

# A Design Space Exploration of Grid Processor Architectures

Karu Sankaralingam

Ramadass Nagarajan, Doug Burger, and Stephen W. Keckler

Computer Architecture and Technology Laboratory

Department of Computer Sciences

The University of Texas at Austin

# Technology and Architecture Trends

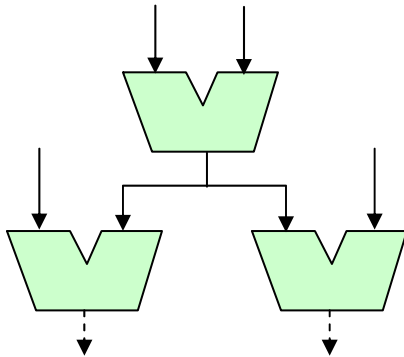
---

- Good news
  - Lots of transistors, faster transistors
- Bad news
  - Pipeline depth near optimal
    - Pipelining limits will slow clock rate improvements by half
  - Performance must come from more ILP
    - IPC has only doubled in one decade, despite considerable effort
  - Global wire delays are growing
    - At 35nm, less than 1% of die is 1-cycle-reachable
- Goals for future architectures
  - Scalability with process technology improvements
  - Fast clock *and* high ILP

# A “New” Approach

---

- ALU chaining



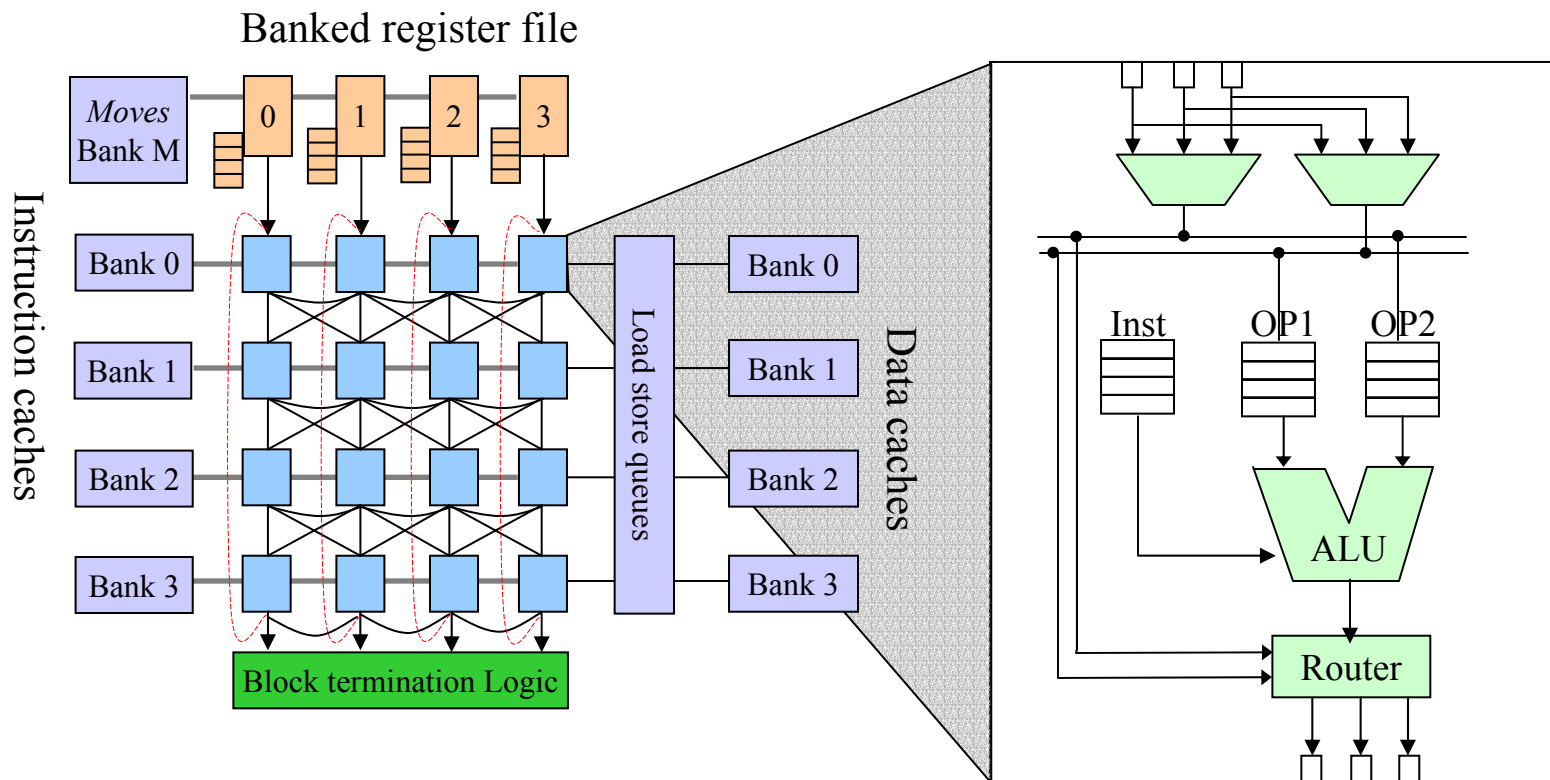
- Execution model eliminates
  - Majority of register reads
  - Associative issue windows
  - Rename tables
  - Global bypass
- Partitions I-Cache and register file banks around ALUs
- *Statically map and dynamically issue*

# Outline

---

- Grid Processor Architecture (GPA)
- Block Compilation
- Program Execution
- Evaluation
- Conclusions and Future Work

# Grid Processor

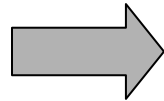


# Block Compilation (1 of 3)

## Intermediate Code

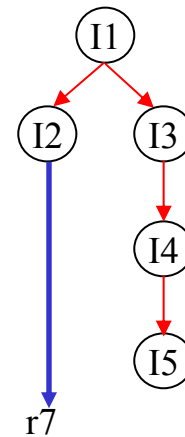
I1) add r1, r2, r3  
I2) sub r7, r2, r1  
I3) ld r4, (r1)  
I4) add r5, r4, r4  
I5) beqz r5, 0xdeac

- Inputs
- Temporaries
- Outputs



## Data flow graph

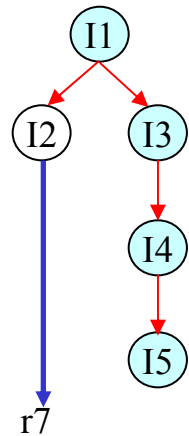
move r2, I1, I2  
move r3, I1



# Block Compilation (2 of 3)

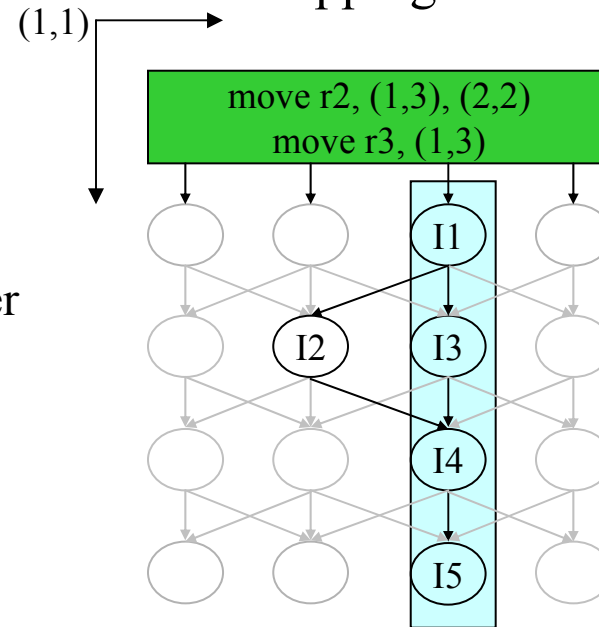
Data flow graph

move r2, I1,I2  
move r3, I1

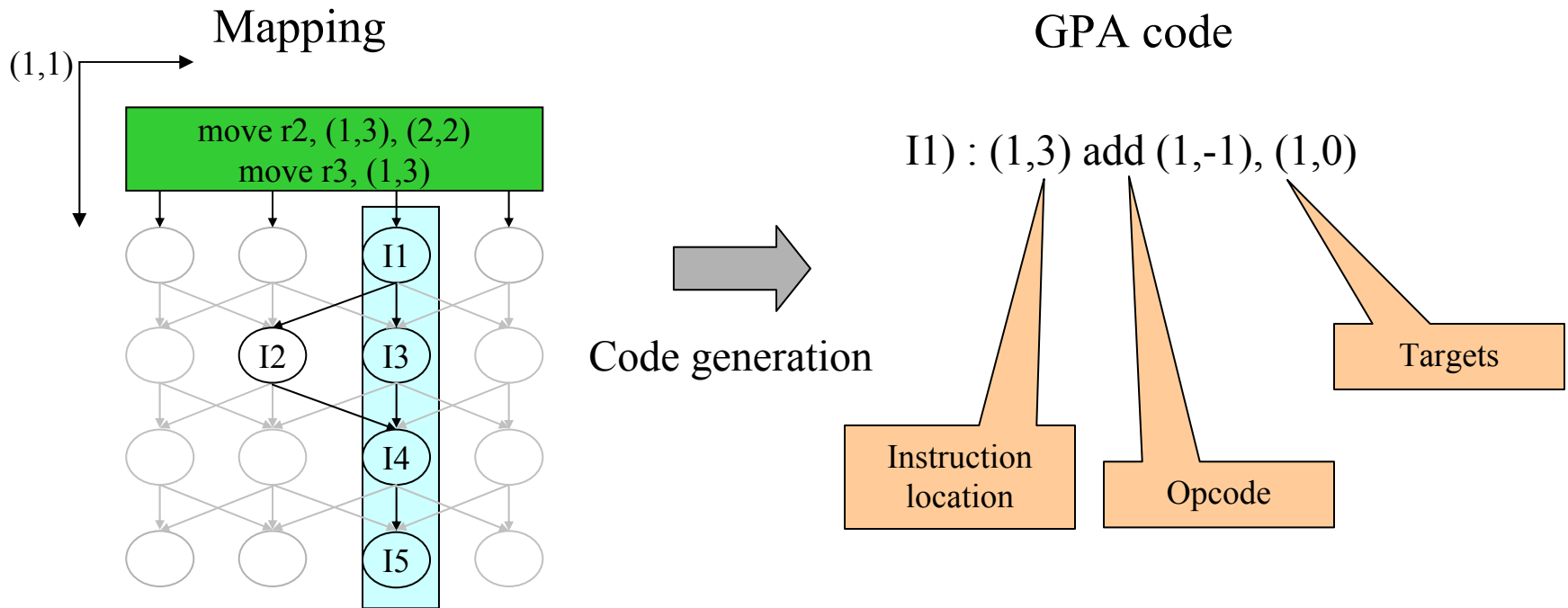


Scheduler

Mapping



# Block Compilation (3 of 3)

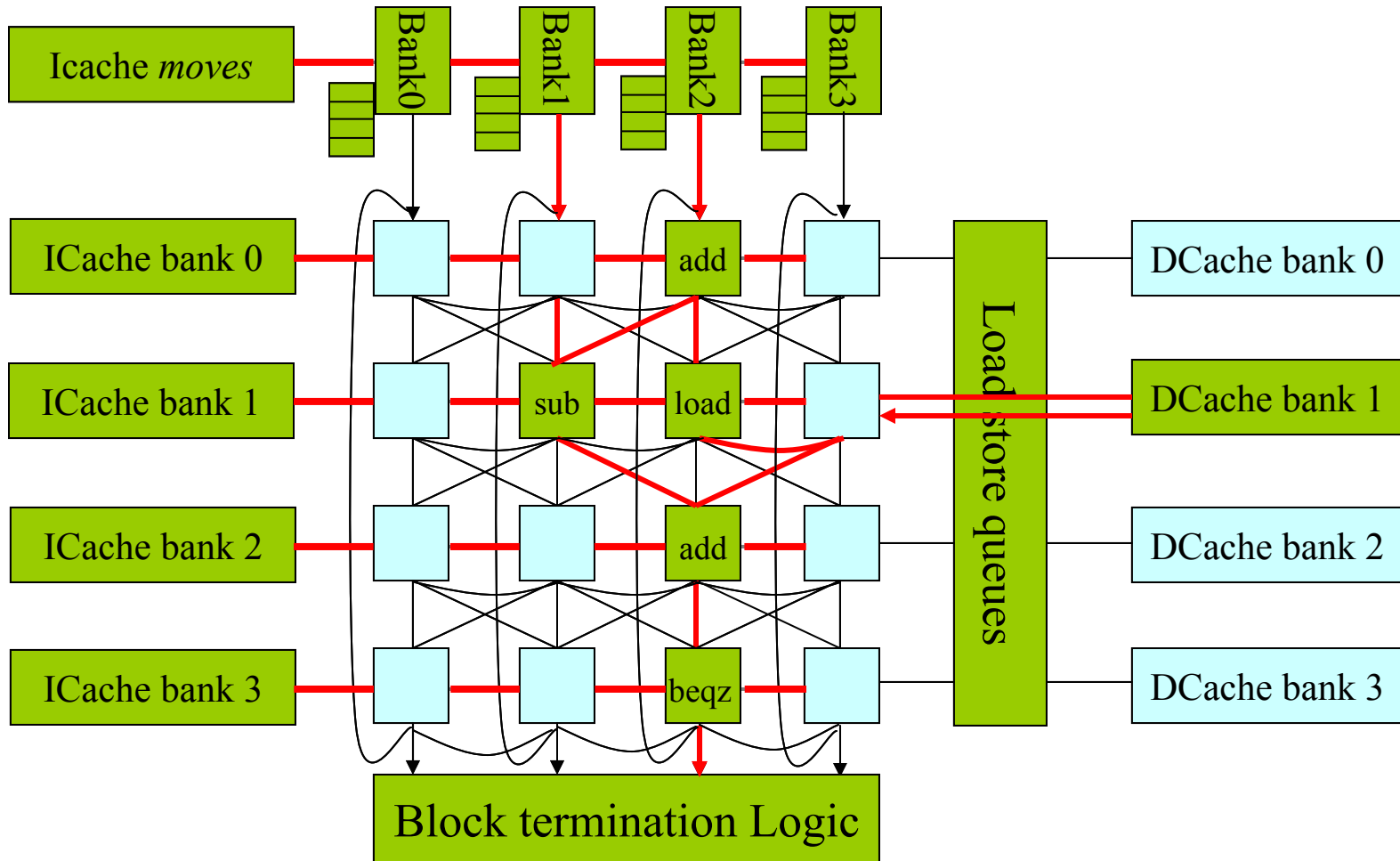


# Block Atomic Execution Model

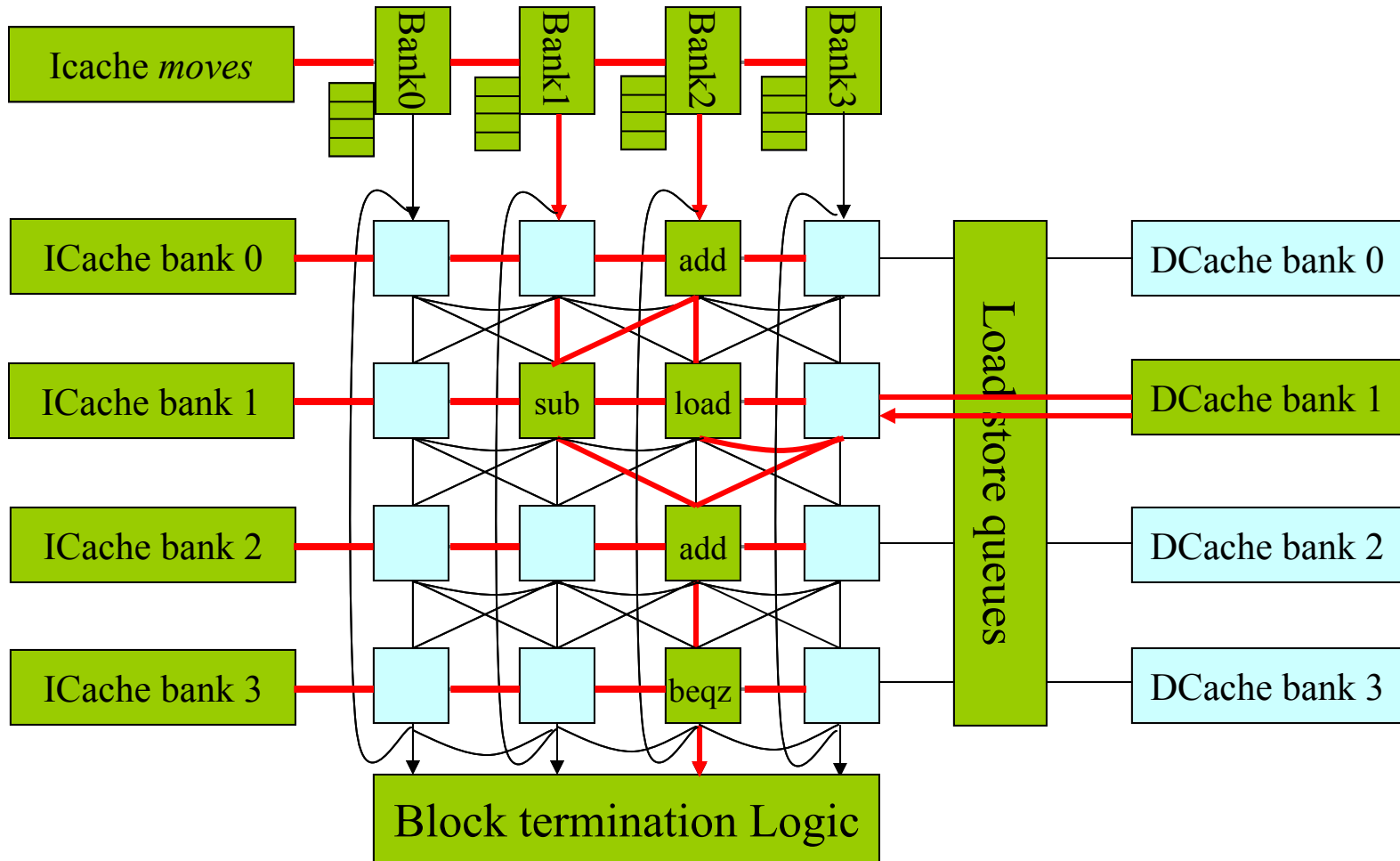
---

- A *block* of instructions is an atomic unit of fetch/schedule/execute/commit
- Blocks expose critical path
  - Operand chains hidden from large structures
  - Instructions specify consumers as explicit targets
- Blocks allow simple internal control flow
  - Single point of entry
  - If-conversion using predication
- Predicated hyperblocks

# Block Execution

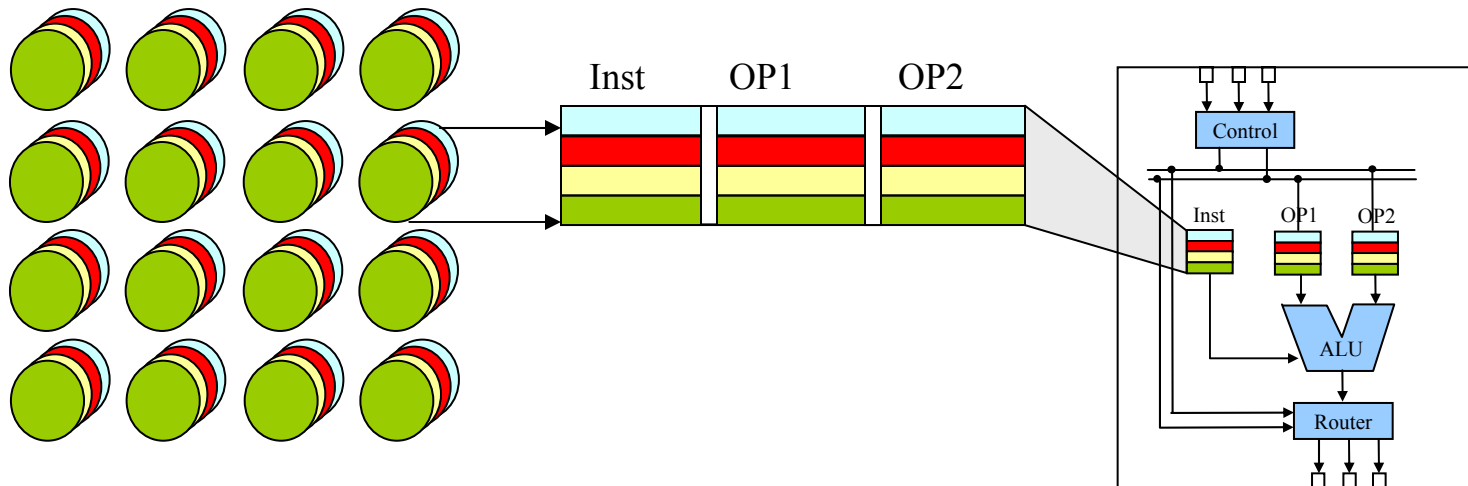


# Block Execution



# Instruction Buffers - *Frames*

- What if?
  - Blocks exceed grid size
  - Overlap fetch and map
- Use *frames*!
  - Virtualize grid
    - 4 slots == 4 frames
  - Logical partitioning of node storage space
  - Local OOO issue



# Execution Opportunities

---

- Serialized block fetch/map and execute
  - Overlapped instruction distribution and execution
- Overlapped fetch/map
  - Next-block predictor
  - Block level squash on mis-prediction
- Overlapped execution of blocks
  - Next-block predictor
  - Block level squash on mis-prediction
  - Block stitching using input/output register masks

# Evaluation

---

- 3 SPECInt2000, 3 SPECint2000, 3 Mediabench benchmarks
  - adpcm, dct, mpeg2encode
  - gcc, mcf, parser
  - ammp, art earthquake
- Compiled using the Trimaran toolset
- Hyperblocks parsed and scheduled using custom tools
- Event driven configurable timing simulator used for performance estimates

# GPA Evaluation Parameters

---

## Superscalar

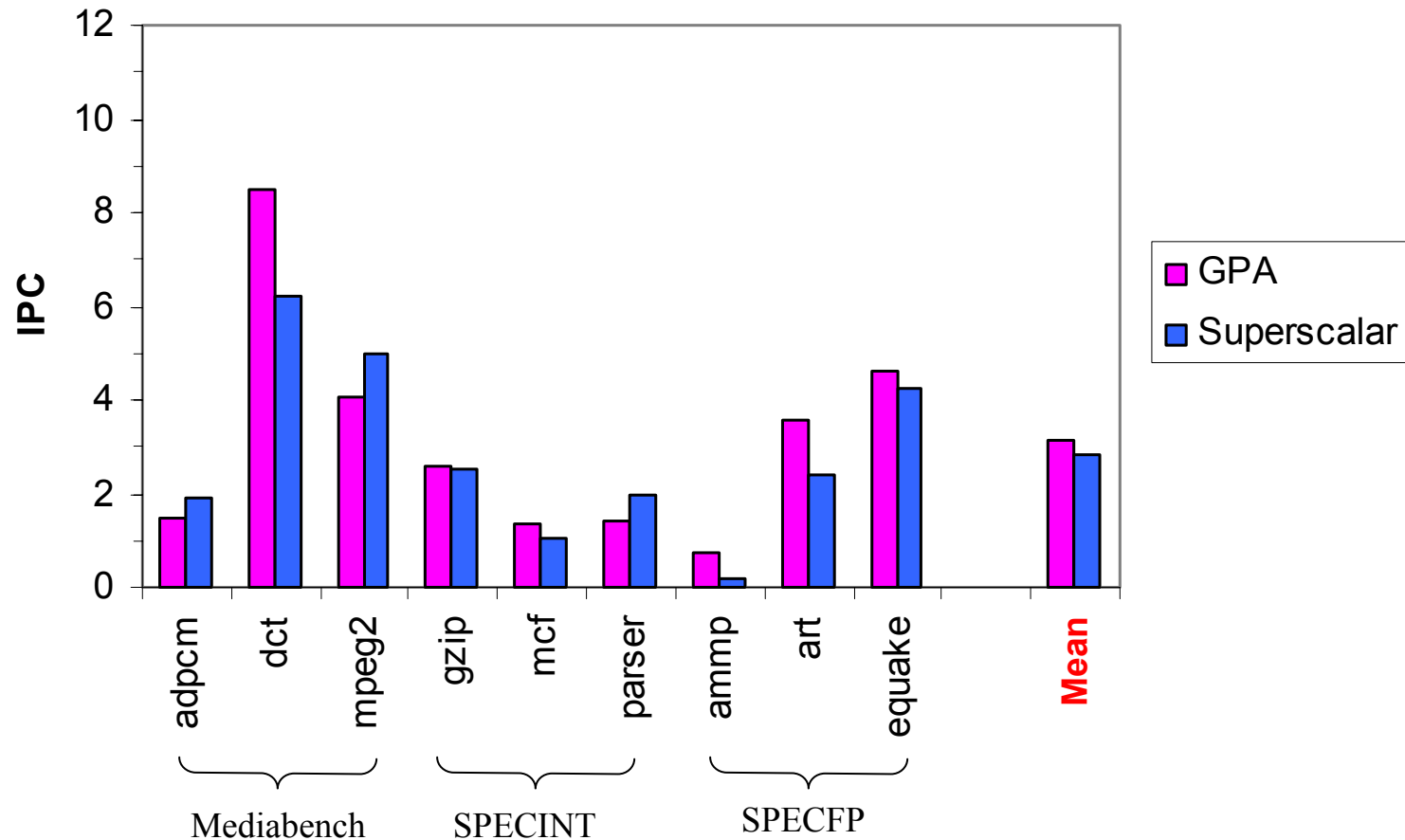
- 5 stage pipeline, 8-wide
- 0 cycle router and wire delay!
- 512 entry instruction window

- Alpha 21264 functional unit latencies
- L1: 3 cycles, L2: 13 cycles, Main memory: 62 cycles

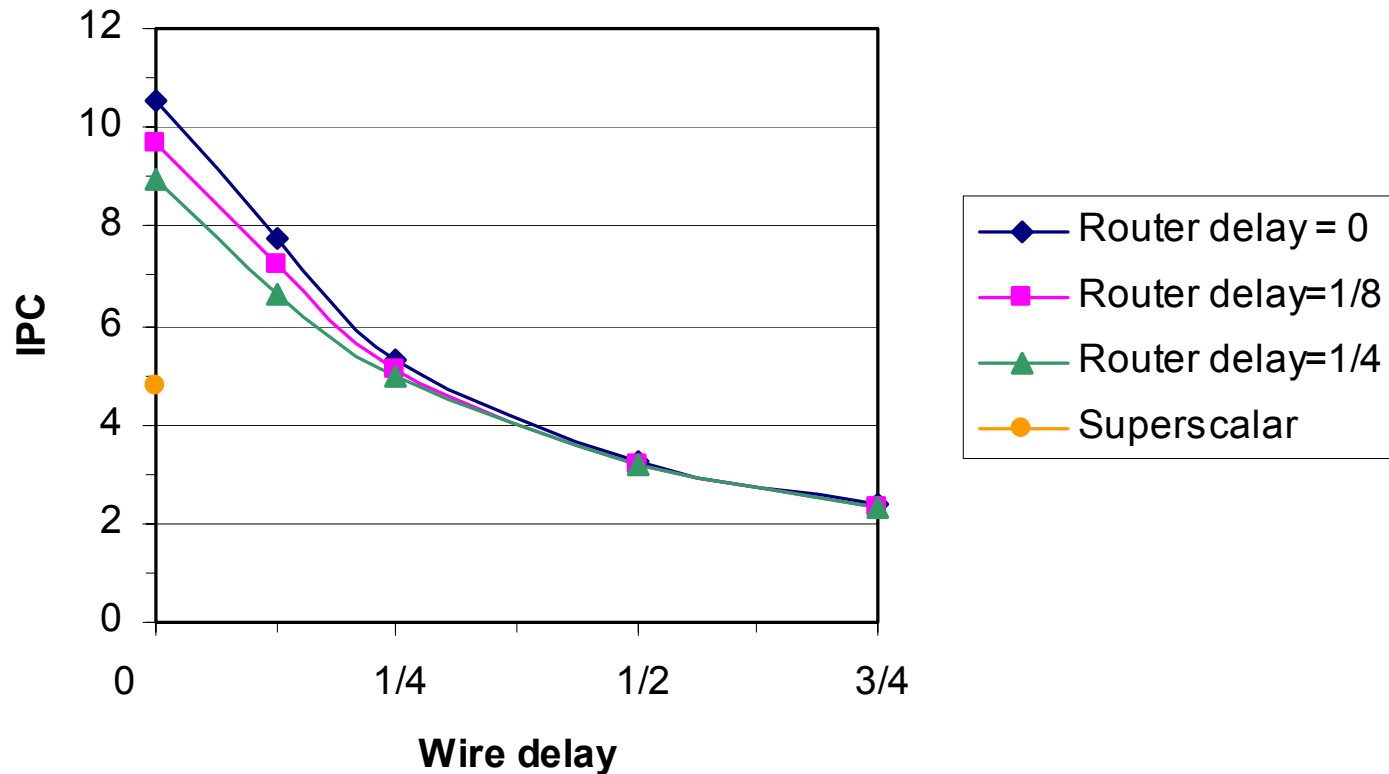
## GPA

- 8x8 grid
- $\frac{1}{4}$  cycle router +  $\frac{1}{4}$  cycle wire delay
- 32 slots at every node

# GPA Performance Comparison



# Sensitivity to Communication Delay



# Conclusions

---

- Technology trends
  - Enforce partitioning
  - Wire delays become first order constraint
- GPA
  - Distributed execution engine with few central structures
  - **Technology scalable, fast clock rate *and* high ILP**
- Challenges
  - Block control mechanisms
  - Distributed memory interface design
  - Optimizing predication mechanisms

# Future Work

---

- Alternate execution models
  - SMT support
    - Use frames to run different threads
  - Stream based execution
    - Loop re-use and data partitioning in caches
  - Scientific vector-based execution
    - Use rows as vector execution units
    - Vector loads read from caches
- Hardware prototype

# Related Work

---

- Dataflow
  - Static dataflow architecture – Dennis and Misunas [1975]
  - Tagged-Token Dataflow – Arvind [1990]
  - Hybrid dataflow execution – Culler et. al [1991]
- RAW architecture – Waingold et. al [1997]
- Multiscalar Processors – Sohi et. al [1995]
- Trace Processors – Vajapeyam [1997]
- Clustered Speculative Multithreaded Processors – Marcuello and González [1999]
- Levo – Uht et. al [2001]

---

# Questions