Overview

This set of notes introduces many of the features available in the FPGAs of today.

The majority use SRAM based configuration cells, which allows fast reconfiguation.

- Allows new design ideas to be quickly implemented and tested.
- Allows evolving standards and protocols to be accommodated.
- Allows the FPGA to carry out multiple functions, such as self-test or board/system test at power-up and something else later.

Another advantage of using SRAM is that SRAM technology is very heavily invested in, and therefore, FPGA companies can leverage this.

Also, the same process used to fabricate the logic gates on the FPGA is used to fabricate SRAM -- no special processing steps are needed.

The drawback of SRAM is volatility, which is overcome with a special external memory device or microprocessor (costly either way).

Security

Another concern with SRAM-based FPGAs is that it can be difficult to protect your intellectual property.

The configuration file lives somewhere on disk, in memory, etc.

At this point, there is no commercially available tool that can read the contents of the configuration file and generate a schematic or netlist.

However, it is not rocket science to accomplish this (although it's not trivial either).

Also, bear in mind, there are **reverse-engineering** companies all over the world.

In many cases, it is not even necessary for them to understand the details of your design.

They can use a *bed-of-nails* tester to extract a complete netlist of your board and then copy the FPGA configuration file from its boot PROM.

Security

Today, there are SRAM-based FPGAs that support the concept of **bitstream encryption**.

Here, the configuration file is encrypted before being stored in the PROM.

The encryption key is loaded into a special SRAM-based register in the FPGA via its JTAG port.

The key is used to *decrypt* the bitstream as it is loaded into the FPGA.

The drawback is that a battery backup is needed to maintain the contents of the encryption key register.

Antifuse-based devices

Antifuse-based FPGAs are programmed *off-line* using a special chip programmer.

Advantages include:

- Configuration is retained during power cycle (non-volatile).

Antifuse-Based Chips

Advantages (cont.)

- They don't require an external memory to store their configuration data, which saves on board cost and real estate.

- Their *interconnect structure* is "rad hard", or relatively immune to the effects of radiation.

  Note that any FFs in these FPGAs remain sensitive to random flips due to radiation, and must be protected using *triple redundancy* design.

- Lastly, the configuration data is buried deep inside them.

  By default, it **is** possible for the chip programmer to read this data out.

  This is necessary during programming in order to know when to move on to the next antifuse, and afterwards for verification.

  Once programmed, it is possible to *grow* a special security antifuse that prevents the presence or absence of antifuses to be read out.

Antifuse-Based Chips

Even "popping the cap and inspecting" does not reveal the programming.

Other claims, that are not fully justified:

- Lower power consumption, standby and active
- Smaller delays, because antifuse is smaller than SRAM cell

Unfortunately, the technology used to fabricate antifuse FPGAs is one or more generations behind the technology used for SRAM versions.

Therefore, these claims are not realistic.

The main disadvantage, of course, is the one-time-programming (OTP).

Not much use in development and prototyping environments.

EEPROM/FLASH-based devices

Similar to SRAM versions, i.e., configuration cells are connected together in a long *shift-register-style* chain (scan chain).

Programming done off-line, some allow in-system programming (ISP).

EEPROM/FLASH-Based Chips

However, **programming time** is about 3 times longer than SRAM version.

For security, some use a *multibit key* (50 to several hundred bits).

Once programmed, you load a user-defined key to secure your data.

Once the key is loaded, reading and writing require that your key be loaded via the JTAG port.

Two-transistor EEPROM and FLASH cells are about 2.5 times larger than the one-transistor EPROM cells (no one currently makes an EPROM version).

They are still much smaller than SRAM, reducing area and delay.

One *drawback* is that these devices require about 5 additional process steps beyond the standard CMOS process.

Similar to antifuse, this causes a lag of these devices, technology wise.

Also, they have high static power because of the internal pull-up resistors.

Hybrid FLASH-SRAM Chips

Here, each configuration element is formed from a combination of FLASH (or EEPROM) and an SRAM cell.

The FLASH elements are *pre*programmed, whose contents, when powered up, are massively transferred to the SRAM cells.

Provides *non-volatility* while powered off and speedy reconfigurability when powered up.

You should become familiar with the information in the following table.

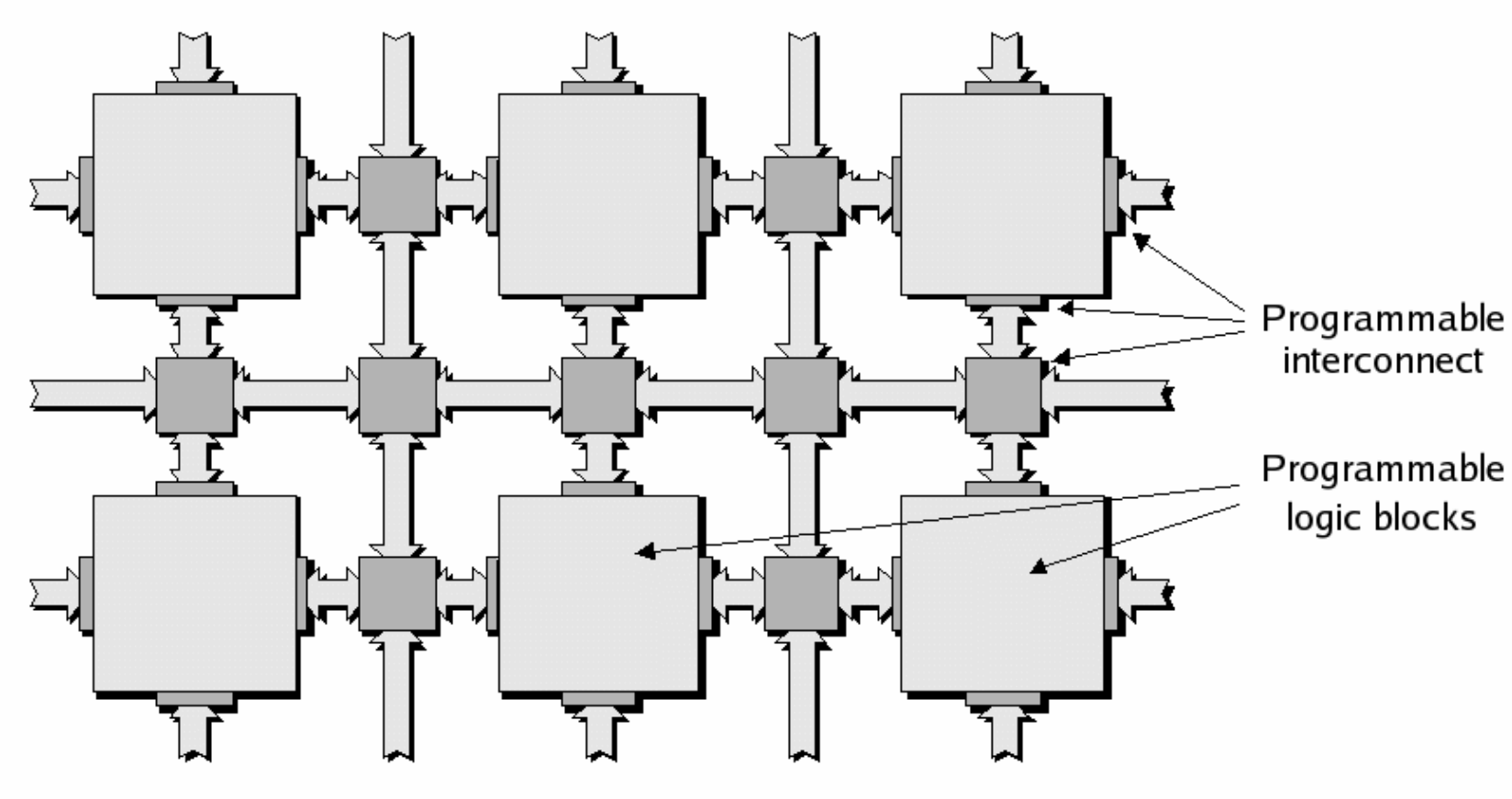
Summary

Feature	SRAM	Antifuse	E2PROM / FLASH
Technology node	State-of-the-art	One or more generations behind	One or more generations behind
Reprogrammable	Yes (in system)	No	Yes (in-system or offline)
Reprogramming speed (inc. erasing)	Fast	----	3x slower than SRAM
Volatile (must be programmed on power-up)	Yes	No	No (but can be if required)
Requires external configuration file	Yes	No	No
Good for prototyping	Yes (very good)	No	Yes (reasonable)
Instant-on	No	Yes	Yes
IP Security	Acceptable (especially when using bitstream encryption)	Very Good	Very Good
Size of configuration cell	Large (six transistors)	Very small	Medium-small (two transistors)
Power consumption	Medium	Low	Medium
Rad Hard	No	Yes	Not really

The Design Warrior's Guide to FPGAs,
ISBN 0750676043,
Copyright(C) 2004 Mentor Graphics Corp

Fine-, Medium-, and Course-Grained Architectures

Basic architecture consists of a large number of PLB islands embedded in a sea of programmable interconnect.



The Design Warrior's Guide to FPGAs,
ISBN 0750676043,
Copyright(C) 2004 Mentor Graphics Corp

Fine-, Medium-, and Course-Grained Architectures

"Level of granularity", when used in the context of an FPGA refers to the complexity of the PLB.

The PLBs of **fine-grained** architectures can only implement simple functions, e.g., 3-input logic gate or storage element.

Good for glue logic, state machines, systolic algorithms (massively parallel), and traditional logic synthesis.

The PLBs of **medium-grained** architectures include more logic and more functionality, i.e., a 4-input LUTs, 4 MUXs, 4 D-FFs + fast carry logic.

This helps with the interconnect problem, e.g., more "compute power" per wire dedicated for interconnections.

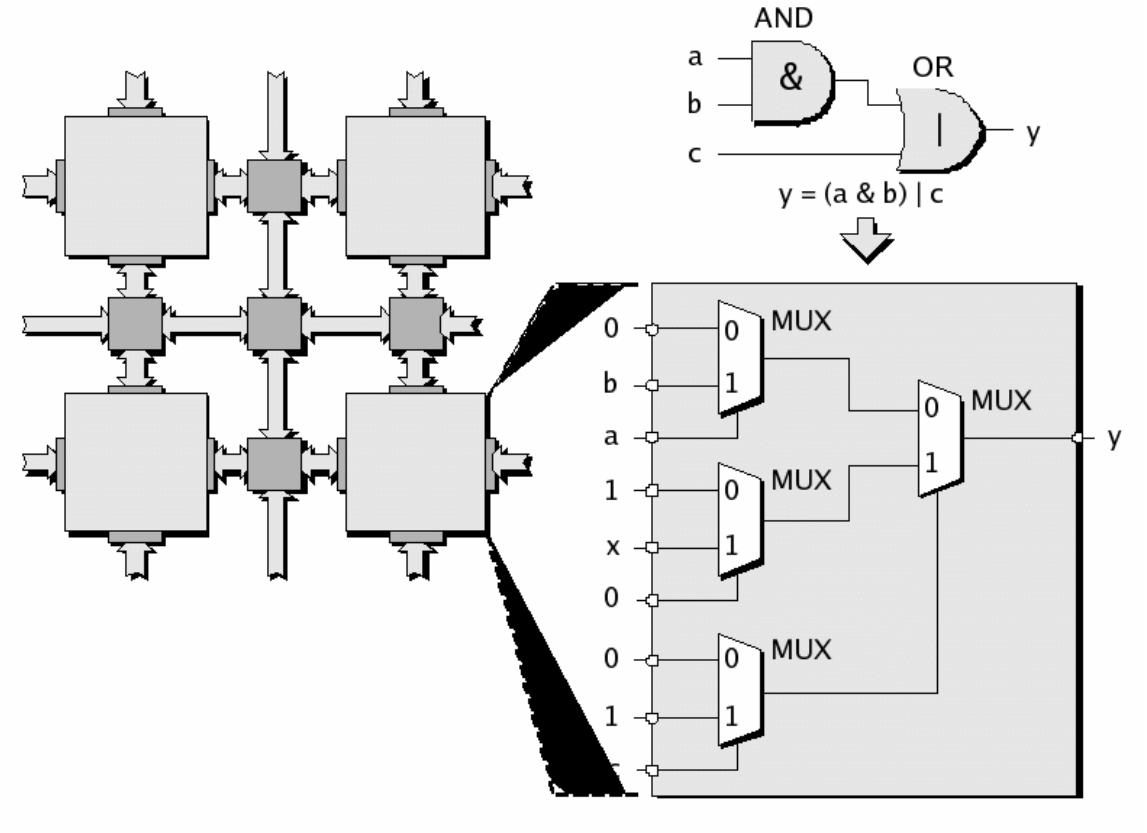
Large-grained architectures incorporate FFT engines and microprocessor cores.

Fine-grained architectures of the mid-90's gave way to the *medium grained* architectures.

MUX- vs. LUT-based Logic Blocks

There are 2 basic flavors of PLBs for *medium-grained* architectures, *multiplexer* (MUX) and *lookup table* (LUT).

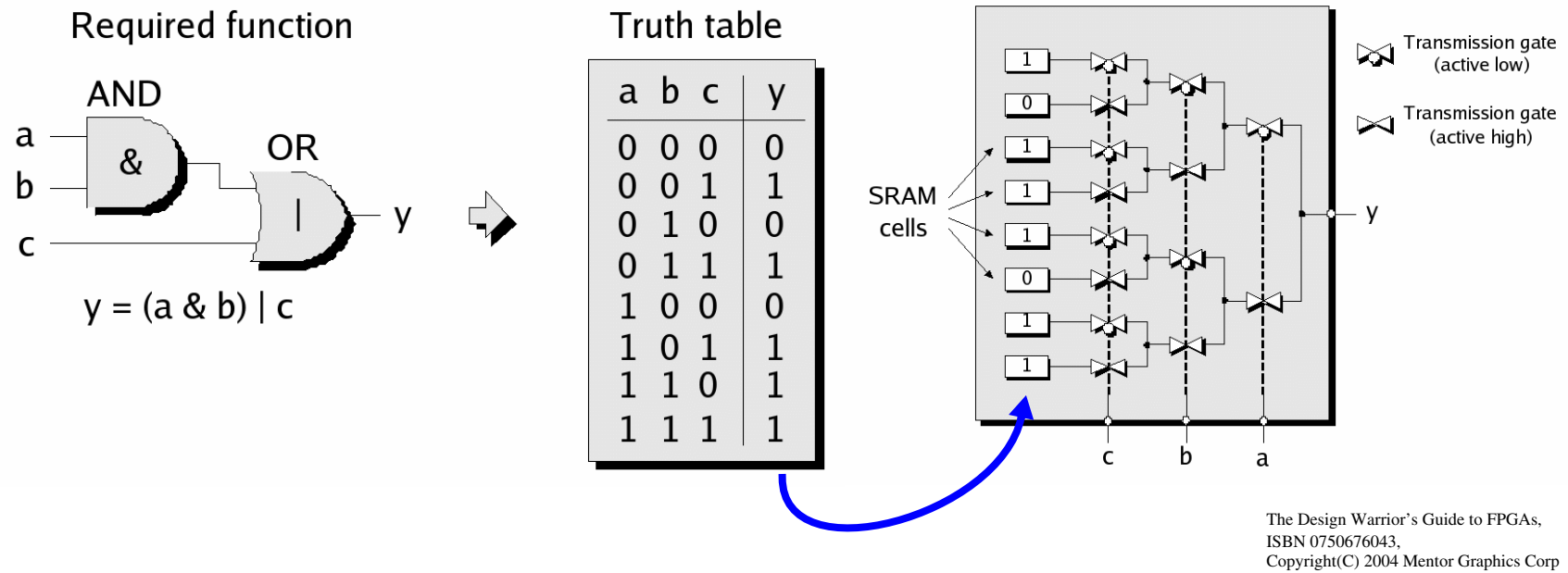
In the MUX-based version, each input can be programmed with a logic 0, 1, or the true or inverted version of a variable.



The Design Warrior's Guide to FPGAs,
ISBN 0750676043,
Copyright(C) 2004 Mentor Graphics Corp

MUX- vs. LUT-based Logic Blocks

Most FPGAs today are LUT-based -- here, the input signals are used as a pointer into a lookup table.



The Design Warrior's Guide to FPGAs,
ISBN 0750676043,
Copyright(C) 2004 Mentor Graphics Corp

Input signals can be decoded using a hierarchy of *transmission-gate* MUXs.

Transmission gates *pass* the value on their inputs or are **high-impedance**.

Note that the diagram does not show the serial connection of the cells (scan chain) for simplicity.

MUX- vs. LUT-based Logic Blocks

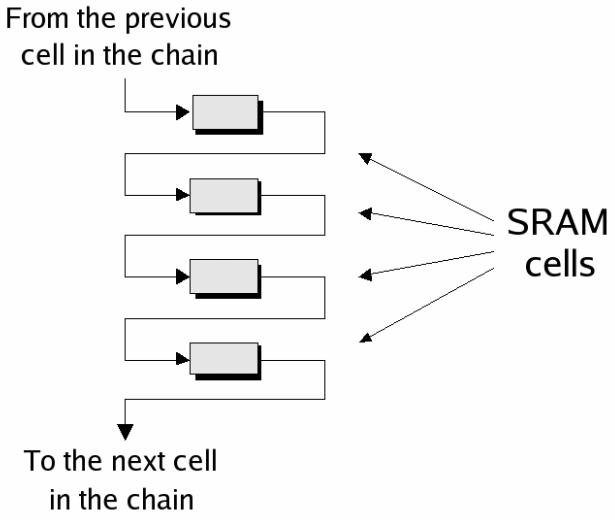
Larger LUTs are possible, e.g., 3-, 4-, 5- and 6-input versions.

Every time an input is added, the size of the table doubles.

4-input versions are believed to provide the optimal balance today.

Some vendors allow the 16 cells in the LUT to play the role as a *16x1 RAM*, and sets to be strewn together to form larger RAMs.

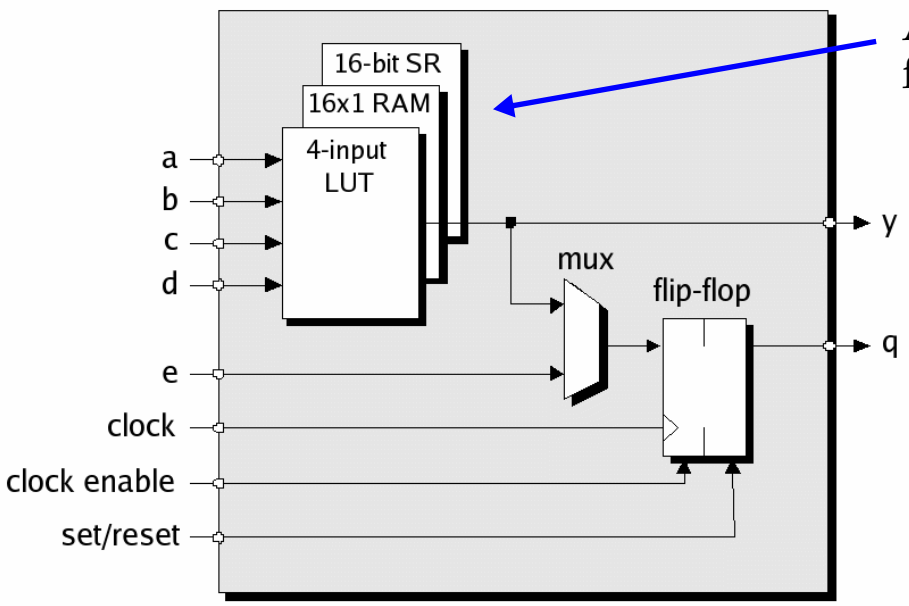
Some vendors allow the 16 cells of the LUT to be decoupled from the larger chain and used as a shift register.



The Design Warrior's Guide to FPGAs,
ISBN 0750676043,
Copyright(C) 2004 Mentor Graphics Corp

Terminology

Xilinx calls them *logic cells* (LC) while Altera calls them *logic elements* (LE).



Any one of several functions

The Design Warrior's Guide to FPGAs,
ISBN 0750676043,
Copyright(C) 2004 Mentor Graphics Corp

A *slice* is defined as 2 LCs.

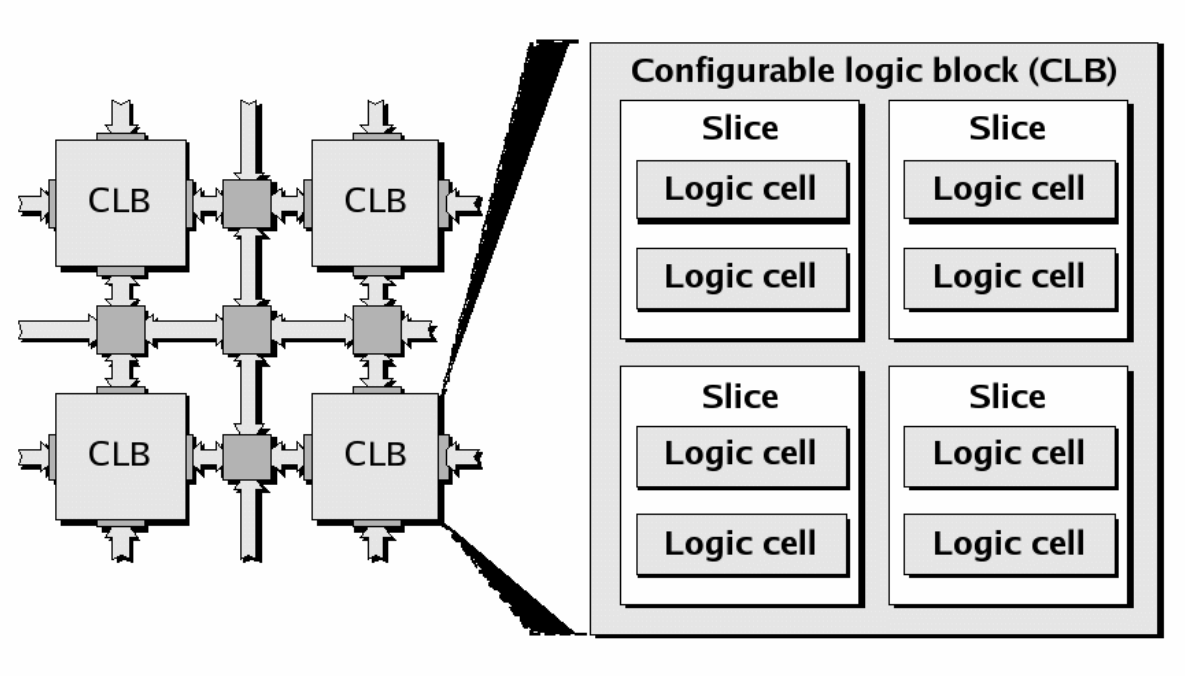
Each instance has its own inputs and outputs but the *clock*, *clock enable*, and *set/reset* signals are common.

A *configurable logic block* (CLB, Xilinx) or *logic array block* (LAB, Altera).

CLBs consist of 2 or 4 *slices*, and conform to the islands shown earlier.

Terminology and Hierarchy

The CLB also has some fast interconnect (not shown), that is used to connect neighboring slices.



The Design Warrior's Guide to FPGAs,
ISBN 0750676043,
Copyright(C) 2004 Mentor Graphics Corp

The organization of *LC -> Slice -> CLB* is complemented by an equivalent hierarchy in the interconnect.

That is, fast interconnect between LCs in a slice, slightly slower between slices in a CLB, followed by the interconnect between LCBs.

FPGA Additional Features and Characteristics

Assuming 4 slices per CLB, several RAM configurations are possible.

- Single-port 16x8, 32x4 bit, 64x2 or 128x1 bit RAM
- Dual-port 16x4, 32x2, 64x1 bit RAM (reads and writes occur through individual ports)

The LCs of the slices and slices of the CLB contain special interconnect to allow the *16 bit shift registers* of each LC to be combined into a *128 bit chain*.

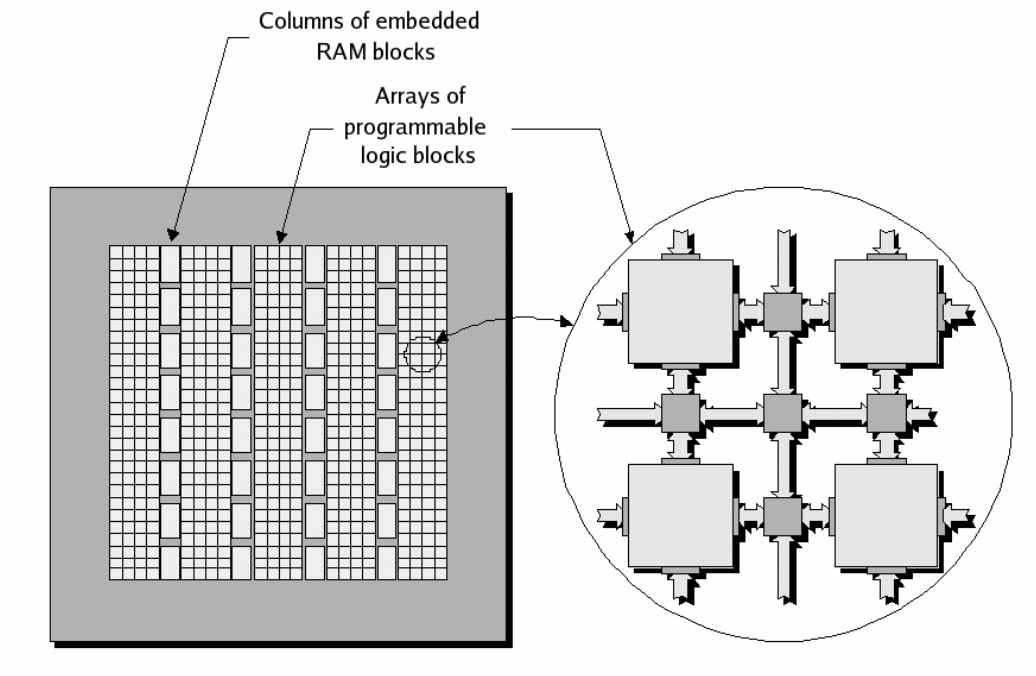
The LCs also include *special carry logic* (for fast carry chains), and dedicated interconnect between LCs in a slice, between slices and between CLBs.

These features boost the performance of logical functions, e.g., counters and adders.

Many applications require memory, so FPGAs now include *embedded RAM* called **e-RAM** or **block RAM**.

FPGA Additional Features and Characteristics

Some FPGAs include these blocks around the periphery of the chip, others distribute the blocks, while others organized it in columns.



The Design Warrior's Guide to FPGAs,
ISBN 0750676043,
Copyright(C) 2004 Mentor Graphics Corp

Each block can hold up to 10K bits, and each chip may contain hundreds of these *eRAM* blocks, for storage capacity up to several million bits.

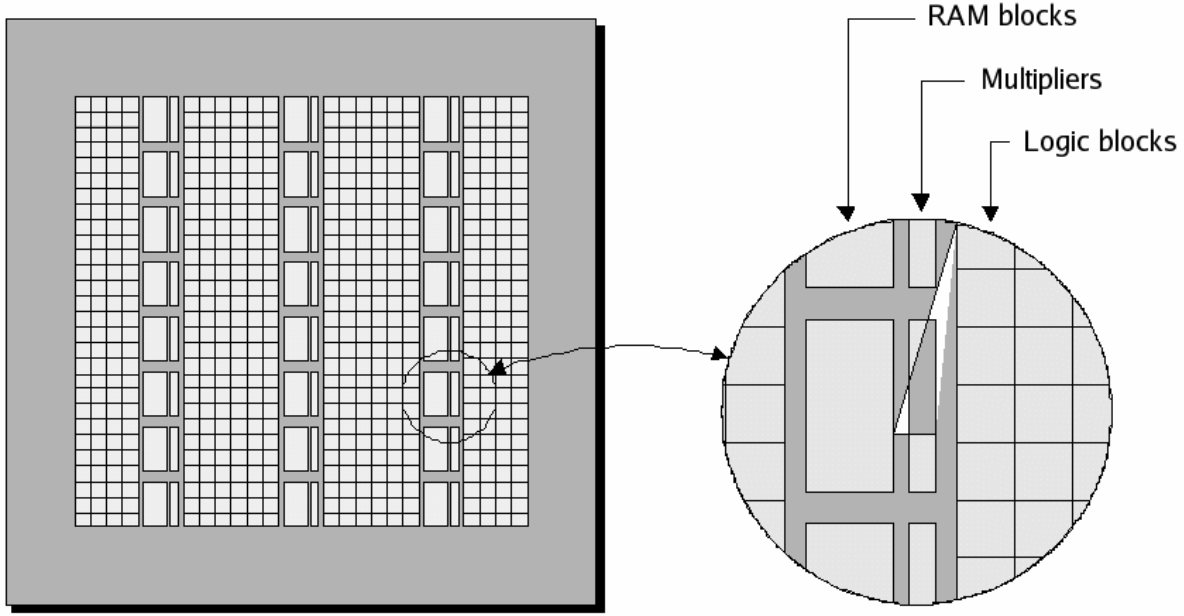
The *eRAM* blocks can be used separately or combined, and are useful for implementing single/dual-port RAMs, FIFO functions, state machines, etc.

FPGA Additional Features and Characteristics

Some functions, like multipliers, are slow if implemented by connecting LCBs together.

Since these functions are common, many include special hardwired multiplier blocks.

They are often located near the *eRAM* blocks.

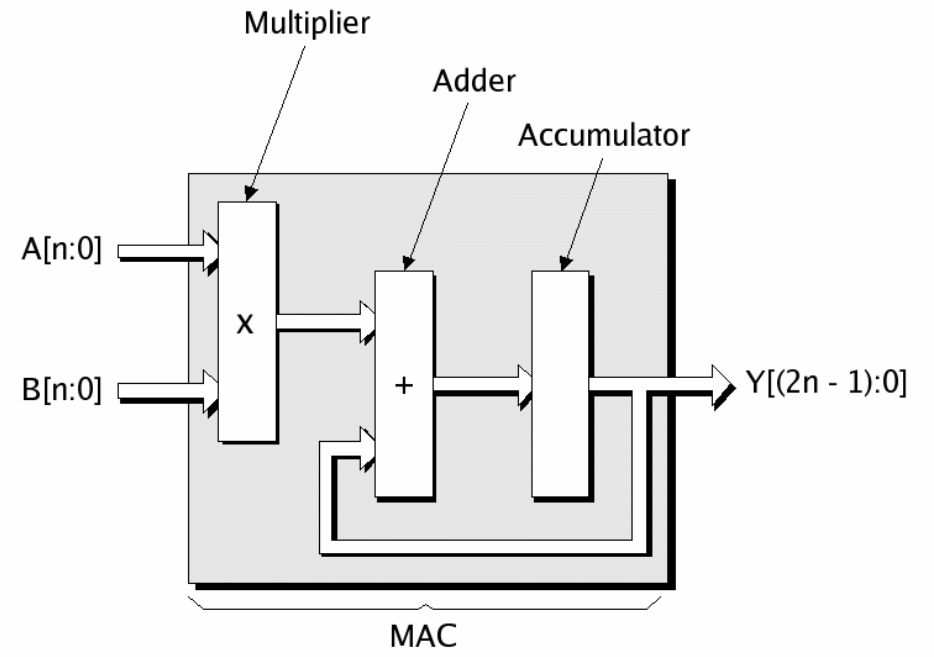


The Design Warrior's Guide to FPGAs,
ISBN 0750676043,
Copyright(C) 2004 Mentor Graphics Corp

FPGA Additional Features and Characteristics

Some FPGAs offer dedicated adder blocks.

These are useful in DSP operations called *multiply-and-accumulate* (MAC).



The Design Warrior's Guide to FPGAs,
ISBN 0750676043,
Copyright(C) 2004 Mentor Graphics Corp

This operation consists of multiplying two numbers and storing the result in as a running total (in the *accumulator*).