

Reconfigurable Computing

Kees A. Vissers
CTO

Chameleon Systems, Inc.
kees@cmln.com

Research Fellow, UC Berkeley
vissers@eecs.berkeley.edu
www.eecs.berkeley.edu/~vissers

MP-SOC, July, 2002



Silicon, 20 years perspective

- Consumer Electronics, embedded systems, e.g. TV, dig. camera
- High performance: Base stations, Network processing

Silicon 10 years ago:

Philips VSP1 video signal processor:

1.2 micron CMOS, 90mm², 206.000 transistors, 27MHZ

Custom design, 3ALUs, crossbar and memory, 1W

Silicon this year:

Trimedia TM32 core:

0.13 micron CMOS(LV), 10mm², 1Million gates, 350MHZ

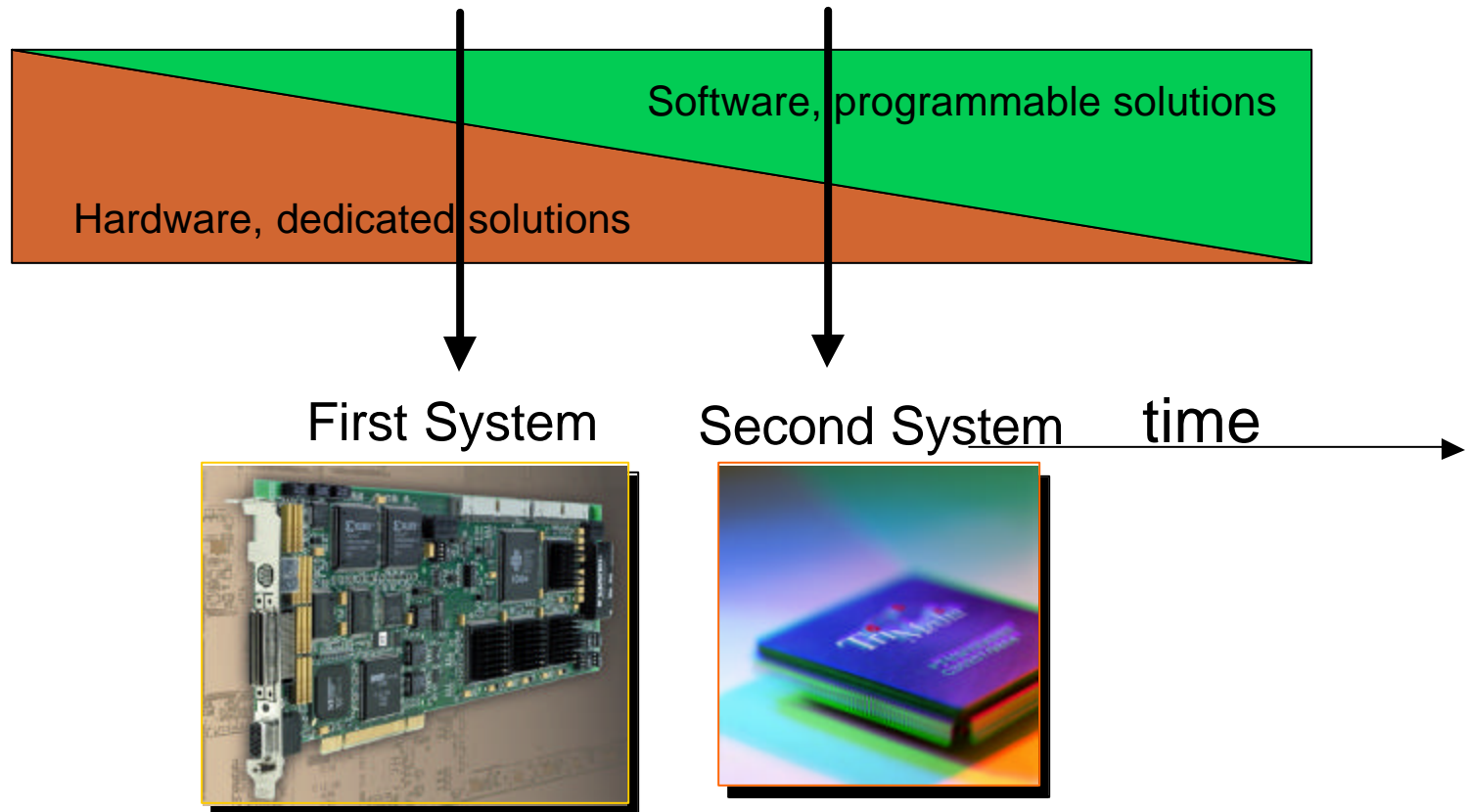
27 Functional units, fully synthesized, 5 issue VLIW, ~350mW

Silicon in 10 years:

0.01 micron CMOS, 10Million gates/mm², 3GHz?, package~4W

System on a chip

Silicon Technology is providing the opportunity to add **new functionality** and **integrate several functions** and **allow more programmable systems**.





Problem definition

- Perform processing on a stream of data
- Sampling rates in the order of 100KHz to 100MHz
- per sample 1000 – 100,000 operations

Perform 10^{10} - 10^{11} operations/sec, like add, multiply etc.

Take a 200MHz Alu, still need 50-500 of them,

Solution: Time multiplex and multi-processor



Revisit processor architectures

- Single Risc like processor
- ILP processing: VLIW for DSP, superscalar for general purpose
- Very successful programming environment: compilers and OS.
- Multi processor: no clear winning model, suggested to move to chapter 11 in Computer Architecture, a quantitative approach from Hennessy and Patterson
- Vector processing



Revisiting the silicon cost picture

Synthesize a DLX like ALU with Synopsis Design Compiler for TSMC 0.13micron LV (core voltage 1.0V) process:

in the range of 0.02mm² with a latency in the range of 2nsec ~ 500MHz

Add registers, forward path!, instruction decoding, and caches: pipelined processor core: 200-500MHz, 1mm²

So what do we do with a 100mm² silicon area?

100 Risc cores?

bit oriented FPGA, byte oriented FPGA?

Network of alus?

How to program?

What are the solution options

Concurrently execute 50- 500 operations every clock cycle.

- Multiple Risc cores, e.g. ARM, MIPS etc.
- Multiple VLIW oriented DSPs, e.g TI, Starcore etc
- Build a bit oriented FPGA and synthesize everything on top of that, including processors cores, packet routing networks etc.
- Build a fabric of interconnected ALUs (coarse grained FPGA)

SoC platforms exploiting the best part for the specific application (part).



Multi processor challenges

- Programming language problem: no concurrency
- Limited extraction of ILP out of sequential program

What is an instruction set processor

- C/C++, Java programming
- Program control translated to branches (most of the time)
 - for
 - if
 - case statements
- Single Program counter
- Data cache and Instruction cache
- Time-multiplex with instructions over ALUs
- Load, Store architecture, contains a Register File
- Debug with single stepping, breakpoints and register views



Multi-processors

- Multiple instruction set processors:
 - programmers model?
 - cache coherence?
 - granularity at the instruction level required
 - Instruction Level parallelism limited to 4-5
 - branch penalty in cycles
 - Operand routing and memory hierarchy are the cost
 - load-store instructions 30% of all instructions
 - L1 cache is half of the processor area
 - cache works poorly for stream oriented computing

Reconfigurable Computing (RC)

■ What is it?

- Compute by building a circuit rather than executing instructions.
- Efficient for long running computations
 - Video and image processing
 - DSP
 - Network processing

Example: $Z[i] = a.X[i] + b.Y[i]$

```
//program
```

```
Load rx, X
```

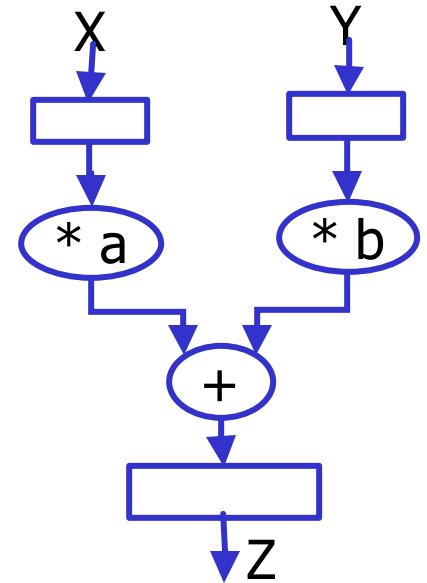
```
Mpy r1, rx, ra
```

```
Load ry, Y
```

```
Mpy r2, ry, rb
```

```
Add r3, r1, r2
```

```
Store r3, Z
```



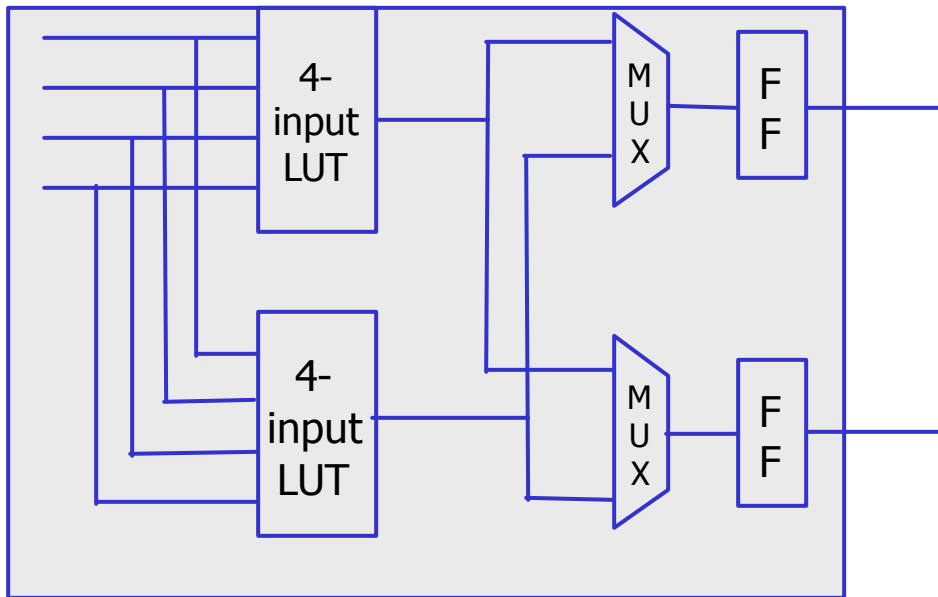


How to RC?

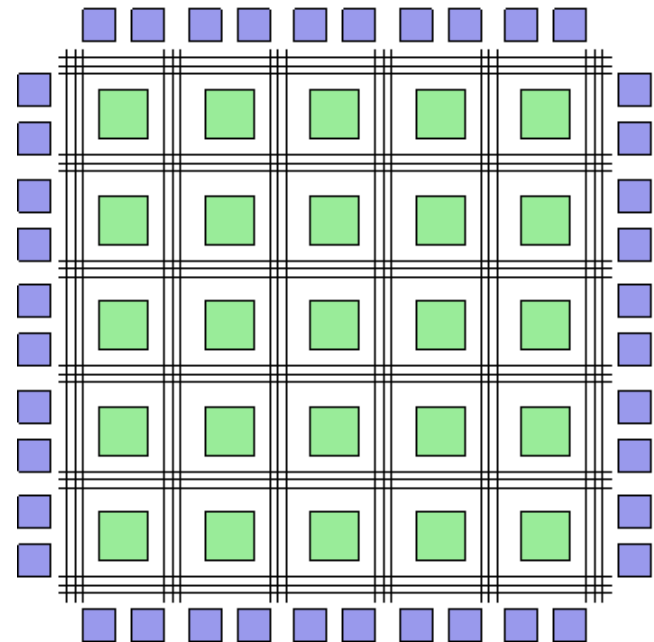
- FPGA-based RC
 - Programmable fabric that can be dynamically reconfigured
 - In the last 10 years the growth of FPGA speed and density has exceeded that of CPUs
- Mapping to FPGA
 - Only the time consuming computations are mapped
 - Computation expressed in HDL (VHDL/Verilog)
- Structure
 - FPGA + Memory on a peripheral board


Configurable Logic


- A.k.a. Field Programmable Gate Array, FPGA
 - Named in contrast to popular gate array semi-custom ASIC style that was popular at the time. More configurable than simpler programmable logic devices (PLDs) like programmable logic arrays (PLAs)
 - Logic blocks, consisting of lookup tables (LUTs), connected by programmable interconnect



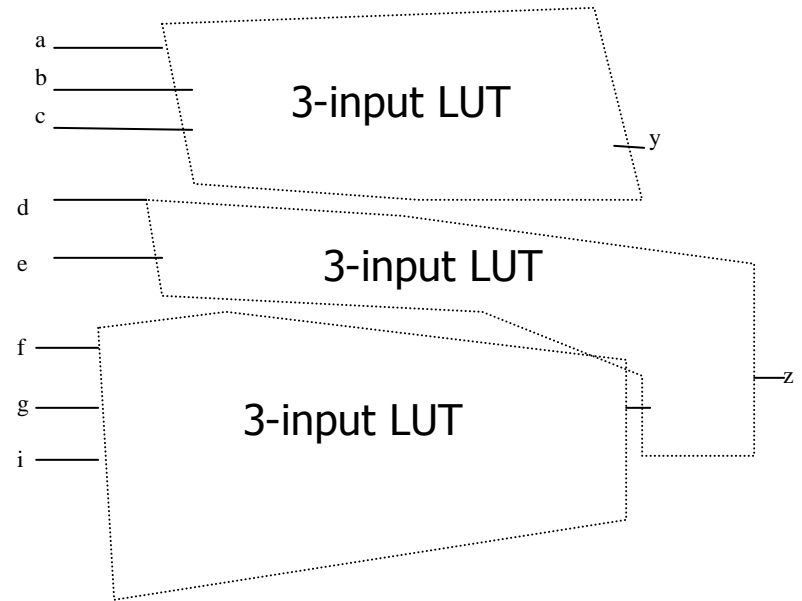
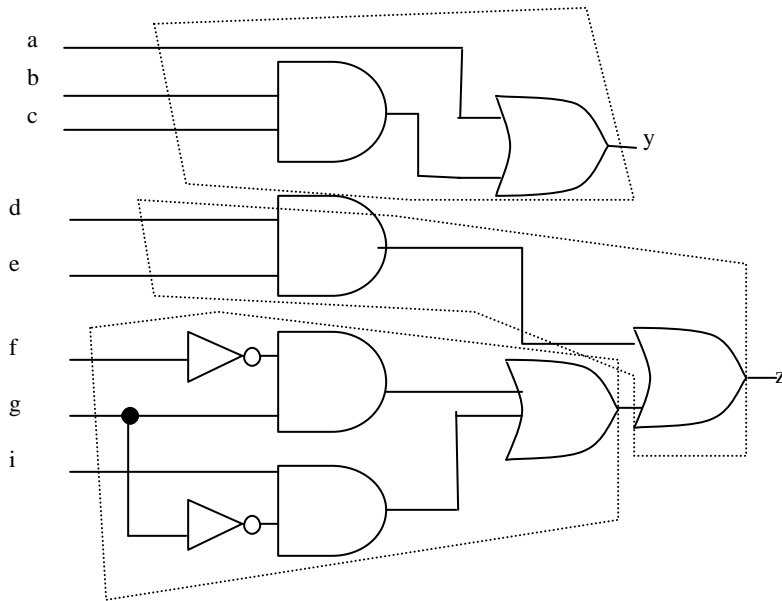
A very (!) simplified view of a logic block



 = Logic Block

 = I/O Block

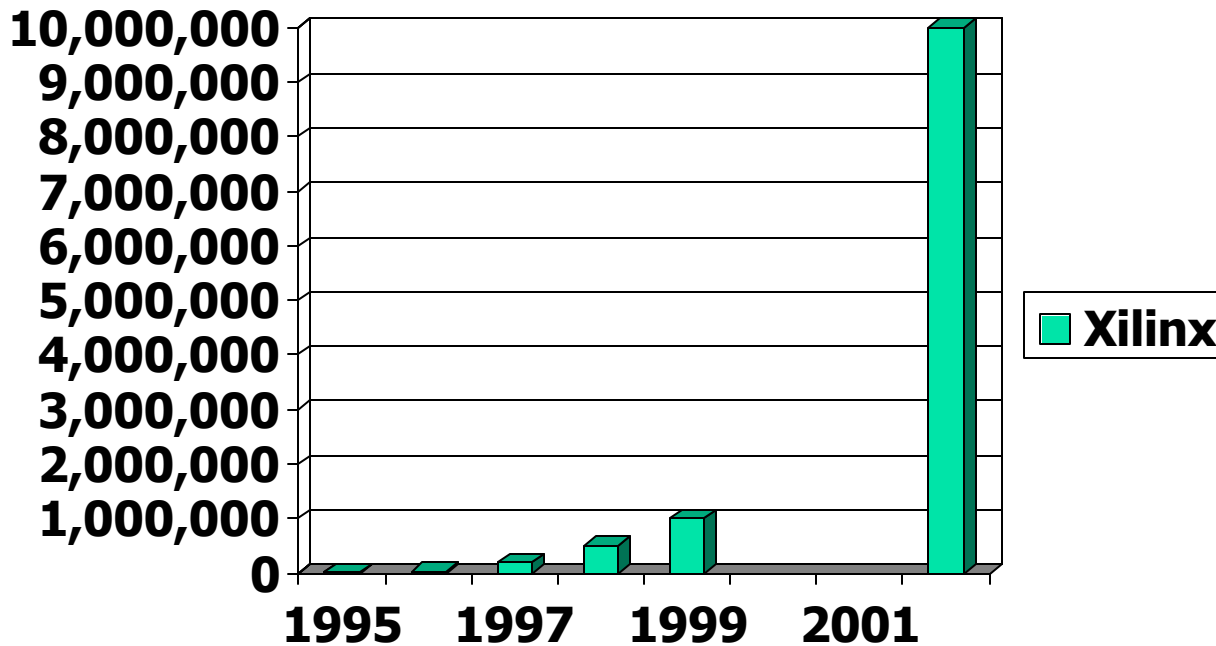
Simple Example Using Configurable Logic



- Three 3-input LUTs equal $8+8+8 = 24$ words
 - Compare with 8-input ROM having 256 words, or PLA with 16-input gates
- FPGA place-and-route tools partition the circuit among logic blocks and programmable interconnect

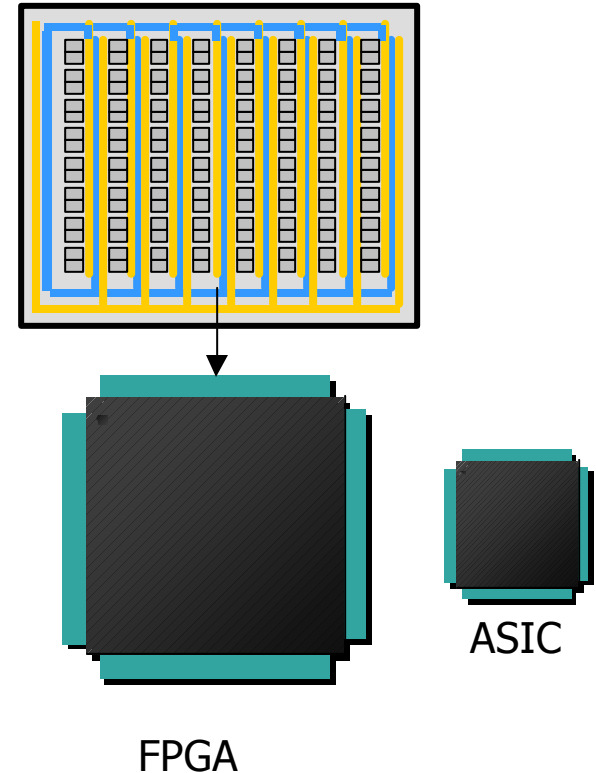
Configurable Logic Density Trend

- Following Moore's Law
- Currently near the era of 10 million gate FPGAs



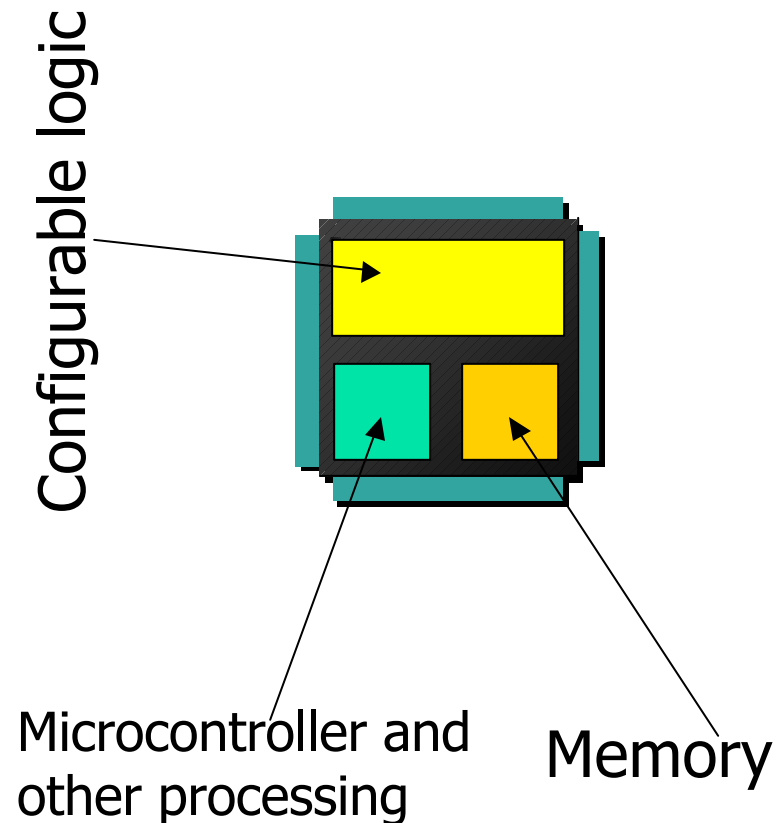
Configurable Logic vs. ASICs

- Roughly order of magnitude difference between FPGA and ASIC
 - ~100-200 MHz clock typical
 - Higher power
 - 10x size difference
 - Price of programmability



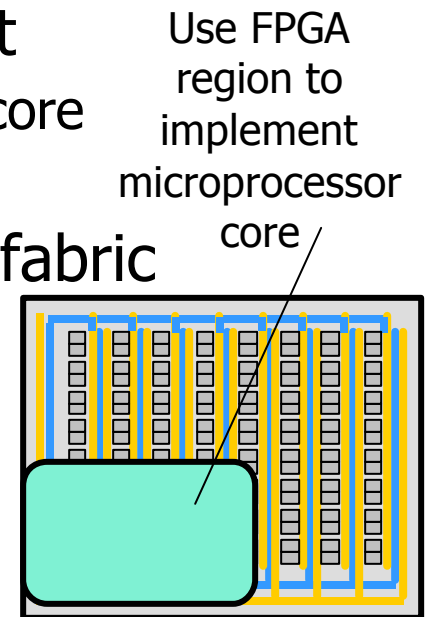
New Platforms Appearing with Microprocessors + Configurable Logic

- Several products incorporate microprocessor and FPGA on one chip
 - Soft core approach
 - Altera Nios
 - Xilinx MicroBlaze
 - Hard core approach
 - Atmel FPSLIC
 - Triscend E5
 - Triscend A7
 - Altera Excalibur
 - Xilinx Virtex II Pro



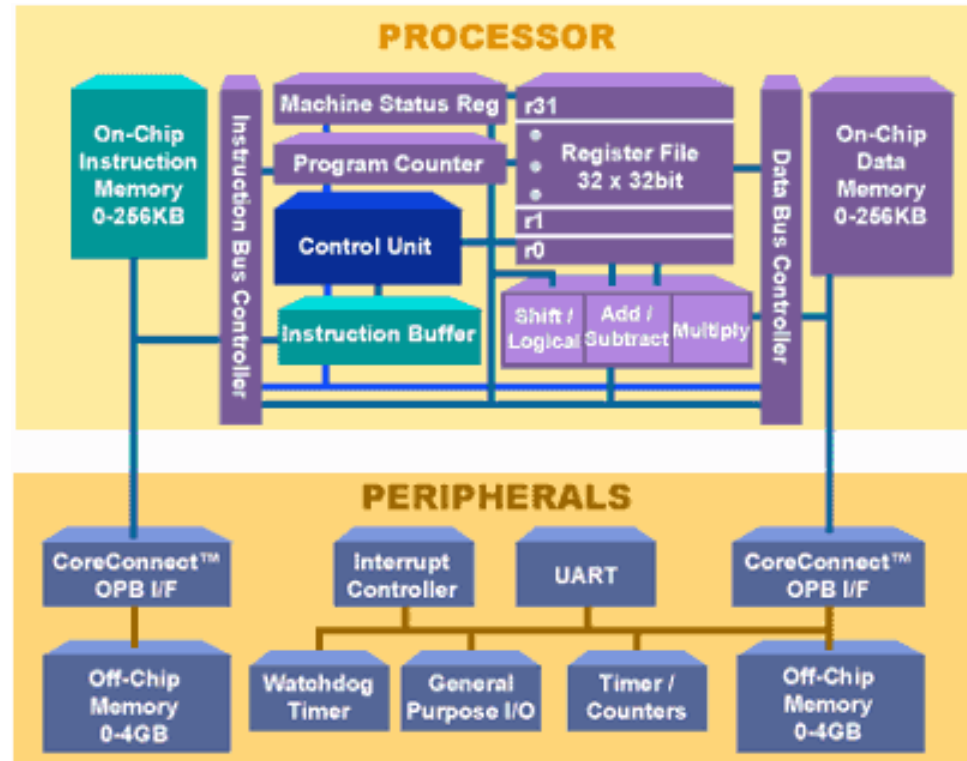
Soft Core Approach

- FPGA can implement almost any digital circuit
 - Use part of it to implement a microprocessor soft core
 - Can synthesize nearly any soft core to FPGA fabric
- Some vendors have soft cores tuned to their fabric
 - Altera Nios 2.0
 - 32 or 16 bit, 5-stage RISC, Regfile 128/256/512, optional multiply instrs., custom instrs.
 - Xilinx MicroBlaze
 - 32-bit RISC
 - Each run at ~100-150 MHz and execute ~100 Dhrystone MIPS
 - Sources: EE Times Oct 16 2001; www.xilinx.com
 - Extensively tuned for the FPGA fabric
 - More efficient than just synthesizing a microprocessor core and then running place and route
 - Typically obtained as VHDL/Verilog structural source



Xilinx MicroBlaze Soft Core Architecture

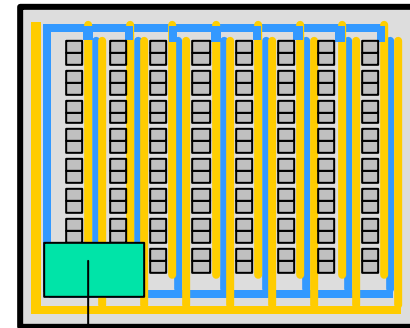
- Note the numerous integrated on-chip peripherals and the on-chip memories



Source: www.xilinx.com

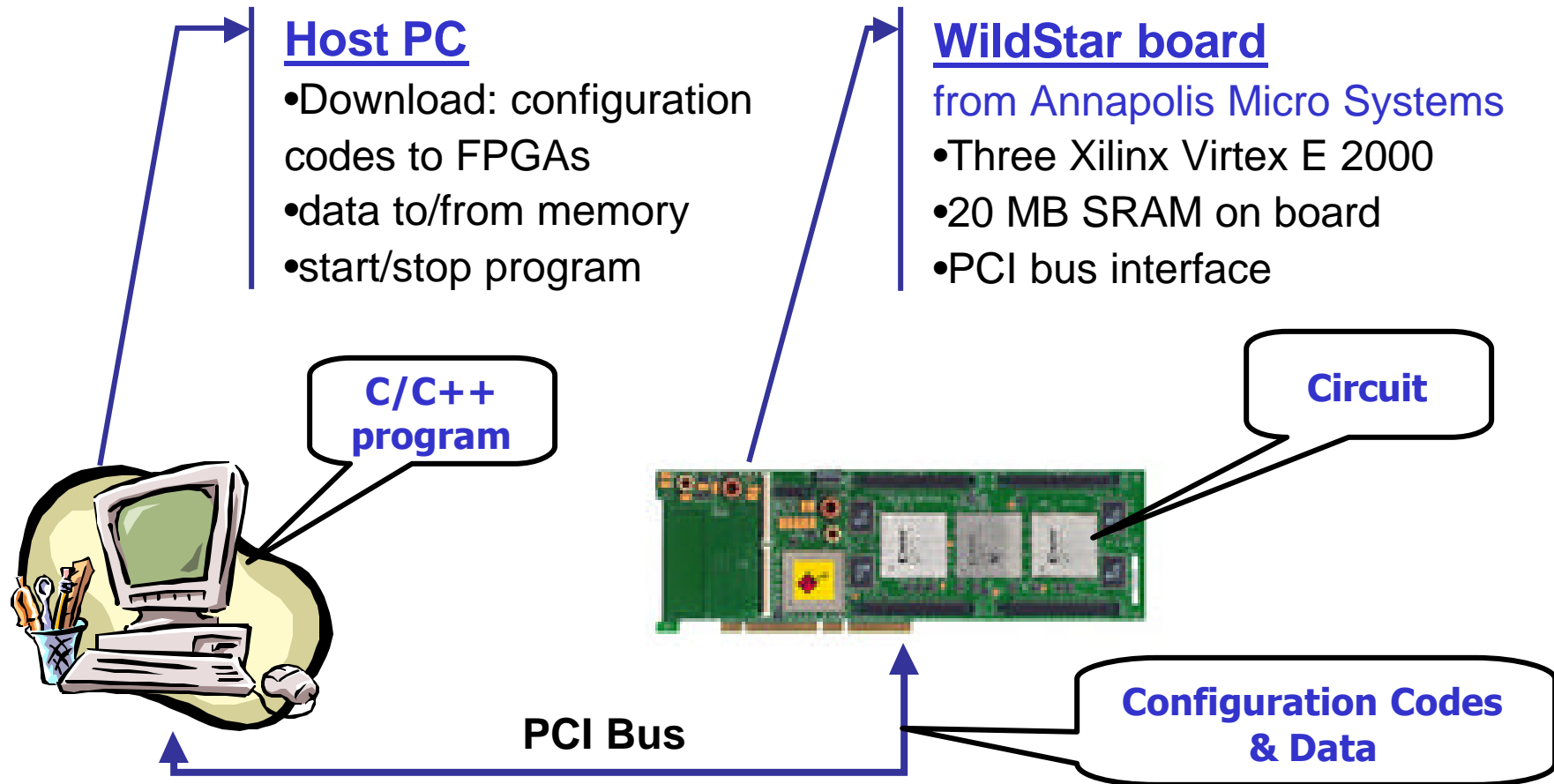
Hard Core Approach

- Recent devices include hard core, cache, RAM, and configurable logic
- Compared to soft cores
 - More area efficient
 - Leave more configurable logic for other uses
 - Faster (2-4 times)
 - Tradeoff: less flexible
 - Can't choose arbitrary number of cores



Replace
FPGA region
by uP hard
core

Hardware Set-Up

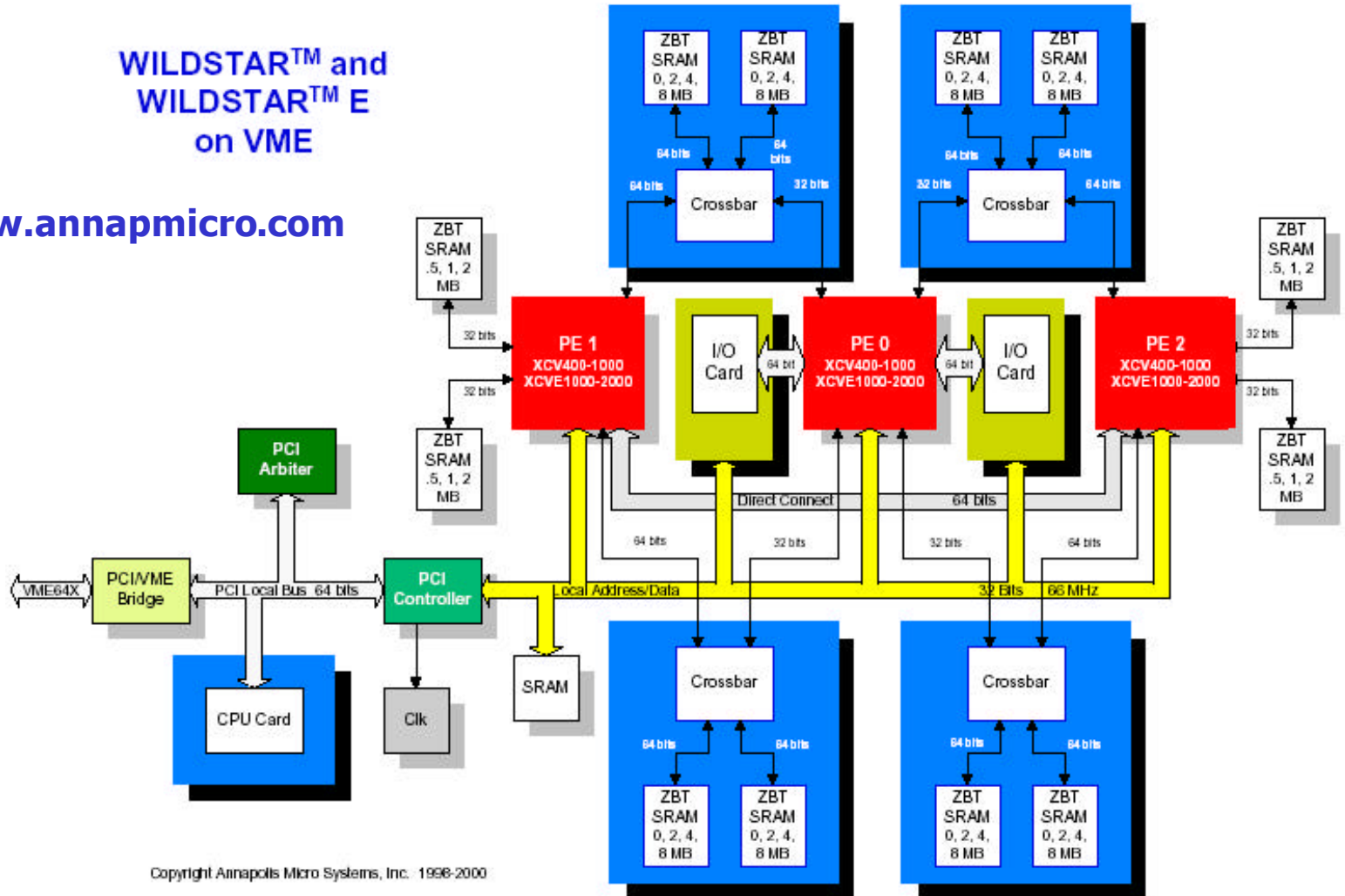


Source: <http://www.annapmicro.com>

WildStar Board Architecture

WILDSTAR™ and
WILDSTAR™ E
on VME

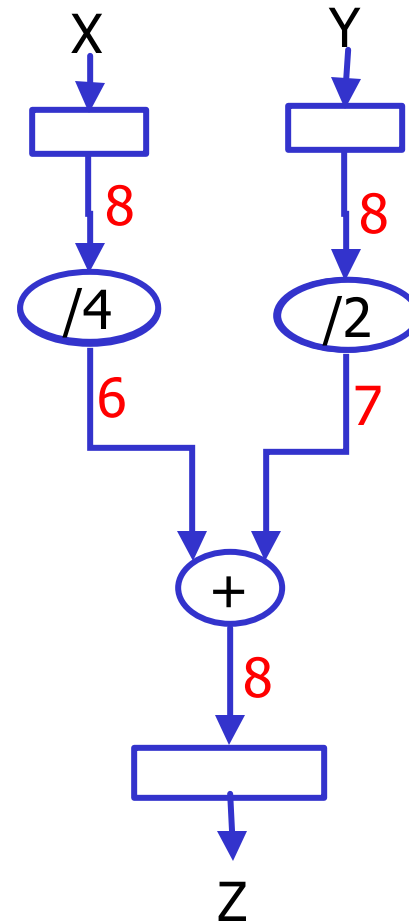
Source: www.annapmicro.com



Copyright Annapolis Micro Systems, Inc. 1998-2000

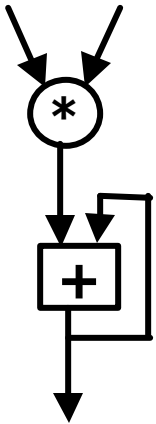
Advantages of RC (1)

- Program
 - No instruction fetch, no I-cache etc.
- Bit width and constants
 - Assume X & Y are 8 bits
 - Assume $a = 0.25$ and $b = 0.5$
 - Much smaller circuit!
- Delay
 - From two shift operations and one addition, all on 32-bits
 - To one 8-bit addition (shifts are free in hardware)

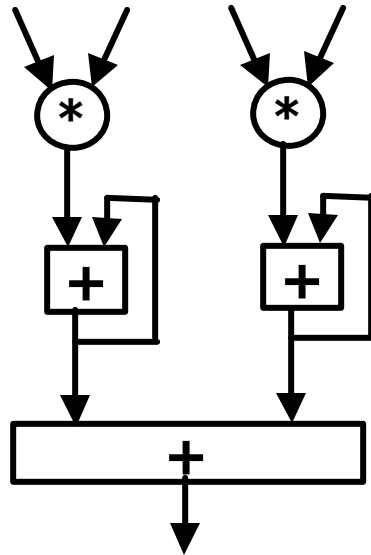


Key Advantage of RC: Parallelism

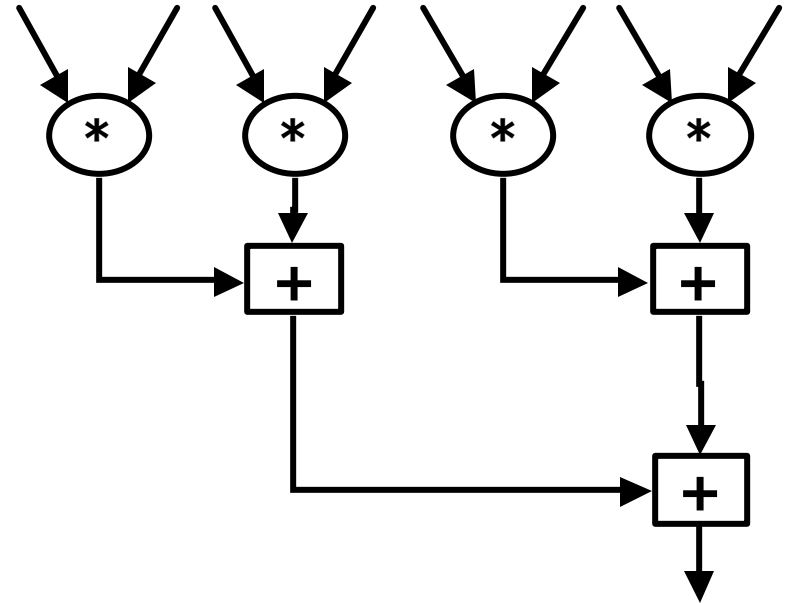
1 MAC
1 tap/cycle



2 MACs + 1 ALU
2 taps/cycle



RC fabric – custom circuit
K taps/cycle



■ Other "tricks"

- Use look-up tables (e.g. cos, sin, sqrt)
- Temporary storage (registers) configured as needed



Advantages of RC (2)

- On-chip parallelism in the custom circuit more than makes up for the lower clock rates on FPGAs
- Smart optimization at algorithm, program and circuit levels can
 - Reduce circuit size
 - Increase parallelism
 - Optimize data re-use (via on-chip storage)
 - Replace computations with table lookups



The Programmability Issue

- How to go from **algorithms** to **circuits**?
 - And integrate the circuit with a program
- Hardware description languages (VHDL/Verilog)
 - Efficient at circuit design
 - Do not provide integration with software
 - Are behavioral (not algorithmic) in nature (semantic)
 - Application developers are not familiar with HDLs



New Languages (1)

- Problems & challenges for C/C++ family
 - Express and/or extract (compile time) parallelism
 - Variable bit precision: 32 bit is overkill for most applications
 - Leverage traditional compiler optimizations
 - Increase productivity
- New languages: bridging the semantic gap
 - Handel-C: low level, timing aware design
 - StreamsC: hw & sw processes, explicit communication
 - SA-C: high level of abstraction
- Old language C: extract ILP out of loop nests

New Languages (2)

- Some common (??) features
 - Variable bit precision
 - Support for fixed point data
 - Parallel constructs
 - Pipelining
 - Allocation of code (circuit) to FPGA and arrays to memories

- Handel-C
 - More structural than behavioral
 - Explicit specification of
 - Delays, timing and synchronization
 - Parallelism
 - Communication channels (based on communicating sequential processes – CSP)
 - Compiles to a netlist

Source: www.celoxica.com

New Languages (3)

- Streams-C

- Using directives (///), user explicitly
 - Partitions program into hardware and software processes (HP & SP)
 - Set up explicit communication channels (also based on CSP)
- SP compiled to C++ and HP to VHDL
- No explicit low level synchronization

rcc.lanl.gov

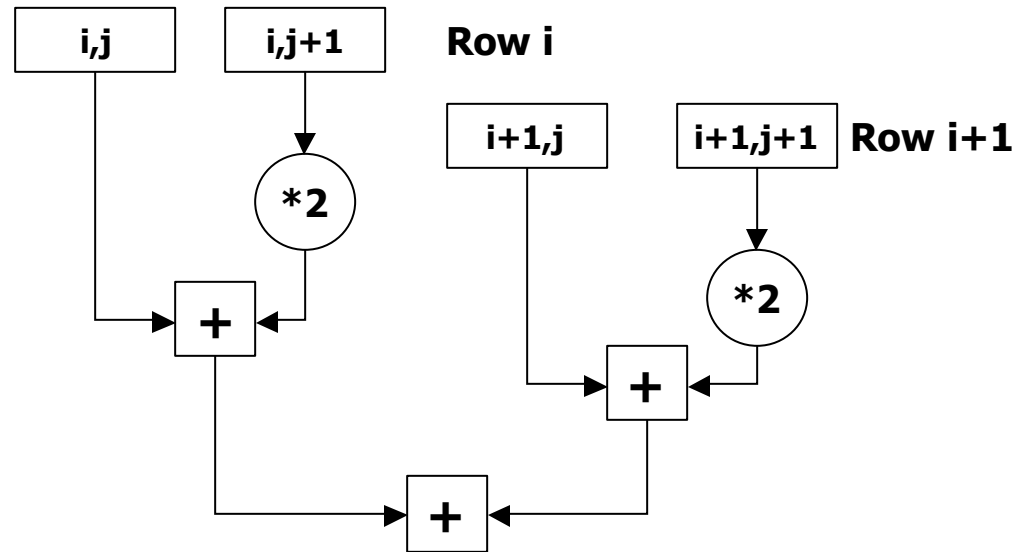
- Single Assignment C: SA-C

- Implicit parallelism
 - Single assignment (functional) semantics
 - Forall construct supports implicit parallel loop iterations
- Extensive compiler optimizations
 - Made easier by functional semantics
 - Focus on loop transformations

www.cs.colostate.edu/cameron

SA-C Example (1)

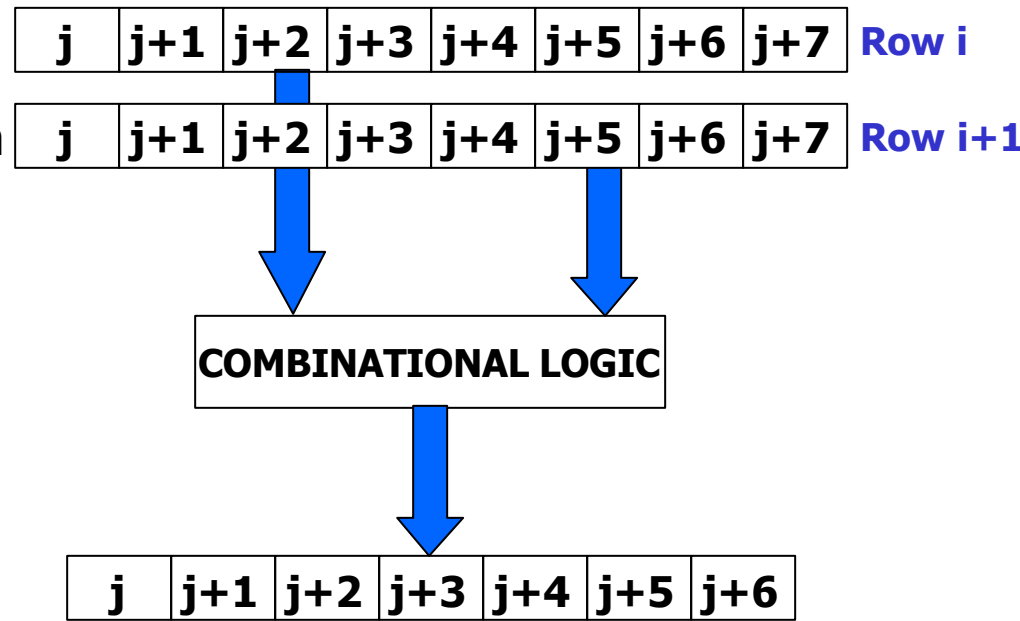
```
Kernel = {{1,2},{2,1}}
//convolution inner loop
result[:,:] =
  for window win[2,2] in Image
    {uint8 conv =
      for elem1 in win
        dot elem2 in Kernel
          return(sum(elem1*elem2));
        }
  return(array(conv));
```



Unrolled inner loop on [2,2] window

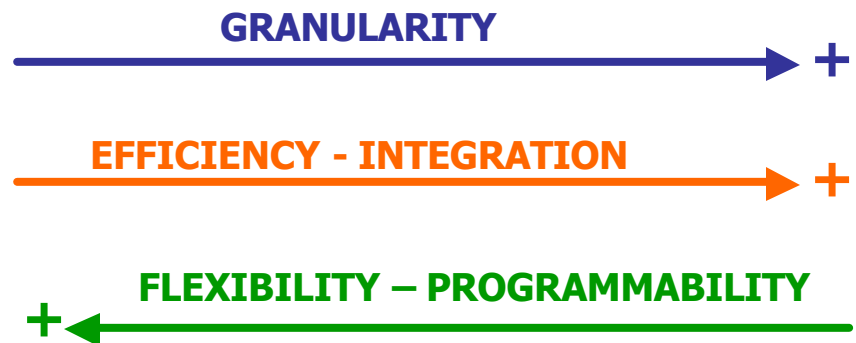
SA-C Example (2)

- Assume
 - 8-bit pixels
 - 64-bit word access/cycle from memory
- Outer loop can be unrolled to generate 7 values per iteration (strip-mining)



The Granularity Issue

- Are FPGAs too fine grained?
 - Therefore inefficient?
 - For some applications: DEFINITELY
 - For some other applications: ABSOLUTELY NOT
- Is there an optimal granularity level?
 - The 1 Mega\$ question
 - Probably NOT





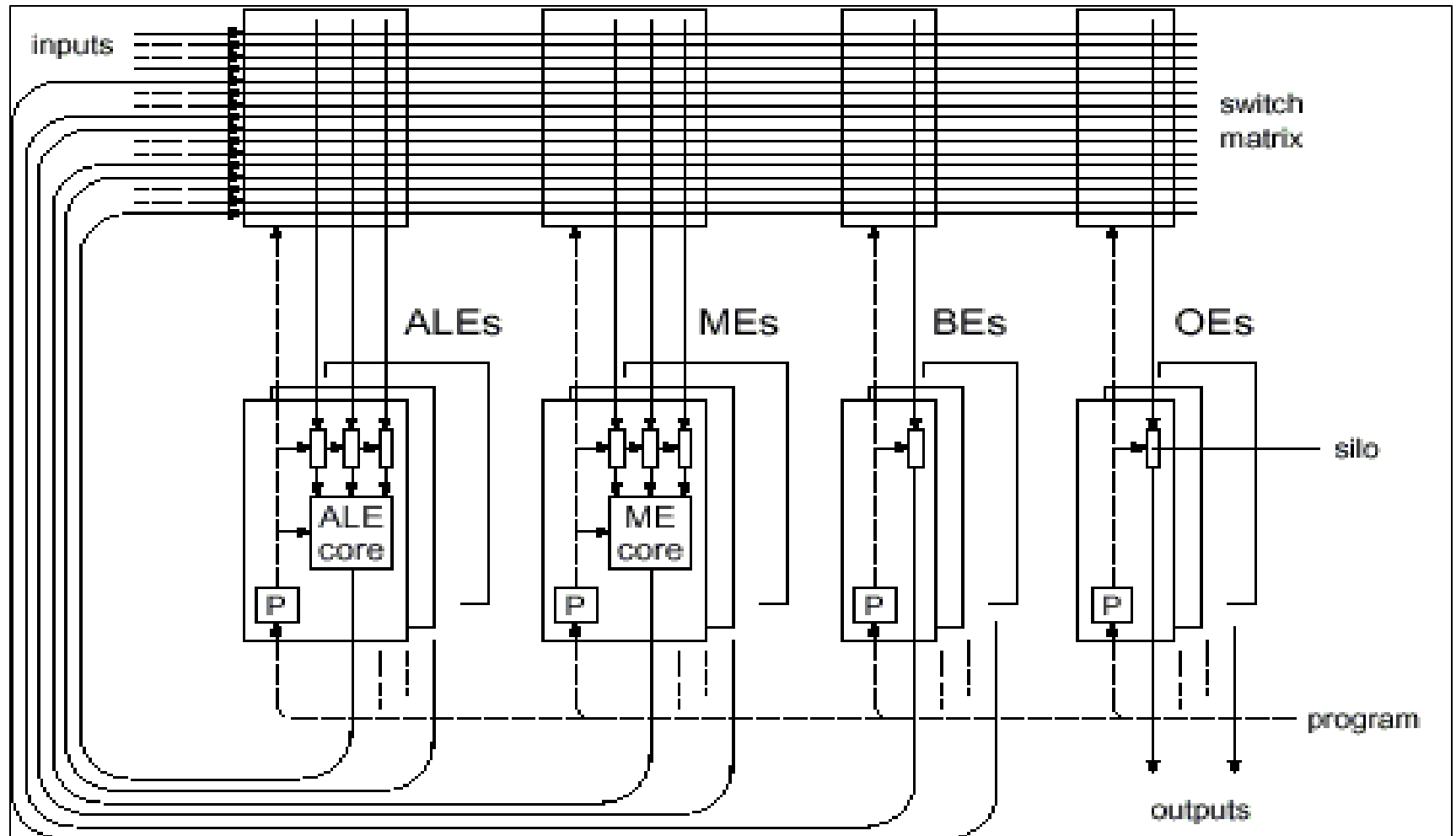
Pros & Cons of RC

- Advantages:
 - Higher computation density than CPUs (MIPS/area)
 - More flexible than ASICs: reconfigurable
 - Large and variable level of parallelism
- Where does an RCS fit?
 - Currently: attached processor (I/O bus: PCI, PC-card, etc)
 - Ideally: co-processor (on memory bus) or as a functional unit within a CPU (share registers)
- Problems:
 - FPGAs are programmed using Hardware Description Languages (HDLs): Verilog or VHDL
 - Applications programmers do not know (or want to know) HDLs
 - RCS are not accessible where they are needed!
 - Back to overlay programming for reconfiguration

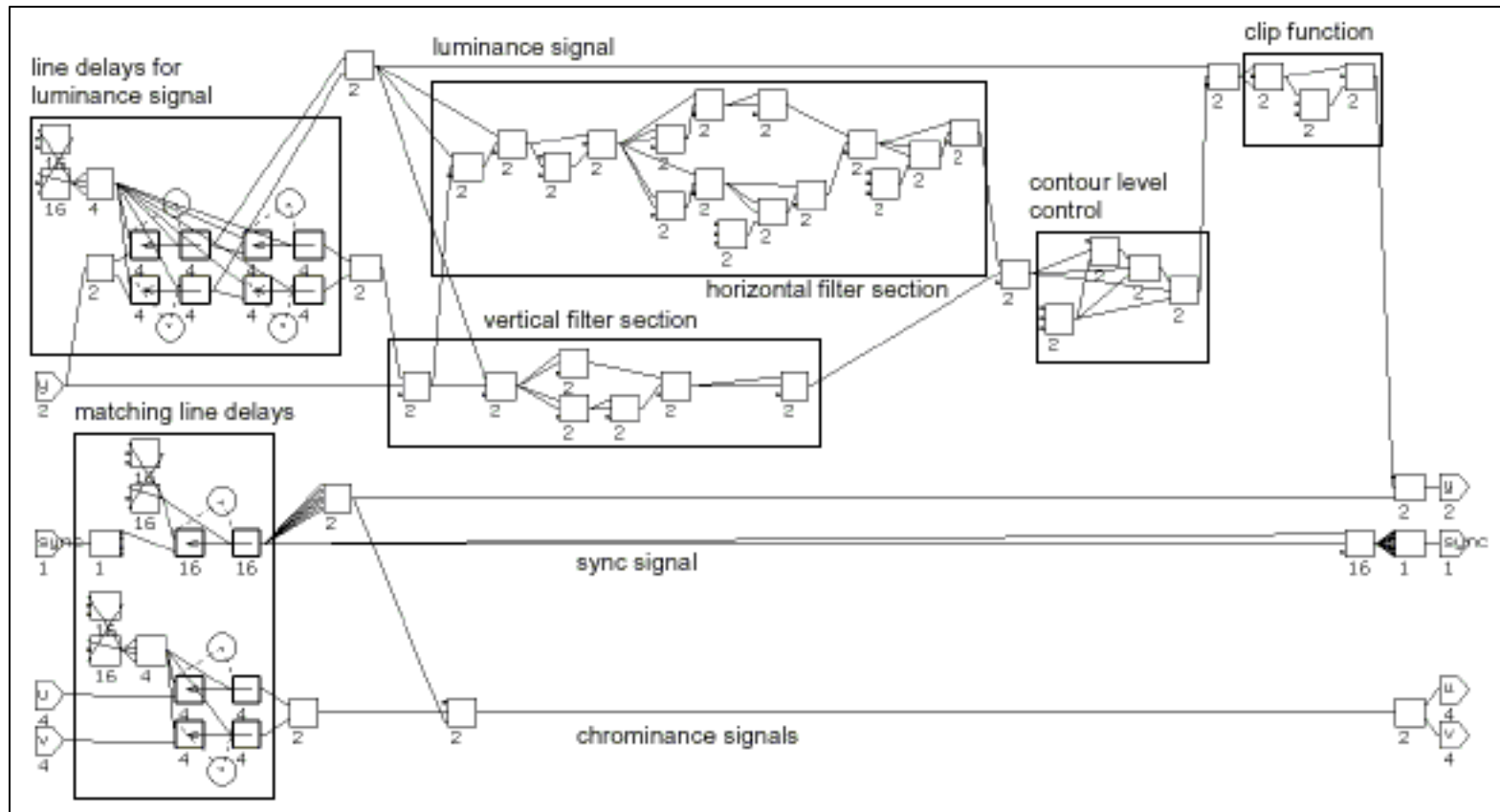
Coarse grain RC: Multiple ALUs connected

- Operand routing with a hierarchical connection network
 - locally full connectivity: crossbar
 - global connectivity limited
- registers are distributed
- configure once and then run -> no Icache
- Potentially an instruction level parallelism of 100 and more
- No branch instruction
- Programmers view:
 - SA-C
 - Signal Flow Graphs
 - Extracted parallelism from inner loops, e.g. Matlab, C loop extraction

Exampe VSP architecture



Example programming VSP with SFG

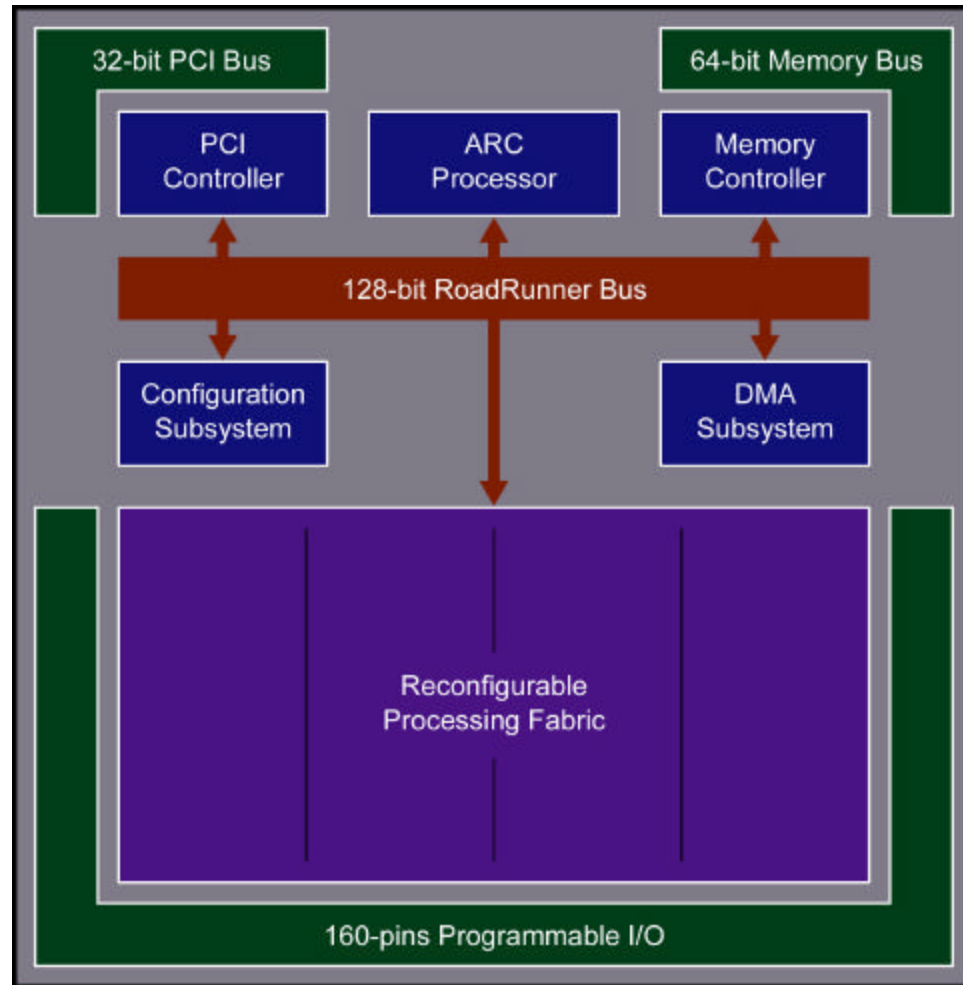




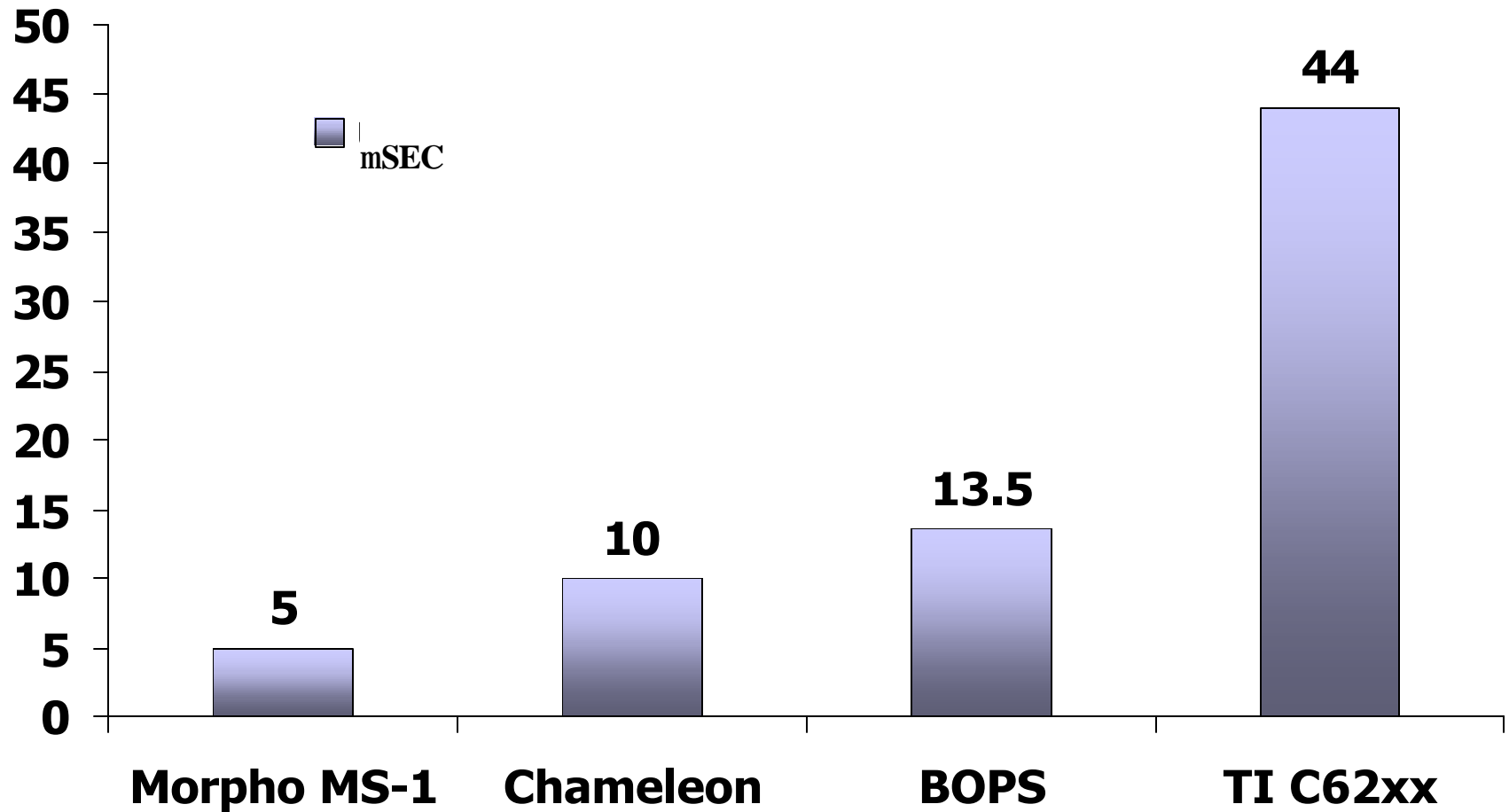
Examples

- Pleiades
- Garp
- Score
- VSP1, VSP2
- Chameleon first generation
- MorphoSys

Chameleon, first generation



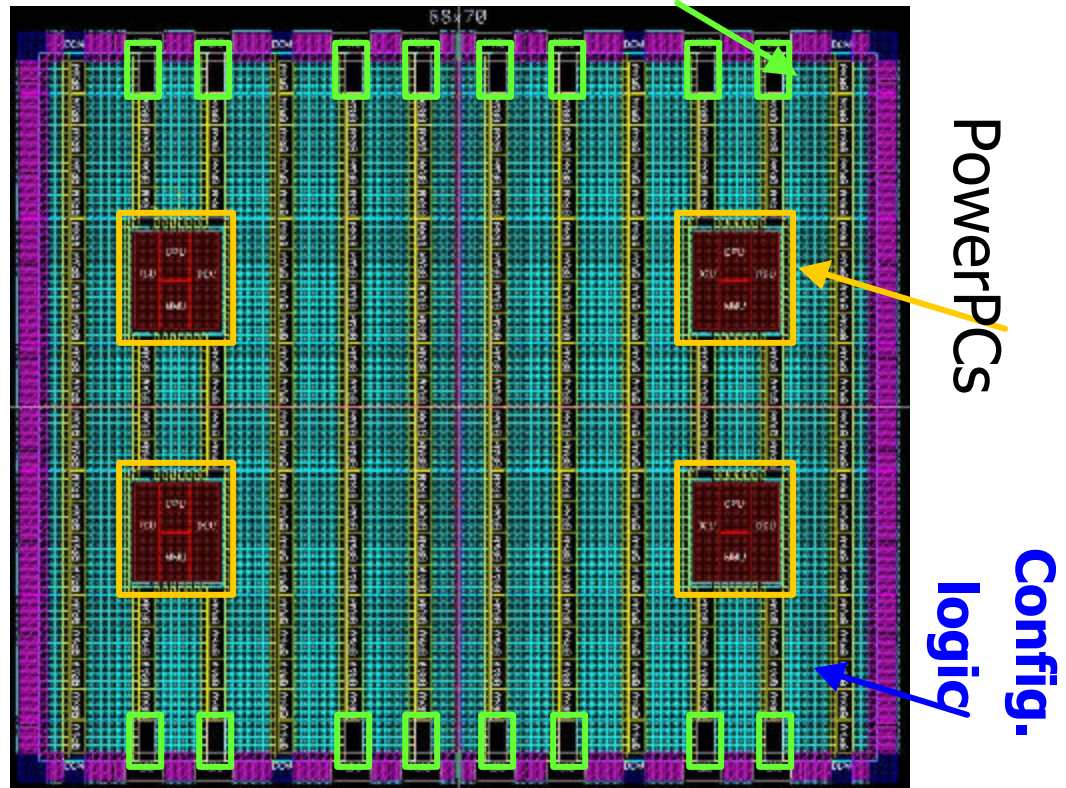
1024-Point 8-bit Complex FFT



Xilinx Virtex II Pro

- PowerPC based
 - 420 Dhrystone MIPS at 300 MHz
 - 1 to 4 PowerPCs
 - 4 to 16 gigabit transceivers
 - 12 to 216 multipliers
 - 3,000 to 50,000 logic cells
 - 200k to 4M bits RAM
 - 204 to 852 I/O
 - \$100-\$500 (>25,000 units)

Up to 16 serial transceivers
• 622 Mbps to 3.125 Gbps

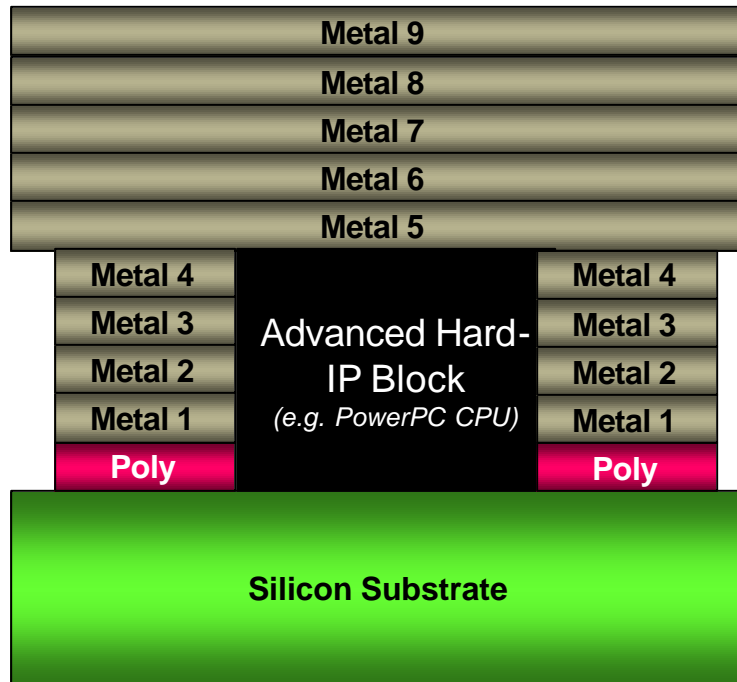


Courtesy of Xilinx

Virtex II Pro Approach to Embedding Microprocessor in the Configurable Logic

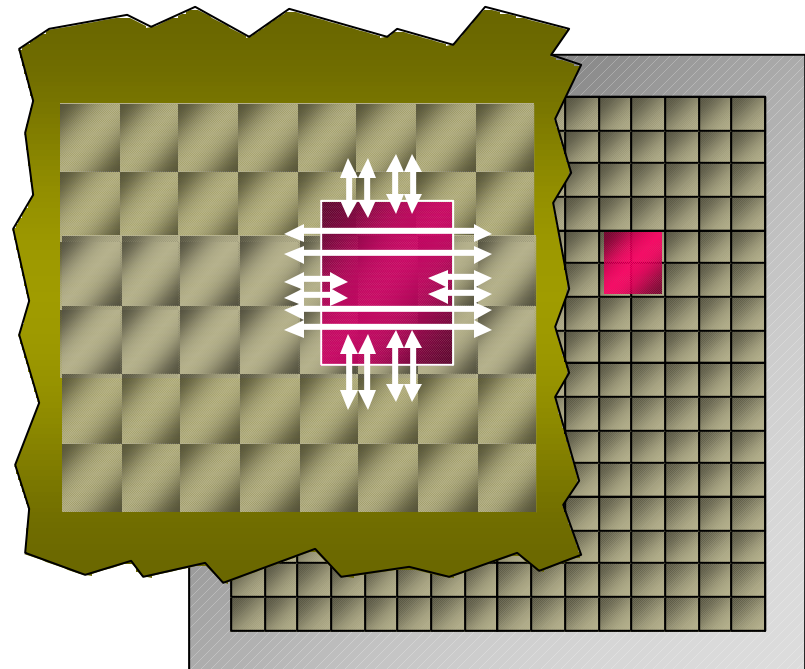
IP Immersion

Metal 'Headroom'
enables immersion



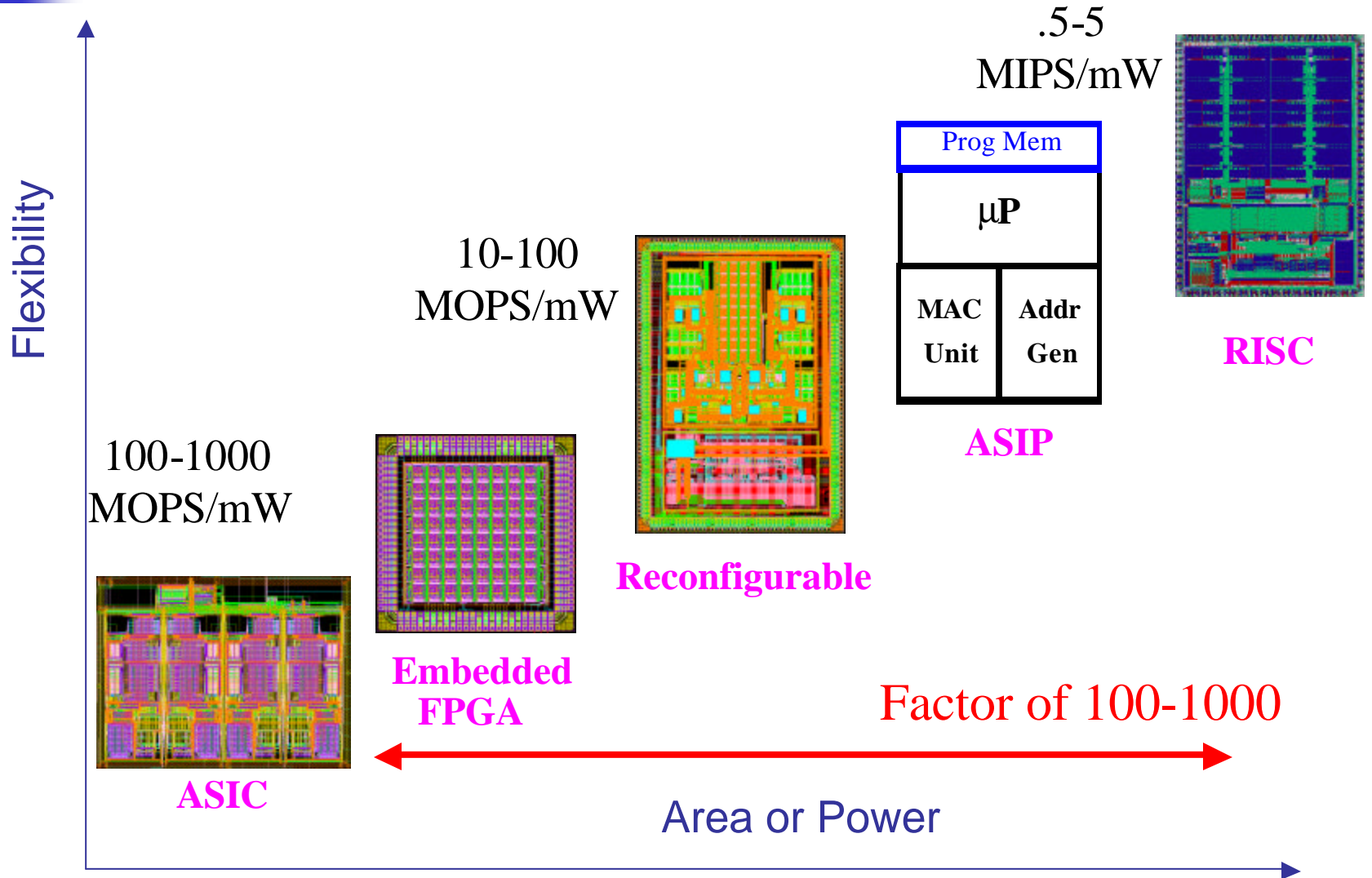
Active Interconnect

Segmented Routing
enables predictability



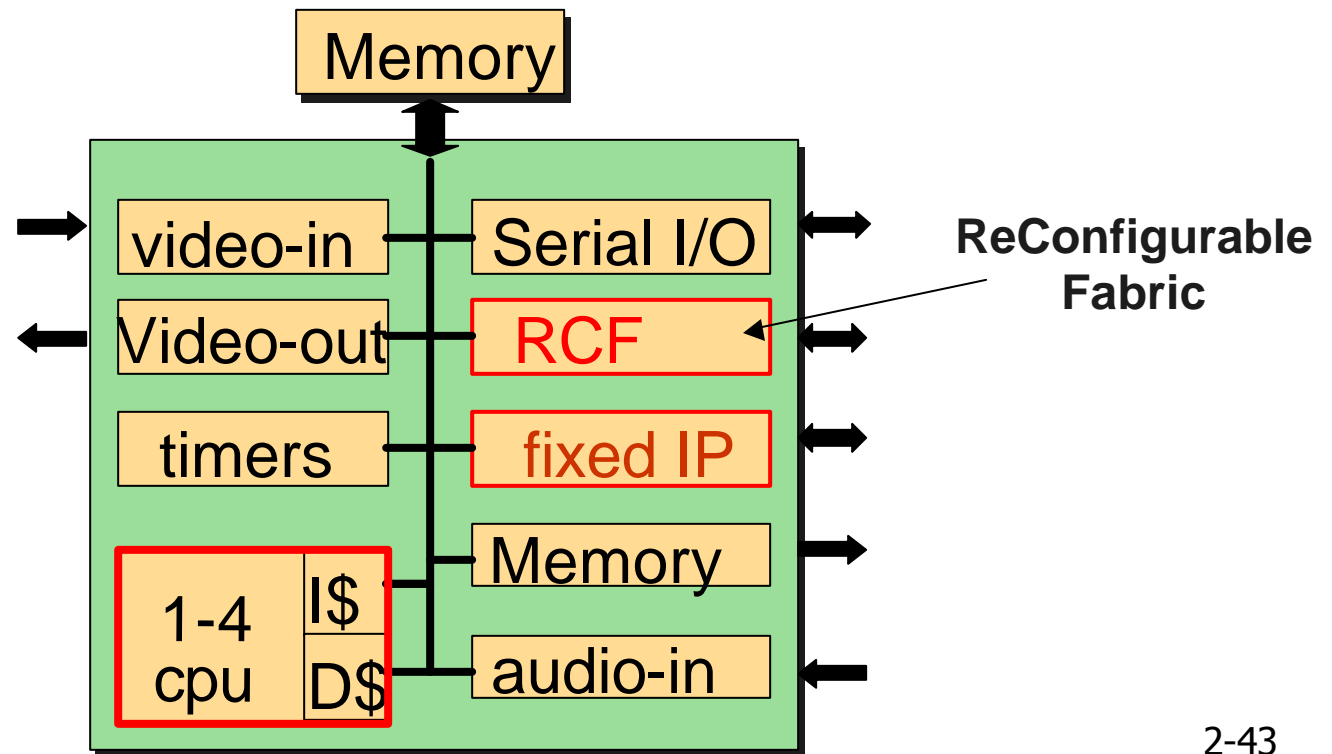
Slide courtesy of Xilinx

System Trade-offs



New Systems

- Understand the application!
- On chip memory
- Multi processor, programmable and reconfigurable
- Power consumption of the complete IC needs to be **constant**
- The **PROGRAMMERS** view is making the difference





Summary

Benefits of word oriented Reconfigurable:

- low power
- low overhead, little silicon
- ILP of 100 or more
- Programming model different

Disadvantages of word oriented Reconfigurable:

- Programming model different
- works for stream oriented problems, not for control dominated problems



Summary

- Reconfigurable computing has many advantages over ASIC and CPU/MPU
 - Large parallelism with no instruction overhead
 - Customizable data path size
 - Flexible (reconfigurable!)
- It is still in its infancy
 - Semantic gap between algorithms and circuits is still a major obstacle
 - Hardware platforms are only now emerging commercially that are designed for RC



Trends

- Very exciting time:
 - new tools
 - new architectures
- Reverse the world: silicon is cheap, concurrency and communication is the problem
- Multiprocessors and dealing with concurrency
- Embedded products!
- Programmable and reconfigurable is emerging.

Slides reference

- Special thanks to Frank Vahid, Walid Najjar, and Joerg Henkel co-Presenters of the tutorial at the DAC2002:
 1. **Introduction**
 2. Standard computer platforms
 3. Domain specific platforms
 4. Customizable processor platforms
 5. Microprocessor/ configurable-logic platforms
 6. Reconfigurable computing platforms
 7. Conclusions

Design Automation Conference, 2002, New Orleans, Tutorial On...

New Computing Platforms for Embedded Systems

Joerg Henkel
NEC USA Inc., Princeton, NJ, henkel@nec-lab.com

Walid Najjar
Assoc. Prof., Computer Science & Engineering, UC Riverside

Frank Vahid
Assoc. Prof., Computer Science & Engineering, UC Riverside
& Center For Embedded Computer Systems, UC Irvine

Kees Vissers
CTO Chameleon Systems;
Research Fellow, UC Berkeley (Mescal)

DAC'02 – New Computing Platforms For Embedded Systems
J. Henkel, W. Najjar, F. Vahid, K. Vissers



References

- <http://www.annapmicro.com/>
- <http://www.proceler.com/>
- <http://www.morphotech.com/>
- <http://www.chameleonsystems.com/>
- <http://www.celoxica.com/home.htm>
- <http://rcc.lanl.gov/>
- <http://www.cs.colostate.edu/cameron/>