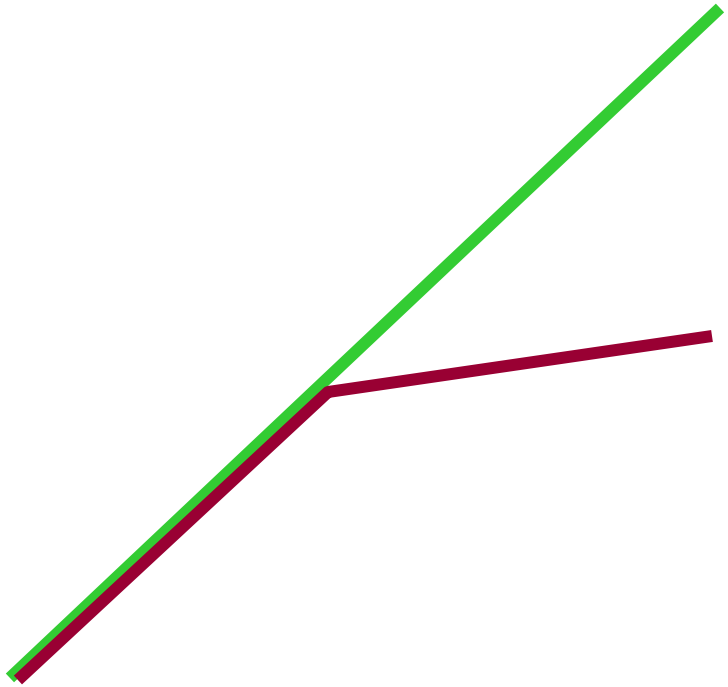
Tiled Multicore Processors:

The Four Stages of Reality

Anant Agarwal
MIT and Tilera

Moore's Gap



- Diminishing returns from sequential processor mechanisms
- Wire delays
- Power envelopes

Tilera's Tile Architecture

Multicore Performance (90nm)

Number of tiles	64
On chip distributed cache	5 MB
Operations @ 750MHz (32, 16, 8 bit)	144-192-384 BOPS
On chip peak interconnect bandwidth	32 Terabits per second
Bisection bandwidth	2 Terabits per second

Power Efficiency

Power per tile (depending on app)	170 – 300 mW
Core power for h.264 encode (64 tiles)	12W
Clock speed	600-1000 MHz

I/O and Memory Bandwidth

I/O bandwidth	40 Gbps
Main Memory bandwidth	200 Gbps

Programming

ANSI standard C
SMP Linux programming
Stream programming

Tile64 Processor

Product reality

2007

Virtual reality

Simulator reality

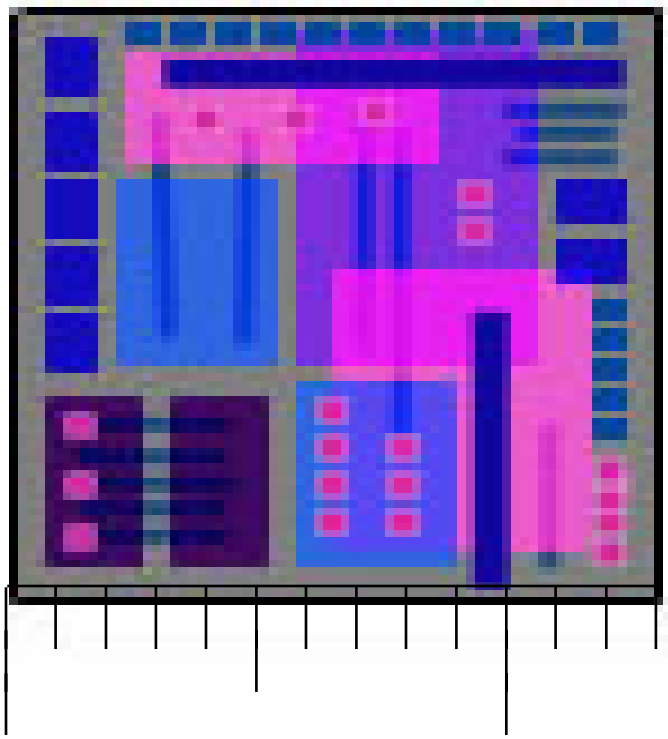
Prototype reality

Product reality

The Opportunity

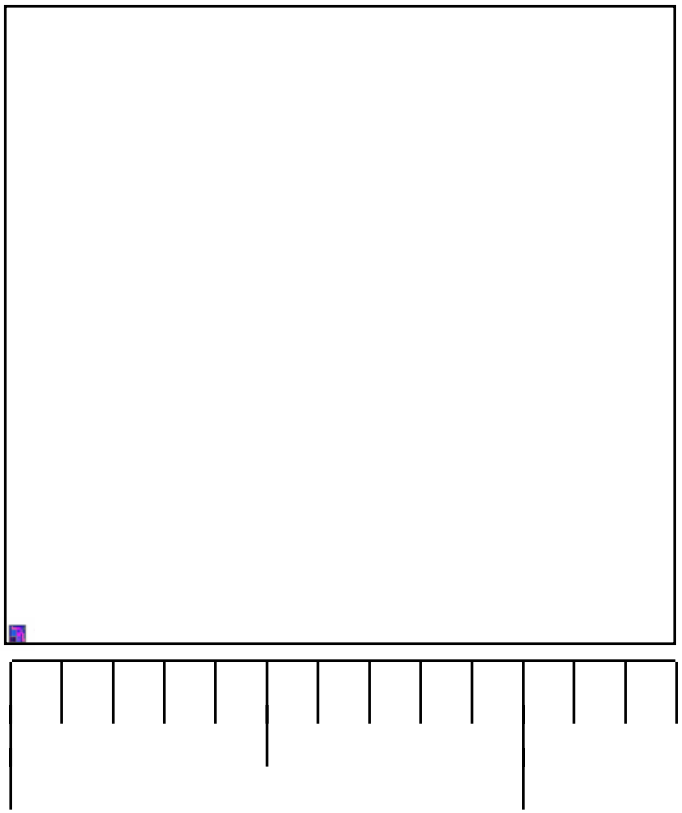
1996…

20MIPS cpu
in 1987



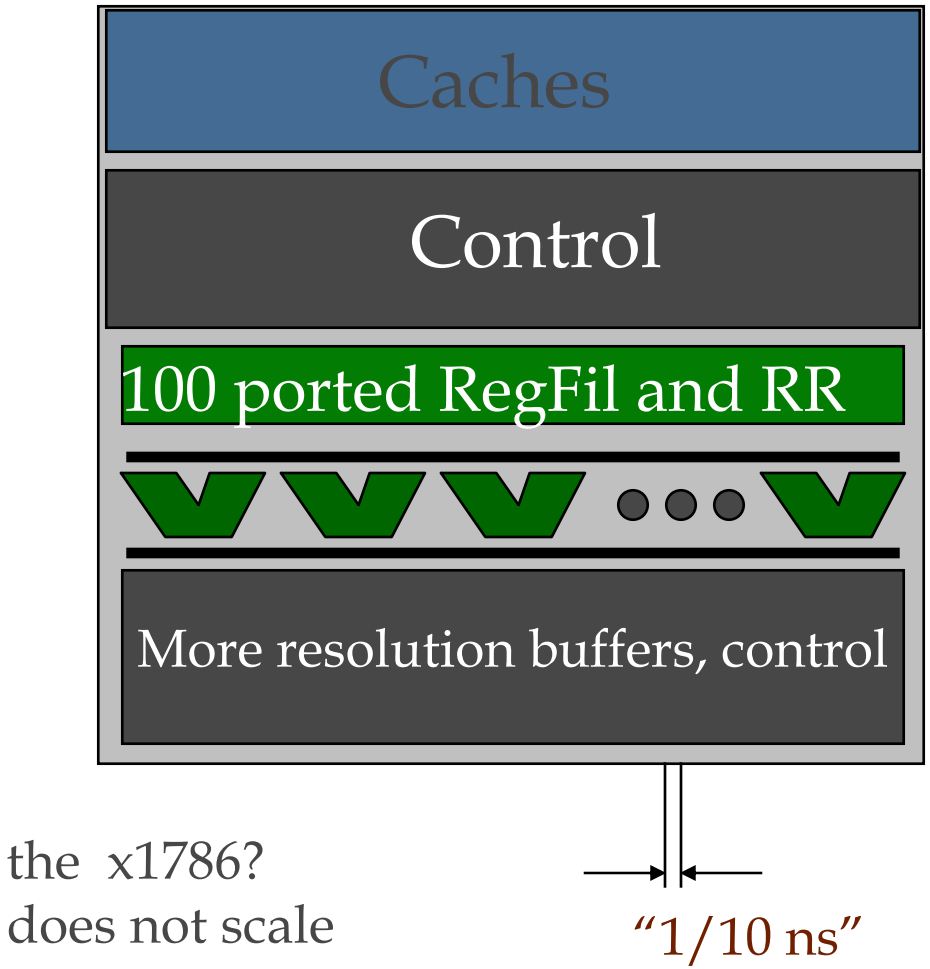
The Opportunity

2007

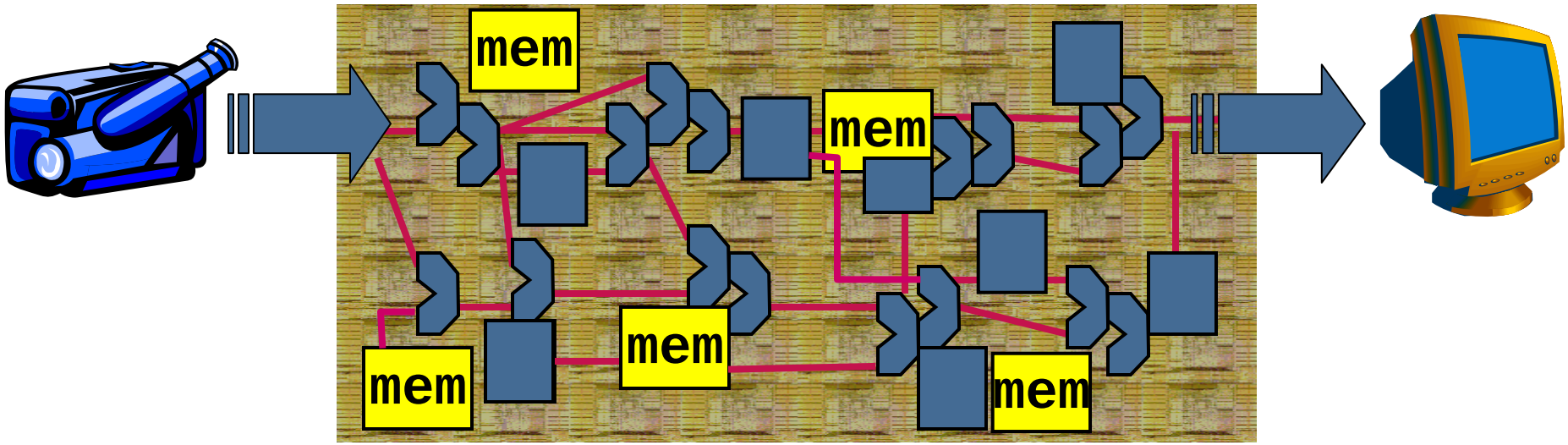


The billion transistor chip

How to Fritter Away Opportunity



Take Inspiration from ASICs

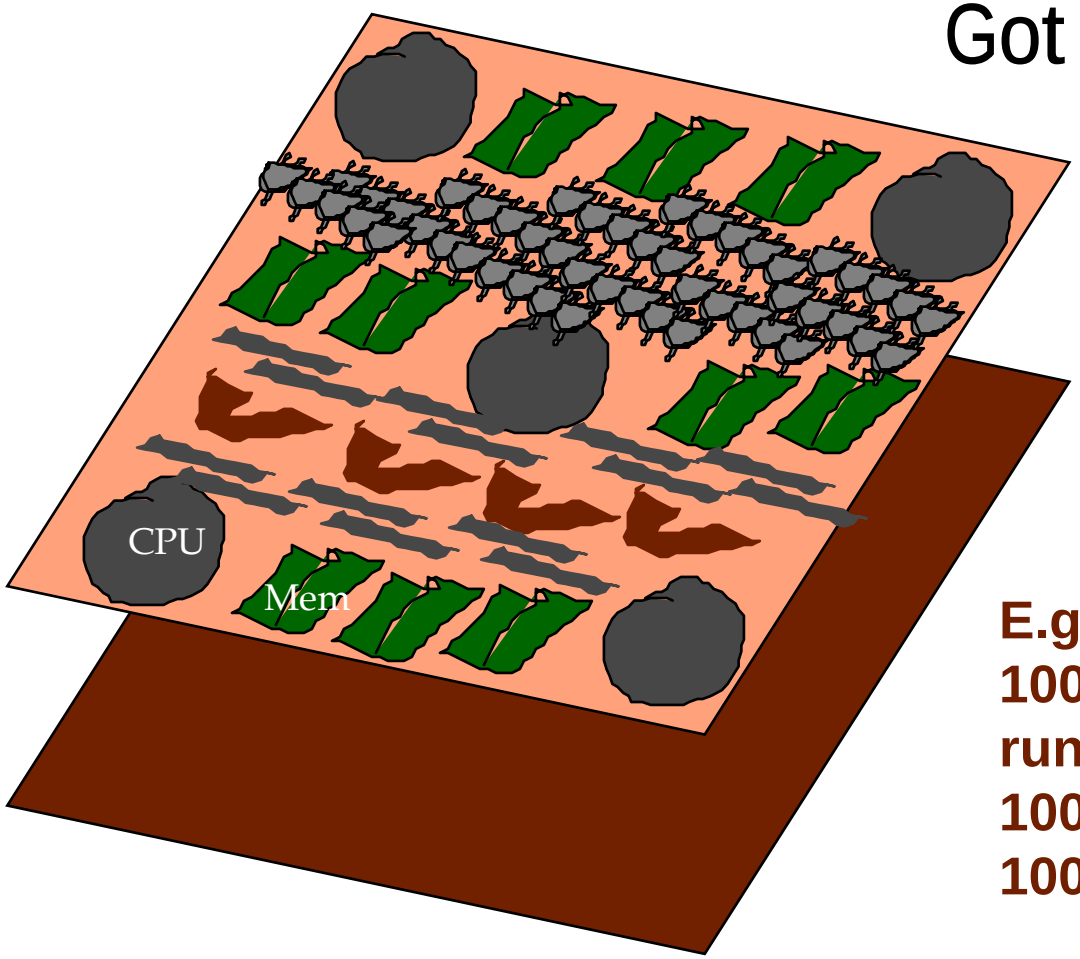


- Lots of ALUs, lots of registers, lots of local memories – huge on-chip parallelism – but with a slower clock
- Custom-routed, short wires optimized for specific applications – locality and orchestrated on-chip communication for scalars and streaming

Fast, low power, area efficient
But not programmable

Our Early Raw Proposal

Got parallelism?

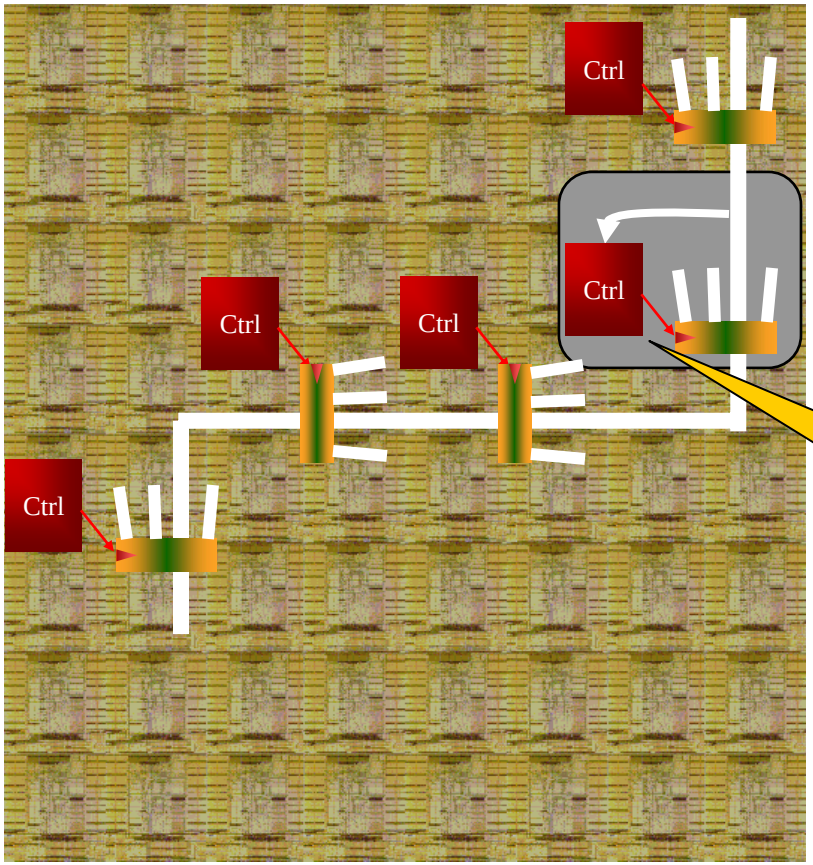


E.g.,
100-way unrolled loop,
running on 100 ALUs,
1000 regs,
100 memory banks

But what about the custom datapaths?

A soft wire…

Pipeline it! Multiplex it! rate it!



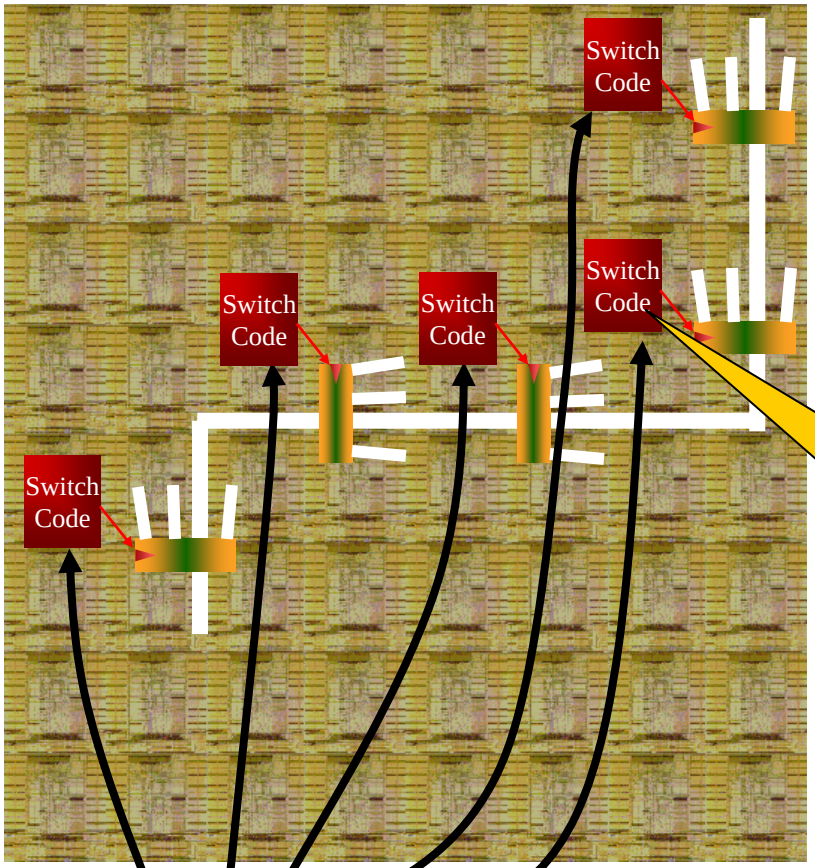
Uh! What were we smoking!

- Fast clock (10GHz in 2010)
- Improve utilization
- Customize to application and imize utilization

A dynamic router!

Custom wires ➔ Routed on-chip networks

Static Router

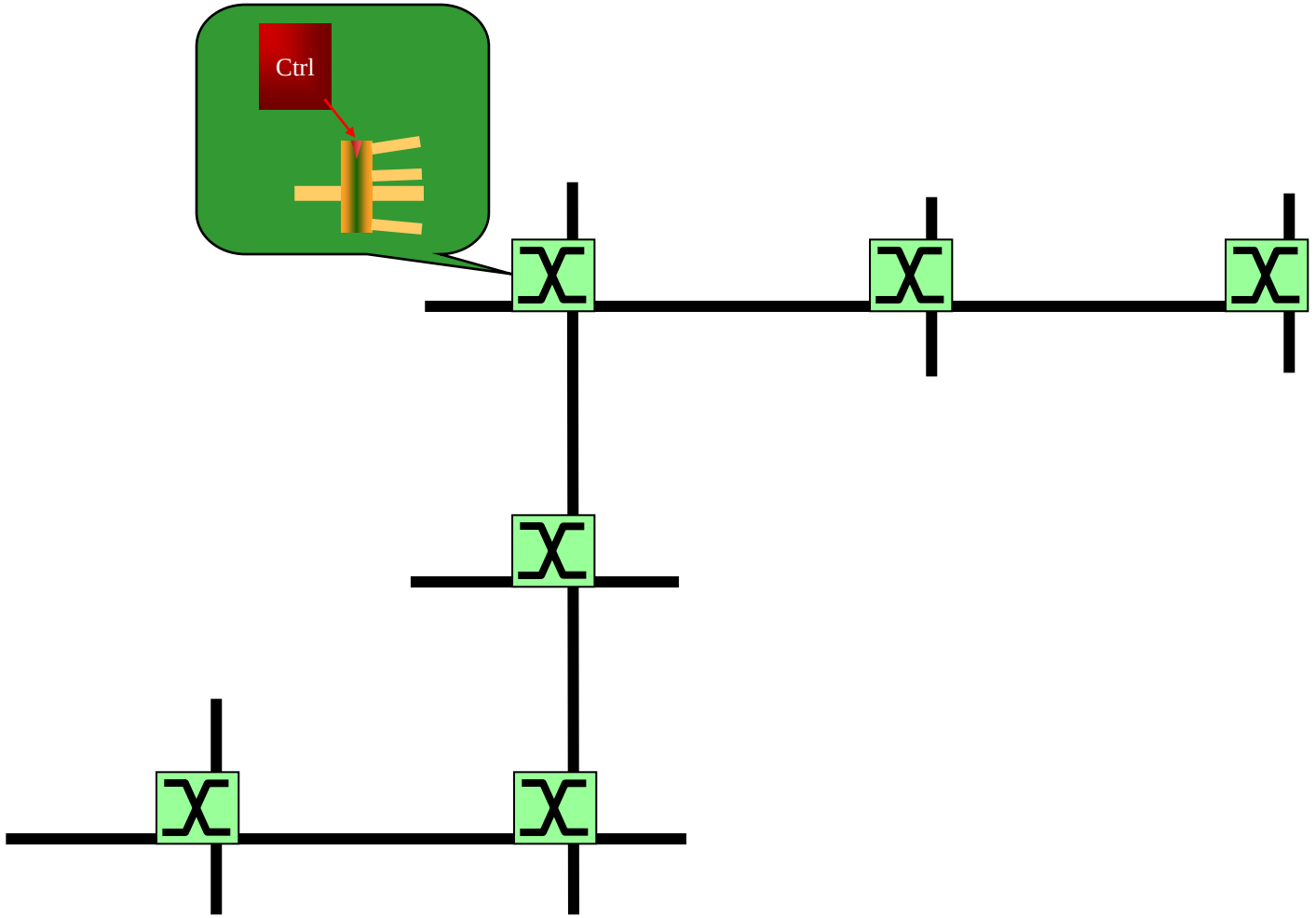


A static router!

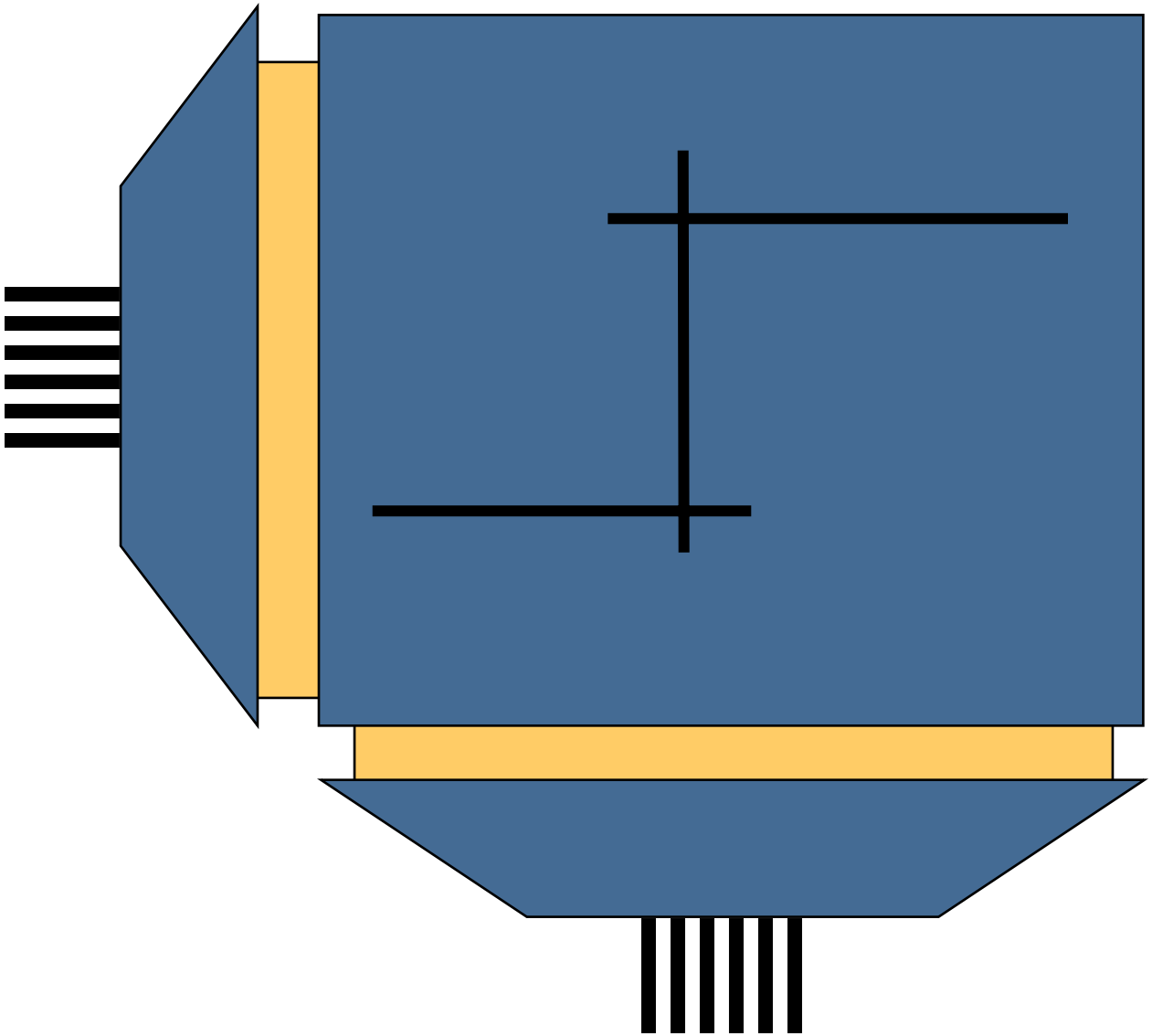
Compiler

Application

Replace Wires with Routed Networks

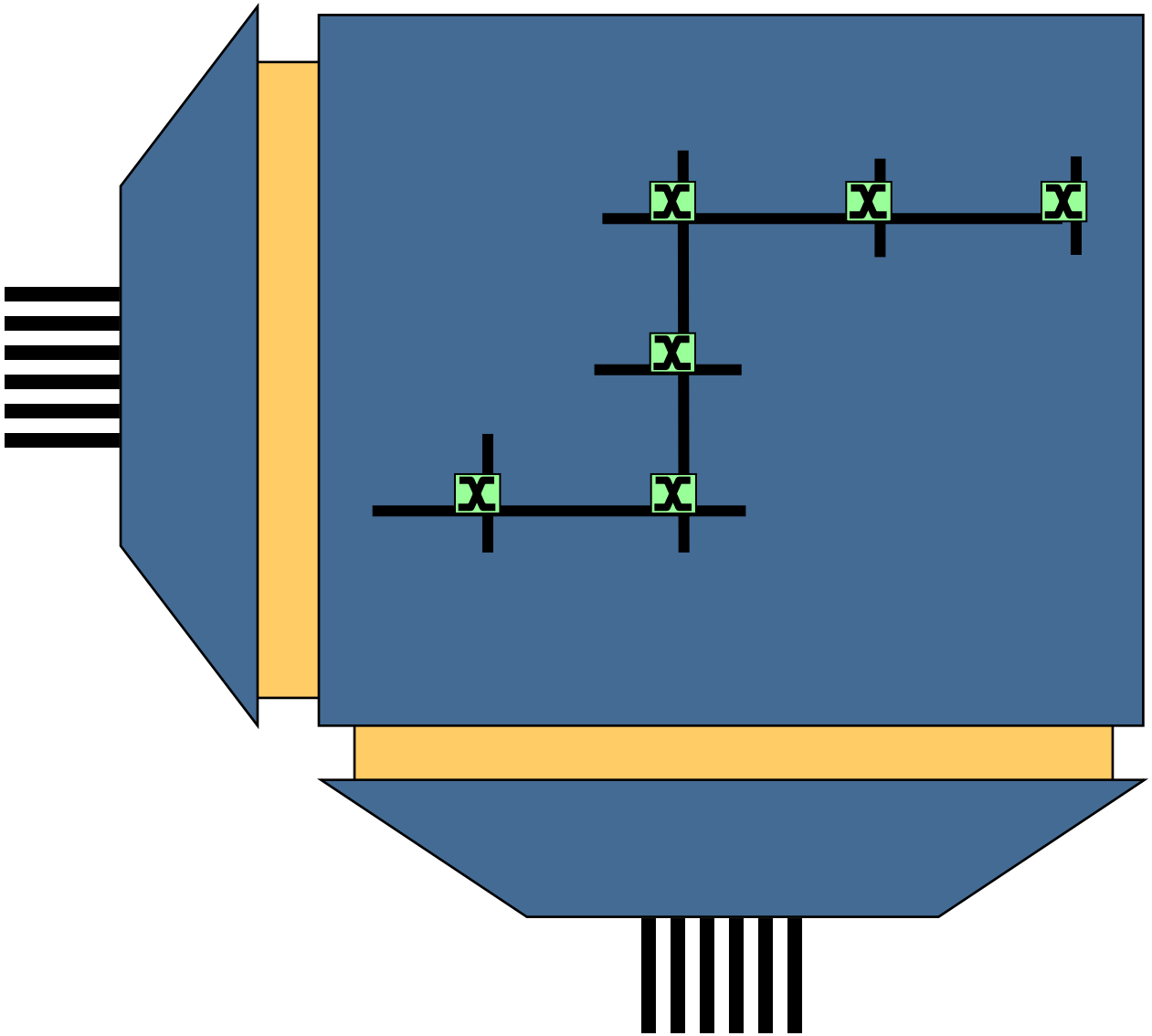


50-Ported Register File → Distributed Registers



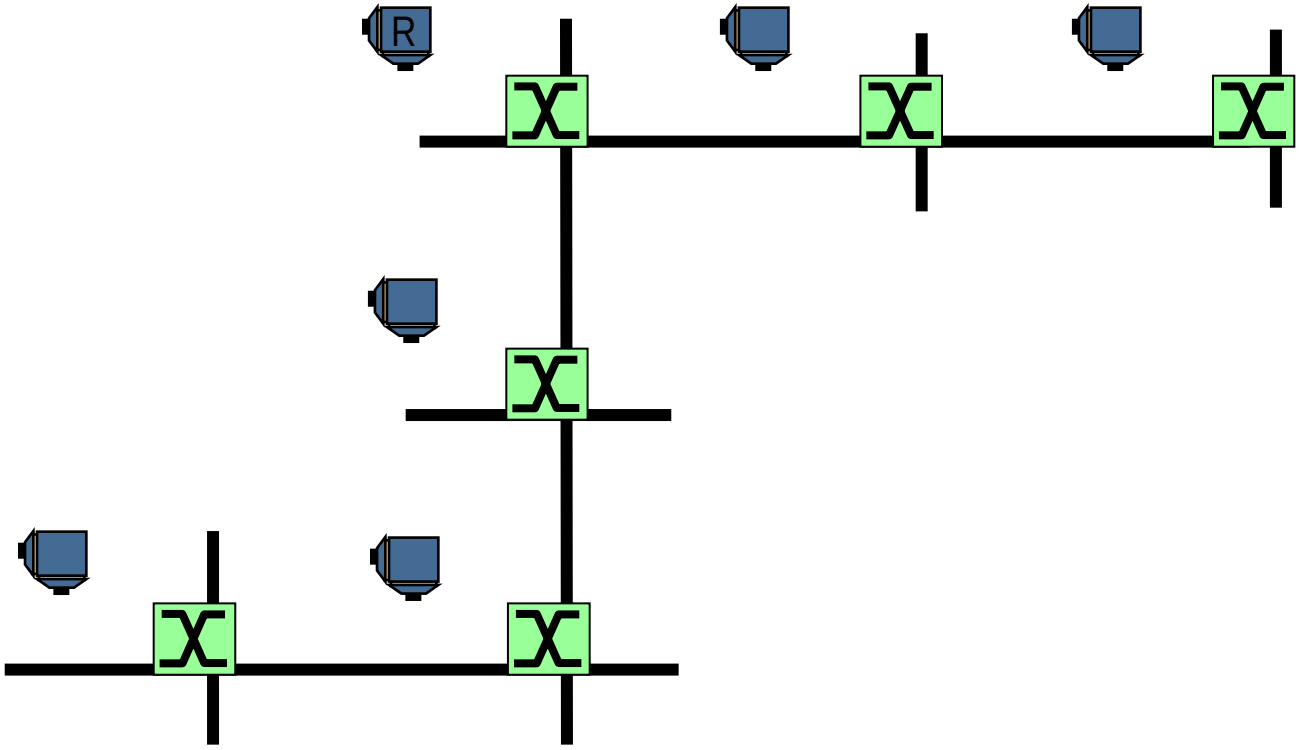
Gigantic 50 ported register file

50-Ported Register File → Distributed Registers



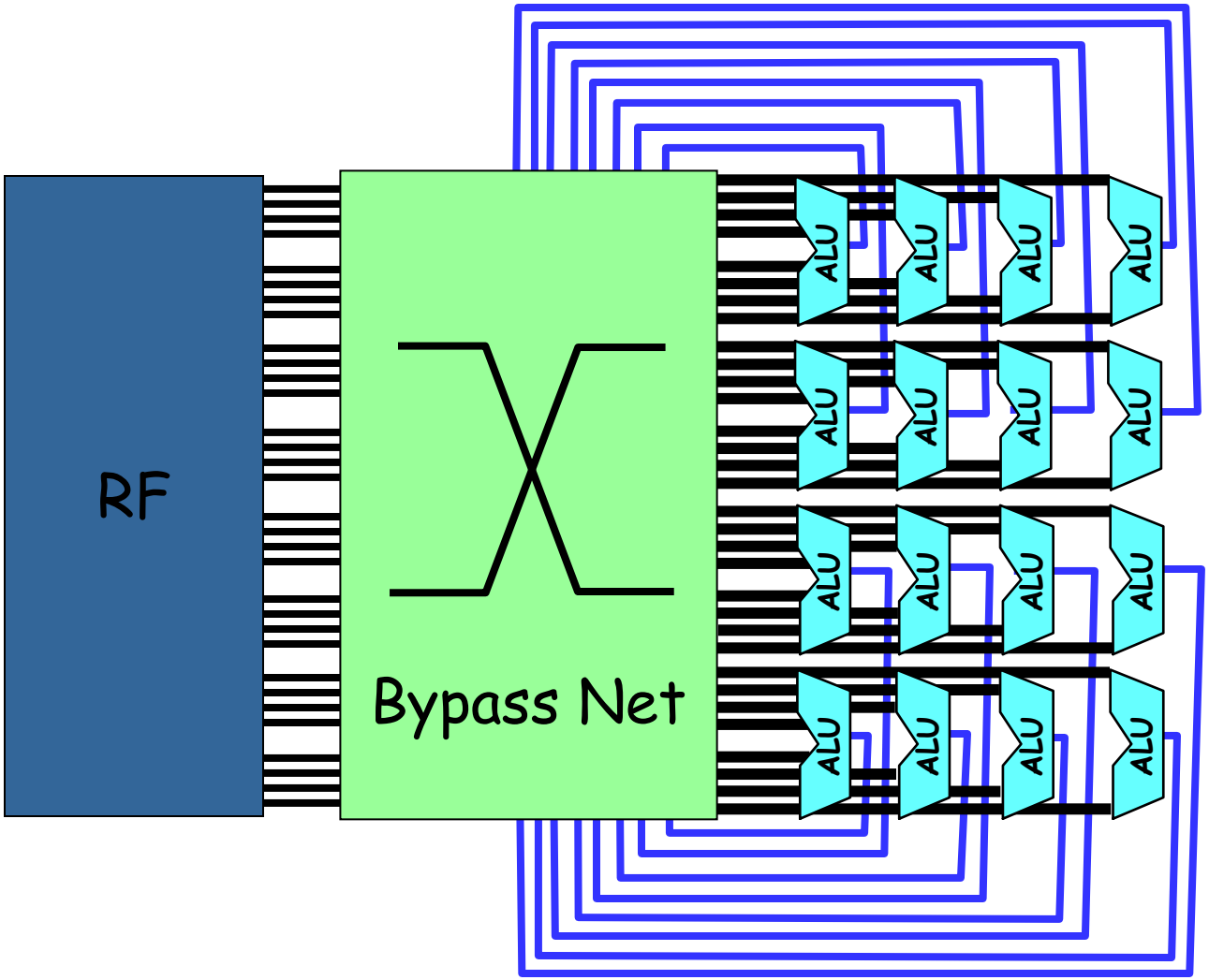
Gigantic 50 ported register file

Distributed Registers + Routed Network

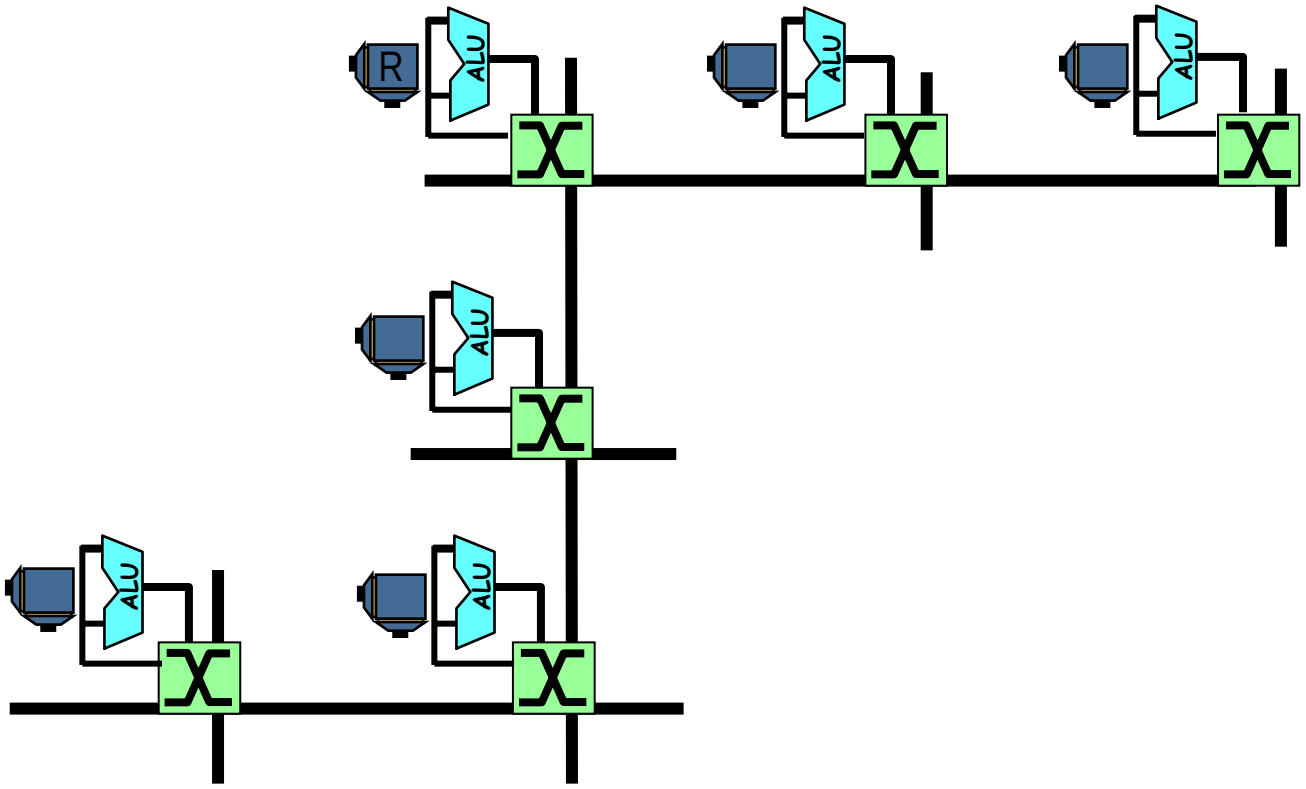


Distributed register file

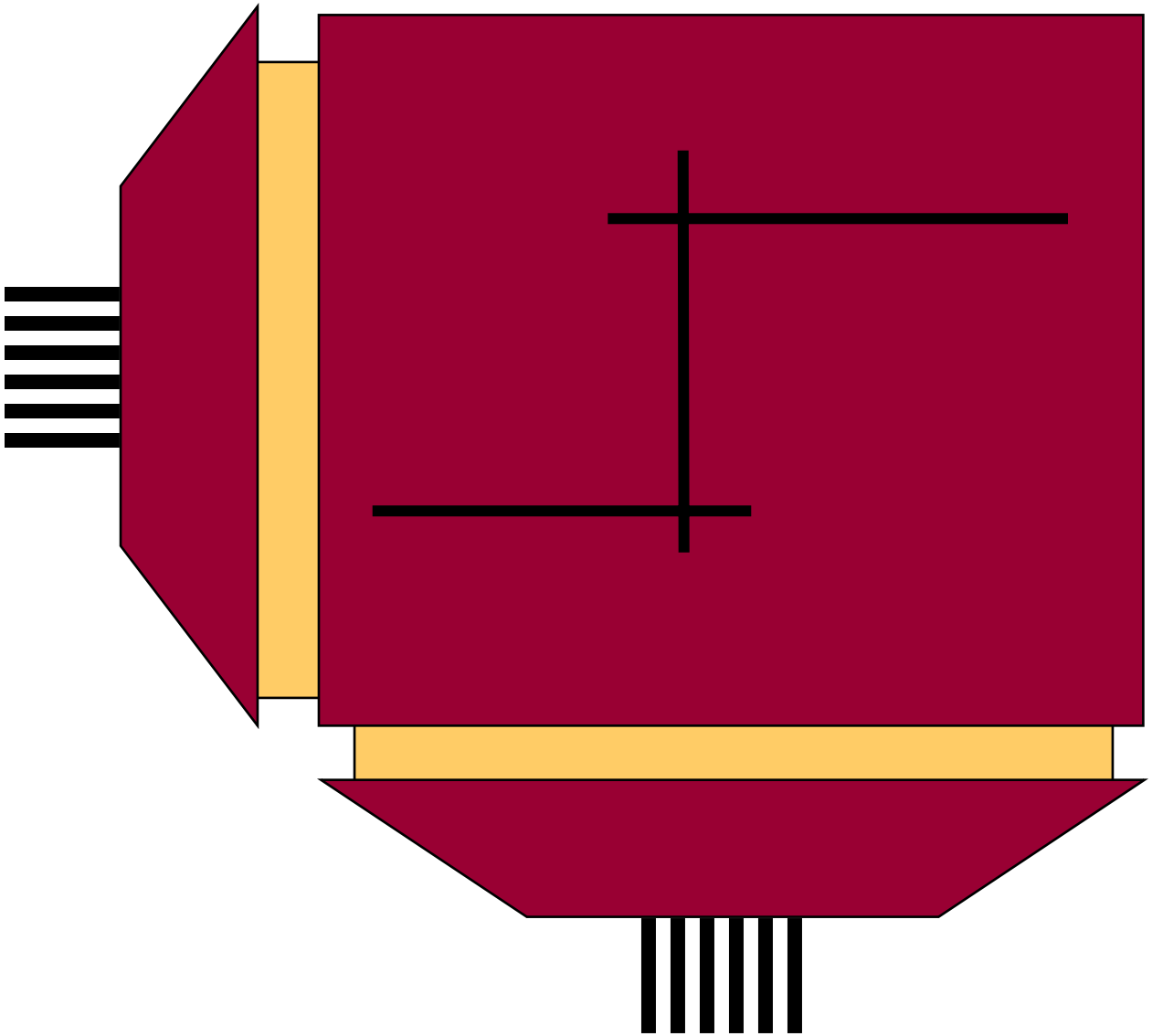
16-Way ALU Clump → Distributed ALUs



Distributed ALUs, Routed Bypass Network

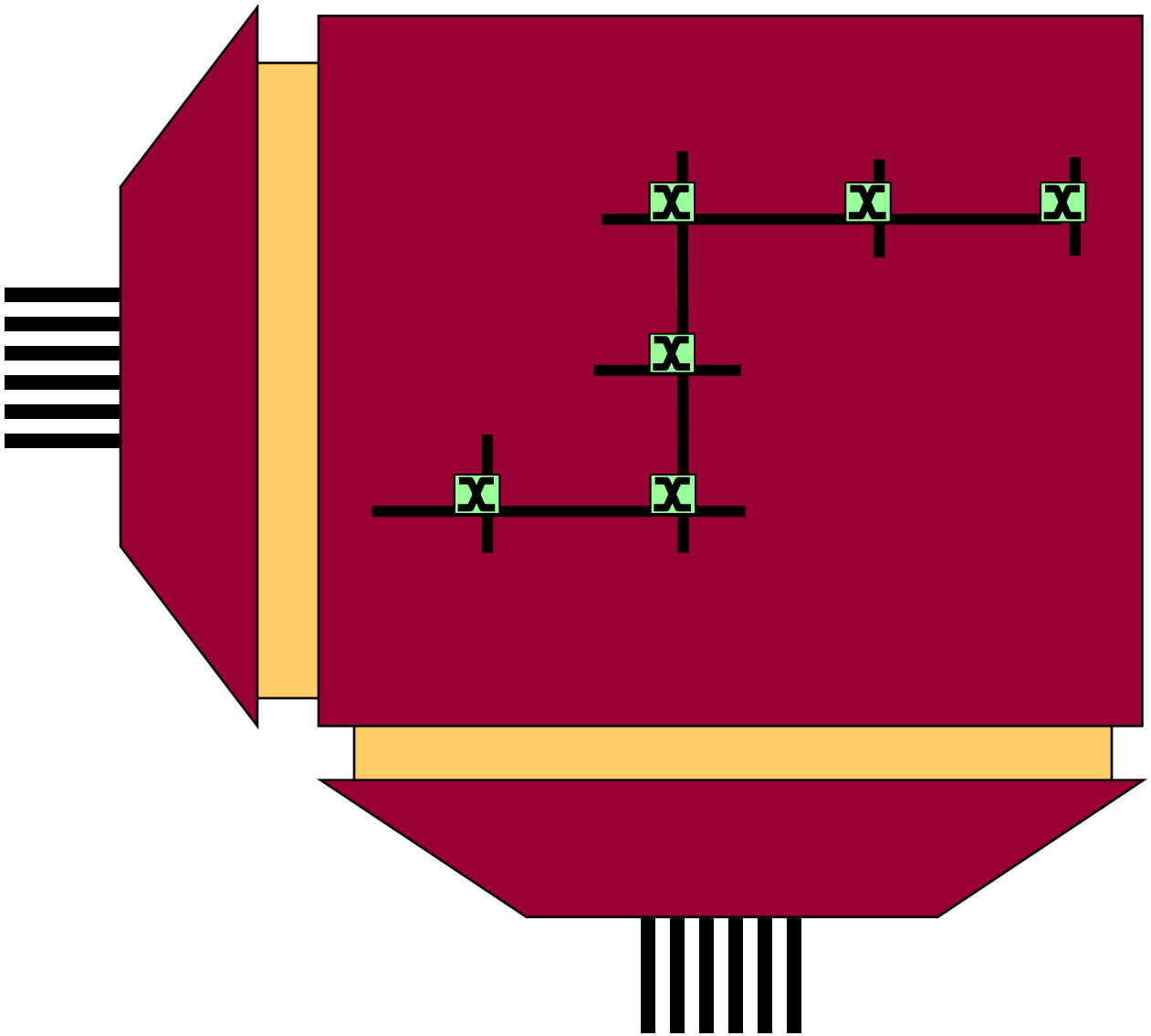


Mongo Cache → Distributed Cache

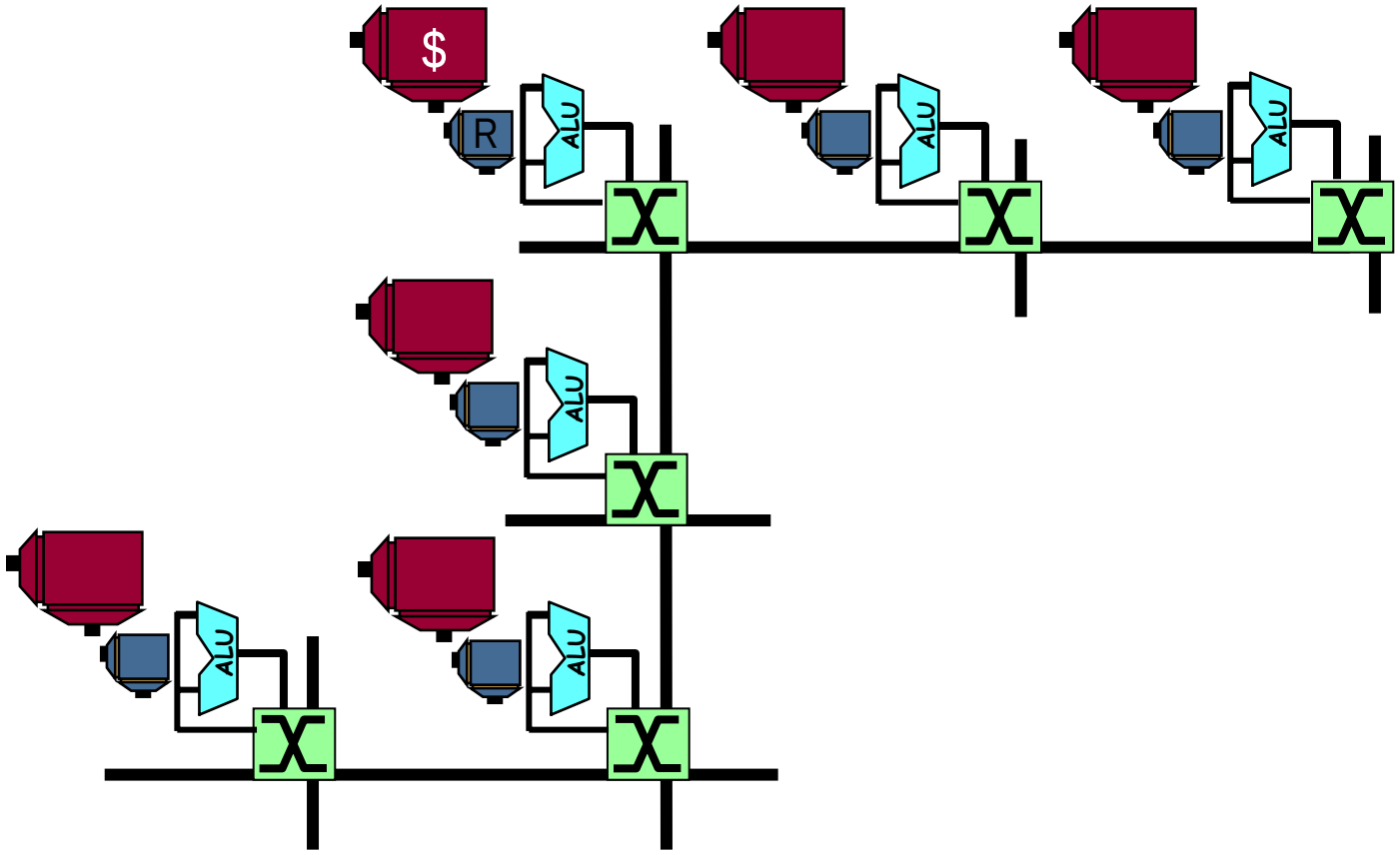


Gigantic
10
ported
cache

Distributing the Cache



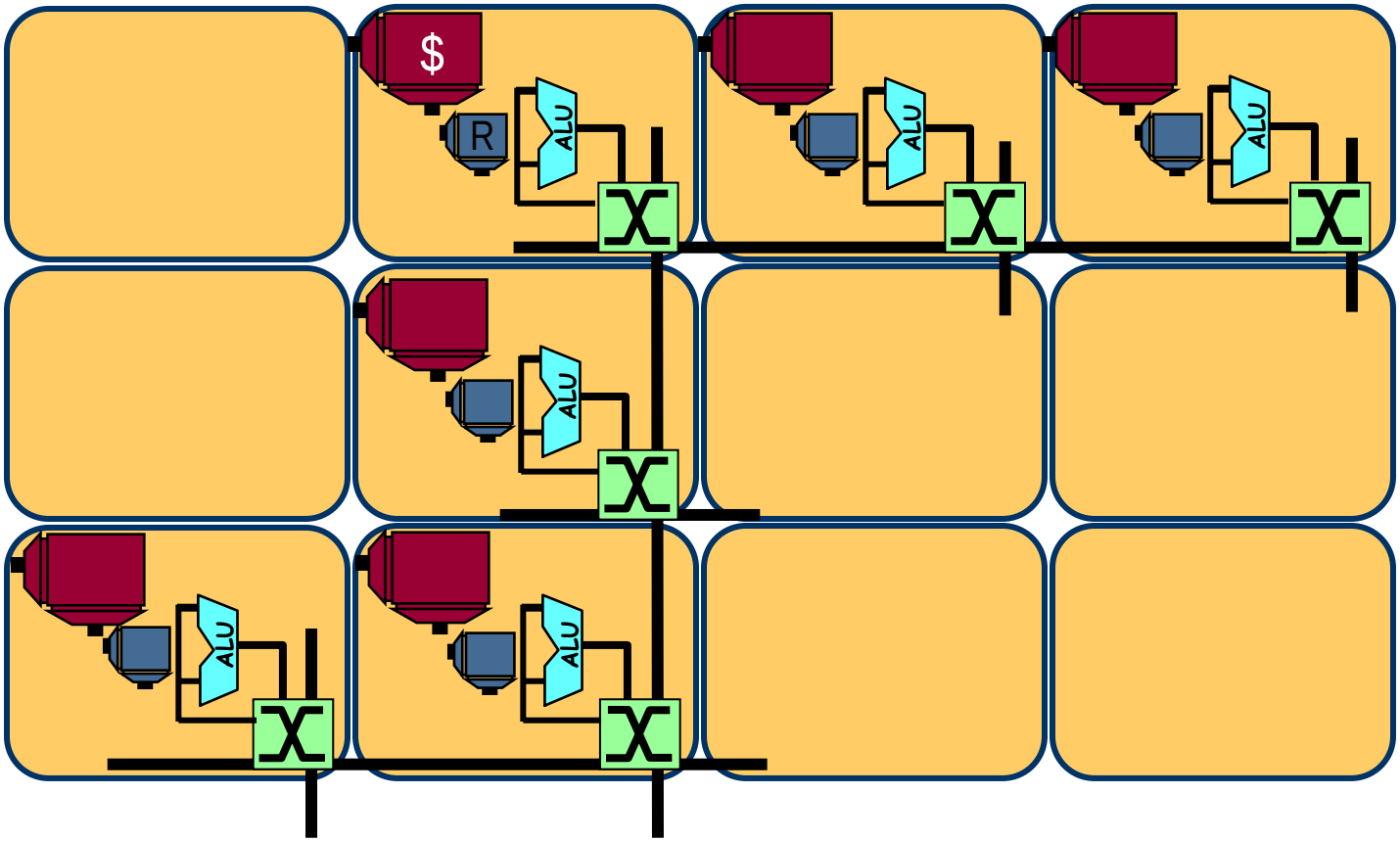
Distributed Shared Cache



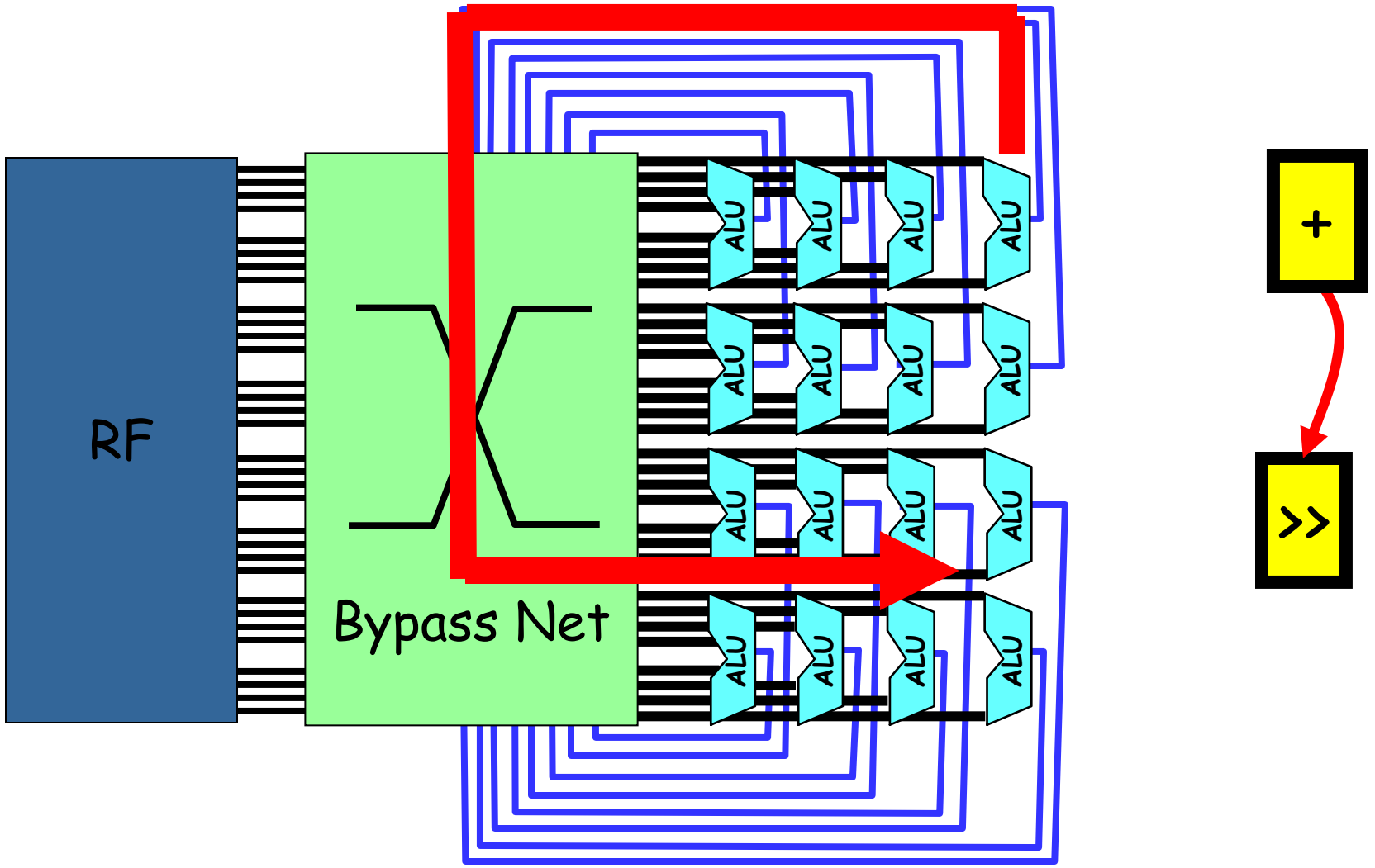
Like DSM (distributed shared memory), cache is distributed

But, unlike NUCA, caches are local to processors, not far away

Tiled Architecture

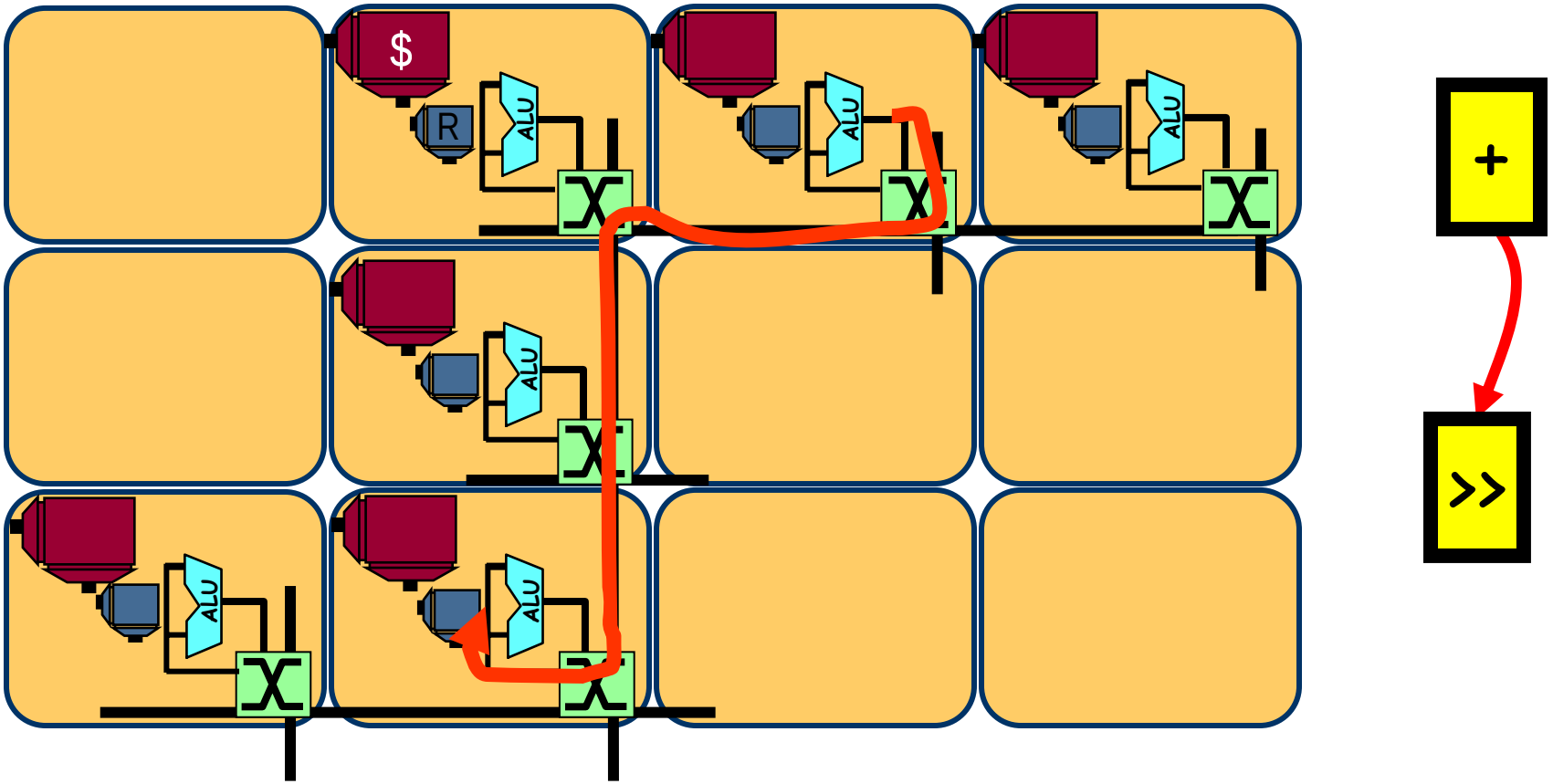


E.g., Operand Routing in 16-way Superscalar



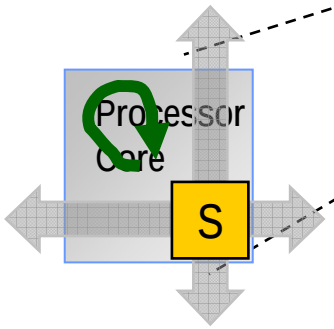
Source: [ISCA04]

Operand Routing in a Tiled Architecture

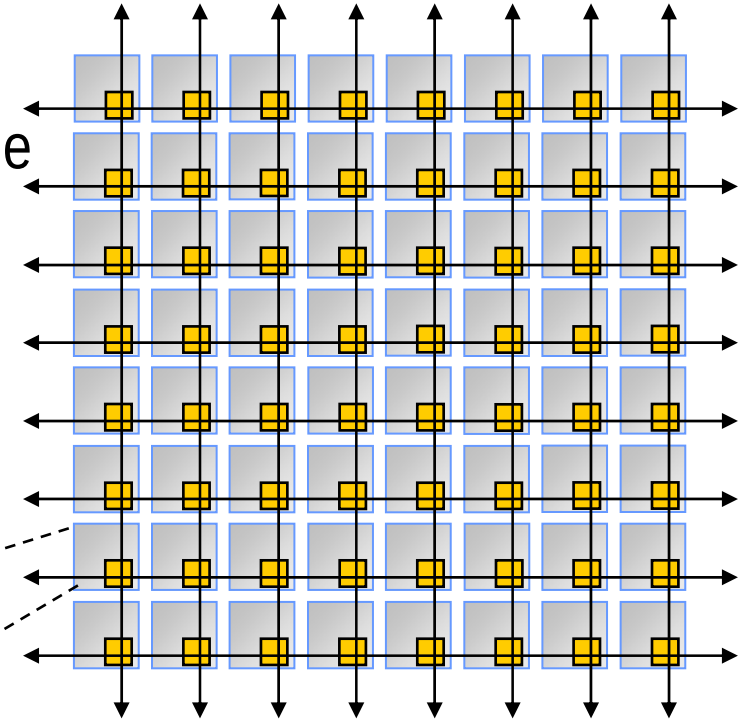


Tiled Multicore

- Scales to large numbers of cores
- Modular – design, layout and verify 1 tile
- Power efficient
  - Short wires CV^2f
  - Chandrakasan effect CV^2f
  - Compiler scheduling

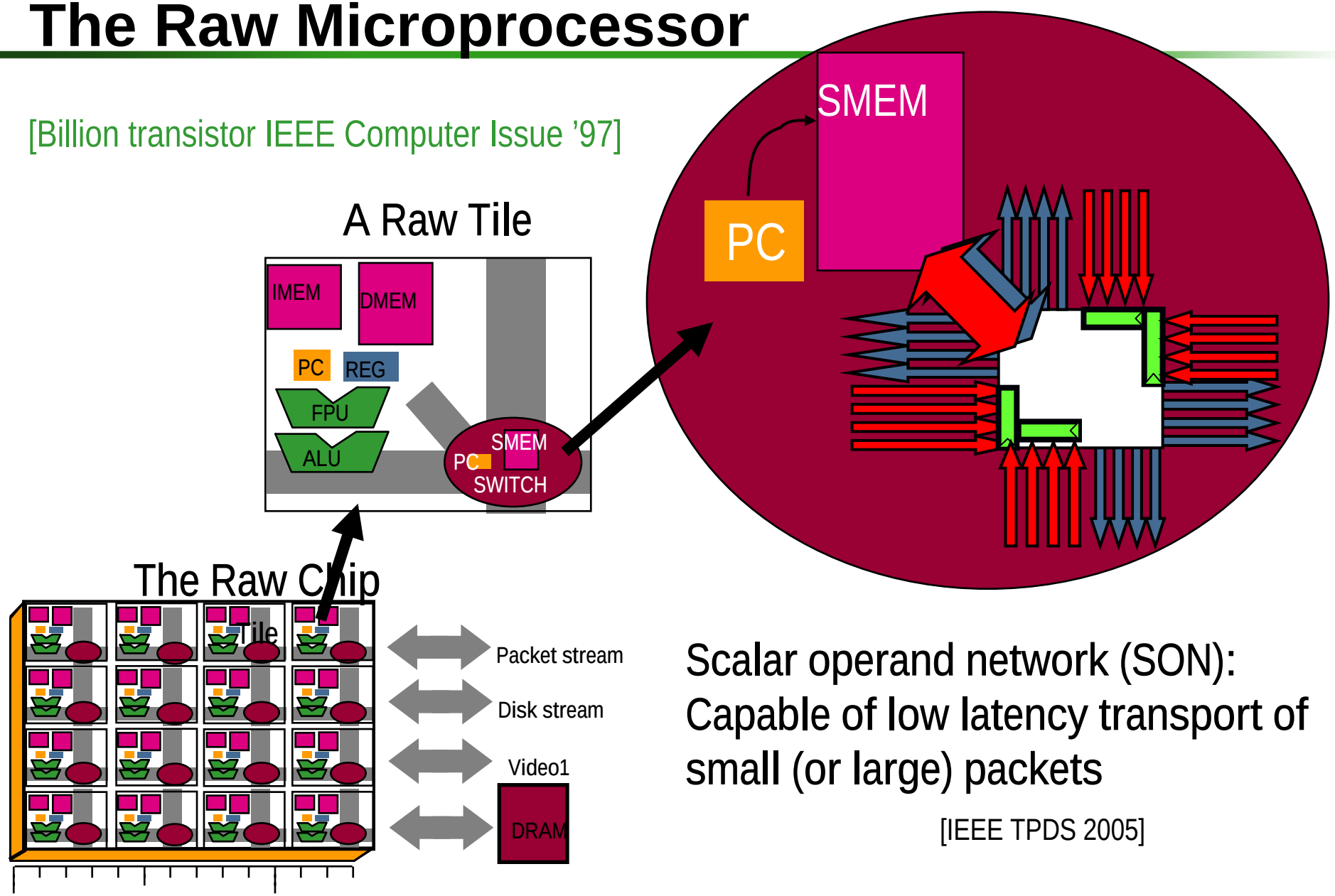


Core + Switch = Tile



A Prototype Tiled Architecture: The Raw Microprocessor

[Billion transistor IEEE Computer Issue '97]



Scalar operand network (SON): Capable of low latency transport of small (or large) packets

[IEEE TPDS 2005]

Virtual reality

Simulator reality

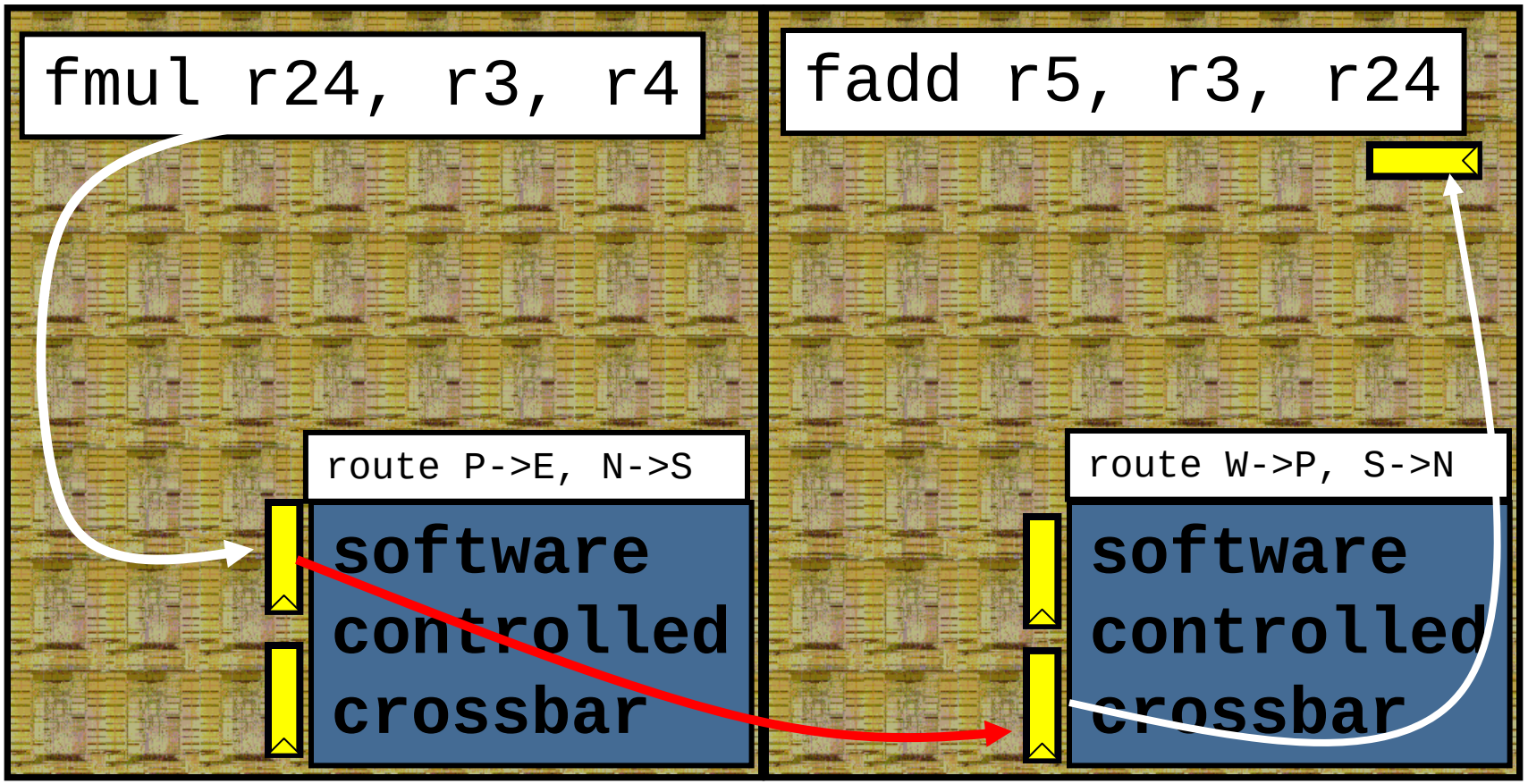
Prototype reality
Product reality

Raw's On-Chip Networks

- Dynamic network
  - Packet switched dynamic routing
  - Used for dynamic events
    - Cache miss handling
    - I/O
    - System and messaging traffic
- Static network
  - Operand transport for ILP
  - Stream data transport

Scalar Operand Transport in Raw

Goal: flow controlled, in order delivery of operands

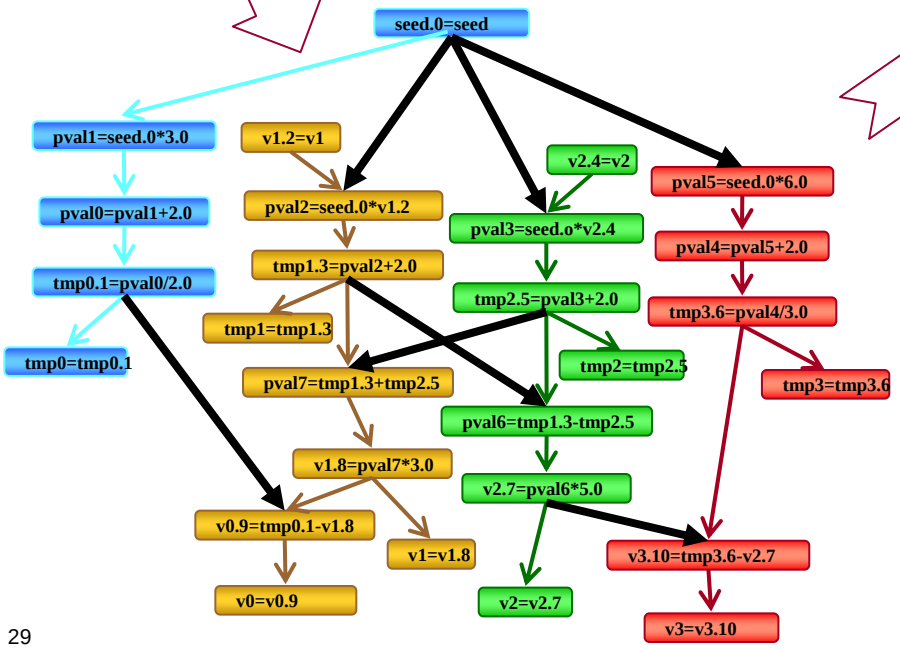


RawCC: Distributed ILP Compilation (DILP)

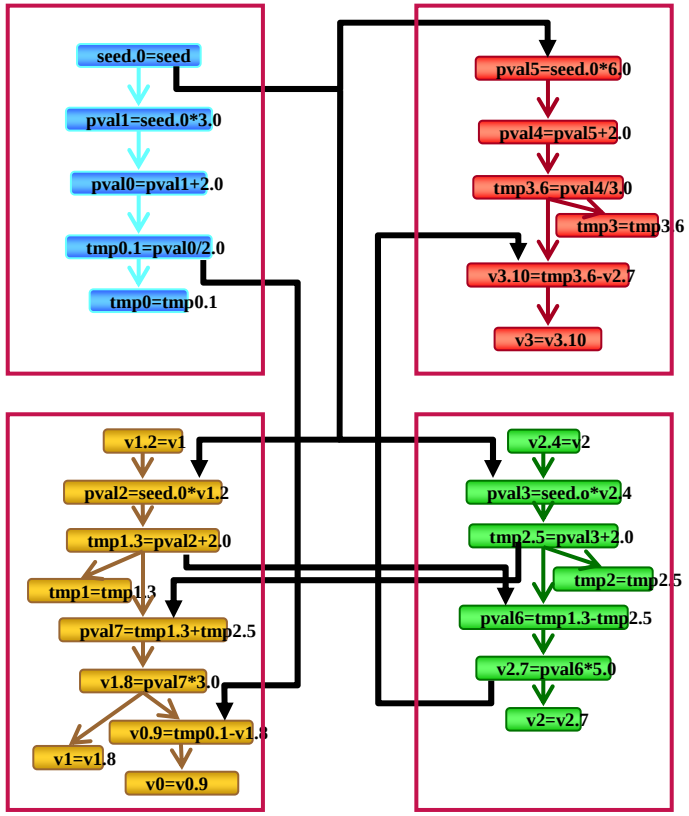
```
tmp0 = (seed*3+2)/2
tmp1 = seed*v1+2
tmp2 = seed*v2 + 2
tmp3 = (seed*6+2)/3
v2 = (tmp1 - tmp3)*5
v1 = (tmp1 + tmp2)*3
v0 = tmp0 - v1
v3 = tmp3 - v2
```

C

Partitioning



Place, Route, Schedule



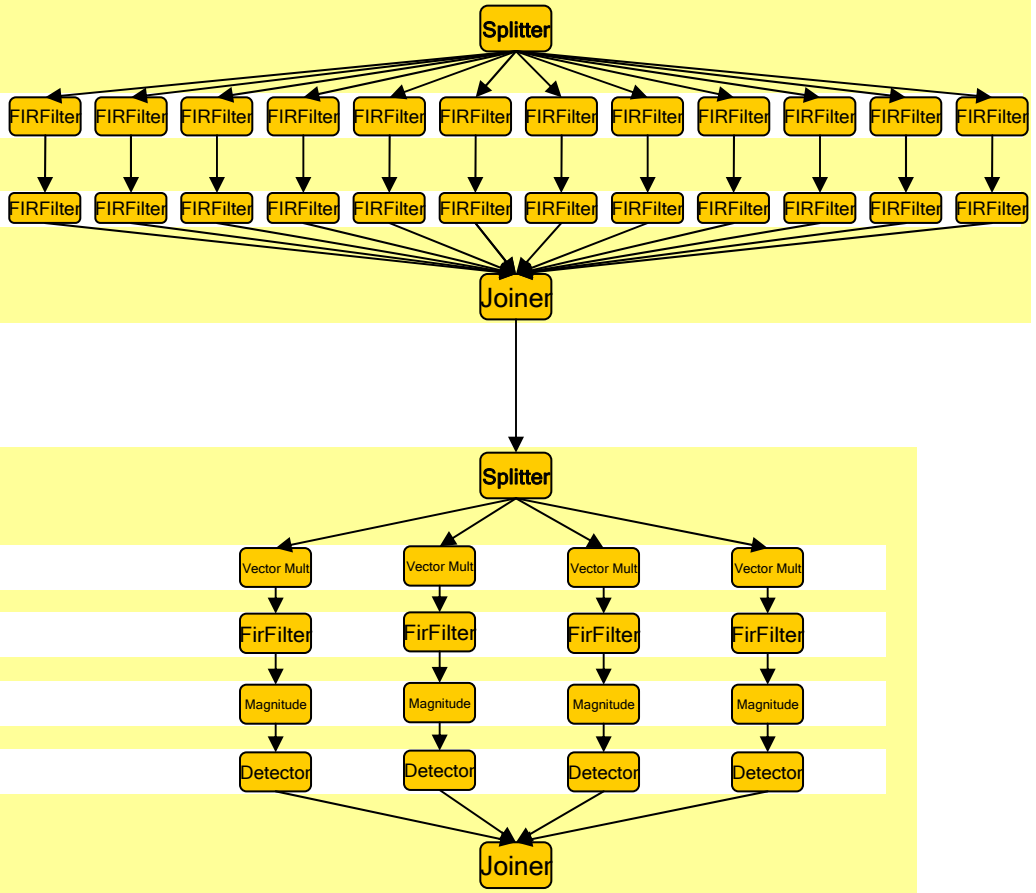
Black arrows = Operand Communication over SON

[ASPLOS 98]

StreamIt: Stream Programming and Compilation

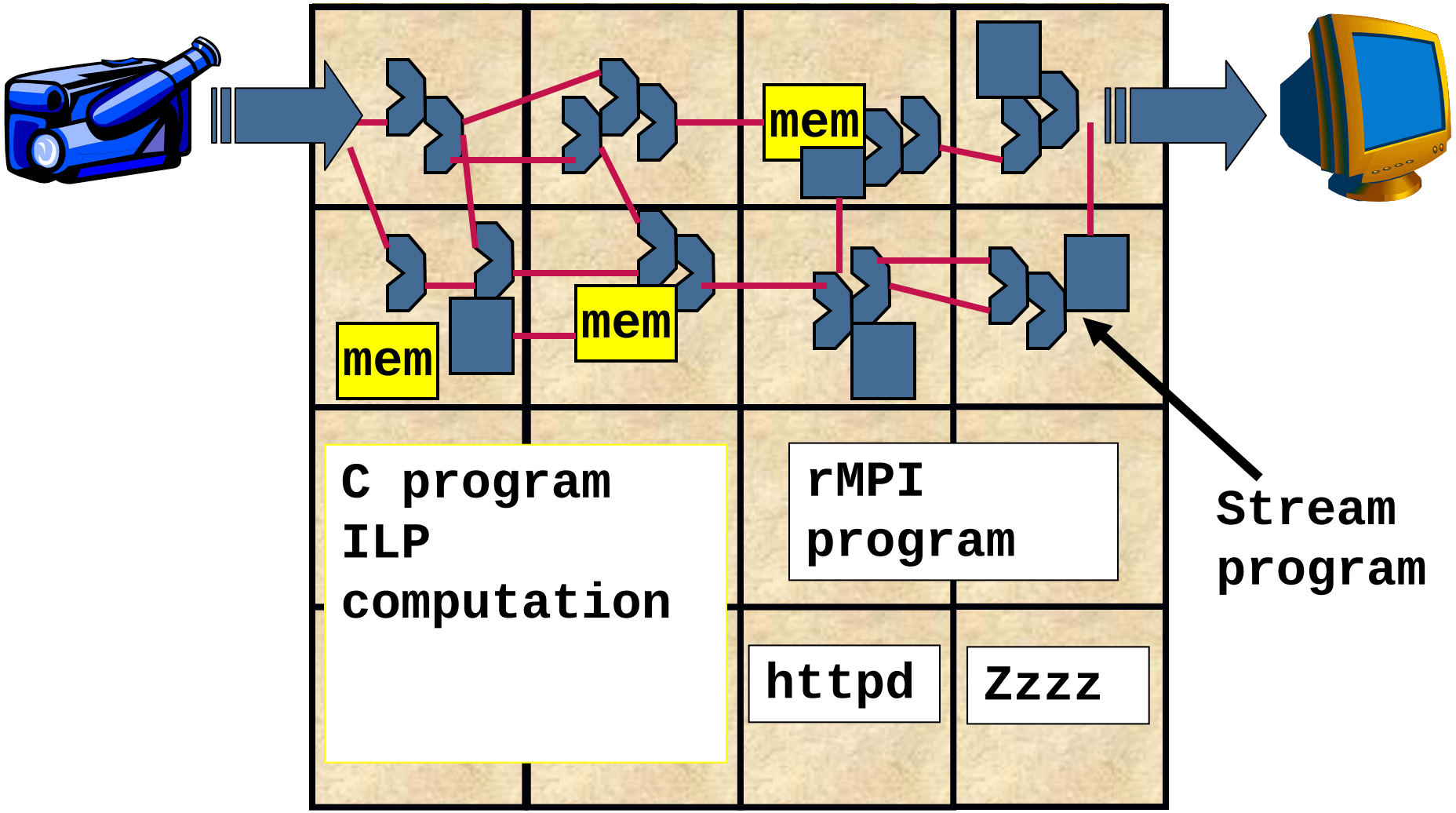
```
class BeamFormer extends Pipeline {
    void init(numChannels, numBeams) {
        add(new SplitJoin() {
            void init() {
                setSplitter(Duplicate());
                for (int i=0; i<numChannels; i++) {
                    add(new FIR1(N1));
                    add(new FIR2(N2));
                }
                setJoiner(RoundRobin());  }});
    }
        add(new SplitJoin() {
            void init() {
                setSplitter(Duplicate());
                for (int i=0; i<numBeams; i++) {
                    add(new VectorMult());
                    add(new FIR3(N3));
                    add(new Magnitude());
                    add(new Detect());
                }
                setJoiner(Null());  }});
}
```

e.g., BeamFormer



[ASPLOS 2002]

The End Result: DILP, Streaming, TLP



Virtual reality

Simulator reality

Prototype reality

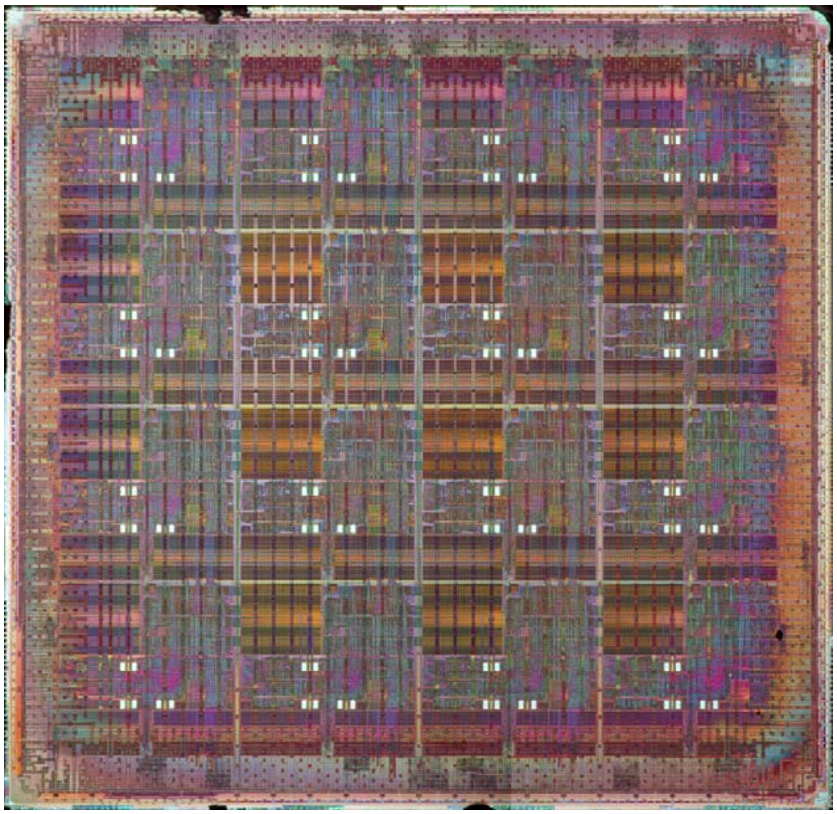
Product reality

A Tiled Processor Architecture Prototype: the Raw Microprocessor

Michael Taylor
Walter Lee
Jason Miller
David Wentzlaff
Ian Bratt
Ben Greenwald
Henry Hoffmann
Paul Johnson
Jason Kim
James Psota
Arvind Saraf
Nathan Shnidman
Volker Strumpen
Matt Frank
Rajeev Barua
Elliot Waingold
Jonathan Babb
Sri Devabhaktuni
Saman Amarasinghe
Anant Agarwal

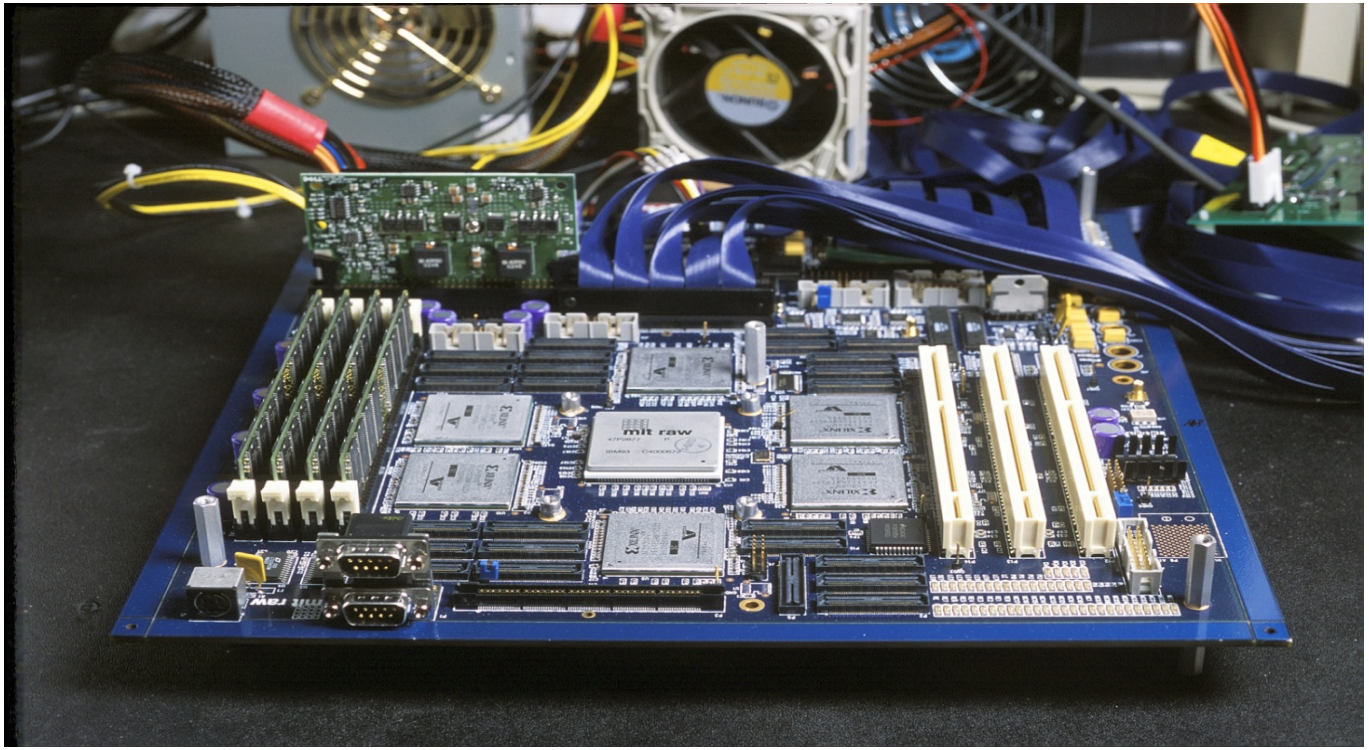
October 02

Raw Die Photo

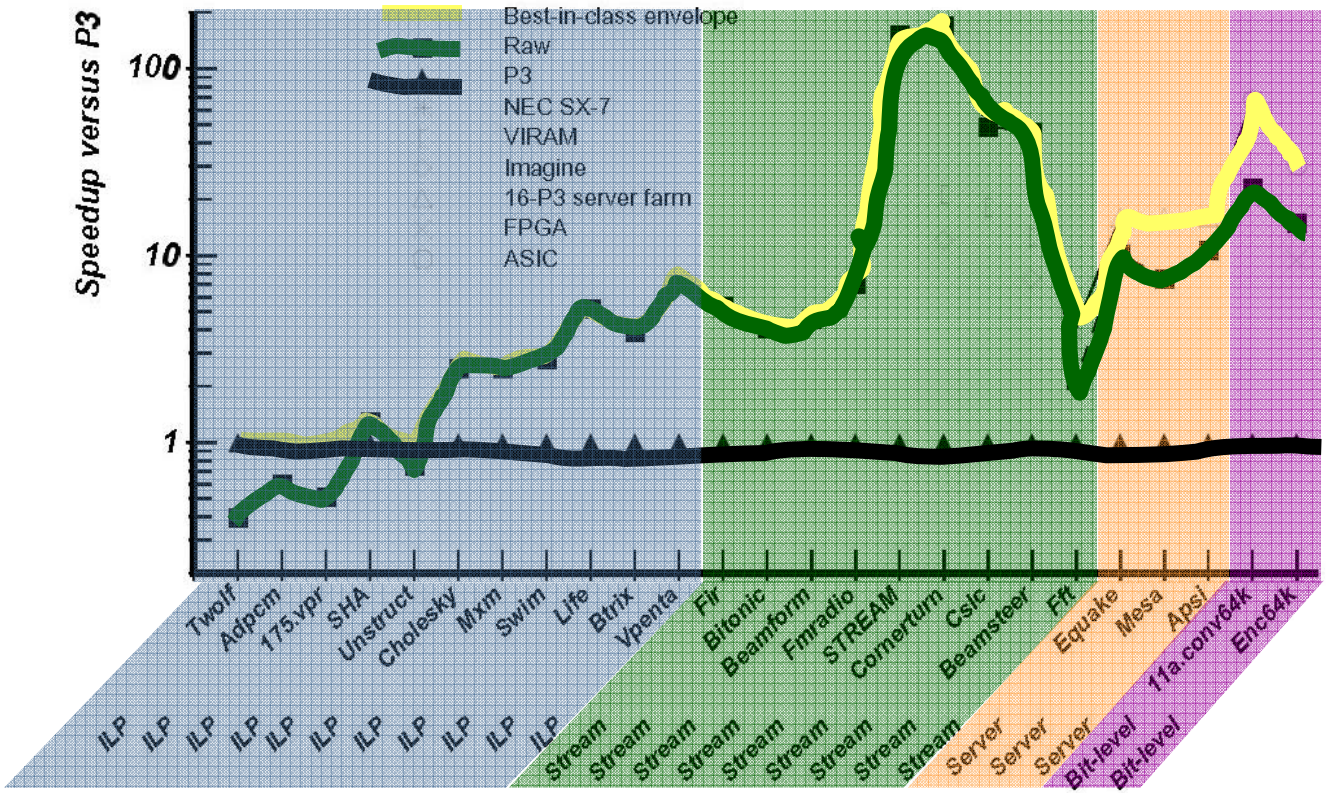


IBM .18 micron process,
16 tiles,
425MHz,
18 Watts (vpenta)

Raw Motherboard

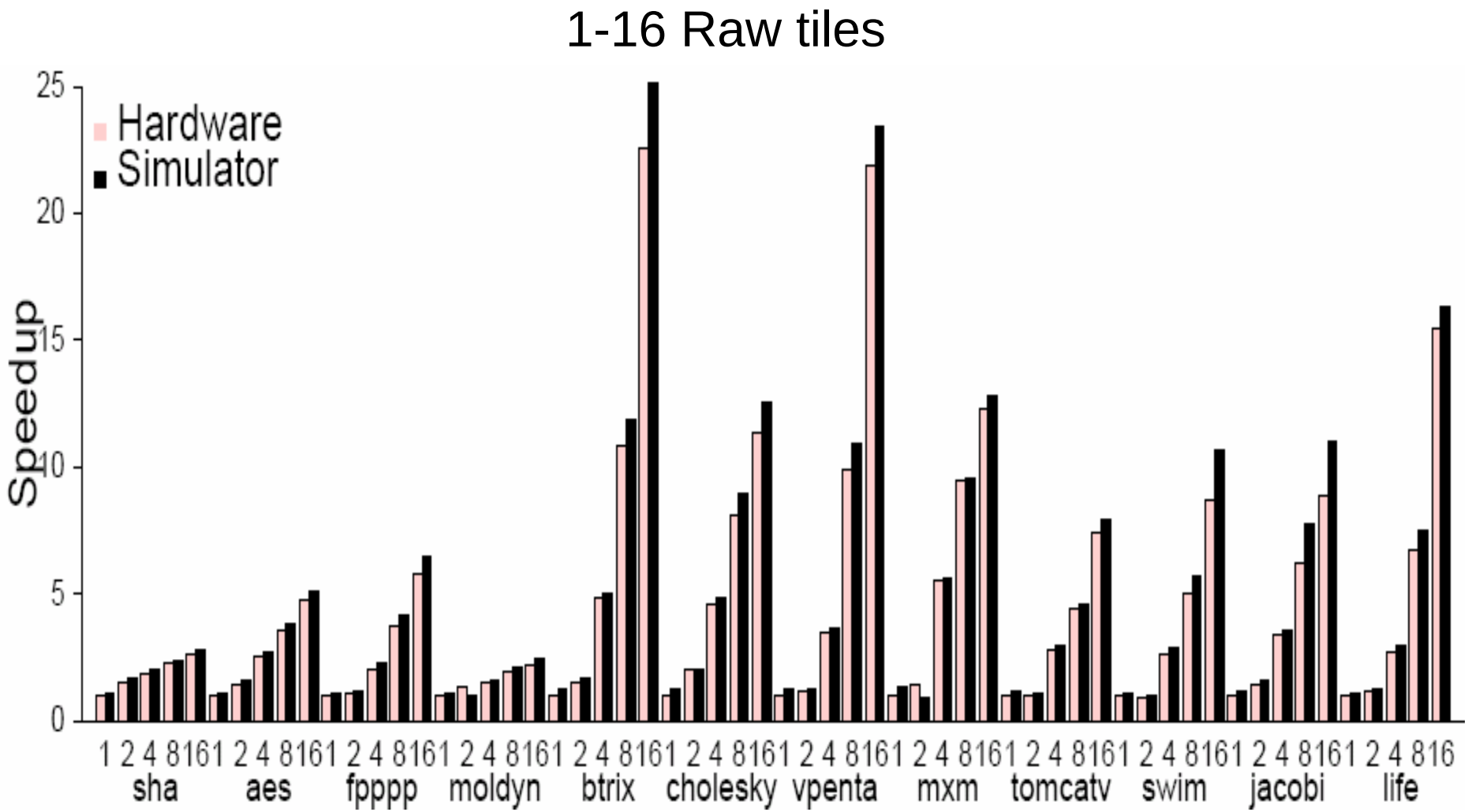


Performance Results



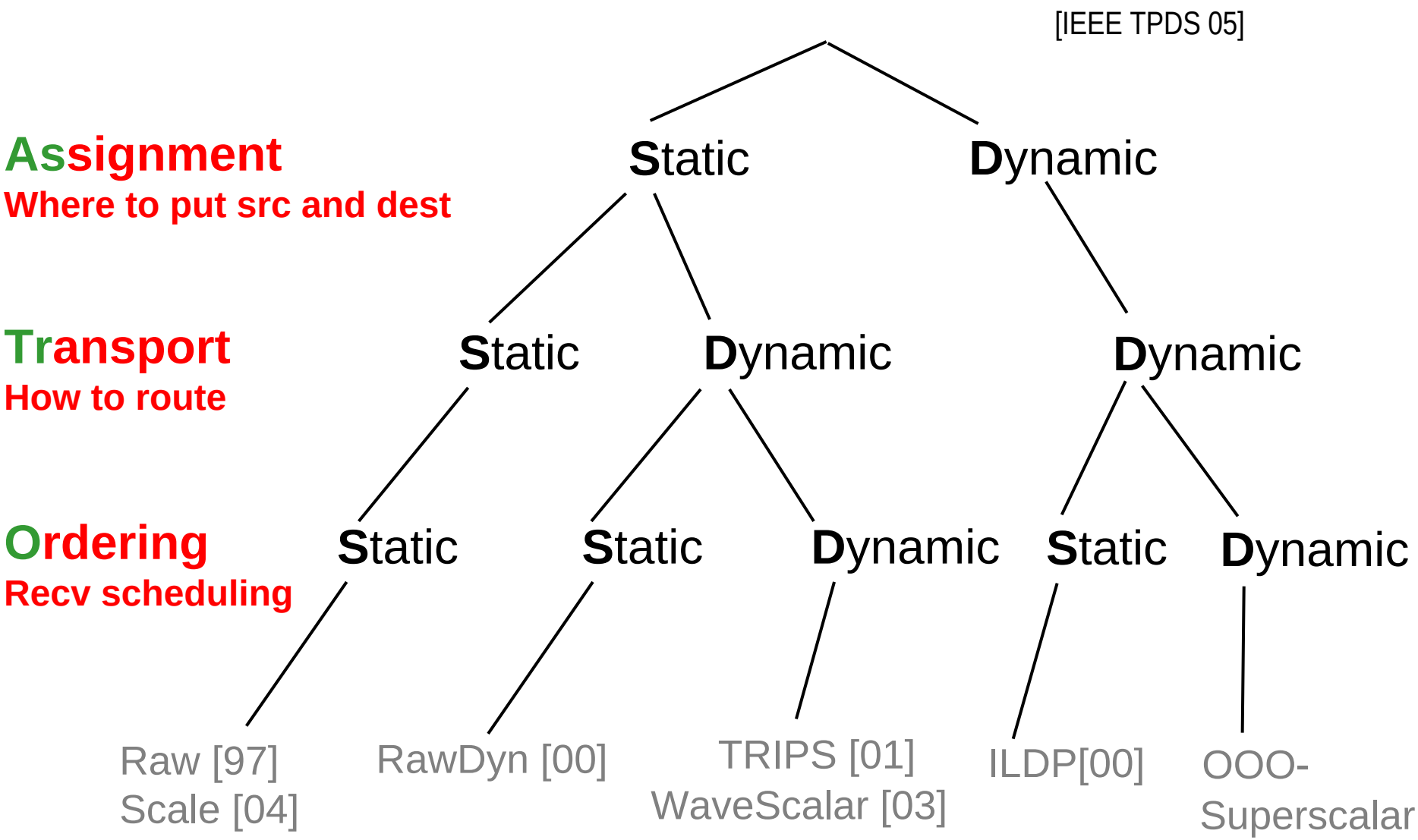
[ISCA04]

Simulator Reality vs Prototype Reality



Some Science: AsTrO Classification of SONs

[IEEE TPDS 05]



Raw Ideas and Decisions: What Worked, What Did Not

- Build a complete prototype system
- Simple processor
  - Single issue cores
  - Software data caching
  - Software instruction caching
- FPGA logic block in each tile
- Distributed ILP
- Stream processing
- Multiple types of computation – ILP, streams, TLP, server
- PC in every tile
- Static network for ILP and streaming

Why Build?

- Compiler (Amarasinghe), OS, apps (ISI, Lincoln Labs, Durand) folks will not work with you unless you are serious about building hardware
- Need motivaion to build software tools -- compilers, runtimes, debugging, visualization – many challenges here
- Run large data sets (simulation takes forever even with 100 servers!)
- Many hard problems show up or are better understood after you begin building (how to maintain ordering for distributed ILP, sw icaching in the presence of interrupts, slack for streaming codes)
- Have to solve hard problems – no magic!
- The more radical the idea, the more important it is to build
  - World will only trust end-to-end results since it is too hard to dive into details and understand all assumptions
  - Would you believe this: "Prof. John Bull has demonstrated a simulation prototype of a 64-way issue out-of-order superscalar"
- Cycle simulator became cycle *accurate* simulator only after hardware got precisely defined
- Don't bother trying to commercialize unless you have a working prototype

Raw Ideas and Decisions: Simple Processor

- Simple processor -- How simple is too simple?
  - Single issue cores
    - Single issue was probably too simple
    - Simplefit study [IEEE TPDS 2001] argued for slightly higher issue width, 2-way
    - But, higher way was not worth the engineering effort in university
    - Tilera's Tile processor went 3-way VLIW
    - Remains one of the biggest research questions in multicore – grain size vs number of cores
  - Software data caching
    - Saw the light early on; included a hardware data cache
    - But did include a mode in the instruction cache to study software data caching ideas
    - Within the embedded domain, still need a way to get real-time performance in the face of caches, yet have the ease of programming afforded by caches
    - Tilera's Tile Processor solved this with Page Pinning in the data cache
  - Software instruction caching
    - Got it working and discovered and solved some hard issues [Miller, ASPLOS 06]
    - Good for real time performance, energy
    - But it was a lot of work and a big risk in the project
    - Would not do it again
    - Poor research management to allow a secondary research topic to risk major goals
    - Tilera's Tile Processor has HW instruction cache

Raw Ideas and Decisions: Bit-Level Logic via FPGA Block in Each Tile

- Jettisoned this piece in Raw
- In simulation, could not show performance improvement with it that was not also achievable by using more tiles
- Understood later that this is indeed useful: the real benefit of bit-level configurable logic is in area and energy efficiency [FCCM 2004]
- Tilera's Tile Processor has both subword arithmetic support and special instructions for video and networking
  - These provide area and energy efficiency
    - SAD instruction (sum of absolute differences – motion estimation in video codecs)
    - 16-bit and 8-bit SIMD ops – much better computational density for embedded workloads
    - Checksum instructions for networking

Raw Ideas and Decisions: Distributed ILP (DILP)

- Distributed ILP using Scalar Operand Networks
  - Worked well for apps with statically analyzable ILP (whether irregular or regular)
  - Not so well for apps with unanalyzable mem refs (Turnstiles [ISCA 1999], like SW LSQs)
  - However, there is a bigger question

Will the widespread acceptance of multicores make distributed ILP (ILP across 2 or more cores) more or less important?

- More important
  - Virtually all processors sold are becoming multicore – so running easy-to-program sequential codes across cores will be important
  - Cores are getting simpler, so DILP will be critical to single thread performance
- Less important
  - Everyone is learning how to program in parallel; sequential will go the way of assembly code
  - Intel Webinar: "Think parallel or perish"

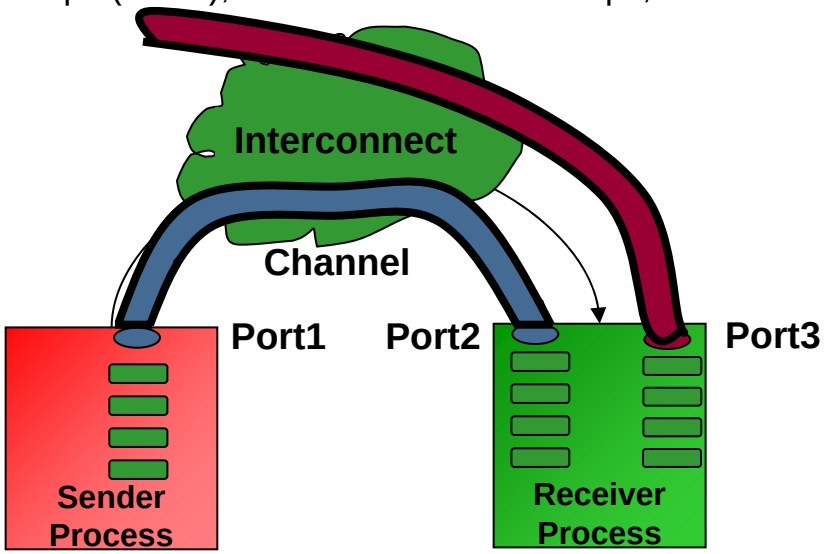
Answer probably somewhere in the middle
Most important apps with lots of parallelism will be rewritten
Legacy apps with low parallelism can benefit from 2X improvement thru DILP

Raw Ideas and Decisions: "Multicore" Nature of Raw

- PC's in every tile was a big win (vs SIMD style)
- Each tile contains a full fledged processor or core – this is what makes it a multicore
- Simple idea, but fundamental to generality and programmability
- Key to supporting existing languages and programming models
- Can take an off-the-shelf program and run it on any tile – enables a gentle-slope programming model
- Key to supporting DILP, TLP, streaming, server workloads

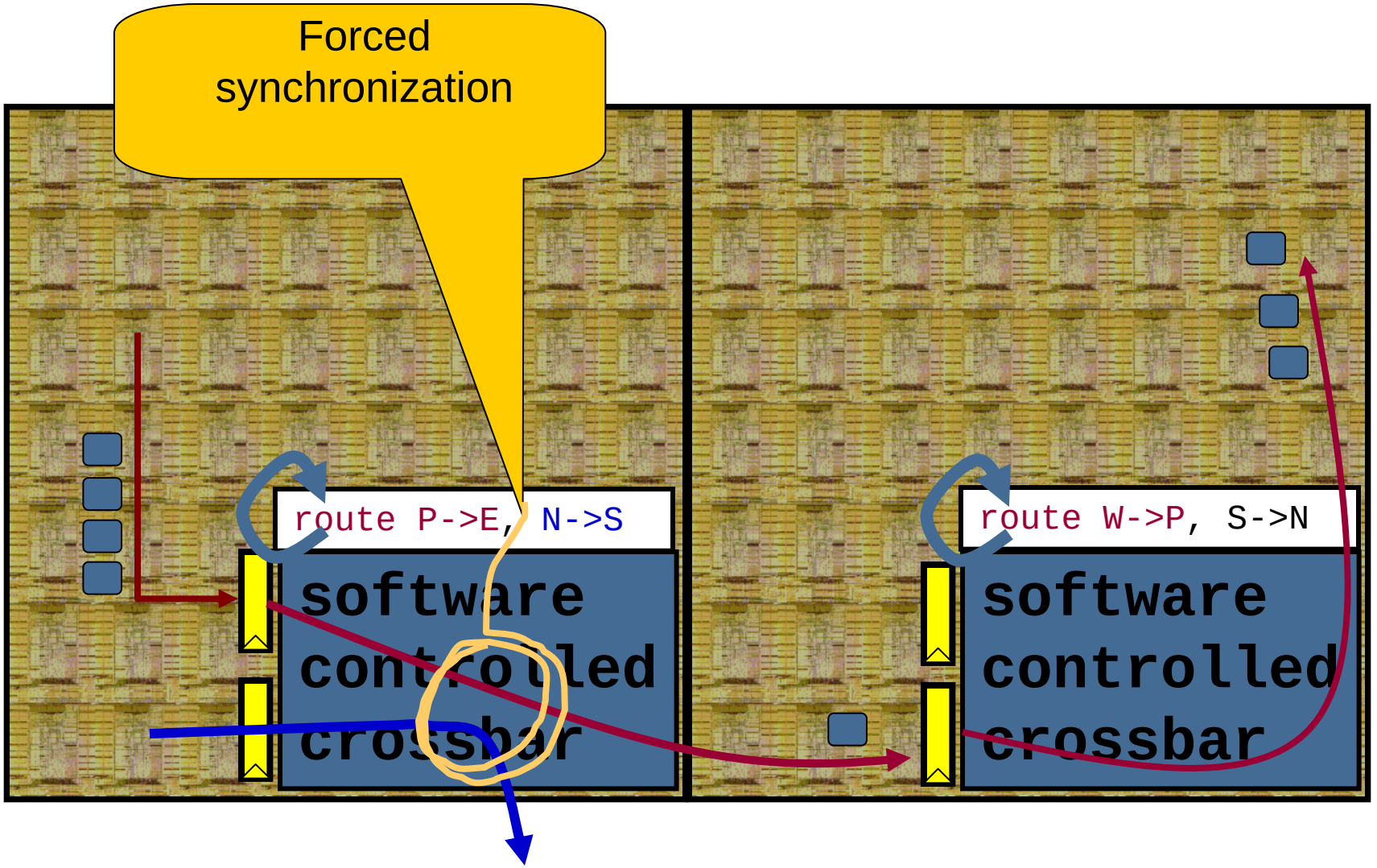
Raw Ideas and Decisions: Streaming – New Language Streamit [ICCC 02]

- Stream processing is a powerful computing paradigm
  - Extremely useful for embedded and scientific applications
  - Tile to tile word transfer 5pJ (90nm), 32KB cache access 30 pJ, DRAM access 500 pJ



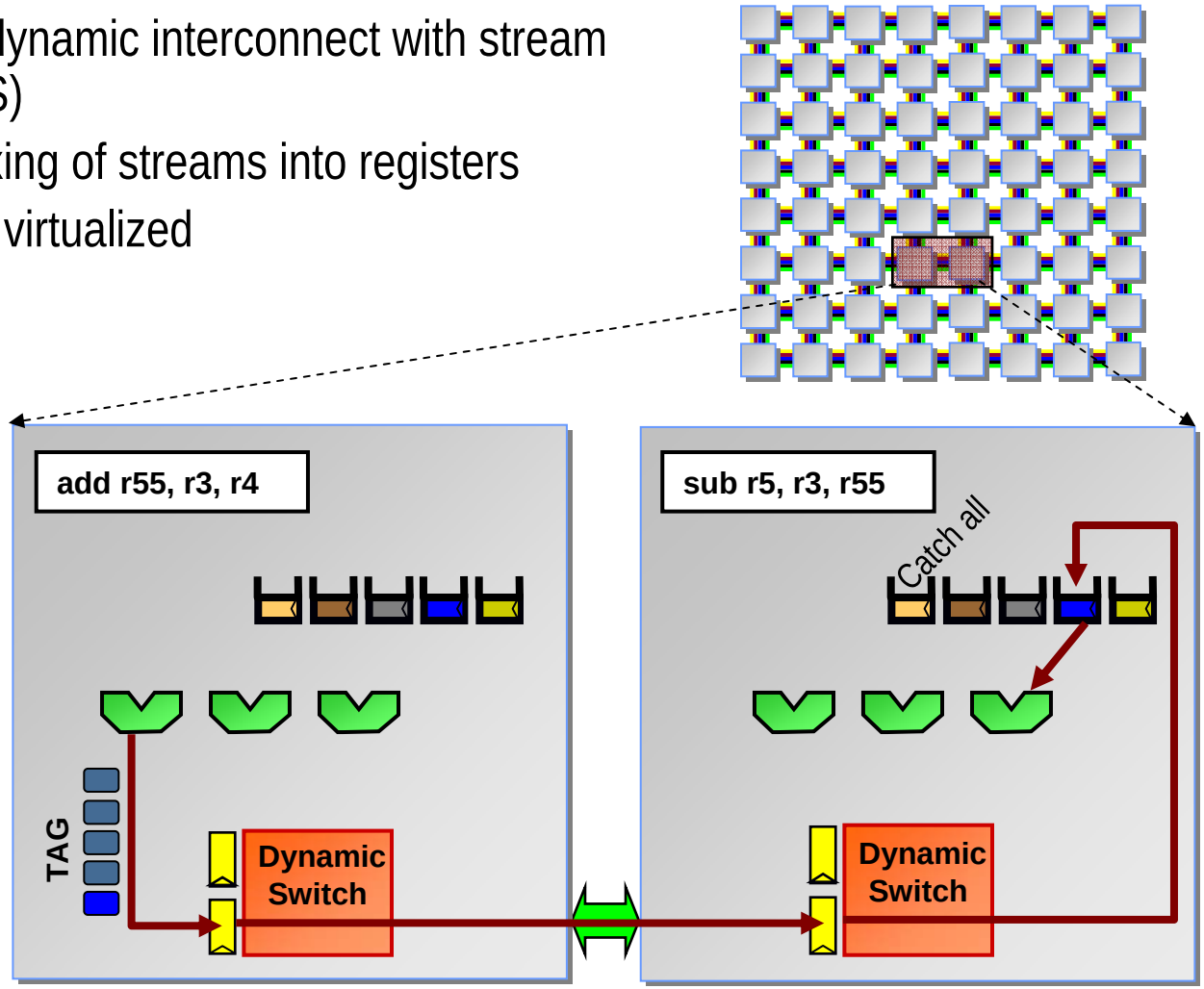
- A new language for streaming?
  - Raw project went with Streamit
  - Hard to make Streamit-like radical language changes in industry
  - Tilera went with ilib: C + lightweight sockets-like stream API

Raw Ideas and Decisions: Streaming – Interconnect Support



Streaming in the Tile Processor

- Streaming done over dynamic interconnect with stream demuxing (AsTrO SDS)
- Automatic demultiplexing of streams into registers
- Number of streams is virtualized



Virtual reality

Simulator reality

Prototype reality

Product reality

Why do we care?
Markets Demanding More Performance

Networking market

- Demand for high performance – 10Gbps
- Demand for more services, intelligence

Security Appliances

Switches

Routers

Digital Multimedia market

- Demand for high performance – H.264 HD
- Demand for more services – VoD, transcode

Cable & Broadcast

Video Conferencing

Surveillance DVR

… and with power efficiency and programming ease

Tilera's TILE64™ Processor

Multicore Performance (90nm)

Number of tiles	64
On chip distributed cache	5 MB
Operations @ 750MHz (32, 16, 8 bit)	144-192-384 BOPS
On chip peak interconnect bandwidth	32 Terabits per second
Bisection bandwidth	2 Terabits per second

Power Efficiency

Power per tile (depending on app)	170 – 300 mW
Core power for h.264 encode (64 tiles)	12W
Clock speed	600-1000 MHz

I/O and Memory Bandwidth

I/O bandwidth	40 Gbps
Main Memory bandwidth	200 Gbps

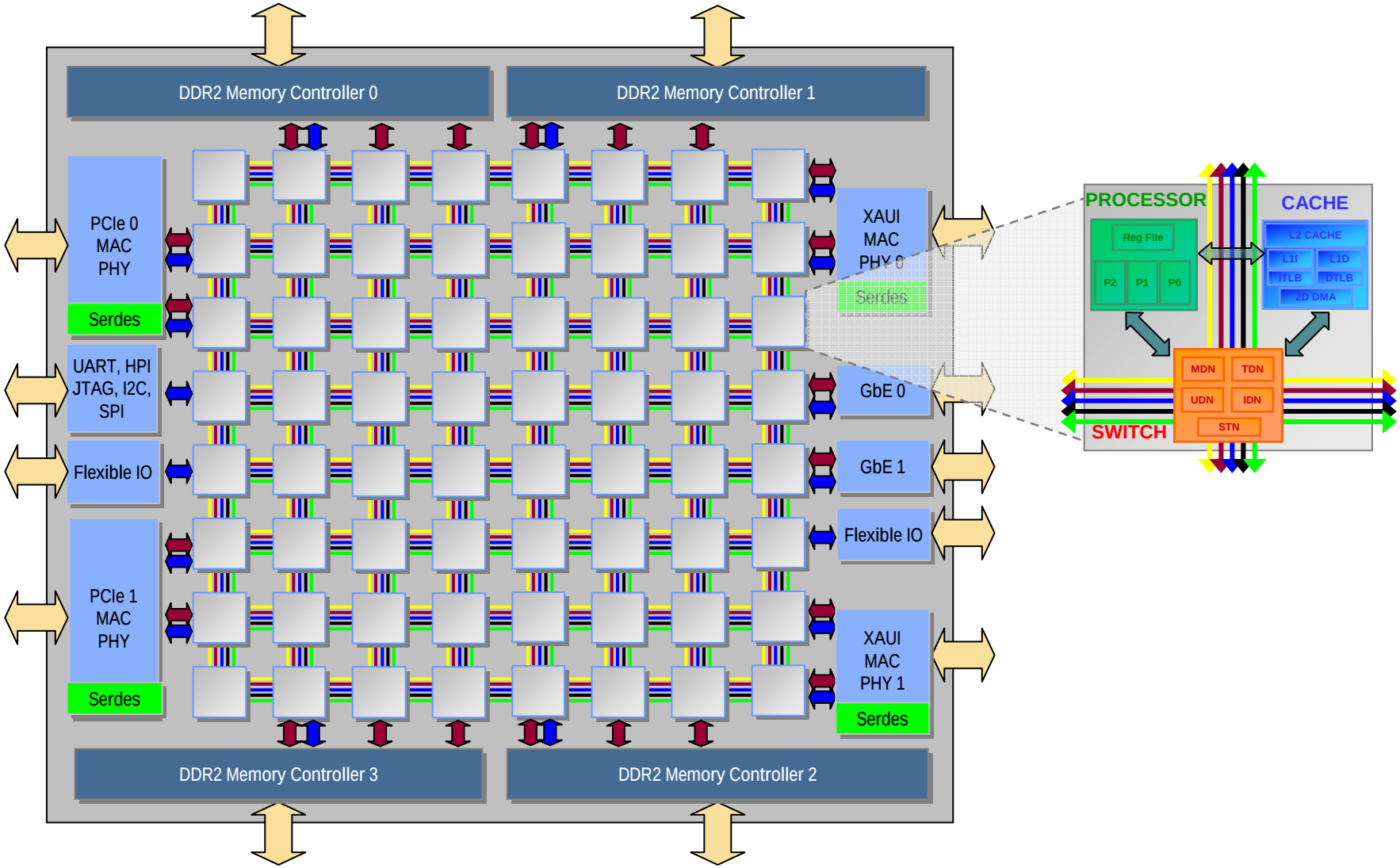
Programming

ANSI standard C
SMP Linux programming
Stream programming

Product reality

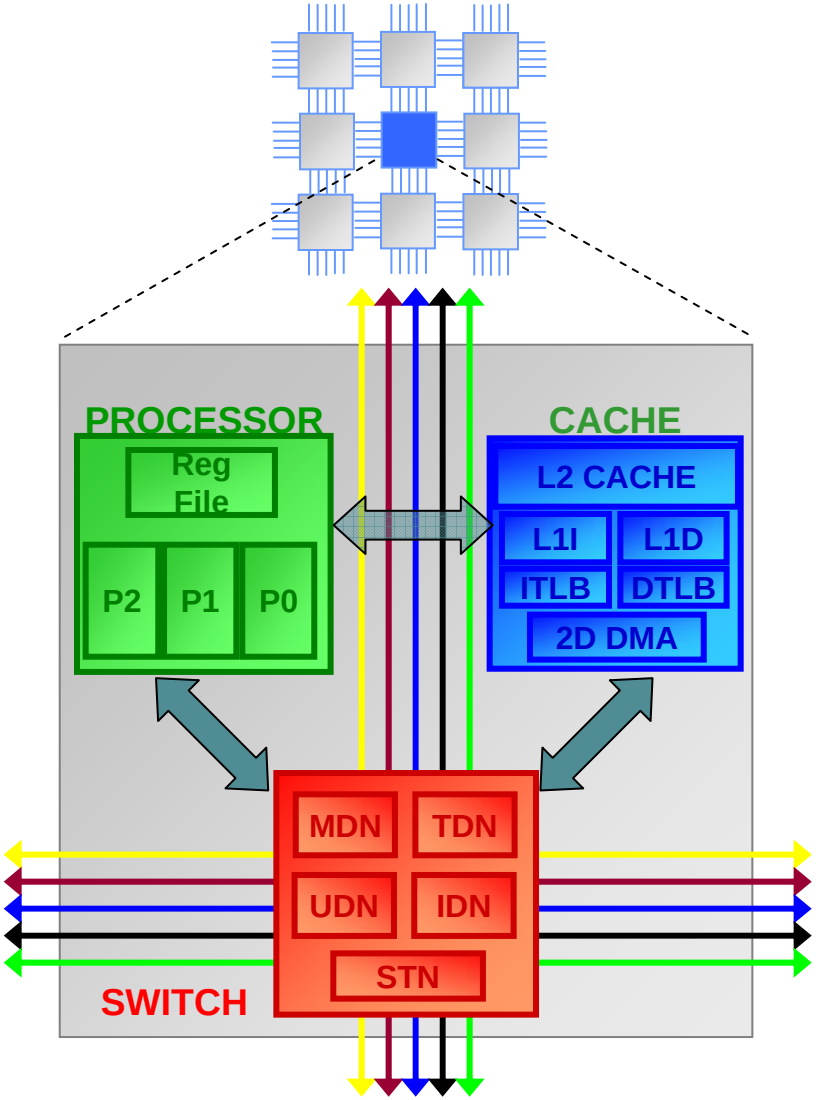
TILE64 Processor Block Diagram

A Complete System on a Chip



Full-Featured General Purpose Cores

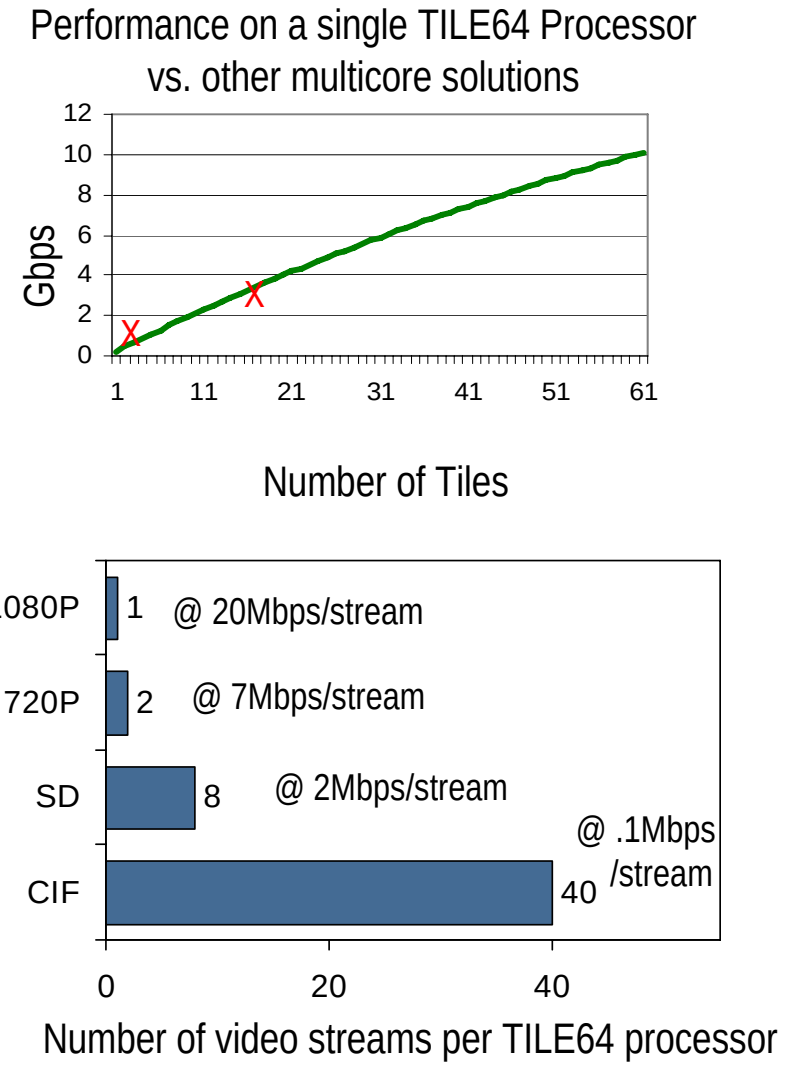
- Processor
  - Homogeneous cores
  - 3-way VLIW CPU, 64-bit instruction size
  - SIMD instructions: 32, 16, and 8-bit ops
  - Instructions for video (e.g., SAD) and networking (e.g., hashing)
  - Protection and interrupts
- Memory
  - L1 cache: 8KB I, 8KB D, 1 cycle latency
  - L2 cache: 64KB unified, 7 cycle latency
  - Off-chip main memory, ~70 cycle latency
  - 32-bit virtual address space per process
  - 64-bit physical address space
  - Instruction and data TLBs
  - Cache integrated 2D DMA engine
- Switch in each tile
- Runs SMP Linux
- 7 BOPS/watt



Performance in Networking and Video

- Performance in networking
  - 10Gbps of SNORT
  - Complete SNORT database
  - Open source SNORT software base

- Performance in video
  - H.264 video encode
  - Encodes 40 CIF video streams @ 30fps
  - Encodes two 720p HD streams @ 30fps
  - PSNR of 35 or more
  - Open source X264 software base



SpecFP does not matter!

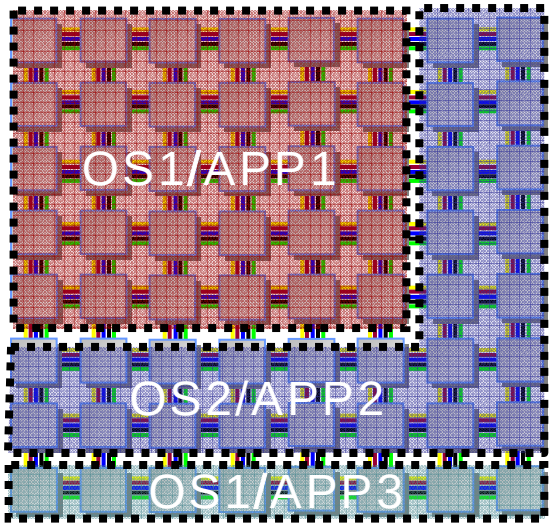
Multicore Hardwall Technology for Protection and Virtualization

The protection and virtualization challenge

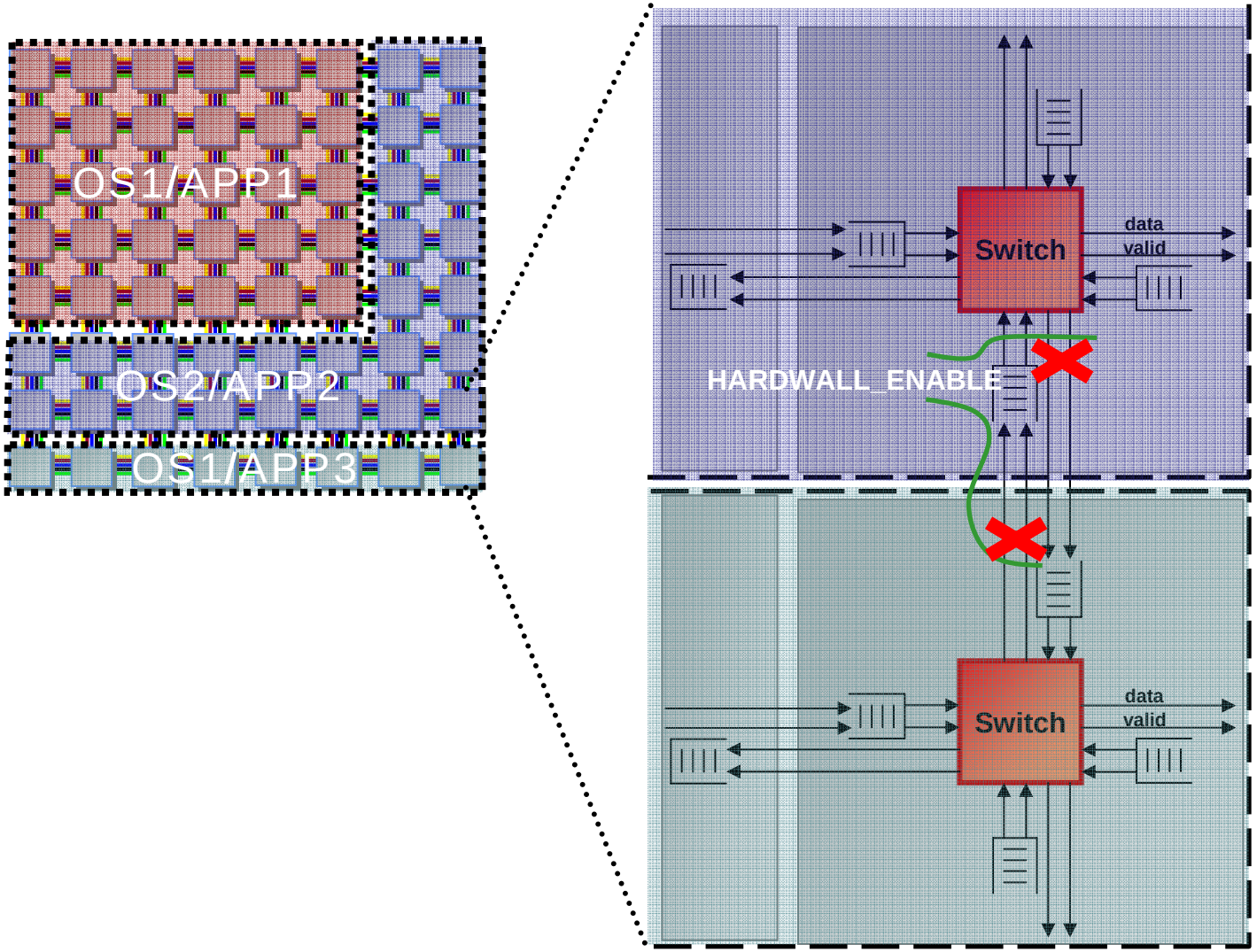
- Multicore interactions make traditional architectures hard to debug and protect
- Memory based protection will not work with direct IO interfaces and messaging
- Multiple OS's and applications exacerbate this problem

Multicore Hardwall technology

- Protects applications and OS by prohibiting unwanted interactions
- Configurable to include one or many tiles in a protected area



Multicore Hardwall Implementation



Product Reality Differences

- Market forces
  - Need crisper answer to “who cares”
  - C + API approach to streaming
  - Also support SMP Linux programming with pthreads
  - Special instructions for video, networking
  - Floating point needed in research project, but not in product for embedded market
- Lessons from Raw
  - E.g., Dynamic network for streams
  - HW instruction cache
  - Protected interconnects
- More substantial engineering
  - 3-way VLIW CPU, subword arithmetic
  - Engineering for clock speed and power efficiency
  - Completeness – I/O interfaces on chip – complete system chip. Just add DRAM for system
  - Support for virtual memory, 2D DMA
  - Runs SMP Linux (can run multiple OSes simultaneously)

Virtual reality

Simulator reality

Prototype reality

Product reality

What Does the Future Look Like?

Corollary of Moore's law: Number of cores will double every 18 months

	'02	'05	'08	'11	'14
Research	**16**	**64**	**256**	**1024**	**4096**
Industry	**4**	**16**	**64**	**256**	**1024**

1K cores by 2014! Are we ready?

(Cores minimally big enough to run a self respecting OS!)

Vision for the Future

- The ‘core’ is the logic gate of the 21st century

Research Challenges for 1K Cores

- 4-16 cores not interesting. Industry is there. University must focus on “1K cores”
- Everything we know about computer architecture, and for that matter, computer systems and programming, is up for grabs
- What should on-chip interconnects look like?
- How do we program 1K cores?
  Intel webinar likens pthreads to the assembly of parallel programming
- Can we add architectural support for programming ease? E.g., suppose I told you cores are free. Can you discover mechanisms to make programming easier?
- Can we use 4 cores to get 2X through DILP?
- What is the right grain size for a core?
- How must our computational models change in the face of small memories per core?
- Locality WILL matter for 1K cores. Compilers and OSes will need to manage locality, and languages will need to expose it
- How to “feed the beast”? I/O and external memory bandwidth
- Can we assume perfect reliability any longer?

Computer architecture is hot!