UNIVERSITÀ
DEGLI STUDI DI TRIESTE

dipartimento
di ingegneria
e architettura

Programmable logic devices and ASICs

A.Carini – Progettazione di sistemi elettronici

Standard logic circuits

- Till few years ago, the design of digital systems was based on standard logic circuits of series 54/74, with small, medium scale integration (SSI, MSI).
- Initially designed in TTL technology, were later developed in TTL Schottky and Schottky low power technologies, and also in CMOS technology.
- System design was based on *three basic steps:*
  1. Specifics definition,
  2. System description using an interconnection of standard logic circuits,
  3. Circuit assembly on an electronic board.
- Increasing the complexity of the system design, more and more SSI and MSI circuits were necessary, with *increase of cost, power dissipation, propagation delay, size*. These circuits could be easily cloned.

Programmable logic devices (PLDs)

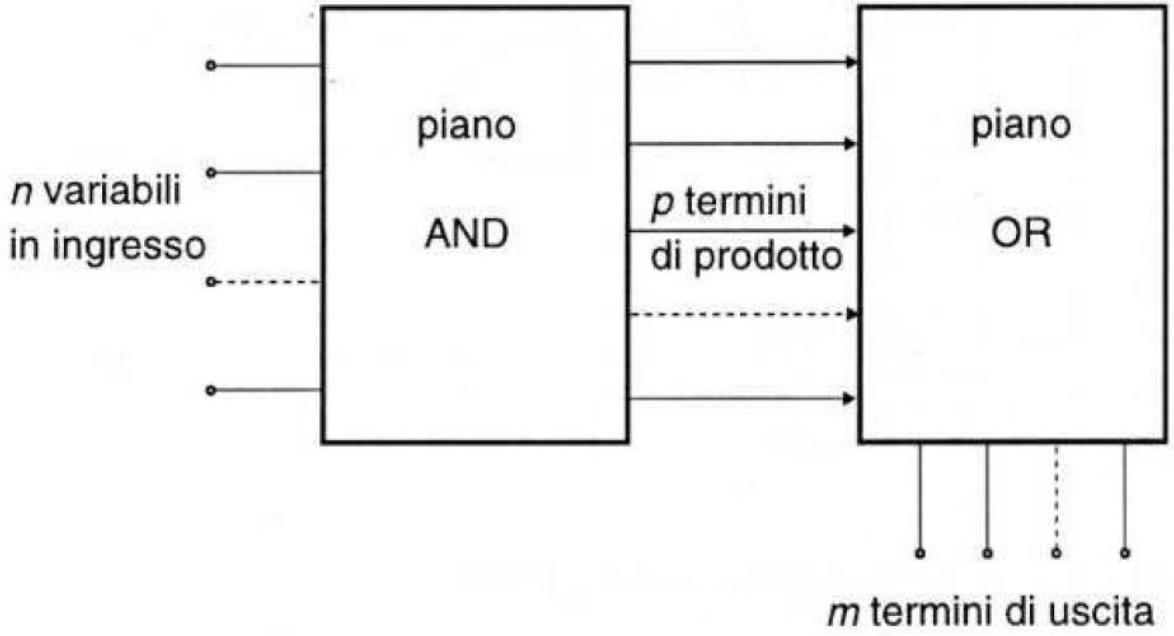
- To overcome these limitations, programmable logic devices were introduced.
- A PLD is a circuits with high integration scale (LSI, VLSI, ULSI, GLSI), which can be programmed and personalized by the final user to implement a specific function.
- A single PLD can replace many SSI, MSI circuits, with improvements in terms of occupied area in the printed board, reliability, and cost.
- Interconnections at printed board level are replaced by interconnections at chip level.
- The reduction of parasitic capacities improve propagation delays and power dissipation.
- PLD systems are more flexible: many PLDs are electrically reprogrammable. We can modify the system functions without adding of removing components.

Programmable logic devices (PLDs)

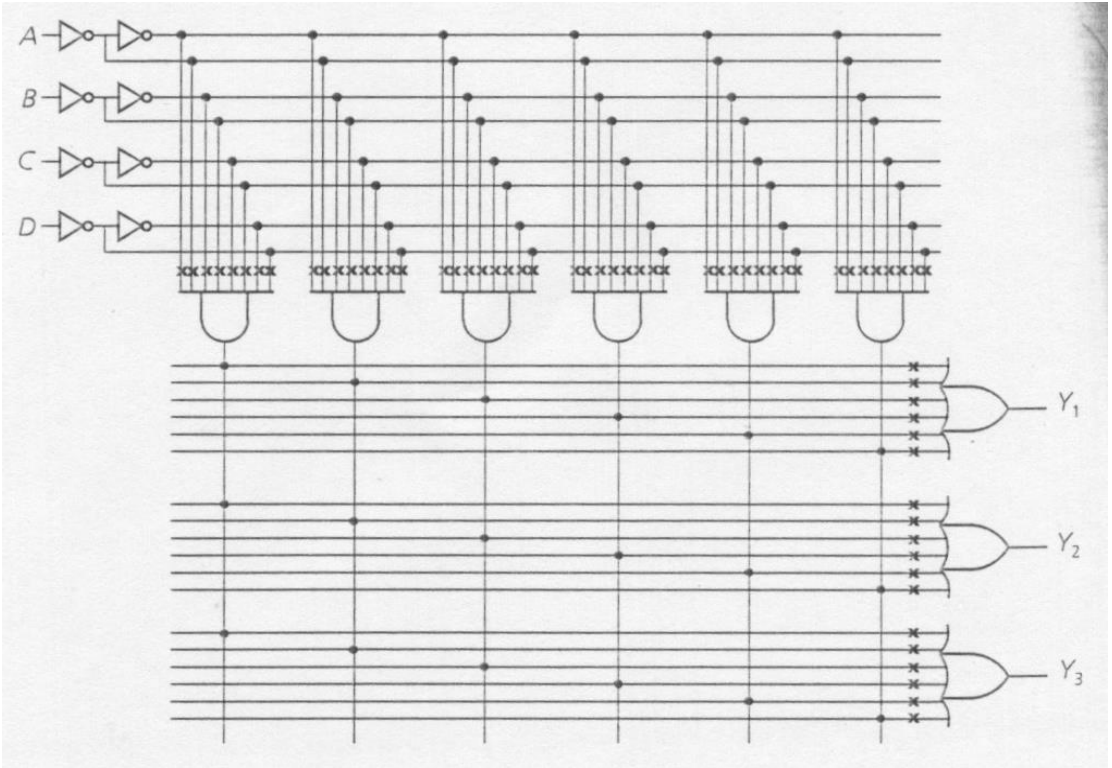
- The electronic system design based on PLDs exploits appropriate EDA tools, which allow to obtain in an almost automatic manner the PLD programming map from a high level description using an hardware description language (HDL) or a schematic representation of the digital system.
- In the following, we will see the evolution followed by PLDs, studying
  - Programmable Logic Arrays (PLAs),
  - Programmable Array Logic (PAL),
  - Sequential PLDs,
  - Complex PLDs (CPLDs)
  - Field Programmable Gate Arrays (FPGAs).

Programmable Logic Arrays (PLAs)

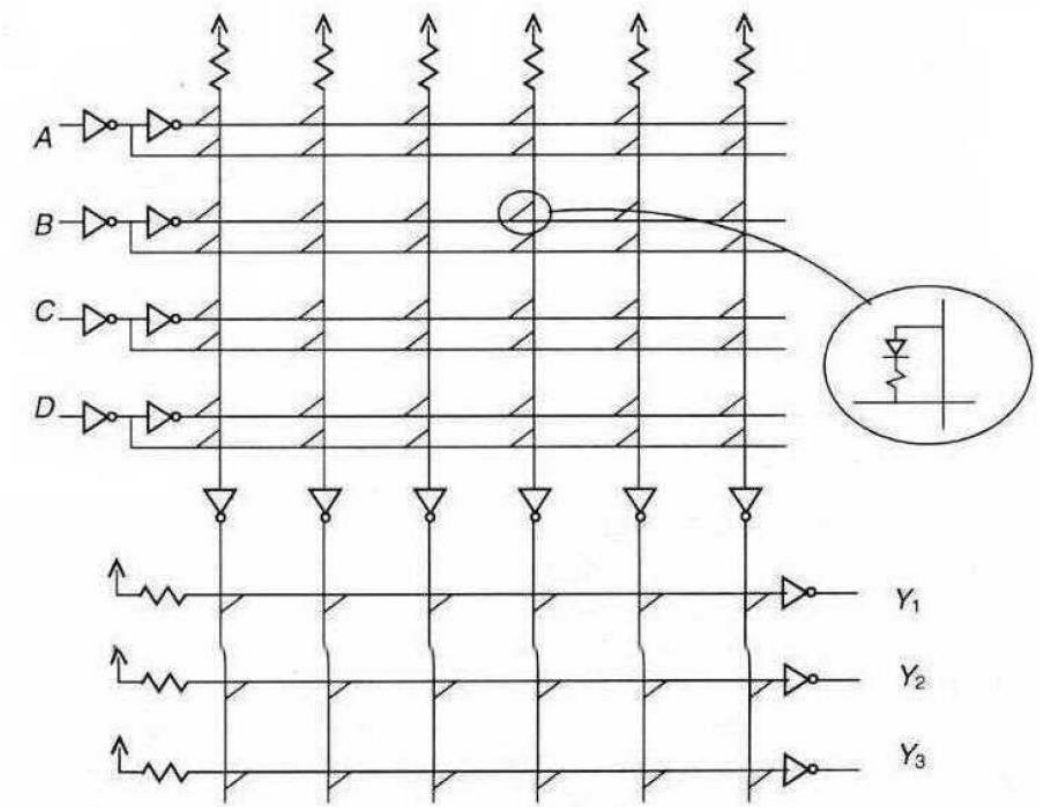
- Implement combinatorial logic functions in the form of sum of products.



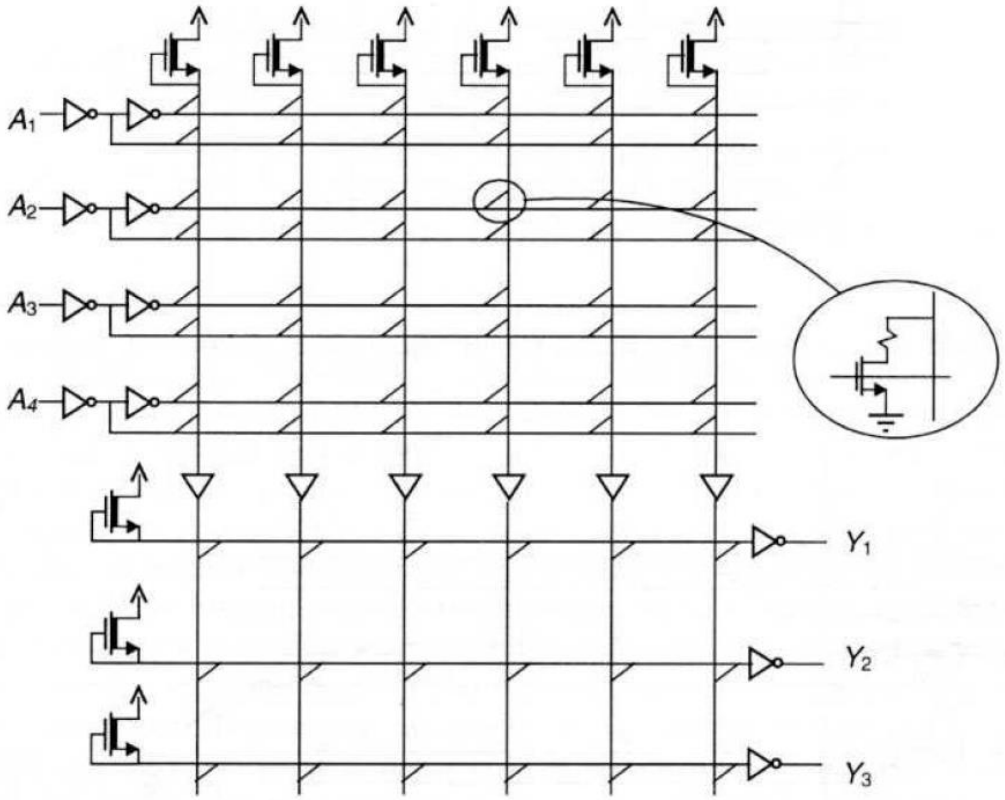
Programmable Logic Arrays (PLAs)



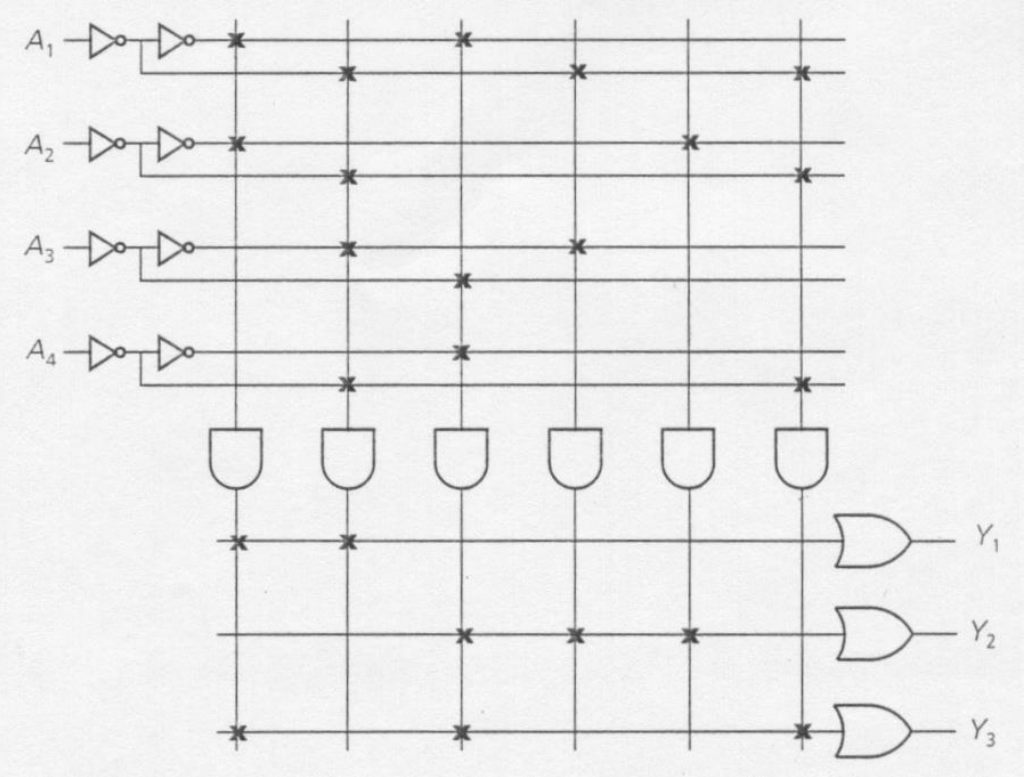
Programmable Logic Arrays (PLAs)



Programmable Logic Arrays (PLAs)

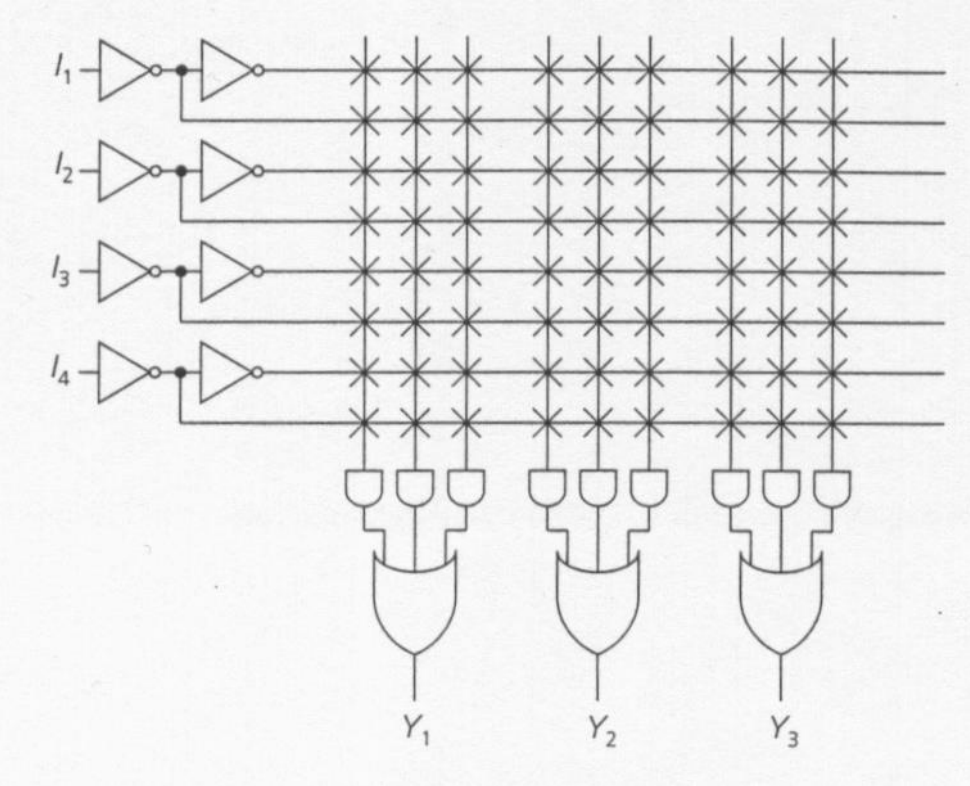


Programmable Logic Arrays (PLAs)

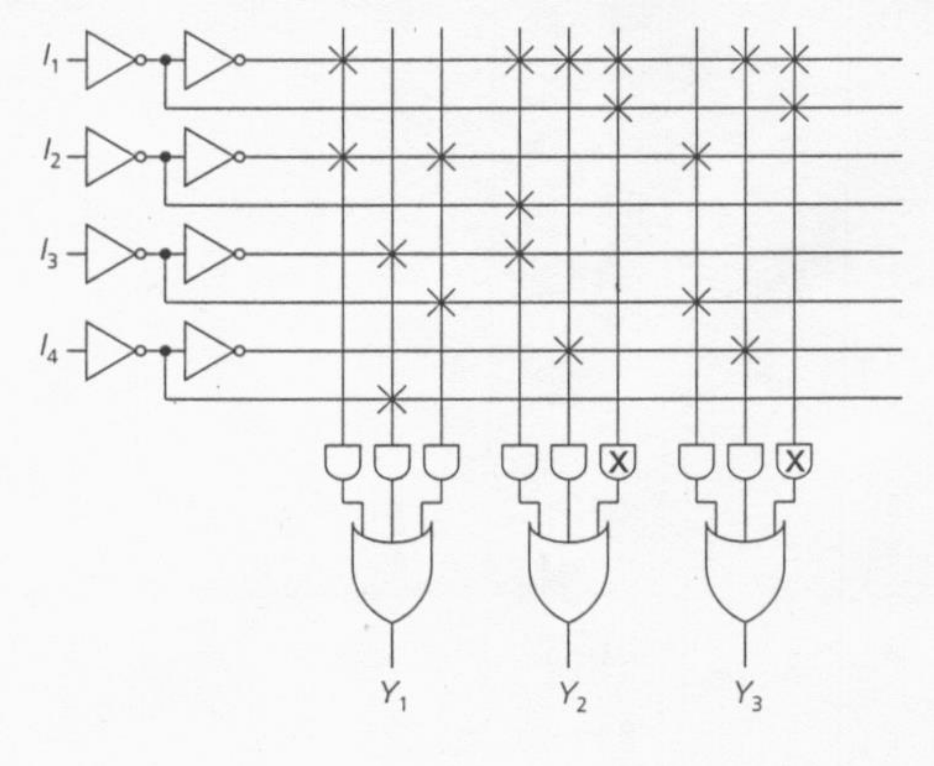


$$Y_1 = A_1 \cdot A_2 + \overline{A_1} \cdot \overline{A_2} \cdot A_3 \cdot \overline{A_4}$$
$$Y_2 = A_1 \cdot \overline{A_3} \cdot A_4 + \overline{A_1} \cdot A_3 + A_2$$
$$Y_3 = A_1 \cdot A_2 + A_1 \cdot \overline{A_3} \cdot A_4 + \overline{A_1} \cdot \overline{A_2} \cdot \overline{A_4}$$

Programmable Array Logics (PALs)

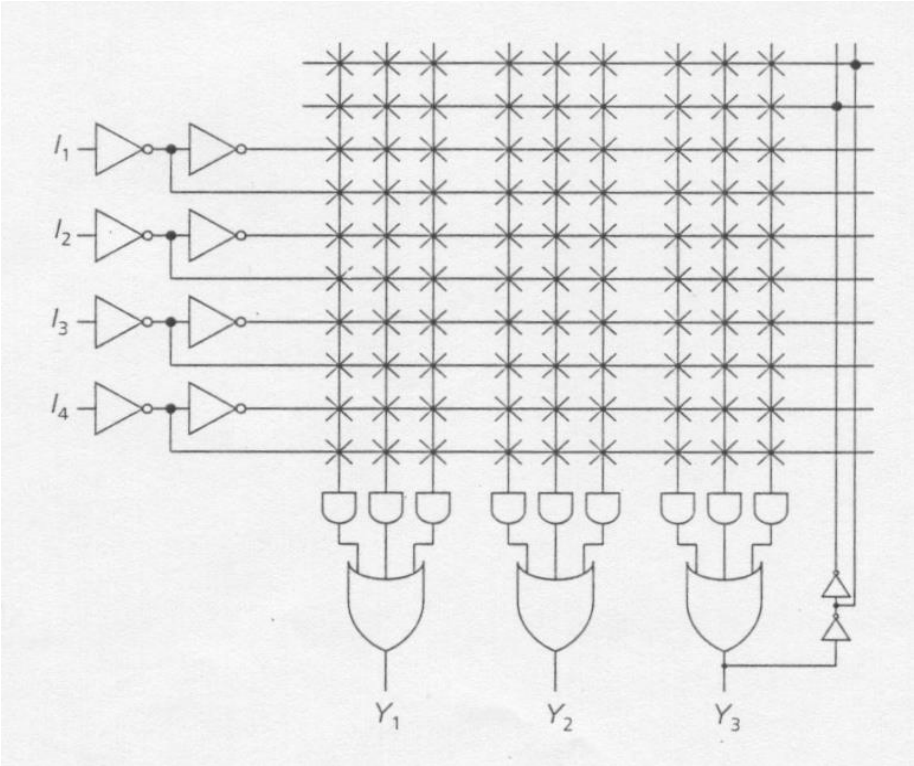


Programmable Array Logics (PALs)



$$Y_1 = I_1 I_2 + I_3 \overline{I_4} + I_2 \overline{I_3}$$
$$Y_2 = I_1 \overline{I_2} I_3 + I_1 I_4$$
$$Y_3 = I_2 \overline{I_3} + I_1 I_4$$

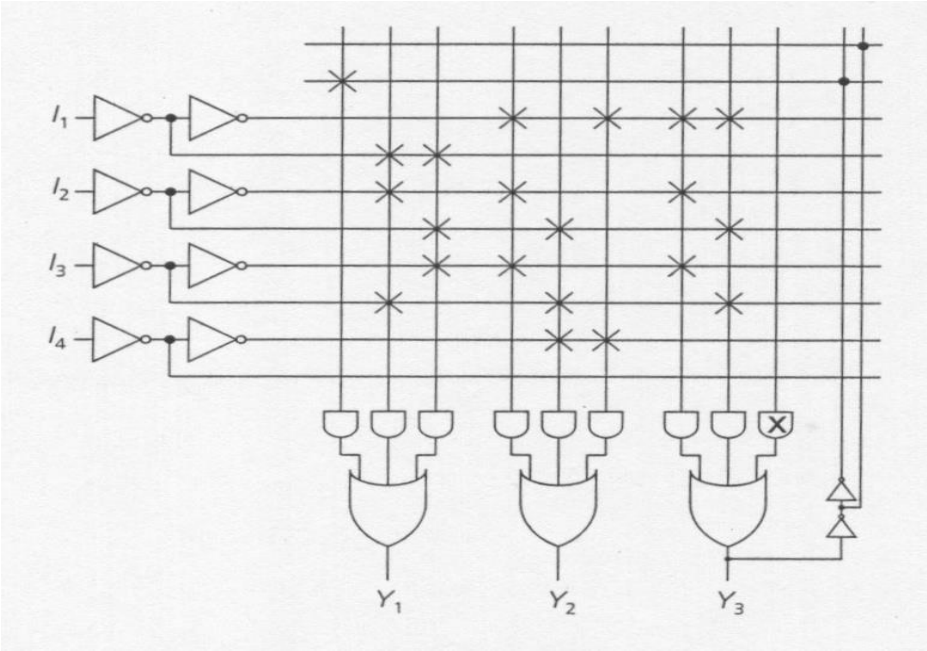
Two step logic



Two step logic

$$Y_1 = I_1 I_2 I_3 + I_1 \overline{I_2}\,\overline{I_3} + \overline{I_1} I_2 \overline{I_3} + \overline{I_1}\,\overline{I_2} I_3$$
$$Y_2 = I_1 I_2 I_3 + \overline{I_2}\,\overline{I_3} I_4 + I_1 I_4$$

$$Y_3 = I_1 I_2 I_3 + I_1 \overline{I_2}\,\overline{I_3}$$
$$Y_1 = Y_3 + \overline{I_1} I_2 \overline{I_3} + \overline{I_1}\,\overline{I_2} I_3$$



Output polarity programming

- The following function cannot be implemented with our network with a unique step:

$$Y_2 = I_1 + I_2 + I_3 + I_4$$

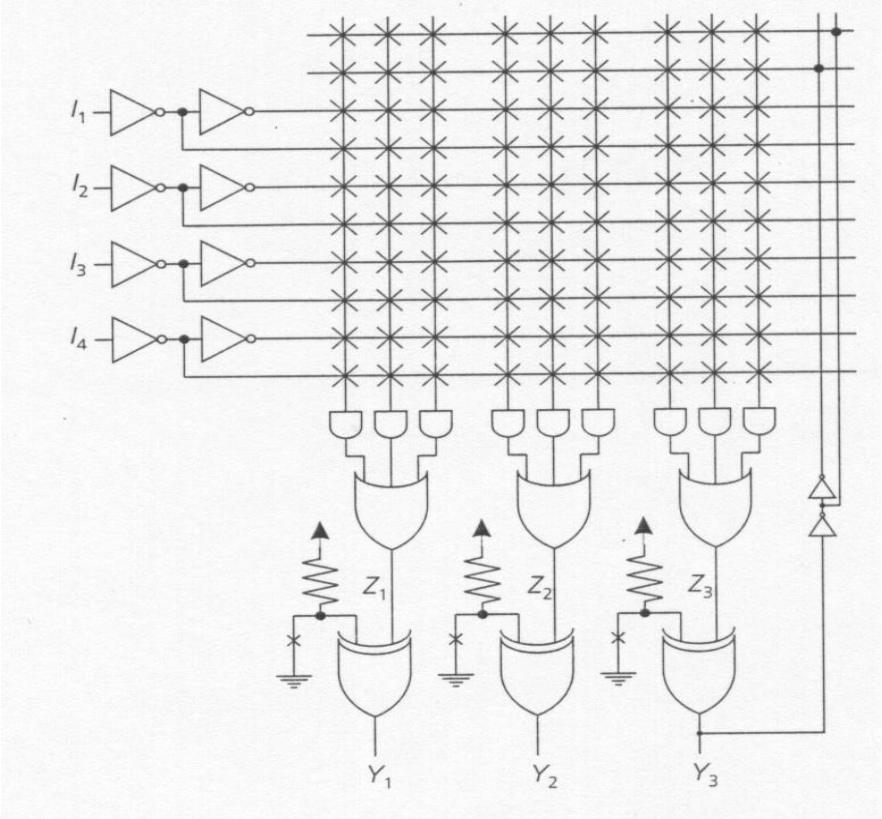
- while this function can be easily implemented!

$$\overline{Y_2} = \overline{I_1}\,\overline{I_2}\,\overline{I_3}\,\overline{I_4}$$

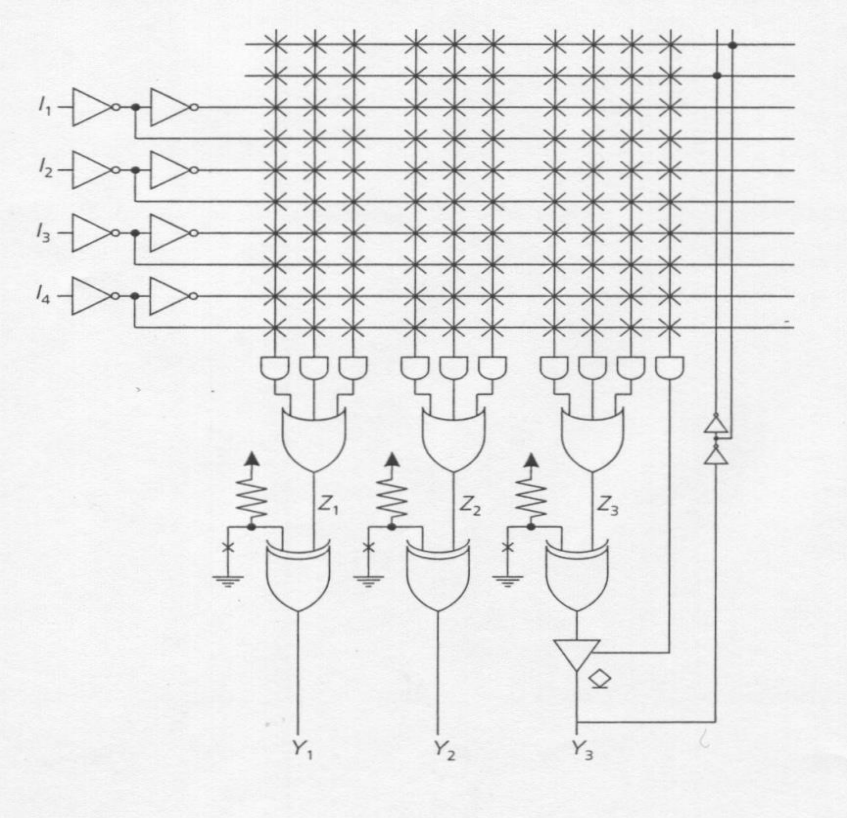
Output polarity programming

$$Y = Z \text{ XOR } 0 = Z$$

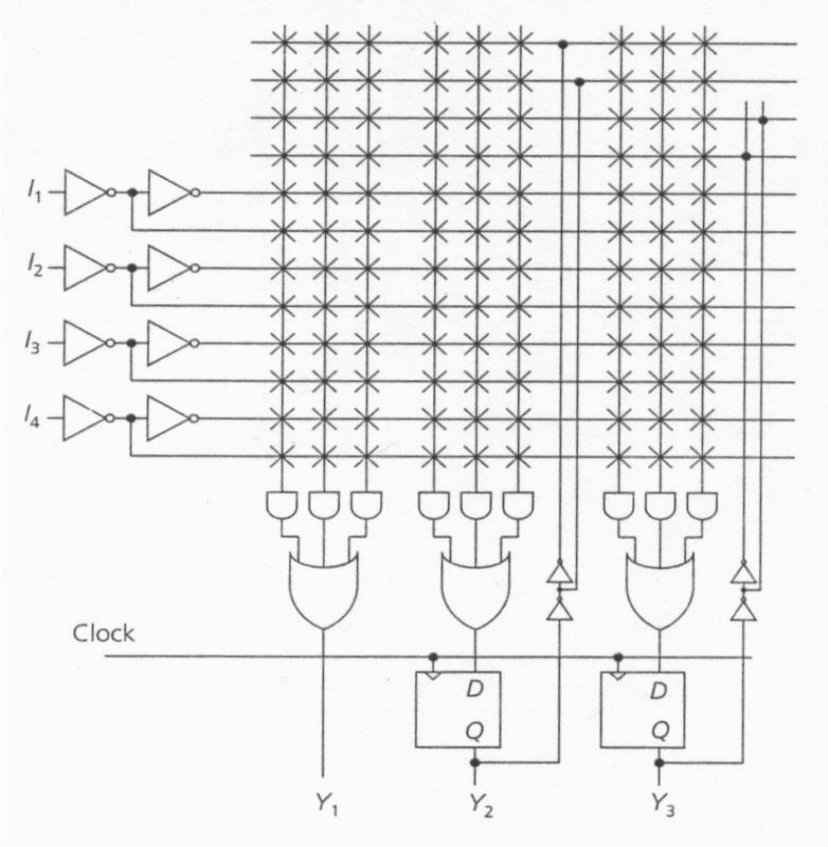
$$Y = Z \text{ XOR } 1 = \overline{Z}$$



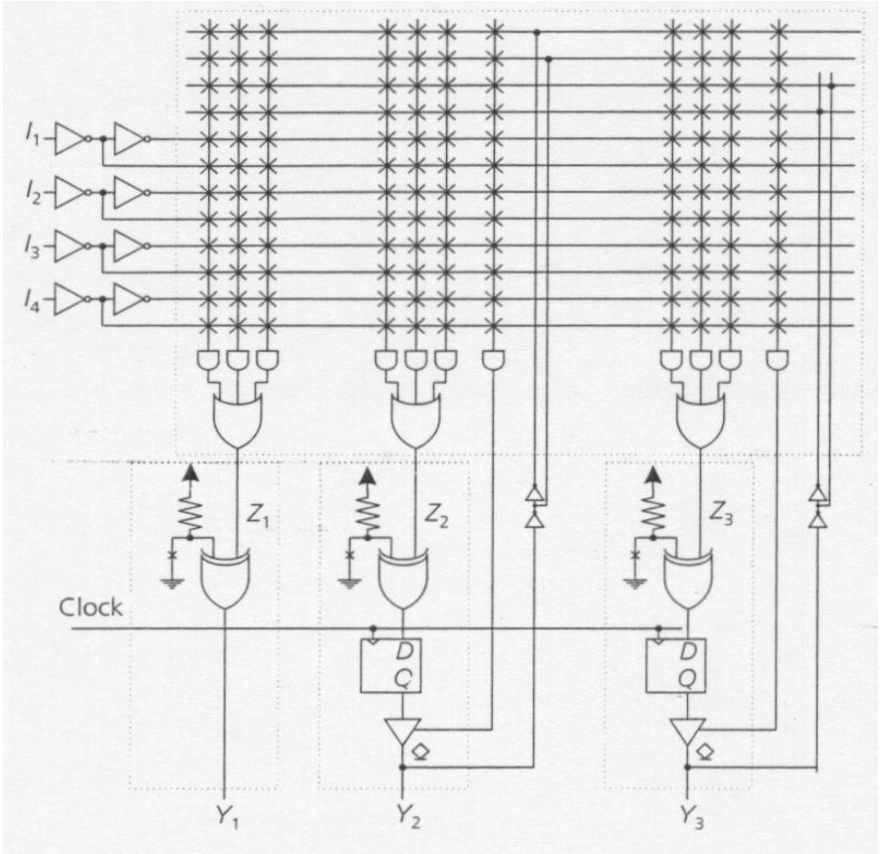
Programmable I/O terminals



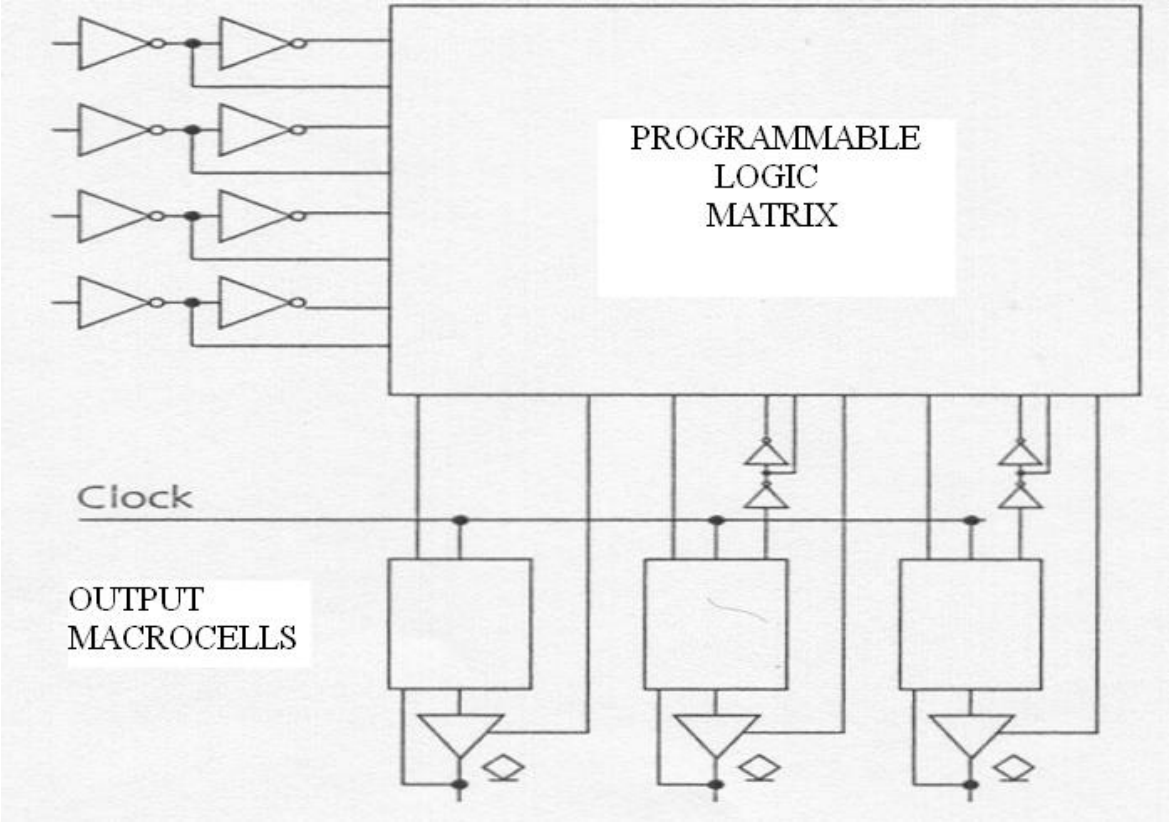
Sequential PLDs



Output macrocells

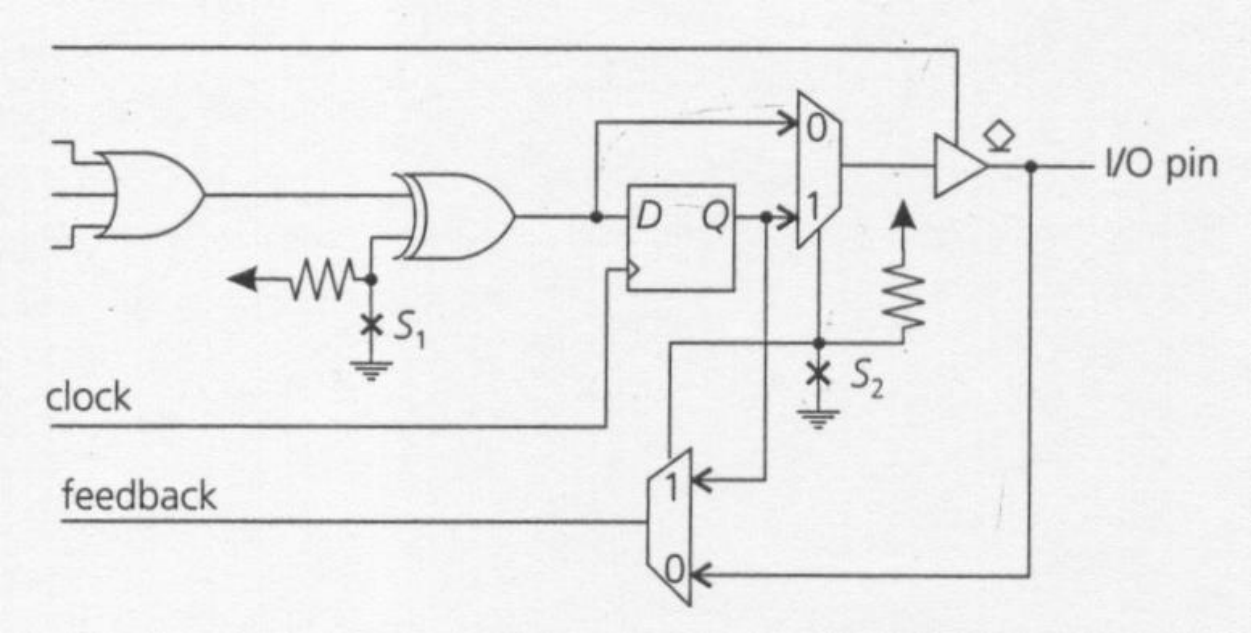


Output macrocells

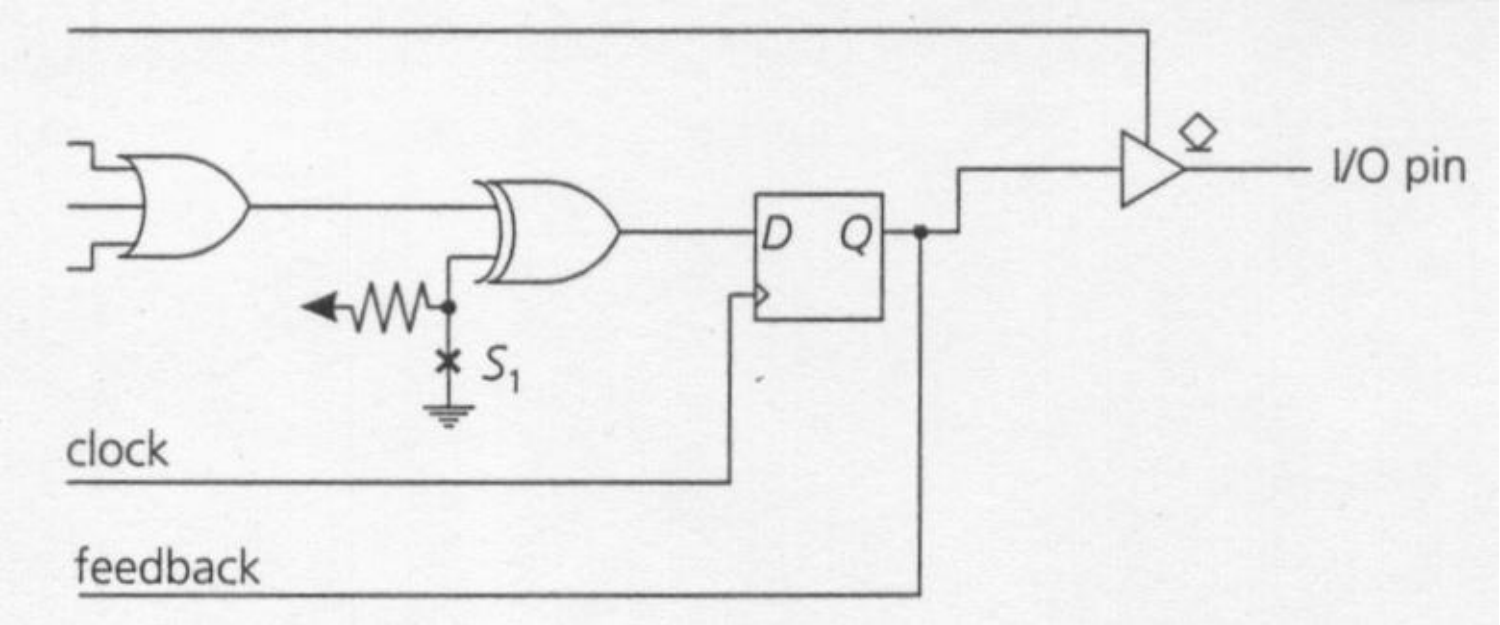


Output macrocells

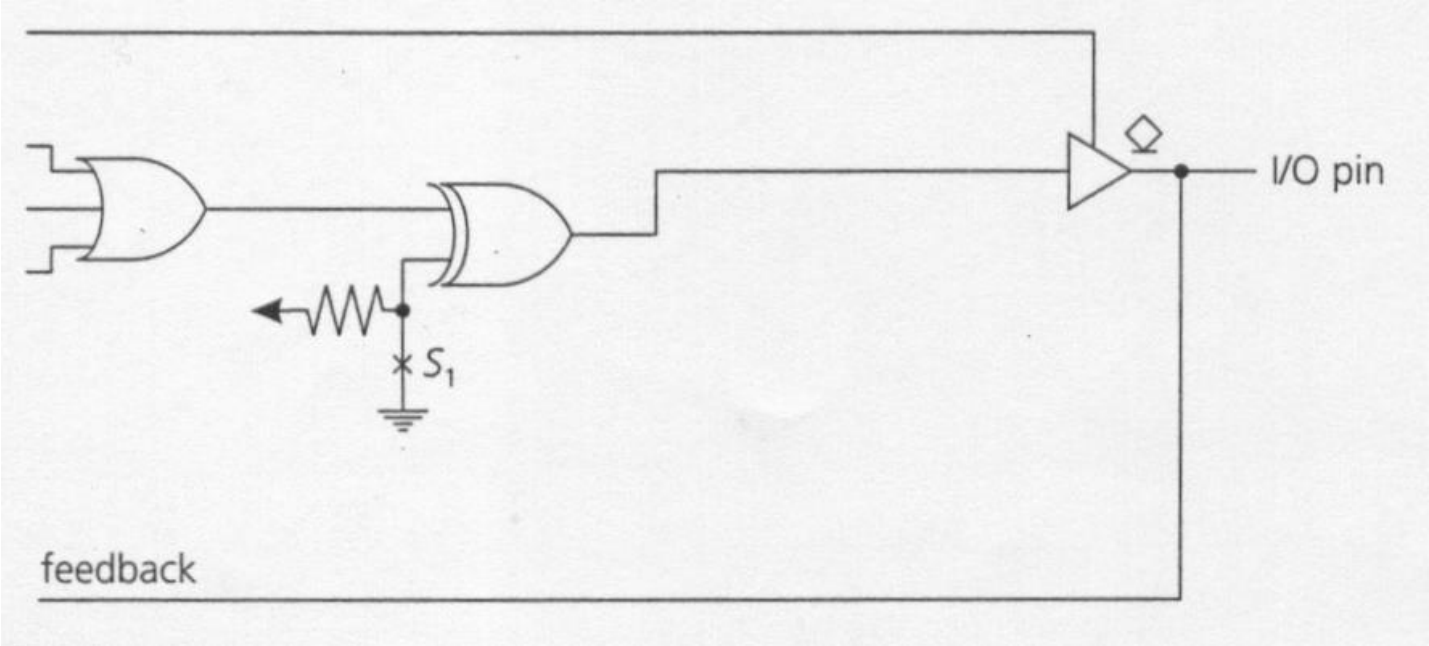
- Limitation of previous macrocells:
  - They don’t allow a two step logic
  - It is not possible to realize a finite state machine whose outputs are in high impedance state.
- Solution:



Output macrocell sequetial mode



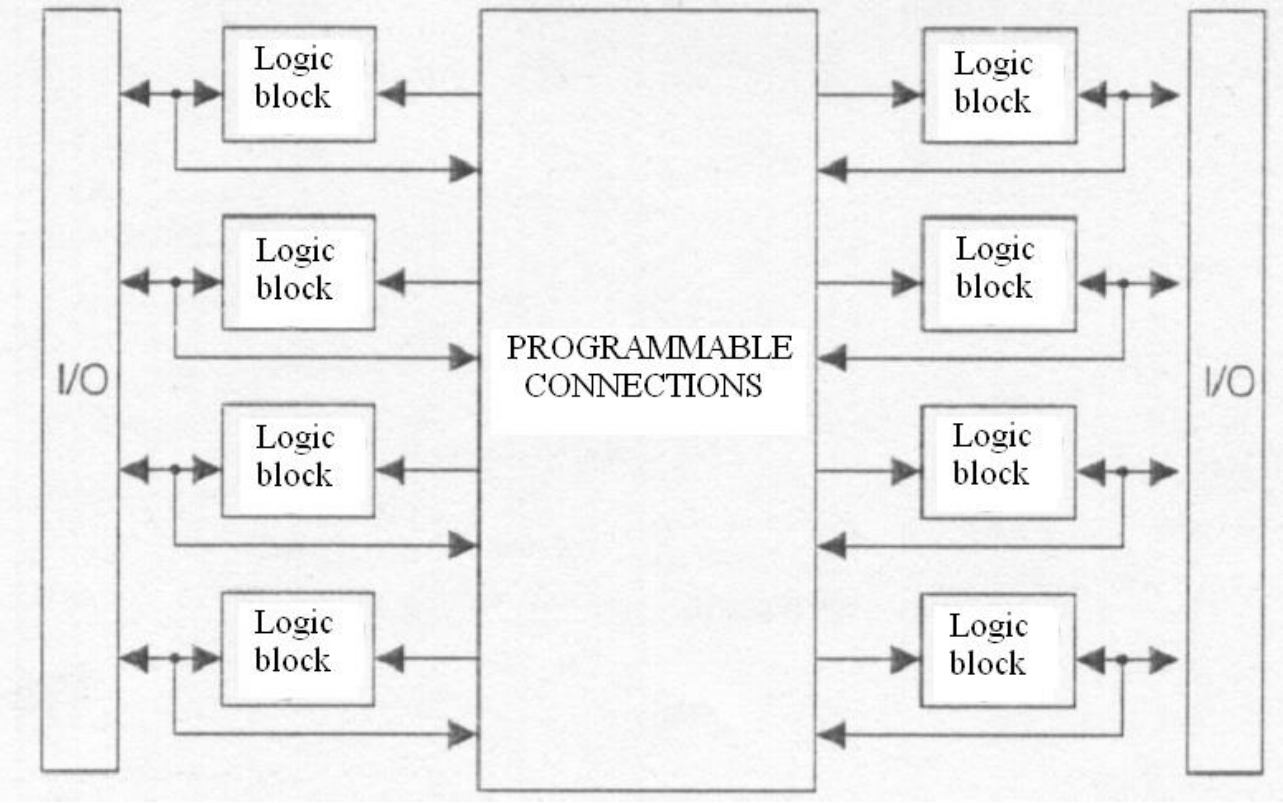
Output macrocell combinatorial mode



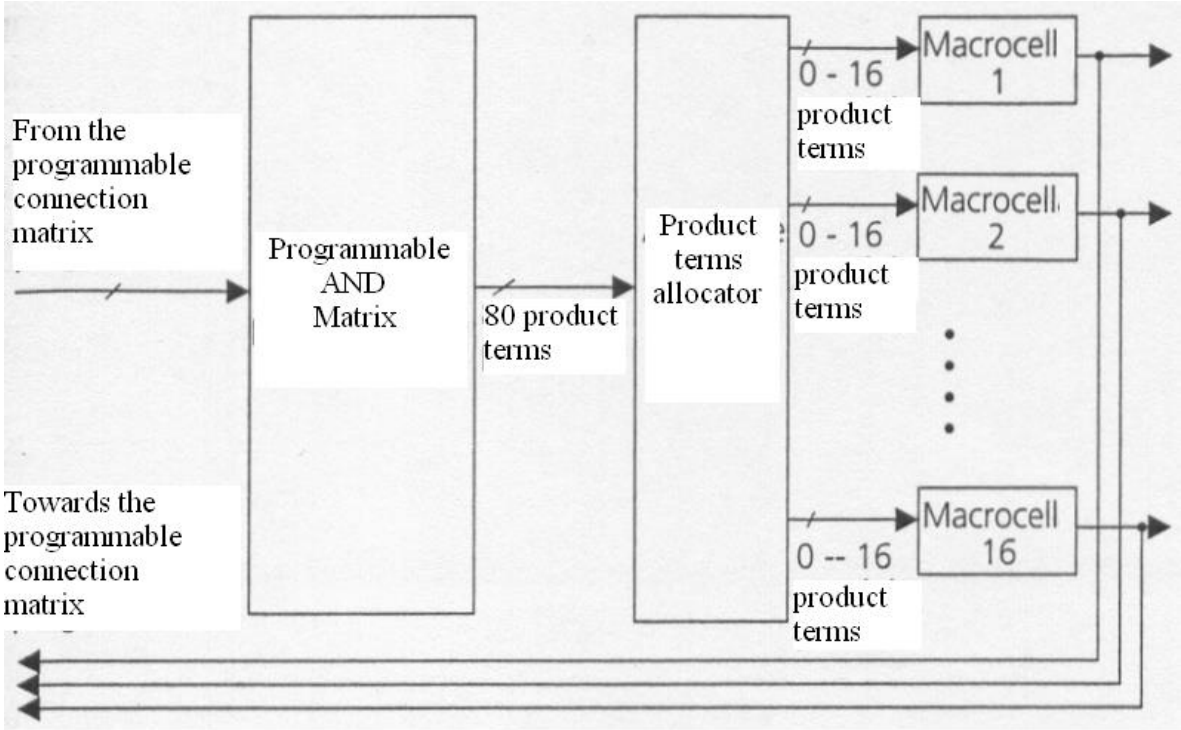
Complex PLDs (CPLDs)

- A parameter often used for providing an index of the *capacity* of a PLD is the maximum number of equivalent gates.
- A PLD has a capacity of 1000 gates if it can implement logic functions that would need 1000 two-inputs NAND gates for its realization.
- In PAL circuits, an increase of the capacity translates in an increase of the inputs of the AND plane. This increases the AND plane gates fan-in with a deterioration of the propagation delay.
- Instead of increasing the number of inputs of the AND plane, it is much more convenient to increase the PLD capacity by integrating on the same chip different logic blocks similar to a PLA.
- The various blocks are interconnected with programmable interconnections and the resulting devices are called complex PLDs or CPLDs.

Complex PLDs (CPLDs)

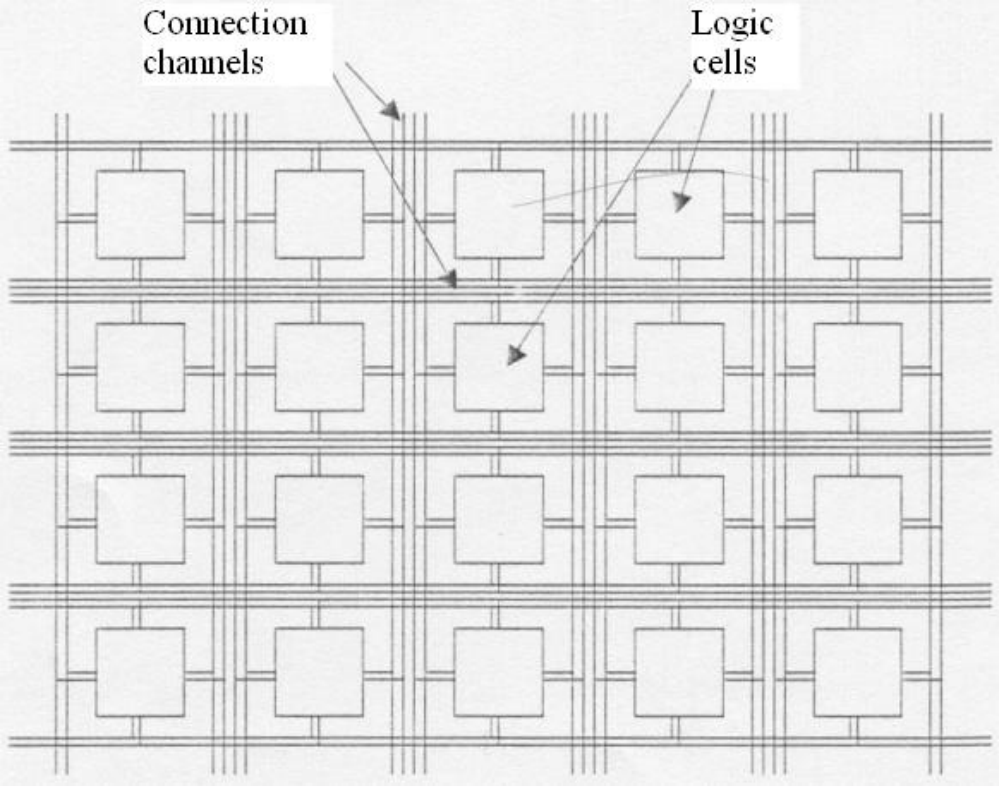


Complex PLDs (CPLDs)

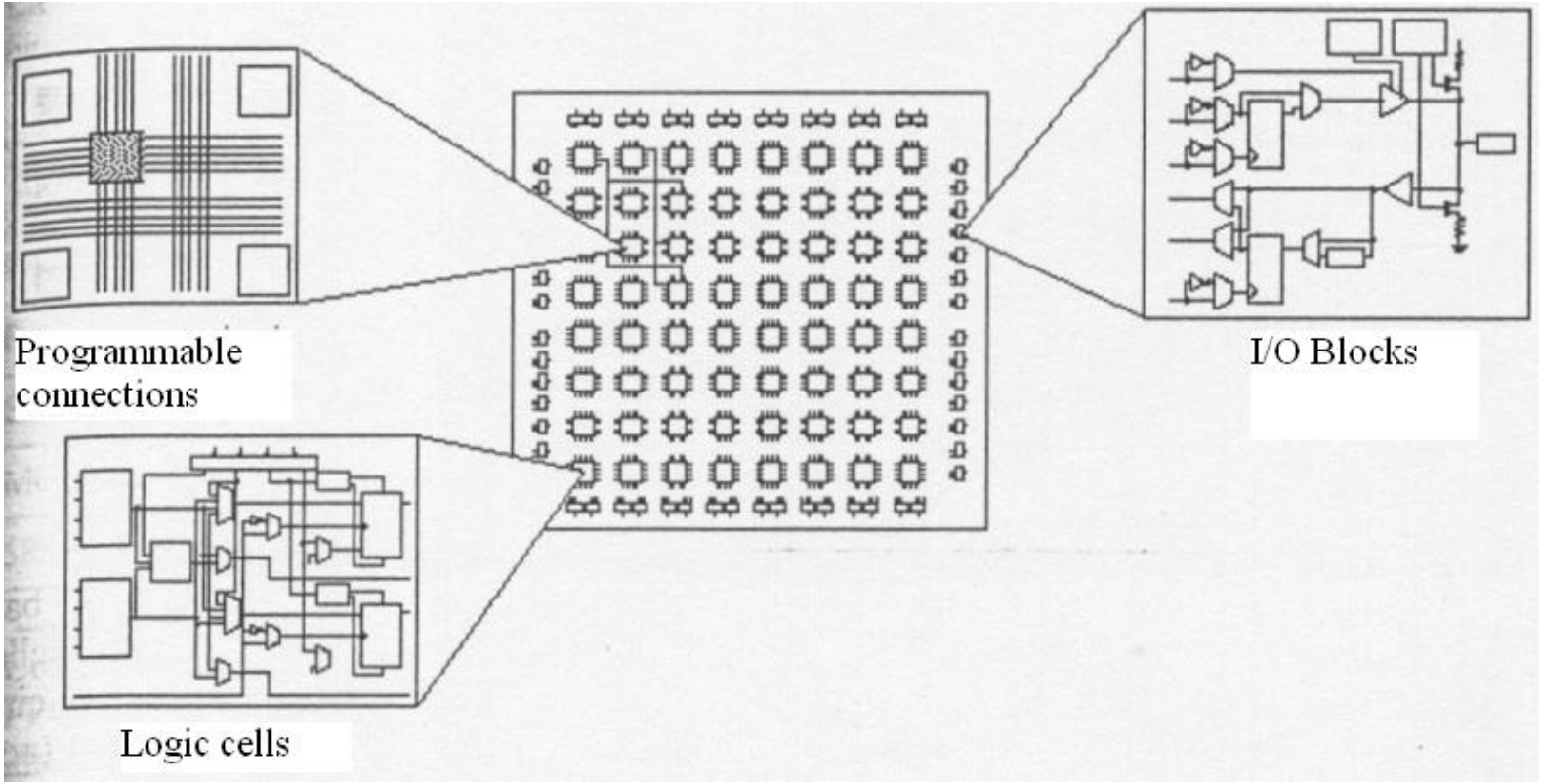


Logic block of family 370 of Cypress

Mask programmable gate arrays



Field programmable gate arrays (FPGAs)



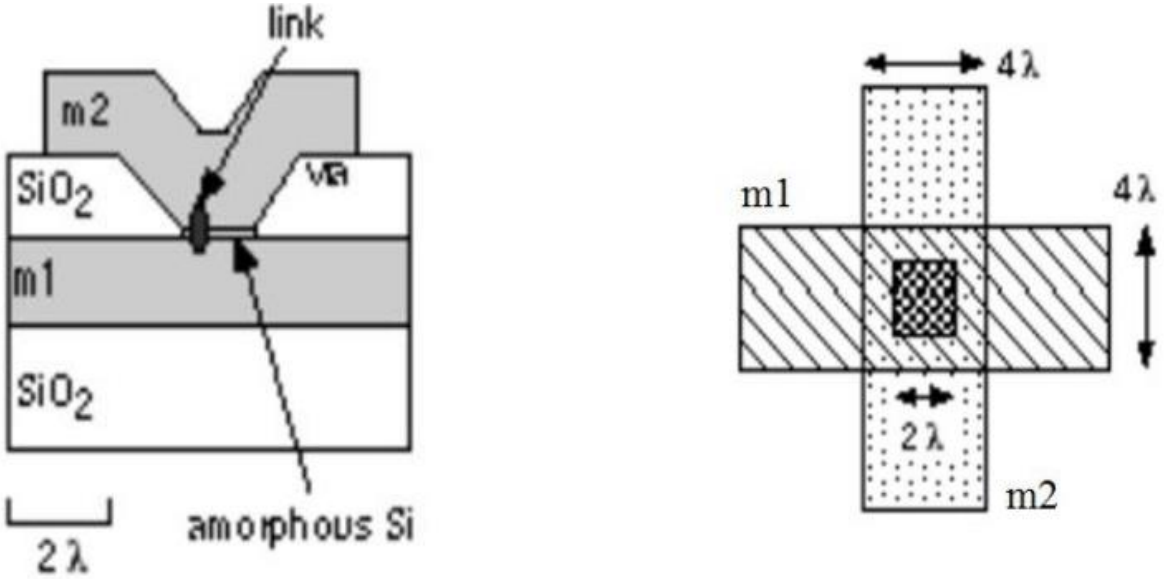
Differences between CPLDs and FPGAs

- Both are composed by logic blocks and programmable interconnections.
- Main difference: the logic cells of FPGAs are in general logically simpler (more limited) than the logic blocks of CPLDs.
- At the same time, the number of logic cells in FPGAs is in general much larger than the logic blocks of CPLDs.
- Thus, FPGAs have a fine grain while CPLDs have a larger grain.
- Another difference comes from the propagation delay:
- In FPGAs, the logic function is realized from the cascade of different logic cells. The propagation delay is determined by the number of logic levels needed to implement the functions and is known only at the end of the place and routing.
- In CPLDs, most logic functions can be implemented with a two step logic or with at most a global feedback (two blocks). The propagation delay can often be determined in the early phases of the design.

Programming techniques

- In PLAs, PALs, and CPLDs, the programming technique is based on the use of floating-gate MOS transistors and it has followed the technological evolution of non-volatile memories.
- In FPGAs, the one-time programmable devices use antifuse, while the reprogrammable devices SRAM memories (thus, requiring an external memory for storing the programming map).

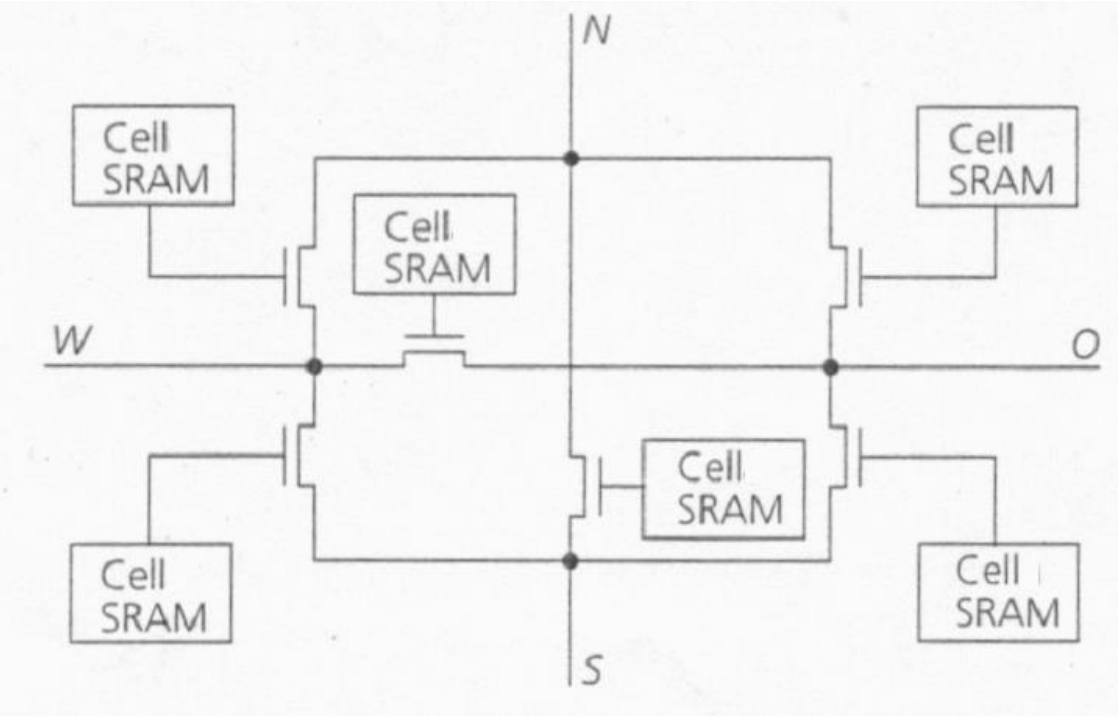
Programming techniques



Cross section of a Metal-metal antifuse

Programmable interconnections

- Larger area occupation and longer propagation delays with respect to antifuse.



Example: Altera MAX 3000 A CPLD (2003)

MAX 3000A

Programmable Logic Device Family

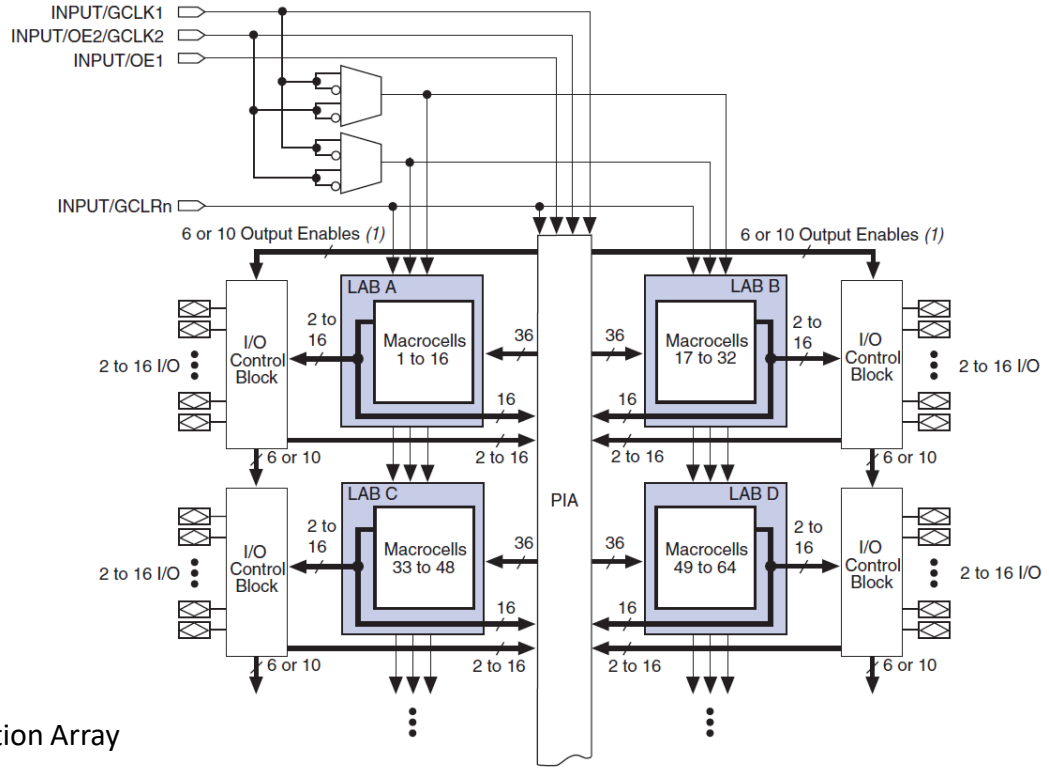
June 2006, ver. 3.5 Data Sheet

Table 1. MAX 3000A Device Features

Feature	EPM3032A	EPM3064A	EPM3128A	EPM3256A	EPM3512A
Usable gates	600	1,250	2,500	5,000	10,000
Macrocells	32	64	128	256	512
Logic array blocks	2	4	8	16	32
Maximum user I/O pins	34	66	98	161	208
t_{PD} (ns)	4.5	4.5	5.0	7.5	7.5
t_{SU} (ns)	2.9	2.8	3.3	5.2	5.6
t_{CO1} (ns)	3.0	3.1	3.4	4.8	4.7
f_{CNT} (MHz)	227.3	222.2	192.3	126.6	116.3

Example: Altera MAX 3000 A CPLD (2003)

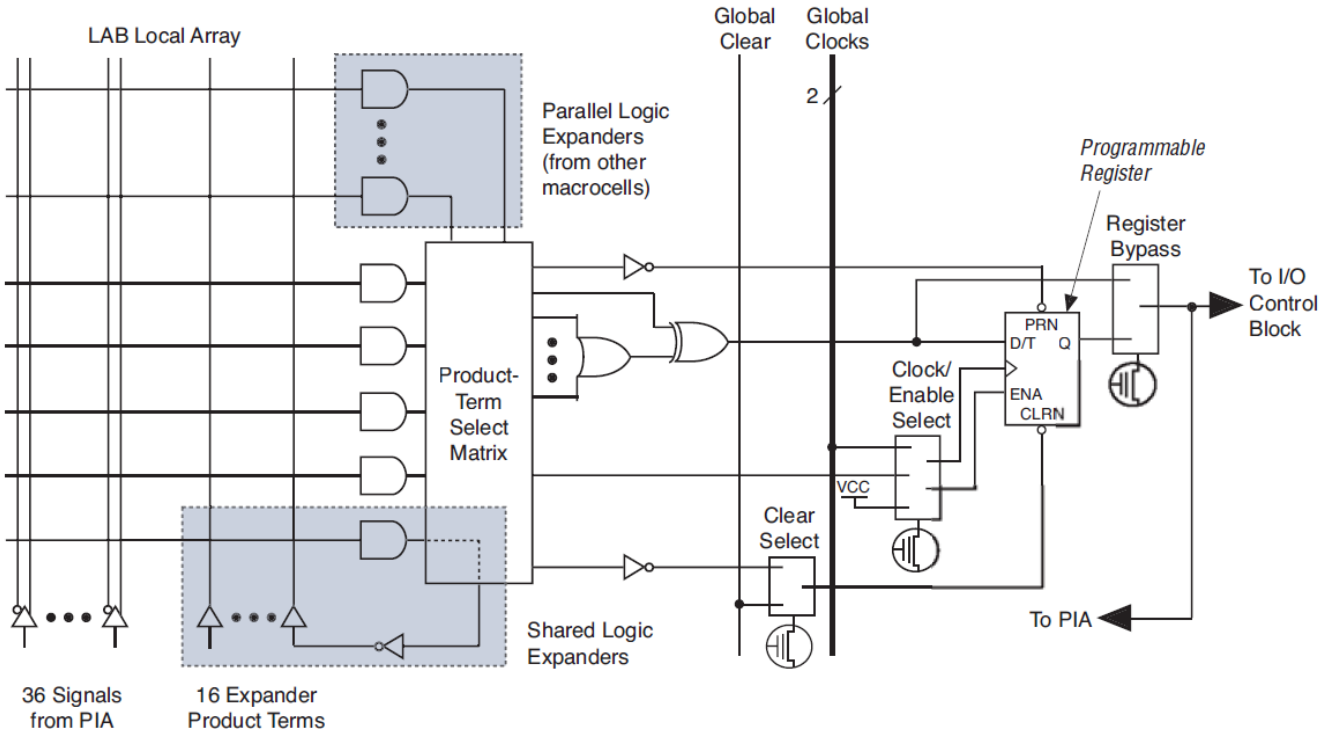
Figure 1. MAX 3000A Device Block Diagram



PIA Programmable Interconnection Array

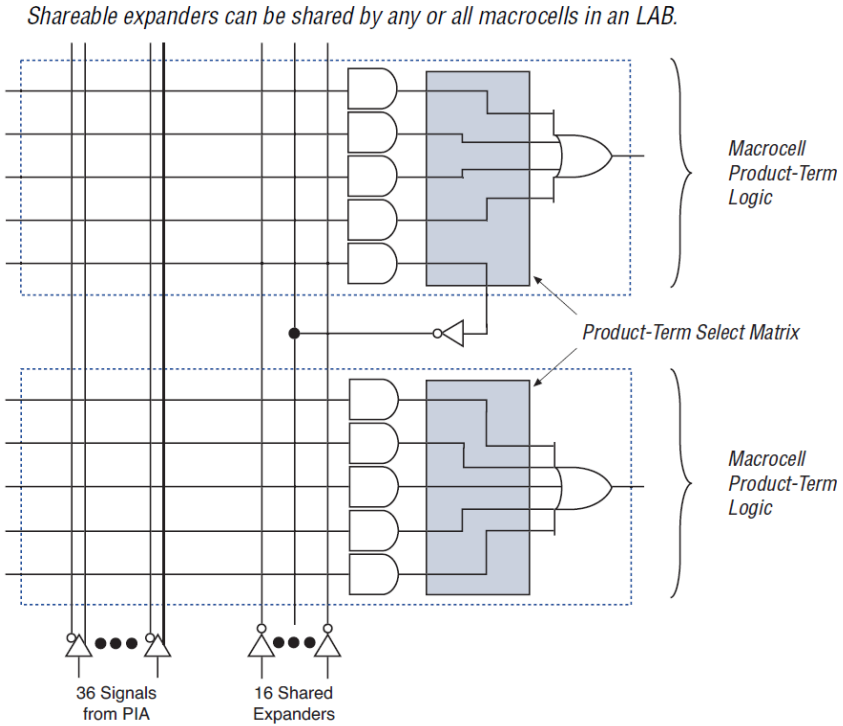
Example: Altera MAX 3000 A CPLD (2003)

Figure 2. MAX 3000A Macrocell



Example: Altera MAX 3000 A CPLD (2003)

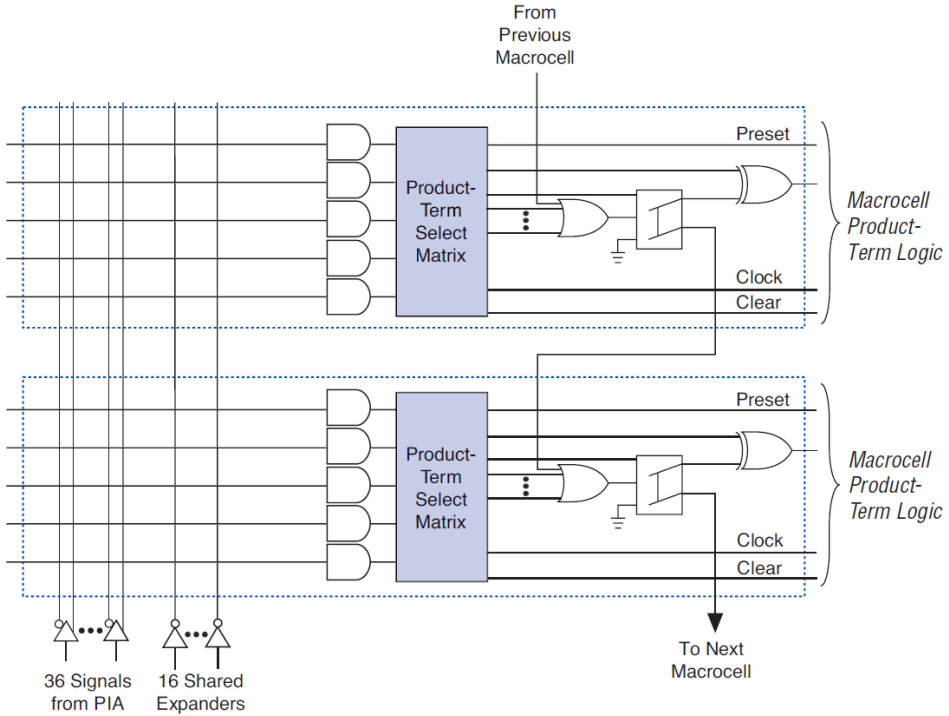
Figure 3. MAX 3000A Shareable Expanders



Example: Altera MAX 3000 A CPLD (2003)

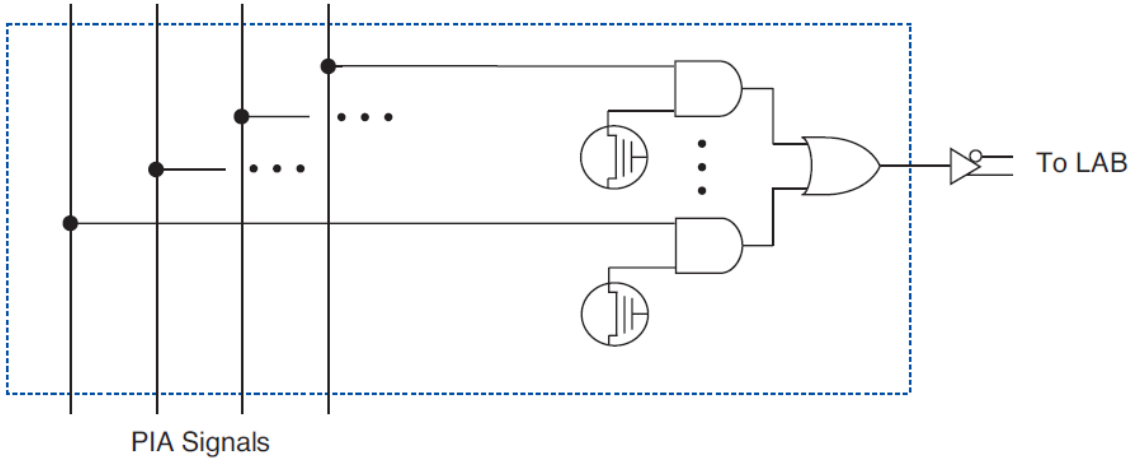
Figure 4. MAX 3000A Parallel Expanders

Unused product terms in a macrocell can be allocated to a neighboring macrocell.



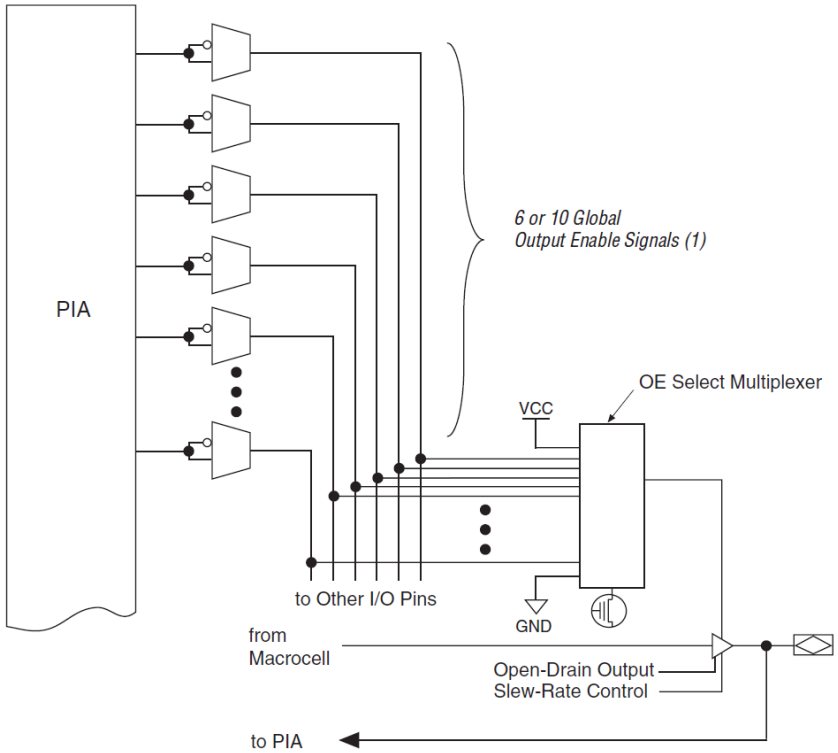
Example: Altera MAX 3000 A CPLD (2003)

Figure 5. MAX 3000A PIA Routing



Example: Altera MAX 3000 A CPLD (2003)

Figure 6. I/O Control Block of MAX 3000A Devices



Example: Altera MAX II (2006)

MAX® II

ALTERA®

1. Introduction

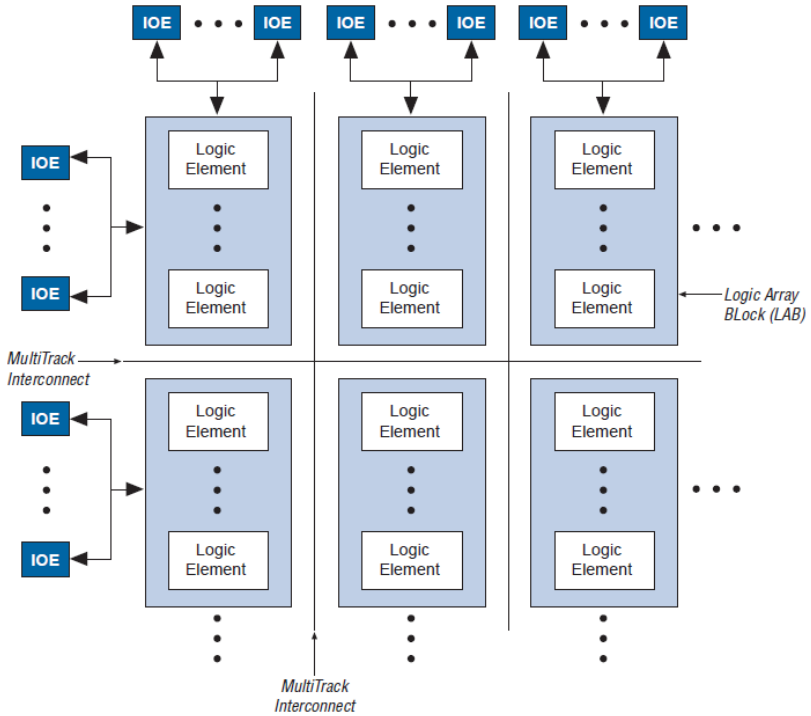
MII51001-1.9

Introduction

The MAX® II family of instant-on, non-volatile CPLDs is based on a 0.18-µm, 6-layer-metal-flash process, with densities from 240 to 2,210 logic elements (LEs) (128 to 2,210 equivalent macrocells) and non-volatile storage of 8 Kbits. MAX II devices offer high I/O counts, fast performance, and reliable fitting versus other CPLD architectures. Featuring MultiVolt core, a user flash memory (UFM) block, and enhanced in-system programmability (ISP), MAX II devices are designed to reduce cost and power while providing programmable solutions for applications such as bus bridging, I/O expansion, power-on reset (POR) and sequencing control, and device configuration control.

Example: Altera MAX II (2006)

Figure 2–1. MAX II Device Block Diagram



Example: Altera MAX II (2006)

Table 2–1. MAX II Device Resources

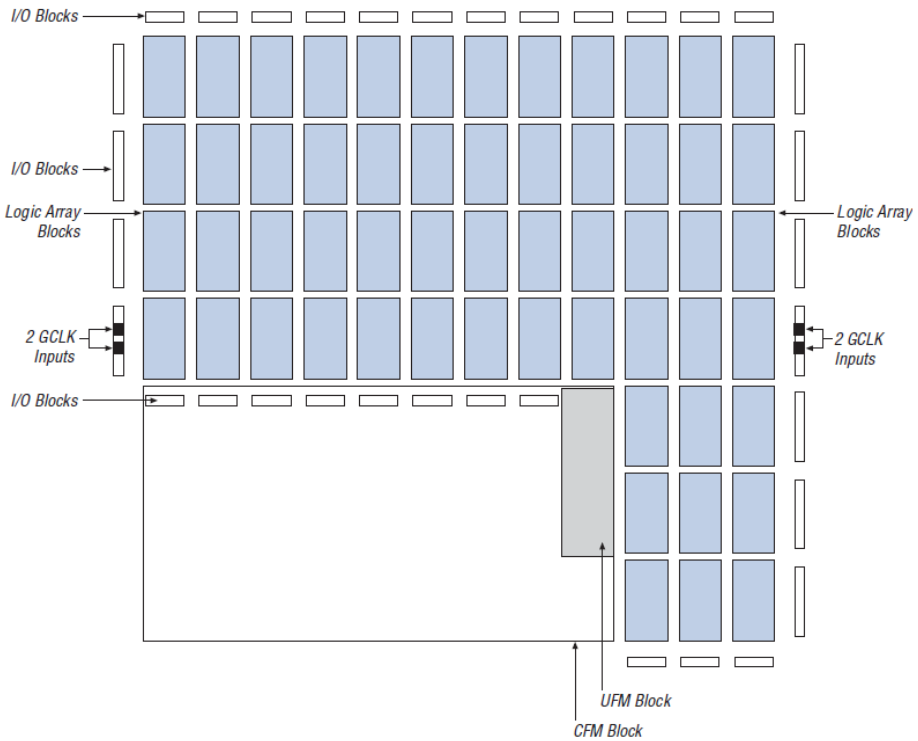
Devices	UFM Blocks	LAB Columns	LAB Rows		Total LABs
			Long LAB Rows	Short LAB Rows (Width) *(1)*	
EPM240	1	6	4	—	24
EPM570	1	12	4	3 (3)	57
EPM1270	1	16	7	3 (5)	127
EPM2210	1	20	10	3 (7)	221

Note to Table 2–1:

(1) The width is the number of LAB columns in length.

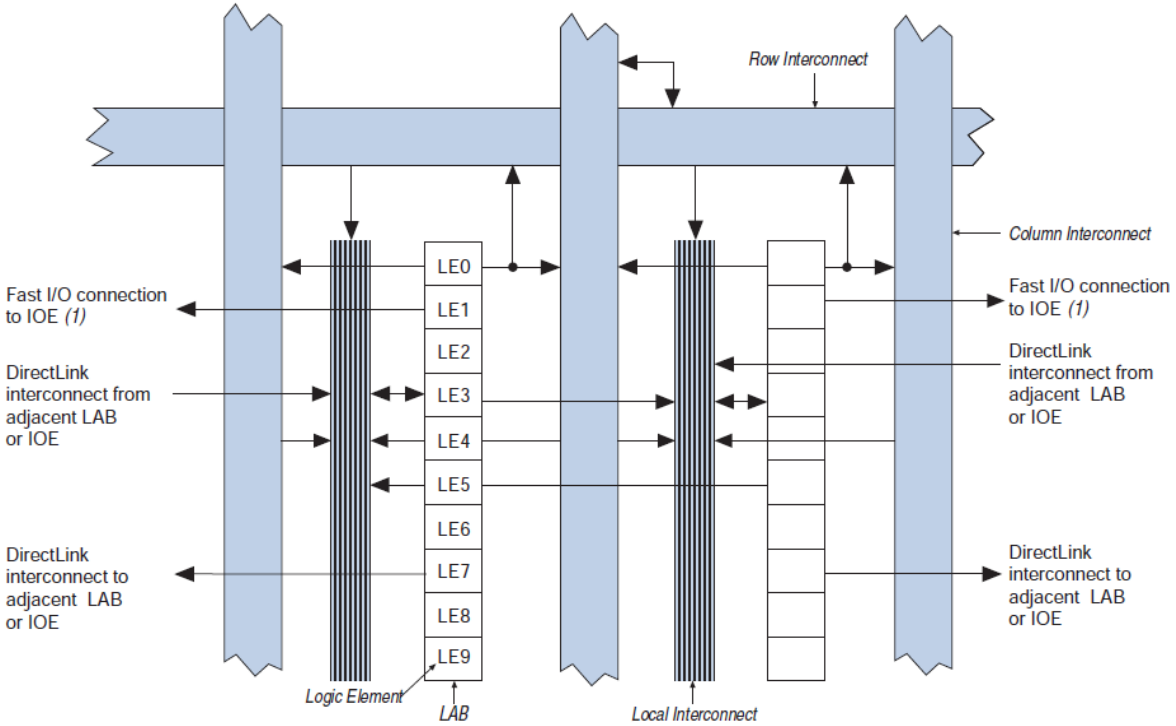
Example: Altera MAX II (2006)

Figure 2–2. MAX II Device Floorplan *(Note 1)*



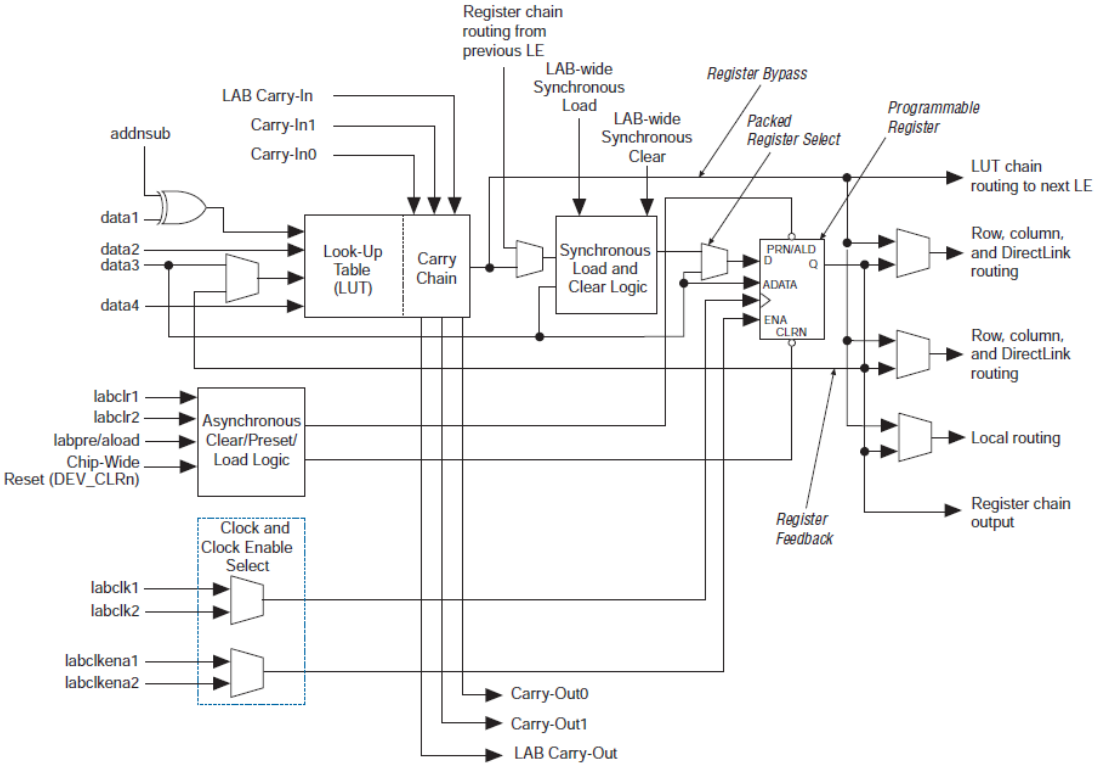
Example: Altera MAX II (2006)

Figure 2–3. MAX II LAB Structure



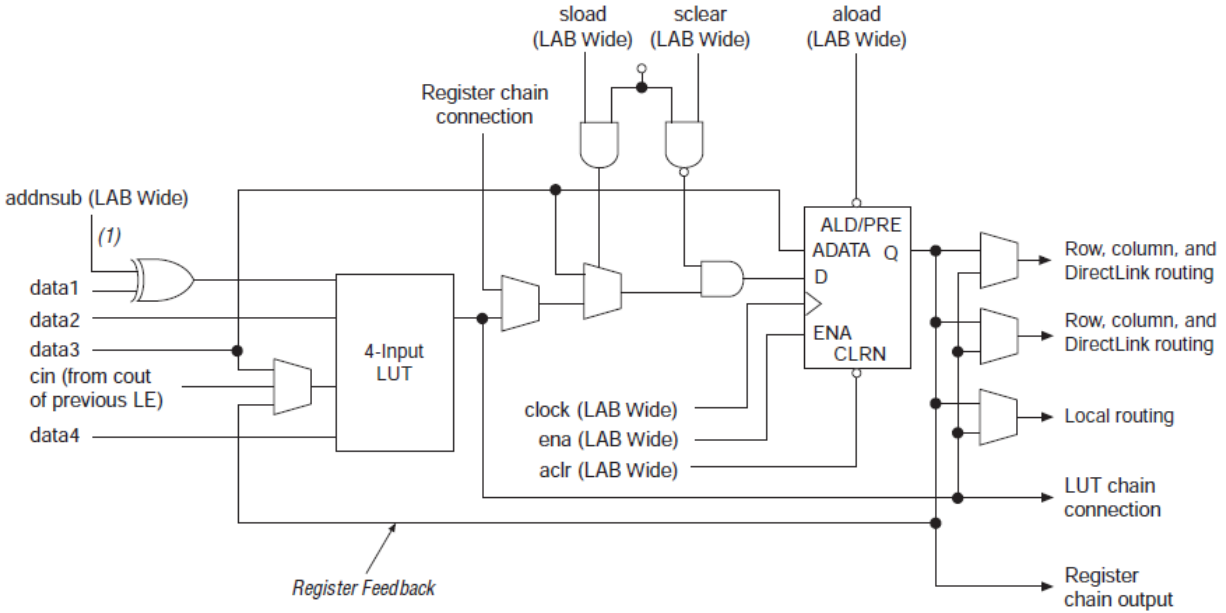
Example: Altera MAX II (2006)

Figure 2–6. MAX II LE



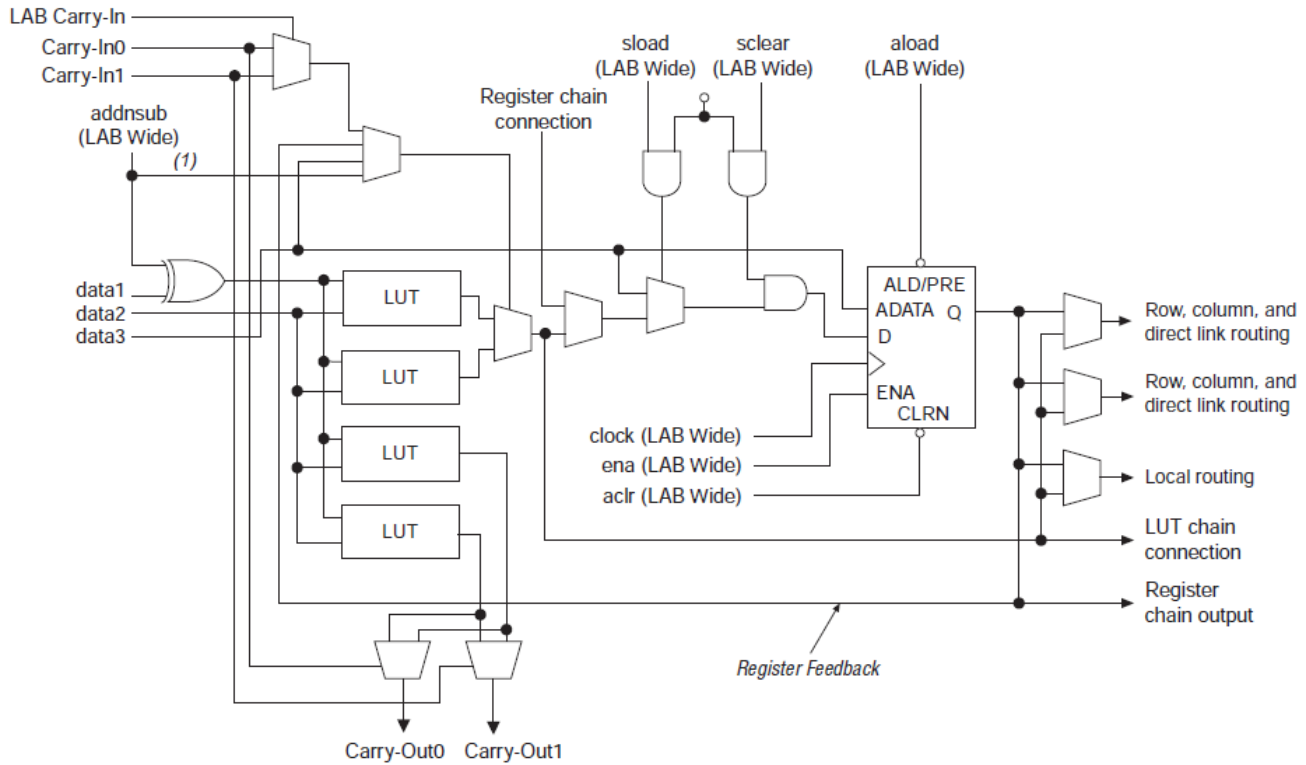
Example: Altera MAX II (2006)

Figure 2–7. LE in Normal Mode



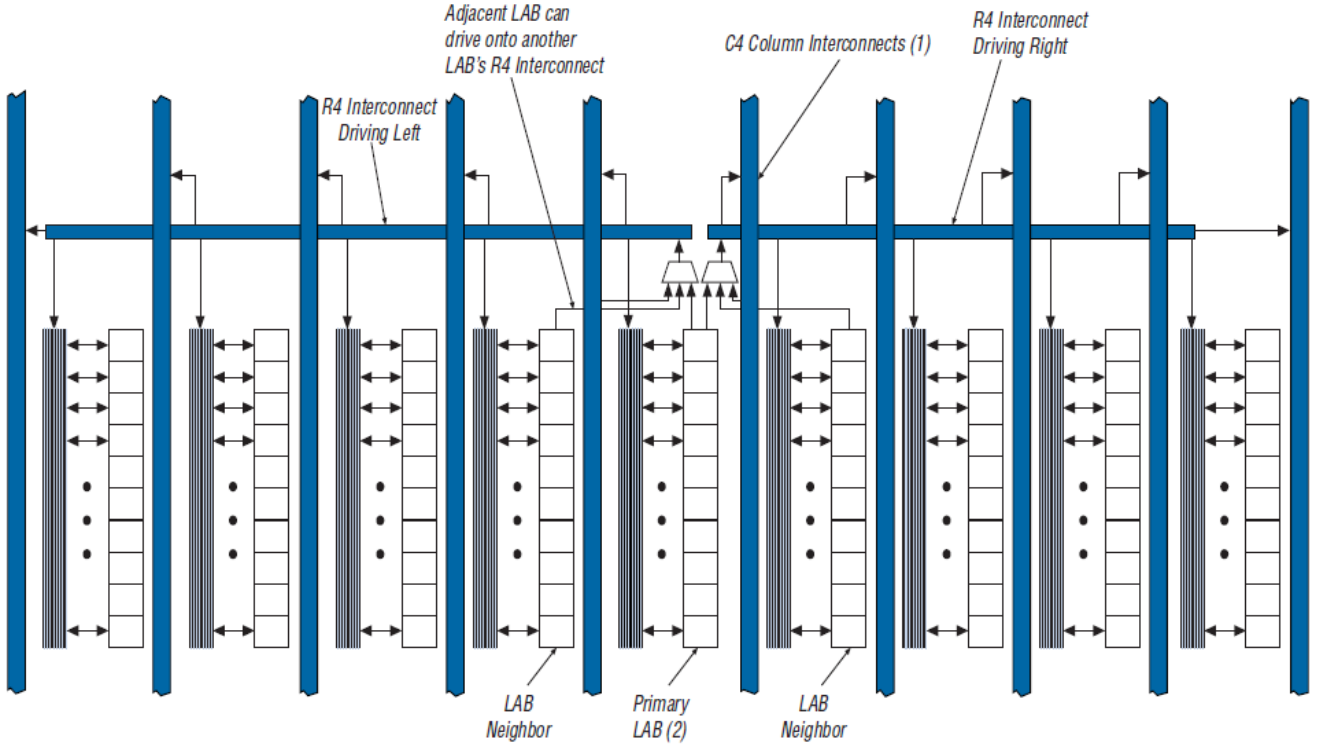
Example: Altera MAX II (2006)

Figure 2–8. LE in Dynamic Arithmetic Mode



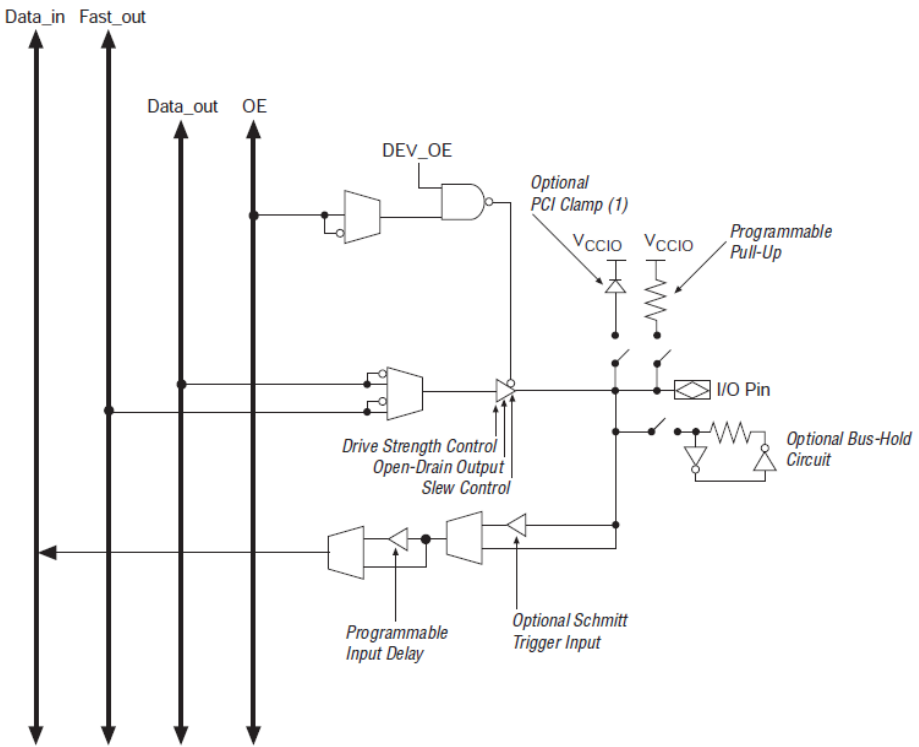
Example: Altera MAX II (2006)

Figure 2–10. R4 Interconnect Connections



Example: Altera MAX II (2006)

Figure 2–19. MAX II IOE Structure



Example: XILINX Spartan-3 (2006)

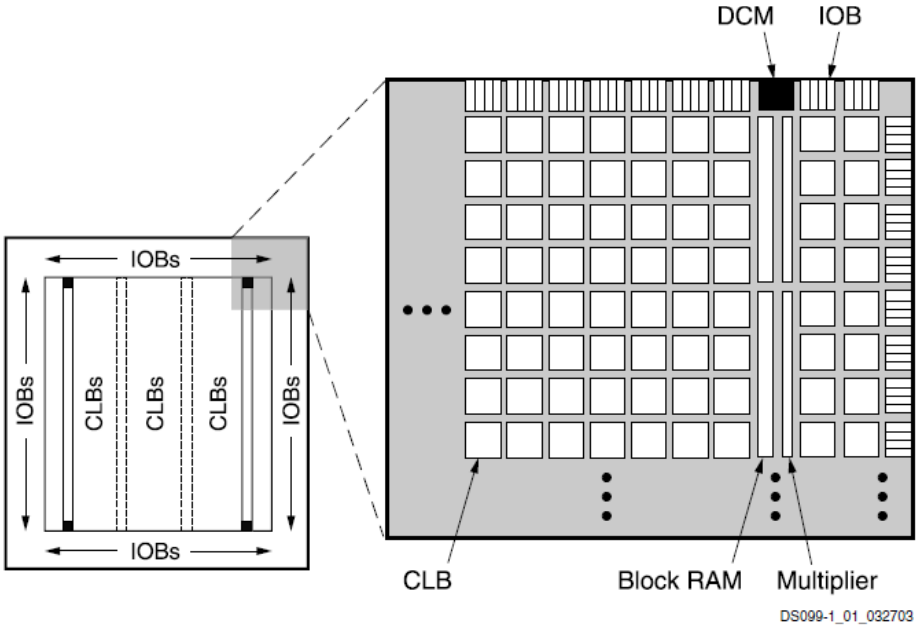
Table 1: **Summary of Spartan-3 FPGA Attributes**

Device	System Gates	Equivalent Logic Cells(1)	CLB Array (One CLB = Four Slices)			Distributed RAM Bits (K=1024)	Block RAM Bits (K=1024)	Dedicated Multipliers	DCMs	Max. User I/O	Maximum Differential I/O Pairs
			Rows	Columns	Total CLBs						
XC3S50(2)	50K	1,728	16	12	192	12K	72K	4	2	124	56
XC3S200(2)	200K	4,320	24	20	480	30K	216K	12	4	173	76
XC3S400(2)	400K	8,064	32	28	896	56K	288K	16	4	264	116
XC3S1000(2)	1M	17,280	48	40	1,920	120K	432K	24	4	391	175
XC3S1500	1.5M	29,952	64	52	3,328	208K	576K	32	4	487	221
XC3S2000	2M	46,080	80	64	5,120	320K	720K	40	4	565	270
XC3S4000	4M	62,208	96	72	6,912	432K	1,728K	96	4	633	300
XC3S5000	5M	74,880	104	80	8,320	520K	1,872K	104	4	633	300

Notes:

1. Logic Cell = 4-input Look-Up Table (LUT) plus a 'D' flip-flop. "Equivalent Logic Cells" equals "Total CLBs" x 8 Logic Cells/CLB x 1.125 effectiveness.
2. These devices are available in Xilinx Automotive versions as described in DS314: *Spartan-3 Automotive XA FPGA Family.*

Example: XILINX Spartan-3 (2006)



Notes:

1. The two additional block RAM columns of the XC3S4000 and XC3S5000 devices are shown with dashed lines. The XC3S50 has only the block RAM column on the far left.

Figure 1: **Spartan-3 Family Architecture**

Example: XILINX Spartan-3 (2006)

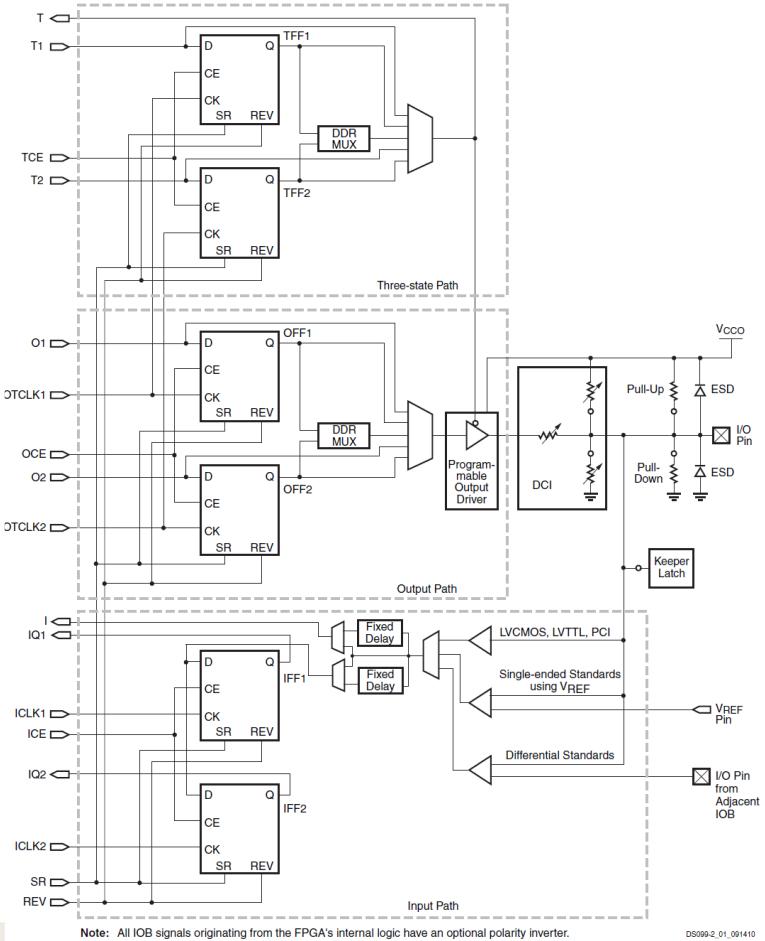


Figure 7: **Simplified IOB Diagram**

Example: XILINX Spartan-3 (2006)

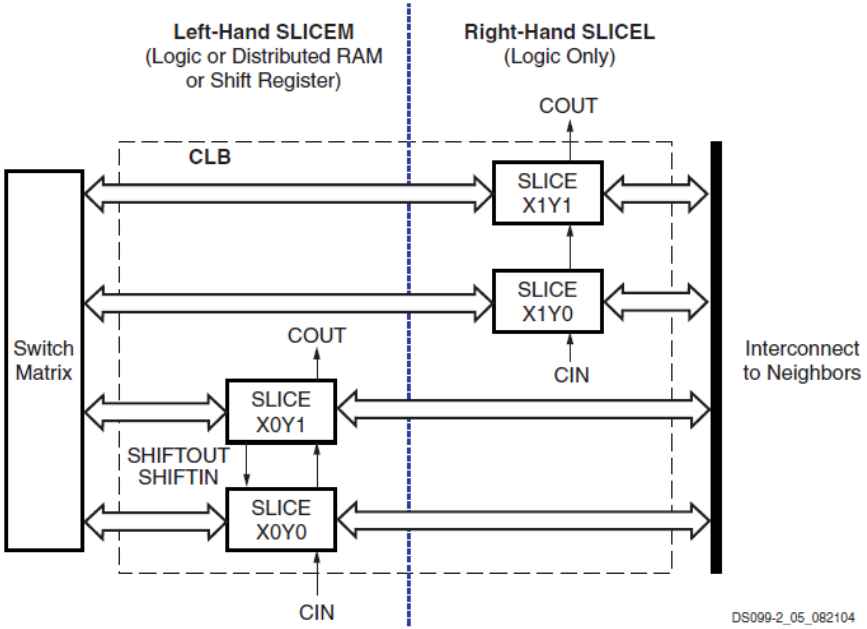
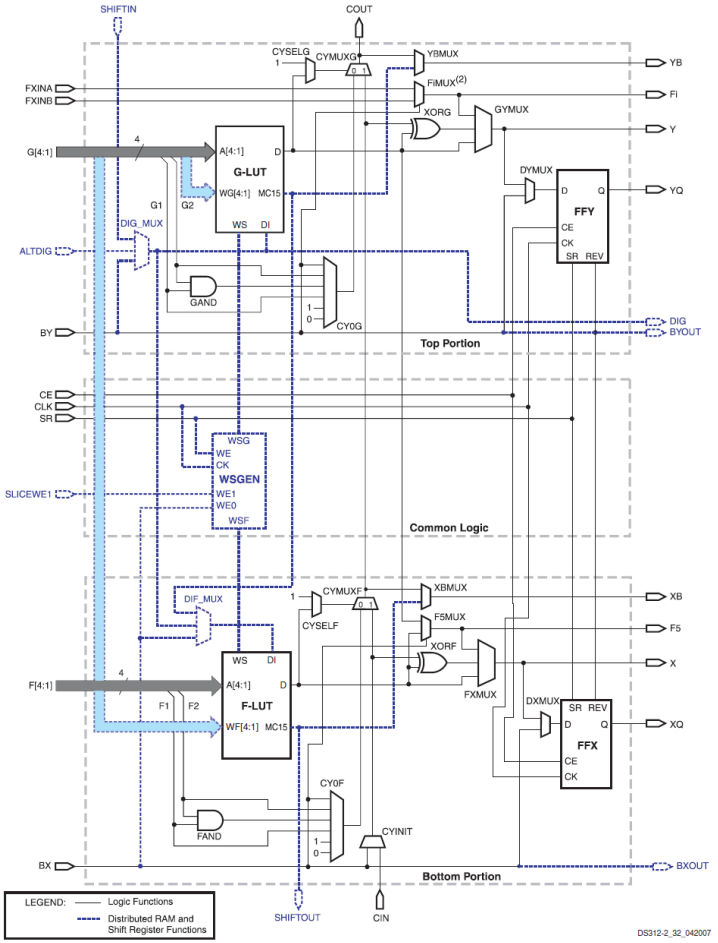


Figure 11: **Arrangement of Slices within the CLB**

Example: XILINX Spartan-3 (2006)



Notes:

1. Options to invert signal polarity as well as other options that enable lines for various functions are not shown.
2. The index i can be 6, 7, or 8, depending on the slice. In this position, the upper right-hand slice has an F8MUX, and the upper left-hand slice has an F7MUX. The lower right-hand and left-hand slices both have an F6MUX.

Figure 12: **Simplified Diagram of the Left-Hand SLICEM**

Example: XILINX Spartan-3 (2006)

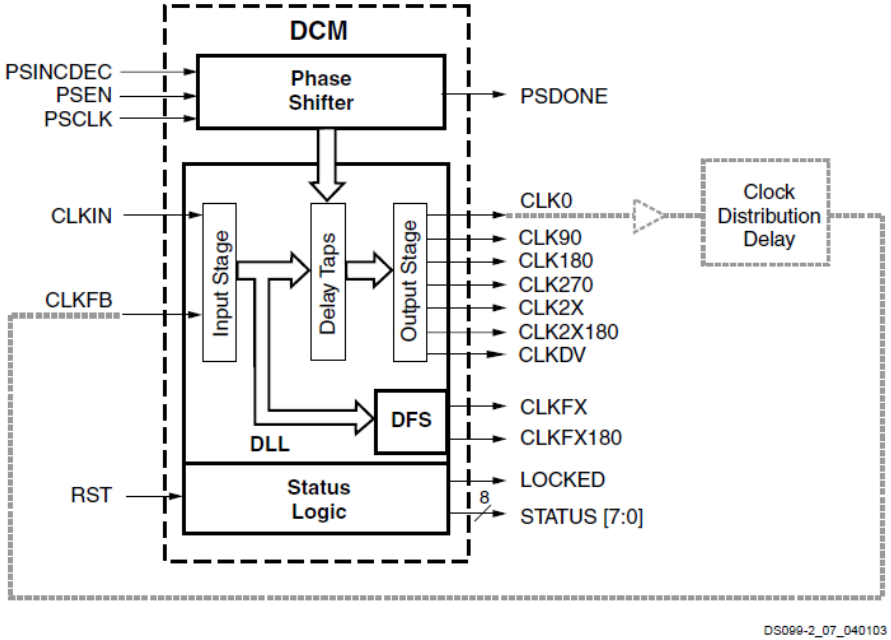


Figure 19: **DCM Functional Blocks and Associated Signals**

Example: Altera Cyclone V (2012)

Table 1: Key Advantages of the Cyclone V Device Family

Advantage	Supporting Feature
Lower power consumption	• Built on TSMC's 28 nm low-power (28LP) process technology and includes an abundance of hard intellectual property (IP) blocks • Up to 40% lower power consumption than the previous generation device
Improved logic integration and differentiation capabilities	• 8-input adaptive logic module (ALM) • Up to 13.59 megabits (Mb) of embedded memory • Variable-precision digital signal processing (DSP) blocks
Increased bandwidth capacity	• 3.125 gigabits per second (Gbps) and 5 Gbps transceivers • Hard memory controllers
Hard processor system (HPS) with integrated ARM® Cortex™-A9 MPCore processor	• Tight integration of a dual-core ARM Cortex-A9 MPCore processor, hard IP, and an FPGA in a single Cyclone V system-on-a-chip (SoC) FPGA • Supports over 128 Gbps peak bandwidth with integrated data coherency between the processor and the FPGA fabric
Lowest system cost	• Requires only two core voltages to operate • Available in low-cost wirebond packaging • Includes innovative features such as Configuration via Protocol (CvP) and partial reconfiguration

Example: Altera Cyclone V (2012)

Maximum Resources

Table 10: Maximum Resource Counts for Cyclone V SE Devices—Preliminary

Resource		Member Code			
		A2	A4	A5	A6
Logic Elements (LE) (K)		25	40	85	110
ALM		9,434	15,094	32,075	41,509
Register		37,736	60,376	128,300	166,036
Memory (Kb)	M10K	1,400	2,240	3,970	5,140
	MLAB	138	220	480	621
Variable-precision DSP Block		36	58	87	112
18 x 18 Multiplier		72	116	174	224
FPGA PLL[11]		4	5	6	6
HPS PLL		3	3	3	3
FPGA GPIO[12]		145	145	288	288
HPS I/O		188	188	188	188
LVDS[13]		31	31	72	72
FPGA Hard Memory Controller		1	1	1	1

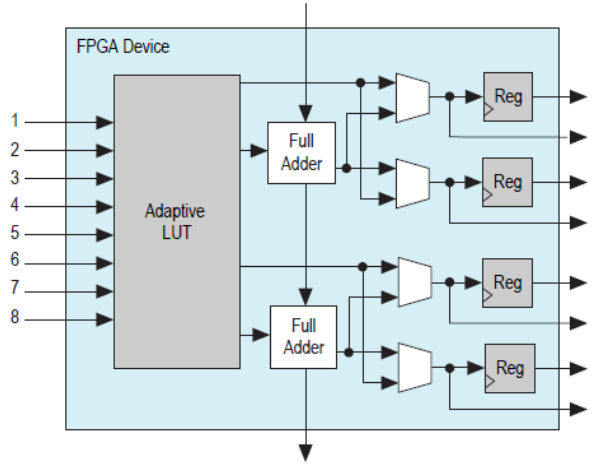
Example: Altera Cyclone V (2012)

Adaptive Logic Module

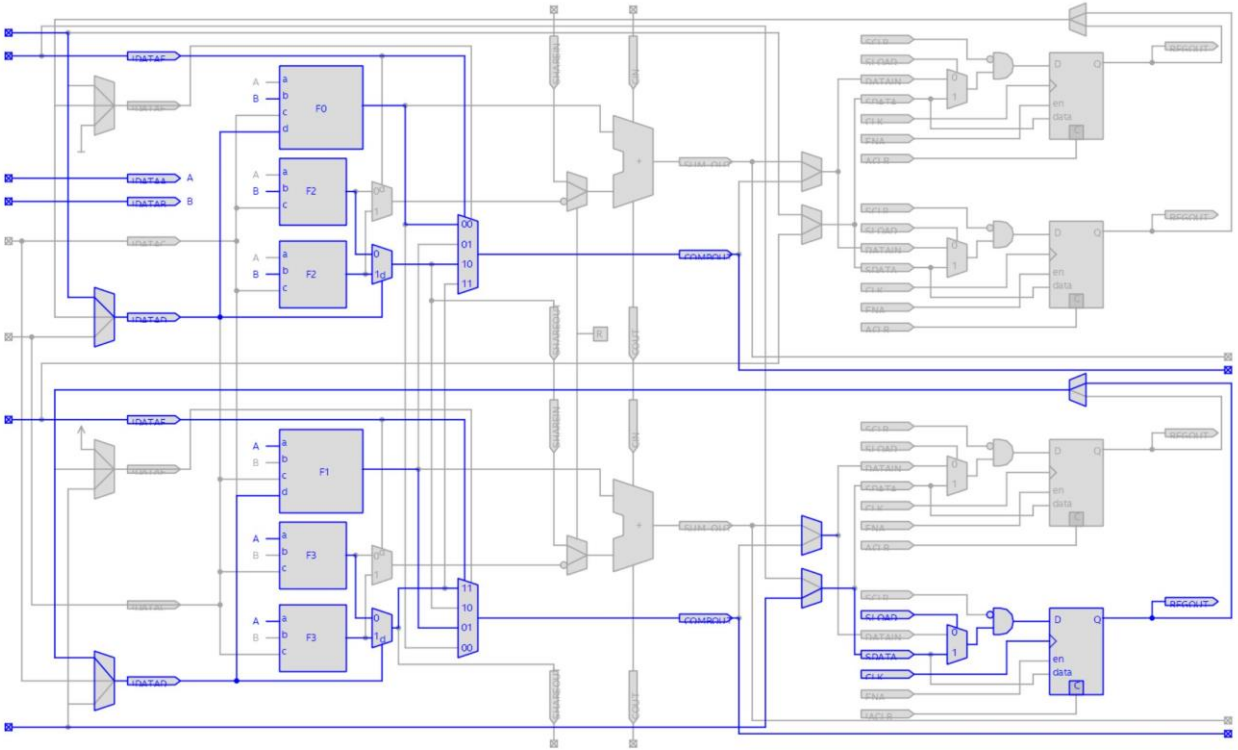
Cyclone V devices use a 28 nm ALM as the basic building block of the logic fabric.

The ALM, as shown in following figure, uses an 8-input fracturable look-up table (LUT) with four dedicated registers to help improve timing closure in register-rich designs and achieve an even higher design packing capability than previous generations.

Figure 8: ALM for Cyclone V Devices



Example: Altera Cyclone V (2012)



Example: Altera Cyclone V (2012)

Variable-Precision DSP Block

Cyclone V devices feature a variable-precision DSP block that supports these features:

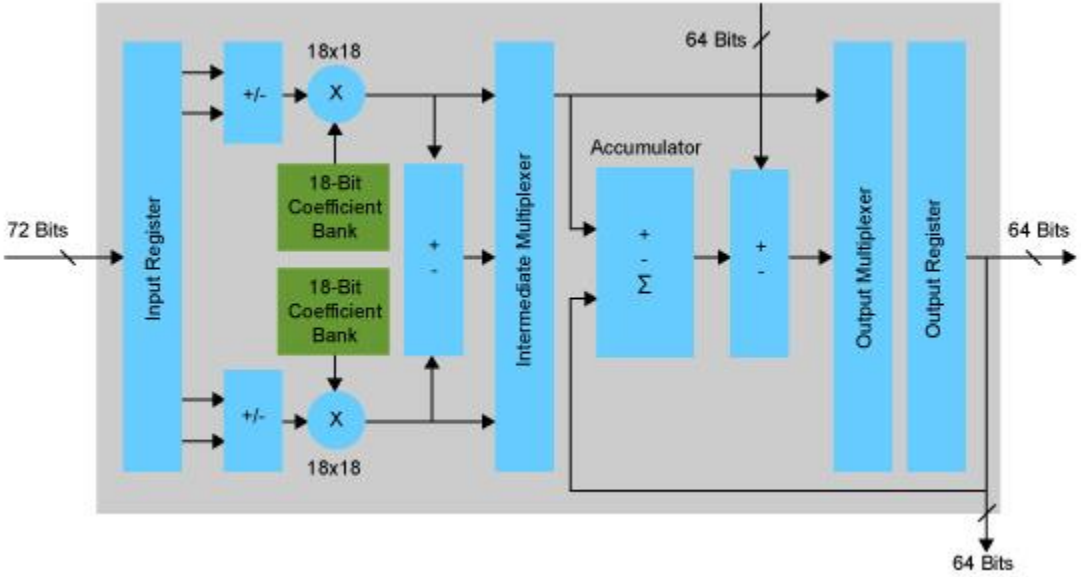
- Configurable to support signal processing precisions ranging from 9 x 9, 18 x 18, 27 x 27, and 36 x 36 bits natively
- A 64-bit accumulator
- A hard preadder that is available in both 18- and 27-bit modes
- Cascaded output adders for efficient systolic finite impulse response (FIR) filters
- Internal coefficient register banks, 8 deep, for each multiplier in 18- or 27-bit mode
- Fully independent multiplier operation
- A second accumulator feedback register to accommodate complex multiply-accumulate functions
- Efficient support for single-precision floating point arithmetic
- The inferability of all modes by the Quartus II design software

Example: Altera Cyclone V (2012)

Table 16: Variable-Precision DSP Block Configurations for Cyclone V Devices

Usage Example	Multiplier Size (Bit)	DSP Block Resource
Low precision fixed point for video applications	Three 9 x 9	1
Medium precision fixed point in FIR filters	Two 18 x 18	1
FIR filters and general DSP usage	Two 18 x 18 with accumulate	1
High precision fixed- or floating-point implementations	One 27 x 27 with accumulate	1

Example: Altera Cyclone V (2012)



Example: Altera Cyclone V (2012)

Types of Embedded Memory

The Cyclone V devices contain two types of memory blocks:

- 10 Kb M10K blocks blocks—blocks of dedicated memory resources. The M10K blocks blocks are ideal for larger memory arrays while still providing a large number of independent ports.
- 640 bit memory logic array blocks (MLABs)—enhanced memory blocks that are configured from dual-purpose logic array blocks (LABs). The MLABs are ideal for wide and shallow memory arrays. The MLABs are optimized for implementation of shift registers for digital signal processing (DSP) applications, wide shallow FIFO buffers, and filter delay lines. Each MLAB is made up of ten adaptive logic modules (ALMs). In the Cyclone V devices, you can configure these ALMs as ten 32 x 2 blocks, giving you one 32 x 20 simple dual-port SRAM block per MLAB.

Example: Altera Cyclone V (2012)

Clock Networks and PLL Clock Sources

Cyclone V devices have 16 global clock networks capable of up to 550 MHz operation. The clock network architecture is based on Altera's global, quadrant, and peripheral clock structure. This clock structure is supported by dedicated clock input pins and fractional PLLs.

Note: To reduce power consumption, the Quartus II software identifies all unused sections of the clock network and powers them down.

PLL Features

The PLLs in the Cyclone V devices support the following features:

- Frequency synthesis
- On-chip clock deskew
- Jitter attenuation
- Counter reconfiguration
- Programmable output clock duty cycles
- PLL cascading
- Reference clock switchover
- Programmable bandwidth
- User-mode reconfiguration of PLLs

Example: Altera Cyclone V (2012)

Low-Power Serial Transceivers

Cyclone V devices deliver the industry's lowest power 5 Gbps transceivers at an estimated 88 mW maximum power consumption per channel. Cyclone V transceivers are designed to be compliant with a wide range of protocols and data rates.

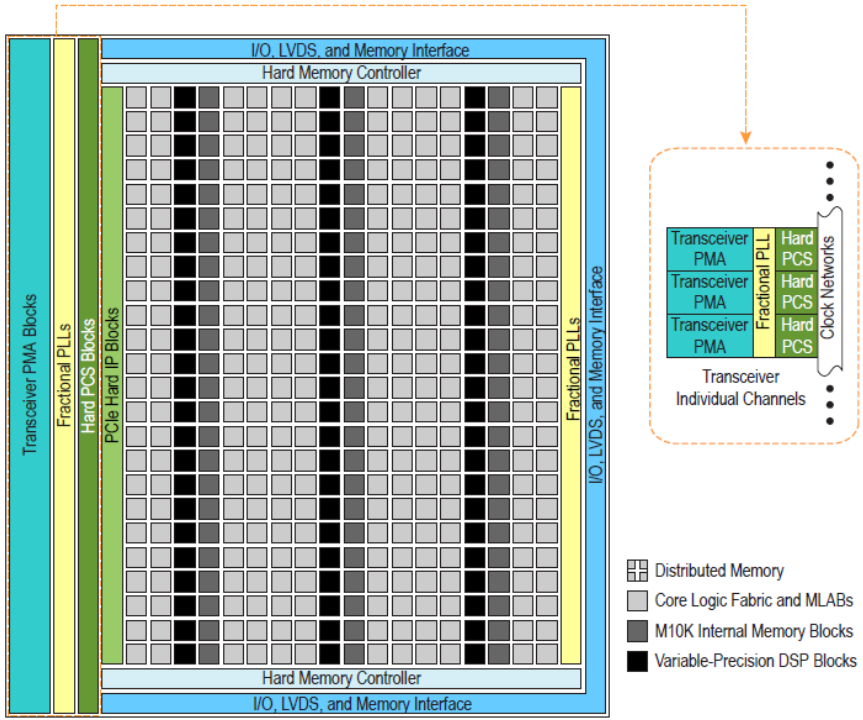
Transceiver Channels

The transceivers are positioned on the left outer edge of the device. The transceiver channels consist of the physical medium attachment (PMA), physical coding sublayer (PCS), and clock networks.

Example: Altera Cyclone V (2012)

Figure 10: Device Chip Overview for Cyclone V GX and GT Devices

The figure shows a Cyclone V FPGA with transceivers. Different Cyclone V devices may have a different floorplans than the one shown here.



PMA Physical Medium attachment
PCS Physical coding sublayer

Example: Altera Cyclone V (2012)

SoC FPGA with HPS

Each SoC FPGA combines an FPGA fabric and an HPS in a single device. This combination delivers the flexibility of programmable logic with the power and cost savings of hard IP in these ways:

- Reduces board space, system power, and bill of materials cost by eliminating a discrete embedded processor
- Allows you to differentiate the end product in both hardware and software, and to support virtually any interface standard
- Extends the product life and revenue through in-field hardware and software updates

Example: Intel AGILEX (2019)

- Curretly, the most performant FPGA of Altera/Intel.
- «Intel® Agilex™ FPGA family leverages heterogeneous 3D system-in-package (SiP) technology to integrate Intel's first FPGA fabric built on 10nm process technology and 2nd Gen Intel® Hyperflex™ FPGA Architecture to deliver up to 40% higher performance[1] or up to 40% lower power[1] for applications in Data Center, Networking, and Edge compute. Intel® Agilex™ SoC FPGAs also integrate the quad-core Arm* Cortex-A53 processor to provide high system integration.»

This Comparison based on Intel® Agilex™ FPGA and SoC family vs. Intel® Stratix® 10 FPGA using simulation results and is subject to change.

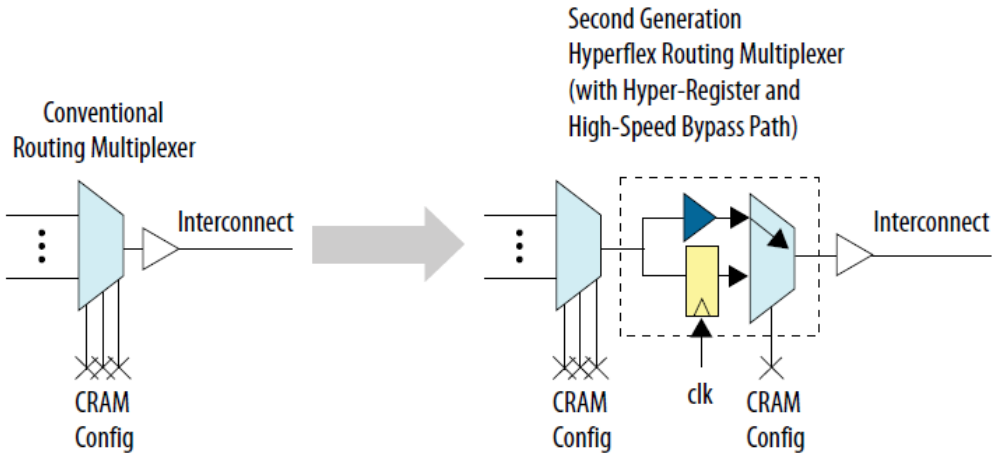
Example: Intel AGILEX (2019)

INTEL® AGILEX™ F-SERIES FPGA AND SOC FPGA PRODUCT TABLE

	PRODUCT LINE	AGF 004	AGF 006	AGF 008	AGF 012	AGF 014	AGF 022	AGF 027
Resources	Logic elements (LEs)	392,000	573,480	764,640	1,200,000	1,437,240	2,200,000	2,692,760
	Adaptive logic modules (ALMs)	132,881	194,400	259,200	406,780	487,200	745,763	912,800
	ALM registers	531,525	777,600	1,036,800	1,627,119	1,948,800	2,983,051	3,651,200
	eSRAM memory blocks	0	0	0	2	2	0	0
	eSRAM memory size (Mb)	0	0	0	36	36	0	0
	M20K memory blocks	1,900	2,844	3,792	5,568	7,110	11,616	13,272
	M20K memory size (Mb)	38	56	74	110	139	210	259
	MLAB memory count	6644	9720	12960	20,338	24,360	32,788	45,640
	MLAB memory size (Mb)	4.3	6.2	8.3	13	15.6	21	29.2
	Variable-precision digital signal processing (DSP) blocks	1,640	1,640	2,296	4,000	4,510	6,250	8,736
	18 x 19 multipliers	2,300	3,280	4,592	8,000	9,020	12,500	17,056
	Single-precison or half-precision tera floating point operations per second (TFLOPS)	1.7 / 3.4	2.5 / 5.0	3.5 / 6.9	6.0 / 12.0	7.0 / 13.9	9.4 / 18.8	11.8 / 23.6
Maximum Available Device Resources	Maximum differential (RX or TX) pairs	192	192	288	384	384	384	384
	Memory devices supported	DDR4, QDR IV, RLDRAM 3						
	Secure data manager	AES-256/SHA-256 bitstream encryption/authentication, physically unclonable function (PUF), ECDSA 256/384 boot code authentication, side channel attack protection						
	Hard processor system	Quad-core 64 bit Arm* Cortex*-A53 up to 1.5 GHz with 32 KB I/D cache, NEON* coprocessor, 1 MB L2 cache, direct memory access (DMA), system memory management unit, cache coherency unit, hard memory controllers, USB 2.0x2, 1G EMAC x3, UART x2, serial peripheral interface (SPI) x4, I2C x5, general purpose timers x7, watchdog timer x4						
F-Tile Device Resources	F-Tile PCI Express* (PCIe*) hard IP blocks (Gen4x16) or bifurcateable 2X PCIe Gen4 x8 (EP) or 4X Gen4 x4 (RP)	1	1	1	1	1	1	1
	F-Tile transciever channel count PAM4 (up to 58 Gbps) - RS and KP forward error correction (FEC) Non-return-to-zero (NRZ) (up to 32 Gbps)	24x PAM-4 32x NRZ	24x PAM-4 32x NRZ	24x PAM-4 32x NRZ	24x PAM-4 32x NRZ	24x PAM-4 32x NRZ	48x PAM-4 64x NRZ	48x PAM-4 64x NRZ
	F-Tile 10/25/50/100/200/400G Ethernet MAC + FEC hard intellectual property (IP) blocks	2	2	2	2	2	4	4
E-Tile / P-Tile Device Resources	P-Tile PCIe hard IP blocks (Gen4x16) or bifurcateable 2X PCIe Gen4 x8 (EP) or 4X Gen4 x4 (RP)	–	–	1	1	1	1	1
	E-Tile transceiver channel count PAM4 (up to 57.8 Gbps) - RS and KP FEC NRZ (up to 28.9 Gbps)	–	–	12x PAM-4 24x NRZ	12x PAM-4 24x NRZ	12x PAM-4 24x NRZ	12x PAM-4 24x NRZ	12x PAM-4 24x NRZ
	E-Tile 100G Ethernet MAC + FEC hard IP blocks	–	–	4	4	4	4	4

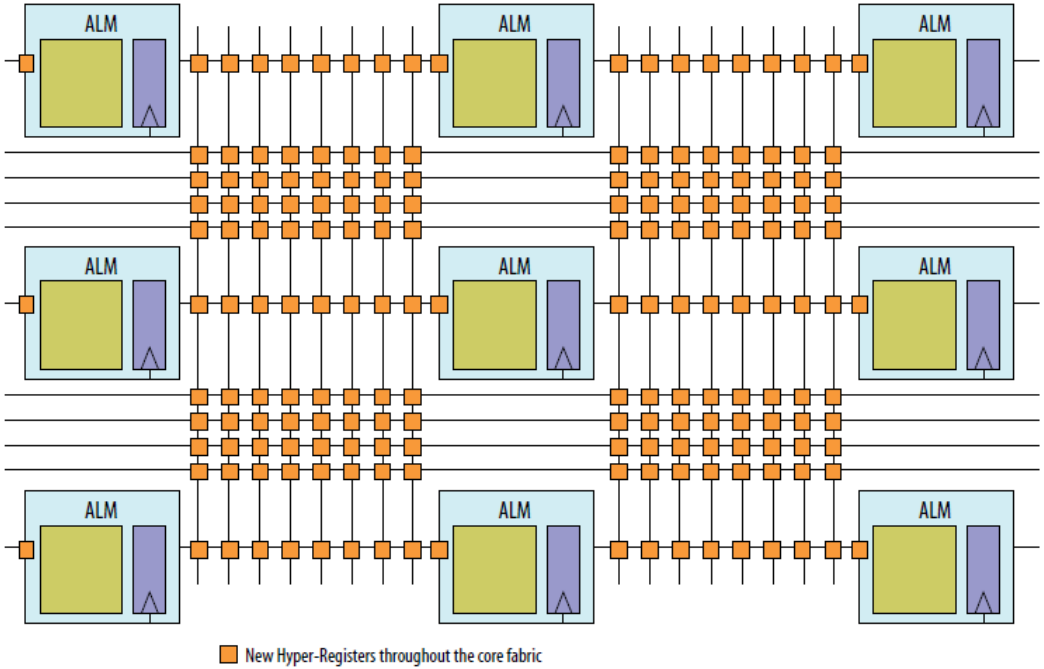
Example: Intel AGILEX (2019)

Figure 3. Bypassable Hyper Register



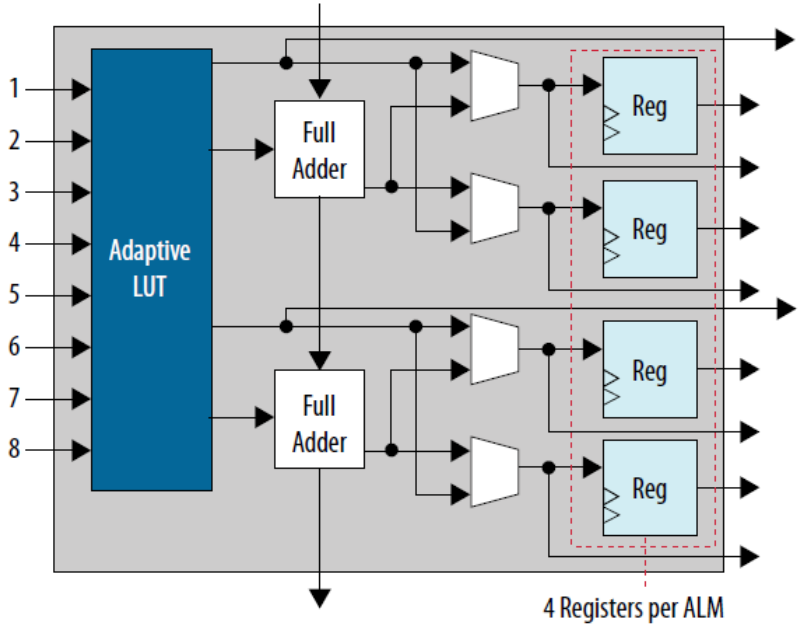
Example: Intel AGILEX (2019)

Figure 4. Intel Hyperflex Core Architecture



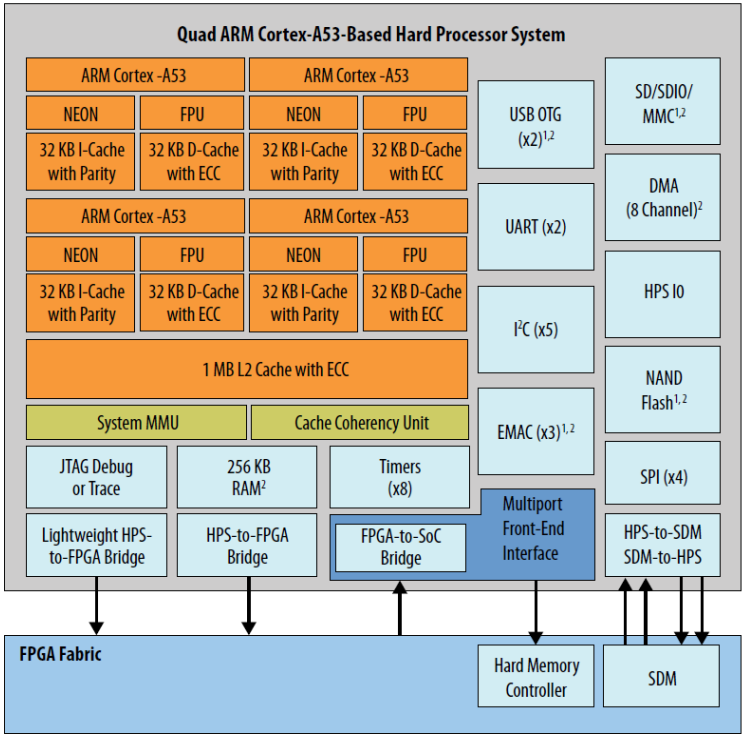
Example: Intel AGILEX (2019)

Figure 7. ALM Block Diagram



Example: Intel AGILEX (2019)

Figure 13. HPS Block Diagram



Notes:
1. Integrated direct memory access (DMA)
2. Integrated error correction code (ECC)

Direct design of application specific integrated circuits (ASICs)

- Allow to optimize the performances of the circuit in terms of propagation delay, power dissipation and area occupation.
- *Standard cell design*
  - Implement the system by means of standard cells.
  - Project similar to that of standard logic circuits, but on the silicon plane now.
- *Full custom design*
  - Every single transistor is optimized.
  - Maximize performance but the design cost is high.
  - Used only for very large volume productions, e.g., for processors, memories, FPGAs, etc.

Standard cell library

Cell function	Characteristics
inverter/buffer/tristate	
NAND/AND gate	2-8 inputs
NOR/OR gate	2-8 inputs
XOR/XNOR gate	2-8 inputs
AOI gate	2-8 inputs
Multiplexer	2-8 inputs
Decoder	2-8 inputs
Adder	normal or fast
latch	D, asynchronous, synchronous
Register	D, JK, asynchronous, synchronous
IO circuits	In, out, inout, tristate
ROM	

What Intel says about FPGAs and ASICs

Comparing FPGAs and ASICs

FPGAs are generally more flexible and cost-effective than ASICs. In the following sections, I explain why.

Costs and flexibility

Using FPGAs, you can implement any logical function that an ASIC can do but with the distinct advantage of updating the functionality after chip manufacture, which is desirable for many applications. FPGAs are more cost-effective than ASICs because the customer can program FPGAs according to its requirements instead of contracting a vendor to design and construct an ASIC to meet its needs.

What Intel says about FPGAs and ASICs

Design time risk reduction versus speed

If you set out to use the most advanced semiconductor process in the world, no matter the cost, you could always design an ASIC that would run faster than the fastest available FPGA. But almost no one uses the most advanced process: Doing so would be risky, very difficult, and witheringly expensive. In fact, only a handful of ASSP companies leap on a new process as soon as it's available. Everyone else uses a process that is one, two, or three generations old. And the fact is, the fastest FPGA you can get can compete directly with those older ASIC processes. And the FPGA brings reduced design work and far less risk.

If, for example, you're designing a system with specific power efficiency and performance requirements and are planning to use an older 65 nanometer (nm) ASIC, did you know you can achieve similar results with a current 20 nm FPGA?

What Intel says about FPGAs and ASICs

And using the FPGA would shorten your design time, reduce your risk of design errors, and offer a lower total cost of ownership (TCO) than the ASIC. For most applications, the FPGA's power consumption will be acceptable for your needs. Therefore, due to their lower TCO and greater flexibility, FPGAs are often the best technological choice.

Choosing an FPGA for a system offers the designer greater configurability as well as less risk of impact to the development schedule because, as demonstrated by the building blocks analogy, small parts of FPGAs can be modified without impact to the rest of the design.

See:

- Paolo Spirito, «Elettronica digitale» McGraw-Hill, seconda edizione
  - Chapter 12