Introduction to Modern FPGAs

Arturo Díaz Pérez

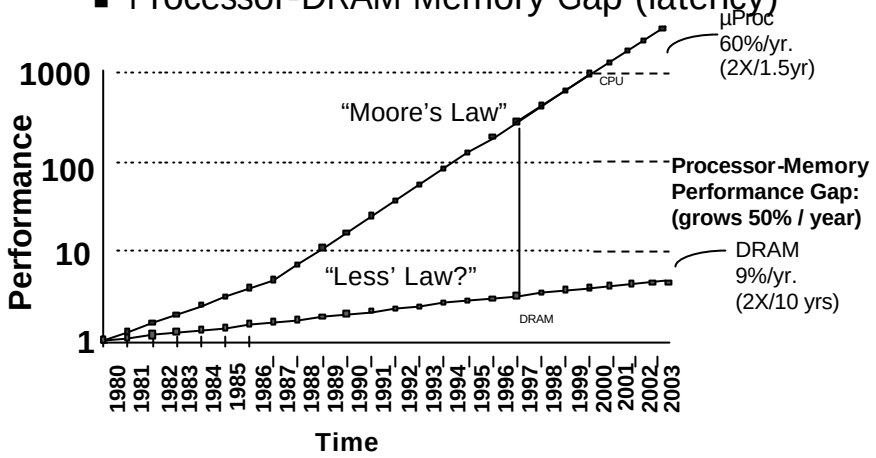
Centro de Investigación y de Estudios Avanzados del IPN
Departamento de Ingeniería Eléctrica
Sección de Computación
adiaz@cs.cinvestav.mx

Outline

- Technology Evolution
- Field Programmable Gate-Arrays
- Applications Trends
- Future Processors
- Conclusions

Moore´s Law

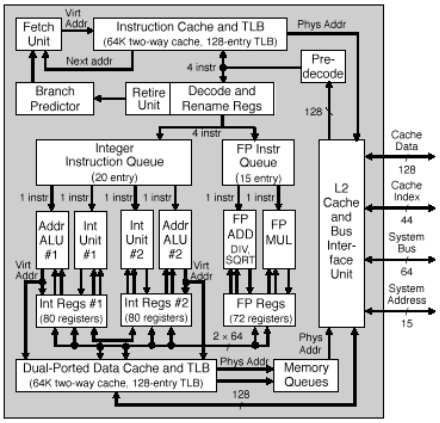
- Processor-DRAM Memory Gap (latency)



Today Conventional Microprocessors

- Instructions sets
- Advanced memory systems
  - (L3 caches, memory, virtual memory)
- Advanced Instruction Level Parallelism
  - (pipelining, superscalar, vectors, VLIW, speculative, branch prediction)
- Interconnection Technology
- Basic parallel processing

Modern Processor Architecture



Alpha 21264 500Mhz

Hardware Trends

- In 1994 Semiconductor Industry Association predicted that by 2010
  - 800-million-transistor processors
  - Thousands of pins
  - 1,000-bit bus
  - Clock speed over 2 GHz
  - 180 W

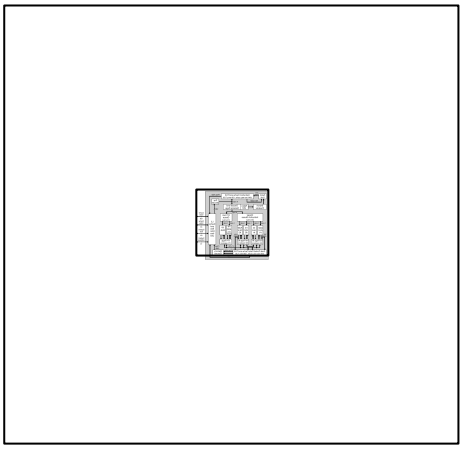
What will integrated circuits be at 2010?

- Microprocessors will have more than one billion logic transistors on a single chip
- Transistors and wires will have feature sizes less than a tenth of a micron
- The time to send a signal along an on-chip wire will become proportionally much greater than that needed for a transistor to switch
- Off-chip communication will become relatively slower
- Minimizing power dissipation and the resultant heat will be enormous, despite reduce voltage levels

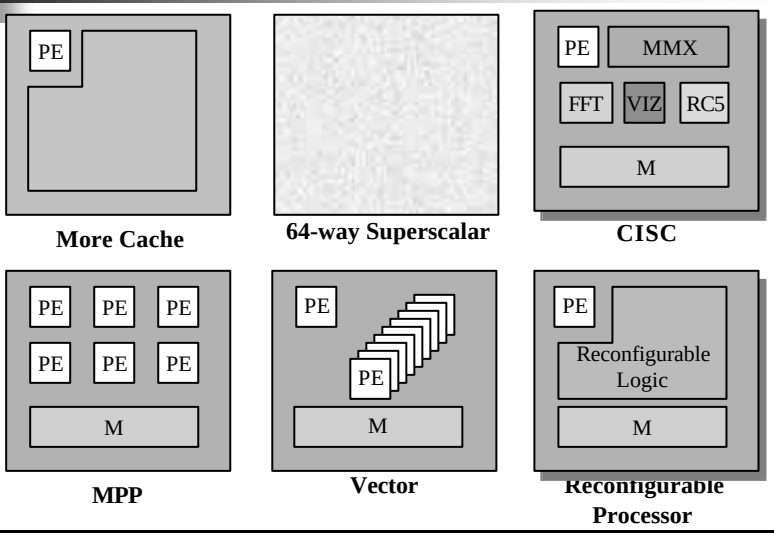
Hardware Trends and Physical Limits

- On-chip wires are becoming much slower relative to logic gates as the on-chip devices shrink.
  - It will be impossible to keep one global clock over the entire chip
- *It will be necessary to distribute silicon area among independent but coordinated different modules*

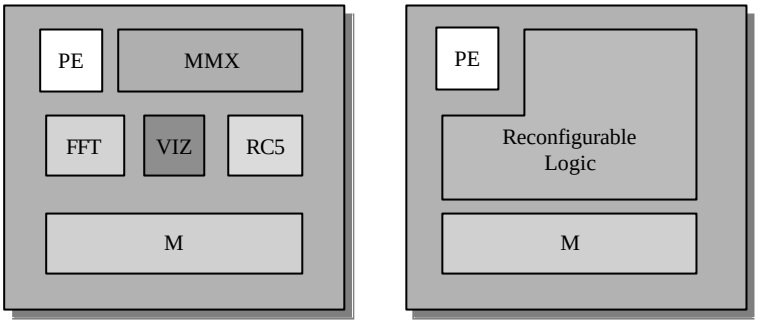
In 10 Years!



Using the Silicon



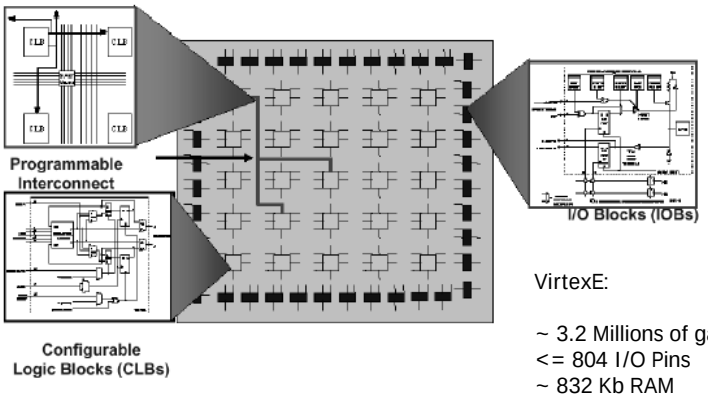
Using the Silicon



CISC **Reconfigurable Processor**

FPGA: Field Programmable Gate-Array

- **Gate-Array-like architecture**
- **Programmable logic, I/O & interconnect**

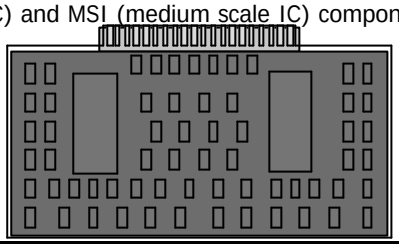


VirtexE:

~ 3.2 Millions of gates
<= 804 I/O Pins
~ 832 Kb RAM

Why FPGAs? (1 / 5)

- By the early 1980's most of logic circuits in typical systems were absorbed by a handful of standard large scale integrated circuits (LSI ICs).
  - Microprocessors, bus/IO controllers, system timers, ...
- Every system still needed random small "glue logic" ICs to help connect the large ICs:
  - generating global control signals (for resets etc.)
  - data formatting (serial to parallel, multiplexing, etc.)
- Systems had a few LSI components and lots of small low density SSI (small scale IC) and MSI (medium scale IC) components.



Printed Circuit (PC) board with many small SSI and MSI ICs and a few LSI ICs

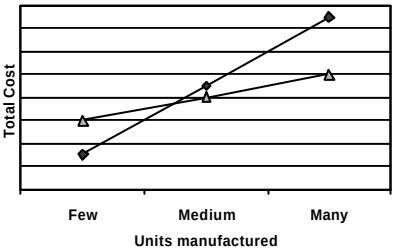
Why FPGAs? (2 / 5)

- Custom ICs sometimes designed to replace glue logic:
  - reduced complexity/manufacturing cost, improved performance
  - But custom ICs expensive to develop, and delay introduction of product ("time to market") because of increased design time
- Note: need to worry about two kinds of costs:
  1. cost of development, "Non-Recurring Engineering (NRE)", fixed
  2. cost of manufacture per unit, variable

  Usually tradeoff between NRE cost and manufacturing costs
- Therefore custom IC approach was only viable for products with very high volume (where NRE could be amortized), and not sensitive in time to market (TTM)

NRE

NRE



Why FPGAs? (3 / 5)

- FPGAs introduced as alternative to custom ICs for implementing glue logic:
  - improved PC board density vs. discrete SSI/MSI components (within around 10x of custom ICs)
  - computer aided design (CAD) tools meant circuits could be implemented quickly (no physical layout process, no mask making, no IC manufacturing), relative to Application Specific ICs (ASICs) (3-6 months for these steps for custom IC)
    - lowers NREs (Non Recurring Engineering)
    - shortens TTM (Time To Market)
- Because of Moore's law the density (gates/area) of FPGAs continued to grow through the 80's and 90's to the point where major data processing functions can be implemented on a single FPGA.

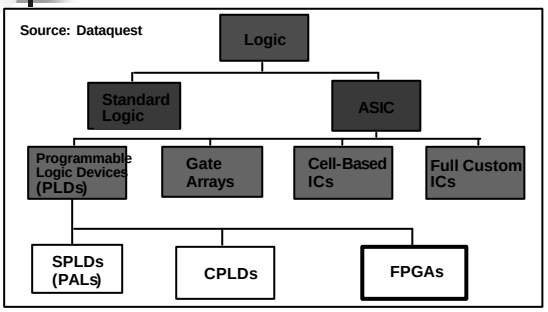
Why FPGAs? (4 / 5)

- FPGAs continue to compete with custom ICs for special processing functions (and glue logic) but now try to <u>compete with microprocessors in dedicated and embedded applications</u>
  - Performance advantage over microprocessors because circuits can be customized for the task at hand. Microprocessors must provide special functions in software (many cycles)
- MICRO: Highest NRE, SW: fastest TTM
- ASIC: Highest performance, worst TTM
- FPGA: Highest cost per chip (unit cost)

Why FPGAs? (5 / 5)

- As Moore's Law continues, FPGAs work for more applications as both can do more logic in 1 chip and faster
- Can easily be "patched" vs. ASICs
- Perfect for courses:
  - Can change design repeatedly
  - Low TTM yet reasonable speed

Programmable Logic Device Families



Acronyms
SPLD = Simple Prog. Logic Device
PAL = Prog. Array of Logic
CPLD = Complex PLD
FPGA = Field Prog. Gate Array

Common Resources

Configurable Logic Blocks (CLB)
- Memory Look-Up Table
- AND-OR planes
- Simple gates

Input / Output Blocks (IOB)
- Bidirectional, latches, inverters, pullup/pulldowns

Interconnect or Routing
- Local, internal feedback, and global

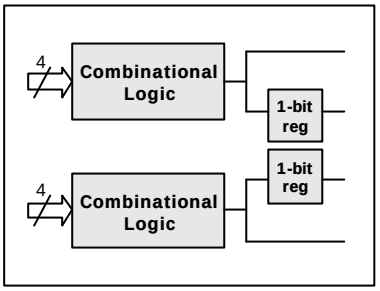
What is FPGA?

- Field Programmable Gate Arrays
- Array of logic cells connected via routing channels
- Special I/O cells
- Logic cells are mainly LUT with associated registers
- Interconnection on SRAM basis or antifuse elements
- Architecting a FPGA
  - Performance
  - Density and capacity
  - Ease of use
  - In-system programmability and in-circuit reprogrammability

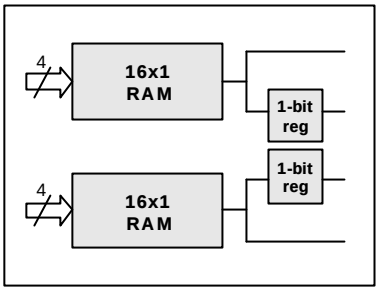
Other FPGA features

- Besides primitive logic elements and programmable routing, some FPGA families add other features
- Embedded memory
  - Many hardware applications need memory for data storage. Many FPGAs include blocks of RAM for this purpose
- Dedicated logic for carry generation, or other arithmetic functions
- Phase locked loops for clock synchronization, division, multiplication.

Configurable Logic Block



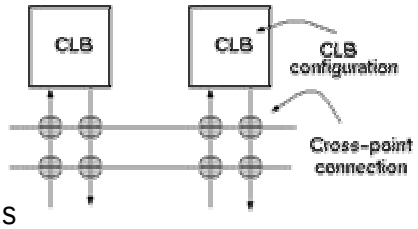
Logic Mode



Memory Mode

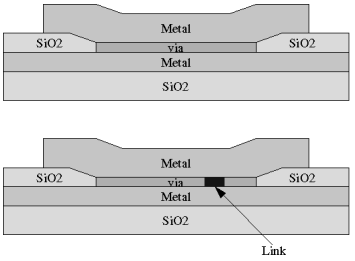
FPGA Variations

- Families of FPGA's differ in:
  - physical means of implementing user programmability,
  - arrangement of interconnection wires, and
  - basic functionality of logic blocks
- Most significant difference is in the method for providing flexible blocks and connections



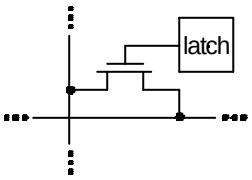
Technology and Architecture Tradeoffs

- Antifuse elements
- high density
- non volatile
- not reprogrammable



User Programmability

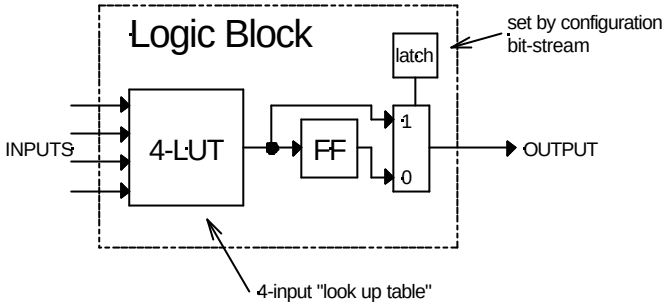
- Latch-based (Xilinx, Altera, ...)



  + reconfigurable
  - volatile
  - relatively large die size
- Note: Today 90% die is interconnect, 10% is gates

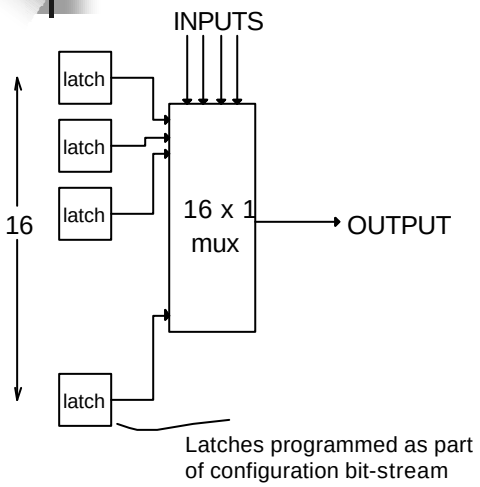
- Latches are used to:
  1. make or break cross-point connections in interconnect
  2. define function of logic blocks
  3. set user options:
     - within the logic blocks
     - in the input/output blocks
     - global reset/clock
- "Configuration bit stream" loaded under user control:
  - All latches are strung together in a shift chain
  - "Programming" => creating bit stream

Idealized FPGA Logic Block



- 4-input *Look Up Table (4-LUT)*
  - implements combinational logic functions
- Register
  - optionally stores output of LUT
  - Latch determines whether read reg or LUT

4-LUT Implementation



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

LUT as general logic gate

- An n-lut as a direct implementation of a function truth-table
- Each latch location holds value of function corresponding to one input combination

Example: 2-lut

INPUTS	AND	OR
00	0	0
01	0	1
10	0	1
11	1	1

Implements *any* function of 2 inputs.

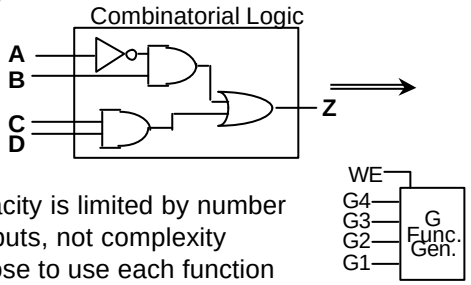
How many functions of n inputs?

Example: 4-lut

INPUTS		
0000	F(0,0,0,0)	← store in 1st latch
0001	F(0,0,0,1)	← store in 2nd latch
0010	F(0,0,1,0)	←
0011	F(0,0,1,1)	←
0011		
0100	⋮	
0101		
0110		
0111		
1000		
1001		
1010		
1011		
1100		
1101		
1110		
1111		

Look Up Tables

- Combinatorial Logic is stored in 16x1 SRAM Look Up Tables (LUTs) in a CLB
- Example:



- Capacity is limited by number of inputs, not complexity
- Choose to use each function generator as 4 input logic (LUT) or as high speed sync.dual port RAM

Look Up Table

4-bit address

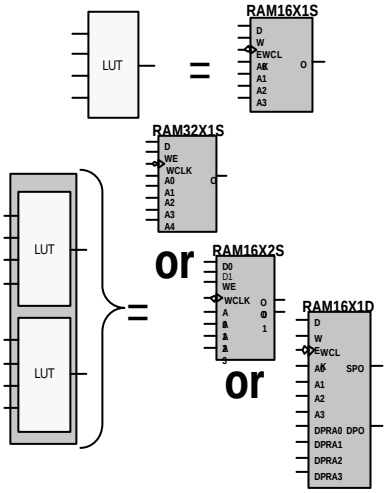
A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
	.	.	.	
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

More functionality for “free”?

- Given basic idea
  - LUT built from RAM
  - Latches connected as shift register
- What other functions could be provided at very little extra cost?
- Using CLB latches as little RAM vs. logic
- Using CLB latches as shift register vs. logic

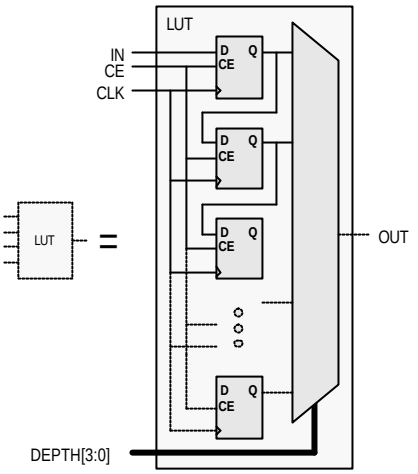
1. “Distributed RAM”

- CLB LUT configurable as Distributed RAM
  - A LUT equals 16x1 RAM
  - Implements Single and Dual-Ports
  - Cascade LUTs to increase RAM size
- Synchronous write
- Synchronous/Asynchronous read
  - Accompanying flip-flops used for synchronous read



2. Shift Register

- Each LUT can be configured as shift register
  - Serial in, serial out
- Saves resources: can use less than 16 FFs
- Faster: no routing
- Note: CAD tools determine with CLB used as LUT, RAM, or shift register, rather than up to designer



Design Flow

1 **Design Entry** in schematic, ABEL, VHDL, and/or Verilog. Vendors include Synopsys, Aldec (Xilinx Foundation), Mentor, Cadence, Viewlogic, and 35 oth

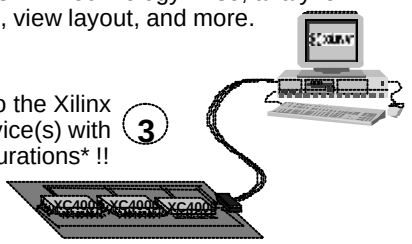
SYNOPSYS Mentor Graphics VIEWlogic CADENCE ALDEC

XILINX

M1 Technology

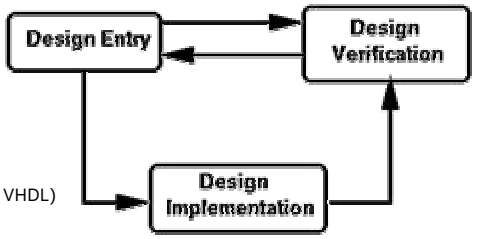
2 **Implementation** includes Placement & Routing and bitstream generation using Xilinx's M1 Technology. Also, analyze timing, view layout, and more.

Download directly to the Xilinx hardware device(s) with unlimited reconfigurations* !! 3



*XC9500 has 10,000 write/erase cycles

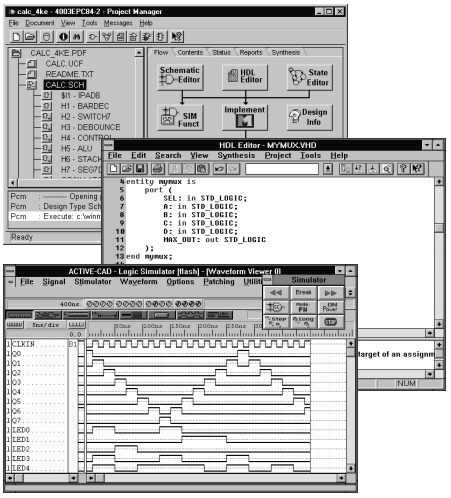
How Program: FPGA Generic Design Flow



- Design Entry:
  - Create your design files using:
    - schematic editor or
    - hardware description language (Verilog, VHDL)
- Design "implementation" on FPGA:
  - *Partition, place, and route* ("PPR") to create bit-stream file
  - Divide into CLB-sized pieces, place into blocks, route to blocks
- Design verification:
  - Use Simulator to check function,
  - Other software determines max clock frequency.
  - Load onto FPGA device (cable connects PC to board)
    - check operation at full speed in real environment.

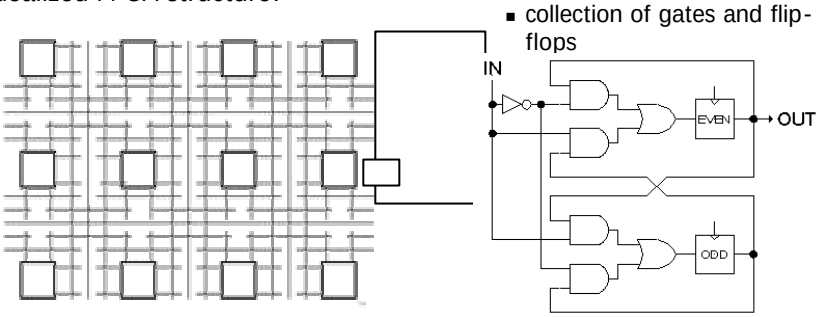
Foundation Series Delivers Value & Ease of Use

- **Complete, ready-to-use software solution**
- **Simple, easy-to-use design environment**
- **Easy-to-learn schematic, state-diagram, ABEL, VHDL, & Verilog design**
- **Synopsys**
- **FPGA**
- **Express**
- **Integration***



Example Partition, Placement, and Route

- Idealized FPGA structure:
- Example Schematic Circuit:
  - collection of gates and flip-flops



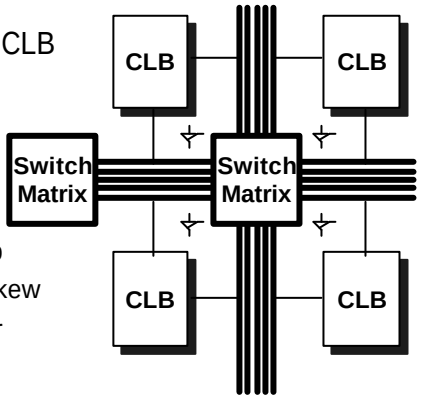
Circuit combinational logic must be “covered” by 4-input 1-output “gates”.

Flip-flops from circuit must map to FPGA flip-flops.
(Best to preserve “closeness” to CL to minimize wiring.)

Placement in general attempts to minimize wiring.

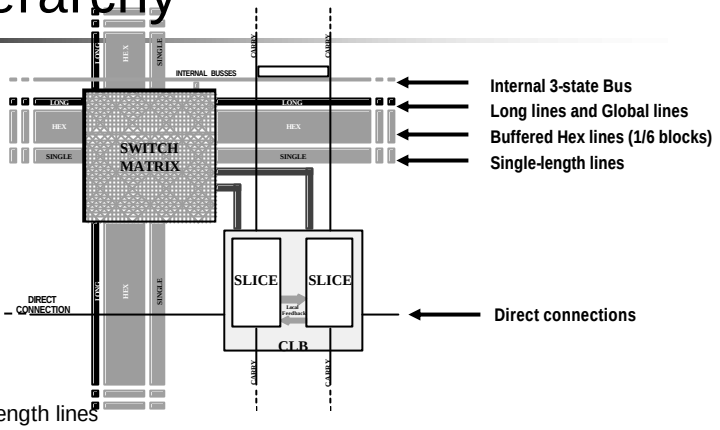
Xilinx FPGA Routing

- 1) Fast Direct Interconnect - CLB to CLB
- 2) General Purpose Interconnect - Uses switch matrix
- 3) Long Lines
  - Segmented across chip
  - Global clocks, lowest skew
  - 2 Tri-states per CLB for busses



Xilinx Virtex-E Routing Hierarchy

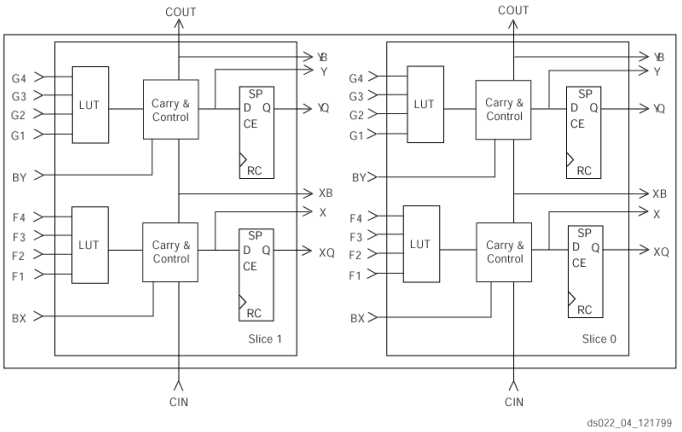
Note:
CAD tools do PPR, not designers



- 24 single-length lines
  - Route GRM signals to adjacent GRMs in 4 directions
- 96 buffered hex lines
  - Route GRM (general routing matrix) signals to another GRMs six blocks away in each of the 4 directions
- 12 buffered Long lines
  - Routing across top and bottom, left and right

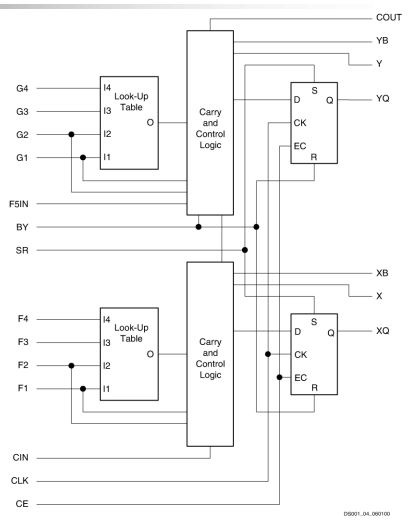
Virtex-E Configurable Logic Block (CLB)

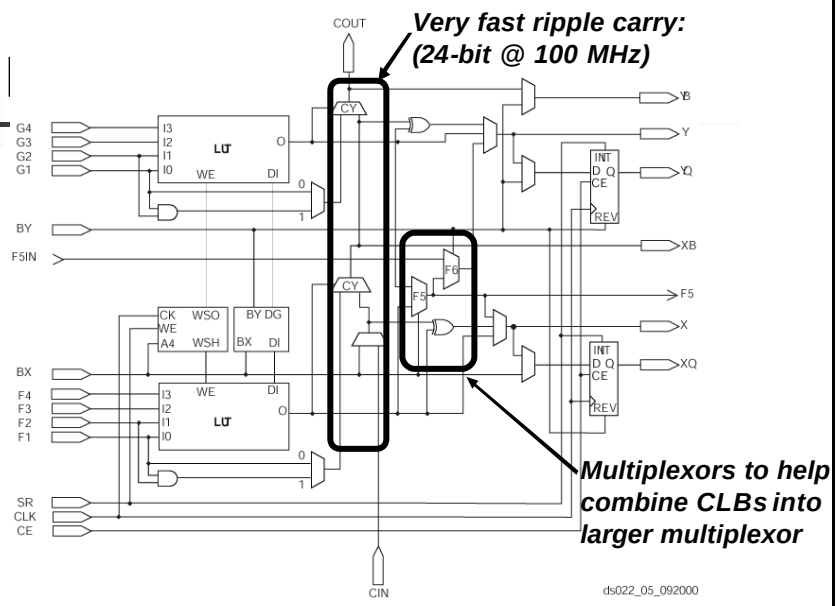
2 "logic slices" / CLB, two 4-LUTs / slice
=> Four 4-LUTs / CLB



Virtex-E CLB Slice Structure

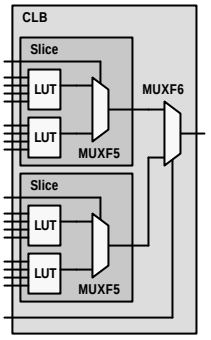
- Each slice contains two sets of the following:
  - Four-input LUT
    - Any 4-input logic function
    - Or 16-bit x 1 sync RAM
    - Or 16-bit shift register
  - Carry & Control
    - Fast arithmetic logic
    - Multiplexer logic
    - Multiplier logic
  - Storage element
    - Latch or flip-flop
    - Set and reset
    - True or inverted inputs
    - Sync. or async. control



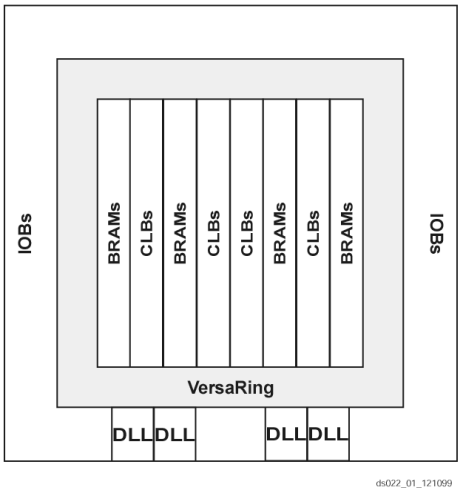


Virtex-E Dedicated Expansion Multiplexers

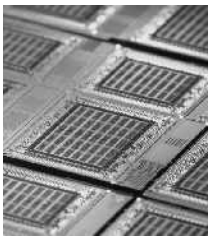
- Since 4-LUT has 4 inputs, max is 2:1 Mux (2 inputs, 1 control line)
- MUXF5 combines 2 LUTs to create
  - 4x1 multiplexer
  - Or any 5-input function (5-LUT)
  - Or selected functions up to 9 inputs
- MUXF6 combines 2 slices to form
  - 8x1 multiplexer
  - Or any 6-input function (6-LUT)
  - Or selected functions up to 19 inputs
- Dedicated muxes are faster and more space efficient



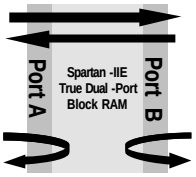
Xilinx Virtex-E Chip Floorplan



- Input / Output Blocks (IOBs)
- Configurable Logic Blocks (CLBs)
- Block RAMs (BRAMs)
- Delay Locked Loop (DLL)



Block RAM (Extra RAM not using LUTs)

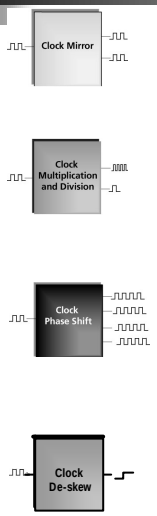


- Most efficient memory implementation
  - Dedicated blocks of memory
- Ideal for most memory requirements
  - Virtex-E XCV2000 has 160? blocks
    - 4096 bits per blocks
  - Use multiple blocks for larger memories
- Builds both single and true dual-port RAMs
- CORE Generator provides custom-sized block RAMs
  - Quickly generates optimized RAM implementation

Virtex-E Block RAM

- Flexible 4096-bit block... Variable aspect ratio
  - 4096 x 1
  - 2048 x 2
  - 1024 x 4
  - 512 x 8
  - 256 x 16
- Increase memory depth or width by cascading blocks

Virtex-E Delay Lock Loop (DLL) Capabilities

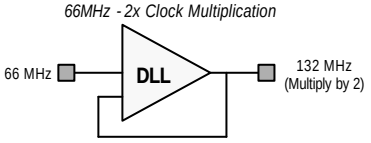


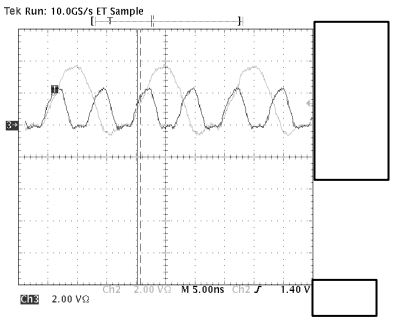
- Easy clock duplication
  - System clock distribution
  - Cleans and reconditions incoming clock
- Quick and easy frequency adjustment
- Single crystal easily generates multiple clocks
- Excellent for advanced memory types
- De-skew incoming clock
- Generate fast setup and hold time or fast clock-to-outs

DLL: Multiplication of Clock Speed

- Have faster internal clock relative to external clock source
- Use 1 DLL for 2x multiplication
- Combine 2 DLLs for 4x multiplication
- Reduce board EMI
  - Route low-frequency clock externally and multiply clock on-chip

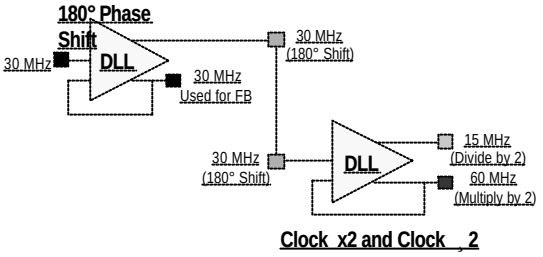
66MHz - 2x Clock Multiplication





DLL: Division of Clock Speed

- Selectable division values
  - 1.5, 2, 2.5, 3, 4, 5, 8, or 16
- Cascade DLLs to combine functions
  - Combine DLLs to multiply and divide to get desired speed
- 50/50 duty cycle correction available



Clock Management Summary

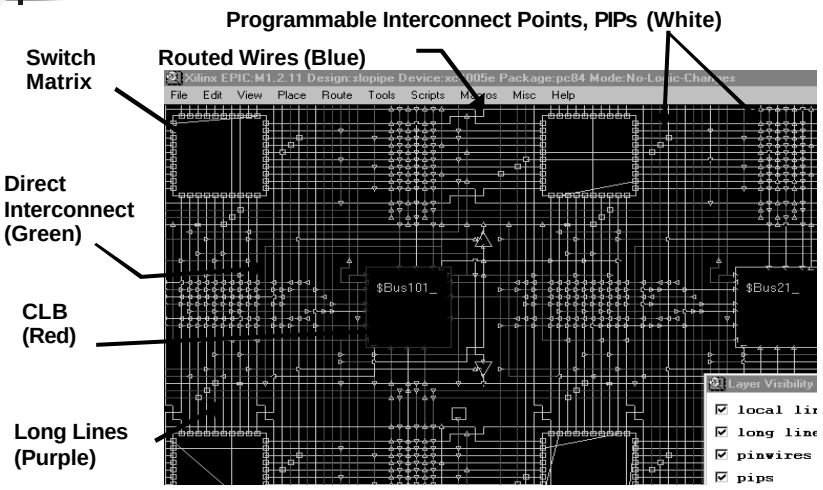
- All digital DLL Implementation
  - Input noise rejection
  - 50/50 duty cycle correction
- Clock mirror provides system clock distribution
- Multiply input clock by 2x or 4x
- Divide clock by 1.5, 2, 2.5, 3, 4, 5, 8, or 16
- De-skew clock for fast setup, hold, or clock-to-out times

Virtex-E Family of Parts

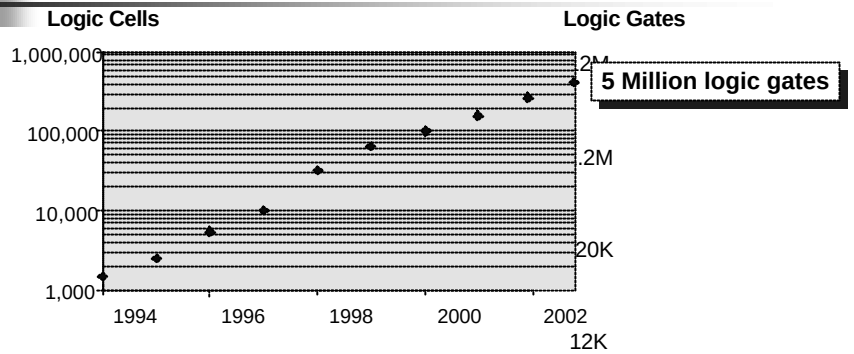
Table 1: **Virtex-E Field-Programmable Gate Array Family Members**

Device	System Gates	Logic Gates	CLB Array	Logic Cells	Differential I/O Pairs	User I/O	BlockRAM Bits	Distributed RAM Bits
XCV50E	71,693	20,736	16 x 24	1,728	83	176	65,536	24,576
XCV100E	128,236	32,400	20 x 30	2,700	83	196	81,920	38,400
XCV200E	306,393	63,504	28 x 42	5,292	119	284	114,688	75,264
XCV300E	411,955	82,944	32 x 48	6,912	137	316	131,072	98,304
XCV400E	569,952	129,600	40 x 60	10,800	183	404	163,840	153,600
XCV600E	985,882	186,624	48 x 72	15,552	247	512	294,912	221,184
XCV1000E	1,569,178	331,776	64 x 96	27,648	281	660	393,216	393,216
XCV1600E	2,188,742	419,904	72 x 108	34,992	344	724	589,824	497,664
XCV2000E	2,541,952	518,400	80 x 120	43,200	344	804	655,360	614,400
XCV2600E	3,263,755	685,584	92 x 138	57,132	344	804	753,664	812,544
XCV3200E	4,074,387	876,096	104 x 156	73,008	344	804	851,968	1,038,336

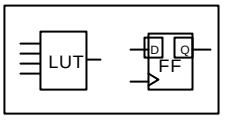
What's Really In that Chip?



Exponential Growth in Density



- **Nov. 1997- shipping world's largest FPGA, XC40125XV (10,982 logic cells, 250K System Gates)**
- **1 Logic cell = 4-input LUT + FF**
- **175,000 Logic cells = 2.0 M logic gates in 2001**



Virtex-II Pro

Feature/Product	XC 2VP2	XC 2VP4	XC 2VP7	XC 2VP20	XC 2VP30	XC 2VP40	XC 2VP50	XC 2VP70	XC 2VP100	XC 2VP125
EasyPath cost reduction	-	-	-	-	XCE 2VP30	XCE 2VP40	XCE 2VP50	XCE 2VP70	XCE 2VP100	XCE 2VP125
Logic Cells	3,168	6,768	11,088	20,880	30,816	43,632	53,136	74,448	99,216	125,136
Slices	1,408	3,008	4,928	9,280	13,696	19,392	23,616	33,088	44,096	55,616
BRAM (Kbits)	216	504	792	1,584	2,448	3,456	4,176	5,904	7,992	10,008
18x18 Multipliers	12	28	44	88	136	192	232	328	444	556
Digital Clock Management Blocks	4	4	4	8	8	8	8	8	12	12
Config (Mbits)	1.31	3.01	4.49	8.21	11.36	15.56	19.02	25.6	33.65	42.78
PowerPC Processors	**0**	**1**	**1**	**2**	**2**	**2**	**2**	**2**	**2**	**4**
Max Available Multi-Gigabit Transceivers*	4	4	8	8	8	12*	16*	20	20*	24*
Max Available User I/O*	204	348	396	564	644	804	852	996	1164	1200

1 Logic Cell = (1) 4-input LUT + (1) FF + (1) Carry Logic
1 CLB = (4) Slices

http://www.xilinx.com/products/tables/fpga.htm#v2p

Issues in FPGA Technologies

- Complexity of Logic Element
  - How many inputs/outputs for the logic element?
  - Does the basic logic element contain a FF? What type?
- Interconnect
  - How fast is it? Does it offer 'high speed' paths that cross the chip? How many of these?
  - Can I have on-chip tri-state busses?
  - How routable is the design? If 95% of the logic elements are used, can I route the design?
    - More routing means more routability, but less room for logic elements

Issues in FPGA Technologies (cont)

- Macro elements
  - Are there SRAM blocks? Is the SRAM dual ported?
  - Is there fast adder support (i.e. fast carry chains?)
  - Is there fast logic support (i.e. cascade chains)
  - What other types of macro blocks are available (fast decoders? register files?)
- Clock support
  - How many global clocks can I have?
  - Are there any on-chip Phase Logic Loops (PLLs) or Delay Locked Loops (DLLs) for clock synchronization, clock multiplication?

What about applications?

Reconfigurable Computing Issues

- Attributes
  - Highly regular
  - Pipeline-able
  - Variable bit-width
  - Logical, integer or fixed-point operations
  - Local interconnections
- Better tools
  - There is currently no way to program RC systems in a general purpose manner. Current tools are slow and hardware oriented.
- Better architectures
  - FPGA is current basis

Application Sampling

- Binary operations over large operands
- Arithmetic operations over non standard length operands
- Encryption, decryption and compression
- Sequence and string matching
- Sorting
- Physical system simulation
- Video and image processing
- Relaxation methods
- Neural networks implementation
- DSP
- Genetic programming
- Dynamic programming

Applications

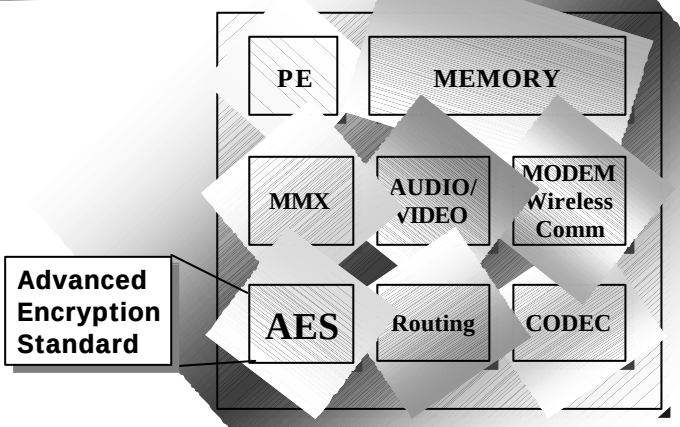
- There is a clear trend in personal computing toward multimedia-rich and mobile applications
  - Audio and Video Compression
  - 2D Image Processing
  - 3D Graphics
  - Speech and handwriting recognition
  - media mining
  - narrow-/broadband signal processing for communication

Mobile computing

Mobile Computing

- Requirements for mobile services are
  - wireless communications
  - stability,
  - bandwidth/cost considerations,
  - integration into the familiar environment,
  - application transparency,
  - extendibility, and
  - Security
    - Authentication and encryption

Future Mo sor



Conclusion: FPGAs (1)

- FPGAs are basically interconnect plus distributed RAM that can be programmed to act as any logical function of 4 inputs
- How they differ from idealized array:
  - In addition to their use as general logic “gates”, LUTs can alternatively be used as general purpose RAM or shift register
    - Each 4-LUT can become a 16x1-bit RAM array
  - Special circuitry to speed up “ripple carry” in adders and counters
    - Therefore adders assembled by the CAD tools operate much faster than adders built from gates and LUTs alone.
  - Many more wires, including tri-state capabilities.

Conclusion: FPGAs (2)

- CAD tools due the partitioning, routing and placement functions onto CLBs
- FPGAs offer compromise of performance, unit cost, time to market vs. ASICs and microprocessors plus software

Conclusions: Reconfigurable Computing

- The use of reconfigurable logic attached to a processor is feasible
- A number of ways in which the architecture of solutions to application problems might change in the future
- Reconfigurable hardware seems to have all the right characteristics to ensure its long-term position in the market
- Until arbitrary programs can be transformed into efficient hardware implementations, it will be necessary for programmers to think carefully about the “hardware programs” they write
- Hardware compilation and FPGA are key new ideas for research in computer architecture
- Reconfigurable hardware should be an increasingly important component in the future systems