Stream Architectures

Saman Amarasinghe and William Thies
Massachusetts Institute of Technology

PACT 2003
September 27, 2003

Schedule

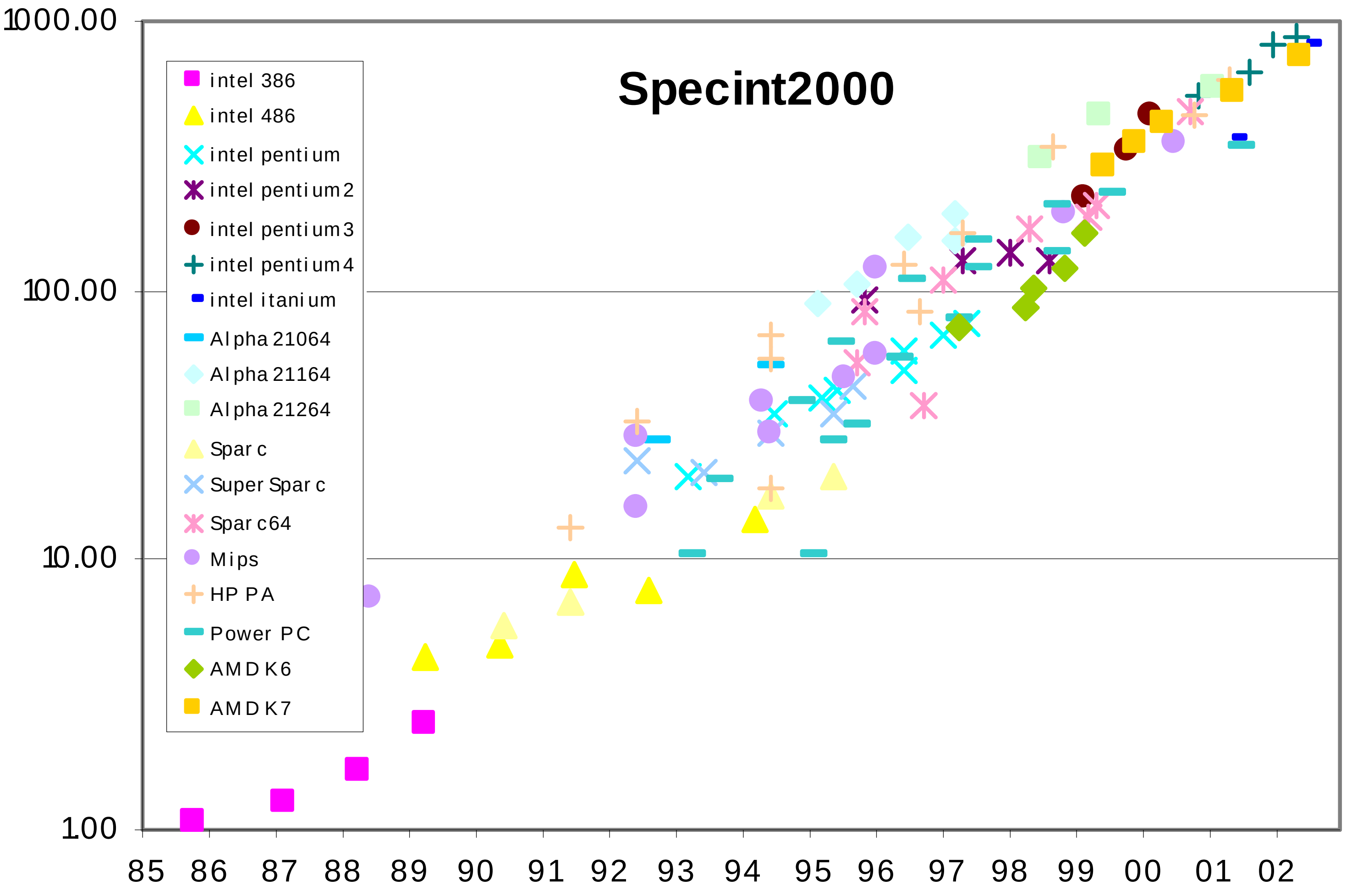
1:30-1:40	Overview (Saman)
1:40-2:20	Stream Architectures (Saman)
2:20-3:00	Stream Languages (Bill)
3:00-3:30	Break
3:30-3:55	Stream Compilers (Saman)
3:55-4:20	Domain-specific Optimizations (Saman)
4:20-5:00	Scheduling Algorithms (Bill)

Processor Model

- User model has been very stable for 30 years
  - Sequentially executes instructions
  - IO operations interact with outside world
- Model has hidden the scaling of technology
  - Efficiently transformed transistors to performance
  - 8008 – 3,500 transistors, and ran at 200kHz
  - P4 – 42M transistors, runs at 3GHz
  - Performance changed from 0.06MIPS to >1000MIPS
- C is a perfect fit to this programming model
  - They grew up together …

This slide compliments of Mark Horowitz, Stanford University

Supporting Data



This slide compliments of Mark Horowitz, Stanford University

The World is About to Change

- Processor performance will not continue to scale
  - We will fall off the current performance curve soon
- Many factors will cause this to occur
  - VLSI wire issues (global structure are hard to build)
  - Insufficient recoverable ILP
  - Power

- This performance growth was partially an illusion

This slide compliments of Mark Horowitz, Stanford University

Technology Scaling

- Scaling CMOS has two direct effects:

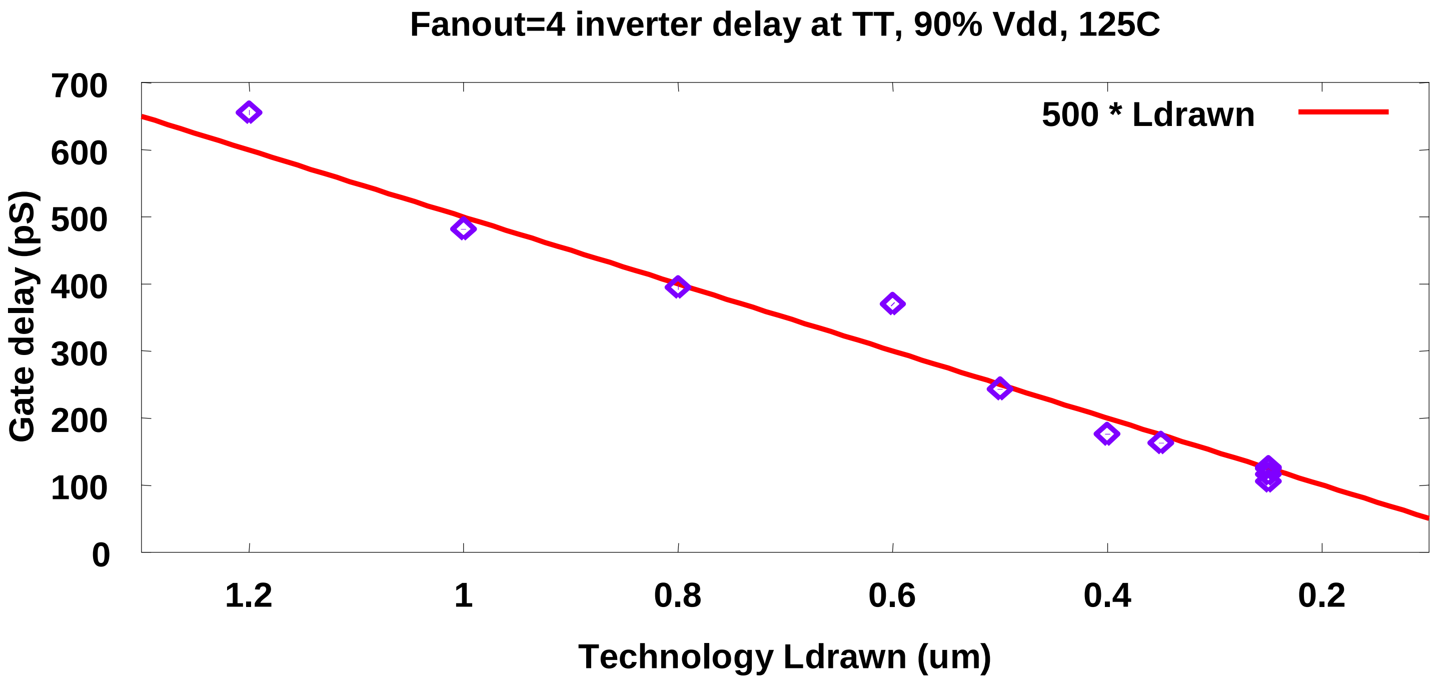
- Devices get smaller
  - Both transistors and wires
  - Get more per square mm
  - Generally means they get cheaper
  - Enables more complex devices

- Transistors get faster
  - So do wires when viewed the ‘right’ way

This slide compliments of Mark Horowitz, Stanford University

FO4 Inverter Delay Under Scaling

- Gate delay varies linearly with process technology (so far)
- Useful rule of thumb: Dgate = 500pS*Ldrawn at TTLH

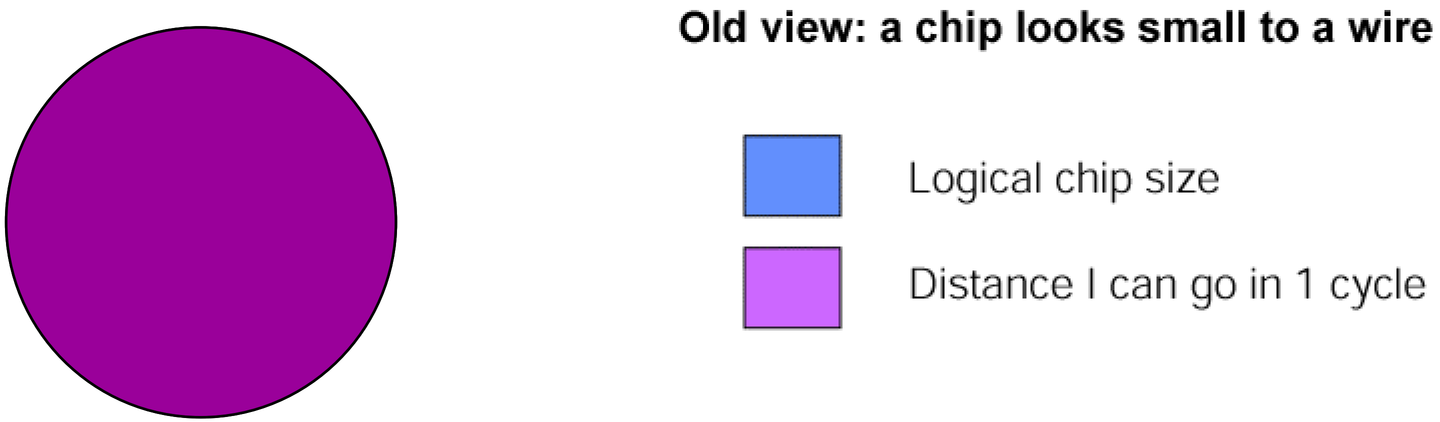


- Issues with being able to continue this scaling
  - Some current technologies are faster (le < feature size)

This slide compliments of Mark Horowitz, Stanford University

The World is Growing

- The problem associated with wires is really due to complexity
- Diagram shows the logical span you reach in a cycle
  - It also show the logical span of a chip

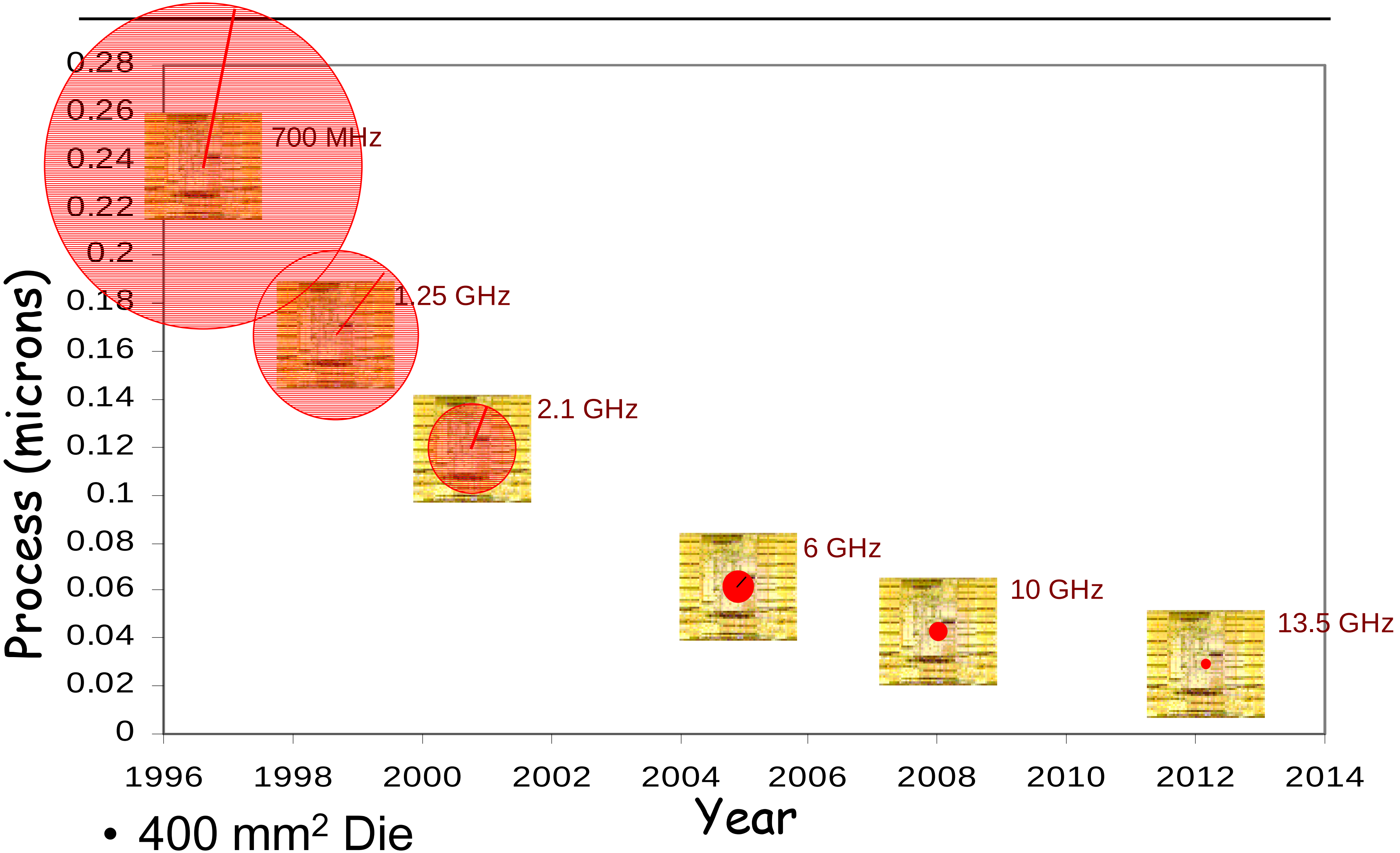


Old view: a chip looks small to a wire

🟦 Logical chip size

🟪 Distance I can go in 1 cycle

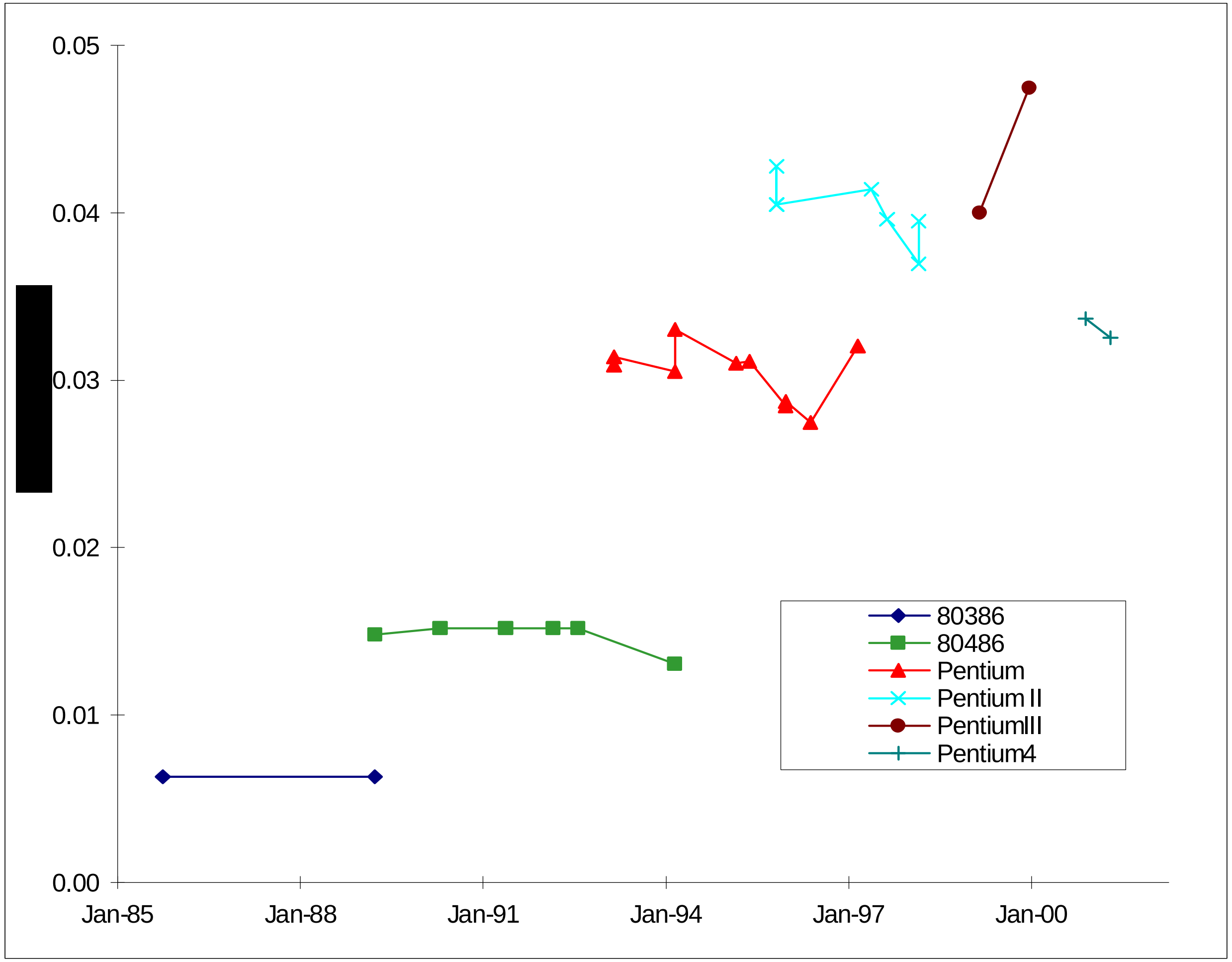
Range of a Wire in One Clock Cycle



- 400 mm^2 Die
- From the SIA Roadmap

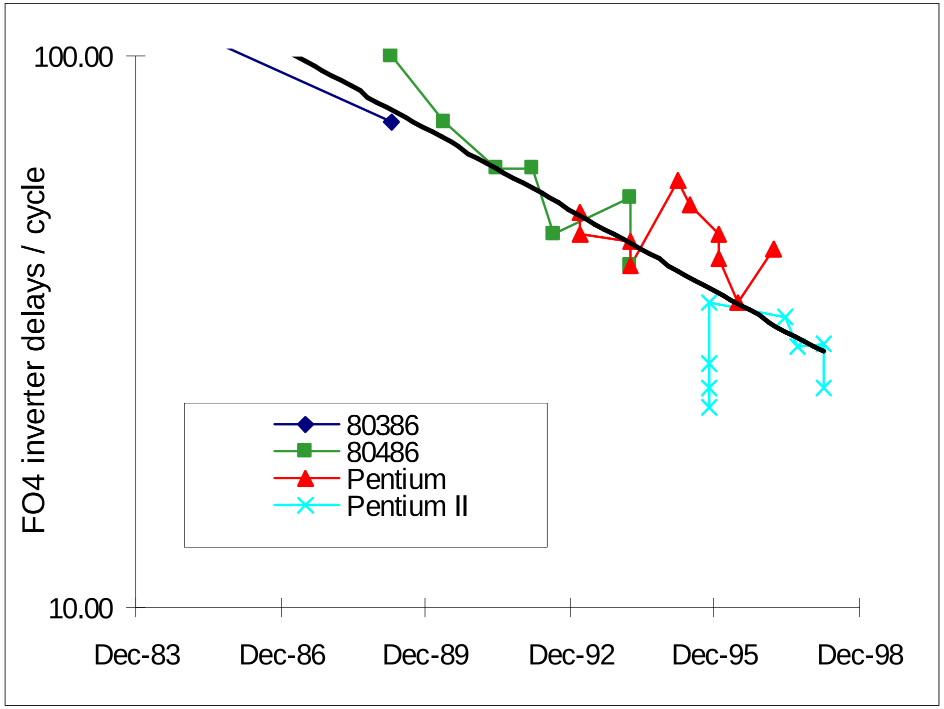
Architecture Scaling

- Plot of IPC
  - Compiler + IPC
  - 1.5x / generation
  - Until PIII, now falling
- There is a lot of hardware to make this happen
  - Many transistors
  - Lots of power
  - Lots of designers



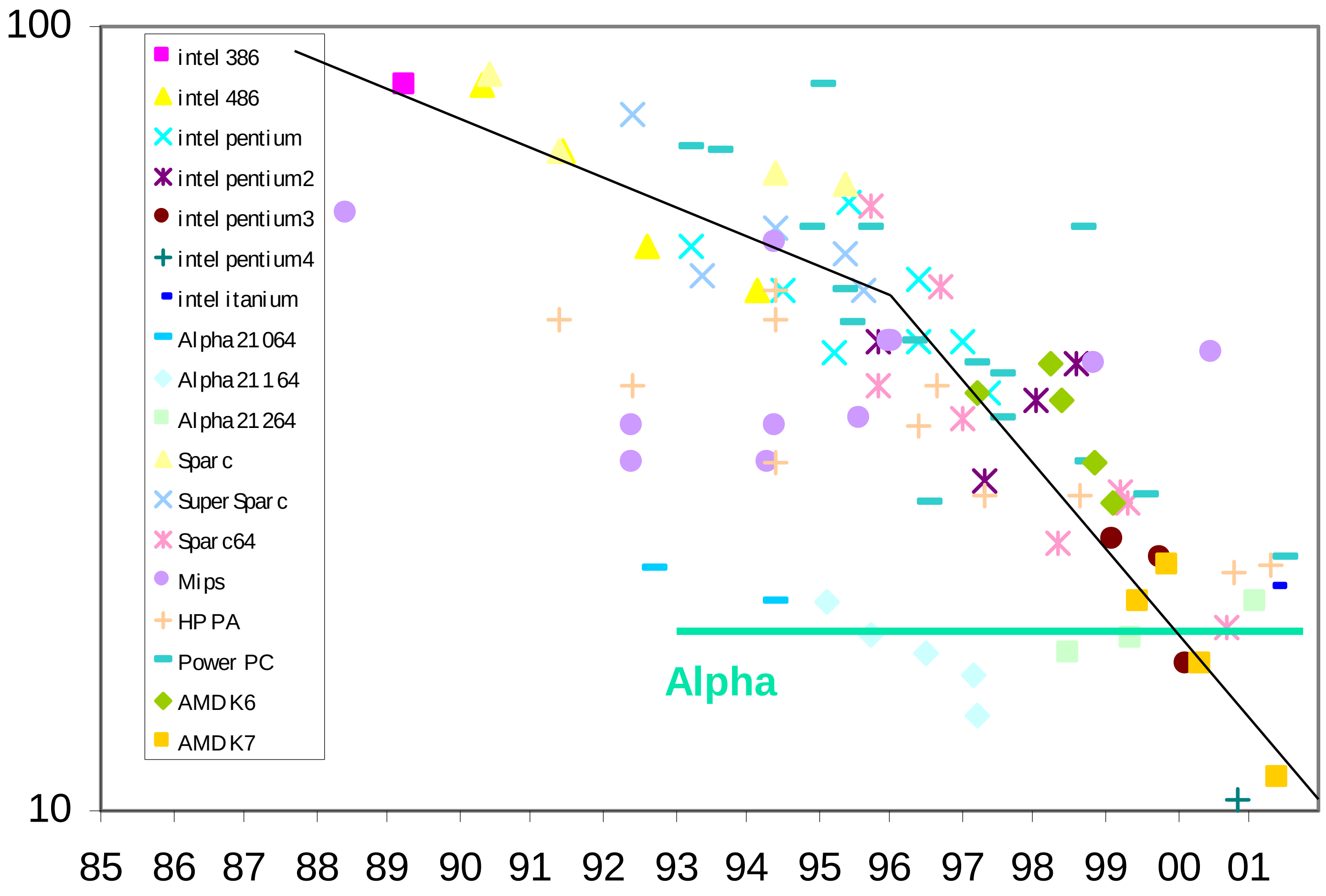
Gates Per Clock

- Clock speed has been scaling faster than base technology
- Number of FO4 delays in a cycle has been falling



- Number of gates decrease 1.4x each generation
- Caused by:
  - Faster circuit families (dynamic logic)
  - Better optimization
  - Better micro-architecture
  - Better adder/mem arch
- All this generally requires more transistors

Clock Cycle in ‘FO4’



Note that the points at around 10 FO4 are not correct. The FO4 for these technologies is about ½ my simple formula

Gates Per Clock

- Current SOA machines are at 16 FO4 gates per cycle
  - Historical low values (Cray) were at this level
- Overhead for short tick machines grows rapidly
  - Power
    - Increases clock power per logic function
  - Latency
    - Flops are already 10-20% of cycle today
  - Logic reach grows smaller
    - What fits in a cycle (how many bits/gates) decreases
- Difficult to generate a clock at less than 8 FO4 gates
- Continued scaling of gates/clock will be hard
  - Performance gain from 16 FO4 to 8 FO4 is only 20% anyhow

This slide compliments of Mark Horowitz, Stanford University

Coming Opportunity

- Conventional processor scaling is going to slow down
  - Design costs are enormous
  - Improving IPC is getting harder
  - Improving cycle time is getting harder

- For performance need to exploit parallelism
  - EV8, Pentium 4 – SMT
  - Power 4 is an explicit multiprocessor
    - Power 5 is explicit multiprocessor with SMT

- How do we do this well?
  - Create other programming models
  - Make the models match VLSI constraints
  - Don't worry about universality

This slide compliments of Mark Horowitz, Stanford University

Making Communication Explicit

- In VLSI, communication is what matters

  It is the wires, stupid

- Another way of saying this is:
  - In VLSI building computation elements is easy
  - Keeping them feed is hard
  - Hence, most of a modern processor stages data
- What a computation model that
  - Makes communication explicit
  - Provides feedback to the programmer about communication

This slide compliments of Mark Horowitz, Stanford University

The “Ideal” VLSI Machine

- Lots of simple compute units
  - Units feed by cheap (in energy, area) sources – local regs
  - Relatively cheap instruction issue logic
- Memory (FIFOs) to decouple data fetch/execute
  - Communication takes time (it is the LAW)
  - Need to enable the machine to tolerate latency
- Interconnection network with high-bandwidth
  - And as small latency as possible
- Connections to large backing store
  - Main memory and disk

- Streams are a programming model that matches this machine

This slide compliments of Mark Horowitz, Stanford University

Next-Generation Architectures

- The new design space
  - How to use a billion transistors?
  - How to accommodate the wire delays?

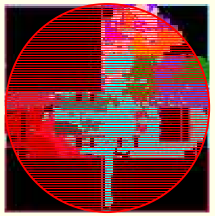
- Many forward looking architecture are addressing this problem
  - MIT Raw processor
  - Stanford Imagine processor
  - Stanford Smart Memories processor
  - UT Austin TRIPS processor
  - Wisconsin ILDP architecture
  - The original IBM BlueGene processor

Next-Generation Architectures

- MIT Raw processor
- Stanford Imagine processor
- UT Austin TRIPS processor
- Berkeley VIRAM Processor

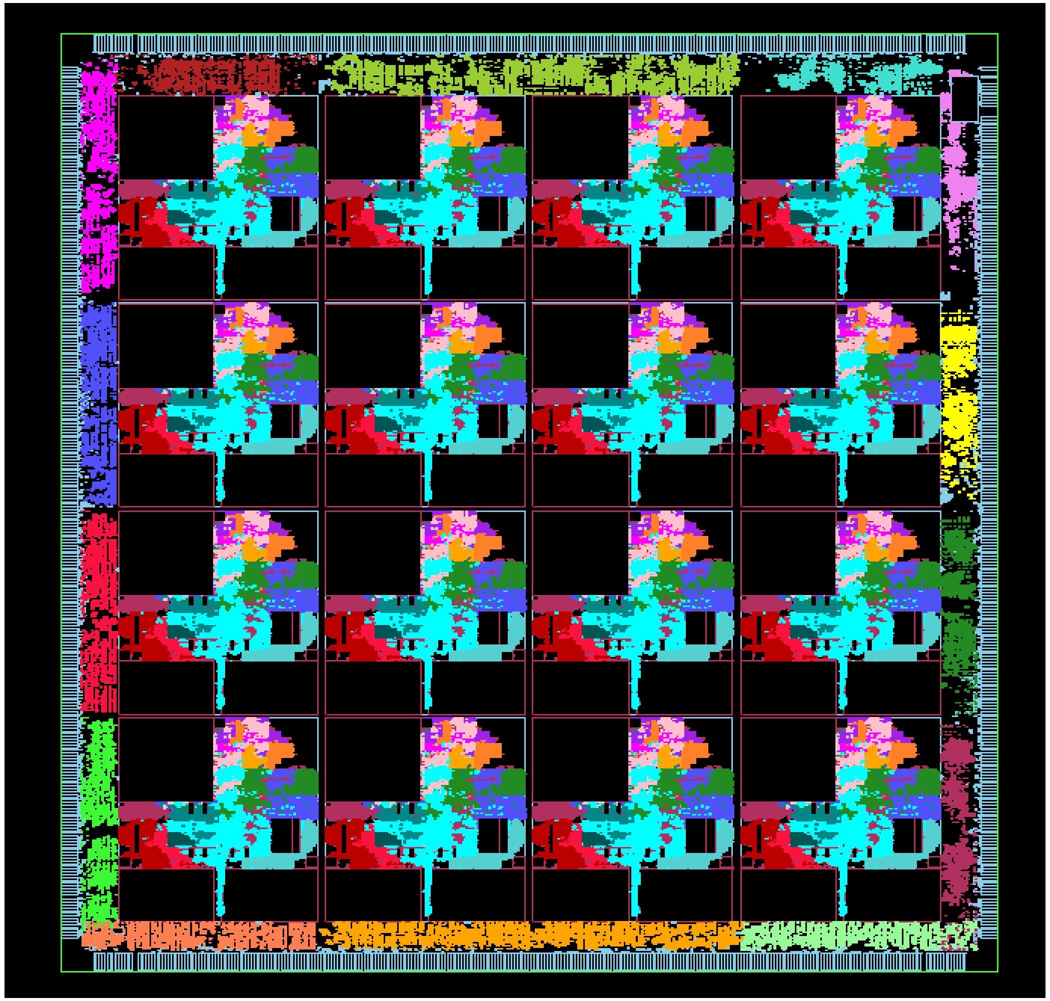
Wire Delay

Make a tile as big as you can go in one clock cycle, and expose longer communication to the programmer

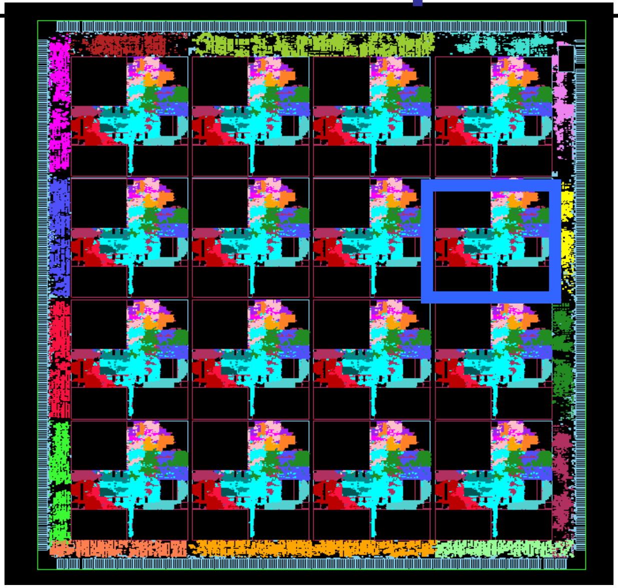


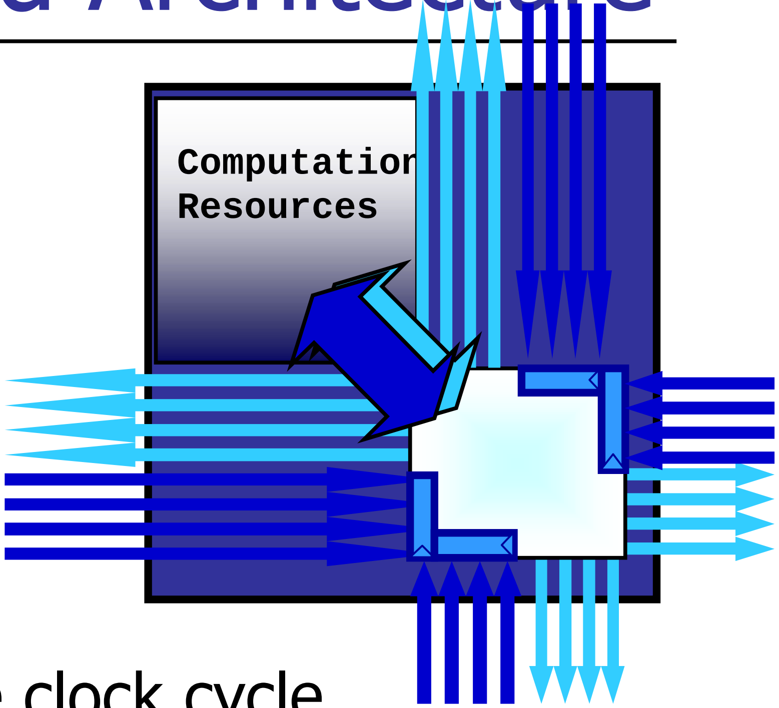
Wire Delay and Tiled Architectures

Make a tile as big as you can go in one clock cycle, and expose longer communication to the programmer



RAW: A Wire Exposed Architecture

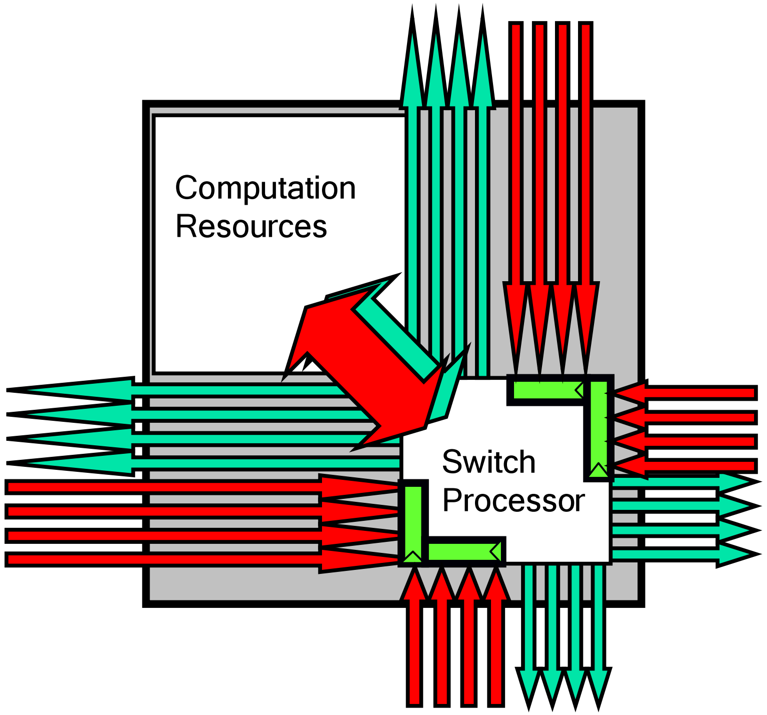




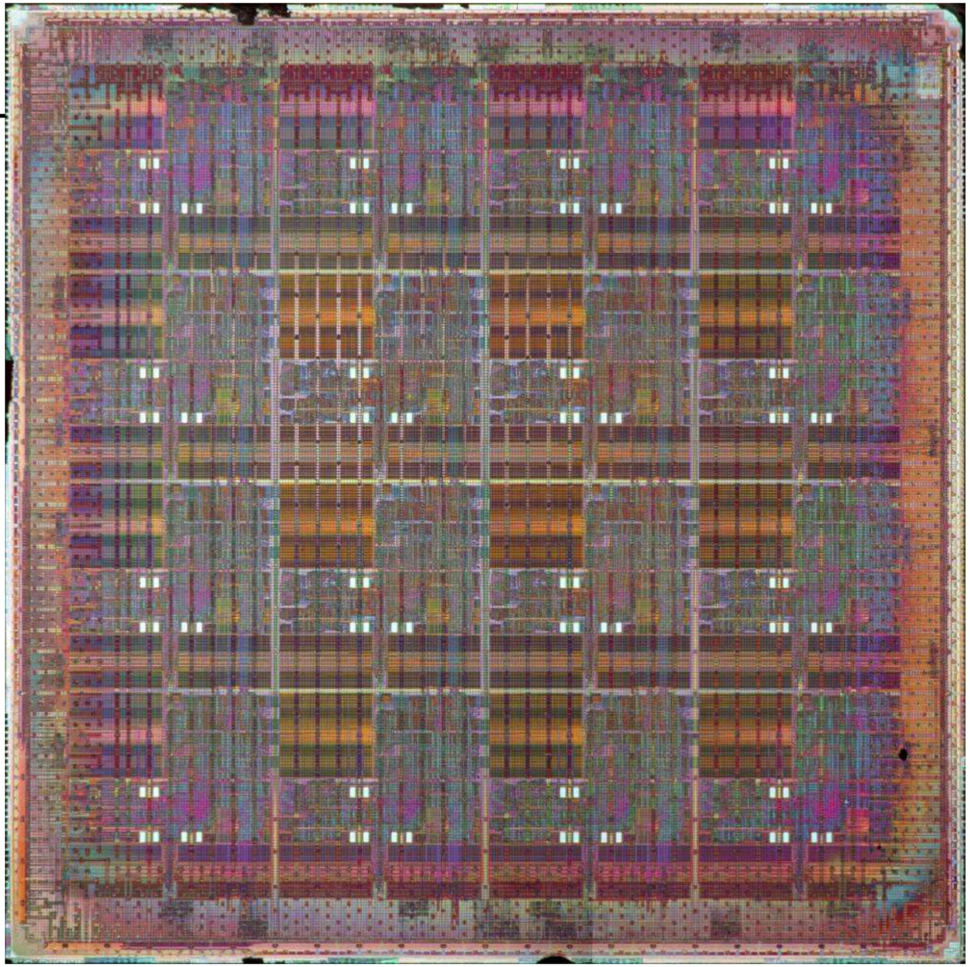
- A wire can cross a tile in a single clock cycle
  - Wire delay is not a issue in the processor design
- Ultra fast interconnect network
  - Exposes the wires to the compiler
  - Compiler orchestrate the communication → hide wire delay
- **Defying the Speed of Light**

On-Chip Networks

- 2 Static Networks
  - Software configurable crossbar
  - 3 cycle latency for nearest-neighbor ALU to ALU
  - Must know pattern at compile-time
  - Flow controlled
- 2 Dynamic Networks
  - Header encodes destination
  - Fire and Forget
  - 15 cycle latency for nearest-neighbor



Raw Chip



IBM SA-27E .15u 6L Cu

18.2 mm x 18.2 mm

16 Flops/ops per cycle

208 Operand Routes / cycle

2048 KB L1 SRAM

1657 Pin CCGA Package

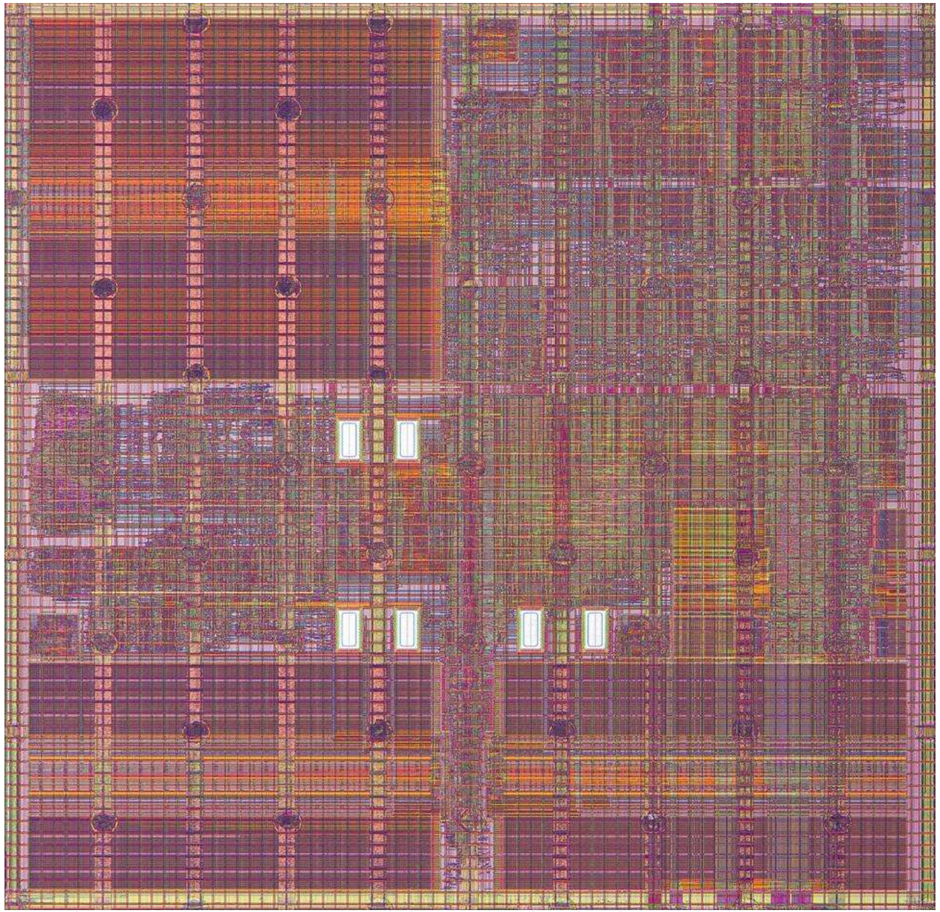
1080 HSTL core-speed signal I/O

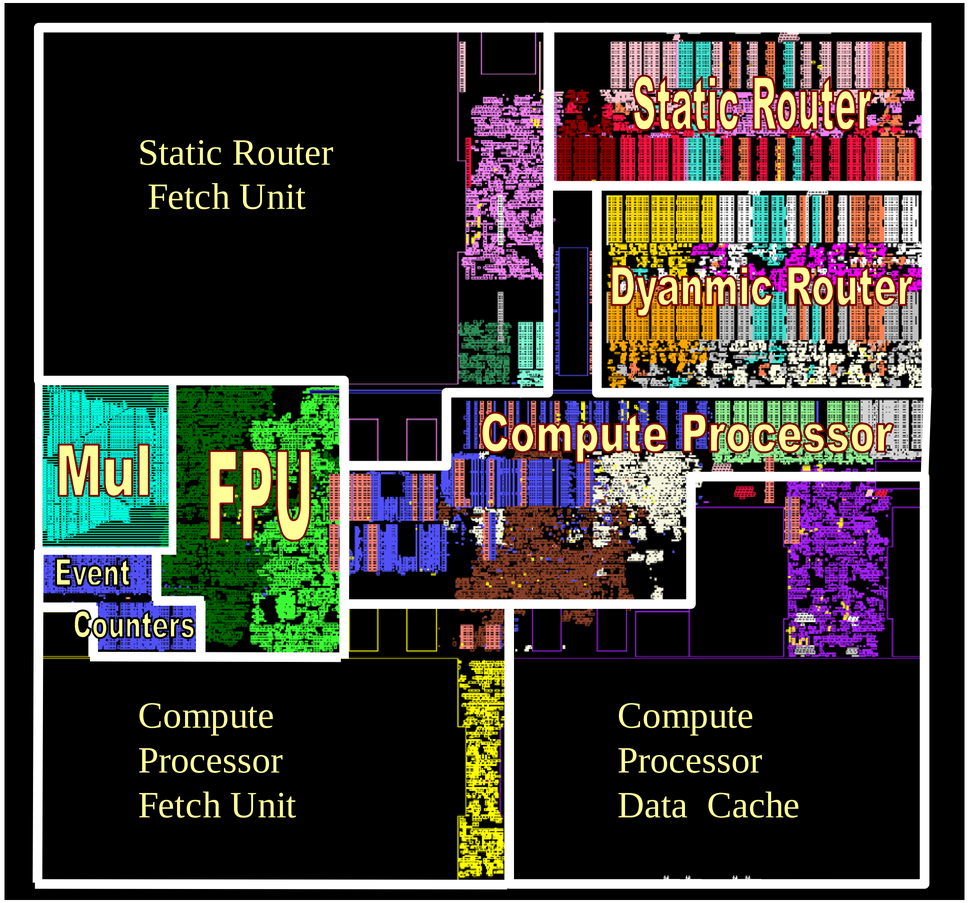
@ 225 MHz Worst Case (Temp, Vdd,

420 MHz Achieved!

6.7 Peak GFLOPS (without FMAC!)
420 Gb/s on-chip bisection bandwidth

Close-up of a single Raw tile





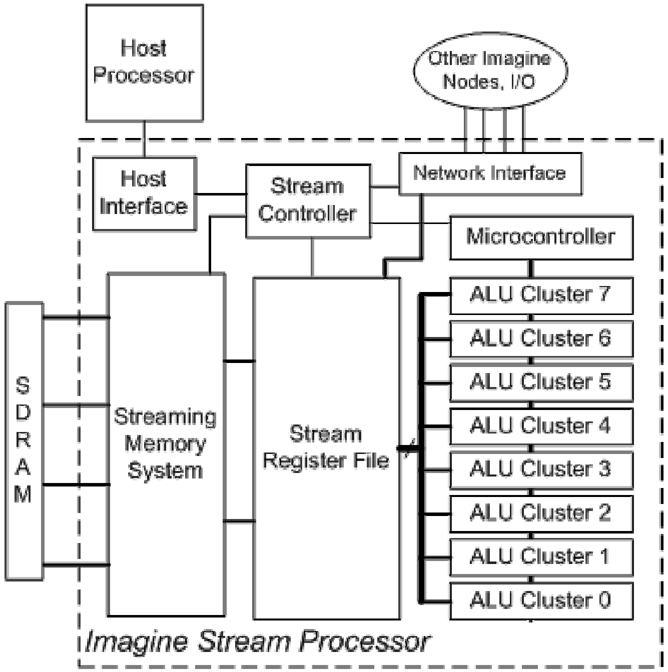
Raw vs. Pentium

	Pentium 4	Raw Prototype
Process	Intel .18u 6L Al	IBM .15u 6L Cu
Issue Width	3	16
On-chip RAM	360 KB	2048 KB
Memory Ports	1 load, 1 store	16 of either
Size	217 mm^2	330 mm^2
Transistors	.042 Billion	.122 Billion
Signal Pins	212	1152
Clock Speed	2.2GHz	420MHz

Next-Generation Architectures

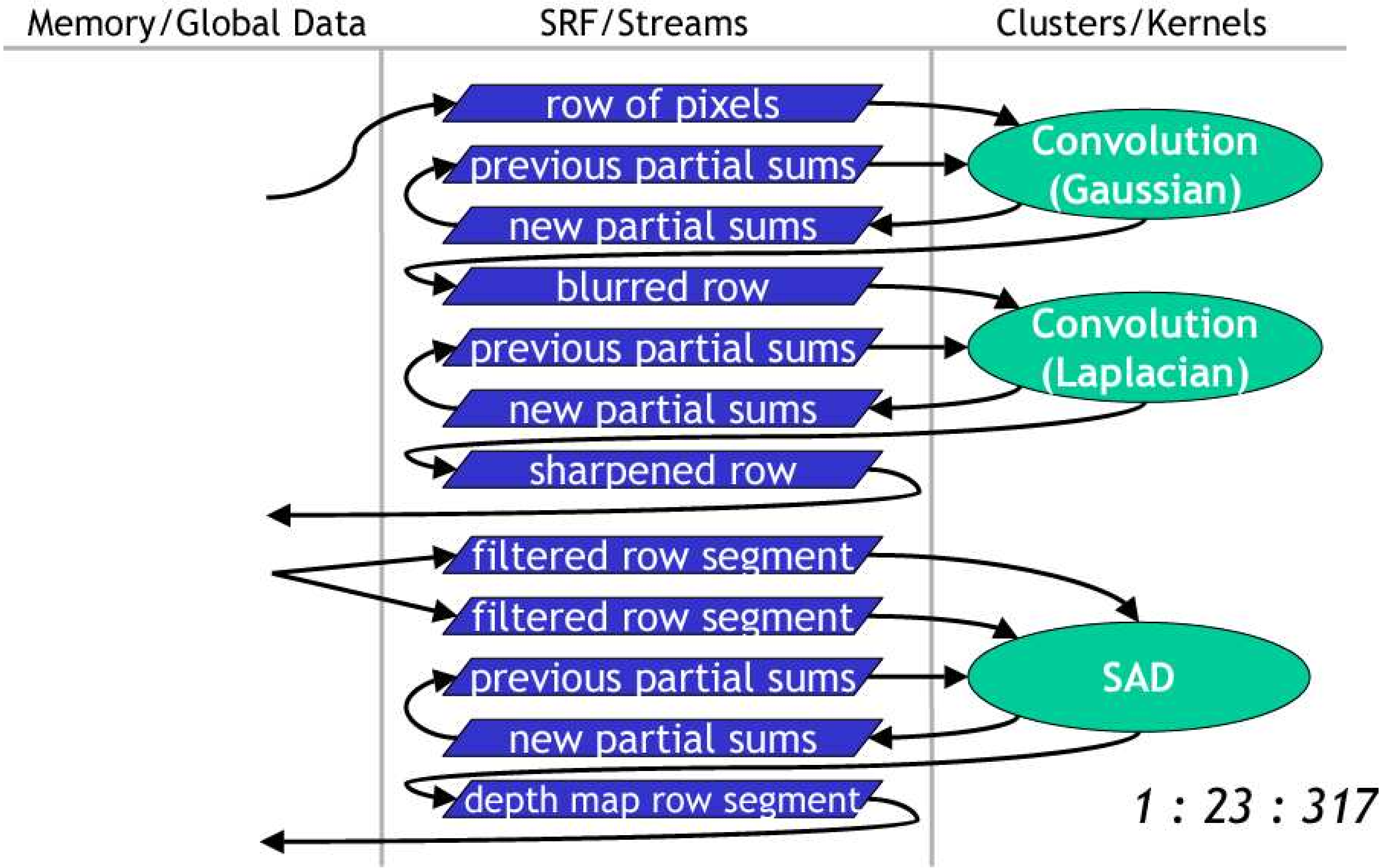
- MIT Raw processor
- Stanford Imagine processor
- UT Austin TRIPS processor
- Berkeley VIRAM Processor

Imagine Stream Processor (Stanford)

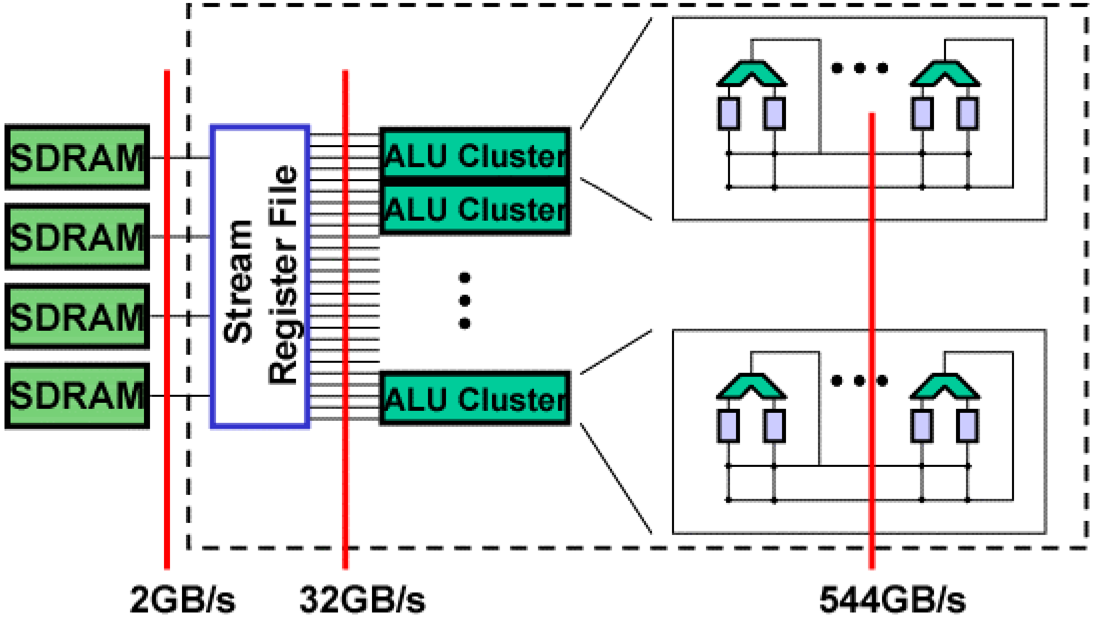


- 48 FP arithmetic units
  - In 8 VLIW clusters
  - SIMD control of clusters
  - Execute "stream kernels"
- To keep ALUs busy, streams of data are buffered in Stream Register File (SRF)
  - SRF is compiler-controlled, on-chip memory
  - 128 KB – can hold large streams of data
  - Distinguishes Imagine from plain vector processor
- Kernel execution, DMA operations, and SRF allocation is orchestrated by control processor

Producer-Consumer Locality in a Depth Extractor



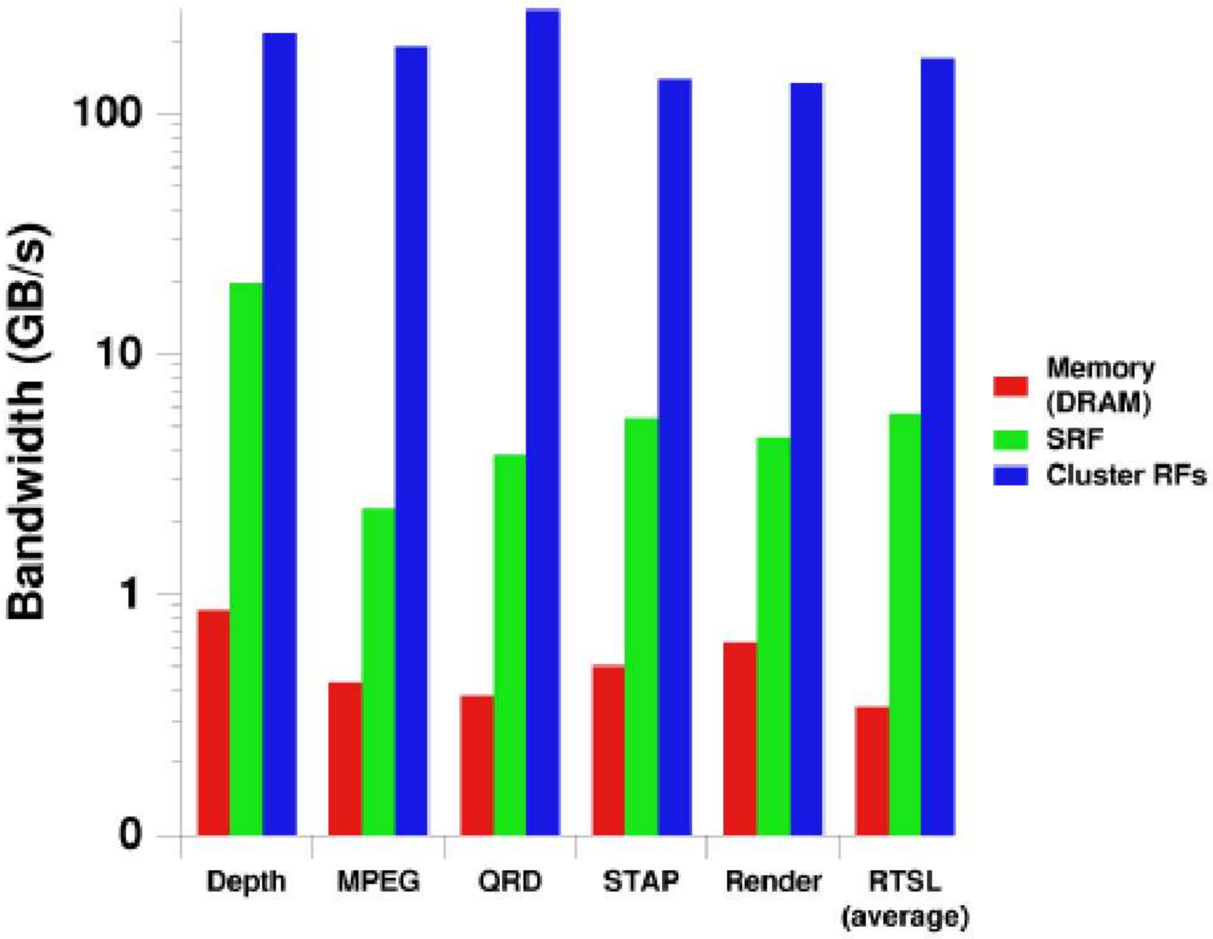
A Bandwidth Hierarchy exploits kernel and producer-consumer locality



	Memory BW	Global RF BW	Local RF BW
Depth Extractor	0.80 GB/s	18.45 GB/s	210.85 GB/s
MPEG Encoder	0.47 GB/s	2.46 GB/s	121.05 GB/s
Polygon Rendering	0.78 GB/s	4.06 GB/s	102.46 GB/s
QR Decomposition	0.46 GB/s	3.67 GB/s	234.57 GB/s

This slide compliments of William Dally, Stanford University

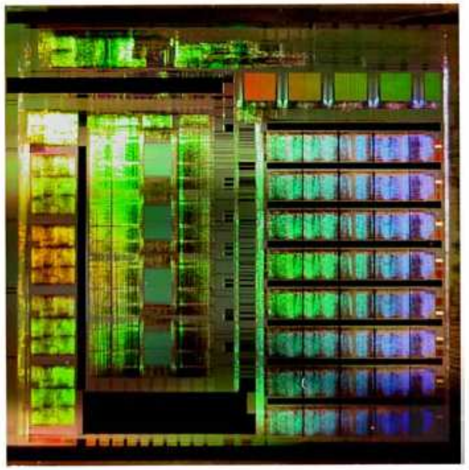
Bandwidth demand of stream programs fits bandwidth hierarchy of architecture



This slide compliments of William Dally, Stanford University

Prototype HW and SW

- Prototype of Imagine architecture
  - Proof-of-concept 2.56cm^2 die in 0.15um TI process, 21M transistors
  - Collaboration with TI ASIC
- Dual-Imagine development board
  - Platform for rapid application development
  - Test & debug building blocks of a 64-node system
  - Collaboration with ISI-East
- Software tools based on Stream-C/Kernel-C
  - Stream scheduler
  - Communication scheduling
- Many Applications
  - 3 Graphics pipelines
  - Image-processing apps – depth, MPEG
  - 3G Cellphone (Rice)
  - STAP

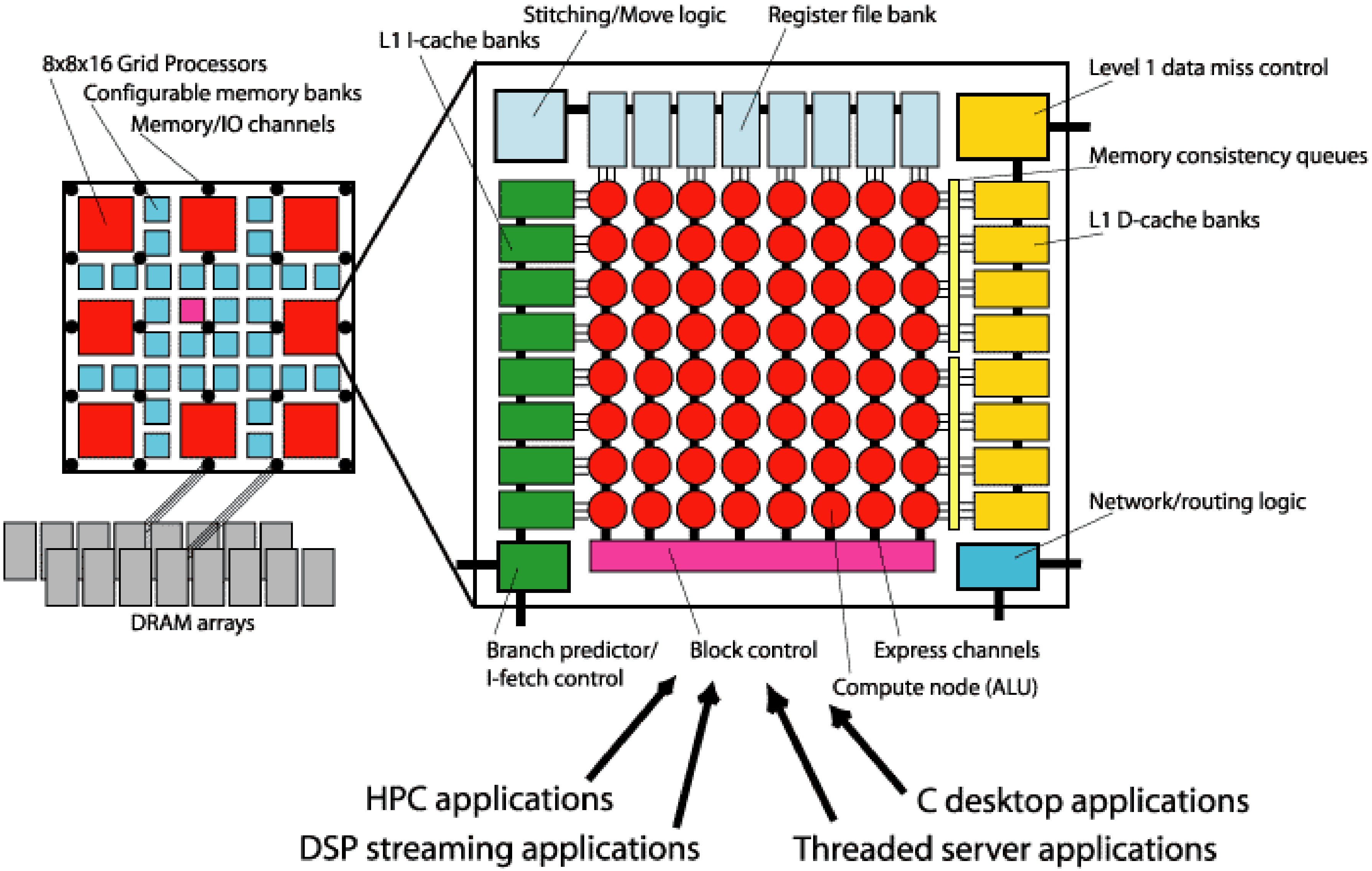


Architectures, Languages, and Compilers for the Streaming Domain
This slide compliments of William Dally, Stanford University

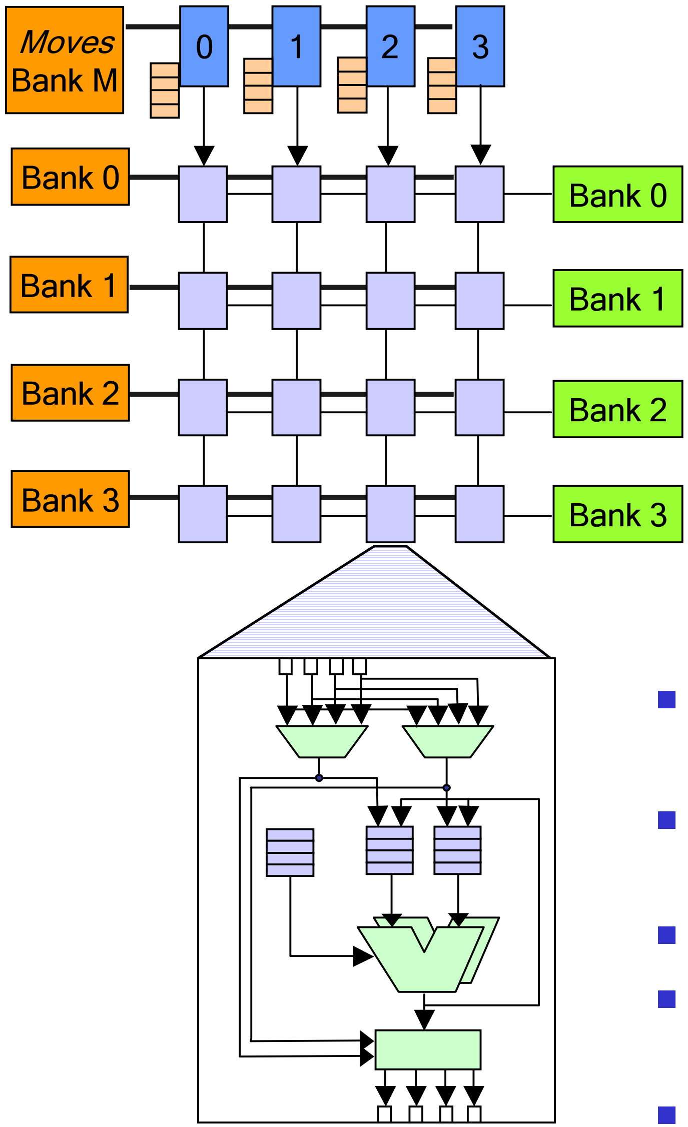
Next-Generation Architectures

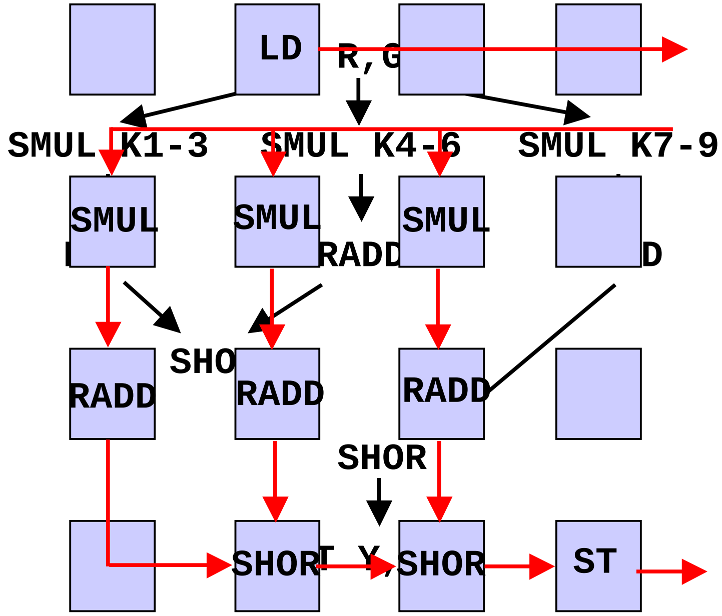
- MIT Raw processor
- Stanford Imagine processor
- UT Austin TRIPS processor
- Berkeley VIRAM Processor

TRIPS (UT Austin)



Dataflow Execution in TRIPS Core

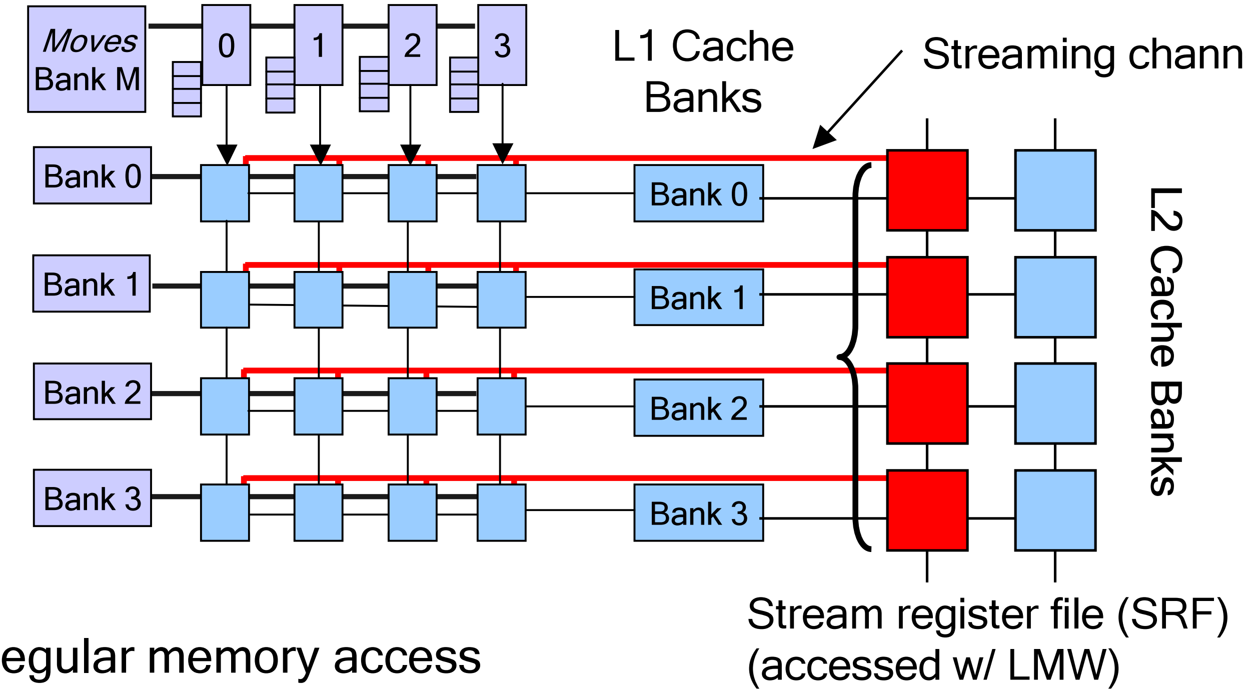




- SPDI: Static Placement, Dynamic Issue
  - Instructions execute in dataflow order
- Instructions stream in from left, data from right
- ALU Chaining
- Pipeline instruction distribution and execution
- Static unrolling for latency tolerance

Architectures, Languages, and Compilers for the Streaming Domain
This slide compliments of Steve Keckler, UT Austin

TRIPS Memory Accesses



- Irregular memory access
  - Map to hardware cache hierachy
- Regular data accesses
  - Subset of L2 cache banks configured as Stream Register File (SRF)
  - High bandwidth data channels to SRF, reduced address BW
  - DMA engines transfer between SRF and DRAM or other SRFs
- Constants saved in reservation stations with corresponding instructions

This slide compliments of Steve Keckler, UT Austin

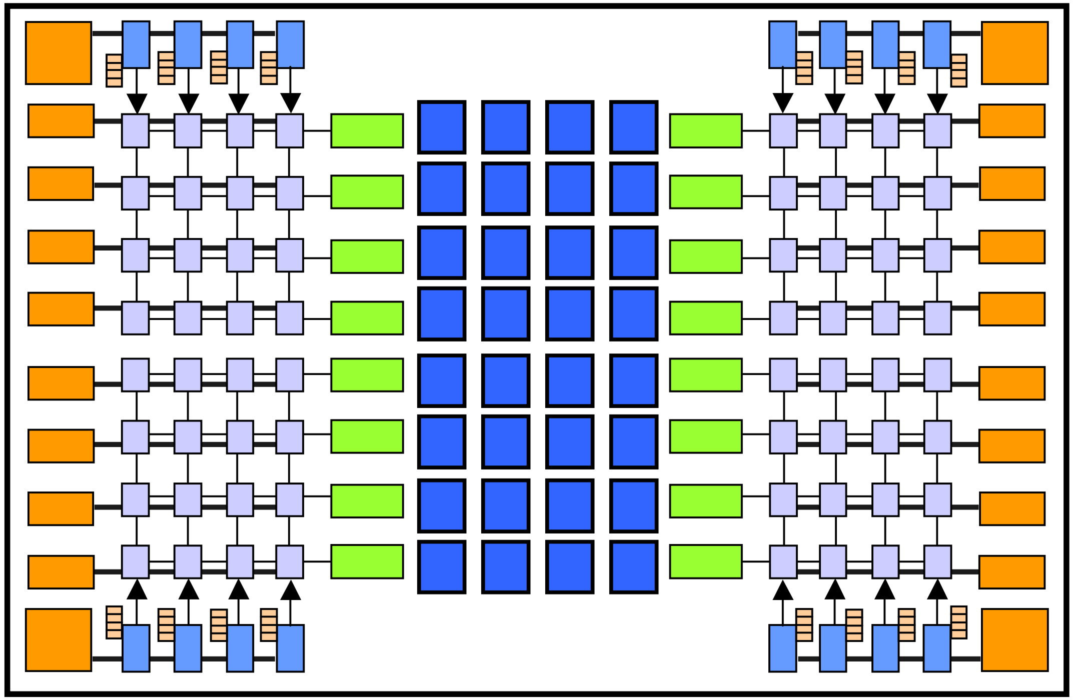
TRIPS: Desirable Attributes

- Performs well on DLP programs with different attributes
  - Synchronous core to minimize synchronization overheads on traditional vector/stream applications
  - MIMD-like capabilities for applications with irregular control
  - Support for different types of data structures
- Partitioned and scalable microarchitecture
  - Dataflow instruction execution
  - Limit/eliminate global broadcast of instructions/data
- Decoupled processor core
  - From memory system to enable memory fetch parallelism
  - From other processor cores to enable kernel pipelining
- Efficient instruction distribution and re-use
  - Exploit spatial/temporal locality

This slide compliments of Steve Keckler, UT Austin

TRIPS Chip

- 4 cores (with streaming support)
- L2 cache and SRF memory banks
- Pipelining across kernels mapped to different cores
  - Extend to system through off-chip channels



This slide compliments of Steve Keckler, UT Austin

Next-Generation Architectures

- MIT Raw processor
- Stanford Imagine processor
- UT Austin TRIPS processor
- Berkeley VIRAM Processor

Vector IRAM Approach (UC Berkeley)

<u>Vector processing</u>

- multimedia ready
- predictable, high performance
- simple
- energy savings
- high code density
- well understood programming model

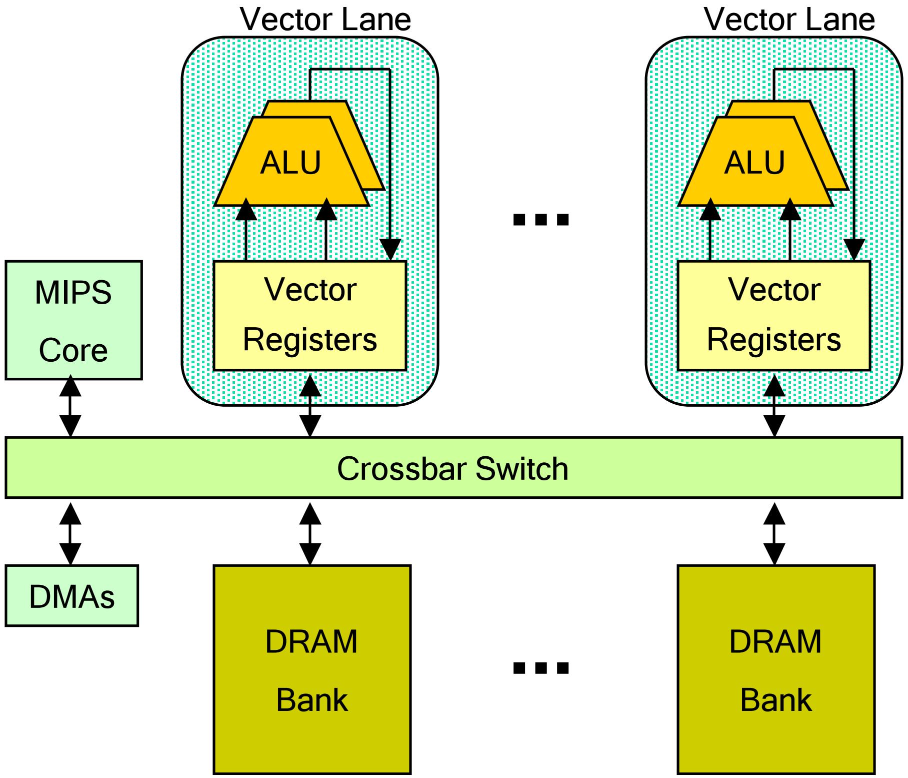
<u>Embedded DRAM</u>

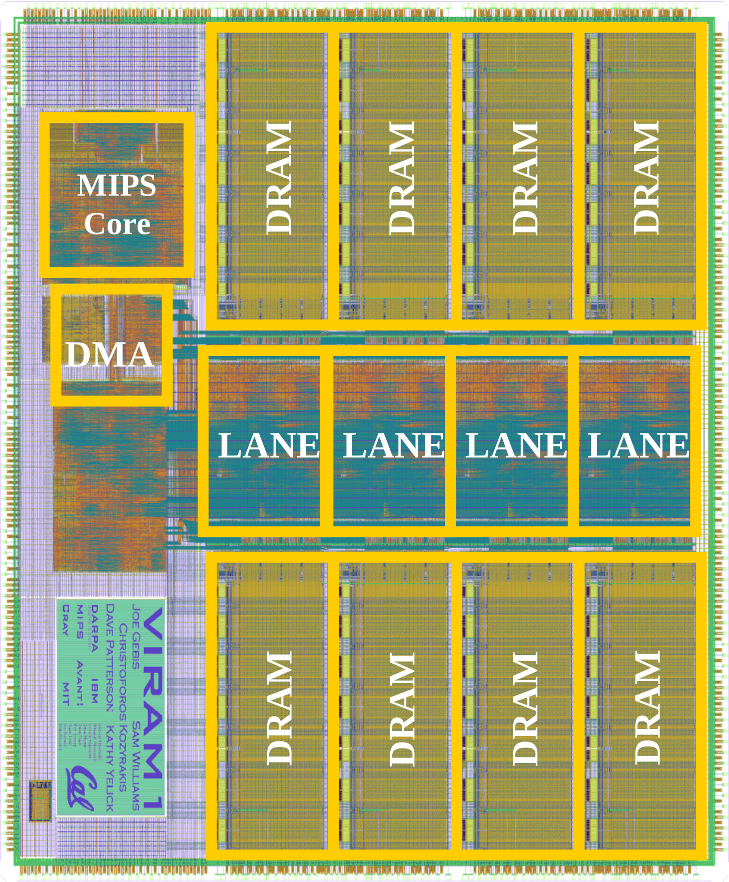
- high memory bandwidth
- low memory latency
- energy savings
- system size benefits

<u>Serial I/O</u>

- Gbit/sec I/O bandwidth
- low pin count
- low power

The VIRAM “Stream” Processor





Architectures, Languages, and Compilers for the Streaming Domain
This slide compliments of Christos Kozyrakis, Stanford University

The VIRAM “Stream” Processor

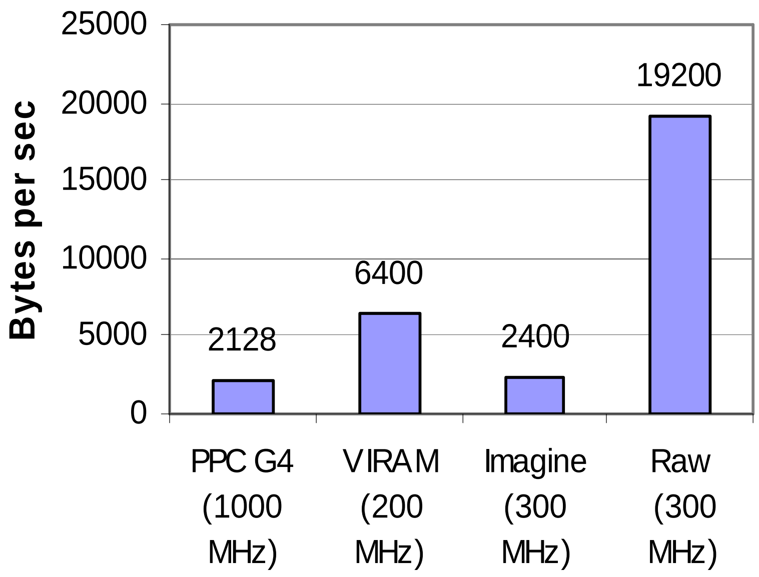
- Vector hardware
  - 125M transistors, 13MB DRAM, 0.18um CMOS
  - Single issue, in order, no hardware caches
  - 4.8 Gops (32b), 200MHz, 2W
- Performance
  - Evaluated for multimedia, telecom, and scientific apps
  - 10x over superscalar and VLIW
  - Even better with a clustered, decoupled vector processor
    - See MICRO’02, IPDPS’02, ISCA’03
- “Streaming” software
  - C with pragmas for data-level parallelism
    - Sufficient for many array-based and SDF computations
  - Vectorizing compiler (based on Cray compiler)

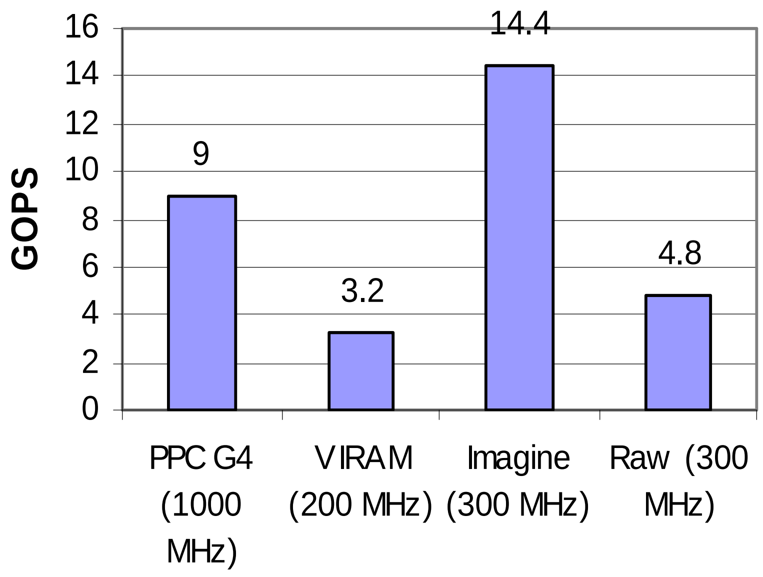
This slide compliments of Christos Kozyrakis, Stanford University

Initial Performance Study

- From an independent study done at ISI East
  - “A Performance Characterization of New Microprocessor Paradigms on Data-Intensive Kernels” - Jinwoo Suh, Eun-Gyu Kim, Stephen P. Crago, Lakshmi Srinivasan, and Matthew C. French, ISCA 2003.
- Many caveats
  - Hard to do apples-to-apples comparison
    - Different process generations
    - Different clock speeds
    - Different tool chains
    - Different languages
  - Only small kernels

Prototype Peak Memory Bandwidth and GOPS





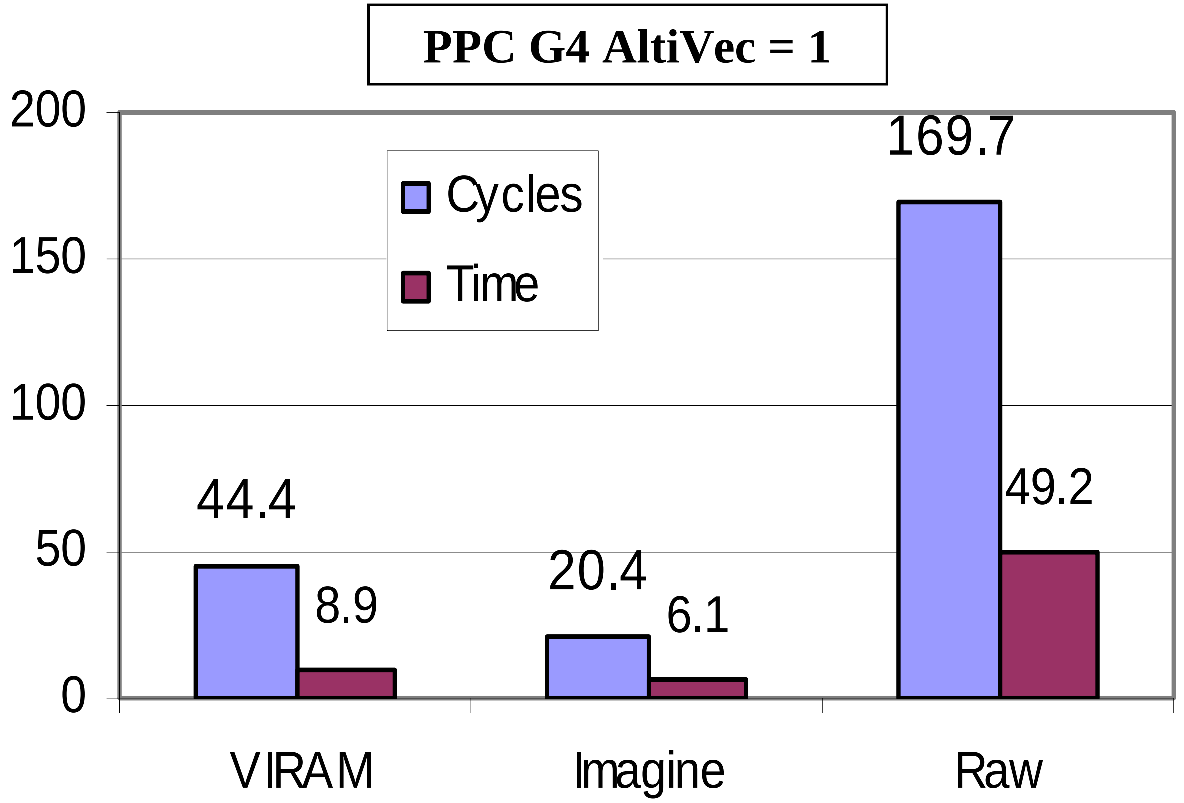
- Memory speed
  - VIRAM, Imagine, Raw: as fast as processors
  - PPC: 266 MHz DDR SDRAM

Overview of Kernels

- Corner turn
  - 1K by 1K matrix transpose
  - Source and destination in memory
  - Out of place
- Coherent sidelobe canceller (CSLC)
  - Radar signal processing
  - Basically convolution in frequency domain
  - FFT – Multiplication – IFFT
- Beam steering
  - Radar signal processing
  - Mostly load/store and add operations

This slide compliments of Jinwoo Suhm, USC/ISI

Speedup for CT

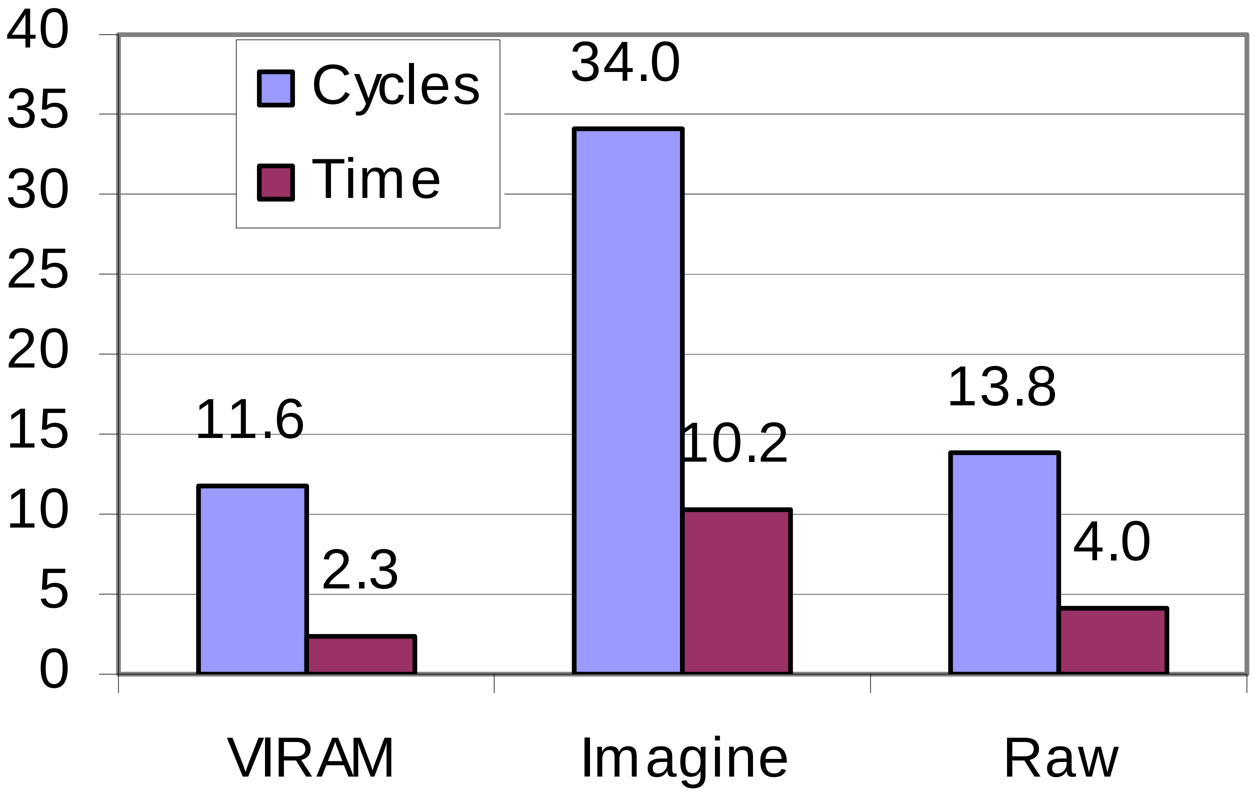


- All three architectures obtain almost peak memory bandwidth

PPC G4: 1000 MHz (Memory 266 MHz), Measured
Imagine: 300 MHz, Simulated

VIRAM: 200 MHz, Simulated
Raw: 300 MHz, Simulated

Speedup for CSLC



PPC G4 AltiVec = 1

Radix-4: PPC & VIRAM

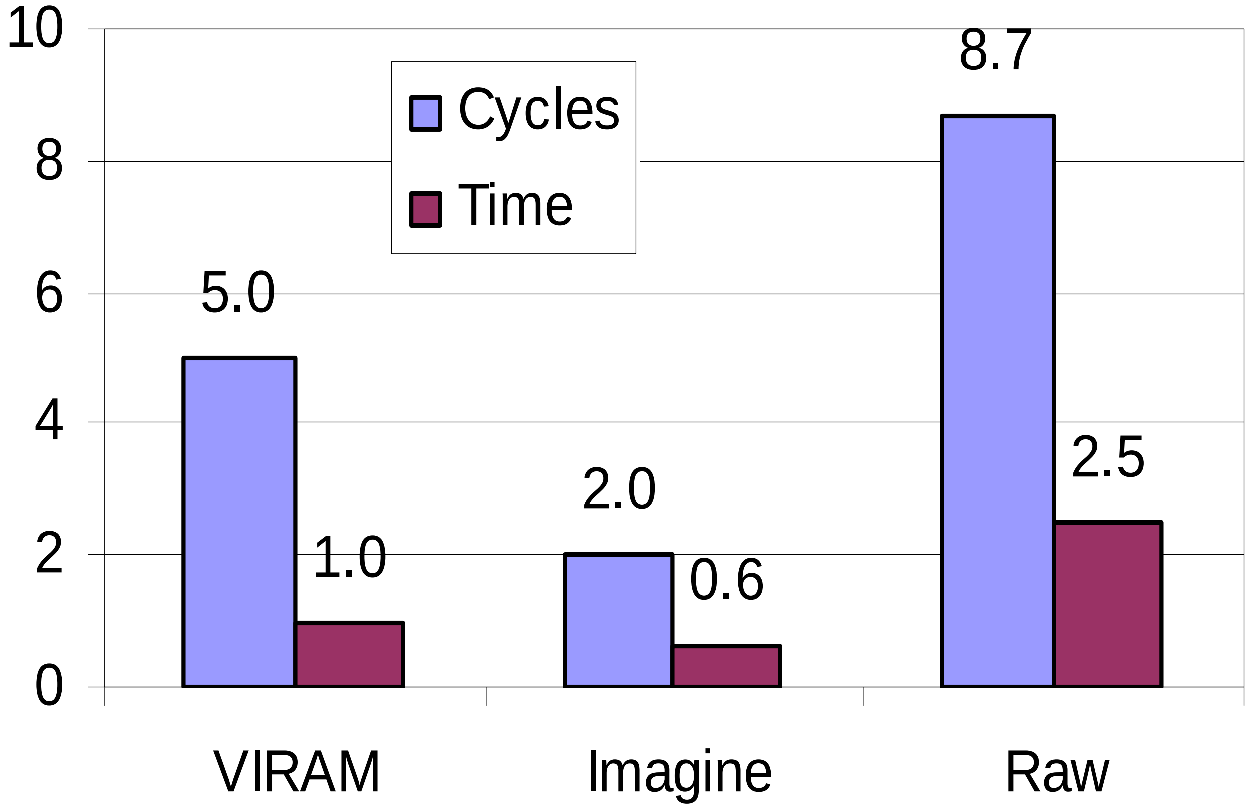
Radix-2: Imagine & Raw

PPC G4: 1000 MHz, Measured
Imagine: 300 MHz, Simulated

VIRAM: 200 MHz, Simulated
Raw: 300 MHz, Simulated

Speedup for Beam steering

PPC G4 AltiVec = 1



PPC G4: 1000 MHz, Measured
Imagine: 300 MHz, Simulated

VIRAM: 200 MHz, Simulated
Raw: 300 MHz, Simulated

Conclusion

- Uniprocessor scaling is near its end
  - Wire delay is a big issue
- Stream architectures have a lot of potential
  - Expand spatially
  - Balance the bandwidth hierarchy
  - Morph general purpose substrates
- Cannot keep supporting the same old programming model

Future Work for Architects: Programming Language Design

- Why C (FORTRAN, C++ etc.) became very successful?
  - Abstracted out the differences of von Neumann machines
    - Register set structure
    - Functional units and capabilities
    - Pipeline depth/width
    - Memory/cache organization
  - Directly expose the common properties
    - Single memory image
    - Single control-flow
    - A clear notion of time
  - Can have a very efficient mapping to a von Neumann machine
  - "C is the portable machine language for von Numann machines"
- Today von Neumann languages are a curse
  - We have squeezed out all the performance out of C
  - We can build more powerful machines
  - But, cannot map C into next generation machines
  - Switching to a data-flow programming paradigm will help