

ADVANCES IN RECONFIGURABLE COMPUTING

Dinesh Bhatia
The University of Texas at Dallas

HELPFUL KNOWLEDGE

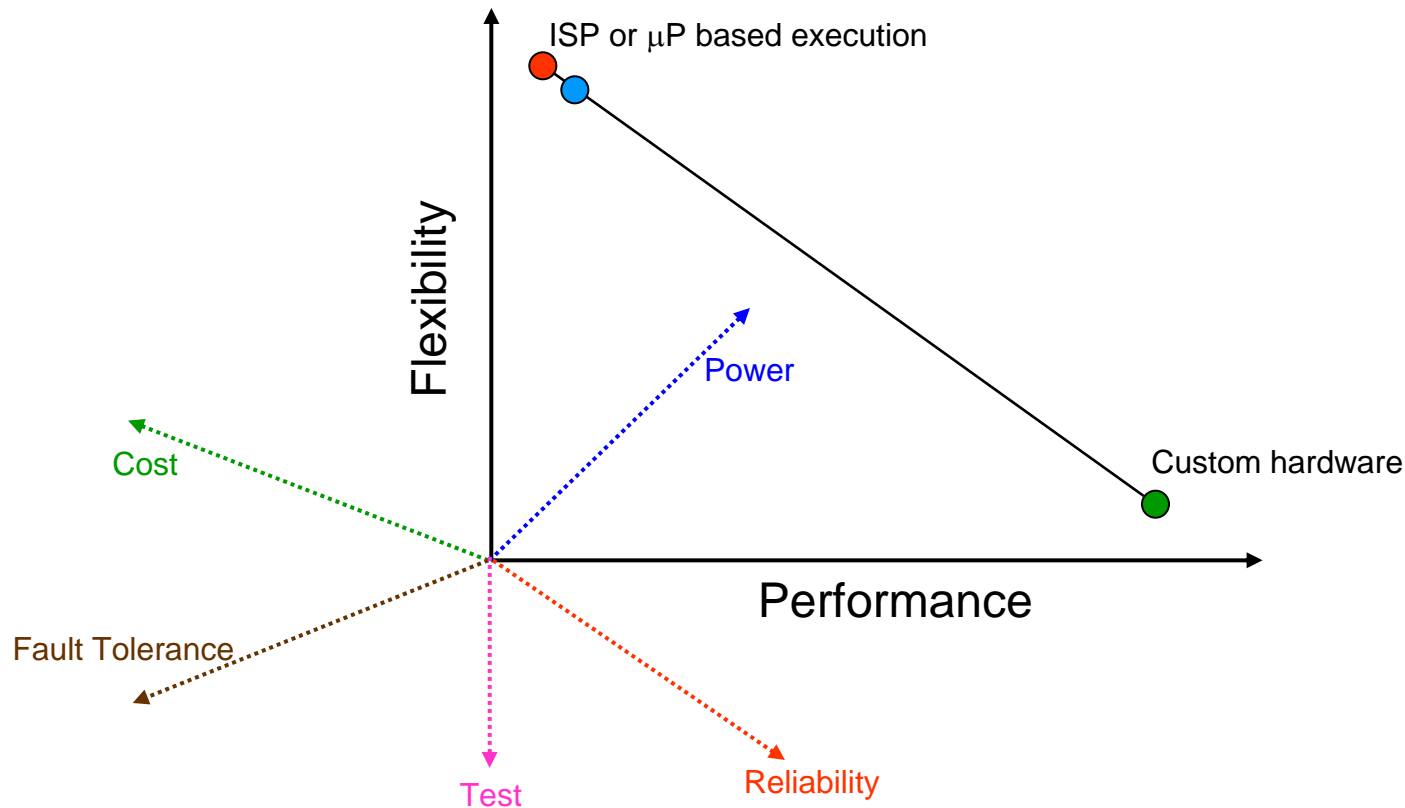
- ◆ Design
 - ◆ VHDL based modeling and design simulation and synthesis
 - ◆ Physical design using FPGAs
- ◆ Electrical and Computer Engineering
 - ◆ Must possess advanced logic design skills.
- ◆ Computer Science
 - ◆ Decent handle on programming using C/C++
 - ◆ Understanding on CS fundamentals
- ◆ Innovation and imagination

MAIN CONFERENCES

- ◆ ACM International Symposium on Field Programmable Gate Arrays (FPGA)
- ◆ International Conference on Field Programmable Logic (FPL), Springer Verlag
- ◆ IEEE International Conference on FPGAs in Custom Computing Machines (FCCM)
- ◆ Some recent proceedings on IEEE/ACM Design Automation Conference.

WHAT IS THIS TALK ABOUT?

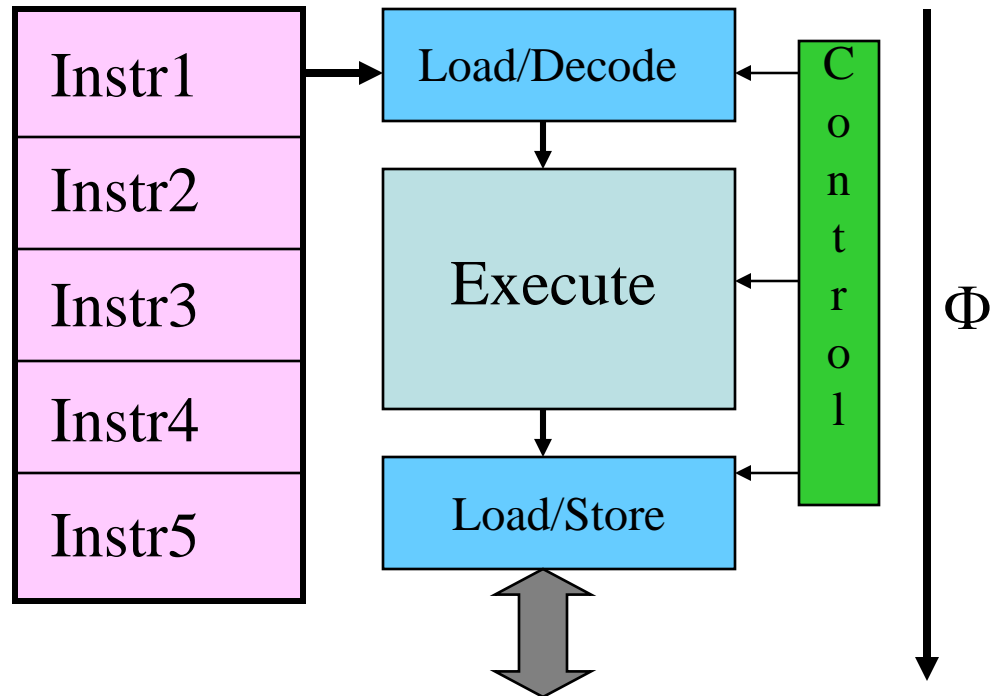
- ◆ It is about Reconfigurable Computing
 - ◆ The ability to use functions that have been created by customization of spatially connected processing elements after the hardware (silicon) has been manufactured.



TEMPORAL EXECUTION ON ISP

- ◆ Load
- ◆ Control
 - ◆ Decode
- ◆ Execute
- ◆ Load/Store

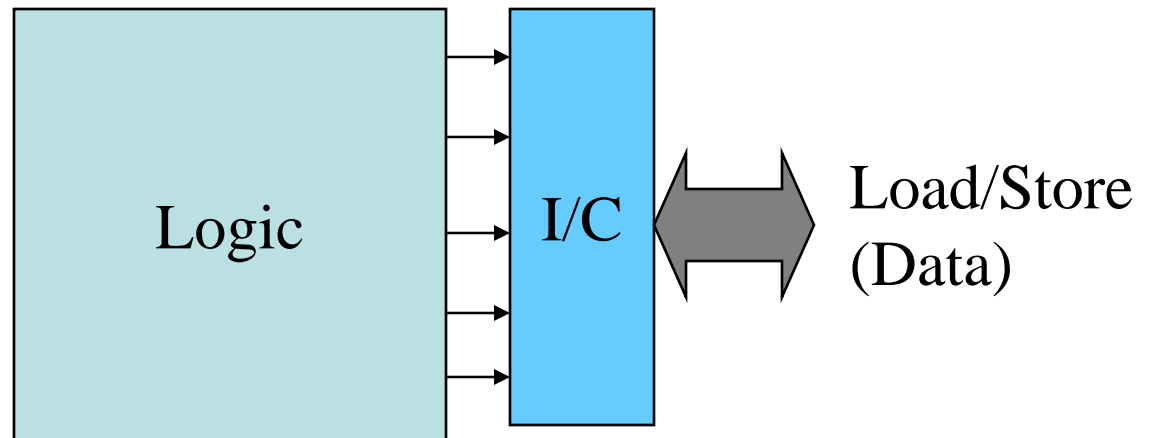
- ◆ Enabled language and compiler technology.
- ◆ Supports upward compatibility
 - ◆ *sort of forces a contract*
- ◆ Re-use of static hardware resources.



SPATIAL EXECUTION ON HARDWARE

- ◆ No Control.
- ◆ Computing is through communication on interconnected logic.

Custom
Expensive
High Performance



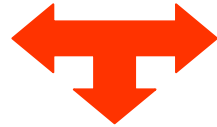
A COMPARISON

ISP

- ◆ High flexibility
- ◆ Mostly high performance
- ◆ Inexpensive solution

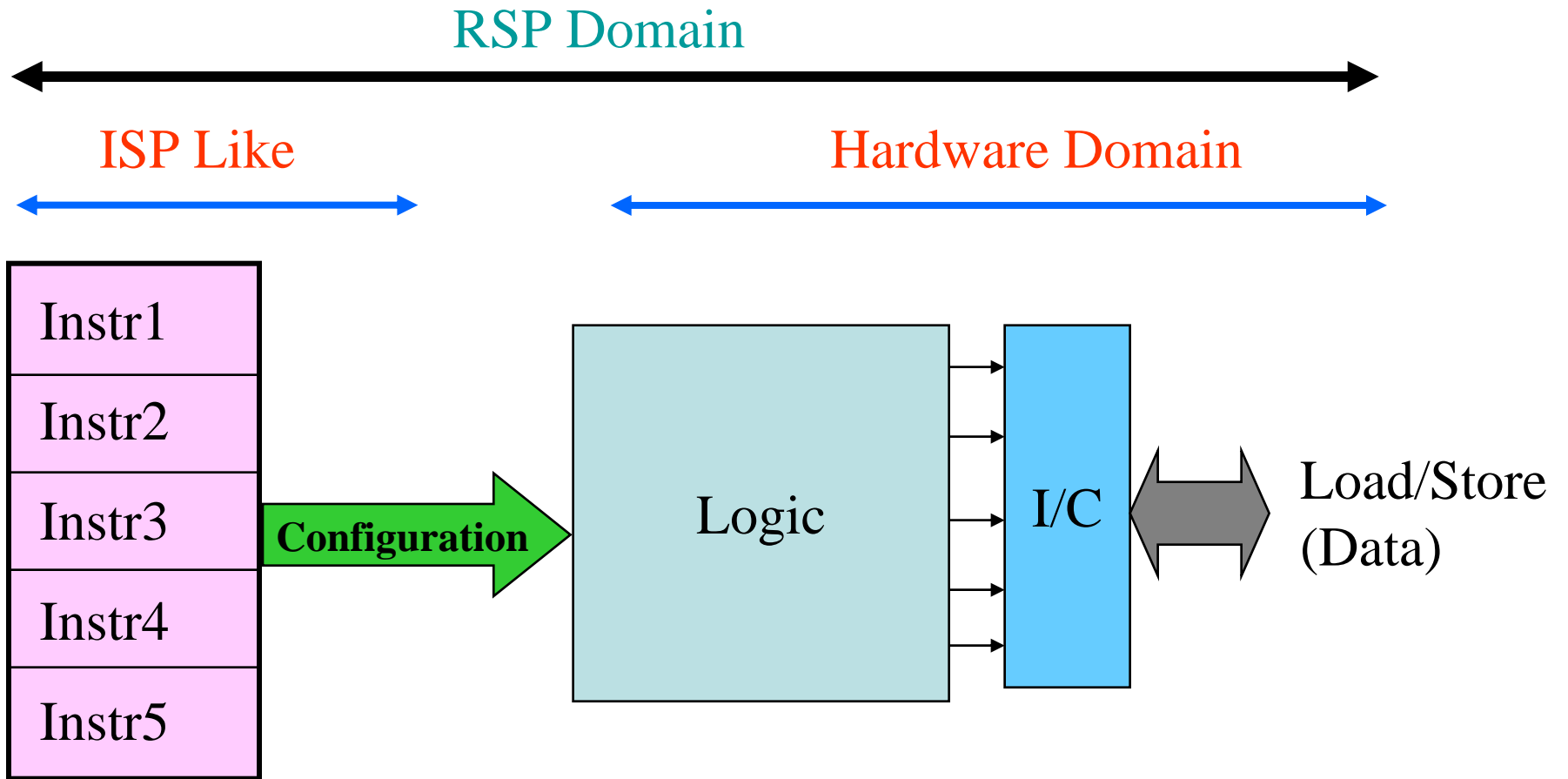
Hardware

- ◆ Very rigid
- ◆ Extremely high performance
- ◆ Very expensive



Reconfigurable Computers

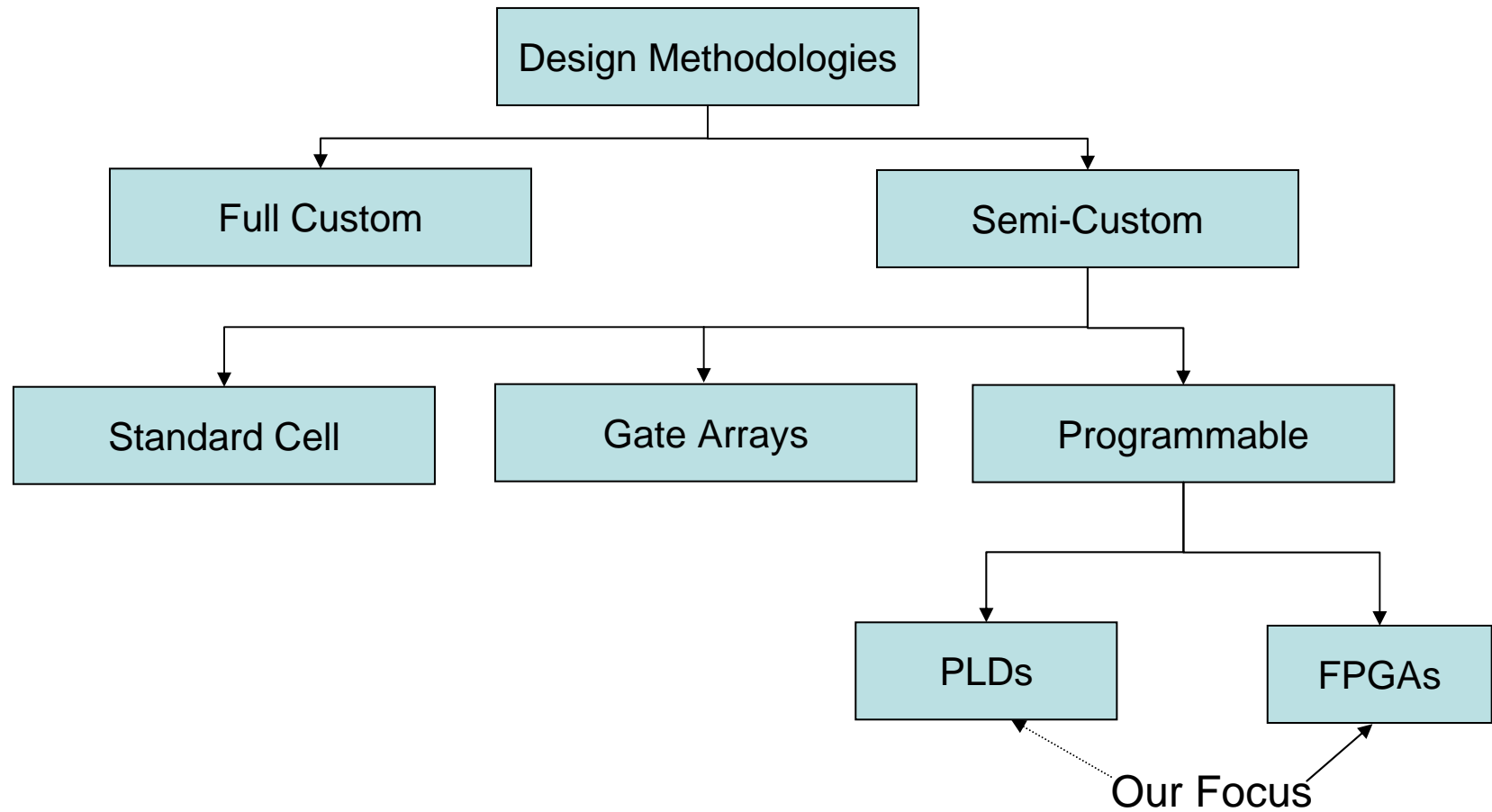
RECONFIGURABLE COMPUTERS



ISP BASED COMPUTING: CONS

- ◆ Performance
 - ◆ temporal execution not the best idea.
 - ◆ Mold application to hardware
 - ◆ *actually it should be mold hardware to application.*
- ◆ Financial
 - ◆ design time and development cost
 - ◆ design challenges
 - ◆ yield and process
 - ◆ complexity and verification
 - ◆ test cost

DESIGN METHODOLOGIES

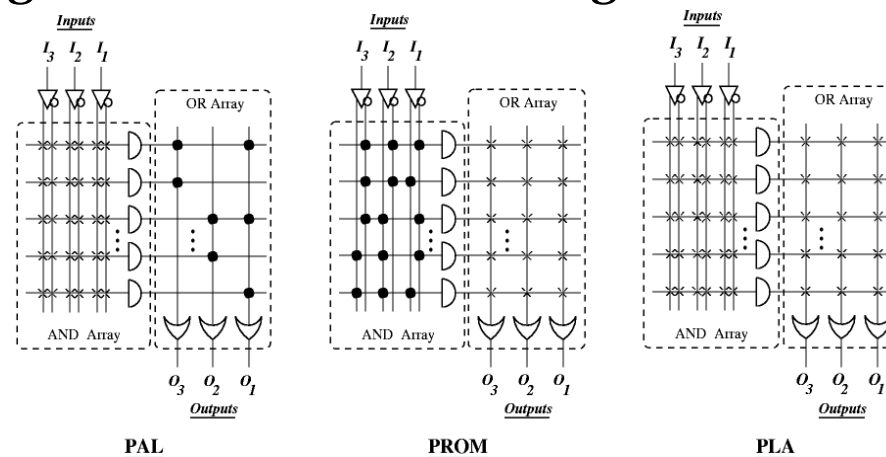


PROGRAMMABLE DEVICES

- ◆ Pre-fabricated silicon.
- ◆ Logic implemented by programming the interconnects and basic cells.
- ◆ Very fast *turnaround time*.
- ◆ Limited *design flexibility*.
- ◆ Low development *time* and *cost*.

PROGRAMMABLE LOGIC DEVICE (PLD)

- ◆ A programmable AND array followed by a fixed fanin OR gates that are followed by flip-flops.
 - ◆ **PLDs with AND/OR logic structure:**
 - ◆ PAL: Programmable AND array, Fixed OR array.
 - ◆ ROM: Fixed AND array, Programmable OR array.
 - ◆ PLA: Programmable AND array, Programmable OR array.
- ◆ Product lines can be any input combination and flip-flops can be fed back to input.
- ◆ Programmable element can be fuse or a transistor.
- ◆ Densities range from 1,000 to 10,000 gates.



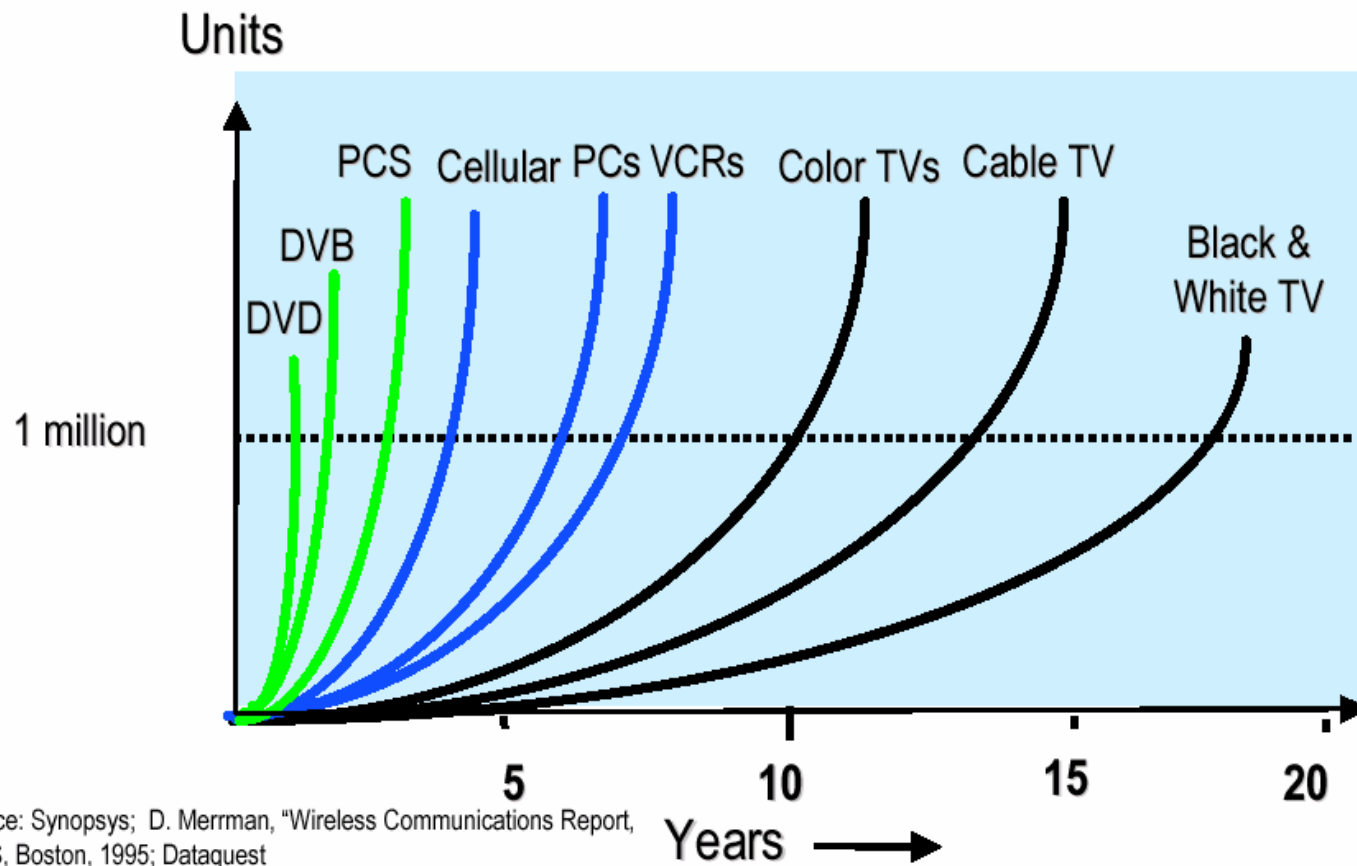
FIELD-PROGRAMMABLE GATE ARRAYS (FPGAs)

- ◆ A successful marriage of PLDs and MPGAs
- ◆ Densities range from 2K to 10,000K+ gates.
- ◆ Array of *Logic blocks* and *programmable interconnect*.
- ◆ Logic block made of universal gates, multiplexers, RAMs, NAND gates, transistors etc.
- ◆ Programmable element made of
 - ◆ **SRAM**,
 - ◆ SRAM bits can be programmed many times.
 - ◆ High area overhead – five transistor pass transistor based SRAM cell.
 - ◆ **EEPROM, or**
 - ◆ Frequently used for PALs and EPLDs
 - ◆ Low area overhead – only one transistor per programming point.
 - ◆ **Antifuse**.
 - ◆ One time programmable, high performance
 - ◆ Low area overhead.

WHY FPGAs

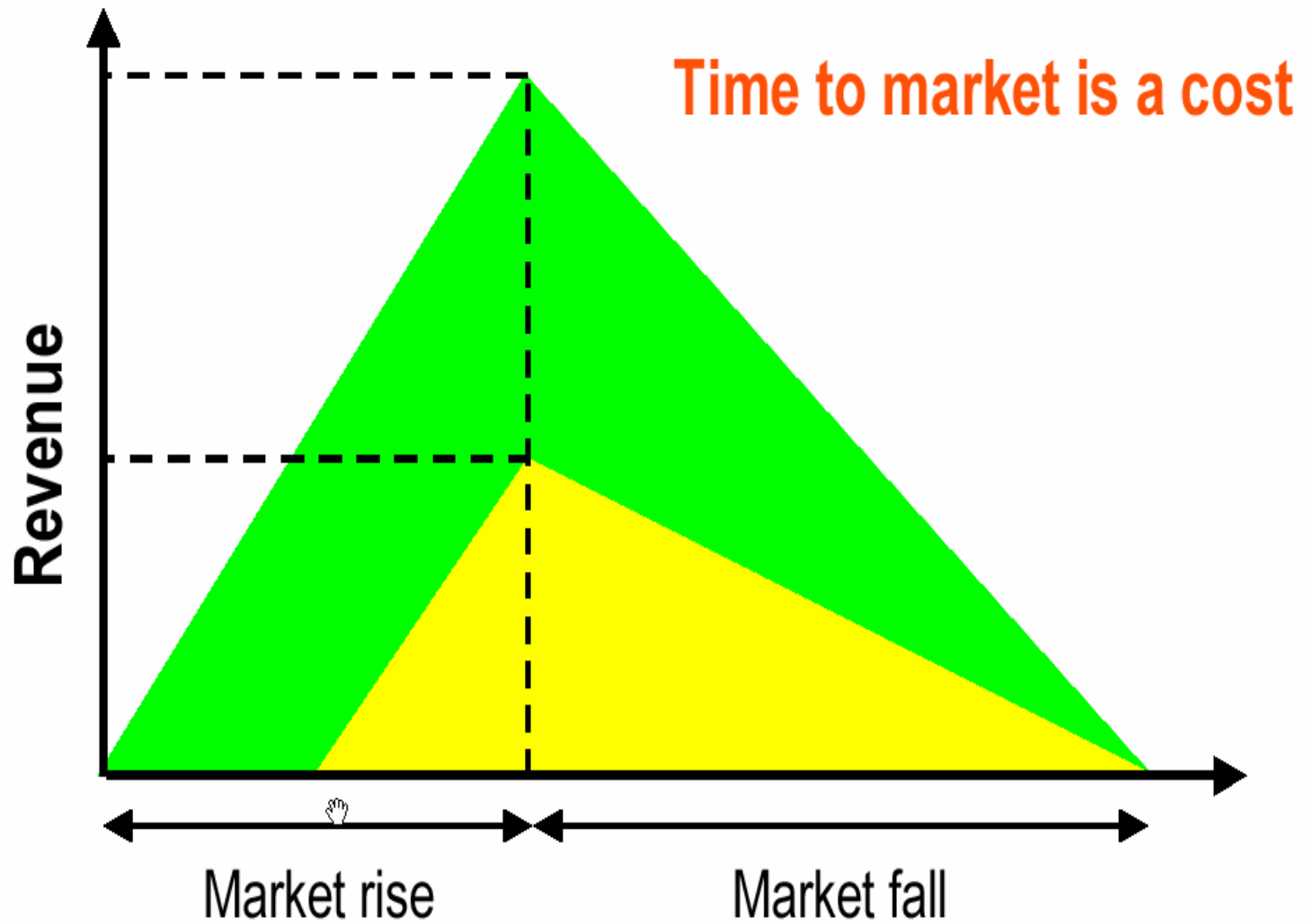
- ◆ Glue logic (replace SSI and MSI parts)
 - ◆ This is where it all started for FPGAs.
- ◆ Inventory burdens
- ◆ Rapid turnaround and Low risk
- ◆ Emulation
- ◆ Custom and super computing
- ◆ Dynamically changing hardware
- ◆ And, in future...
 - ◆ Once we stop our fascination with making things faster and start thinking about making them smarter, we will need reconfigurability.

CHANGING DYNAMICS

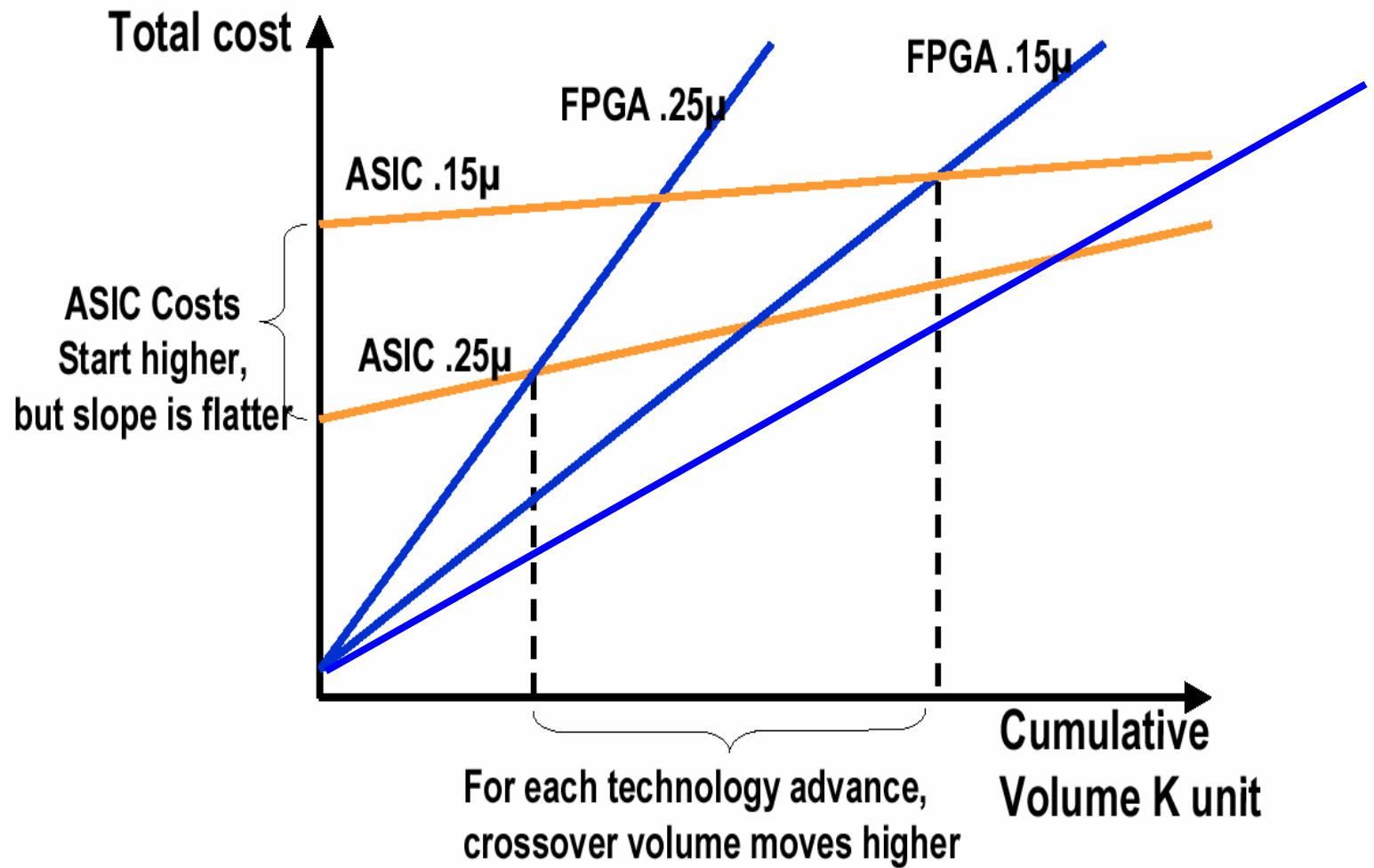


Source: Synopsys; D. Merrman, "Wireless Communications Report," BIS, Boston, 1995; Dataquest

TIME TO MARKET IS CRITICAL

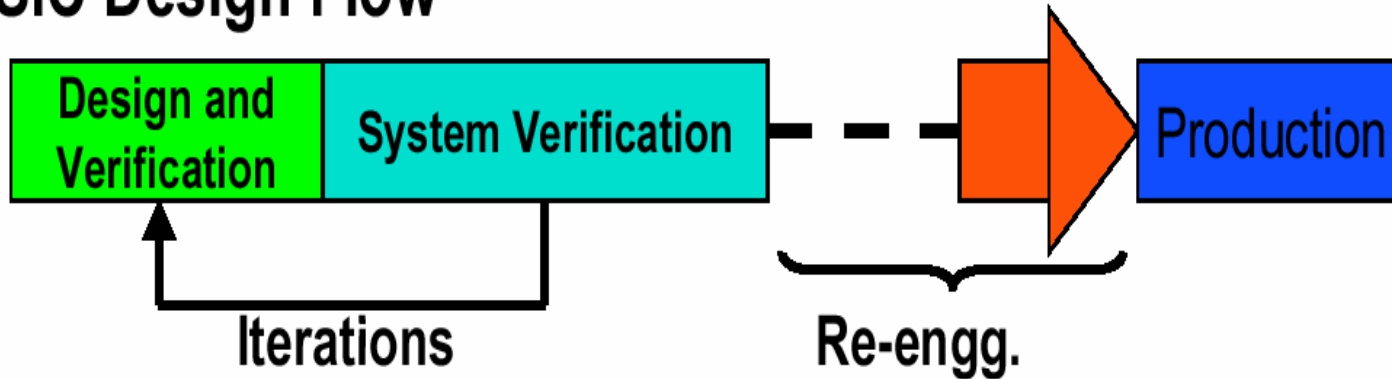


COST STRUCTURING

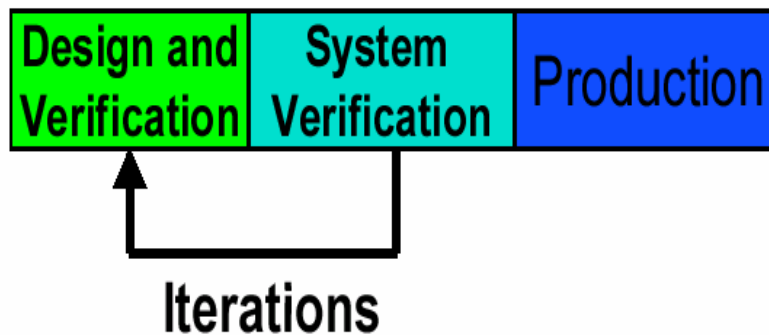


FPGAs VERSUS ASIC

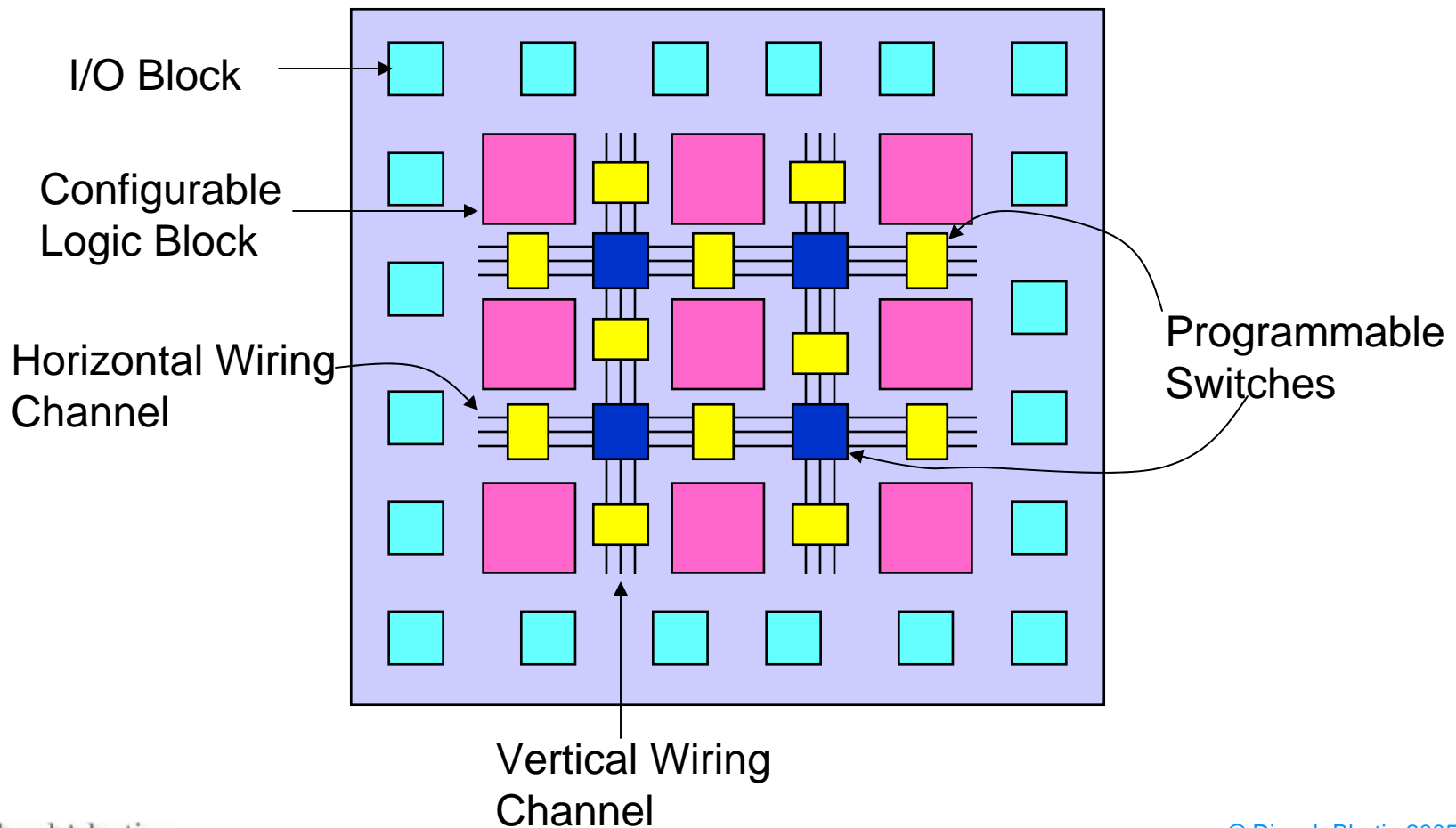
ASIC Design Flow



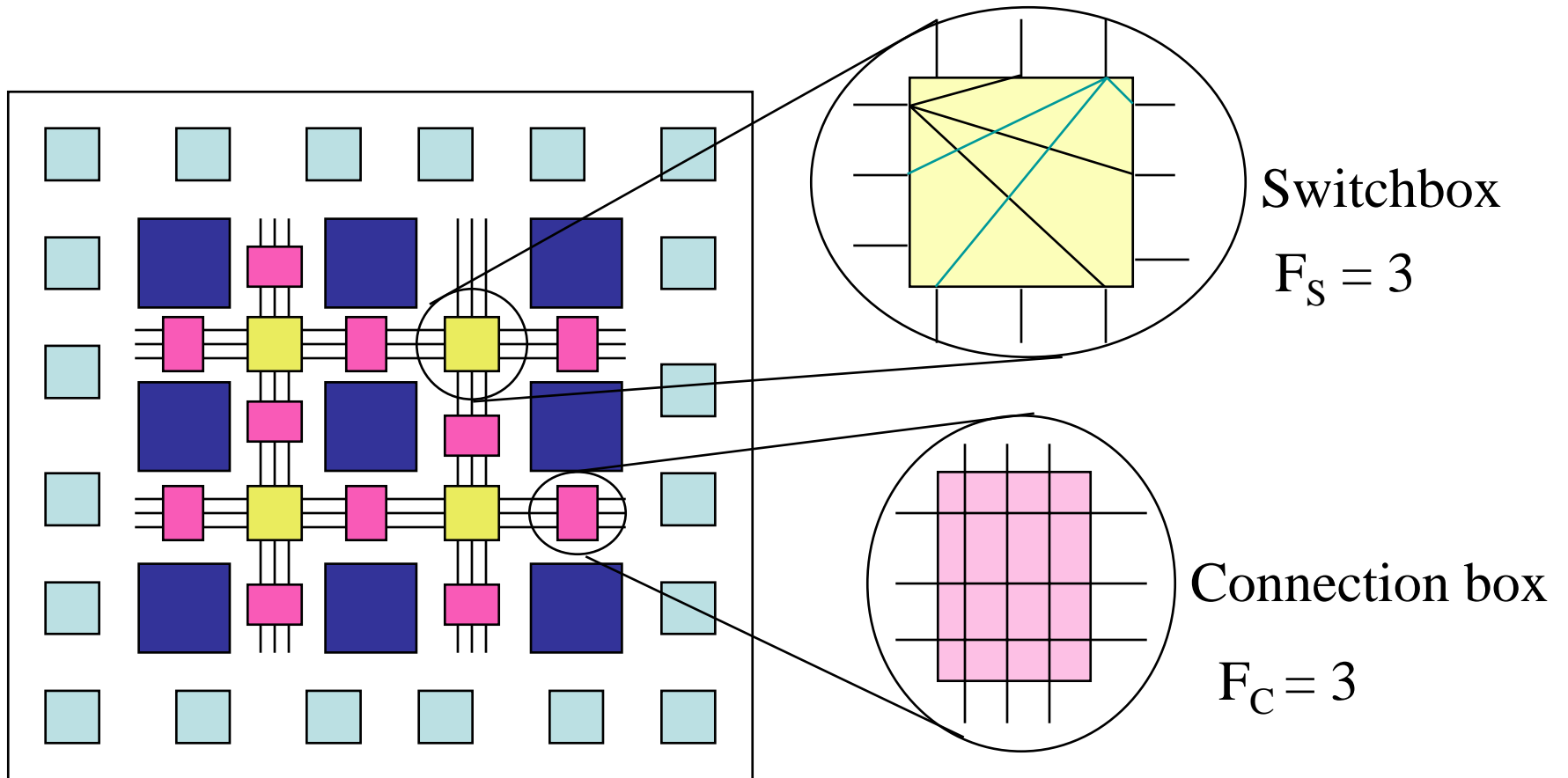
FPGA Design Flow



FIELD PROGRAMMABLE GATE ARRAY



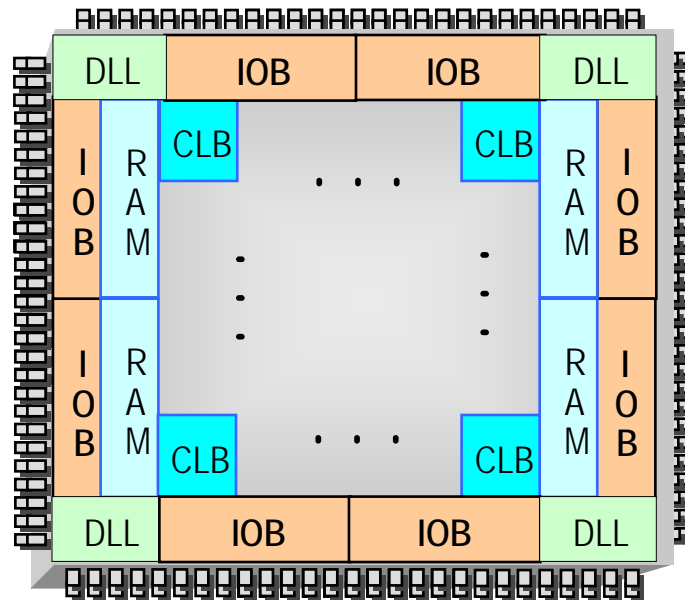
A GENERIC FPGA



XILINX SPARTAN DEVICES

I/O Connectivity

Memory Resources



Logic & Routing

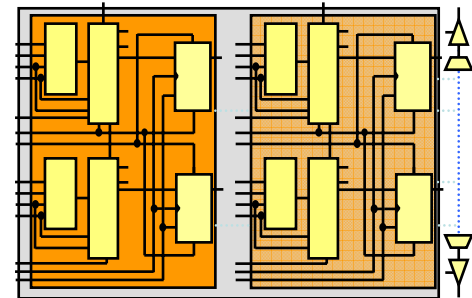
System Clock Management
Digital Delay Lock Loops (DLLs)

1/0/2005

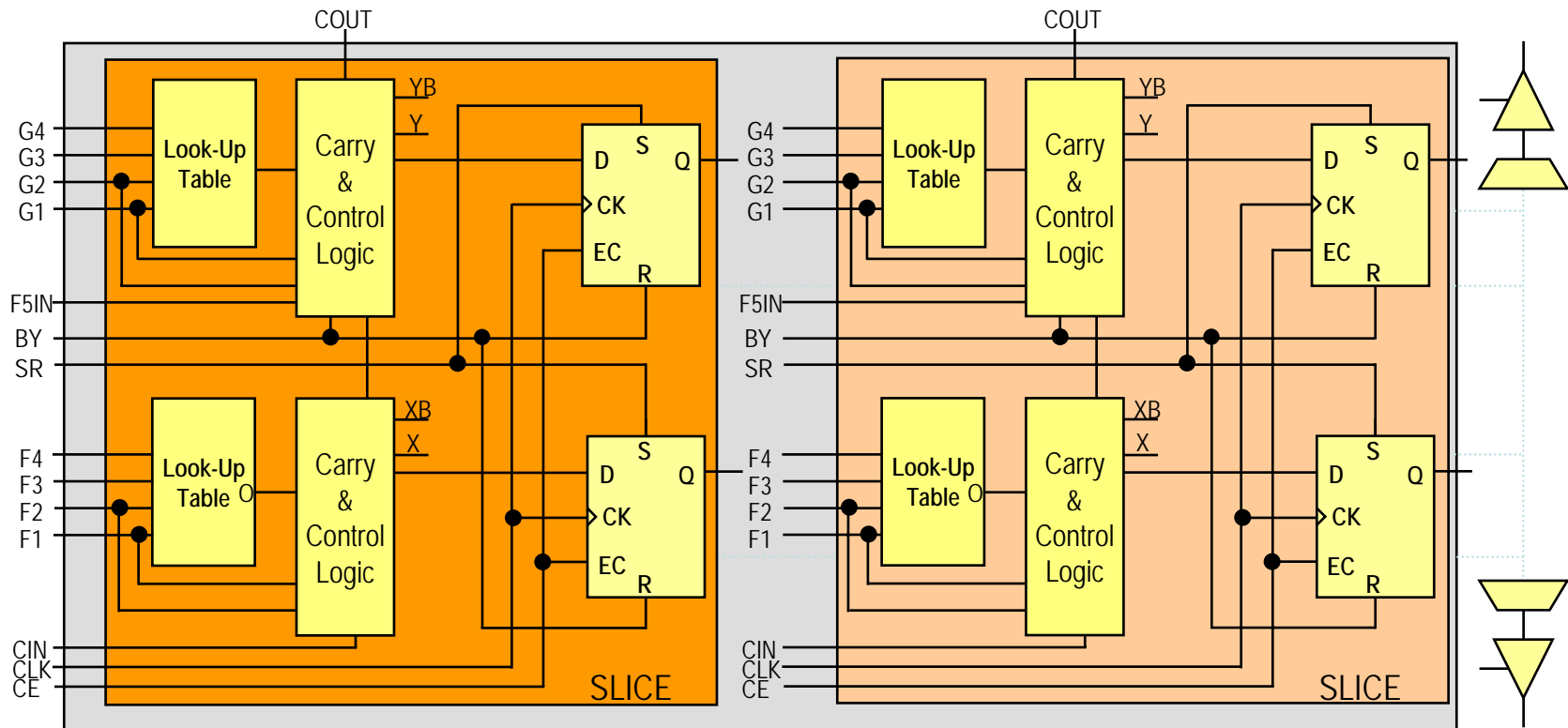
XILINX SPARTAN LOGIC & ROUTING

- ◆ Look Up Table (LUT) versatility
 - ◆ CLB primary building block
 - ◆ Flexible for logic or distributed RAM implementation
- ◆ Fast arithmetic operations
 - ◆ Specialized Carry Logic for arithmetic operations
 - ◆ Fast DSP functions
- ◆ Configurable for simple to complex logic
 - ◆ Allow up to 6 input functions into a one logic level

Configurable Logic Block (CLB)



CLB STRUCTURE



- ◆ Each slice has 2 LUT-FF pairs with associated carry logic
- ◆ Two 3-state buffers (BUFT) associated with each CLB, accessible by all CLB outputs

CLB SLICE STRUCTURE

- ◆ Each slice contains two sets of the following:

- ◆ **Four-input LUT**

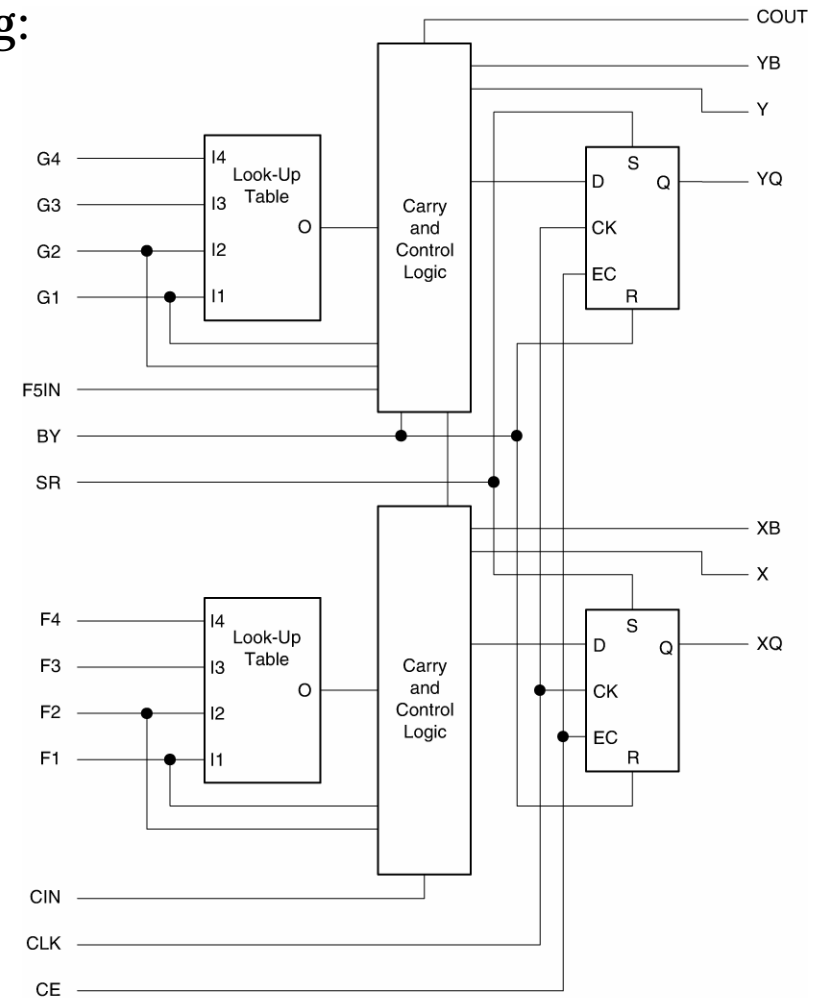
- ◆ Any 4-input logic function
 - ◆ Or 16-bit x 1 sync RAM
 - ◆ Or 16-bit shift register

- ◆ **Carry & Control**

- ◆ Fast arithmetic logic
 - ◆ Multiplier logic
 - ◆ Multiplexer logic

- ◆ **Storage element**

- ◆ Latch or flip-flop
 - ◆ Set and reset
 - ◆ True or inverted inputs
 - ◆ Sync. or async. control



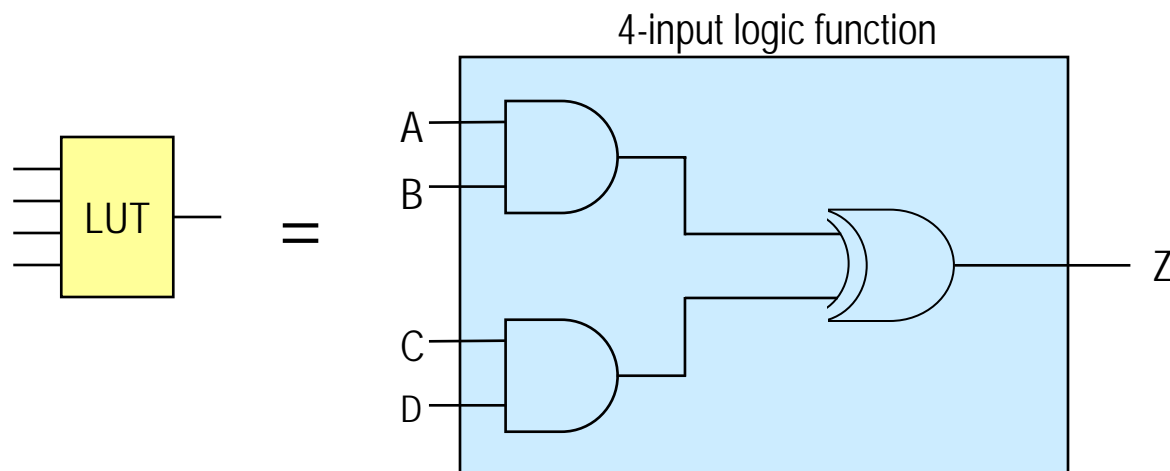
DS001_04_060100

FOUR-INPUT LUT

- ◆ Implements combinatorial logic
 - ◆ Any 4-input logic function
 - ◆ Cascaded for wide-input functions

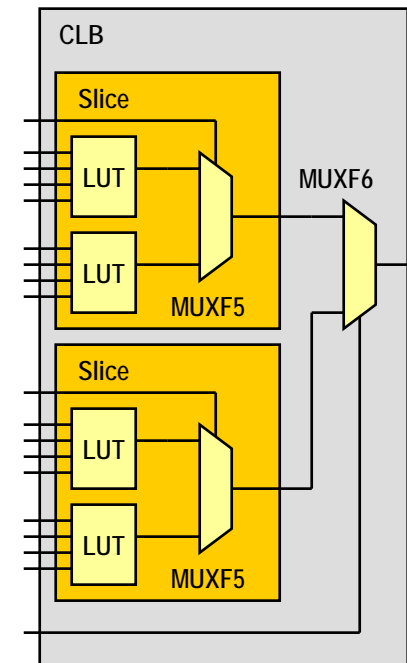
Truth Table

Inputs(ABCD)	Output(Z)
0000	0
0001	0
0010	1
0011	0
.....	..
1110	1
1111	1



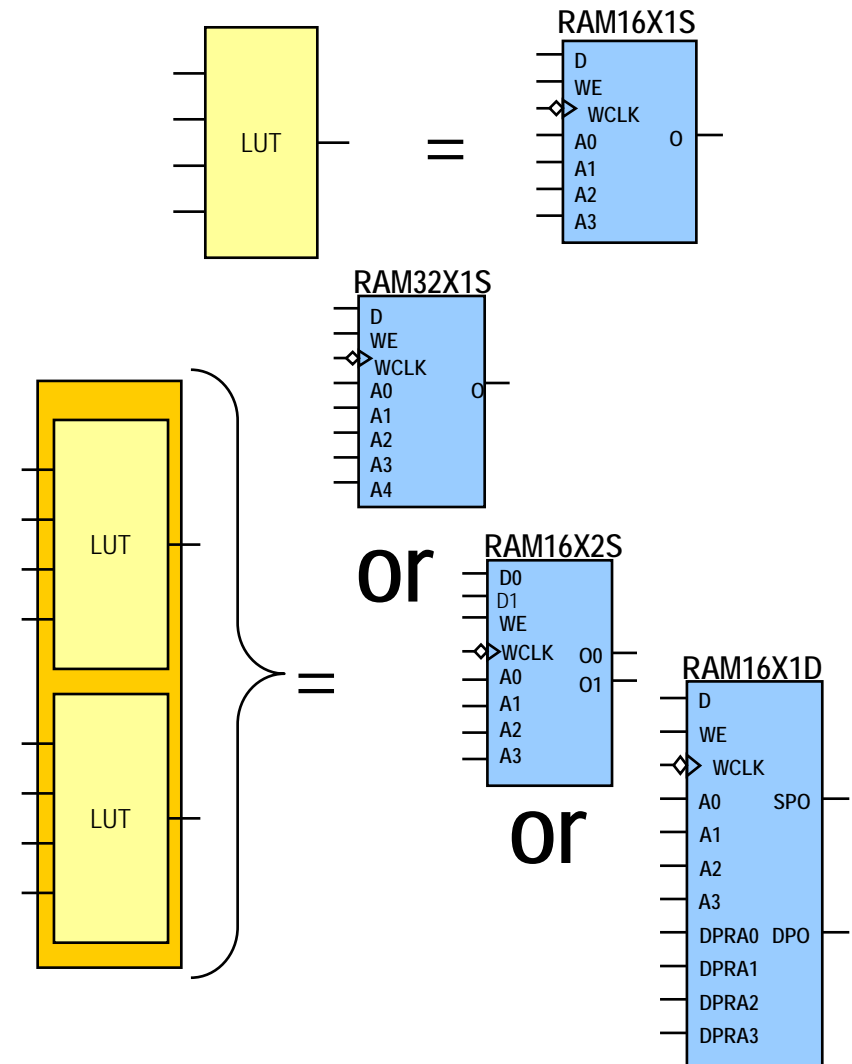
DEDICATED EXPANSION MULTIPLEXERS

- ◆ MUXF5 combines 2 LUTs to create
 - ◆ 4x1 multiplexer
 - ◆ Or any 5-input function (LUT5)
 - ◆ Or selected functions up to 9 inputs
- ◆ MUXF6 combines 2 slices to form
 - ◆ 8x1 multiplexer
 - ◆ Or any 6-input function (LUT6)
 - ◆ Or selected functions up to 19 inputs
- ◆ Dedicated muxes are faster and more space efficient

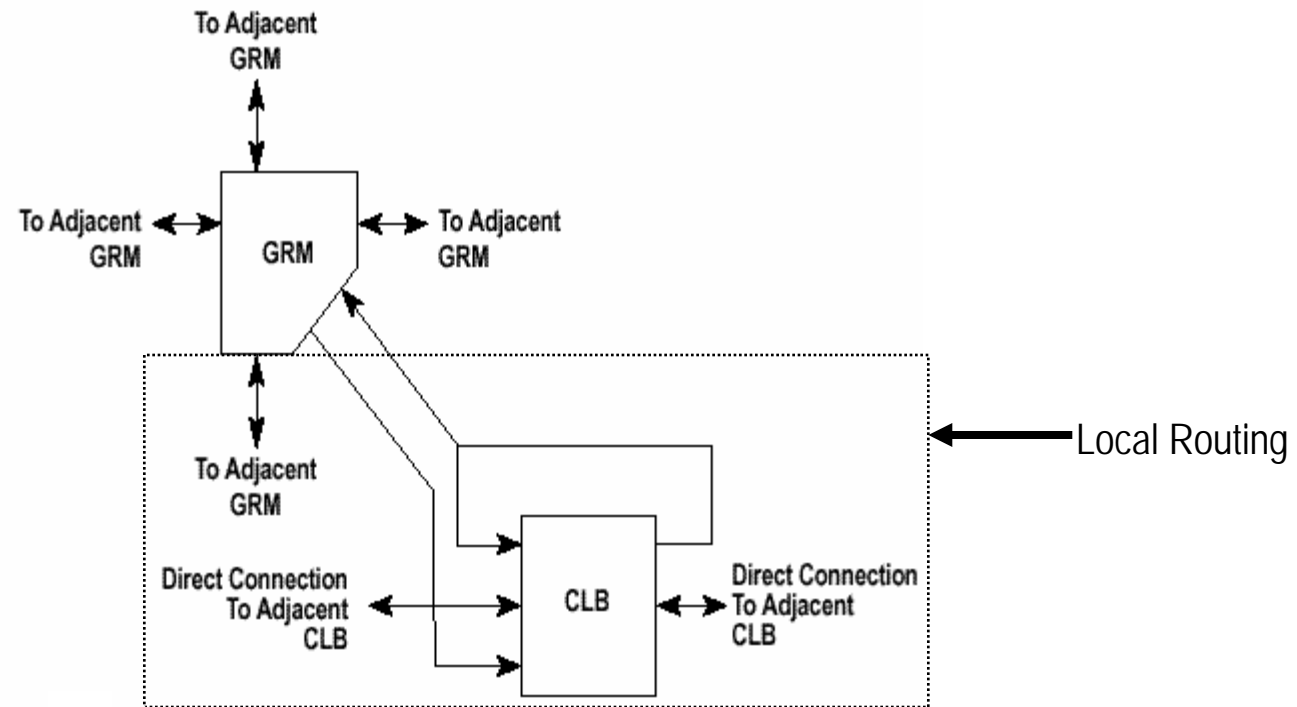


DISTRIBUTED RAM

- ◆ 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

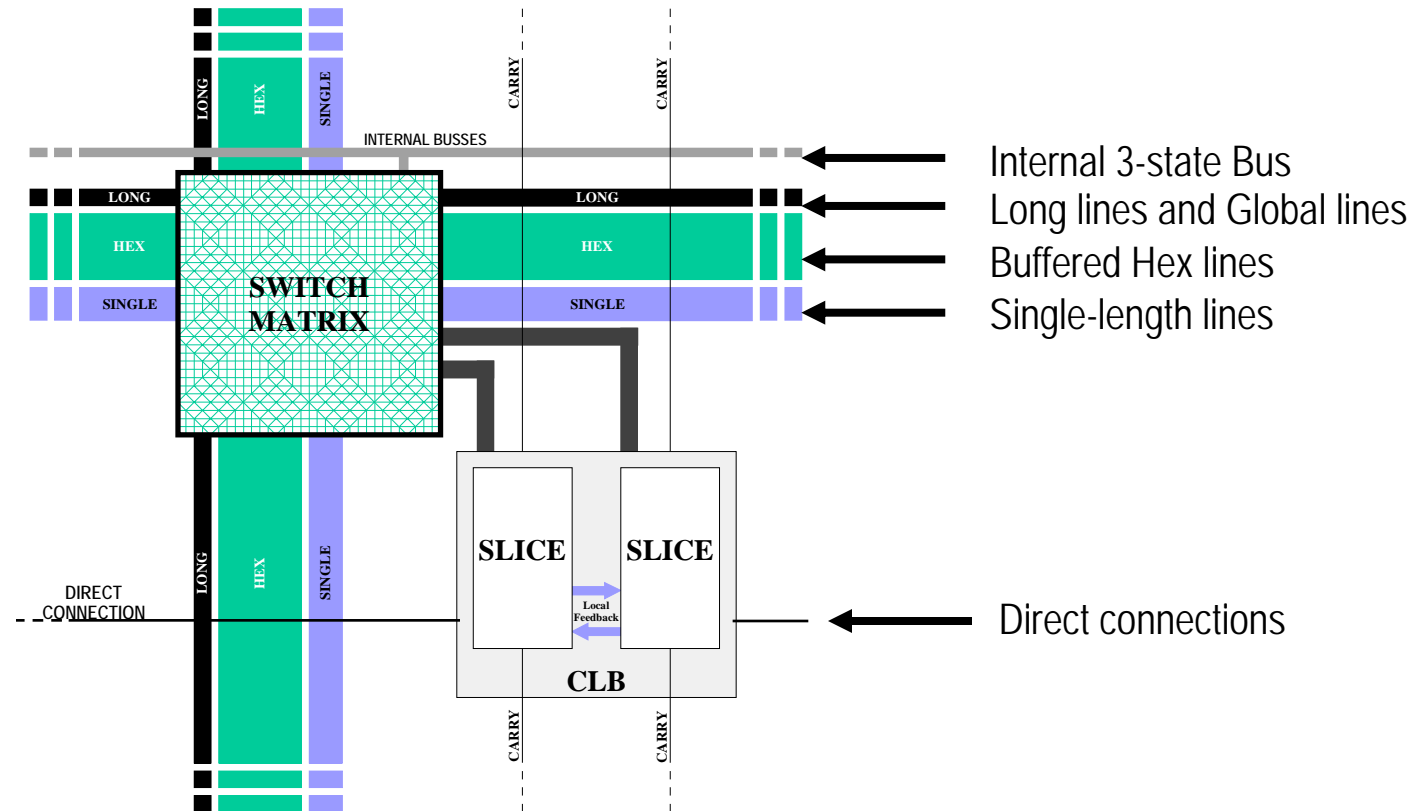


LOCAL ROUTING



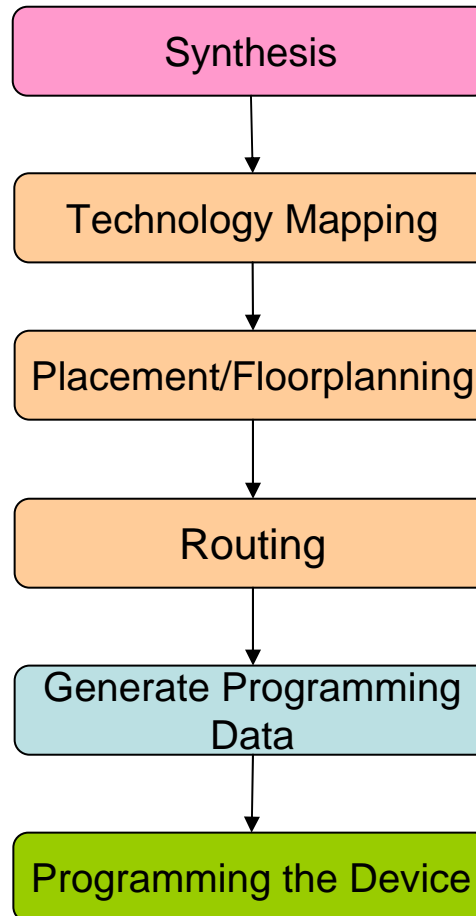
- ◆ Interconnect among LUTs, FFs, GRM
- ◆ CLB feedback path for connections to LUTs in same CLB
- ◆ Direct path between horizontally adjacent CLBs

GENERAL PURPOSE ROUTING



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

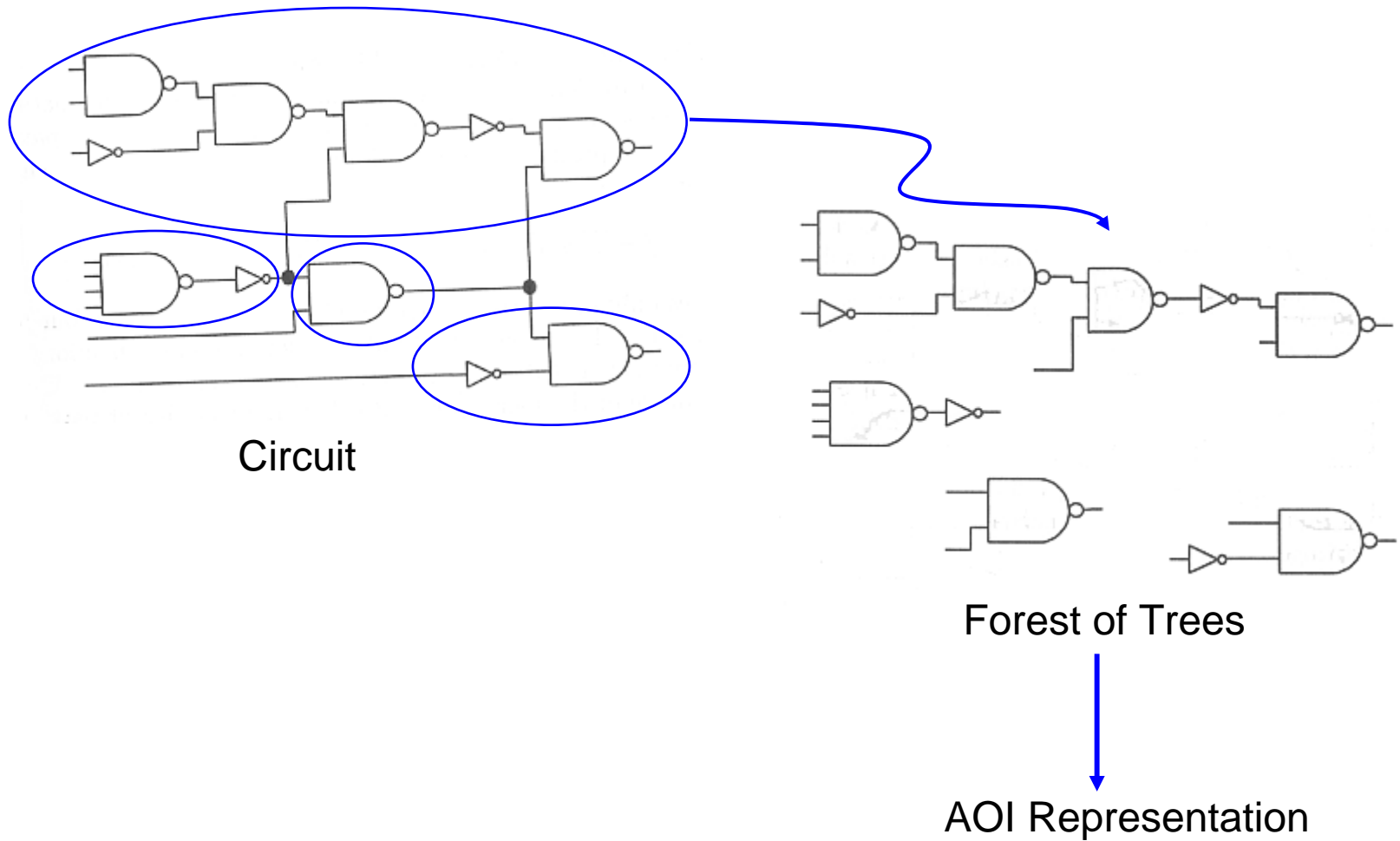
FPGA ARCHITECTURE AND CAD



LOOKUP TABLE BASED TECHNOLOGY MAPPING

- ◆ A k input lookup table is a digital memory that can implement any boolean function of k input variables.
 - ◆ It can implement 2^{2^k} functions.
- ◆ Library based technology mapping techniques are not efficient.
- ◆ New approaches are needed.

CIRCUIT REPRESENTATION



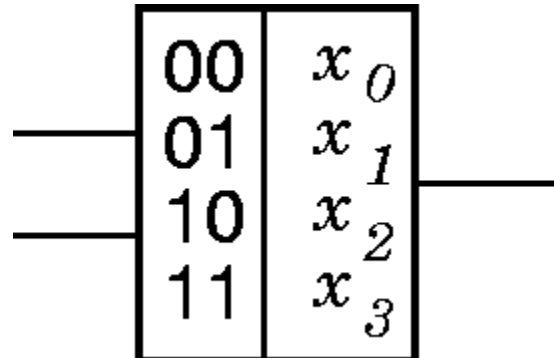
WHY DO FPGA BASED COMPUTING

- ◆ Need to understand
 - ◆ How costly (big) is a solution
 - ◆ How compare to alternatives
 - ◆ Cost and benefit of flexibility
- ◆ Complete implementation of our application
- ◆ For each architectural alternatives
 - ◆ In same implementation technology
 - ◆ with multiple area-time points
- ◆ Start sorting out
 - ◆ custom vs. configurable
 - ◆ spatial configurable vs. temporal

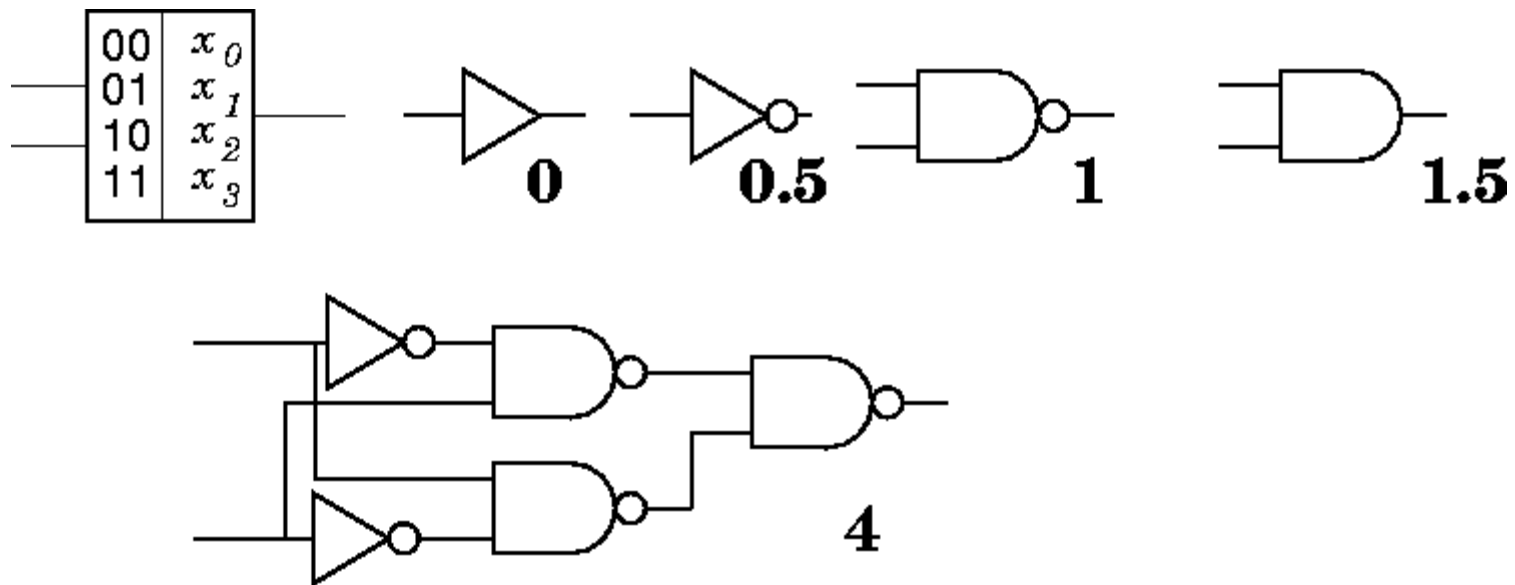
FPGA TABLE

Year	Design	Organization	Max	λ	λ^2 area	cycle
1986	Xilinx 2K	CLB (4-LUT)	100	1μ	500K	20 ns
1988	Xilinx 3K	CLB (2 \times 4-LUT)	320	0.6μ	1.3M	13 ns
1992	Xilinx 4K	CLB (2 \times 4-LUT +)	1024	0.6μ	1.25M	7 ns
1995	Xilinx 5K	CLB (4 \times 4-LUTS)	484	0.3μ	2.25M	6 ns
1995	Altera 8K	LE (4-LUT)	1296	0.3μ	920K	7.5 ns

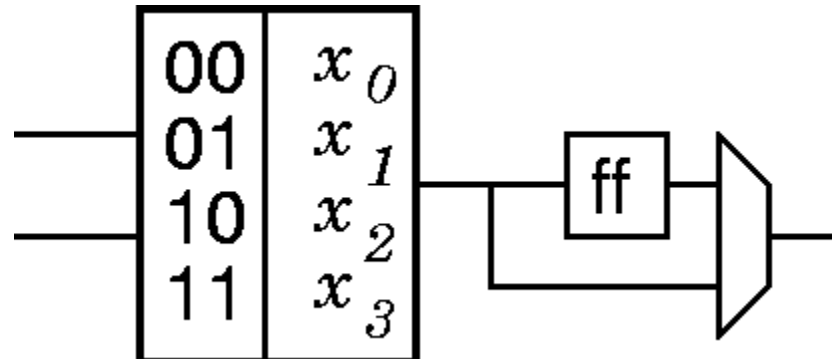
HOW MANY GATES?

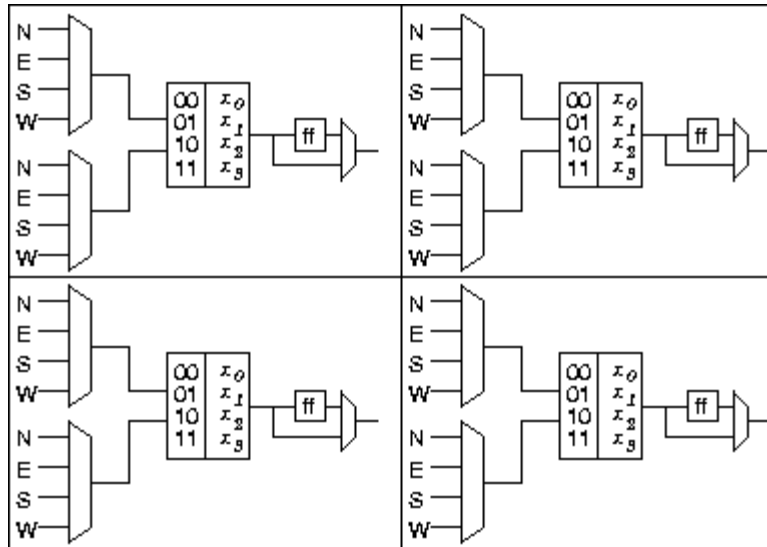


“GATES” IN 2-LUT



NOW HOW MANY?

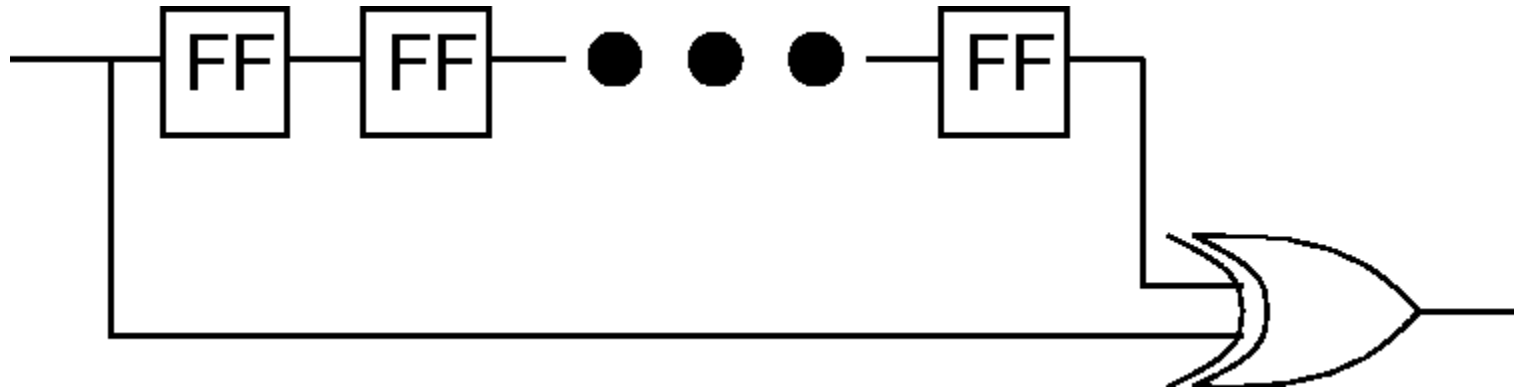




Gates/unit area?

Usable gates?

GATES REQUIRED?

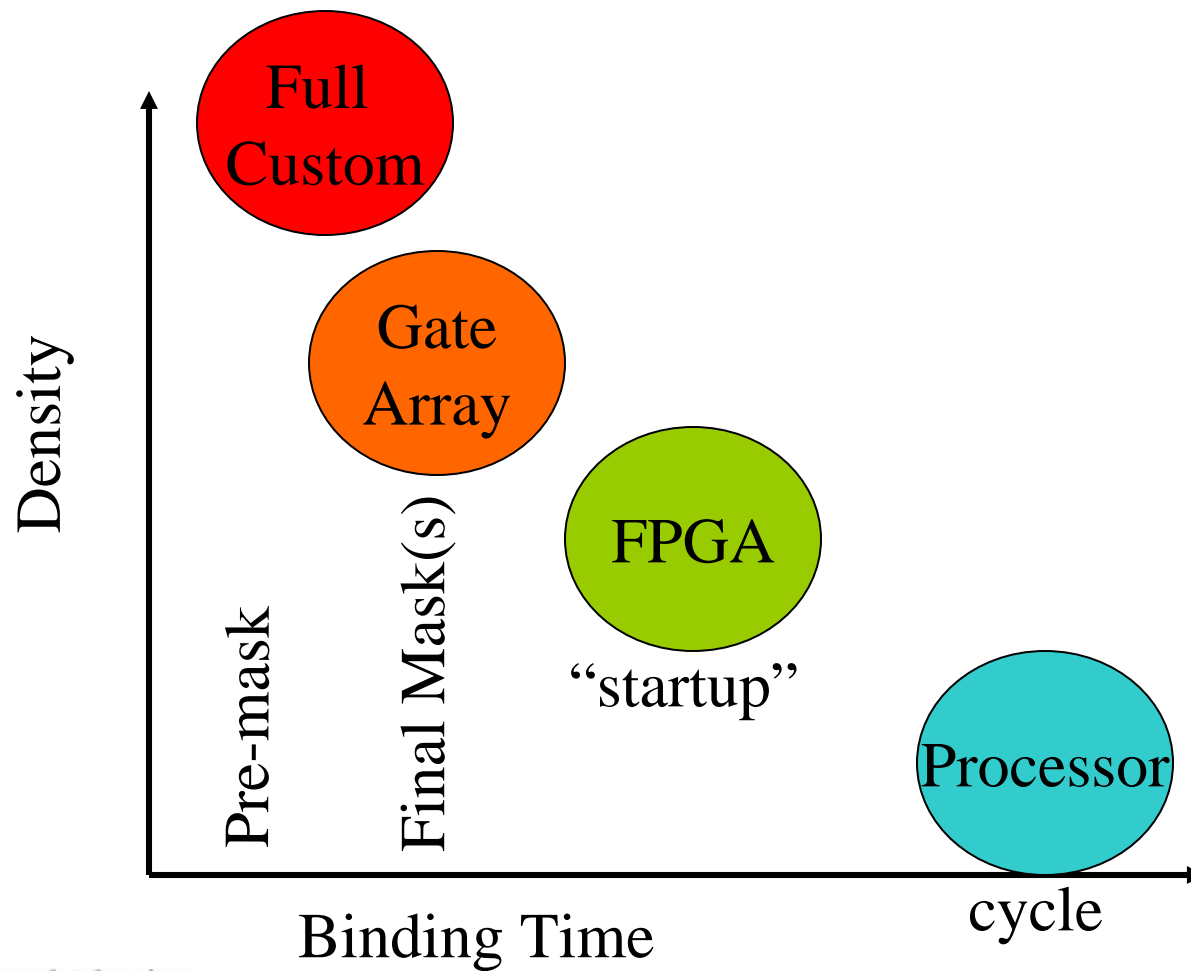


Depth=3, Depth=2048?

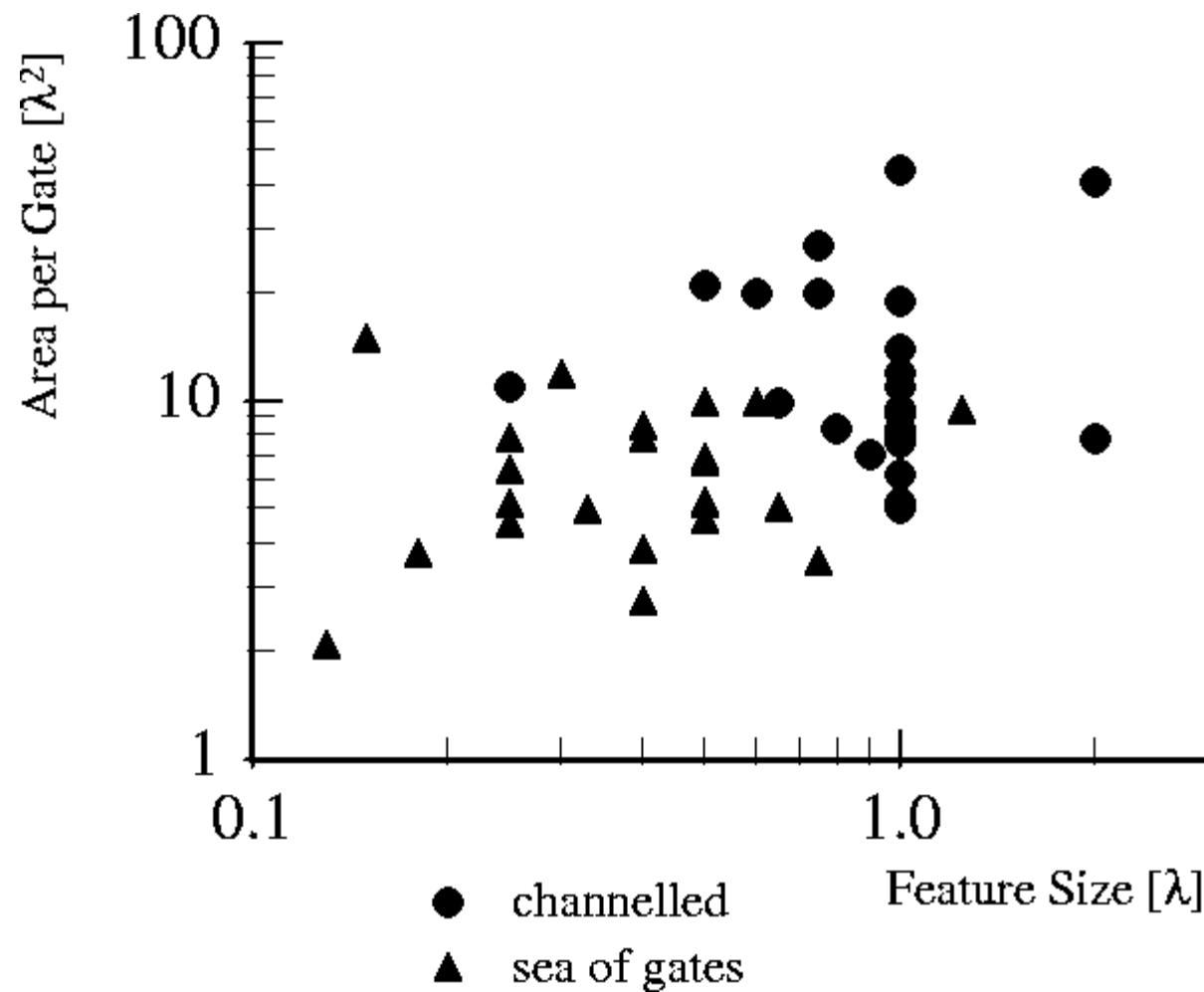
GATE METRIC FOR FPGAs?

- ◆ Several components for computations
 - ◆ **compute element**
 - ◆ **interconnect:**
 - ◆ space
 - ◆ time
 - ◆ **instructions**
- ◆ Not all applications need in same balance
- ◆ Assigning a single “capacity” number to device is an oversimplification

DENSITY VS. BINDING TIME



METAL PROGRAMMABLE GATE ARRAYS



MPGA vs. FPGA

- ◆ MPGA
 - ◆ $5\text{K}\lambda^2/\text{gate}$
 - ◆ 35-70% usable (50%)
 - ◆ $7\text{-}17\text{K}\lambda^2/\text{gate net}$
- ◆ Xilinx XC4K
 - ◆ $1.25\text{M}\lambda^2/\text{CLB}$
 - ◆ 17--48 gates (26?)
 - ◆ $26\text{-}73\text{K}\lambda^2/\text{gate net}$
- ◆ Ratio: 2--10 (5)

Adding $\sim 2\text{x}$ Custom/MPGA,
Custom/FPGA $\sim 10\text{x}$

MPGA vs. FPGA

- ◆ MPGA (SOG GA)
 - ◆ $\lambda=0.6\mu$
 - ◆ $\tau_{gd}\sim 1ns$
- ◆ Ratio: 1--7 (2.5)
- ◆ Xilinx XC4K
 - ◆ $\lambda=0.6\mu$
 - ◆ 1-7 gates in 7ns
 - ◆ 2-3 gates typical

PROCESSORS AND FPGAs

Metric: $\frac{4 \text{ input gate-evaluations}}{\lambda^2 \cdot s}$

Processor: $\frac{2 \times N_{ALU} \times w_{ALU}}{A_{proc} \times t_{cycle}}$

FPGA: $\frac{N_{ALUT}}{A_{array} \times t_{cycle}}$

PROCESSORS AND FPGAs

Year	Design	Organization	λ	λ^2 area	cycle	$\frac{ge's}{\lambda^2 \cdot s}$
Microprocessors						
1984	MIPS	1 × 32	1.5 μm	15M	250ns	17
1987	MIPS-X	1 × 32	1.0 μm	68M	50ns	19
1994	MIPS	1 × 32	0.28 μm	1.7G	2ns	19
1992	Alpha	1 × 64	0.38 μm	1.7G	5ns	15
1995	Alpha	2 × 64	0.25 μm	4.8G	3.3ns	18
1996	Alpha	2 × 64	0.18 μm	6.8G	2.3ns	17
FPGAs						
1986	Xilinx 2K	1 CLB (4 LUT)	1.0 μm	500K	20ns	100
1988	Xilinx 3K	64 CLBs (2 4-LUT)	0.6 μm	83M	13ns	120
1992	Xilinx 4K	49 CLBs (2 4-LUT)	0.6 μm	61M	7ns	230
1995	Xilinx 5K	49 CLBs (4 4-LUT)	0.3 μm	110M	6ns	290

RAW DENSITY SUMMARY

- ◆ Area
 - ◆ MPGA 2-3x Custom
 - ◆ FPGA 5x MPGA
- ◆ Area-Time
 - ◆ Gate Array 6-10x Custom
 - ◆ FPGA 15-20x Gate Array
 - ◆ Processor 10x FPGA

RAW DENSITY CAVEATS

- ◆ Processor/FPGA may solve more specialized problem
- ◆ Problems have different resource balance requirements
 - ◆ ...can lead to low yield of raw density

BROADENNING PICTURE

- ◆ Compare larger computations
- ◆ For comparison
 - ◆ **throughput density metric: results/area-time**
 - ◆ **normalize out area-time point selection**
 - ◆ **high throughput density**
 - -> most in fixed area
 - -> least area to satisfy fixed throughput target

MULTIPLY

Architecture	Feature Size (λ)	Area and Time	16x16		8x8	
			$\frac{\text{mpy}}{\lambda^2 \text{s}}$	scale	$\frac{\text{mpy}}{\lambda^2 \text{s}}$	scale
Custom 16x16	0.63 μm	2.6M λ^2 , 40 ns	9.6	9.6	9.6	9.6
Custom 8x8	0.80 μm	3.3M λ^2 , 4.3 ns			70	70
Gate-Array 16x16	0.75 μm	26M λ^2 , 30ns	1.3	1.3	1.3	1.3
FPGA (XC4K)	0.60 μm	1.25M λ^2 /CLB 316 CLBs, 26 ns 84 CLBs, 40 ns 220 CLBs, 12.1 ns 22 CLBs, 25 ns	0.097	0.24	0.30	1.5
16b DSP	0.65 μm	350M λ^2 , 50 ns	0.057	0.057	0.057	0.057
RISC (no multiplier)	0.75 μm	125M λ^2 , 66 ns/cycle two 16b operands – 44 cycles 16b constant – 7 cycles one 8b operand – 24 cycles 8b constant – 4 cycles	0.0028	0.017	0.0051	0.030

FIR

Architecture	Feature Size (λ)	Area and Time	$\frac{TAPs}{\lambda^2 S}$
32b RISC	$0.75\mu m$	$125M\lambda^2$, 66 ns/cycle \times 6+cycles/TAP	0.020
16b DSP	$0.65\mu m$	$350M\lambda^2$, 50 ns/TAP	0.057
32b RISC/DSP	$0.25\mu m$	$1.2G\lambda^2$, 40 ns/TAP	0.021
64b RISC	$0.18\mu m$	$6.8G\lambda^2$, 2.3 ns/TAP	0.064
FPGA (XC4K)	$0.60\mu m$	240 CLBs, 14.3 ns/8-TAPs	1.9
(Altera 8K)	$0.30\mu m$	30 LEs \times $0.92M\lambda^2/LE$, 10 ns/TAP	3.6
Full Custom	$0.75\mu m$	$400M\lambda^2$, 45 ns/64 TAPs	3.6
	$0.60\mu m$	$140M\lambda^2$, 33 ns/16 TAPs	3.5
	$0.75\mu m$	$82M\lambda^2$, 50 ns/10 TAPs	2.4
(fixed coefficient)	$0.60\mu m$	$114M\lambda^2$, 6.7 ns/43 TAPs (n.b. 16b samples)	56

DES KEYSEARCH

Architecture	Feature Size (λ)	Area	Keys/Second	$\frac{\text{Keys}}{\lambda^2 \text{s}}$
DES IC	$1.5 \mu\text{m}$	$11.1 \text{M}\lambda^2$	310K	0.028
FPGA (Altera 8K)	$0.30 \mu\text{m}$	$81188 (930 \text{M}\lambda^2)$	800K	0.00086
RISC	$0.30 \mu\text{m}$	$1.8 \text{G}\lambda^2$	41K	0.000023

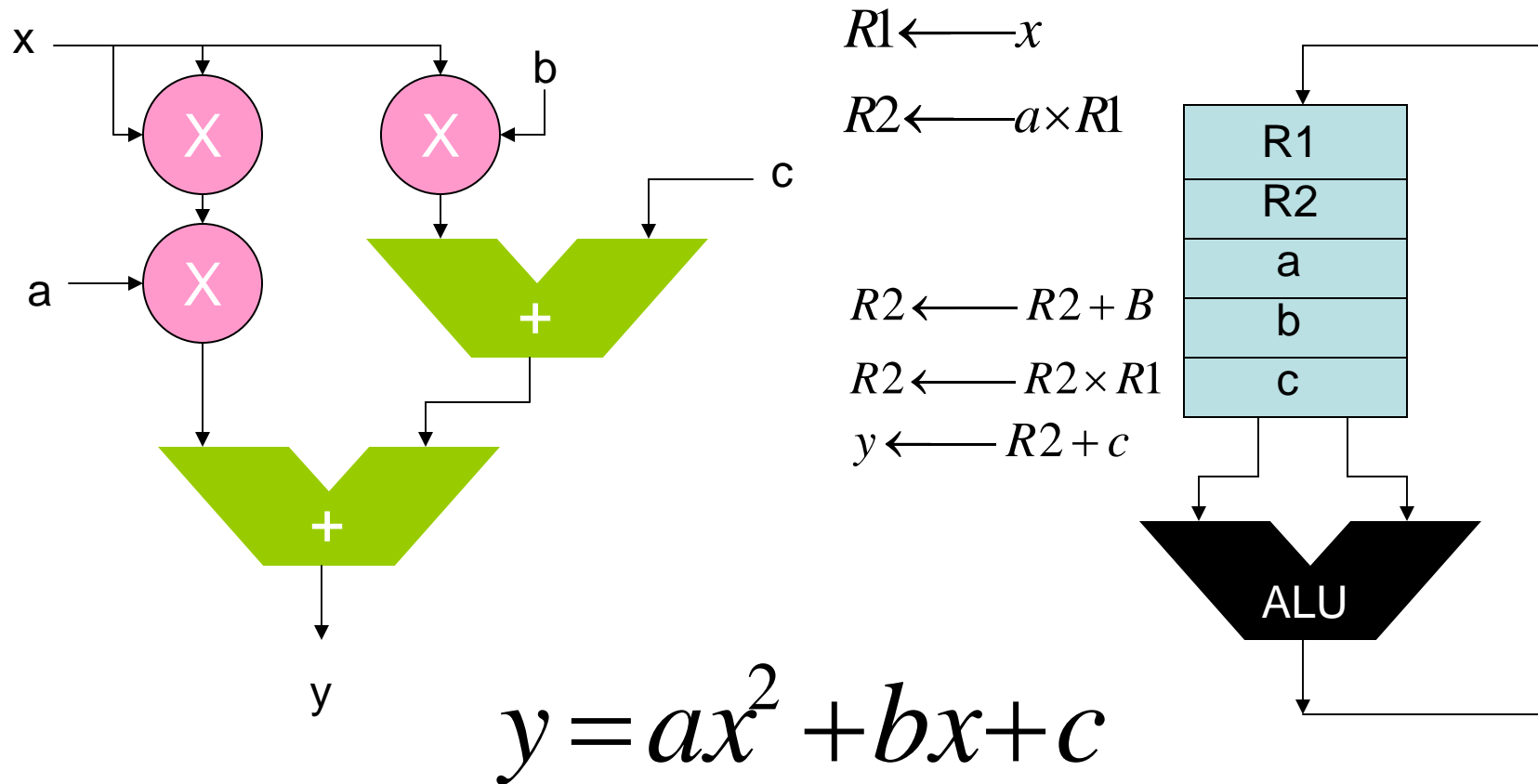
<<http://www.cs.berkeley.edu/~iang/isaac/hardware/>>

DNA SEQUENCE MATCH

Architecture	Feature Size (λ)	Area	Cell Updates per Second	$\frac{CU}{\lambda^2 s}$
Custom FPGA	2.0 μm	270M λ^2	500M	1.9
(SPLASH 2)	0.60 μm	43G λ^2	3,000M	0.070
(SPLASH)	0.60 μm	33G λ^2	370M	0.012
RISC				
(SparcStation 1)	0.75 μm	273M λ^2	0.87M	0.0032
(SparcStation 10)	0.40 μm	1.6G λ^2	1.2M	0.00075

N.B. includes memory area for SPLASH

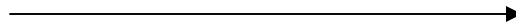
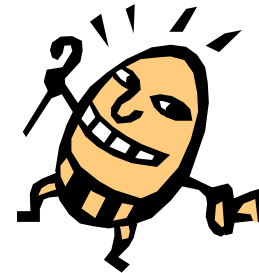
RECONFIGURABLE COMPUTING



WHAT IS IT ?

- ◆ *An ability for software to reach through the hardware layer and change the datapath for optimizing the performance.*

- ◆ *Not your father's X86 [Schmidt]*



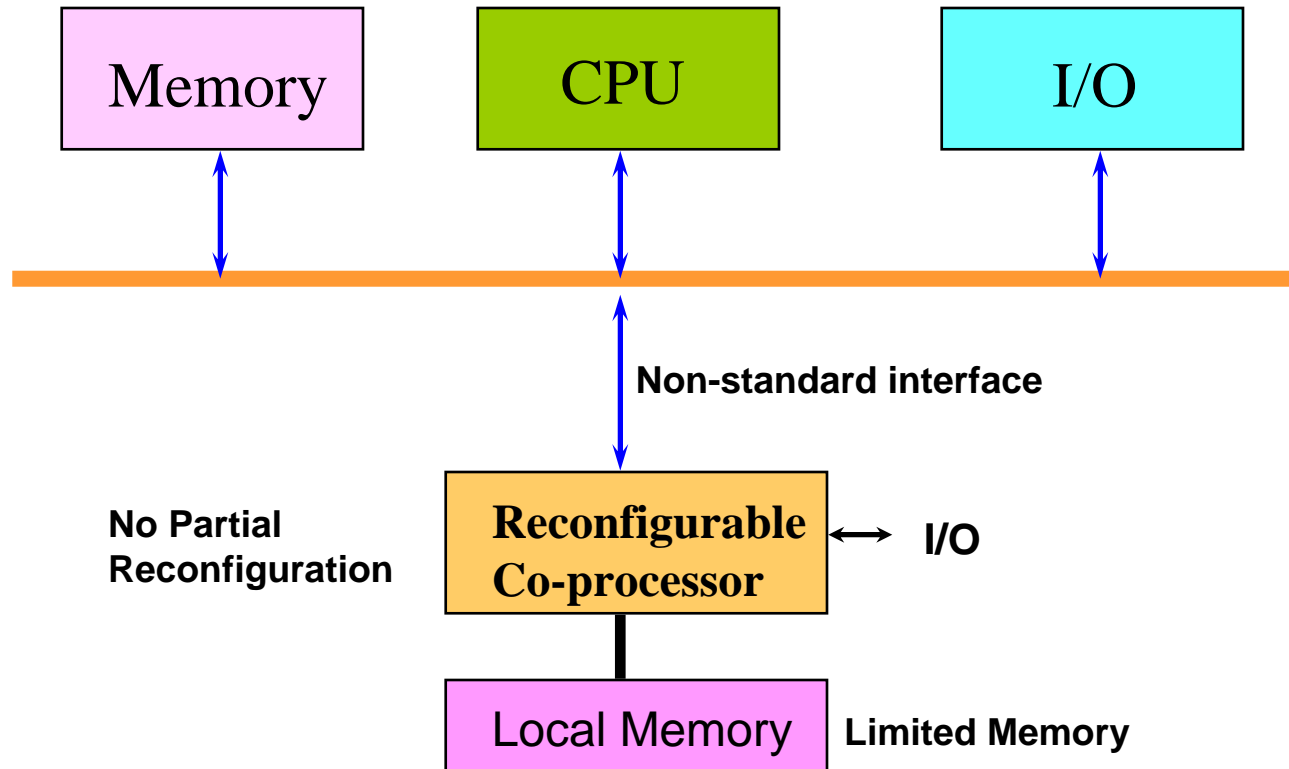
MOTIVATION

<i>Application</i>	<i>RPU</i>	<i>CPU</i>	<i>Speedup</i>
<i>Genetic Algorithms</i>	SPLASH-2	HP PA-RISC 125MHz	4
<i>DNA Matching</i>	SPLASH-2	CM-2	7100
<i>Hidden Markov Model</i>	RASA	SPARC-10	25
<i>Convolution</i>	SPLASH-2	SPARC-20	275
<i>DSP FIR</i>	1 FPGA	DSP Processor	18

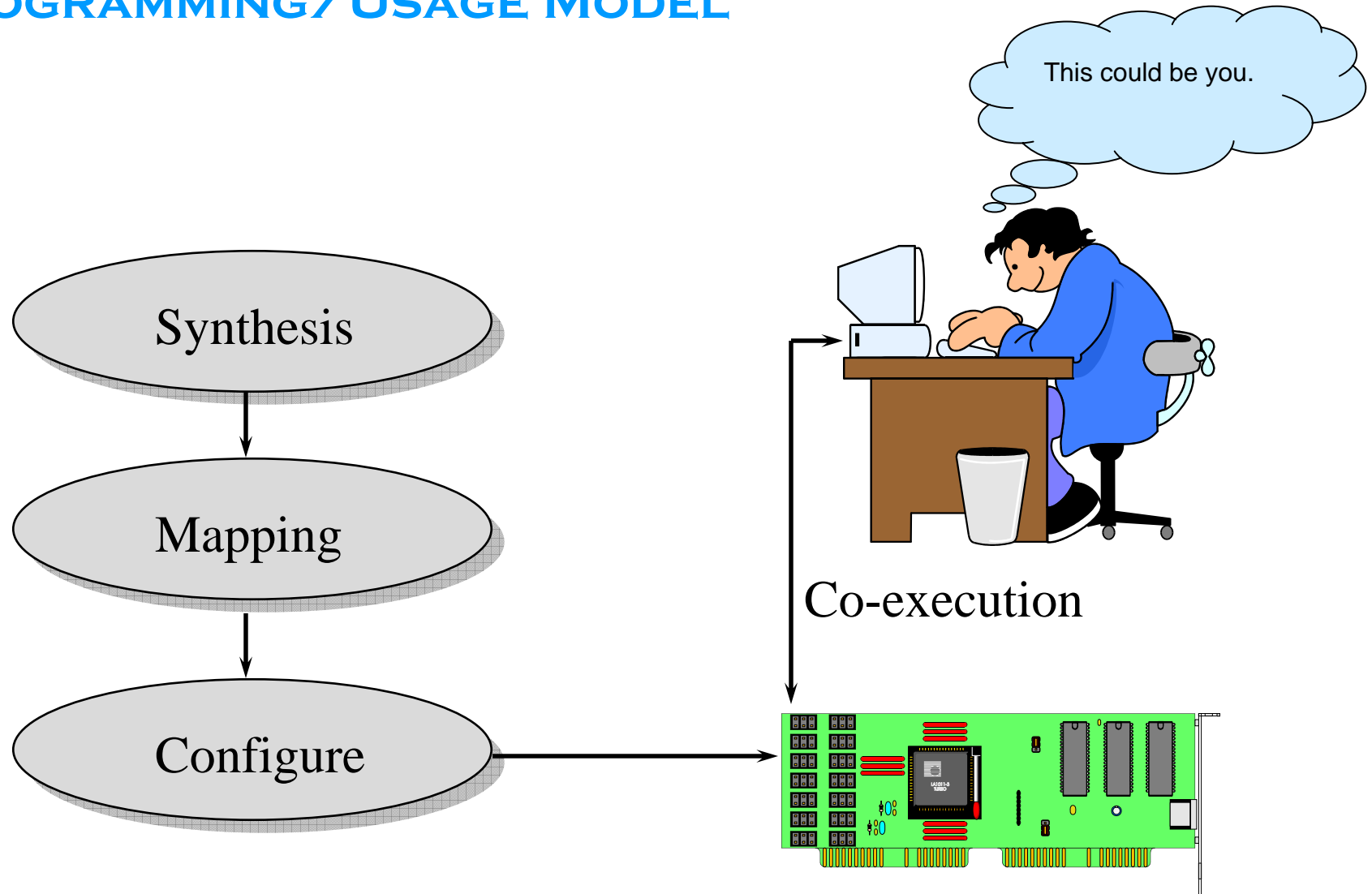
RECONFIGURABLE COMPUTERS

- ◆ **Static Applications**
 - ◆ GANGLION, iPoint, MacTester, SeeD-ROM, ZAREPTA, etc.
- ◆ **Coprocessing**
 - ◆ DISC, Chameleon, Spyder, nP, PRISM-I & II, etc.
- ◆ **Parallel Computing**
 - ◆ Splash-I & II, VC (P4), Hawaii Parallel Computer, NUMAchine, etc.
- ◆ **Application Specific Computing**
 - ◆ ACME, RRANN-I & II, DyANNA, SPACE, MORRH-ISA, Valtram, etc.
- ◆ **Hardware Emulators/Rapid Prototyping**
 - ◆ Quickturn, Zycad, Springbok, RPM, etc.
- ◆ **General Purpose Computing**
 - ◆ RACE-I, EVC1, Transmogripher, BORG, Anyboard, etc.

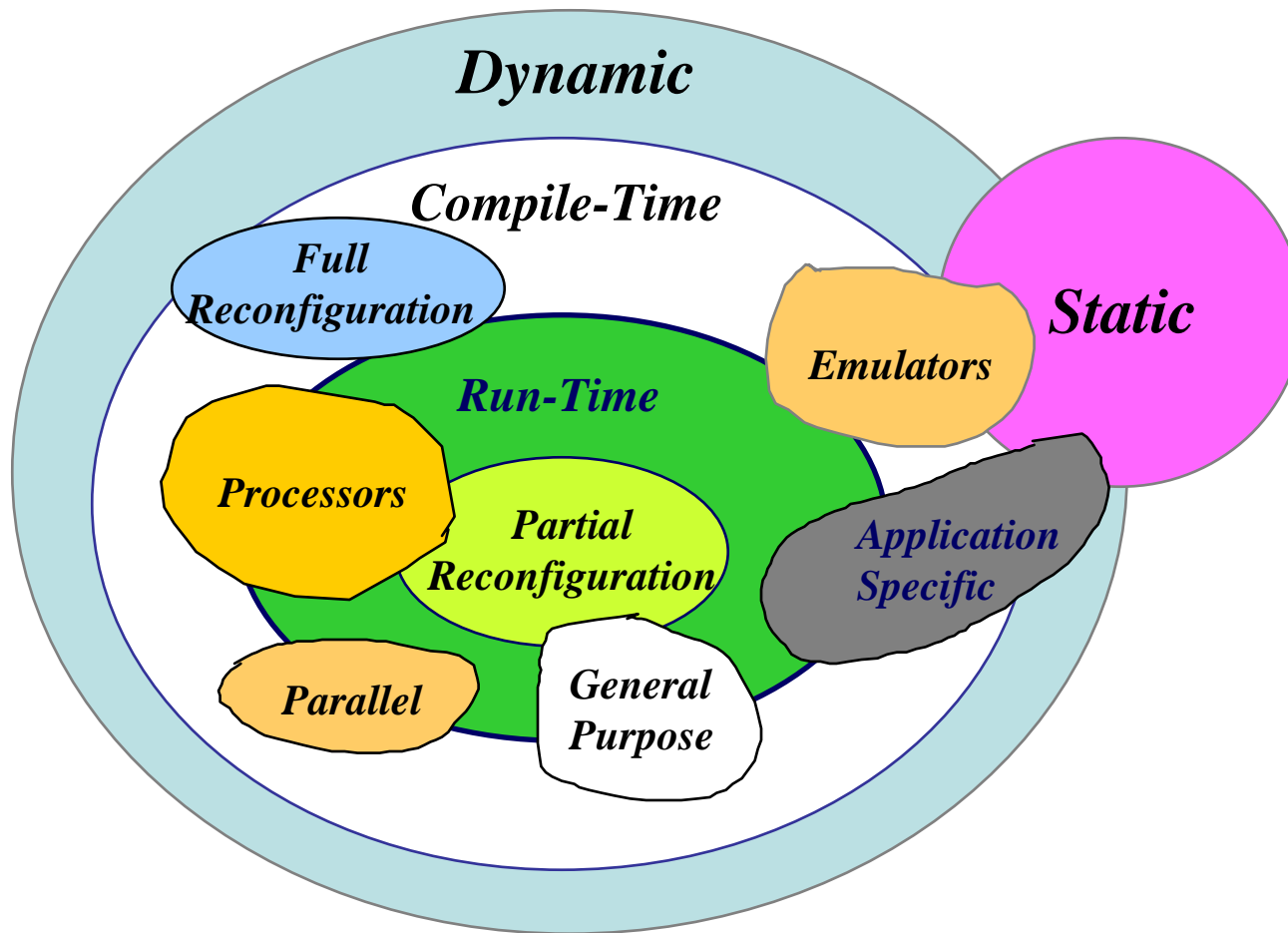
RECONFIGURABLE COMPUTERS



PROGRAMMING/USAGE MODEL



TYPES OF RECONFIGURABILITY



OBSTACLES

- ◆ Overhead
 - ◆ FPGA configuration time
 - ◆ Data transfer
 - ◆ System calls
- ◆ Programming difficulty
 - ◆ Hardware design of applications
 - ◆ Partitioning

RECONFIGURABLE ARCHITECTURE ISSUES

- ◆ More FPGAs are not always better
 - ◆ Dependent on type of application
 - ◆ Increases configuration time/programming difficulty
- ◆ More memory is not always better
 - ◆ Larger address/data bus may pin constrain FPGAs
 - ◆ Swapping/paging of memory
- ◆ More interconnections are not always better
 - ◆ Lack of CAD support
 - ◆ Timing issues/increase in configuration time (using FPICs)

GOOD REQUIRED FEATURES OF RC

- ◆ Easy to use
- ◆ User-friendly
- ◆ Hardware library
- ◆ Run-time reconfigurable
- ◆ CAD tool support

RECONFIGURABLE PROCESSORS

- ◆ Nano Processor
 - ◆ Developed in Brigham Young University
 - ◆ Stored Program Computer + Flexible Logic
 - ◆ Developed in 1994
- ◆ Dynamic Instruction Set Computer (DISC)
 - ◆ Instruction set modification
 - ◆ Partial reconfiguration
 - ◆ Relocatable hardware
- ◆ Target
 - ◆ Reconfigurable platforms with multiple processors on a board

PROBLEMS WITH RECONFIGURABLE SYSTEM DESIGN

- ◆ Conventional hardware development cycle
 - ◆ RTL, schematic capture etc
- ◆ Extensive knowledge required
 - ◆ Strong hardware development background
- ◆ Toolset is expensive
 - ◆ Synthesis tools, P&R etc.
- ◆ Requires careful hardware/software cosynthesis
- ◆ Adoption is tough

WHAT CAN A STORED PROGRAM PROCESSOR GIVE?

- ◆ Programmability
 - ◆ No respins
 - ◆ Hardware reuse
- ◆ Application specific performance
 - ◆ Less than ASIC
 - ◆ Better than general purpose processors for specific applications
- ◆ Can use high level programming language interfaces

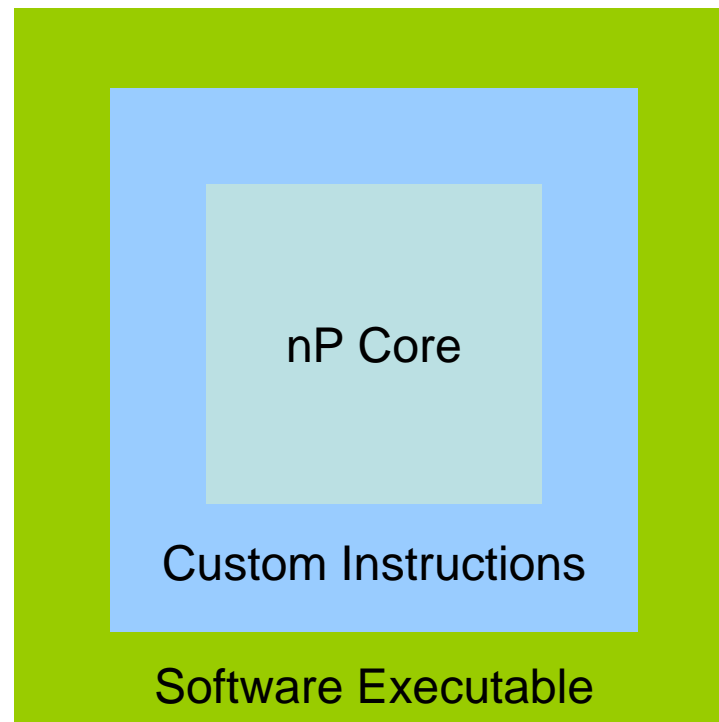
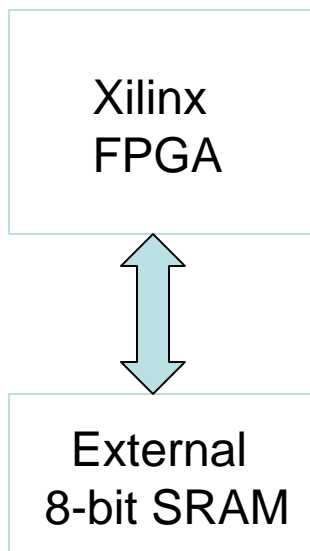
TWO WAYS TO TACKLE

- ◆ Reconfigure the processor hardware
 - ◆ Adapt register file, instruction set and datapath
 - ◆ Tailored for different applications
- ◆ Modify the software program to change processor behavior
 - ◆ More flexibility and adaptability

NANO PROCESSOR (NP)

- ◆ Instruction set processor on programmable fabric
 - ◆ Custom Instructions
 - ◆ Integrated assembler converts custom instructions
- ◆ Implemented within a FPGA
- ◆ Resource-savvy implementation

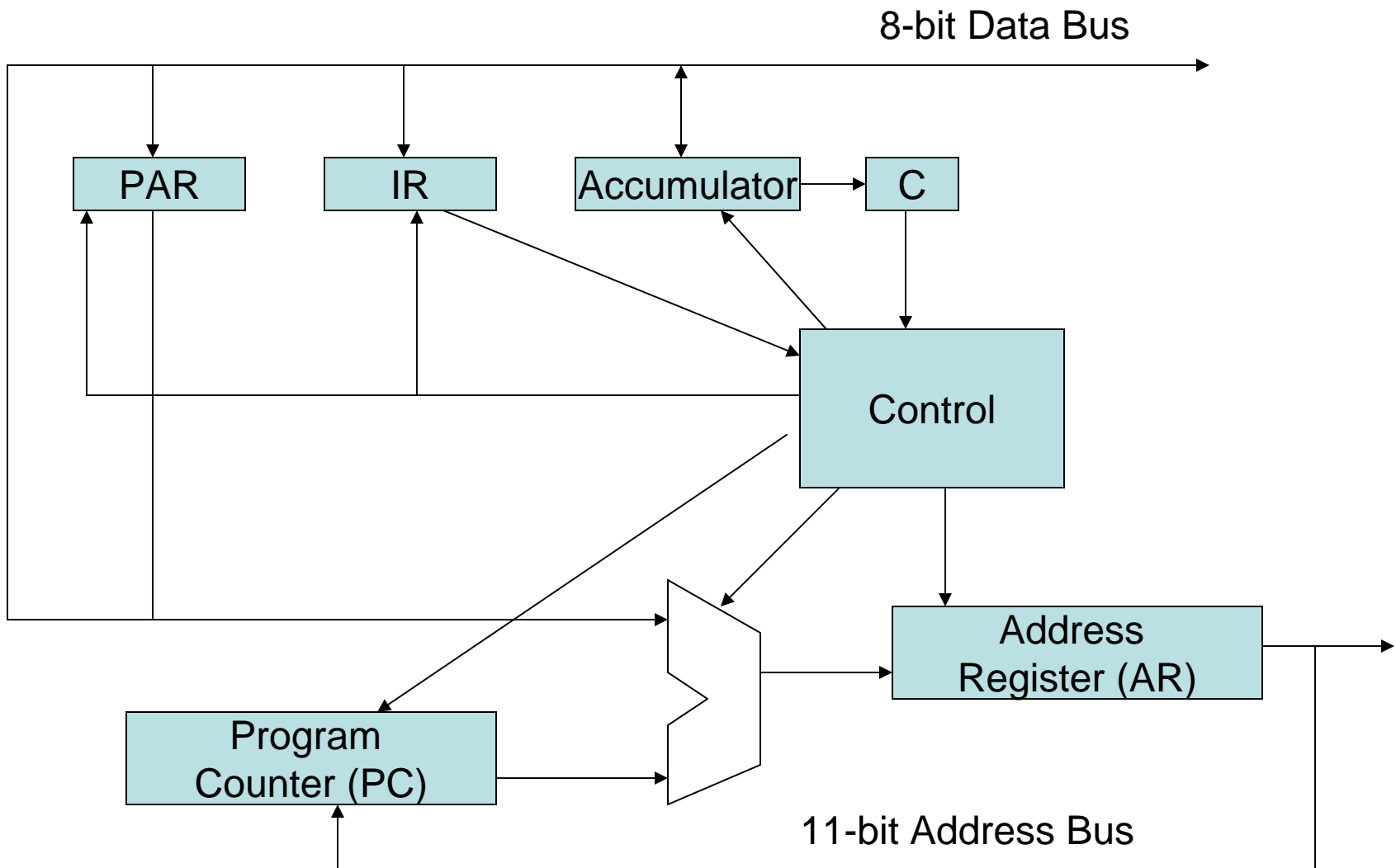
NP ORGANIZATION



NP CORE

- ◆ General purpose processor
 - ◆ Not intended for modification
- ◆ Three stage instruction cycle
 - ◆ Fetch (IF), Decode (ID) and Execute (EX)
- ◆ Six Instructions

ARCHITECTURE



SPECIFICS

- ◆ IR – Instruction Register
 - ◆ Five bits – 32 instructions
- ◆ PAR – Page Address Register
 - ◆ Three bits – Eight pages
- ◆ Program Counter and Address Register
 - ◆ Eleven bits – 2K addressing space
 - ◆ PC controls program flow
 - ◆ AR addresses final memory
- ◆ IR, PAR and AR are stored in IOB Flops

ACCUMULATOR (A)

- ◆ Single data register
 - ◆ Eight bits wide – matches the memory interface
- ◆ Addition and subtraction
- ◆ Single CLB implementation for each bit
 - ◆ Single level of logic
- ◆ Additional functionality using custom instructions

INSTRUCTION SET

- ◆ Six instructions
- ◆ Instruction length – two bytes
- ◆ Two parts – one opcode and one operand



INSTRUCTION CYCLE

- ◆ **IF**

IR \leftarrow mem[PC], 0-4

PAR \leftarrow mem[PC], 5-7

PC \leftarrow PC + 1

- ◆ **ID**

AR \leftarrow mem[PC] + PAR

PC \leftarrow PC + 1

- ◆ **EX**

A \leftarrow A op mem[AR]

INSTRUCTIONS

- ◆ **Store Accumulator** STR $\text{mem}[\text{AR}] \leftarrow A$
- ◆ **Load Accumulator** LD $A \leftarrow \text{mem}[\text{AR}]$
- ◆ **Load Accum + C** LDC $A \leftarrow \text{mem}[\text{AR}] + C$
- ◆ **Add with Carry** ADC $A \leftarrow A + C + \text{mem}[\text{AR}]$
- ◆ **Sub with Carry** SBB $A \leftarrow A - C - \text{mem}[\text{AR}]$
- ◆ **Jump at No Carry** JNC $\text{PC} \leftarrow \text{AR} (\text{if } C=0)$

CUSTOM INSTRUCTIONS

- ◆ Instruction library or added manually
- ◆ Done using schematic entry or HLS tools
- ◆ Example
 - ◆ **Sorting can be done using the simple instructions**
 - ◆ **Can create a new instruction to do hardware sorting**
- ◆ Define the instruction using assembler directive `.INST`
- ◆ Write conventional assembly language programs with custom + six instructions

TOOL FLOW

- ◆ Implement the custom instruction
- ◆ Instantiate the core
- ◆ Combine the netlist and flatten
- ◆ Implement using the vendor specific P&R tools

SOFTWARE EXECUTABLE

- ◆ Write assembly language programs with the combination
- ◆ Share instructions in as many processors as needed
 - ◆ No development cost
- ◆ No need for extra tools
 - ◆ Except for the nP assembler

SAMPLE INSTRUCTION DEFINITION

```
; test.inc
; .INST "<name>" <opcode> <opcode length>
.INST "STR"    0x07, 0x0001
.INST "LD"     0x02, 0x0001
.INST "LDC"    0x03, 0x0001
.INST "ADC"    0x01, 0x0001
.INST "SBB"    0x00, 0x0001
.INST "JNC"    0x05, 0x0001
```

SAMPLE NP PROGRAM

```
.include test.inc
:loop_back
    ld temp
    adc one
    str temp
    sbb count
    jnc stop
    adc zero
    jnc loop_back
stop :
    jnc stop
;data definitions
one:  .db 0x01
zero: .db 0x00
count: .db 0xdd
temp: .db 0x00
```

DYNAMIC INSTRUCTION SET COMPUTER (DISC)

- ◆ Demand driven instruction set modification
- ◆ Uses partial reconfiguration
 - ◆ Instruction modules are paged in/out on demand
 - ◆ Free up resources that are not needed
- ◆ Physically relocates instruction modules to available FPGA space
- ◆ Suited for small FPGAs
 - ◆ Custom processor and all instructions may not fit the FPGA
 - ◆ Use partial reconfiguration to alleviate that problem

ADVANTAGES

- ◆ Idle instructions can be removed
- ◆ Ability to have a much larger instruction set than a statically configured FPGA
- ◆ Partial reconfiguration
 - ◆ **Involves overhead**
 - ◆ Better than full reconfiguration though
 - ◆ 1/60 to 1/3 of the reconfiguration time required
 - ◆ **Instructions must be more coarse grained**
 - ◆ Overhead will kill the runtime advantage otherwise
- ◆ State Saving
 - ◆ **Do not destroy flip-flop contents**

RELOCATABLE HARDWARE

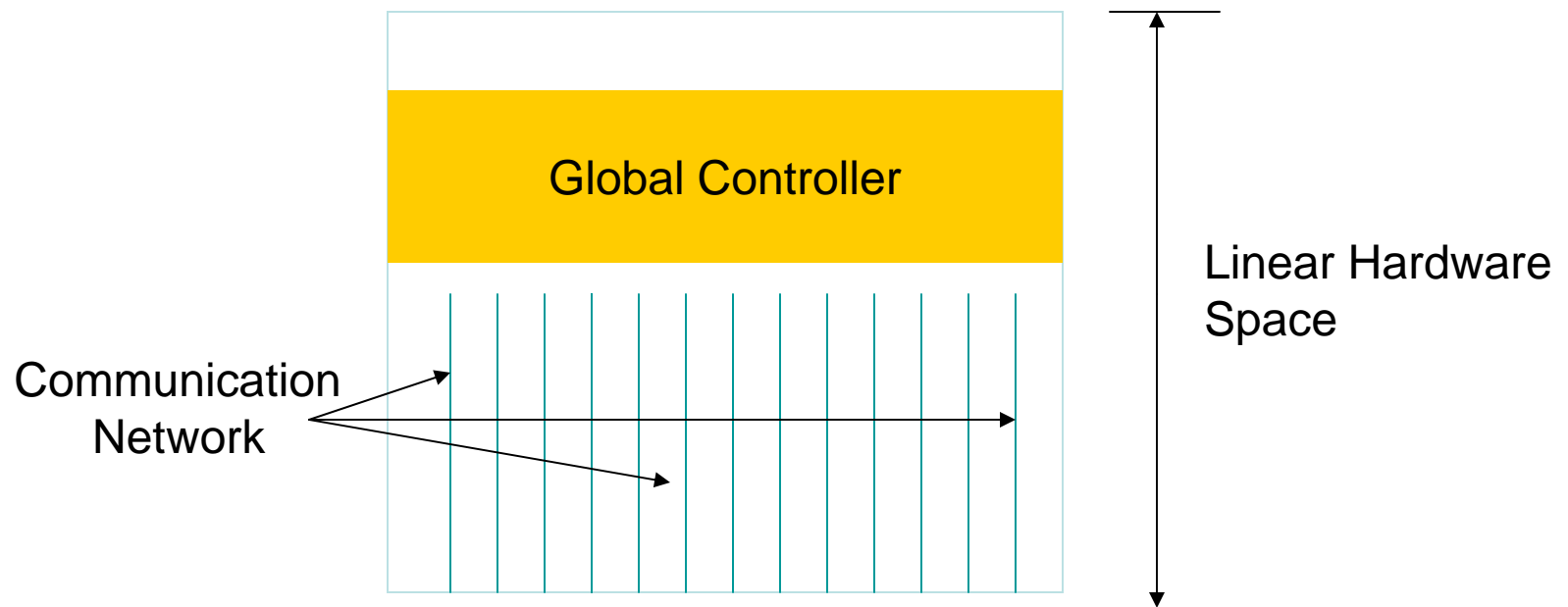
- ◆ Static implementation requires single fixed location
 - ◆ Global optimization is a possibility though
- ◆ Partial reconfiguration
 - ◆ Good for page in/out of idle hardware
 - ◆ Can pose serious performance problems though
 - ◆ Two instructions that are continuously swapped in and out
 - ◆ Instructions are designed in such a way that they can go to any location in the FPGA

MORE ON RELOCATABLE HARDWARE

- ◆ Custom instructions follow a global context
 - ◆ Defines physical locations
 - ◆ Communication network necessary for the modules
- ◆ Divide the available hardware into array of potential locations
 - ◆ Design the instructions in such a way that they are physically independent of each other

LINEAR HARDWARE SPACE

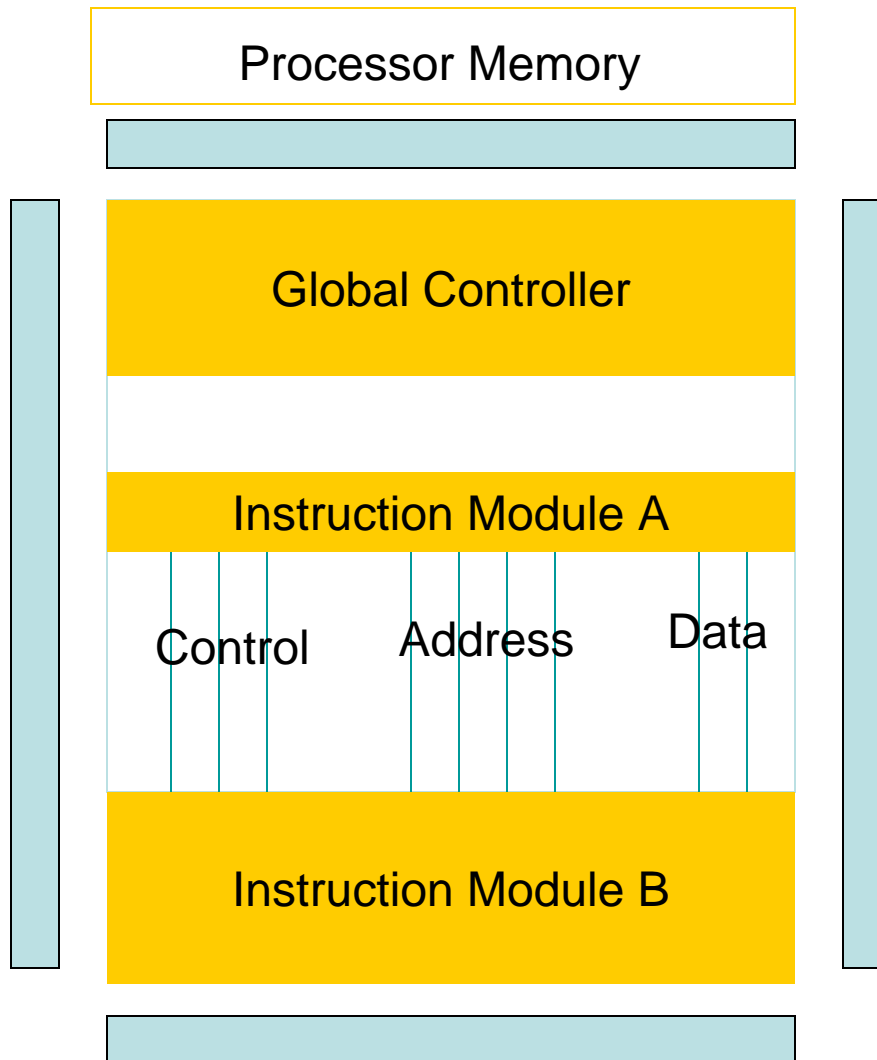
- ◆ One dimensional hardware space
 - ◆ Array of rows
 - ◆ Location is vertical location
 - ◆ Module size is defined by module height



DESIGN OF SUB-MODULES

- ◆ Submodules are designed horizontally
 - ◆ Occupies the width of the FPGA
- ◆ Full access to all global signals
 - ◆ Irrespective of the vertical placement
- ◆ Can consume arbitrary amount of hardware
 - ◆ Vary the height

DISC ARCHITECTURE



GLOBAL CONTROLLER

- ◆ A simple processor
- ◆ Operate and Monitor
 - ◆ External RAMs, internal communication network and global state
 - ◆ 10 rows on a National Semiconductor CLAy31 FPGA
- ◆ 8-bit datapath
- ◆ Single register
- ◆ 16-bit program counter – 32K addressing
- ◆ Sequences through the program

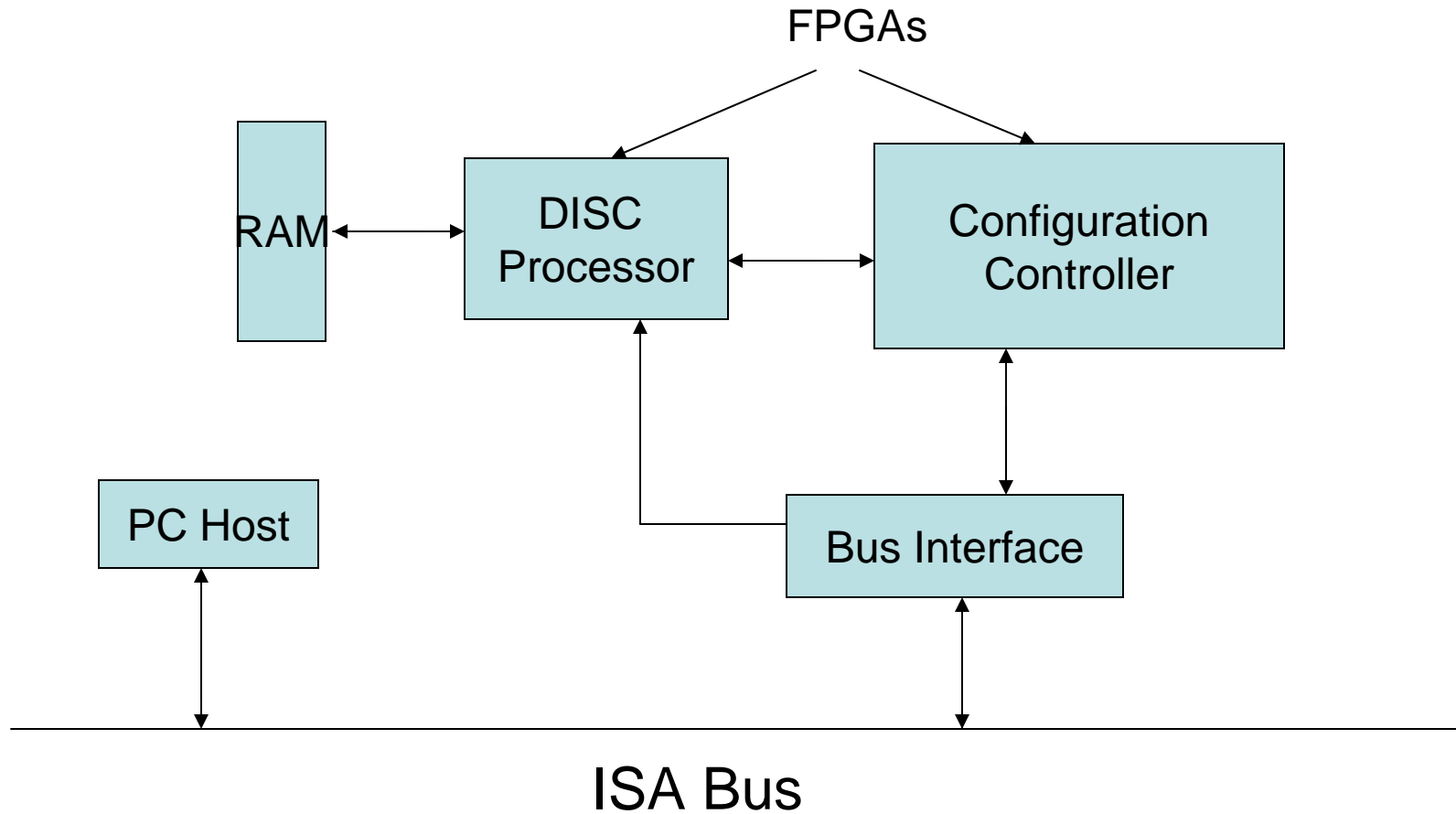
INSTRUCTION MODE

- ◆ Standard Mode
 - ◆ Arithmetic and logic operations
- ◆ Custom mode
 - ◆ Requires more than one cycle
 - ◆ Custom instructions
- ◆ Some default instructions for control
 - ◆ Set and clear carry
 - ◆ Load and store data register
 - ◆ Conditional jump

CUSTOM INSTRUCTION MODULES

- ◆ Each module contains a decode, datapath and control unit
- ◆ Decode unit
 - ◆ Compares the opcode with its designated opcode in the first cycle
 - ◆ On match, signal the controller that h/w is present
- ◆ Datapath performs operations and signals the control unit
- ◆ Control unit monitors all local signals and signals end of operation to global controller

SYSTEM OPERATION



INSTRUCTION EXECUTION

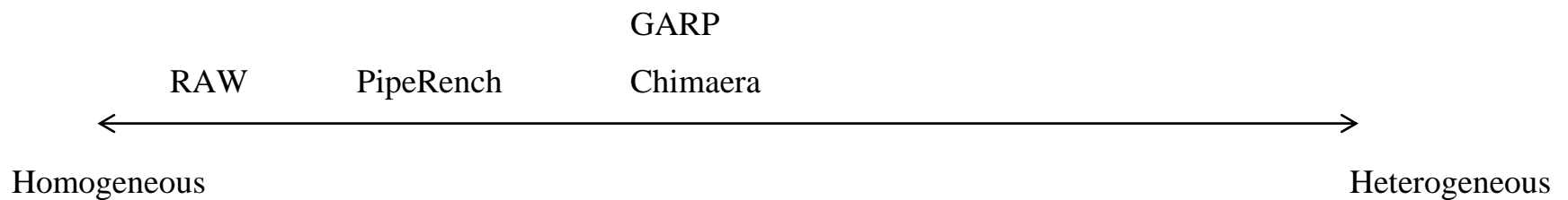
1. Fetch Instruction
2. Instruction present?
 1. Yes, go to Step 7.
3. Hardware Available?
 1. Yes, go to step 7.
4. Remove old instruction
5. Compute New location
6. Configure instruction module
7. Execute instruction; Goto step 1

OTHER POINTERS

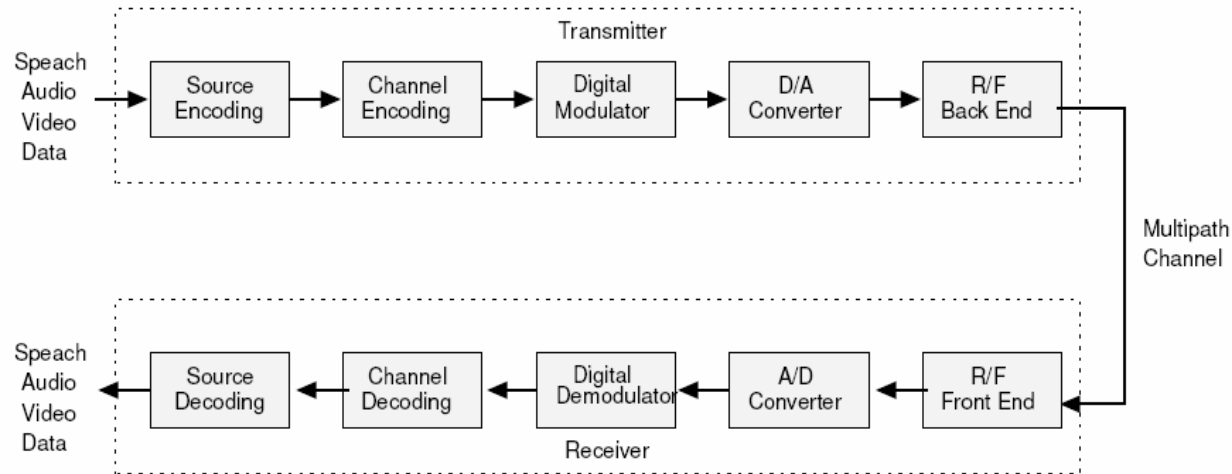
- ◆ NIOS
 - ◆ Altera
- ◆ Microblaze
 - ◆ Xilinx
- ◆ Hybrid ASIC
 - ◆ ASIC with a programmable fabric inside it
 - ◆ IBM + Xilinx
 - ◆ Best of both worlds

A SPECTRUM OF ARCHITECTURES

- ◆ Reconfigurable Processors
 - ◆ Berkeley Garp
 - ◆ CMU PipeRench
 - ◆ Northwestern Chimaera
 - ◆ MIT RAW Project

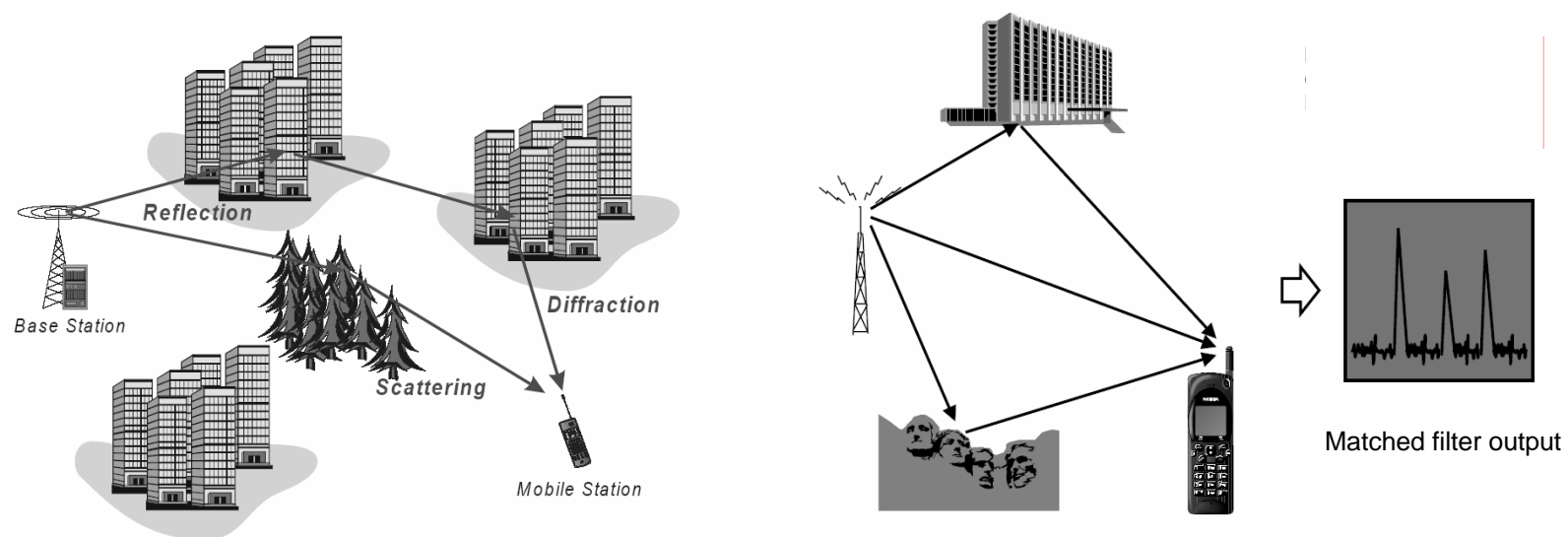


WIRELESS COMMUNICATION SYSTEM



Functional Block Diagram of a Wireless Communication System

MULTIPATH AND FADING



- ◆ **Multipath Propagation**
 - ◆ Multiple reflections, diffractions and attenuation caused by obstacles.
 - ◆ Multiple copies of the signal arrive at different time instants (delays).
- ◆ **Fading**
 - ◆ Multipaths are subjected to fast fading caused by signal cancellation when receiver moves across small distances.

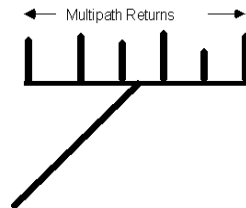
RAKE RECEIVER

- ◆ RAKE Receiver Definition

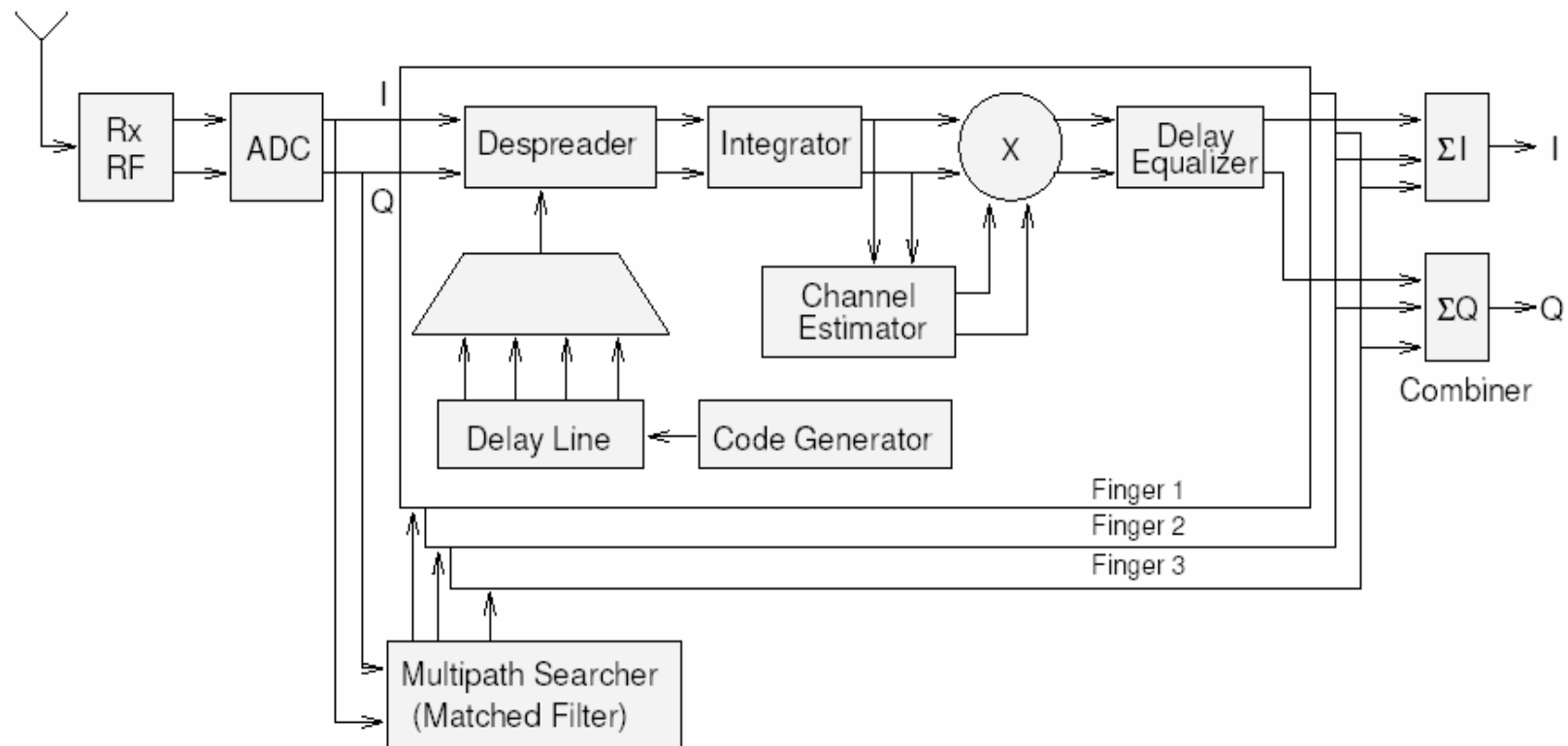
A receiver technique which uses several baseband correlators to individually process several signal multipath components. The correlator outputs are combined to achieve improved communications reliability and performance. (Source: CDMAonline.com)

- ◆ First suggested by Price and Green in 1958.

- ◆ Multiple delays appear at the receiver. By attaching a "handle" to the plot of multipath, a picture of an ordinary garden rake is created and it gets its name.



RAKE RECEIVER BLOCK DIAGRAM



Conventional Architecture

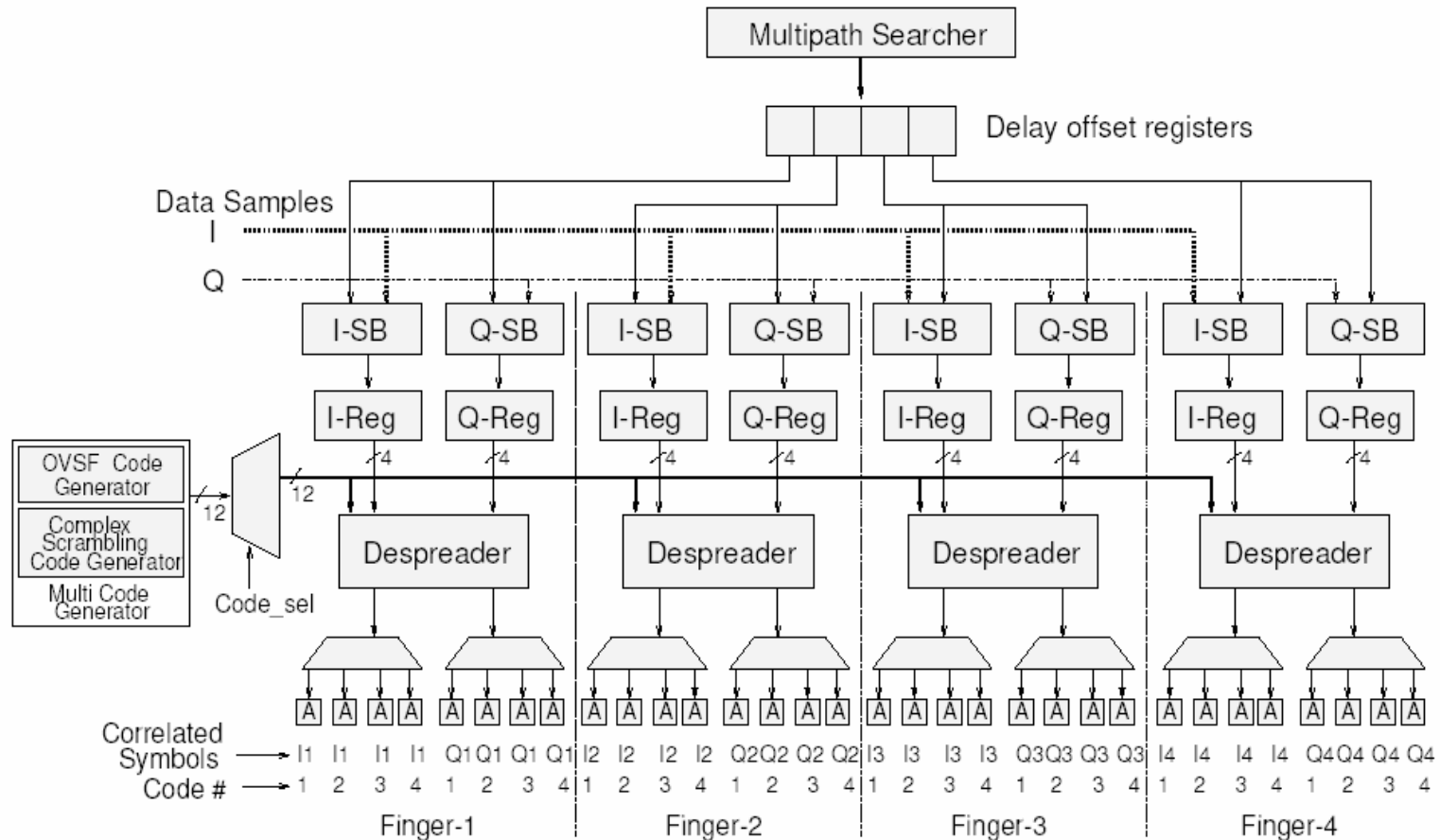
W-CDMA

- ◆ Wide band - Code Division Multiple Access (W-CDMA) is 3G air interface based on DS-SS-SSMA.
- ◆ High Data rates and multiple services on the same connection.
- ◆ Global acceptance of **65%** and beats competing standard cdma2000
- ◆ Wider Bandwidth and Increased Capacity – Carrier Bandwidth of 5MHz and Chip rate of 3.84 Mega chips per second (Mcps).
- ◆ Bandwidth on Demand – Variable user data rates
 - ◆ **Orthogonal Variable Spreading Factor (OVSF) Codes –**
Data rate = $f_c / (SF)$, where f_c = Chip rate, SF = Spreading factor
SF is variable from 4 to 512
 - ◆ **Multiple codes per user – Parallel Channels**
- ◆ Channelization code for channel separation
- ◆ Cell specific scrambling code for identifying base station uniquely.

IMPLEMENTATION ISSUES

- ◆ Rake algorithm is simple but computational complexity increases with number of multipaths.
- ◆ Multiple code feature of W-CDMA increases the complexity by number of codes used.
- ◆ Algorithm is repeatedly performed for detecting each data symbol, hence power consumption is important.
- ◆ MIMO systems employ multiple transmit and receive antennae.
 - ◆ **The processing requirement increases linearly with product of number of receiver antennae and number of multipath components.**

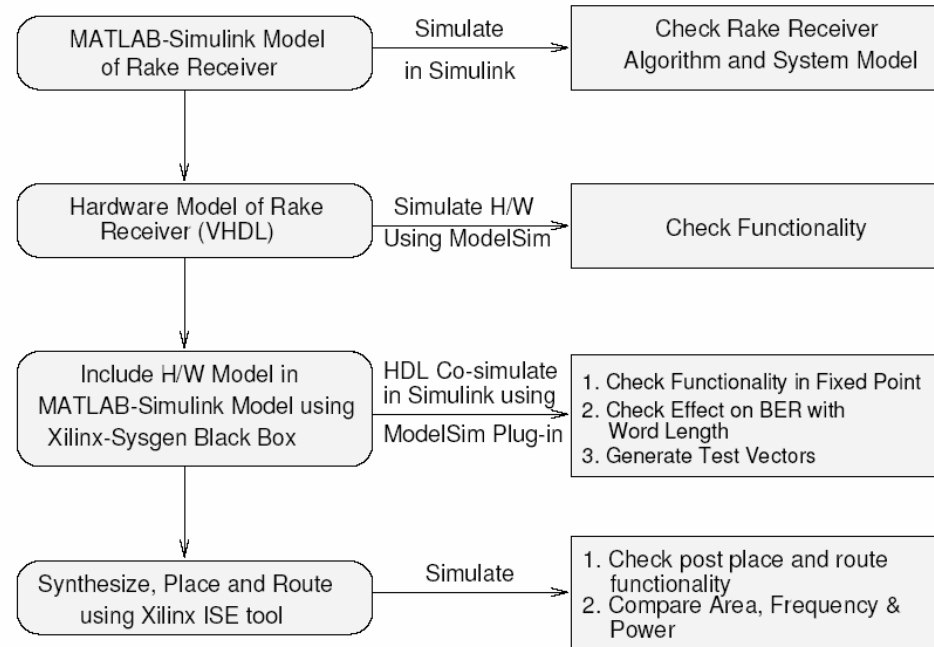
ARCHITECTURE IV - TIME MULTIPLEXED PARALLEL RAKE



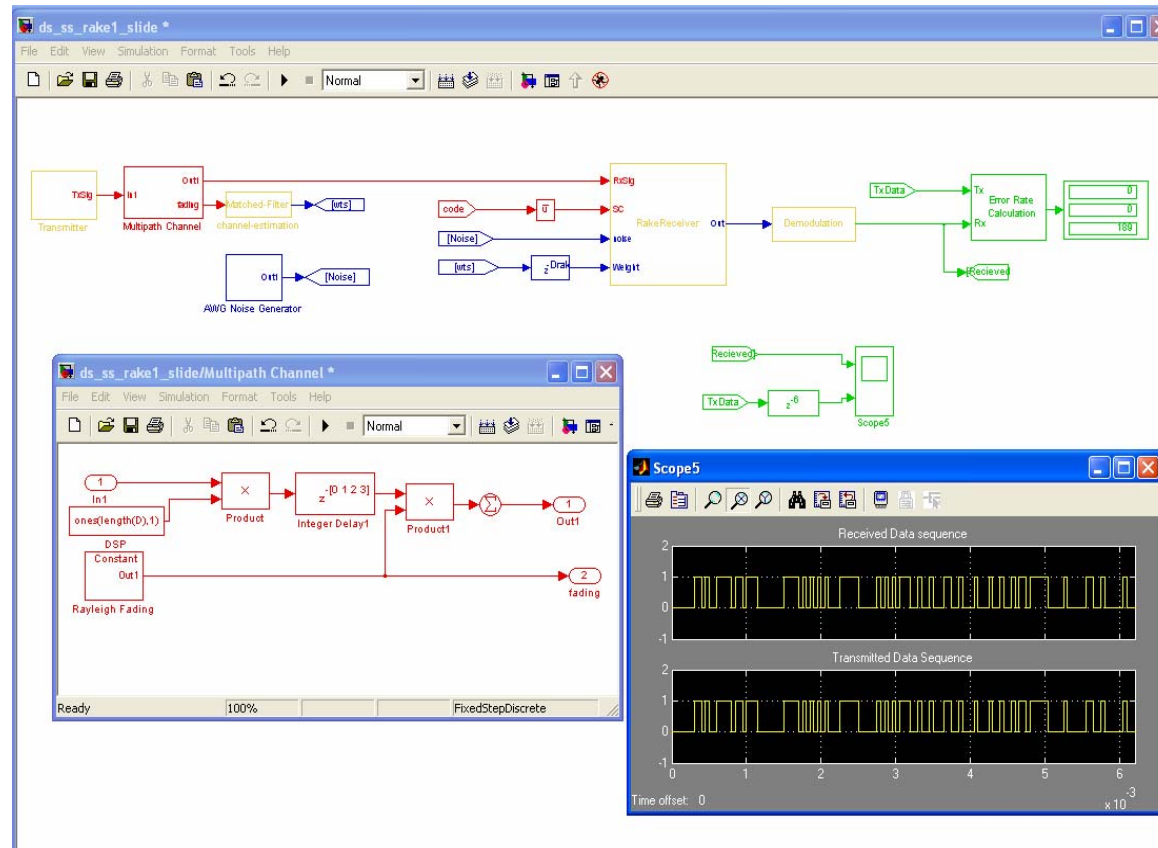
Example: number of multipaths, $L = 4$ & number of codes, $C = 4$

Legend: SB - Sample Buffer, Reg – Register, A - Accumulator

DESIGN FLOW

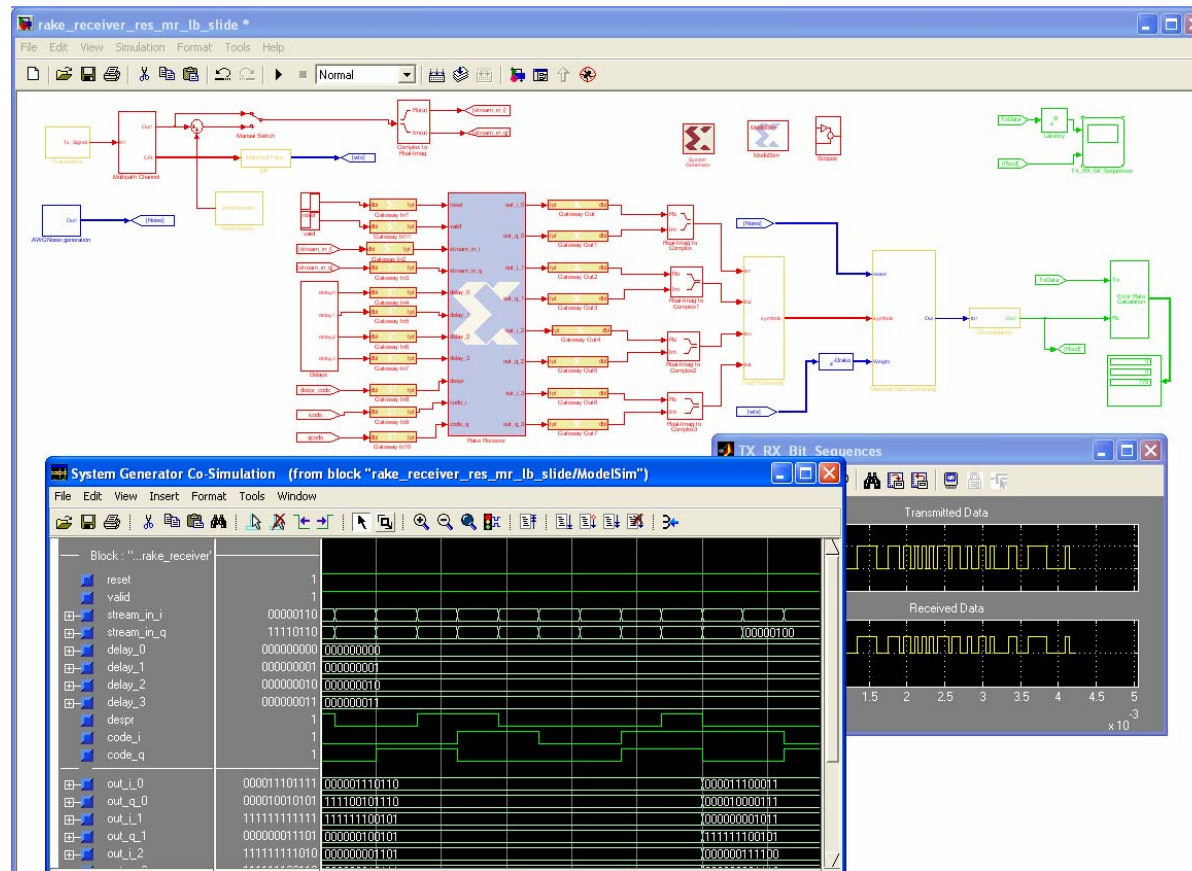


MATLAB-SIMULINK MODEL



- ◆ Using Simulink Blocks, software model is developed. Transmitter, Channel and receiver blocks are developed with smaller Simulink Blocks in hierarchical manner.
- ◆ Bit Error rates are checked with various configurations of decimal point and interference model

HDL Co-Simulation



- ◆ Simulink block replaced with X-Sysgen Black-Box that encapsulates VHDL Code of rake receiver
- ◆ Using HDL –Co Simulation, algorithmic correctness is verified.

IMPLEMENTATION DETAILS

Multipath Diversity	4 Components
Number of Codes	2
Spreading factor (SF)	16
<i>I/Q</i> Data Word size	8 bits
FPGA Device	Xilinx Virtex II xc2v1500

- ◆ All the four architectures are modeled in VHDL and implemented on Xilinx Virtex II FPGA.
- ◆ Low Spreading factor of 16 is chosen as it signifies high data rates.
- ◆ Data word size of 8 bits is chosen based on experimentation in MATLAB Sysgen test bed for BER variation with word lengths.

RESULTS

	Slices	FFs	BRAMs	Frequency
Arch I	<i>12%</i>	<i>10%</i>	<i>33%</i>	<i>107.10 MHz</i>
Arch II	<i>8%</i>	<i>6%</i>	<i>16%</i>	<i>159.89 MHz</i>
Arch III	<i>48%</i>	<i>38%</i>	<i>33%</i>	<i>134.28 MHz</i>
Arch IV	<i>30%</i>	<i>23%</i>	<i>16%</i>	<i>152.55 MHz</i>

- ◆ Area requirement for Architecture I corresponds to single chip finger implementation. Architecture III uses ~ 4x Area as it has multi-chip fingers.
- ◆ Architecture IV has half the desreader area requirement of Architecture III since hardware is not replicated for two codes. It needs some additional area for accumulators for each multipath per code.
- ◆ Architecture II uses least area corresponding to 2 fingers.

FUTURE WORK

- ◆ **Power Estimation**
 - ◆ Comparison of dynamic power consumption for different architectures using Xilinx XPower after the bug is fixed by vendor.
- ◆ **ASIC Implementation**
 - ◆ Complete W-CDMA rake receiver design based on Time multiplexed Parallel Architecture.
 - ◆ All the modules will be designed and coded in VHDL
- ◆ **Rake Receiver for MIMO systems**
 - ◆ Implementation of Rake Receiver for multiple receive antennae systems and show benefit of new architecture over previous architectures.

