Modern FPGA Basics

The *Xilinx XC6200* chip, the *software tools and the board development tools*

What is an FPGA?

- Field Programmable Gate Array
- Fully programmable alternative to a customized chip
- Used to implement functions in hardware
- Also called a Reconfigurable Processing Unit (**RPU**)

Reasons to use an FPGA

- Hardwired logic is **very fast**
- Can interface to outside world
  - Custom hardware/peripherals
  - “Glue logic” to custom co/processors
- Can perform bit-level and systolic operations not suited for traditional CPU/MPU

The Xilinx XC6200 RPU

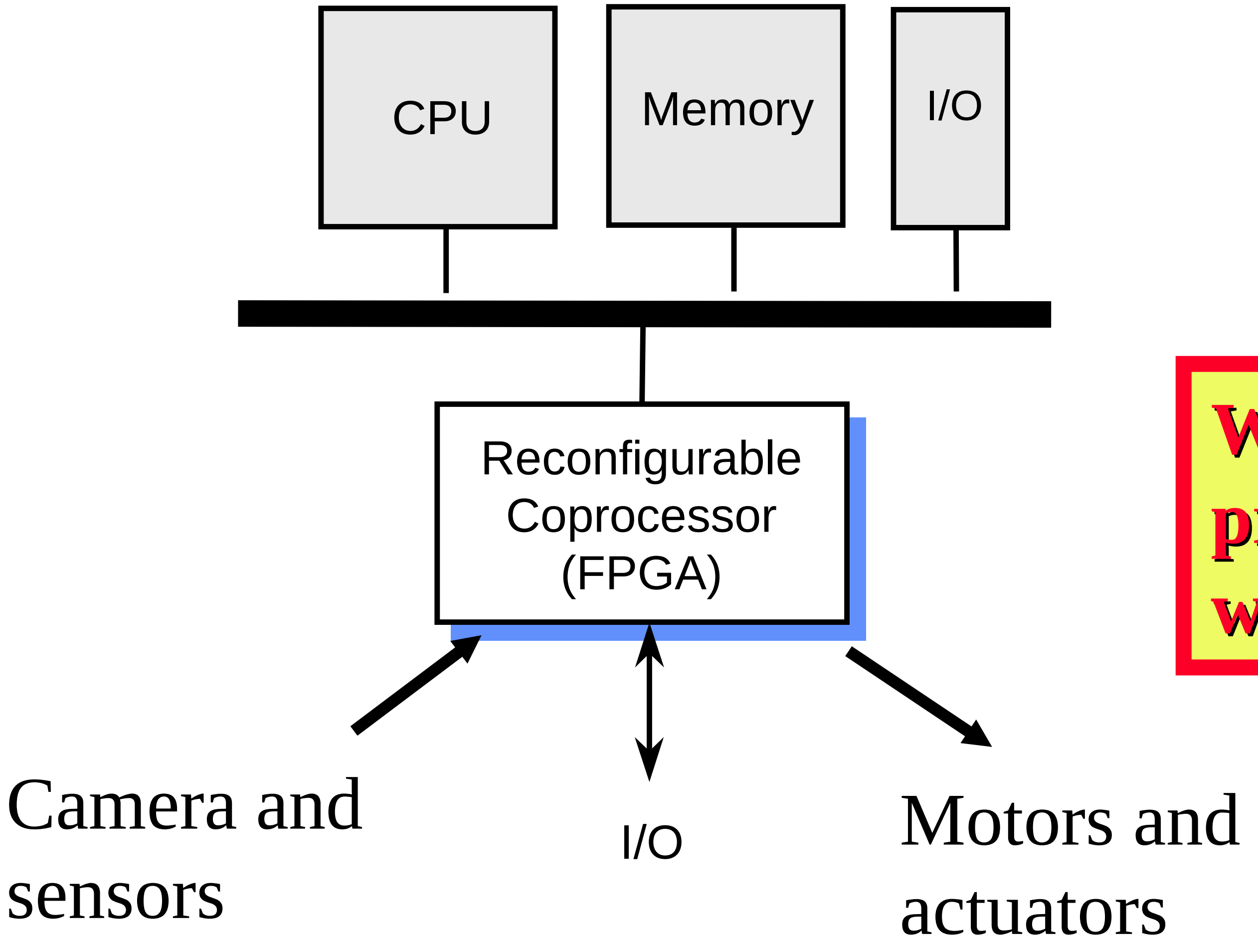
- SRAM-based FPGA
  - Fast, unlimited **reconfiguration**
  - **Dynamic and partially** reconfigurable logic
- Microprocessor interface
- Symmetrical, **hierarchical** and regular structure

XC6200 Family FPGAs

Agenda

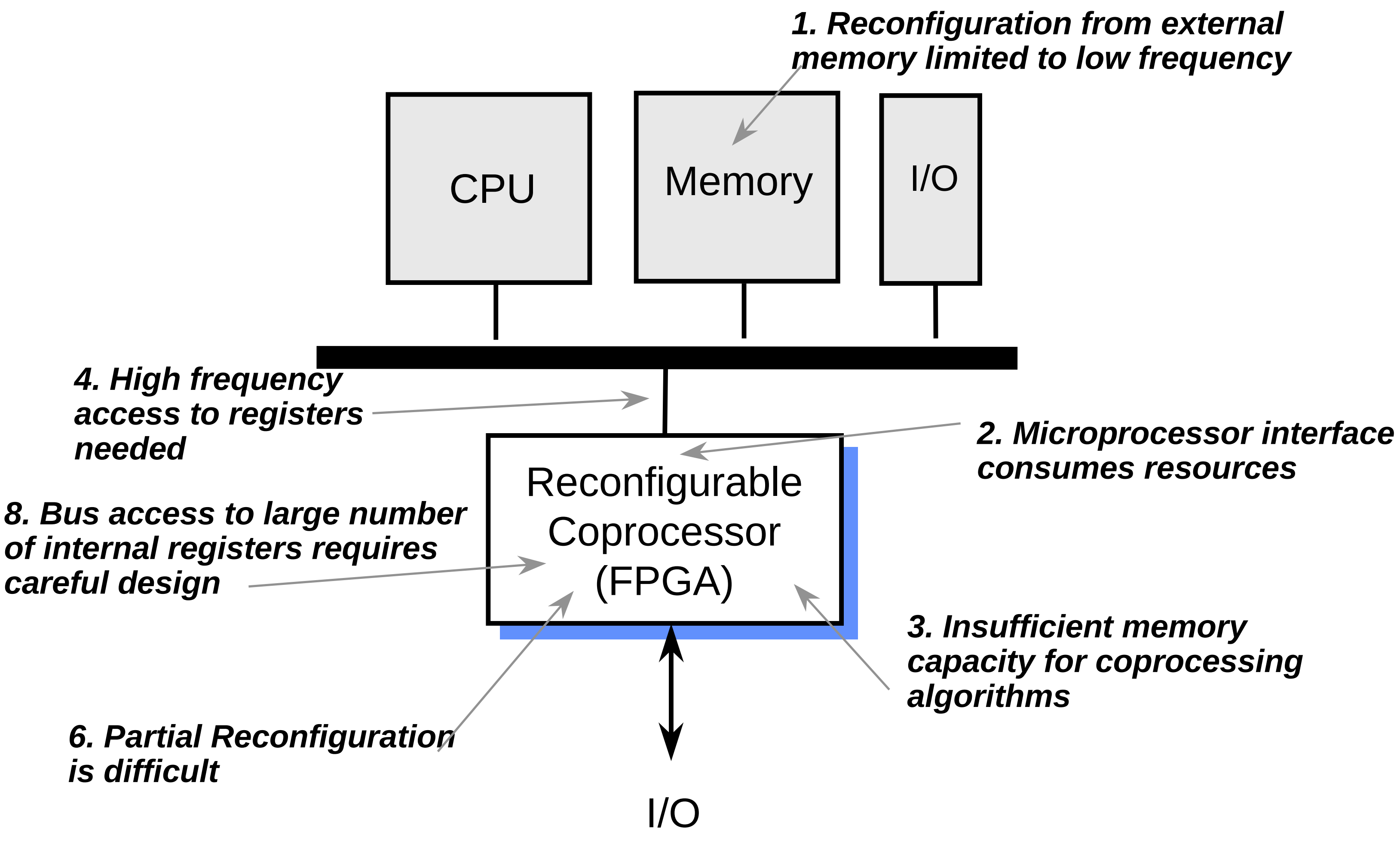
- XC6200 Architecture
- Design Flows
- Library Support
- Applications
- Reconfigurable Processing

Typical
Embedded Control Design

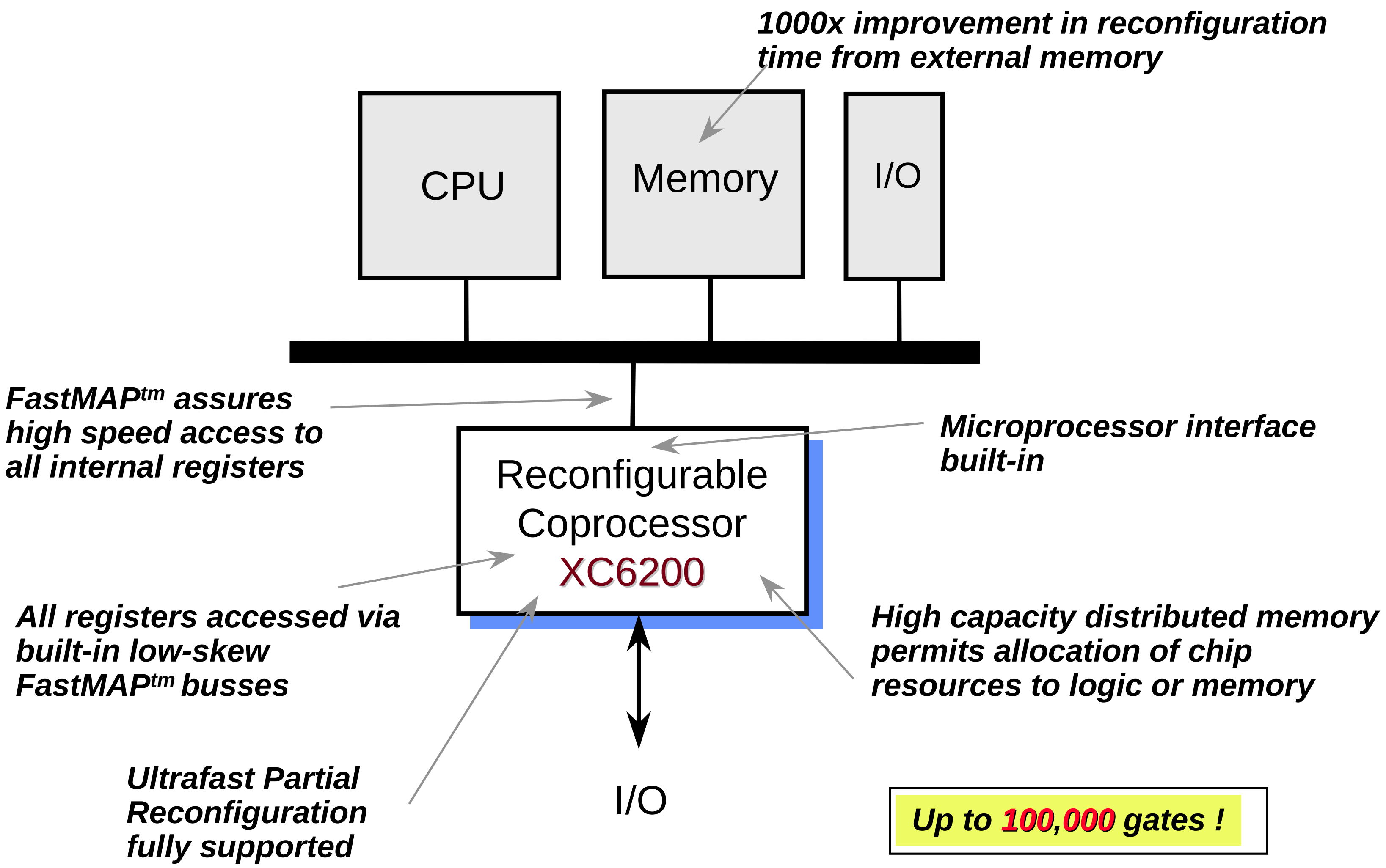


What are the problems with this?

6 Problems Confronting Embedded Control Designers Today


1. Reconfiguration from external memory limited to low frequency
CPU
Memory
I/O
4. High frequency access to registers needed
2. Microprocessor interface consumes resources
8. Bus access to large number of internal registers requires careful design
Reconfigurable Coprocessor (FPGA)
3. Insufficient memory capacity for coprocessing algorithms
6. Partial Reconfiguration is difficult
I/O

XC6200 System Features Meet these requirements of Embedded Coprocessing


1000x improvement in reconfiguration time from external memory
CPU
Memory
I/O
FastMAP^{tm} assures high speed access to all internal registers
Microprocessor interface built-in
Reconfigurable Coprocessor XC6200
All registers accessed via built-in low-skew FastMAP^{tm} busses
High capacity distributed memory permits allocation of chip resources to logic or memory
Ultrafast Partial Reconfiguration fully supported
I/O

Up to 100,000 gates !

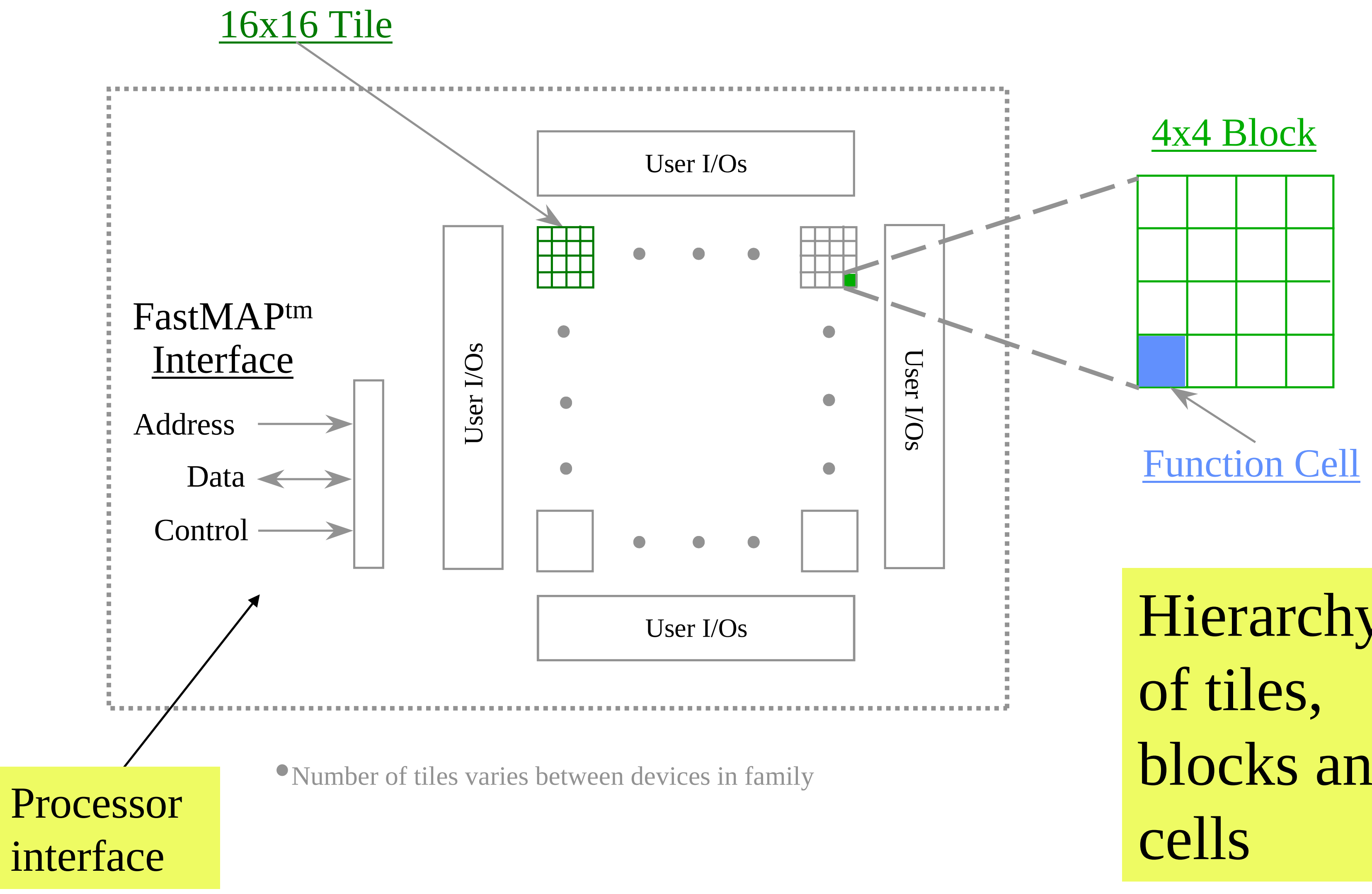
XC6200 Architectural Overview

- Array of **fine grain** function cells, each with a register
  - high gate count for structured logic or regular arrays
- Abundant, hierarchical routing resources
  - global/local
  - use cell logic/use switches
- **Flexible** pin configuration
  - programmable as ***in, out, bidirectional, tristate***
  - CMOS or TTL logic levels

XC6200 Architecture (cont)

- High speed CPU interface for *configuration* and register I/O
  - Programmable bus width (8..32-bits)
  - **Direct** processor read/write access to all user registers
  - All user registers and configuration SRAM mapped into processor address space

XC6200 Architecture

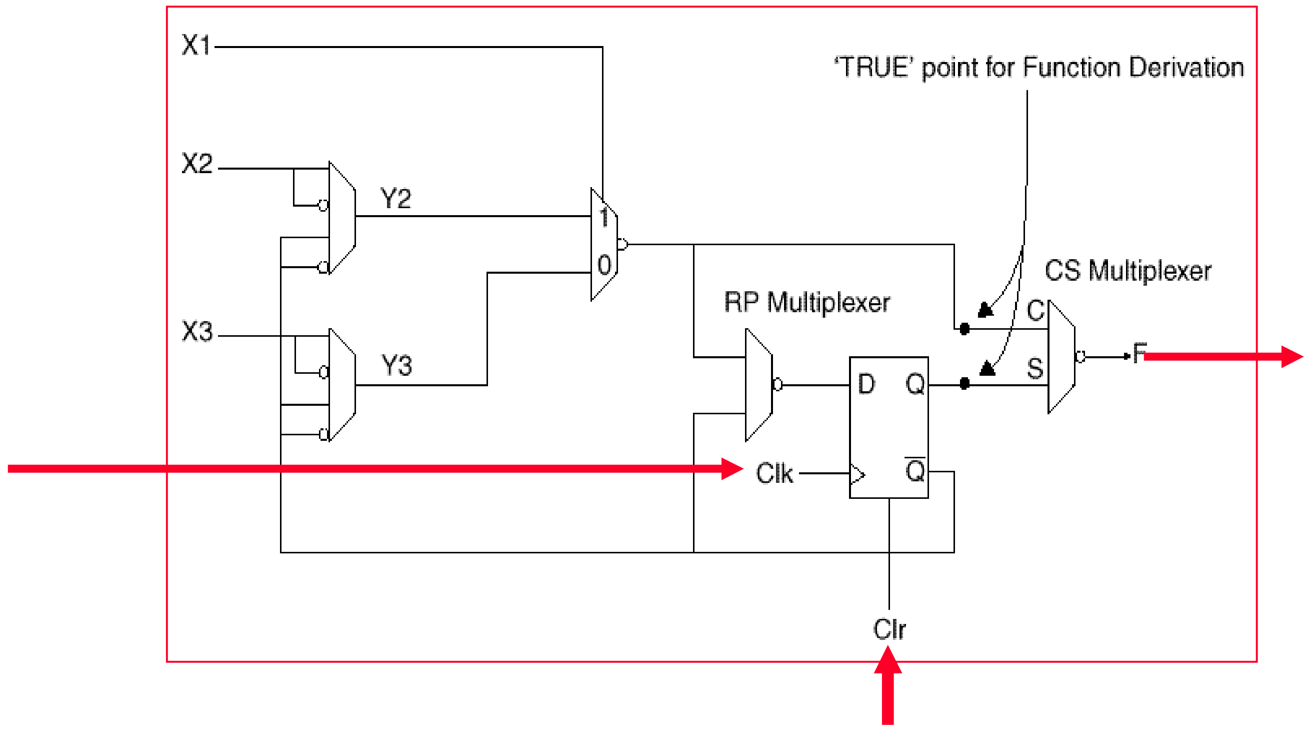


Processor interface

Hierarchy of tiles, blocks and cells

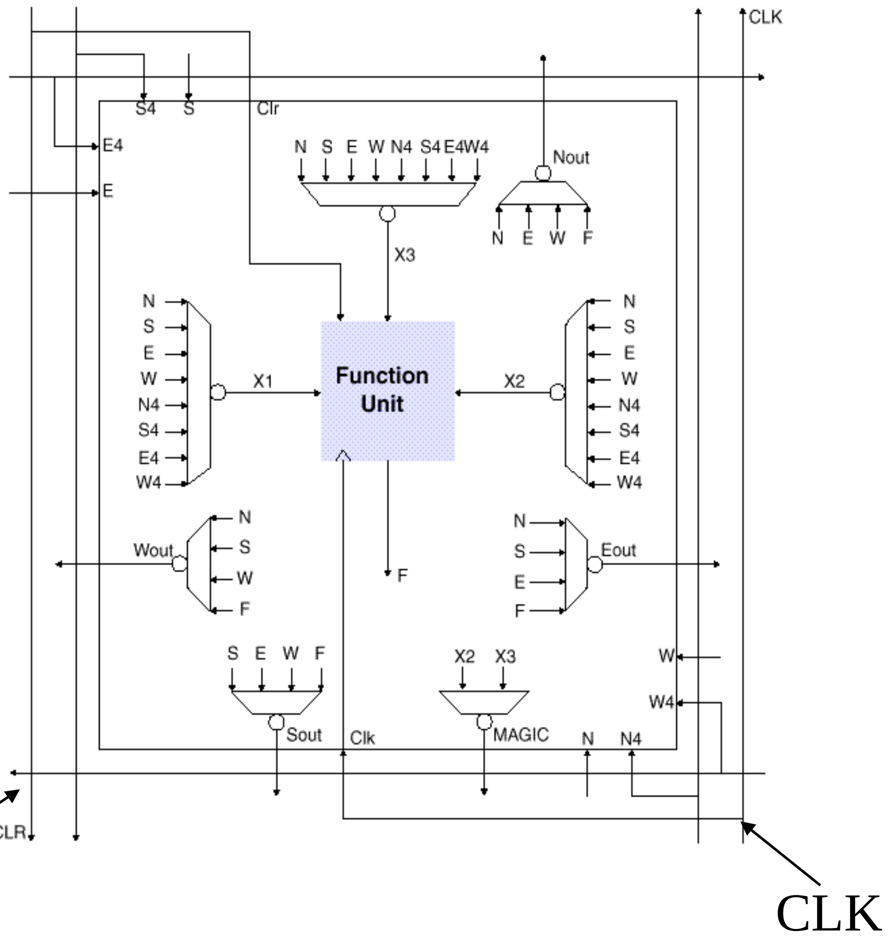
XC6200 Functional Unit

- Design based on the fact that any function of two Boolean variables can be computed by a 2:1 MUX.



XC6200 Unit Cell

- Each unit cell contains a computation unit:
  - D-type register
  - 2-input logic function
  - Nearest neighbor interconnection
  - Individually programmable **from host interface (uP)**



From buses

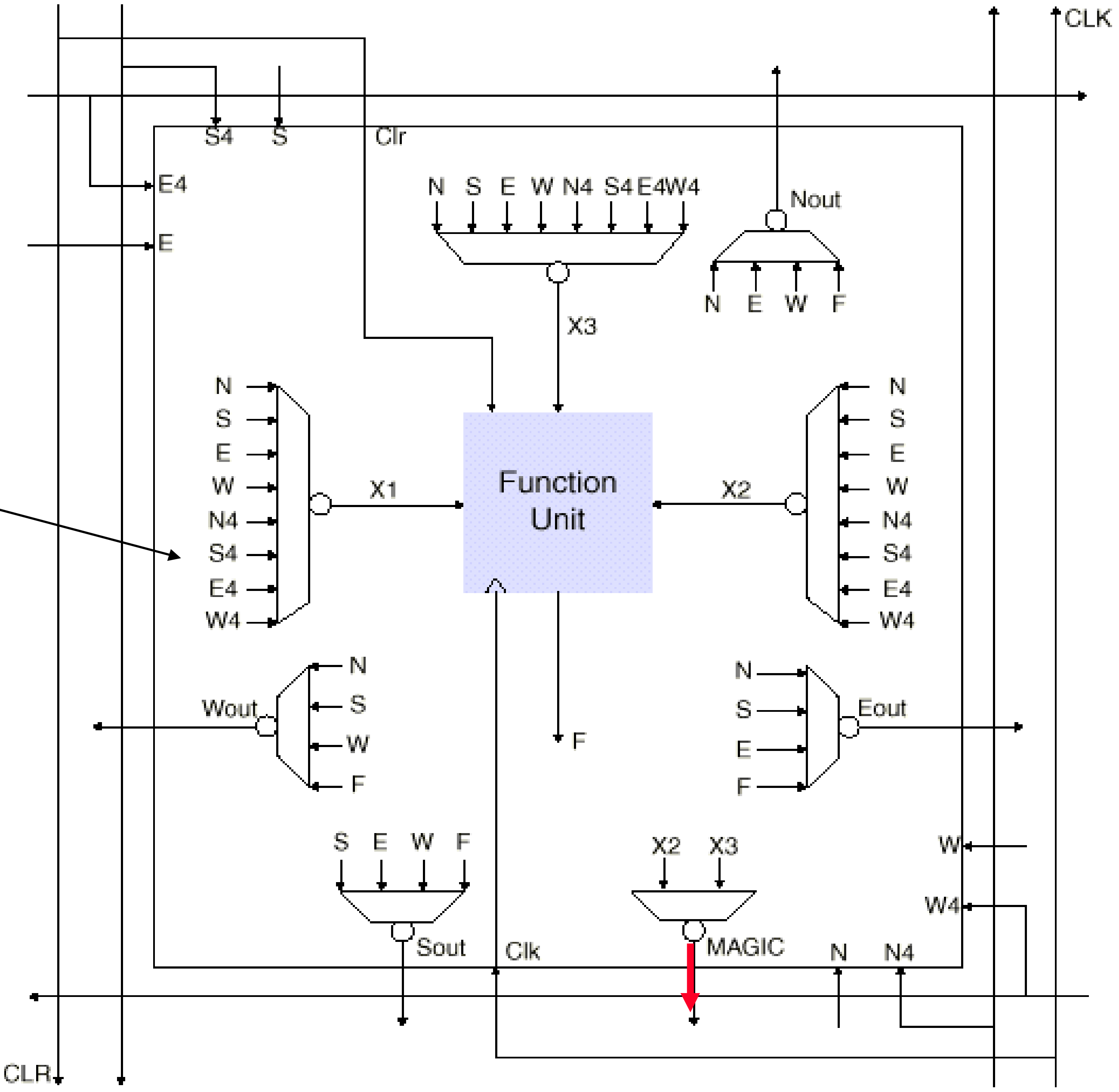


Figure 5. XC6200 Basic Cell

Logical Organization: XC6200 Function Unit

- Function unit allows :
  - any function of 2 variables
  - any flavour of 2:1 mux
  - buffers, inverters, or constant 0s and 1s
  - any of the above in addition to a D-type register
- 3 I/Ps, each from any of 8 directions; O/P to up to 4 directions

Programmable input

Redirect output

Logical Organization: Function Unit.

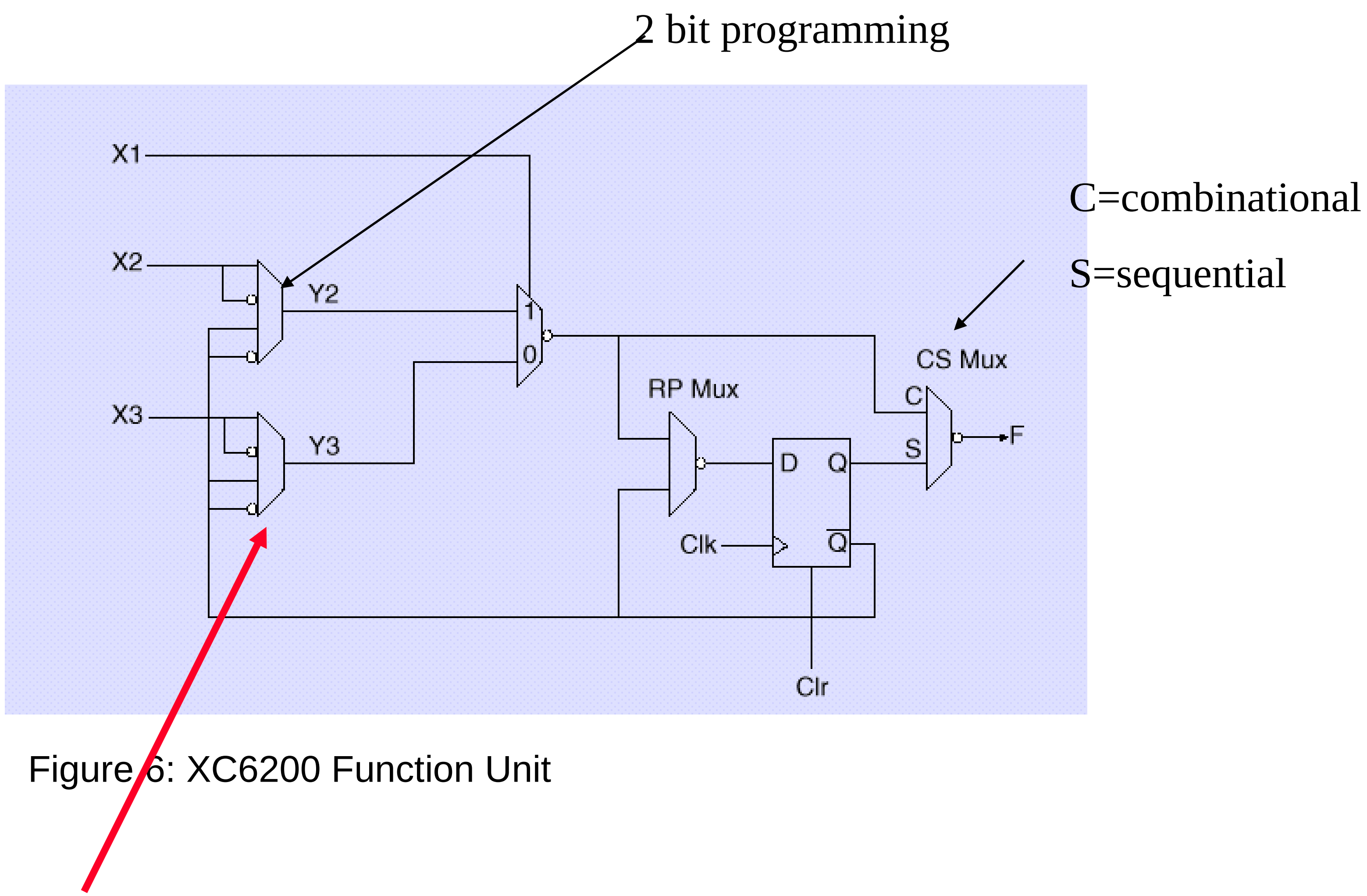


Figure 6: XC6200 Function Unit

Y3 MUX

2+2+1+1 bits

Logical Organization: Function Unit. (cont)

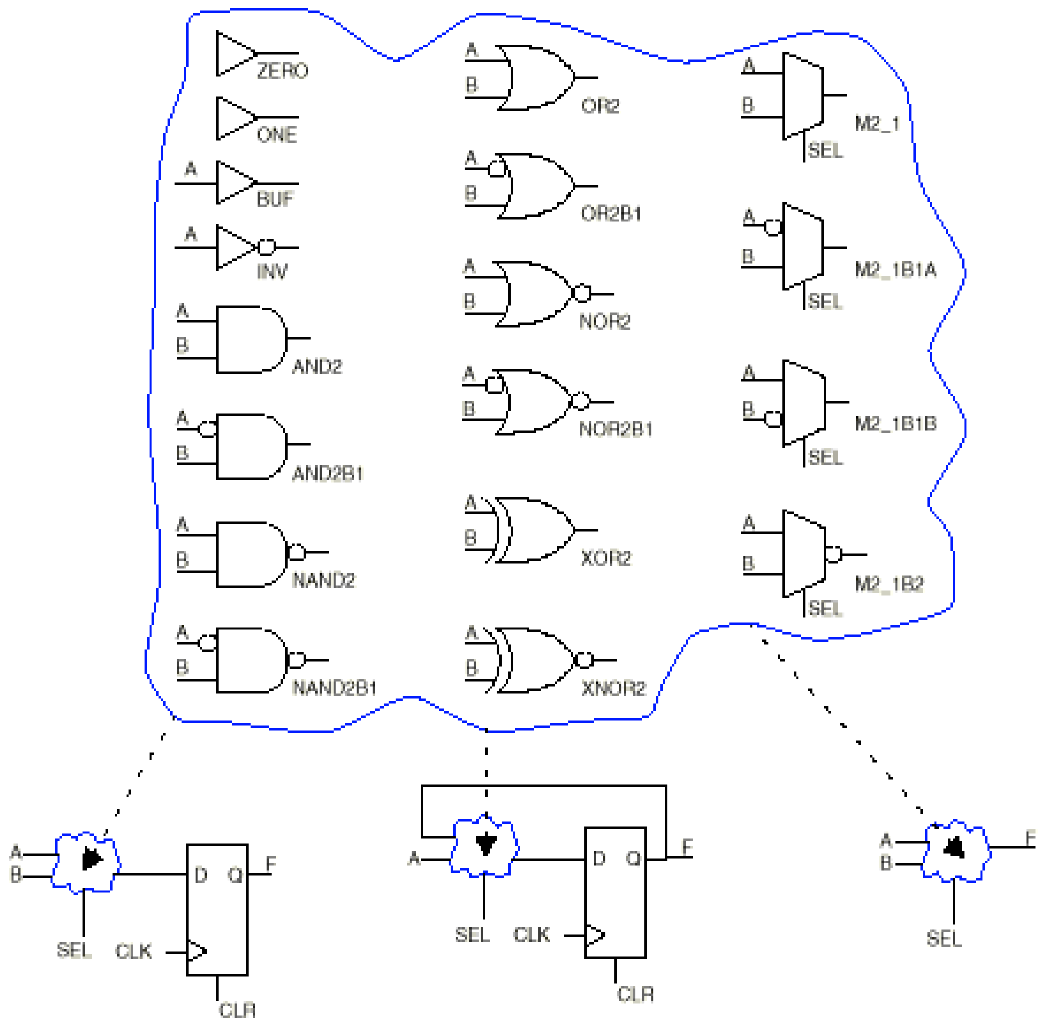
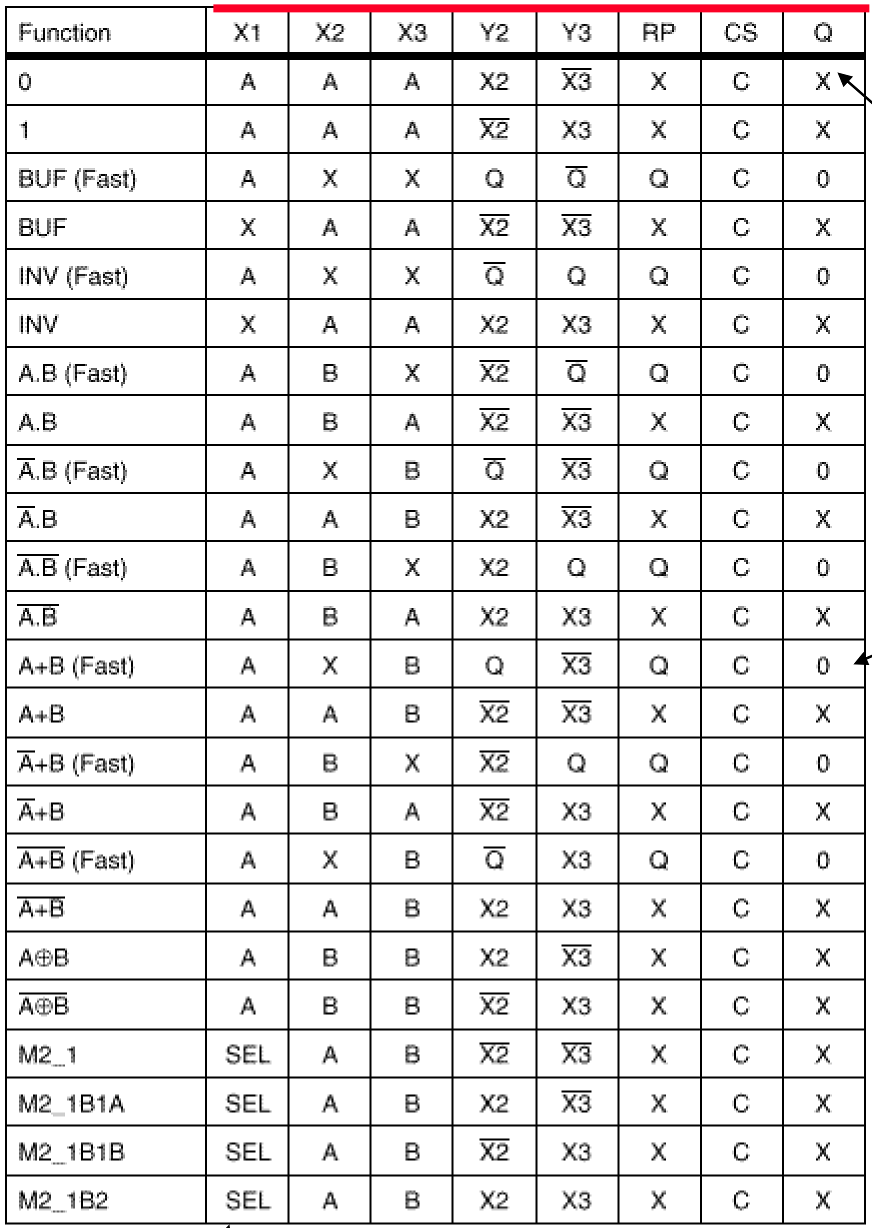


Figure 7. Cell Logic Functions

Three basic modes

Cell logic function table

On any 2 of these arguments



Function	X1	X2	X3	Y2	Y3	RP	CS	Q
0	A	A	A	X2	$\overline{X3}$	X	C	X
1	A	A	A	$\overline{X2}$	X3	X	C	X
BUF (Fast)	A	X	X	Q	$\overline{Q}$	Q	C	0
BUF	X	A	A	$\overline{X2}$	$\overline{X3}$	X	C	X
INV (Fast)	A	X	X	$\overline{Q}$	Q	Q	C	0
INV	X	A	A	X2	X3	X	C	X
A.B (Fast)	A	B	X	$\overline{X2}$	$\overline{Q}$	Q	C	0
A.B	A	B	A	$\overline{X2}$	$\overline{X3}$	X	C	X
$\overline{A}$.B (Fast)	A	X	B	$\overline{Q}$	$\overline{X3}$	Q	C	0
$\overline{A}$.B	A	A	B	X2	$\overline{X3}$	X	C	X
$\overline{A.B}$ (Fast)	A	B	X	X2	Q	Q	C	0
$\overline{A.B}$	A	B	A	X2	X3	X	C	X
A+B (Fast)	A	X	B	Q	$\overline{X3}$	Q	C	0
A+B	A	A	B	$\overline{X2}$	$\overline{X3}$	X	C	X
$\overline{A}$+B (Fast)	A	B	X	$\overline{X2}$	Q	Q	C	0
$\overline{A}$+B	A	B	A	$\overline{X2}$	X3	X	C	X
$\overline{A+B}$ (Fast)	A	X	B	$\overline{Q}$	X3	Q	C	0
$\overline{A+B}$	A	A	B	X2	X3	X	C	X
A⊕B	A	B	B	X2	$\overline{X3}$	X	C	X
$\overline{A⊕B}$	A	B	B	$\overline{X2}$	X3	X	C	X
M2_1	SEL	A	B	$\overline{X2}$	$\overline{X3}$	X	C	X
M2_1B1A	SEL	A	B	X2	$\overline{X3}$	X	C	X
M2_1B1B	SEL	A	B	$\overline{X2}$	X3	X	C	X
M2_1B2	SEL	A	B	X2	X3	X	C	X

Shows what selected by corresponding mux

0 for fast

multiplexers

sel

Logical Organization: Function Unit. (cont)

Function	X1	X2	X3	Y2	Y3	RP	CS	Q
0	A	A	A	X2	$\overline{X3}$	X	C	X
1	A	A	A	$\overline{X2}$	X3	X	C	X
BUF (Fast)	A	X	X	Q	$\overline{Q}$	Q	C	0
BUF	X	A	A	$\overline{X2}$	$\overline{X3}$	X	C	X
INV (Fast)	A	X	X	$\overline{Q}$	Q	Q	C	0
INV	X	A	A	X2	X3	X	C	X
A.B (Fast)	A	B	X	$\overline{X2}$	$\overline{Q}$	Q	C	0
A.B	A	B	A	$\overline{X2}$	$\overline{X3}$	X	C	X

Physical Organization: Cells, Blocks and Tiles

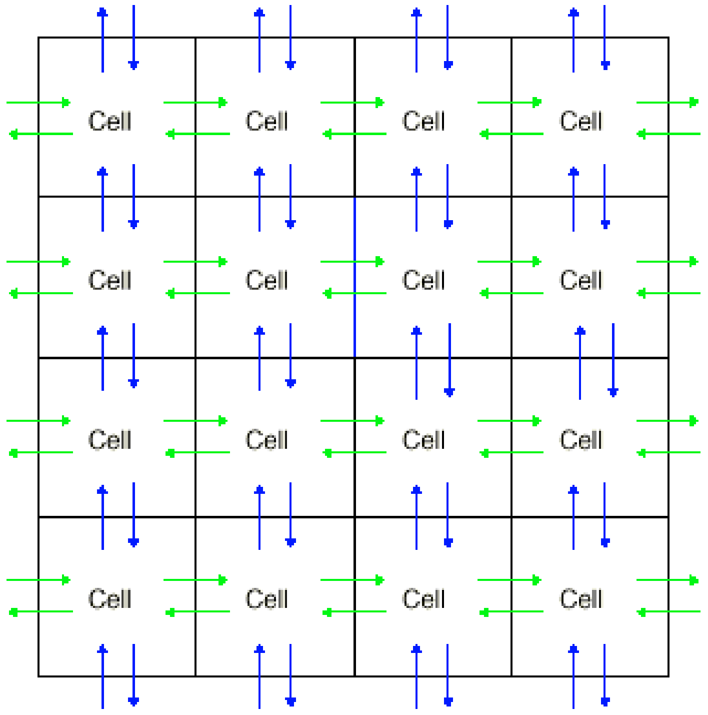


Figure 1. Nearest-Neighbor Interconnect Array Structure

Bus of length 4

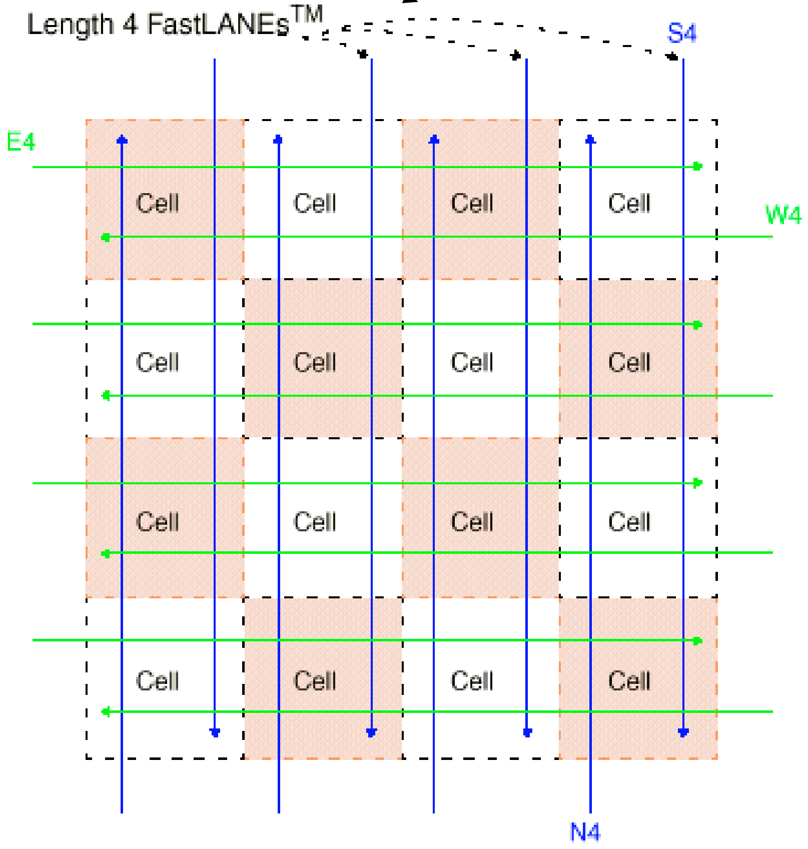
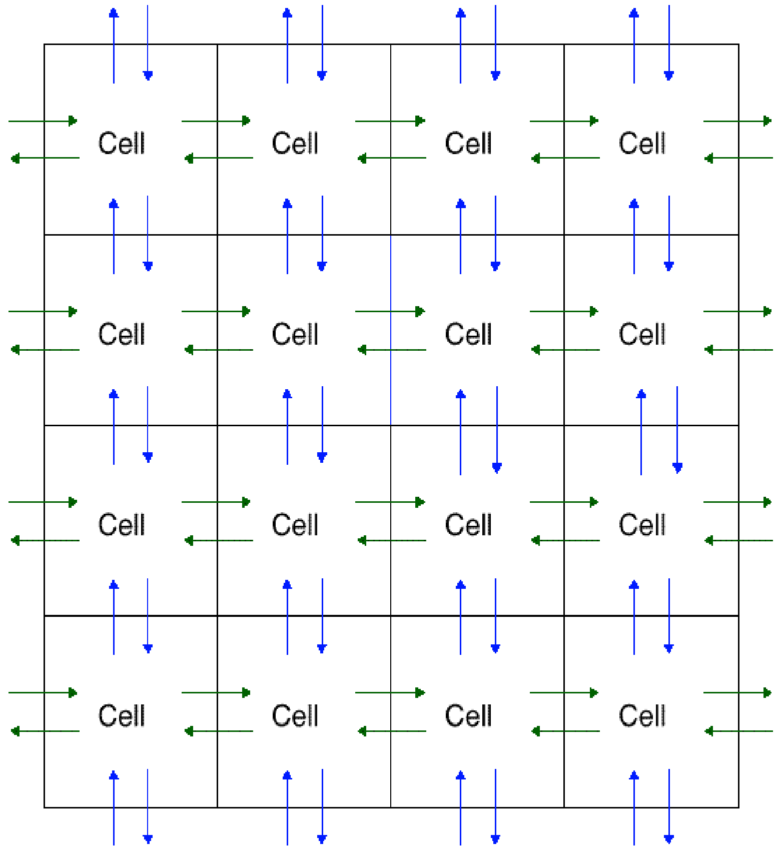


Figure 2. XC6200 4x4 Cell Block

XC6200 Architecture

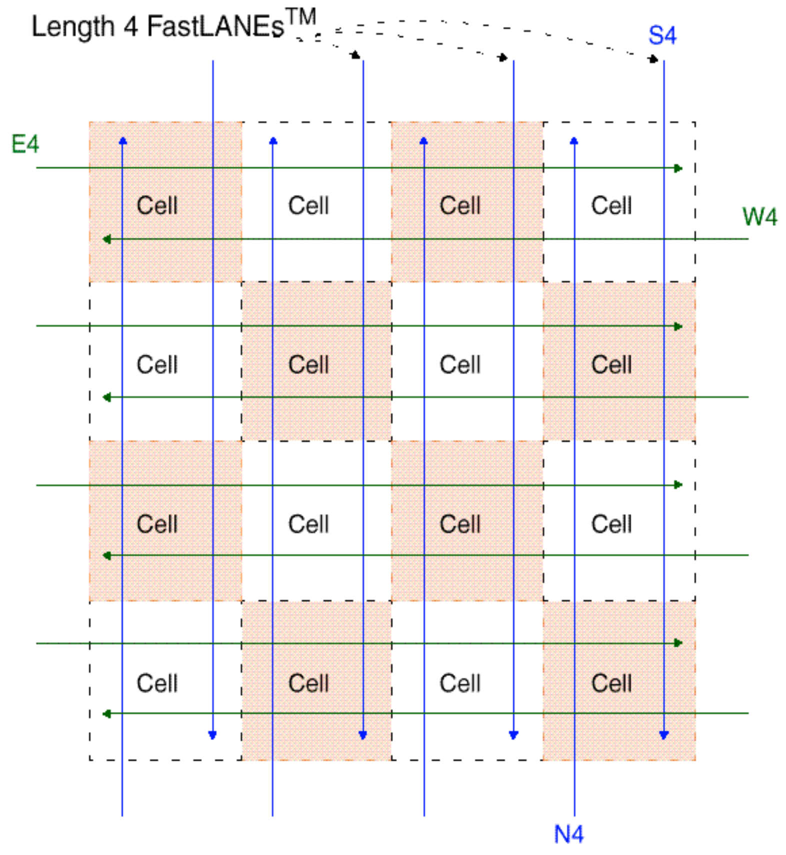
Regular connections to nearest neighbors

- Large array of simple, configurable cells (sea of gates)
- Each cell:
  - D-Type register
  - Logic function
  - **Nearest-neighbor interconnection**
  - **Grouped in 4x4 block**

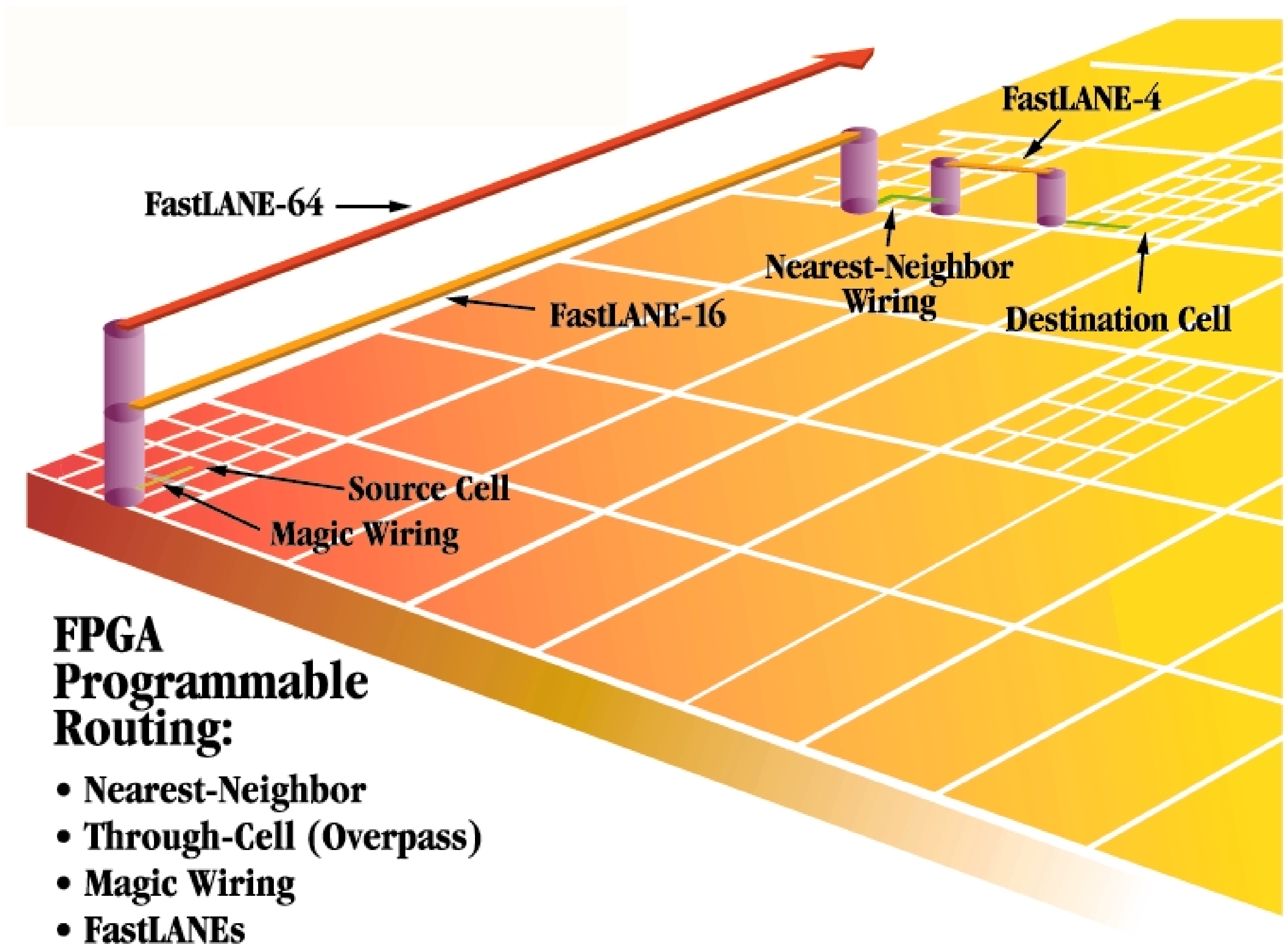


XC6200 Architecture

- 16 (4x4) neighbor-connected cells **are grouped together** to form a larger cellular array
- Communication "lanes" available between neighboring 4x4 cell blocks



Routing Resources Example



FPGA Programmable Routing:

- Nearest-Neighbor
- Through-Cell (Overpass)
- Magic Wiring
- FastLANEs

Physical Organization: Cells, Blocks and Tiles (cont)

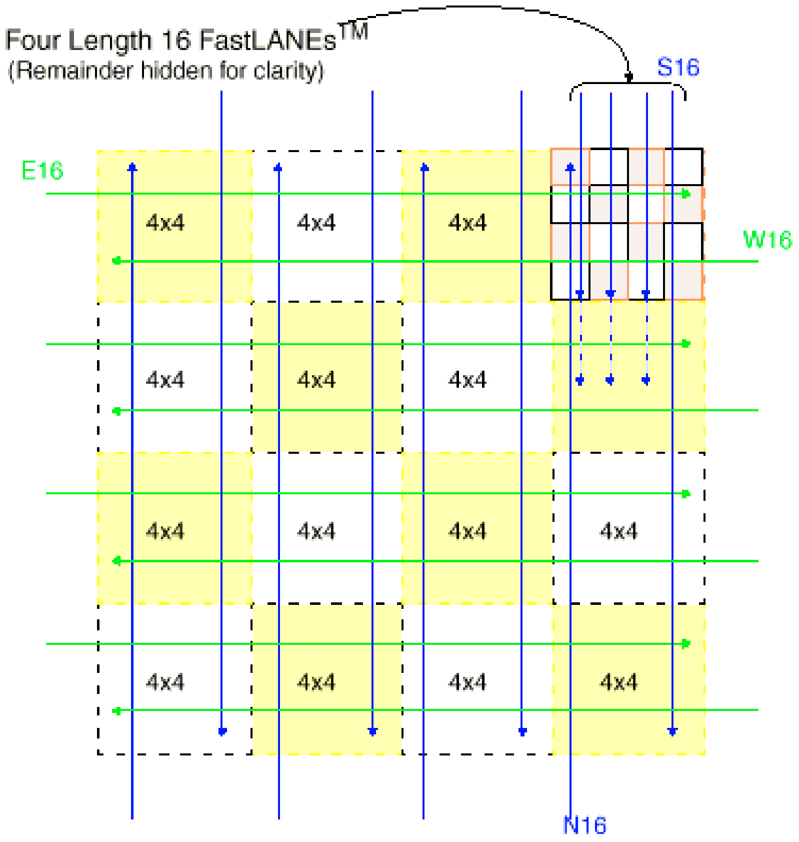
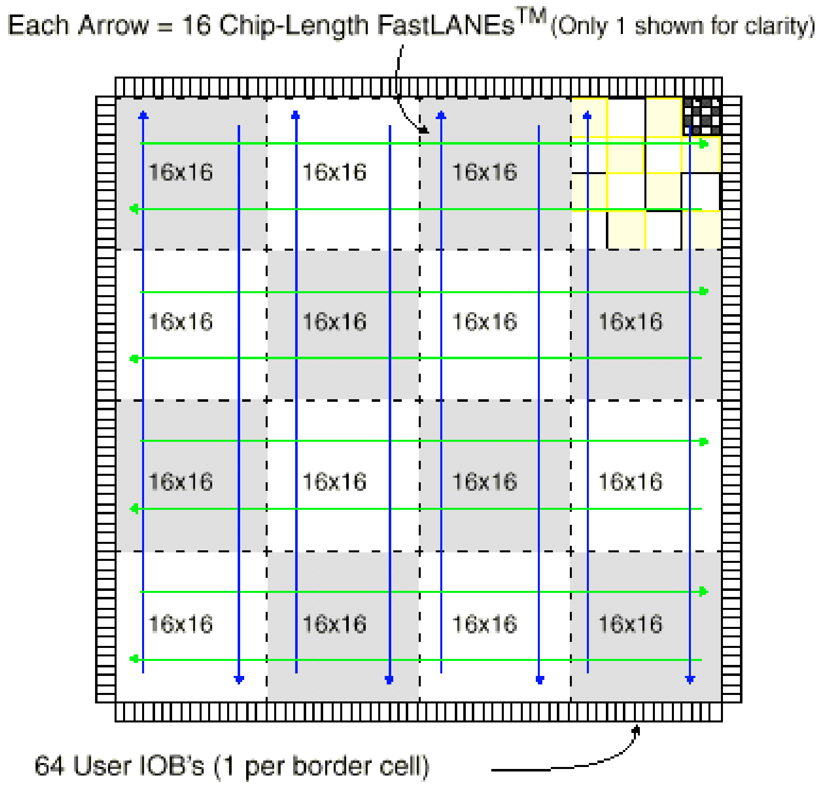


Figure 3. XC6200 16x16 Cell Block



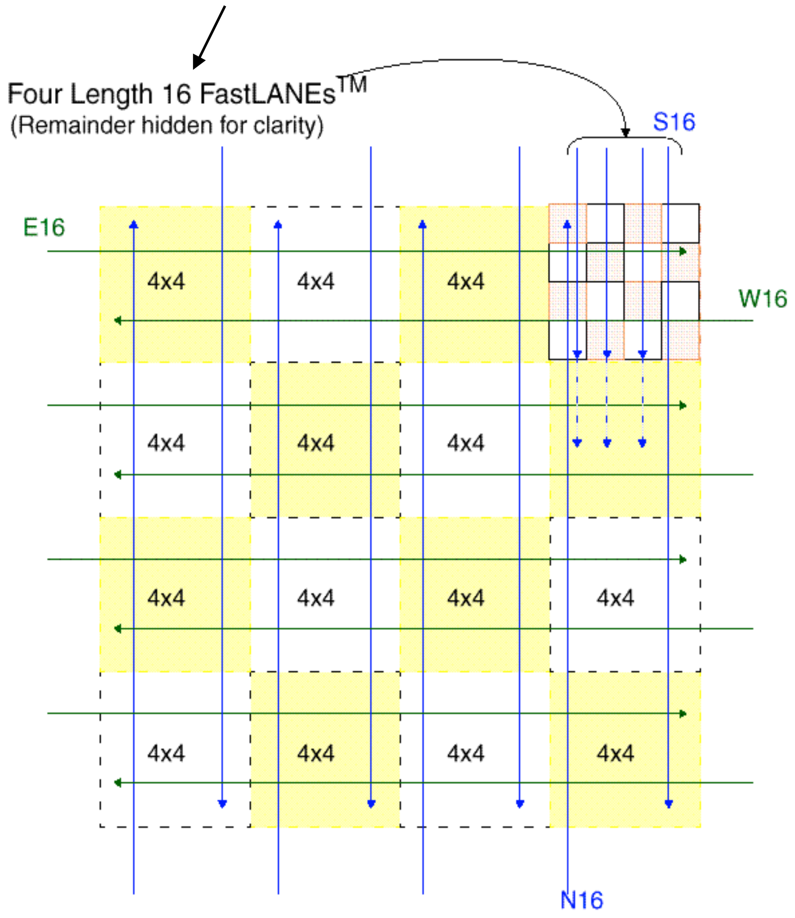
Each Arrow = 16 Chip-Length FastLANEs™ (Only 1 shown for clarity)
16x16
64 User IOB's (1 per border cell)

Figure 4. XC6216 Device

XC6200 Architecture

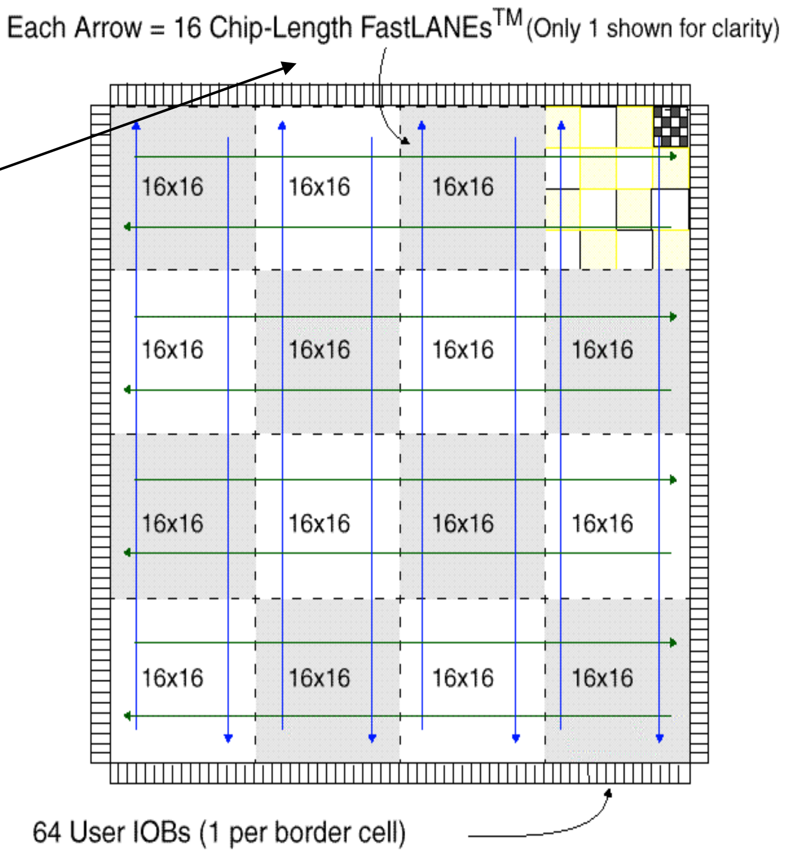
- A 4x4 array of the previously shown 4x4 blocks forms a 16x16 block
- **Length 16 FastLANEs** connect these larger arrays

Length 16 fast lanes



XC6200 Architecture

- A 4x4 array of the 16x16 blocks forms the central 64x64 cell array
- Chip-Length FastLANEs connect
- Central block surrounded by I/O pads


Each Arrow = 16 Chip-Length FastLANEs™ (Only 1 shown for clarity)
16x16
64 User IOBs (1 per border cell)

XC6200 Routing

- Each level of hierarchy has its own associated routing resources
  - **Unit cells, 4x4, 16x16, 64x64 cell blocks**
- Routing does not use a unit cell's resources
- Switches at the *edge of the blocks* provide for connections between the levels of interconnect

Routing Switches:

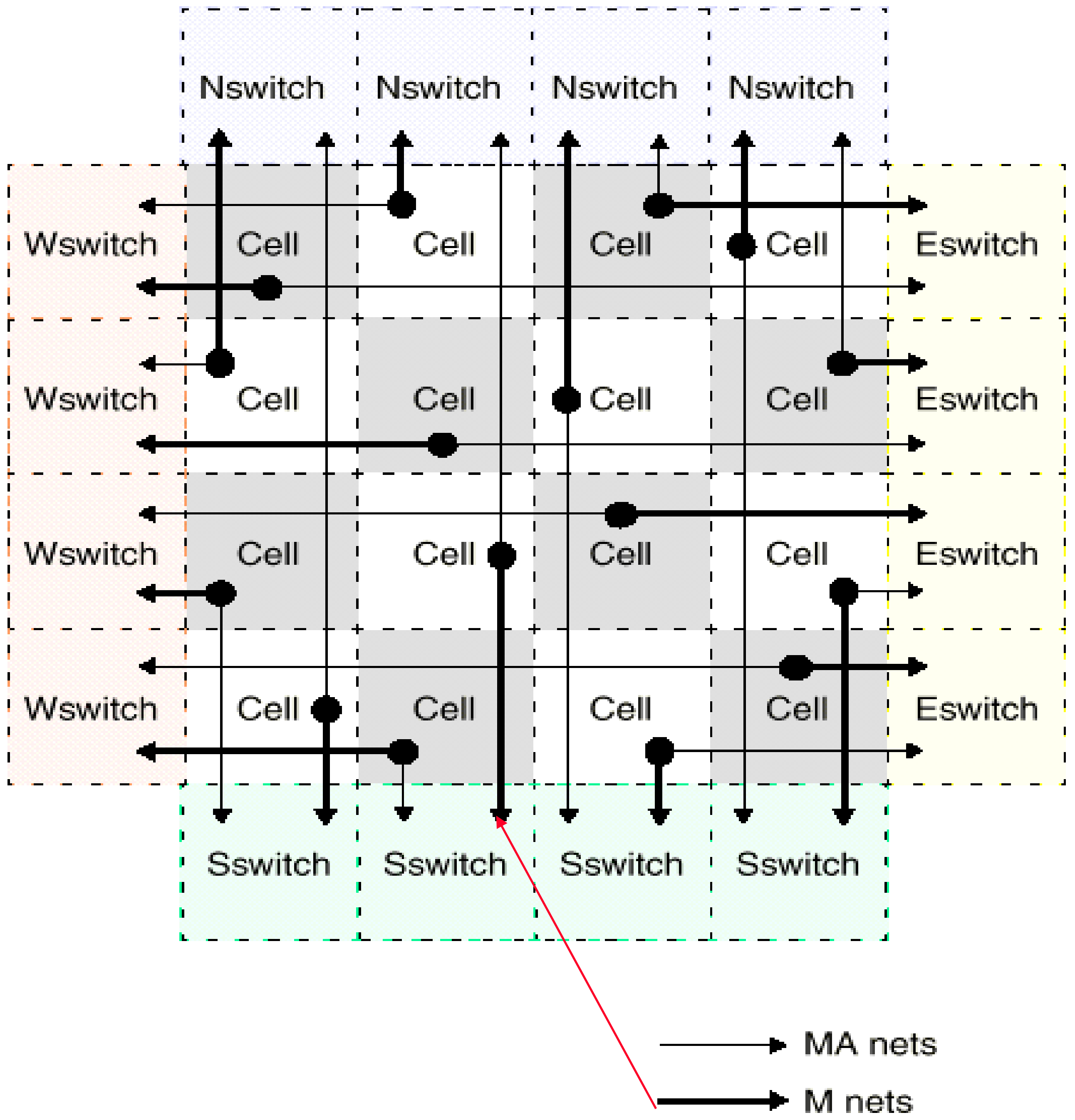


Figure 8. Routing Switches at 4x4 Block Boundary

North and South Switches:

This slide shows what is connected to routing switches

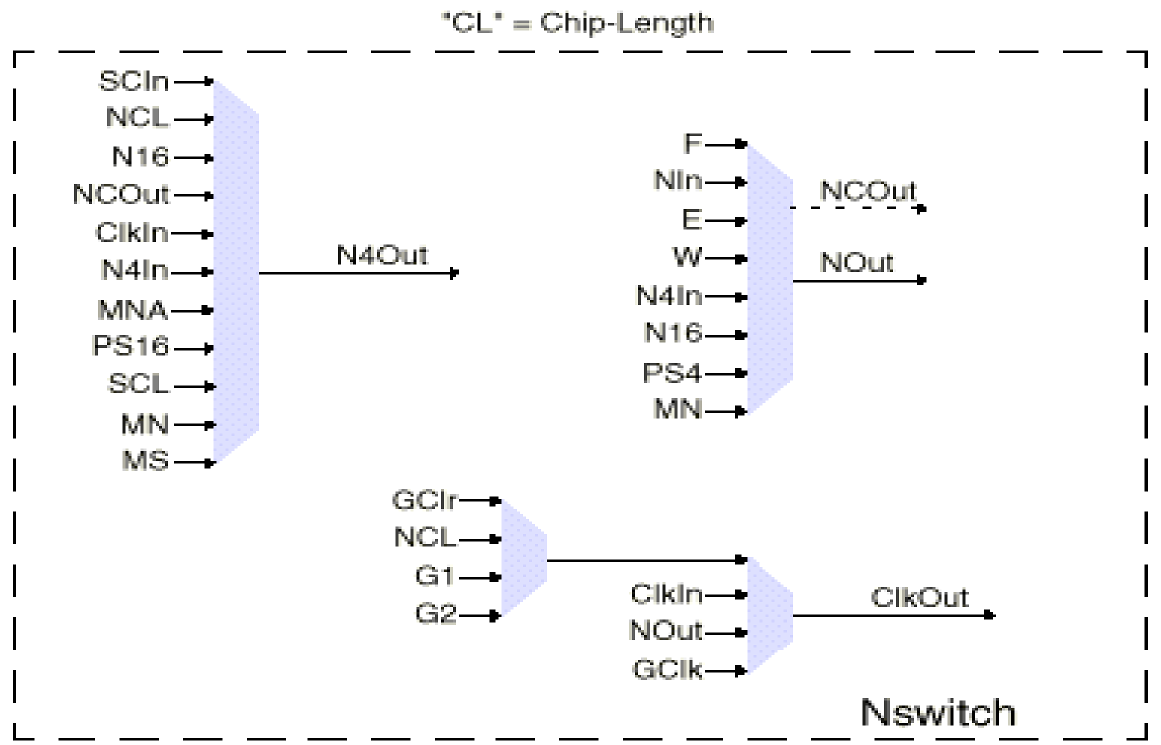


Figure 9. Contents of Nswitch

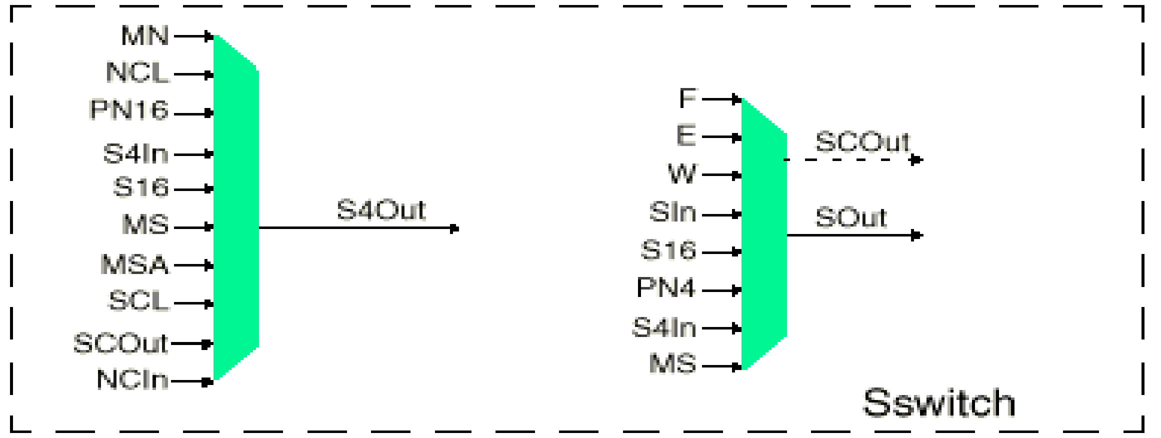


Figure 10. Contents of Sswitch

East and West Switches:

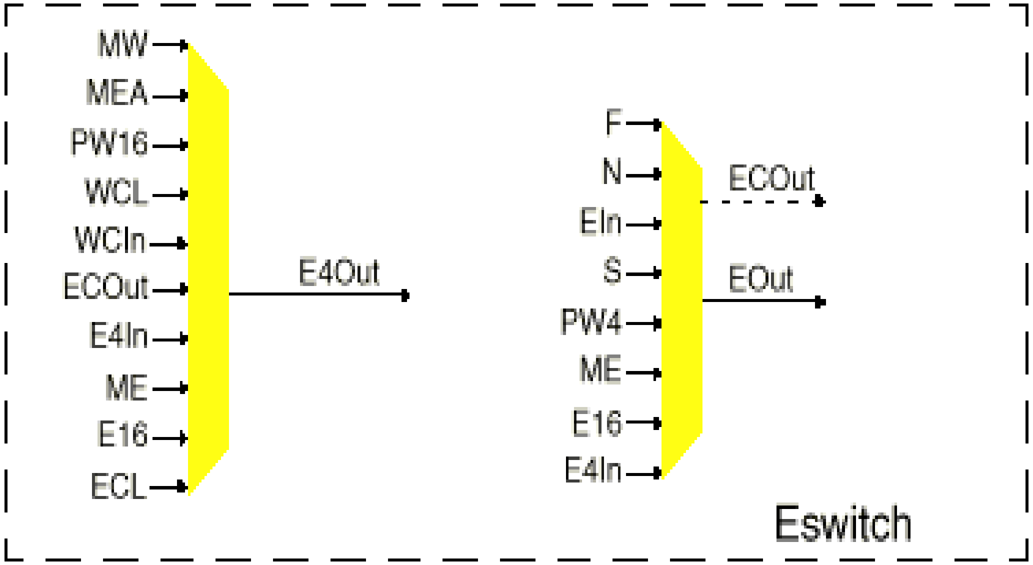


Figure 11. Contents of Eswitch

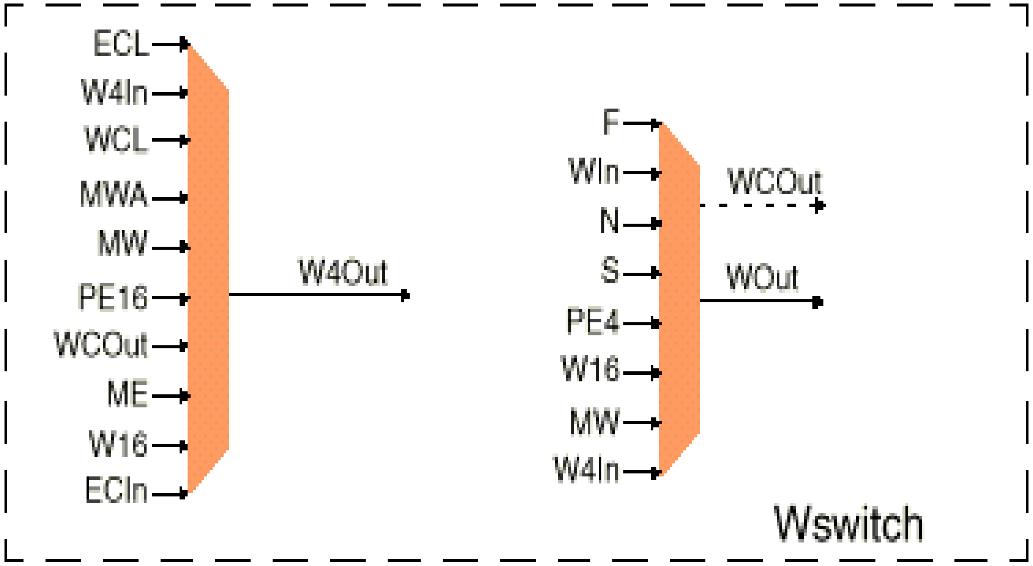
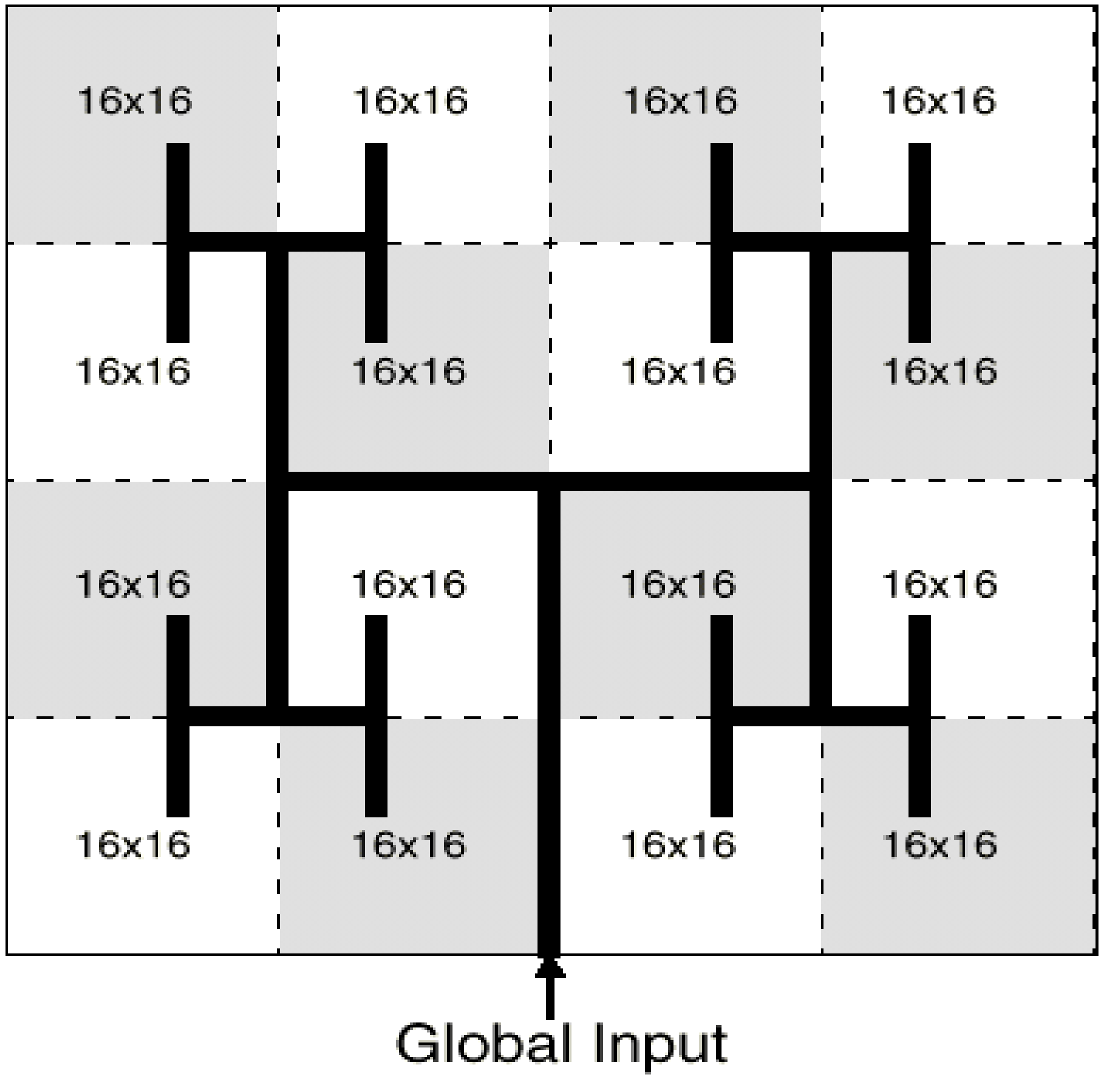


Figure 12. Contents of Wswitch

Clock Distribution:

Clock connections are fixed, for speed



Low Skew 'H' Distribution Of Global Signals (XC6216)

Clear Distribution:

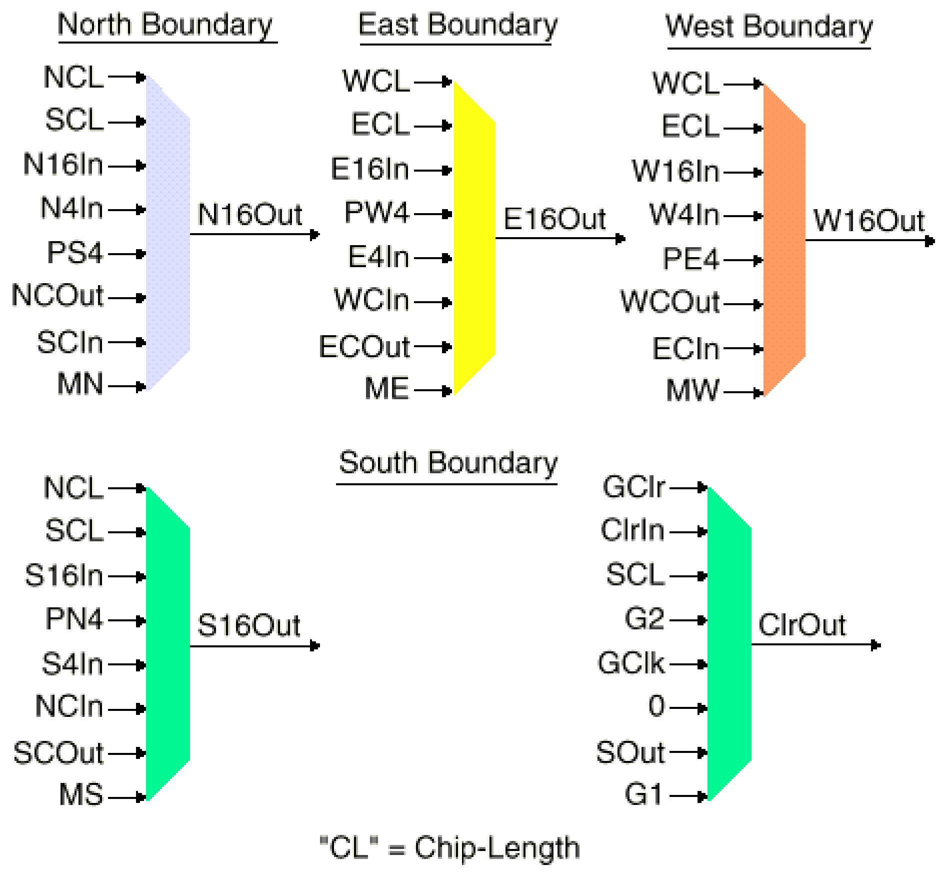


Figure 14. Additional Switches at 16x16 Boundaries

Input/Output Architecture:

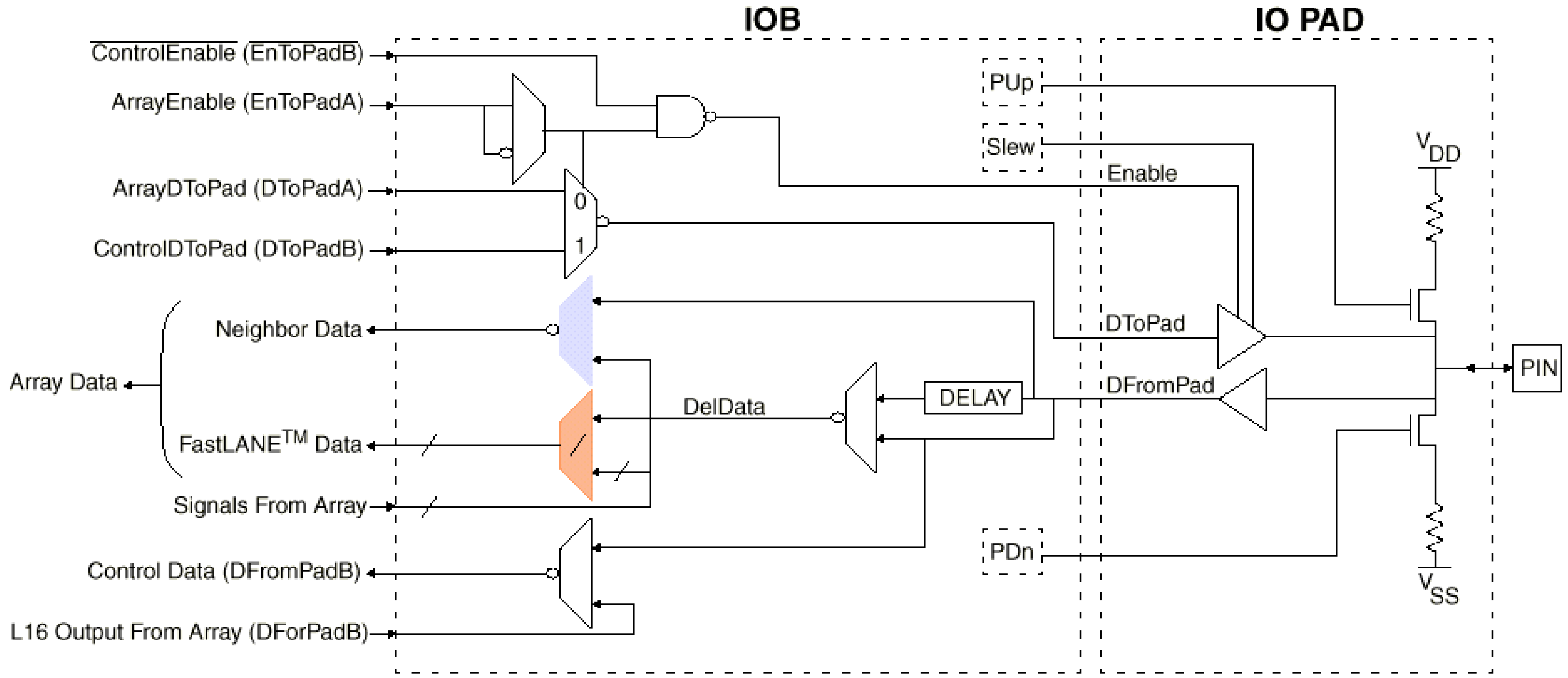


Figure 15. Input/Output Architecture

Flexible pin configuration

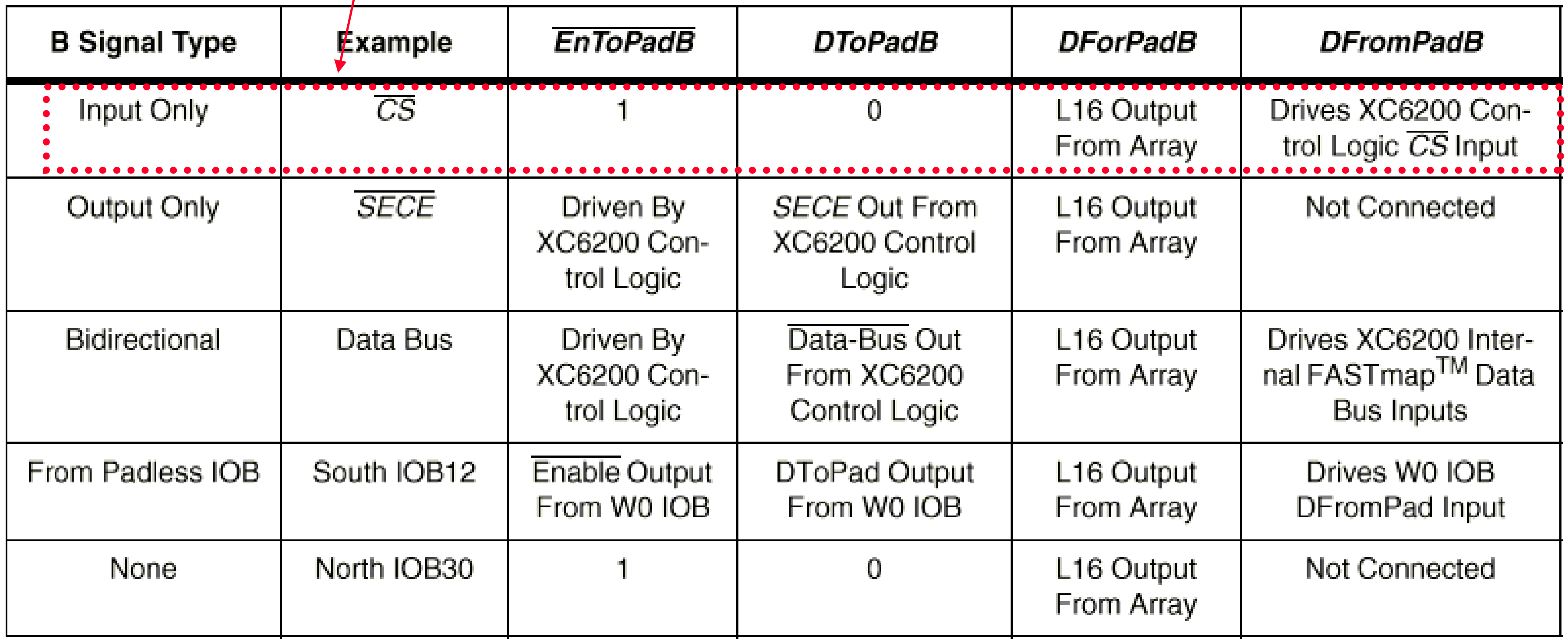
=> programmable as ***in, out, bidirectional, tristate***

=>CMOS or TTL logic levels

Connections Between IOB’s and Built-In XC6200 Control Logic:

This is programmed for input only

Table 3: Connections Between IOB’s And Built-In XC6200 Control Logic



B Signal Type	Example	$\overline{EnToPadB}$	*DToPadB*	*DForPadB*	*DFromPadB*
Input Only	$\overline{CS}$	1	0	L16 Output From Array	Drives XC6200 Control Logic $\overline{CS}$ Input
Output Only	$\overline{SECE}$	Driven By XC6200 Control Logic	$\overline{SECE}$ Out From XC6200 Control Logic	L16 Output From Array	Not Connected
Bidirectional	Data Bus	Driven By XC6200 Control Logic	$\overline{\text{Data-Bus}}$ Out From XC6200 Control Logic	L16 Output From Array	Drives XC6200 Internal FASTmap™ Data Bus Inputs
From Padless IOB	South IOB12	$\overline{\text{Enable}}$ Output From W0 IOB	DToPad Output From W0 IOB	L16 Output From Array	Drives W0 IOB DFromPad Input
None	North IOB30	1	0	L16 Output From Array	Not Connected

Array Data Sources In West IOB's:

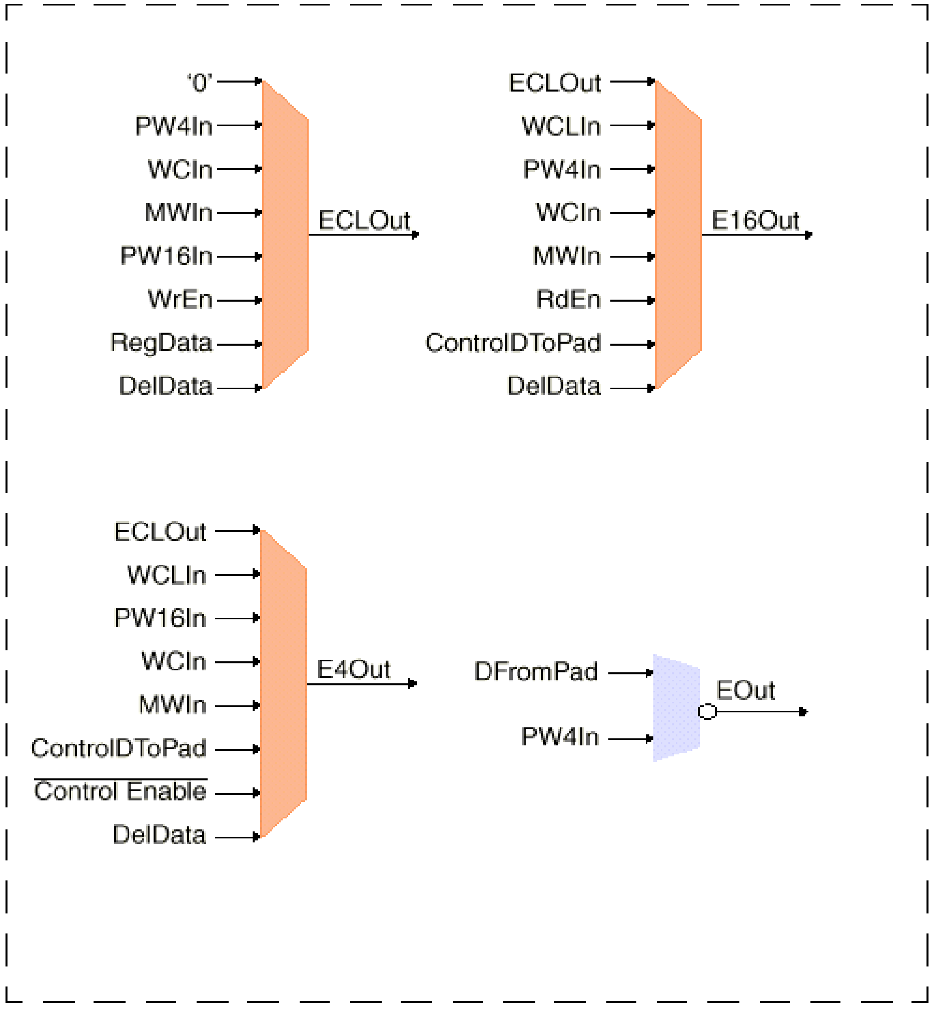
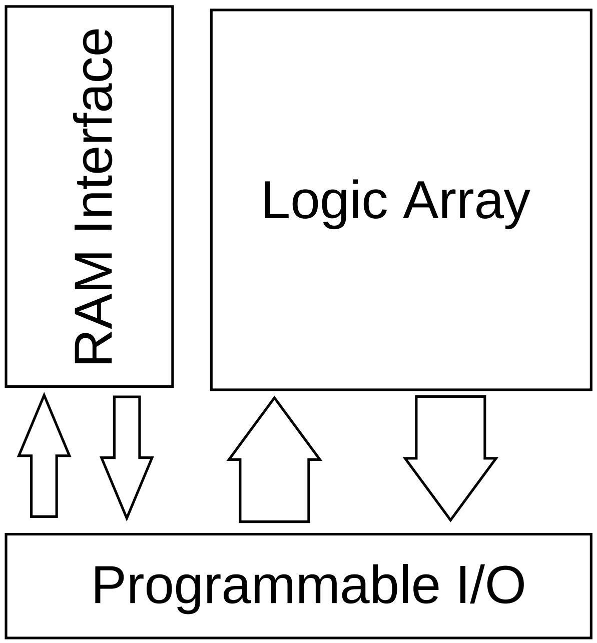


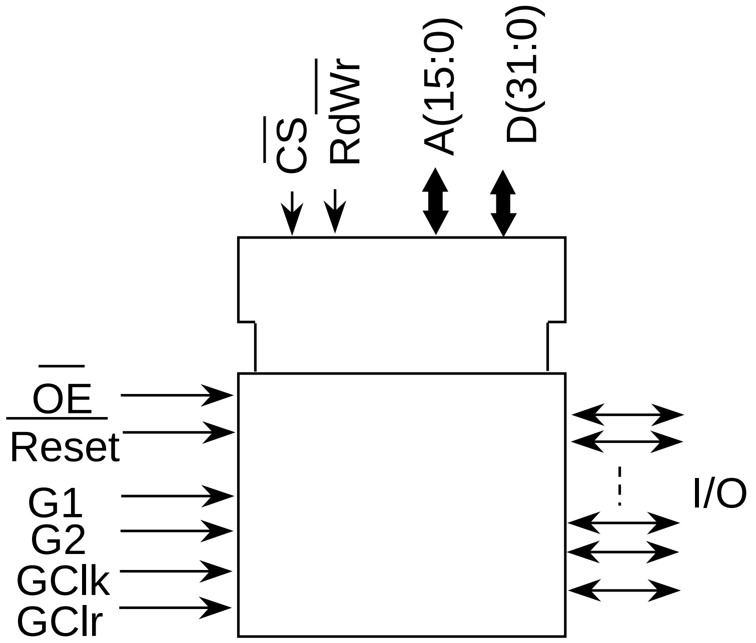
Figure 17. Array Data Sources In West IOB's

XC6200 Device Organization

- Conceptual view



- Logic symbol



RAM on board

Easy interface to uP

FastMAP **CPU Interface**

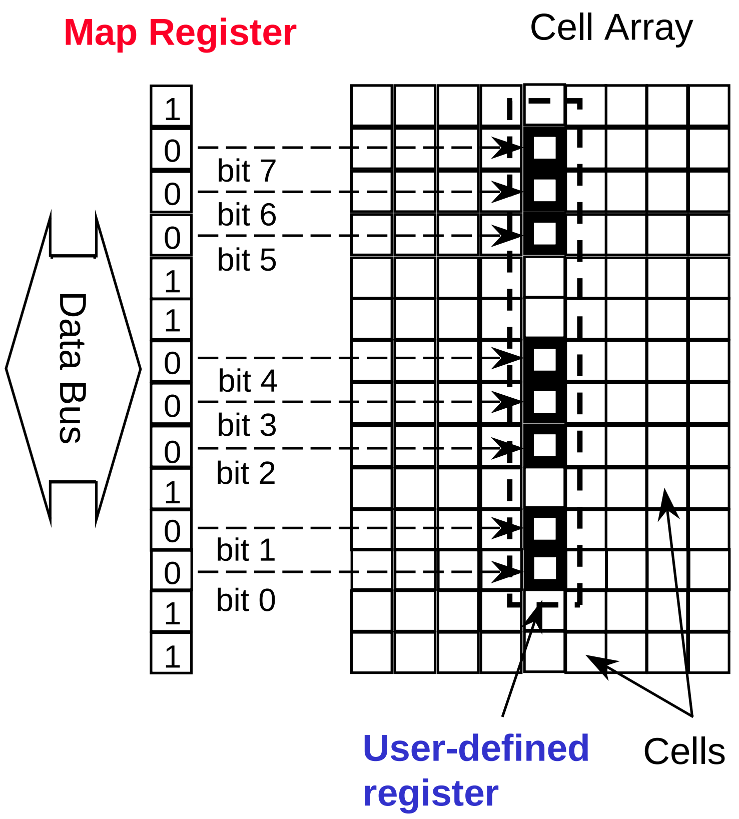
- The industry's only ***random access*** configuration interface
  - allows for extremely fast full or partial device configuration - you only program **the bits you need**
- Allows direct CPU (random) access to **user registers**
  - supports *"coprocessing"* applications.

FastMAP CPU Interface (cont)

- Easily interfaced to most microprocessors and microcontrollers
  - “memory mapped” architecture makes it just like designing with SRAM

FastMAP (cont)

- ***Map Register*** allows mapping of *user registers* on to 8, 16, or 32 bit data bus
- Allows unconstrained register placement
- Obviates need for complex shift and mask operations



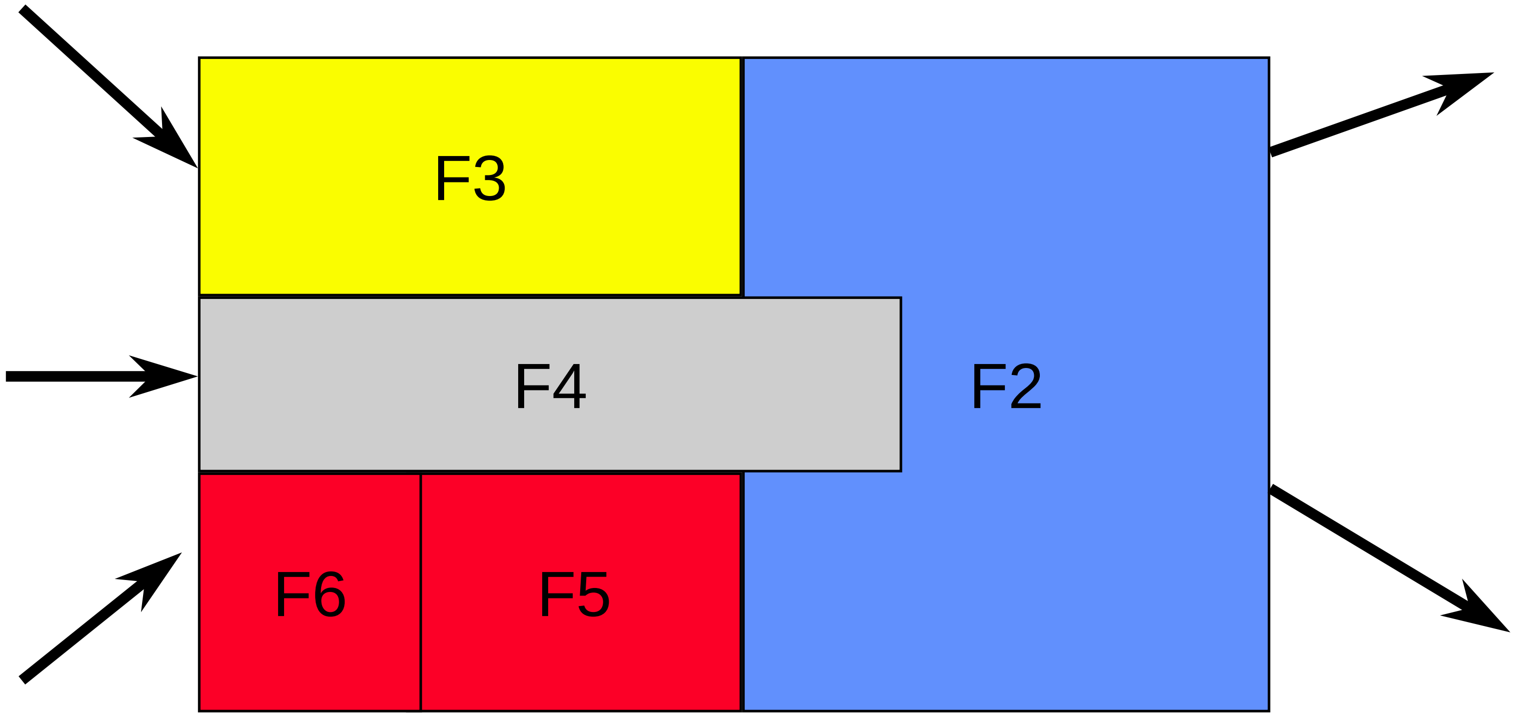
FastMAP (cont)

- *Wildcard Registers* allow *“don’t cares”* on address bits
  - same data can be written to several locations (SRAM and user registers) in one cycle
  - fast configuration of bit-slice type designs
  - **broadcast of data** to registers without tying up valuable routing resources.

Partial Run-time Reconfiguration

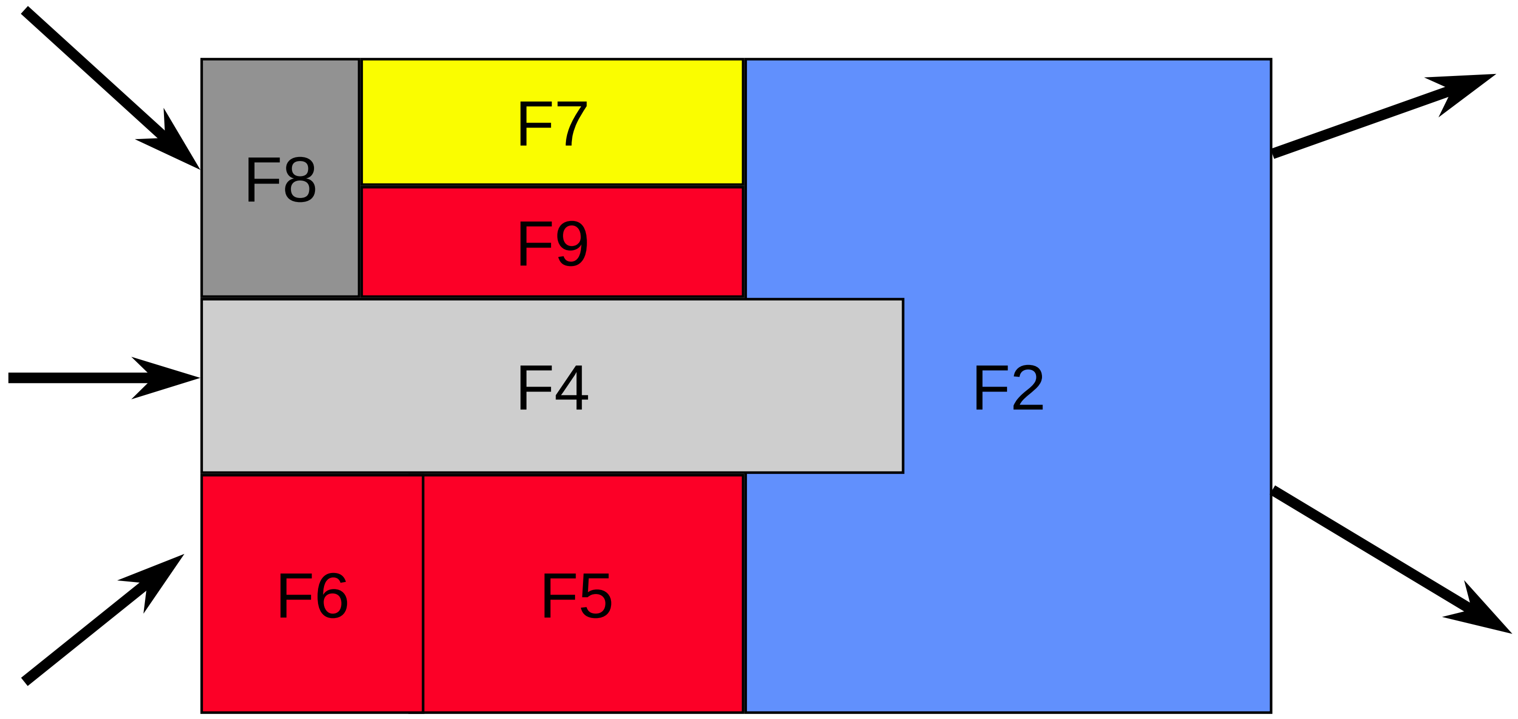
- Extend hardware to a *larger* (virtual) capacity through rapid reconfiguration
- Derive time-varying structures that are smaller and faster than the **ASIC** counterpart
- Make **more transistors** participate in a given computation

Partial Run-time Reconfiguration



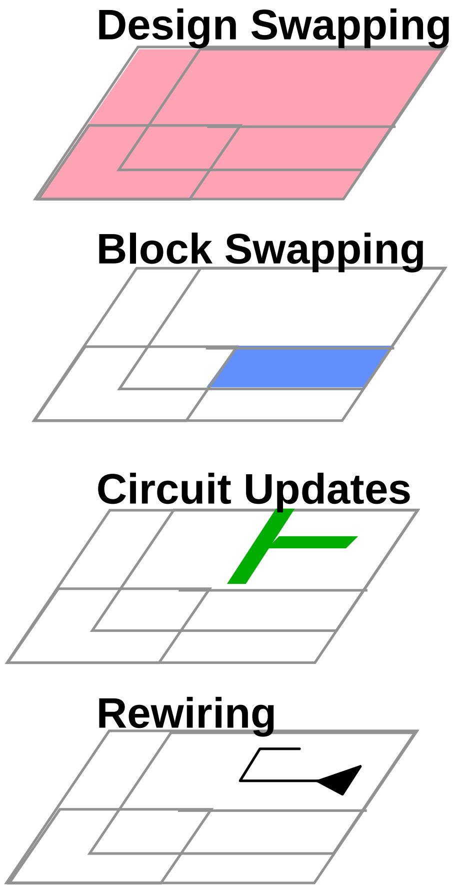
Time = 0

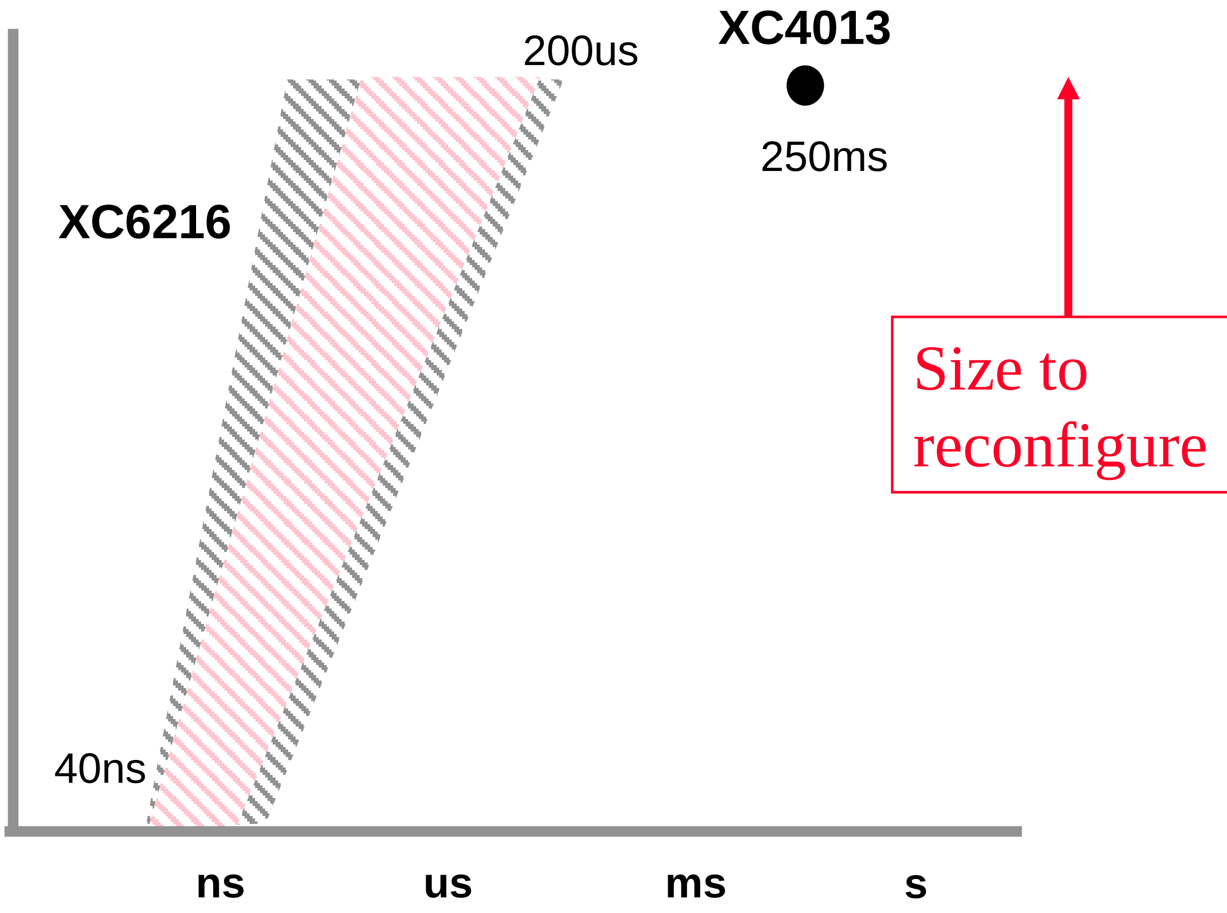
Partial Run-time Reconfiguration



Time = <a short time later>

Reconfiguration Speed vs Traditional Technologies





Time to reconfigure

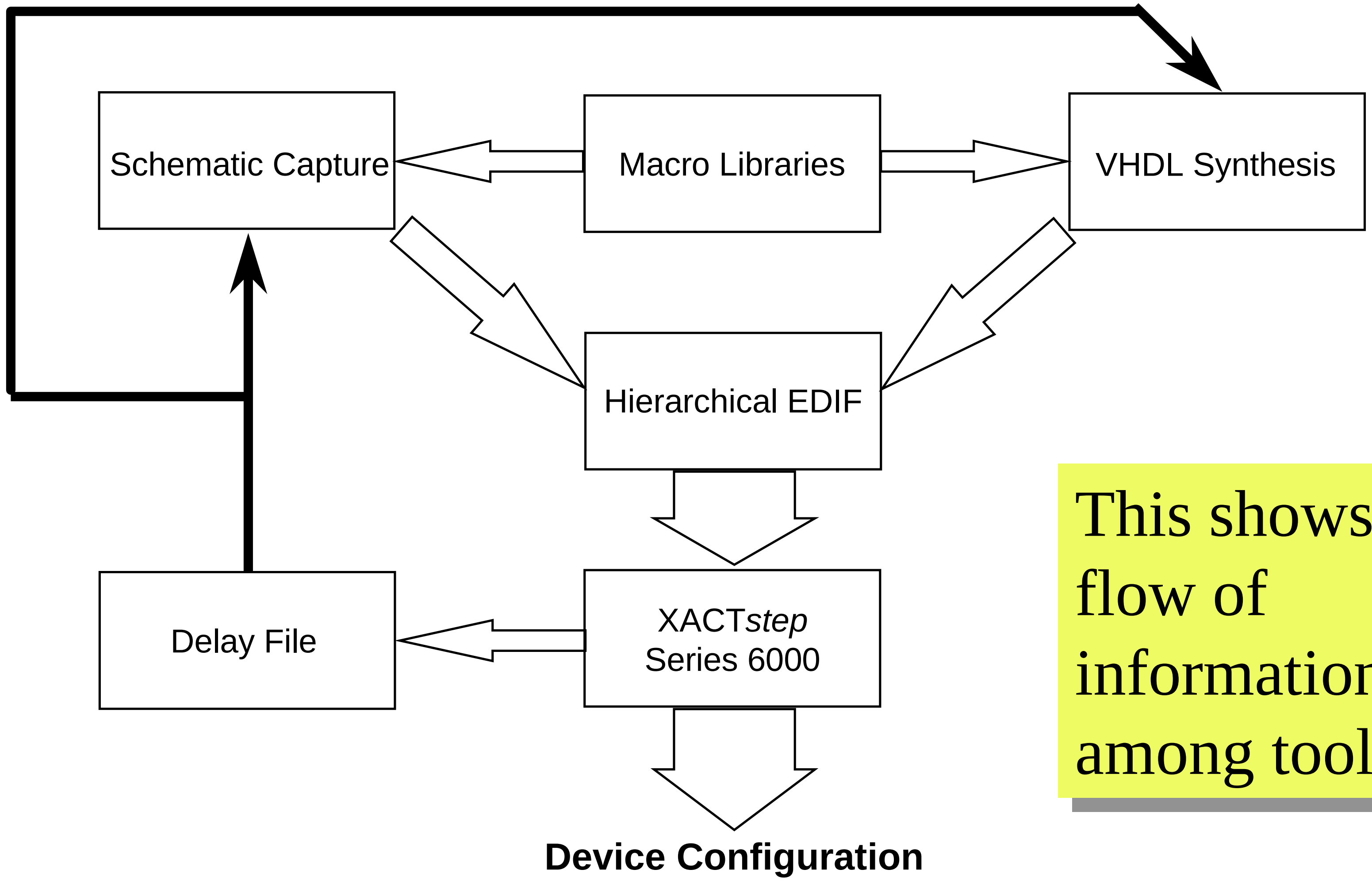
XC6200 Family Members

Device	XC6209	XC6216	XC6236	XC6264
Appr Gate Count	9k	16k	36k	64k
Number of Cells	2304	4096	9216	16384
Max No. of Registers	2304	4096	9216	16384
Number of IOBs	192	256	384	512
Cell Rows x Columns	48x48	64x64	96x96	128x128

Notes :

1. Gate counts are estimated average cases, based on LSI Logic figures - register rich designs can have a much higher equivalent gate count than stated above.
2. Not all IOBs are connected directly to pads - some pads are shared between IOBs.

Design Flows



This shows flow of information among tools

Library Support

- Primitive gates and functions (compatible with other Xilinx parts)
  - AND, OR, ADD, MULT, etc
- More complex macros also to be available
  - memory access
  - DSP functions (FIR, FFT, DCT)
  - JTAG, decoders, etc.

Applications

- Can be used as “regular” FPGA
  - serial interface allows for booting from PROM
- Intended to act as hardware accelerator for microprocessors
  - FastMAP allows for
    - direct microprocessor access to “internal” logic
    - fast reconfiguration of all or part of device

Applications (cont)

- “Context switching” and “virtual hardware” are realistic propositions
- Typical uses might include:
  - DSP,
  - image processing,
  - data paths,
  - etc.

Reconfigurable Processing

- “Custom computing” concept, building on
  - fast configuration
  - virtual hardware
- PCI based development system to be made available
  - can be used as a custom computer in its own right, or
  - as an aid to system development for customers’ designs

XC6000 Software:

- XACT6000 Software From Xilinx.
- Trianus/Hades Design Entry Software for the XC6200
- Velab: Free VHDL Elaborator for the XC6200.
- XC6200 Inspector.

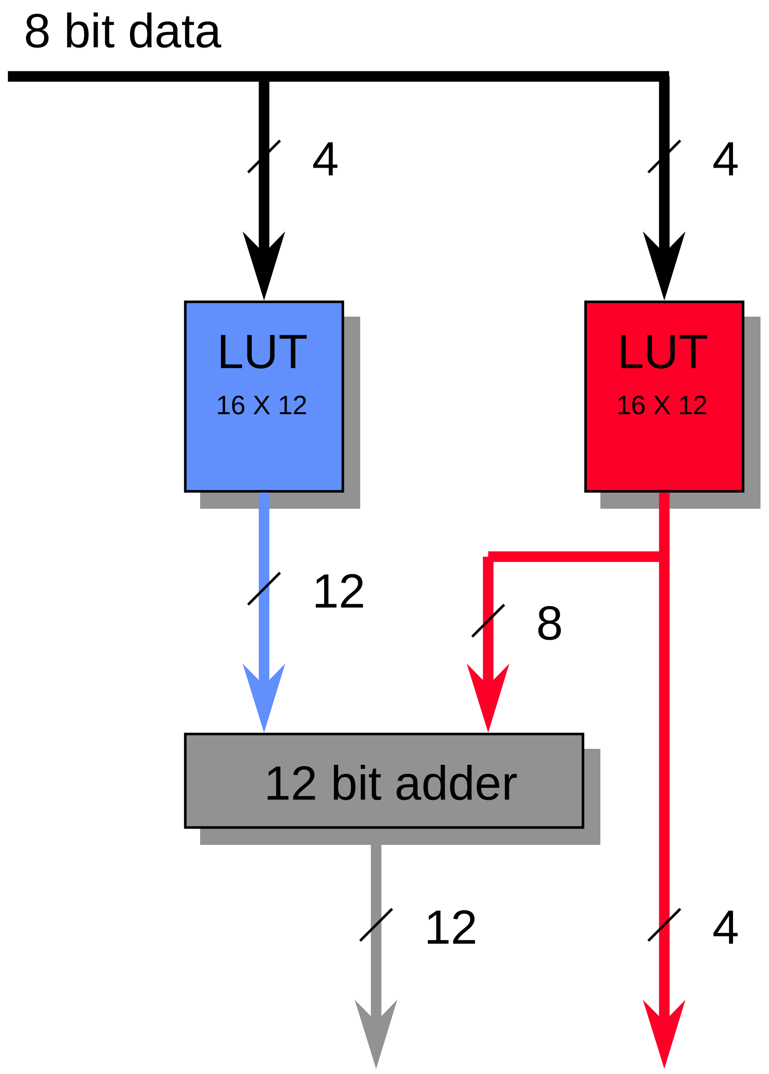
EXAMPLE

A Multiplier for the XC6200

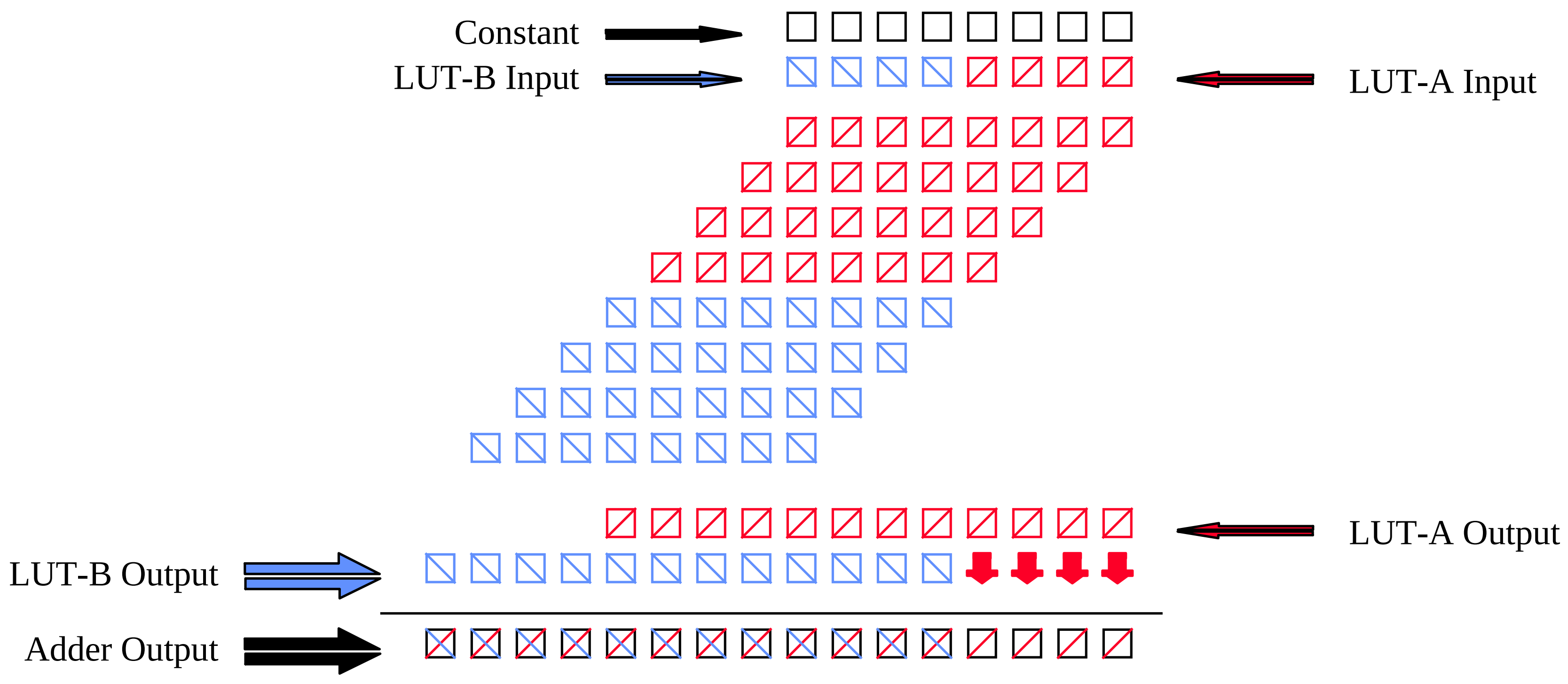
A Multiplier for the XC6200

- Structure
- Math
- Building Lookup Tables
- Area Optimization
- Mapping into an XC6200
- Changing Coefficients
- Performance
- Summary

Distributed Arithmetic (Multiplier)

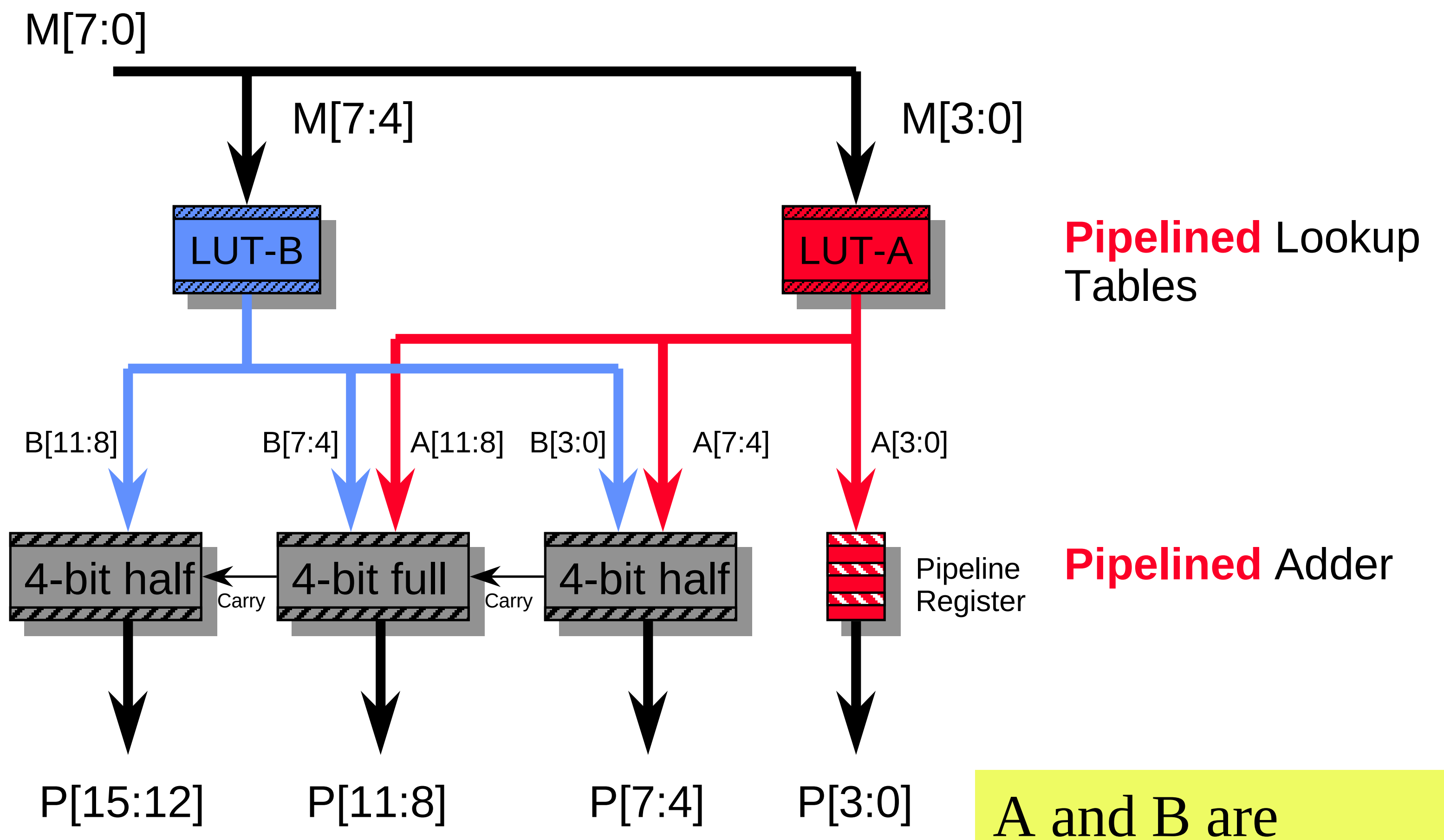


Math Class



Colors related to previous slide

Architecture of the Multiplier



A and B are arguments, pipelined

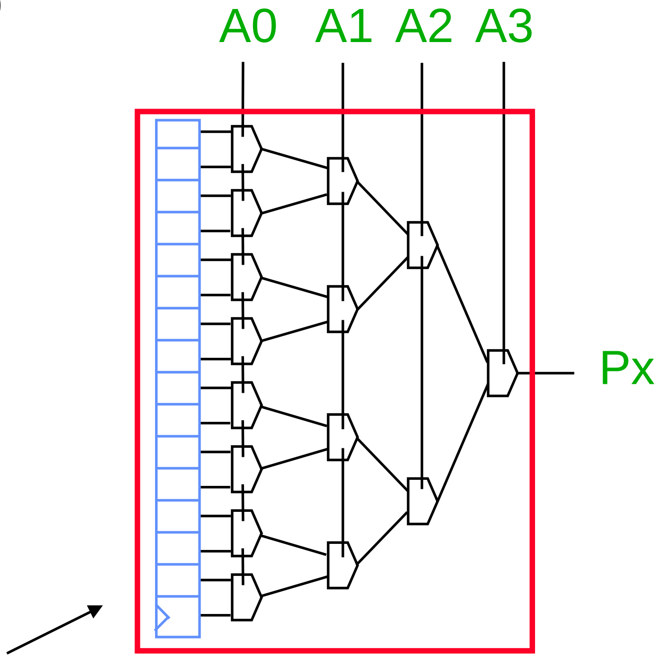
LUTs by Muxing

All logic realized as trees

- Lookup Table contains all pre-calculated partial products.
- Use a Truth Table to determine Mux inputs.

All possible products for multiplying by 0011 (3)

A3	A2	A1	A0	P7	P6	P5	P4	P3	P2	P1	P0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	1	1
0	0	1	0	0	0	0	0	0	1	1	0
0	0	1	1	0	0	0	0	1	0	0	1
0	1	0	0	0	0	0	0	1	1	1	0
0	1	0	1	0	0	0	0	1	1	1	1
0	1	1	0	0	0	0	1	0	0	1	0
0	1	1	1	0	0	0	1	0	1	0	1
1	0	0	0	0	0	0	1	1	0	0	0
1	0	0	1	0	0	0	1	1	0	1	1
1	0	1	0	0	0	0	1	1	1	1	0
1	0	1	1	0	0	1	0	0	0	0	1
1	1	0	0	0	0	1	0	0	1	0	0
1	1	0	1	0	0	1	0	0	1	1	1
1	1	1	0	0	0	1	0	1	0	0	0
1	1	1	1	0	0	1	0	1	1	1	1

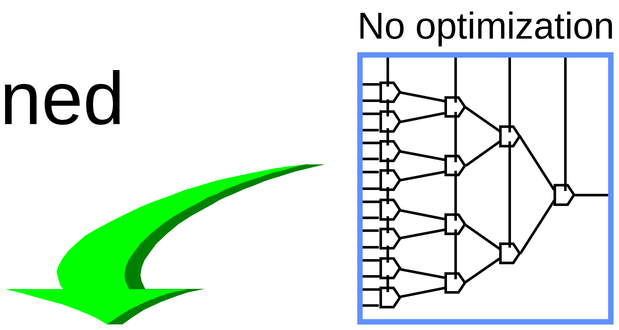


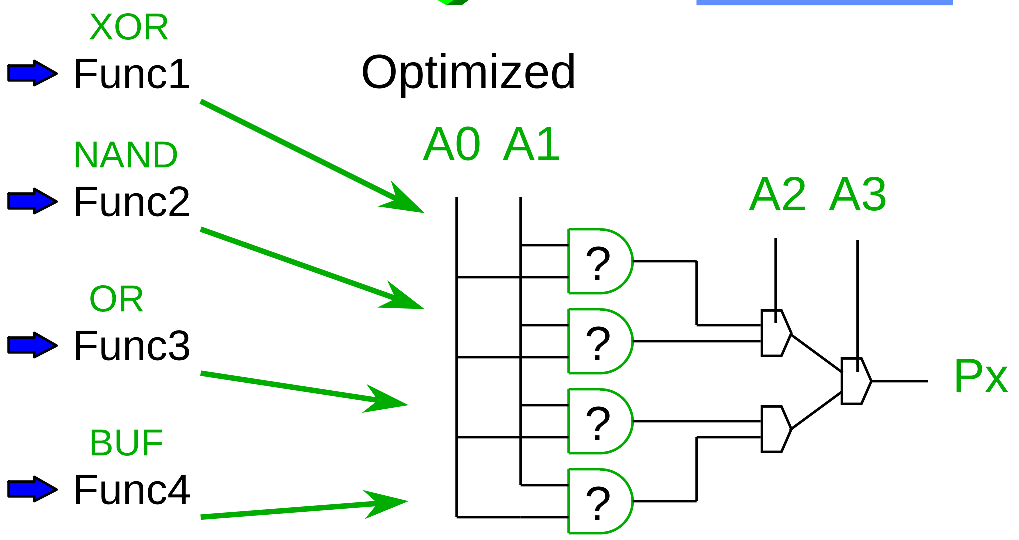
Each Pi realized as this

Optimizing the Lookup

- Two mux levels can be collapsed into a single gate.
- The function can be determined with a truth table.

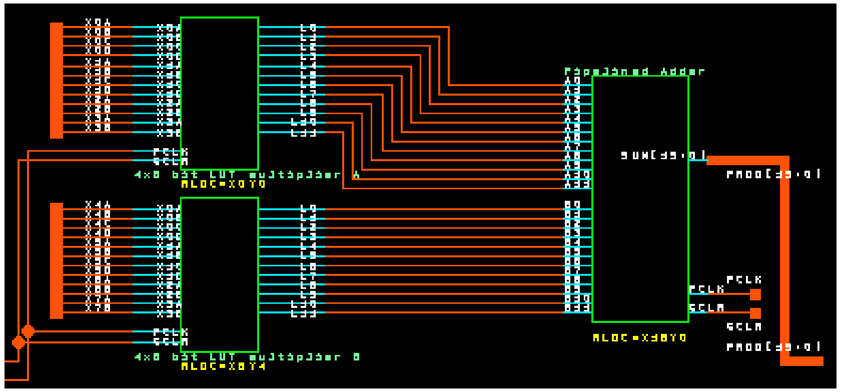
A3	A2	A1	A0	P7	P6	P5	P4	P3	P2	P1	P0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	1	1
0	0	1	0	0	0	0	0	0	1	1	0
0	0	1	1	0	0	0	0	1	0	0	1
0	1	0	0	0	0	0	0	1	1	1	0
0	1	0	1	0	0	0	0	1	1	1	1
0	1	1	0	0	0	0	1	0	0	1	0
0	1	1	1	0	0	0	1	0	1	0	1
1	0	0	0	0	0	0	1	1	0	0	0
1	0	0	1	0	0	0	1	1	0	1	1
1	0	1	0	0	0	0	1	1	1	1	0
1	0	1	1	0	0	1	0	0	0	0	1
1	1	0	0	0	0	1	0	0	1	0	0
1	1	0	1	0	0	1	0	0	1	1	1
1	1	1	0	0	0	1	0	1	0	0	0
1	1	1	1	0	0	1	0	1	1	1	1





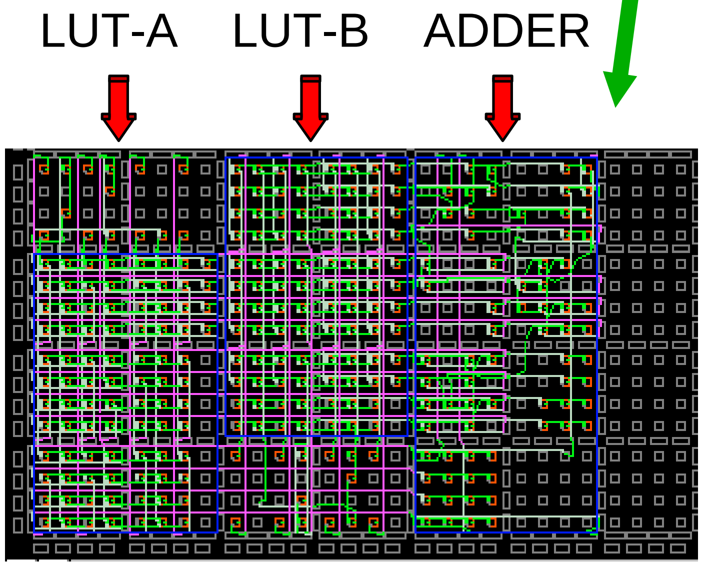
Logic synthesis done by the CAD tool

Multiplier Schematic

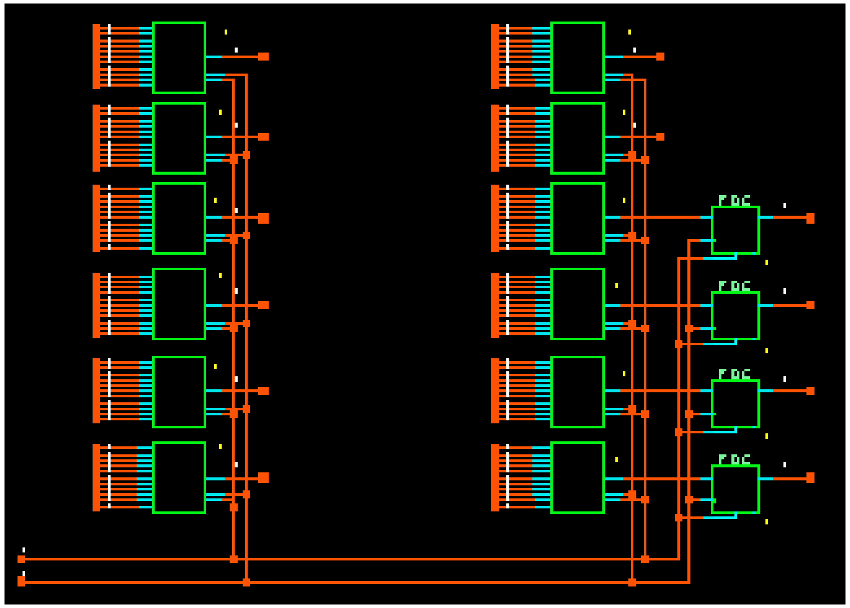


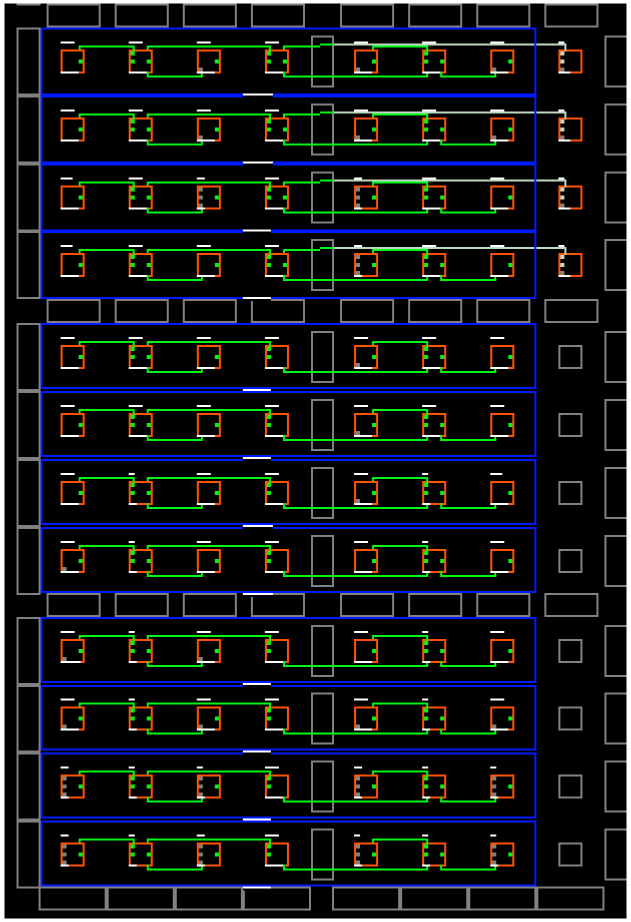
- Schematic resembles the block diagram.
  - Two LUTs sourcing adder.

- The corresponding view in the layout editor.
- The LUTs are offset to line up bits for adder.
- Pipeline registers are cheap.
  - XC6216 has 4096 Flip Flops



A Closer Look at a Lookup



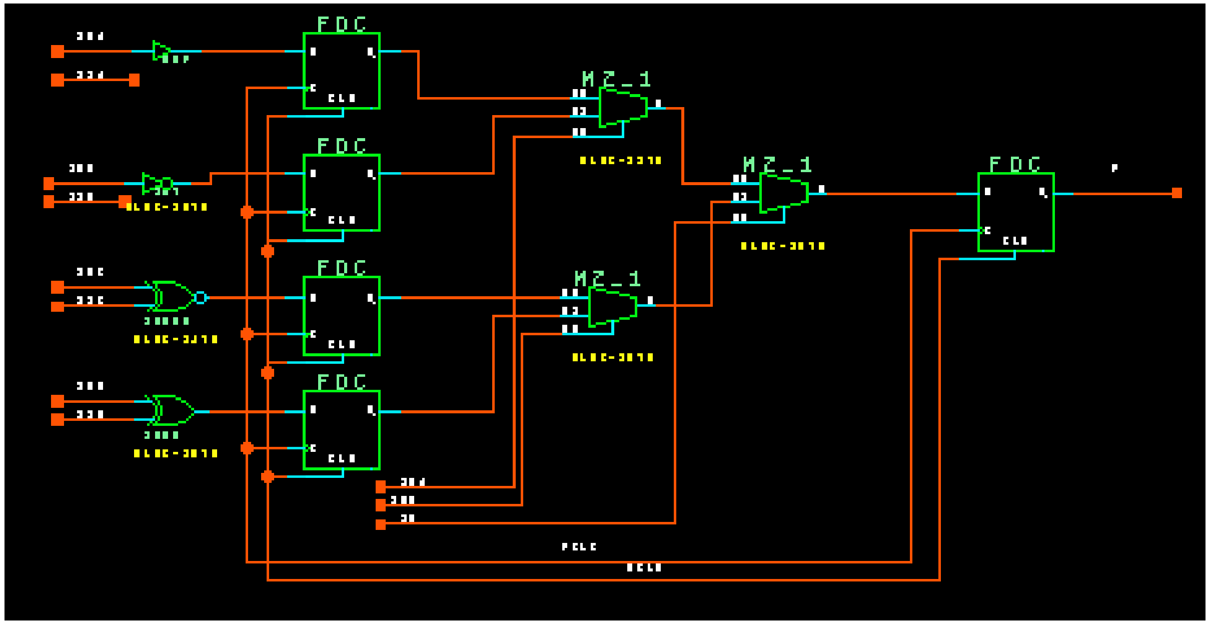


- Each 12-bit LUT is built from 12 one bit LUTs.
- LUTs get stacked vertically.

Determining Coefficients

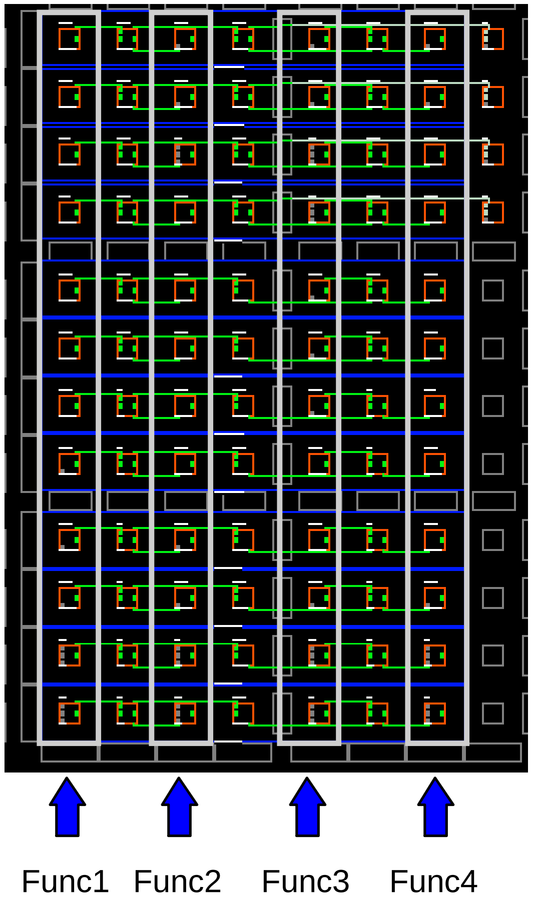
- Schematic for a single 4-input LUT.
- Functions can be determined from the Truth Table.

A3	A2	A1	A0	P
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0



Changing Coefficients

- **Functionality of a cell is contained in one byte.**
  - 32-bit access can change the function of 4 cells *per write cycle.*
- 96 cells need writing, or **24 write cycles.** (worst case)
  - 1.45µs assuming 33MHz



4*24=96

Summary on Multiplier Design

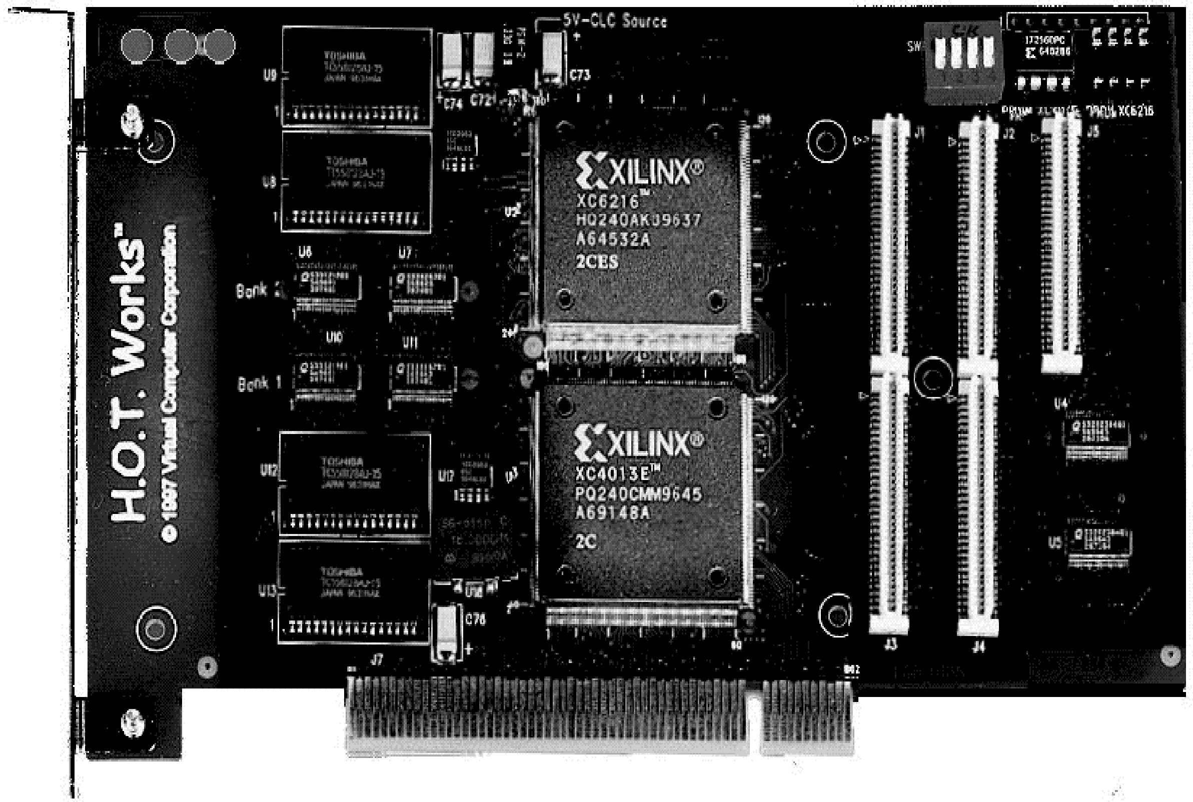
- 8x8 constant coefficient multiplier
- Pipelined - 75+ MHz performance
- Small grain architecture - High degree of LUT optimization
- Coefficients easily changed - Fast reconfiguration times.
- High Performance/Dollar

Complete Hardware Systems

Development Board

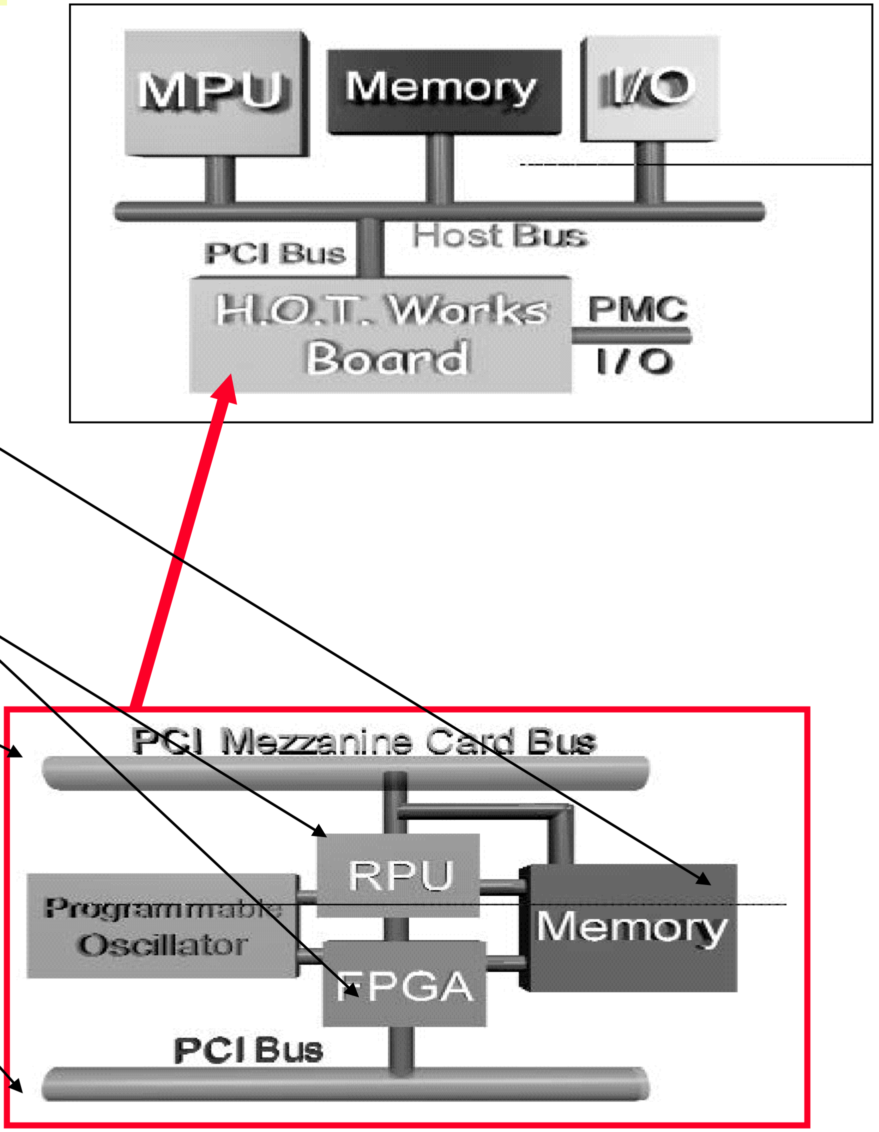
EXAMPLE : H.O.T. Works

- Development system based on the Xilinx XC6200-series RPU
- Includes:
  - H.O.T. Works Configurable Computer Board
  - H.O.T. Works **Development System Software**

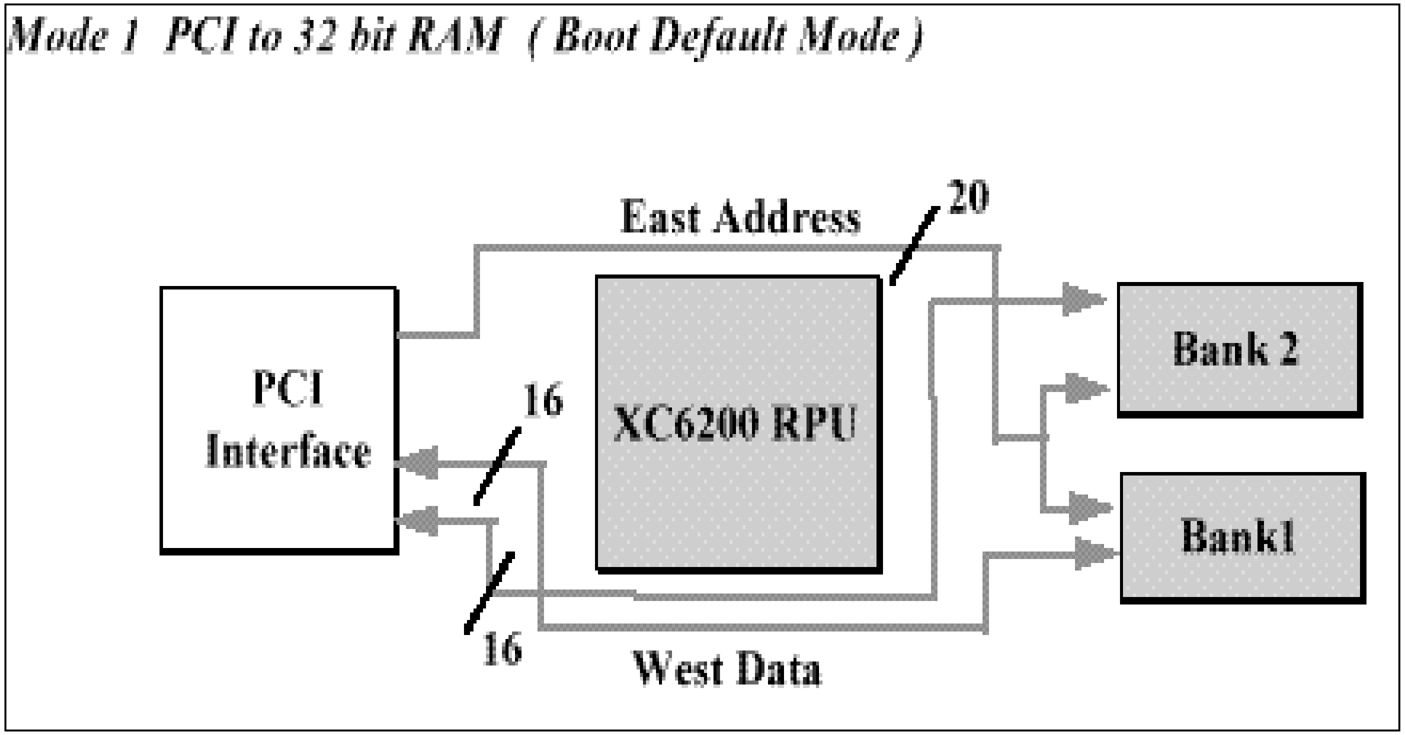


H.O.T. Works Board

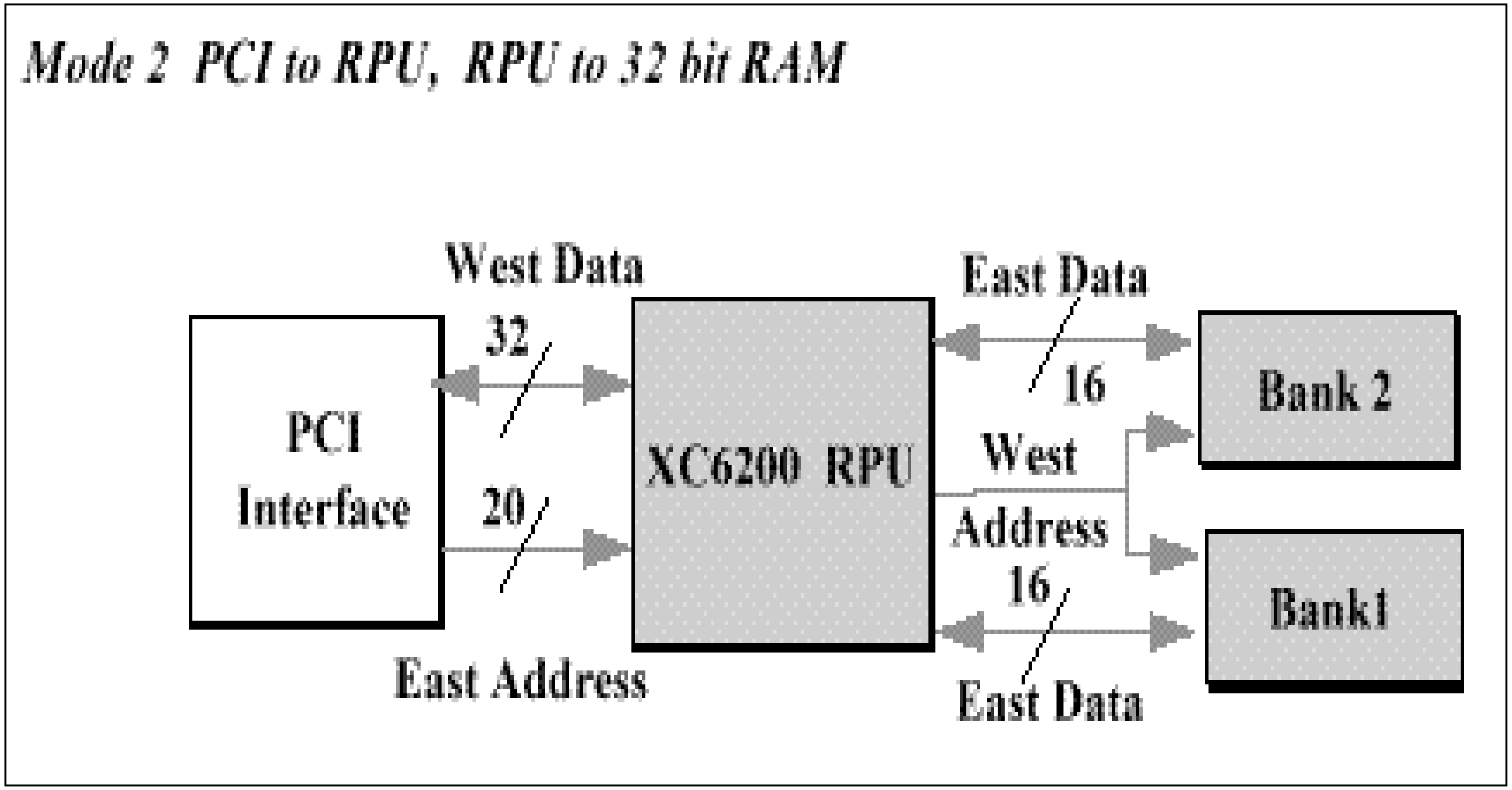
- Interfaces with a host system (Windows95-based PC) on PCI bus
  - 2MB SRAM (memory)
  - XC6200 (RPU)
  - PCI controller on XC4000 (FPGA)
  - Expansion through Mezzanine connector



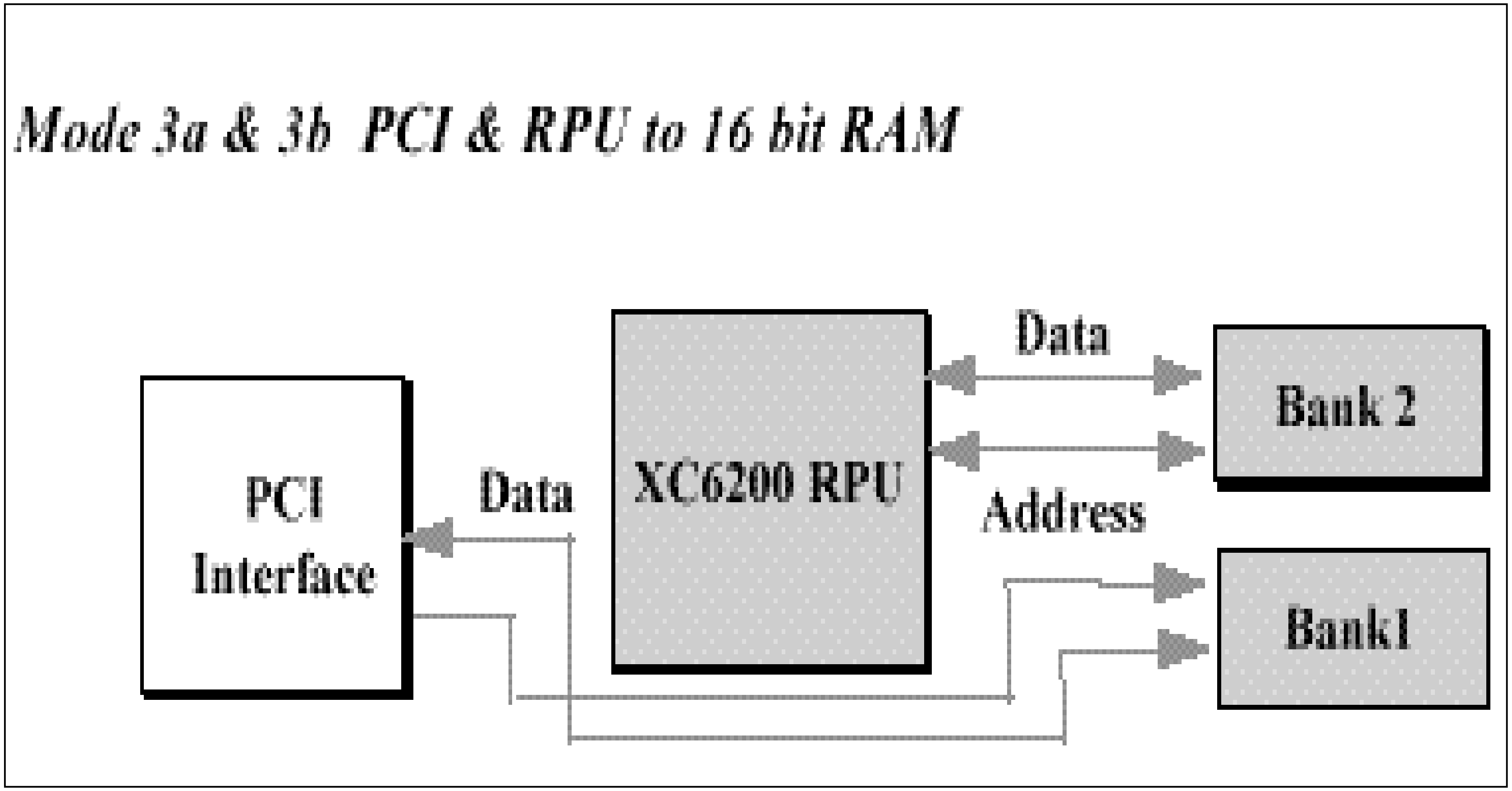
Memory Access Modes



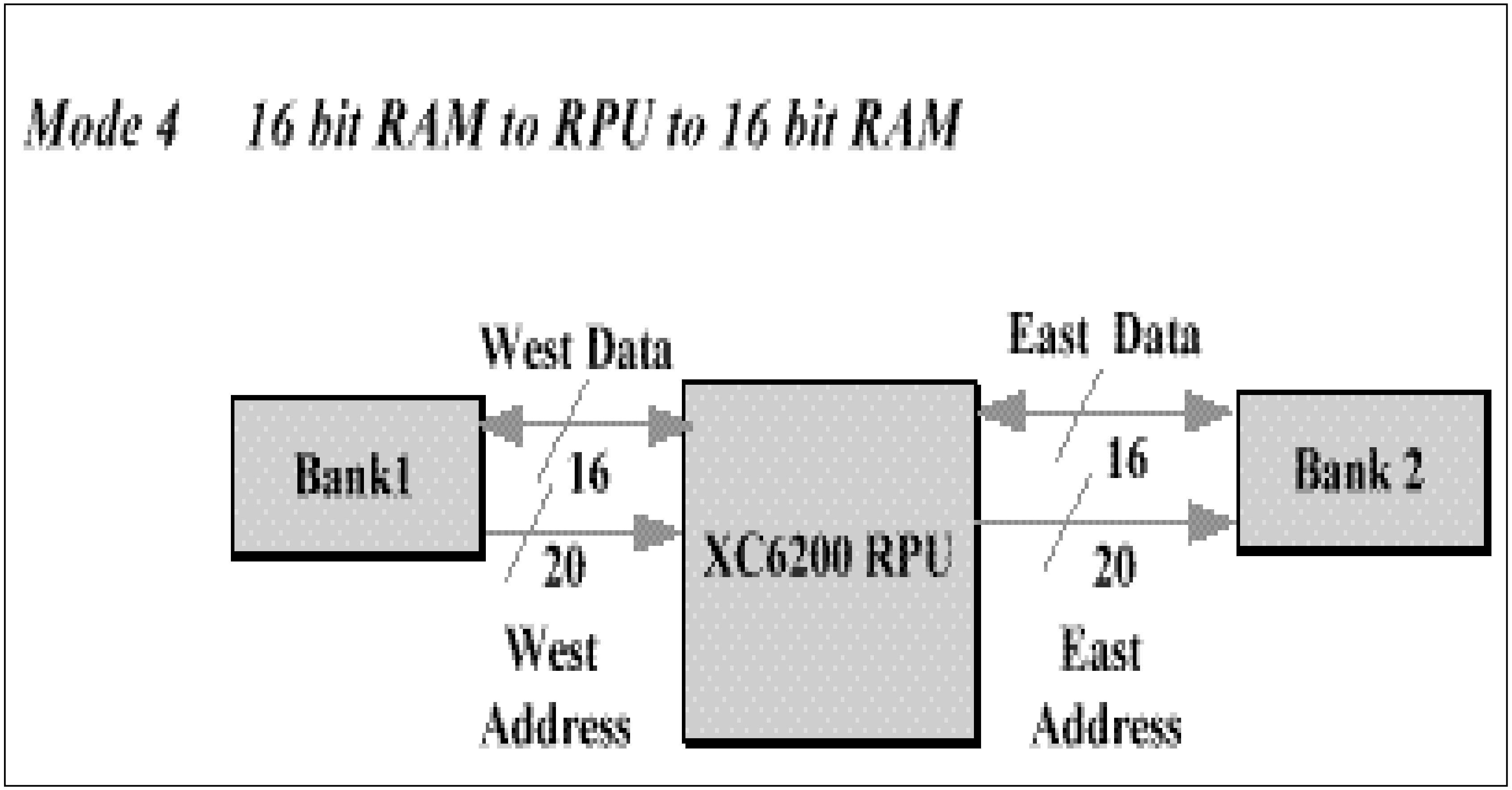
Memory Access Modes



Memory Access Modes



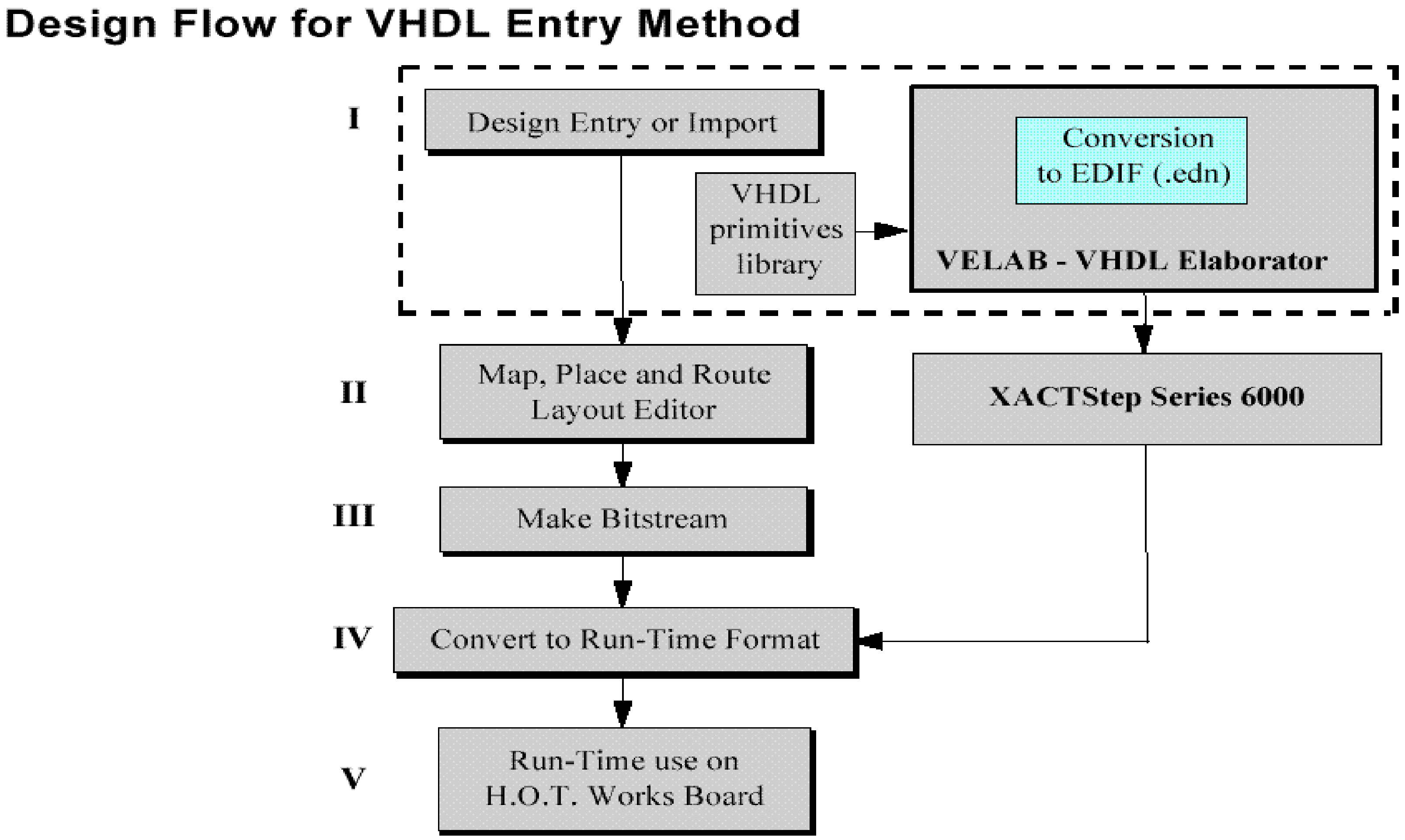
Memory Access Modes



H.O.T. Works Software

- Xilinx XACT*Step*
  - Map, Place and Router for XC6200
- Velab
  - Structural VHDL elaborator
- WebScope
  - Java-based debug tool
- H.O.T. Works Development System
  - C++-based API for **board interfacing**

Design Flow


Design Flow for VHDL Entry Method
I
Design Entry or Import
VHDL primitives library
Conversion to EDIF (.edn)
VELAB - VHDL Elaborator
II
Map, Place and Route Layout Editor
XACTStep Series 6000
III
Make Bitstream
IV
Convert to Run-Time Format
V
Run-Time use on H.O.T. Works Board

Run-Time Programming

- C++ support software is provided for low-level board interface and device configuration
- Digital design is downloaded to the board at execution time
- User-level routines must be written to conduct data input/output and control

Conclusions

- Xilinx XC6200 provides a fast and inexpensive method to obtain great speedups in certain classes of algorithms
- There exist tools that provide a useable development platform to go from structural VHDL to digital design, and a programmable run-time interface in C++.

Sources

Mark L. Chang
<mchang@ece.nwu.edu>

Stephen Churcher

Ahmad Alsolaim