Designing With Programmable Logic Devices

Programmable Logic Devices

Programmable Logic Device (PLD) is an integrated circuit with internal logic gates and/or connections that can in some way be changed by a programming process.

Examples:

- PROM
- Programmable Logic Array (PLA)
- Programmable Array Logic (PAL) device
- Complex Programmable Logic Device (CPLD)
- Field-Programmable Gate Array (FPGA)

A PLD's function is not fixed

- Can be programmed to perform different functions

Why PLDS?

Fact:

- It is most economical to produce an IC in large volumes

But:

- Many situations require only small volumes of ICs
- Many situations require changes to be done in the field, e.g. Firmware of a product under development

A programmable logic device can be:

- Produced in large volumes
- Programmed to implement many different low-volume design

PLD

- Programmable logic technology advances rapidly, and manufacturers are continually offering devices with increased capabilities and speeds.
- Major players in PLDs
  - Altera
  - Atmel
  - Cypress
  - Lattice
  - QuickLogic
  - Xilinx

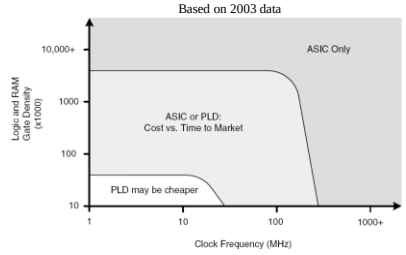
PLD

- PLDs are an alternative to custom ASICs.
- A PLD consists of general-purpose logic resources that can be connected in many permutations according to an engineer's logic design.
- The main benefit of PLD technology is that a design can be rapidly loaded into a PLD, bypassing the time consuming and expensive custom IC development process.

PLD

- Disadvantage of PLDs
  - The penalty paid for the hidden logic that implements the programmable connectivity between logic gates.
    - higher unit cost
    - slower speeds
    - increased power consumption

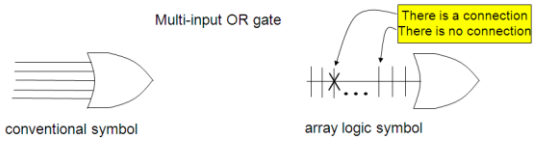
PLD vs. ASIC



PLD Hardware Programming Technologies

- In the Factory - Cannot be erased/reprogrammed by user
  - Mask programming (changing the VLSI mask) during manufacturing
- Programmable only once
  - Fuse
  - Anti-fuse
- Reprogrammable (Erased & Programmed many times)
  - Volatile - Programming lost if chip power lost
    - Single-bit storage element
  - Non-Volatile - Programming survives power loss
    - UV Erasable
    - Electrically Erasable
      - Flash (as in Flash Memory)

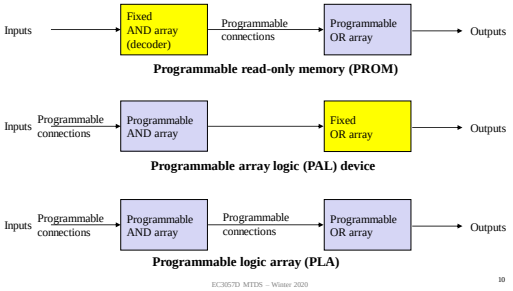
Used symbol in PLD



- Most PLD technologies have gates with very high fan-in
- Fuse map: graphic representation of the selected connections

Programmable Logic Devices (PLDs)

All use AND-OR structure- differ in which is programmable



Read-Only Memory (ROM)

- **ROM:** A device in which "permanent" binary information is stored using a special device (programmer)

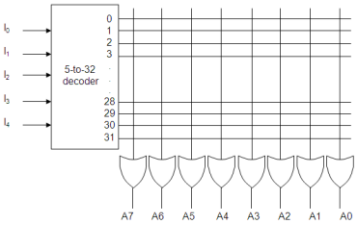
- k inputs (address) → 2^k words each of size n bits (data)
- ROM DOES NOT have a write operation → ROM DOES NOT have data inputs

Word: group of bits stored in one location

ROM Internal Logic

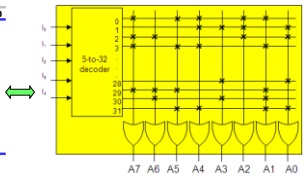
- The decoder stage produces ALL possible minterms
- 32 Words of 8 bits each
- 5 input lines (address)
- Each OR gate has a 32 input
- A contact can be made using fuse/anti-fuse

Internal Logic of a 32x8 ROM



Programming a ROM

Inputs					Outputs							
I4	I3	I2	I1	I0	A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1
0	0	0	1	1	1	0	1	1	0	0	1	0
.								.				
.								.				
.								.				
1	1	1	0	0	0	0	0	0	1	0	0	1
1	1	1	0	1	1	1	1	0	0	0	1	0
1	1	1	1	0	0	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1	1	0	0	1	1



- Every ONE in truth table specifies a closed circuit
- Every ZERO in truth table specifies an OPEN circuit
- Example: At address 00011 → The word 10110010 is stored

Combinational Circuit Implementation with ROM

- ROM = Decoder + OR gates
- Implementation of a combinational circuit is easy
  - Store the truth table by programming the ROM
- Only need to provide the truth table

Example 1

Example: Design a combinational circuit using ROM. The circuit accepts a 3-bit number and generates an output binary number equal to the square of the number.

Solution: Derive truth table:

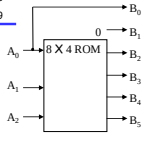
Inputs			Outputs						
A2	A1	A0	B5	B4	B3	B2	B1	B0	SQ
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

Example 1 (cont.)

Inputs			Outputs						
A2	A1	A0	B5	B4	B3	B2	B1	B0	SQ
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

ROM truth table – specifies the required connections

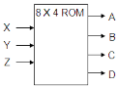
B1 is ALWAYS 0 → no need to generate it using the ROM
B0 is equal to A0 → no need to generate it using the ROM
Therefore: The minimum size of ROM needed is 2^3X4 or 8X4



Example 2

Problem: Tabulate the truth for an 8 X 4 ROM that implements the following four Boolean functions:
A(X,Y,Z) = Σ*m*(3,6,7); B(X,Y,Z) = Σ*m*(0,1,4,5,6)
C(X,Y,Z) = Σ*m*(2,3,4); D(X,Y,Z) = Σ*m*(2,3,4,7)

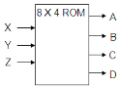
Solution:



Example 2

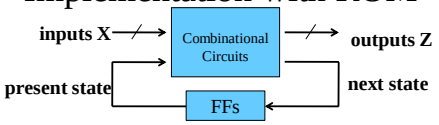
Problem: Tabulate the truth for an 8 X 4 ROM that implements the following four Boolean functions:
A(X,Y,Z) = Σ*m*(3,6,7); B(X,Y,Z) = Σ*m*(0,1,4,5,6)
C(X,Y,Z) = Σ*m*(2,3,4); D(X,Y,Z) = Σ*m*(2,3,4,7)

Solution:



Inputs			Outputs			
X	Y	Z	A	B	C	D
0	0	0	0	1	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	1
0	1	1	1	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	0	0
1	1	0	1	1	0	0
1	1	1	1	0	0	1

Sequential Circuit Implementation with ROM



- sequential circuit = combinational circuit + memory
- Combinational part can be built with a ROM as shown previously
  - Number of address lines = No. of FF + No. of inputs
  - Number of outputs = No. of FF + No. of outputs

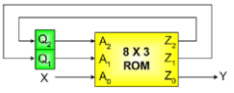
Example

Example: Design a sequential circuit whose state table is given, using a ROM and a register.

State Table

Present State		Input	Next State		Output
Q_2	Q_1	X	Q_2^+	Q_1^+	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	0	1
1	0	0	1	0	0
1	0	1	0	1	0
1	1	0	1	1	0
1	1	1	0	0	1

We need a 8x3 ROM (why?)
3 address lines and 3 data lines



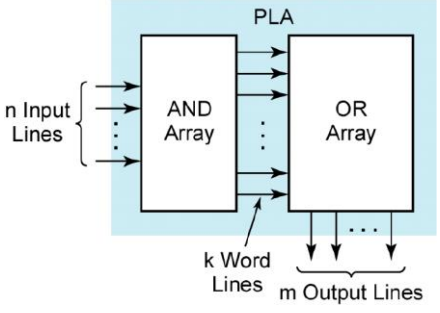
Types of ROMs

- A ROM programmed in four different ways:
- ROM: Mask Programming
  - By a semiconductor company
- PROM (Programmable ROM)
  - User can blow/connect fuses with a special programming device (PROM programmer)
  - Only programmed once!
- EPROM (Erasable PROM)
  - Can be erased using Ultraviolet Light
- Electrically Erasable PROM (EEPROM or E^2PROM)
  - Like an EPROM, but erased with electrical signal

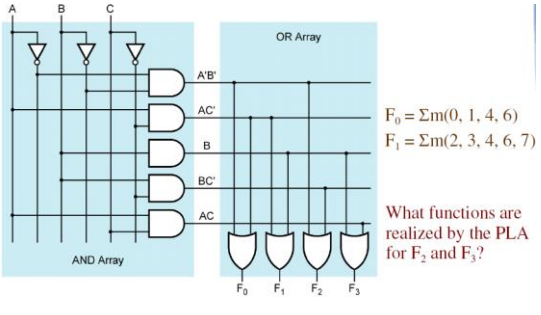
Programmable Logic Array

- A Programmable Logic Array (PLA) performs the same basic function as the ROM.
- A PLA with *n inputs and m outputs can realize*
  - *m functions*
  - of *n variables*
- A PLA consists of
  - An AND array to realize product terms
  - An OR array to realize the output functions
- Thus, a PLA implements SOP expressions.

PLA Basic Structure



Building Logic Functions with PLA

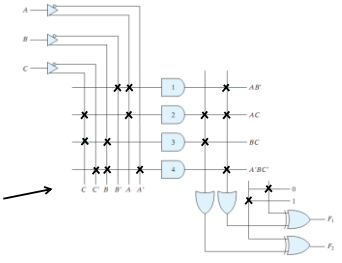


$F_0 = \Sigma m(0, 1, 4, 6)$
$F_1 = \Sigma m(2, 3, 4, 6, 7)$

What functions are realized by the PLA for F_2 and F_3?

Programmable Logic Array (PLA)

- AND array and OR array are programmable
- XOR is available to complement an output if needed

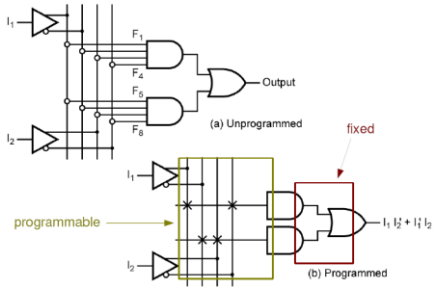


Example:
- 3 inputs/2 outputs
- F_1 = A B' + A C + A' B C'
- F2 = (AC + BC)'

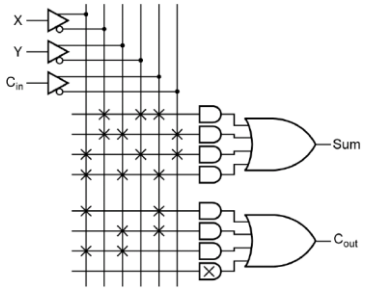
PAL

- The Programmable Array Logic (PAL) is a special case of the PLA
  - AND array is programmable
  - OR array is fixed
- A PAL is less expensive than the more general PLA.

Building Logic Functions with PLA



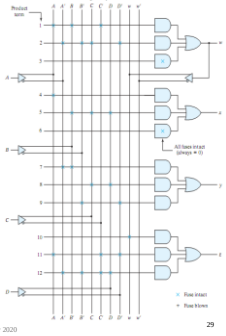
Full adder using PAL



Programmable Array Logic (PAL)

- Fixed OR array and programmable AND array
  - Opposite of ROM
- Feed back is used to support more product terms
- AND output can not be shared here!

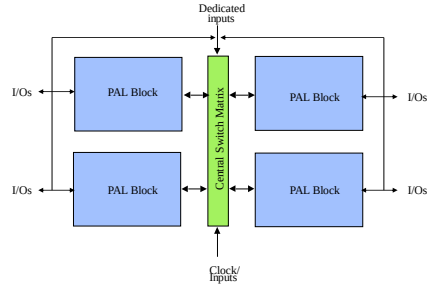
Example:
- 4 inputs/4 outputs with fixed 3-input OR gates
- W = A B C' + A' B' C D'
- X = ?
- Y = ?
- Z = ?



CPLD

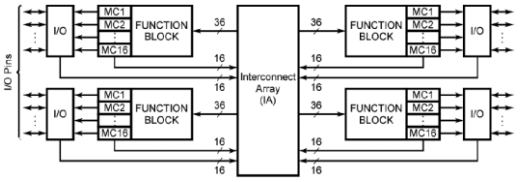
- A Complex Programmable Logic Device integrates many PLAs (or PALs) onto a single chip.
- In addition to the individual PLAs (or PALs) being programmable, the interconnection between these components is also programmable.
- A small digital system can be realized using
  - A single CPLD
  - Necessary memory elements (i.e. flip-flops)

CPLD Architecture



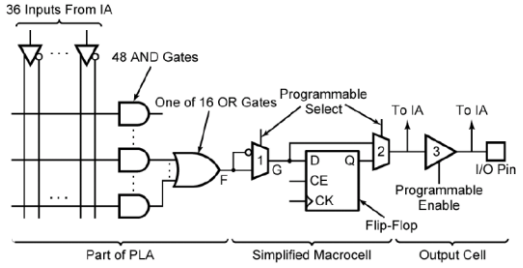
The CPLD is an array of PAL-like devices, interconnected by a switch matrix.

CPLD



Architecture of the Xilinx XCR3064XL CPLD

CPLD



CPLD Function Block and Macrocell
(A Simplified Version of XCR3064XL)

FPGA

What is an FPGA?

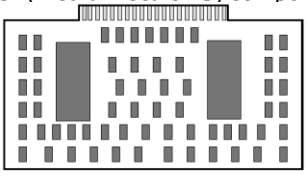
- **F**ield **P**rogrammable **G**ate **A**rrays
- Field programmability is achieved through switches (Transistors are controlled by memory elements or fuses)
- Switches control the following aspects
  - Interconnection among wire segments
  - Configuration of logic blocks

Why FPGAs?

- By the early 1980's most of the logic circuits in typical systems where absorbed by a handful of standard large scale integrated circuits (LSI).
  - Microprocessors, bus/IO controllers, system timers,…..
- Every system still had the need for random "glue logic" to help connect the large ICs:
  - generating global control signals (for resets etc.)
  - data formatting (serial to parallel, multiplexing, etc.)

Why FPGAs?

- Systems had a few LSI components and lots of small low density SSI (small scale IC) and MSI (medium scale IC) components.



Why FPGAs?

- Custom ICs where sometimes designed to replace the large amount of glue logic:
  - reduced system complexity and manufacturing cost, improved performance.
  - However, custom ICs are relatively very expensive to develop, and delay in introduction of product to market (time to market) because of increased design time.
- Note: need to worry about two kinds of costs:
  - cost of development, sometimes called non-recurring engineering (NRE)
  - cost of manufacture

Why FPGAs?

- Therefore the custom IC approach was only viable for products with very high volume (where NRE could be amortized), and which were not time to market sensitive.
- FPGAs were introduced as an alternative to custom ICs for implementing glue logic:
  - improved density relative to discrete SSI/MSI components (within around 10x of custom ICs)
  - with the aid of computer aided design (CAD) tools circuits could be implemented in a short amount of time (no physical layout process, no mask making, no IC manufacturing)

Two competing implementation approaches

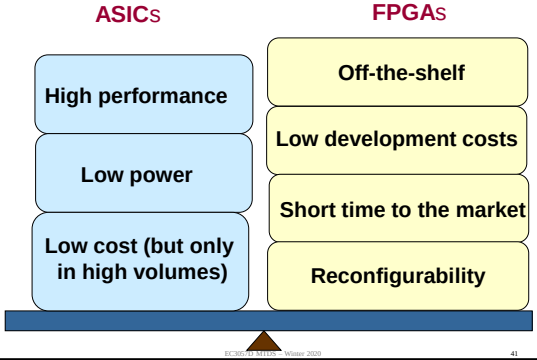
ASIC
Application **S**pecific **I**ntegrated **C**ircuit

- designed all the way from behavioral description to physical layout
- designs must be sent for expensive and time consuming fabrication in semiconductor foundry

FPGA
Field **P**rogrammable **G**ate **A**rray

- no physical layout design; design ends with a bitstream used to configure a device
- bought off the shelf and reconfigured by designers themselves

FPGAs vs ASICs



Comparison

- Summary

performance	NREs	Unit cost	TTM
ASIC	ASIC	FPGA	ASIC
FPGA	FPGA	MICRO	FPGA
MICRO	MICRO	ASIC	MICRO

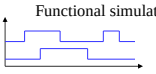
FPGA Design process

Specification (Lab Experiments)

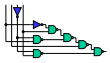
On-paper hardware design
(Block diagram & ASM chart)

VHDL description (Your Source Files)

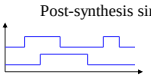
Functional simulation



Synthesis



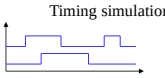
Post-synthesis simulation



FPGA Design process (contd..)

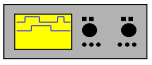
Implementation

Timing simulation

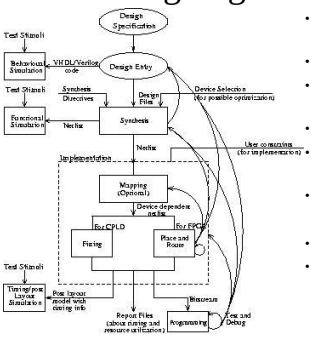


Configuration

On chip testing



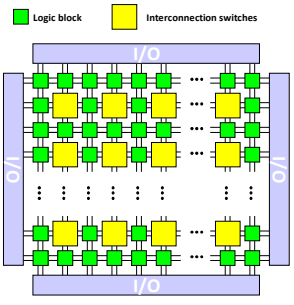
Design Steps Involved in Designing With FPGAs



- Understand and define design requirements
- Design description
- Behavioural simulation (Source code interpretation)
- Synthesis
- Functional or Gate level simulation
- Implementation
  - Place and Route
- Timing or Post layout simulation
- Programming, Test and Debug

FPGA - Generic Structure

FPGA building blocks



- **Programmable logic blocks** Implement combinational and sequential logic
- **Programmable interconnect** Wires to connect inputs and outputs to logic blocks
- **Programmable I/O blocks** Special logic blocks at the periphery of device for external connections

Classification of FPGAs

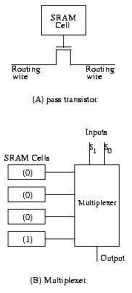
- Families of FPGA's differ in:
  - physical means of implementing user programmability,
  - Granularity
  - how logic is organised

I. Based on physical means of implementing user programmability

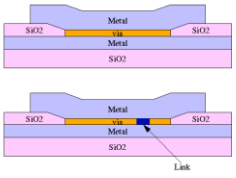
– Can be classified into three categories:
  - SRAM based
  - Fuse based
  - EPROM/EEPROM/Flash based

SRAM Programming Technology

- Employs SRAM (Static RAM) cells to control pass transistors and/or transmission gates
- SRAM cells control the configuration of logic block as well
- Volatile
  - Needs an external storage
  - Needs a power-on configuration mechanism
  - In-circuit re-programmable
- Lesser configuration time
- Occupies relatively larger area



Anti-fuse Programming Technology



- Though implementation differ, all anti-fuse programming elements share common property
  - Uses materials which normally resides in high impedance state
  - But can be fused irreversibly into low impedance state by applying high voltage
- OTP devices
- Fuse blown = connection established

Anti-fuse Programming Technology

- Very low ON Resistance (Faster implementation of circuits)
- Limited size of anti-fuse elements; Interconnects occupy relatively lesser area
  - Offset : Larger transistors needed for programming
- One Time Programmable
  - Cannot be re-programmed (Design changes are not possible)
  - Retain configuration after power off

Antifuse FPGAs

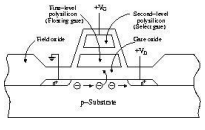
Advantages

- Highest density - a mere cross point - 10X the density of SRAM
- Lowest switch resistance - 25 Ohms
- Very low capacitance 1 fF per node.- approaching the metal line capacitance
- non- volatile
- Nearly impossible to reverse engineer
- Radiation hard - Space appns -
- Live within 1 millisecond of the power supply reaching spec voltage
- Software is easy to place and route

Disadvantages

- Requires programmer
- Requires a socket - a problem for devices with > 200 pins
- Requires one to two transistors per wire for programming
- Some antifuse defects not testable until programming

EPROM, EEPROM or Flash Based Programming Technology



- EPROM Programming Technology
  - Two gates: Floating and Select
  - Normal mode:
    - No charge on floating gate
    - Transistor behaves as normal n-channel transistor
  - Floating gate charged by applying high voltage
    - Threshold of transistor (as seen by gate) increases
    - Transistor turned off permanently
  - Re-programmable by exposing to UV radiation

FPGA Comparison

	SRAM	Antifuse	Flash	EPROM
Speed	Worst	Best	Worst	Medium
Power	Varies	Near Best	Best	Worst
Density	Medium	Second	Best	Worst
Radiation	Worst	Best	Medium	Medium
Routing Cell size	1	1/10	1/7	1/-5-
Reprogrammable	Yes	No	Yes	Yes

Classification of FPGAs

II. Based on Granularity

1. Coarse Grained (SRAM Based) - e.g. Altera, Xilinx

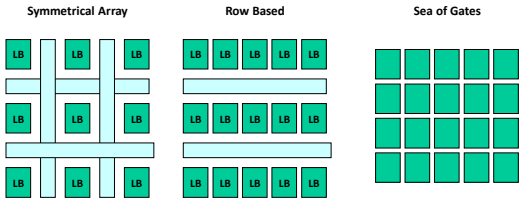
- Large complex logic blocks
- Dedicated functions, fast carry etc.
- Re-programmable
- Unpredictable propagation delays

2. Fine Grained (Antifuse Based) - e.g. Actel

- Sea of small logic blocks
- Predictable propagation delays
- High performance timing
- One time programmable (OTP)

Classification of FPGAs

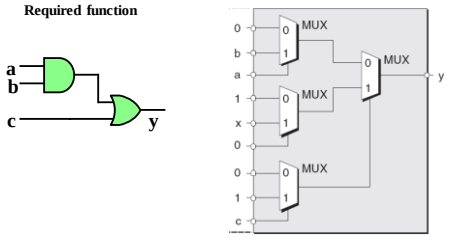
III. Based on how logic is organised



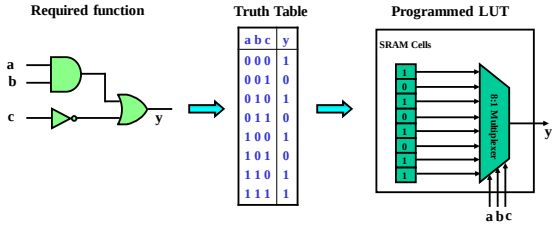
Classification of FPGAs

- FPGAs are also classified based on how the logic blocks are implemented.
  - MUX based
  - LUT based

MUX Implementation



LUT Implementation



Manufacturers are already specialised!

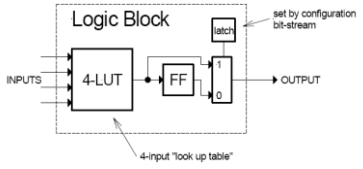
Different companies make different types of FPGAs!

Company	Architecture	Logic Block	Technology
Xilinx	Symetric matrix	Look up table	SRAM
Actel	Row based	Multiplexer	anti-fuse
Plessey	Sea of gates	NAND	SRAM
Quick Logic	Symetric matrix	Multiplexer	anti-fuse
Concurent	Sea of gates	Multiplexers and gates	SRAM
Crosspoint	Row based	Transistors and multiplexers	anti-fuse

Requirement to programmable element
✔ Small R_{on} and large R_{off}

✔ Small area
Small parasitic capacitance
Standard CMOS process

Idealized FPGA Logic Block



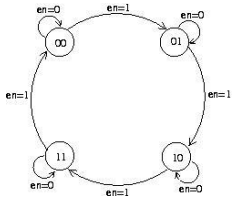
- *look up table (LUT)* implements combinational logic
- Register for sequential circuits
- Additional logic (not shown):
  - Carry logic for arithmetic functions
  - Expansion logic for functions requiring more than 4 inputs

LUT Implementation

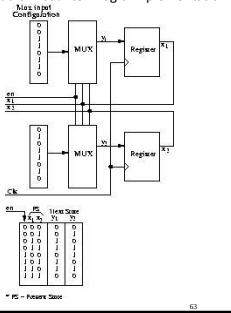
- n-bit LUT is implemented as a 2^n x 1 memory:
  - inputs choose one of 2^n memory locations.
  - memory locations are normally loaded with values from user's configuration bit stream.
  - Inputs to mux control are the Configurable Logic Block (CLB) inputs.
- Result is a general purpose "logic gate".
  - – n-LUT can implement *any* function of n inputs!

An Example

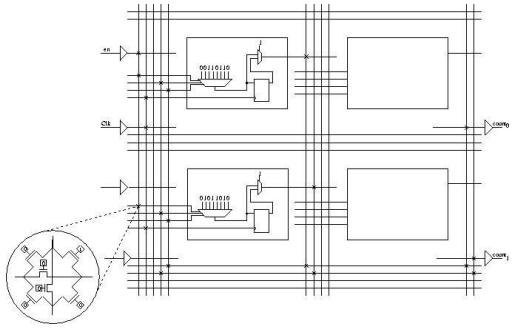
- Modulo-4 counter: Specification



- Modulo-4 counter: Logic Implementation



FPGA Implementation of Modulo-4 Counter



Commercially Available Devices

- Architecture differs from vendor to vendor
- Characterized by
  - Structure and content of logic block
  - Structure and content of routing resources

Classification of PLDs

Actel	ProASIC, Axcelerator, SX-A, SX, MX, eX
Altera	Excalibur, Stratix, APEX 20K, Cyclone, FLEX 10K, MAX 3000A, MAX 7000
Atmel	AT40K, AT6000, 22V10, 1500
Cypress	Ultra37000, Delta39K, FLASH370i, MAX340, Quantum38K
Lattice	ORCA, ispXPGA, ispXPLD, ispMACH, ispGDX
Quicklogic	QuickMIPS, Eclipse, pASIC
Xilinx	Virtex, Spartan, XC9500, CoolRunner, microBLAZE

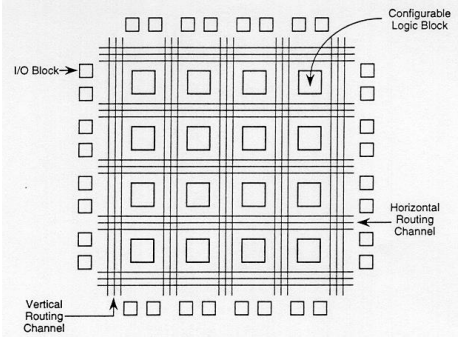
Xilinx family of FPGAs

Artix 7
Kintex 7
Virtex 7
Virtex 6
Virtex 5
Virtex IV
Virtex II Pro
Virtex II
Spartan 3 (1.2V)
Spartan 2E (1.8V)
Spartan 2 (2.5V)
Spartan XL (3.3V)
XC4000
XC3000
XC2000

FPGA Comparison Table

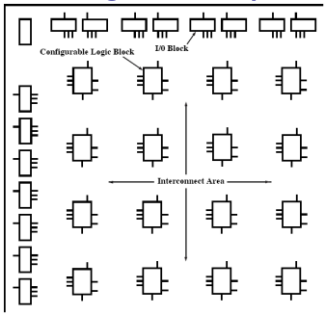
Features	Artix-7	Kintex-7	Virtex-7	Spartan-6	Virtex-6
Logic Cells	352,000	480,000	2,000,000	150,000	760,000
BlockRAM	19Mb	34Mb	85Mb	4.8Mb	38Mb
DSP Slices	1,040	1,920	5,280	180	2,016
DSP Performance (symmetric FIR)	1,129GMACS	2,450GMACS	6,737GMACS	140GMACS	2,419GMACS
Transceiver Count	16	32	96	8	72
Transceiver Speed	6.6Gb/s	12.5Gb/s	28.05Gb/s	3.2Gb/s	11.18Gb/s
Total Transceiver Bandwidth (full duplex)	211Gb/s	800Gb/s	2,784Gb/s	50Gb/s	536Gb/s
Memory Interface (DDR3)	1,066Mb/s	1,866Mb/s	1,866Mb/s	800Mb/s	1,066Mb/s
PCI Express® Interface	Gen2x4	Gen2x8	Gen3x8	Gen1x1	Gen2x8
Agile Mixed Signal (AMS)/XADC	Yes	Yes	Yes		Yes
Configuration AES	Yes	Yes	Yes	Yes	Yes
I/O Pins	600	500	1,200	576	1,200
I/O Voltage	1.2V, 1.35V, 1.5V, 1.8V, 2.5V, 3.3V	1.2V, 1.35V, 1.5V, 1.8V, 2.5V, 3.3V	1.2V, 1.35V, 1.5V, 1.8V, 2.5V, 3.3V	1.2V, 1.5V, 1.8V, 2.5V, 3.3V	1.2V, 1.5V, 1.8V, 2.5V
EasyPath Cost Reduction Solution	-	-	Yes	-	Yes

General Architecture of Xilinx FPGAs



Xilinx calls the logic cells as CLBs

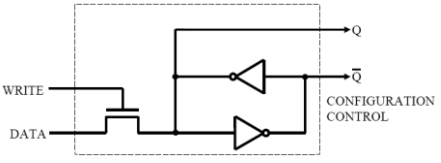
Layout of Part of a Programmable Logic Cell Array



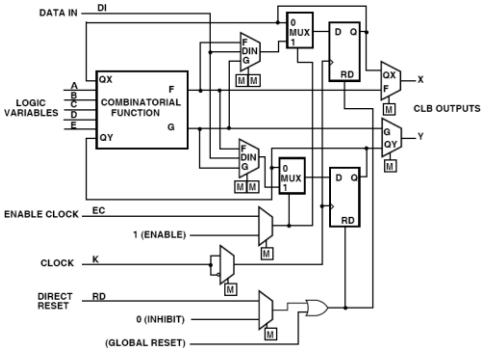
Xilinx 3000 series FPGAs

- Consists of an array of
  - Internal configuration memory cells
  - 64 Configurable Logic Blocks (CLBs) and
  - 64 Input-Output interface blocks(I/O Blocks)

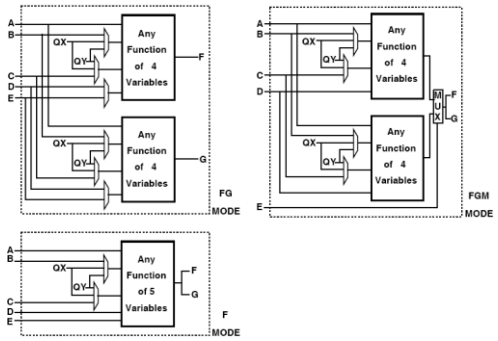
Internal configuration memory cells



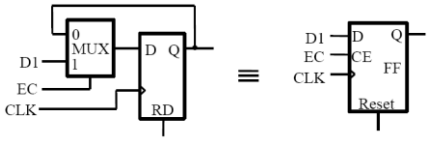
Xilinx 3000 series Logic Cell



Combinatorial Logic Options

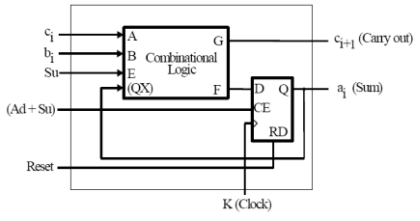


Flip-flops with Clock Enable



$Q^+ = EC\ D1 + EC'\ Q$

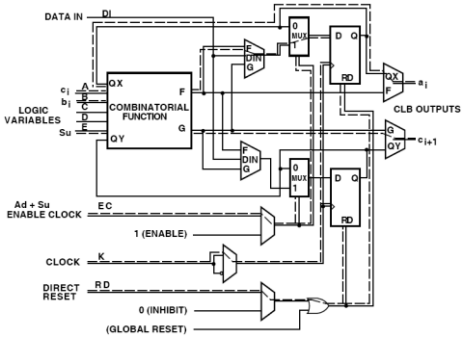
Parallel Adder-Subtracter Logic Cell



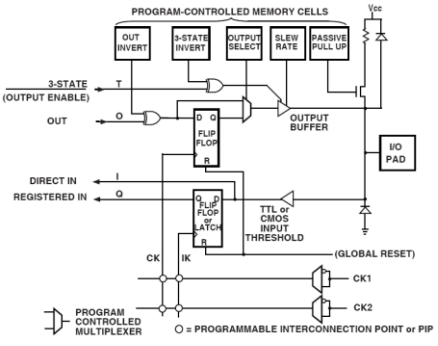
$F = sum = a_i^+ = a_i \oplus (b_i \oplus Su) \oplus c_i$

$G = c_{i+1}$ = carry out $= a_i\, c_i + (a_i + c_i)(b_i \oplus Su)$

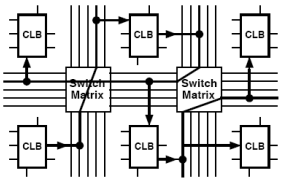
Signal Paths Within Adder-Subtractor Logic Cell



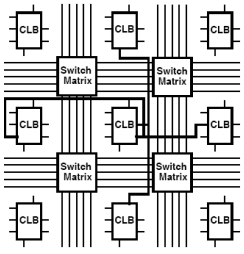
Xilinx 3000 Series I/O Block



General-purpose Interconnects

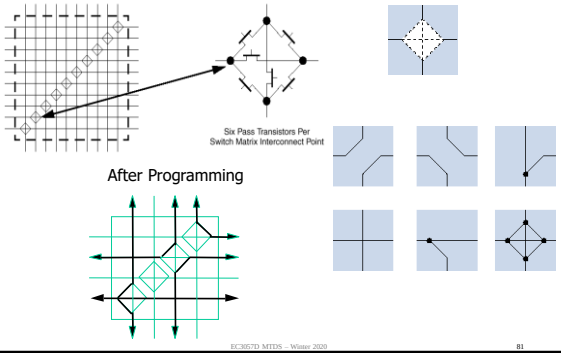


Direct Interconnects Between Adjacent CLBs

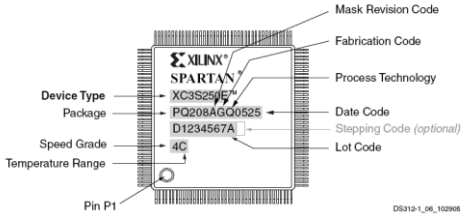


Programmable Switch Matrix

programmable switch element

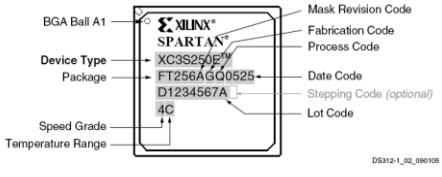


Package Marking



Spartan-3E QFP Package Marking Example

Package Marking



Spartan-3E BGA Package Marking Example

Spartan-3E FPGAs

Example: XC3S250E -4 FT 256 C S1 (optional code to specify Stepping 1)

Device Type
Speed Grade
Package Type
Temperature Range
Number of Pins

DS312_03_082499

Device	Speed Grade		Package Type / Number of Pins		Temperature Range (T_J)	
XC3S100E	–4	Standard Performance	VQ100 VQG100	100-pin Very Thin Quad Flat Pack (VQFP)	C	Commercial (0°C to 85°C)
XC3S250E	–5	High Performance	CP132 CPG132	132-ball Chip-Scale Package (CSP)	I	Industrial (–40°C to 100°C)
XC3S500E			TQ144 TQG144	144-pin Thin Quad Flat Pack (TQFP)		
XC3S1200E			PQ208 PQG208	208-pin Plastic Quad Flat Pack (PQFP)		
XC3S1600E			FT256 FTG256	256-ball Fine-Pitch Thin Ball Grid Array (FTBGA)		
			FG320 FGG320	320-ball Fine-Pitch Ball Grid Array (FBGA)		
			FG400 FGG400	400-ball Fine-Pitch Ball Grid Array (FBGA)		
			FG484 FGG484	484-ball Fine-Pitch Ball Grid Array (FBGA)		

Spartan-3E FPGAs

Example: XC3S250E -4 FT 256 C S1 (optional code to specify Stepping 1)

Device Type
Speed Grade
Package Type
Temperature Range
Number of Pins

Device	Speed Grade		Package Type / Number of Pins		Temperature Range (T_J)
XC3S100E	–4	Standard Performance	VQ100 VQG100	100-pin Very Thin Quad Flat Pack (VQFP)	C Commercial (0°C to 85°C)
XC3S250E	–5	High Performance	CP132 CPG132	132-ball Chip-Scale Package (CSP)	I Industrial (–40°C to 100°C)
XC3S500E			TQ144 TQG144	144-pin Thin Quad Flat Pack (TQFP)	
XC3S1200E			PQ208 PQG208	208-pin Plastic Quad Flat Pack (PQFP)	
XC3S1600E			FT256 FTG256	256-ball Fine-Pitch Thin Ball Grid Array (FTBGA)	
			FG320 FGG320	320-ball Fine-Pitch Ball Grid Array (FBGA)	
			FG400 FGG400	400-ball Fine-Pitch Ball Grid Array (FBGA)	
			FG484 FGG484	484-ball Fine-Pitch Ball Grid Array (FBGA)	

Xilinx Spartan-III FPGAs

Table 1: Summary of Spartan-3 FPGA Attributes

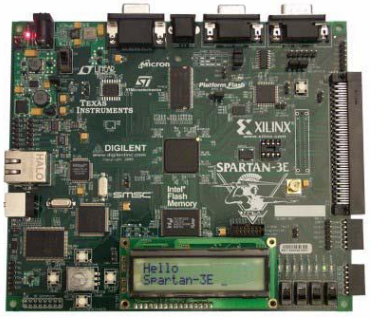
Device	System Gates	Logic Cells	CLB Array (One CLB = Four Slices) Rows	Columns	Total CLBs	Distributed RAM (bits[1])	Block RAM (bits[1])	Dedicated Multipliers	DCMs	Maximum User I/O	Maximum Differential I/O Pairs
XC3S50	50K	1,728	16	12	192	12K	72K	4	2	124	56
XC3S200	200K	4,320	24	20	480	30K	216K	12	4	173	76
XC3S400	400K	8,064	32	28	896	56K	288K	16	4	264	116
XC3S1000	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	712	312
XC3S5000	5M	74,880	104	80	8,320	520K	1,872K	104	4	784	344

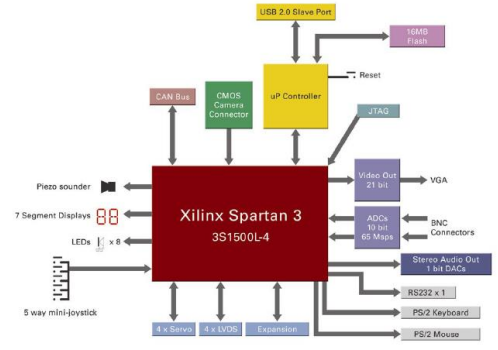
Xilinx Virtex FPGAs

Table 1: Virtex Field-Programmable Gate Array Family Members.

Device	System Gates	CLB Array	Logic Cells	Maximum Available I/O	Block RAM Bits	Maximum SelectRAM+™ Bits
XCV50	57,906	16x24	1,728	180	32,768	24,576
XCV100	108,904	20x30	2,700	180	40,960	38,400
XCV150	164,674	24x36	3,888	260	49,152	55,296
XCV200	236,666	28x42	5,292	284	57,344	75,264
XCV300	322,970	32x48	6,912	316	65,536	98,304
XCV400	468,252	40x60	10,800	404	81,920	153,600
XCV600	661,111	48x72	15,552	512	98,304	221,184
XCV800	888,439	56x84	21,168	512	114,688	301,056
XCV1000	1,124,022	64x96	27,648	512	131,072	393,216

Educational Board





Reference

1. Fundamentals of Logic Design, Charles H Roth.
2. Digital System Design using VHDL, Charles H Roth.
2. Xilinx FPGA data sheets and manuals.