on circuits and systems, June 19-22, 2005, Qubec, Canada.
- [11] Pavel Sinha , Dhruba Basu , Amitabha Sinha " A Reconfigurable "SFMD Architecture " for a Class of Signal Processing Applications", Proc. IEEE 7th Emerging Technologies Workshop : " Circuits and Systems fir 4G mobile wireless Communications", June 23-24,2005, St. Petersburg, Russia.

THANK YOU!!