A Design Space Evaluation of Grid Processor Architecture

Jiening Jiang
May, 2005

The presentation based on the paper written by Ramadass Nagarajan, Karthikeyan Sankaralingam, Doug Burger, Stephen W. Keckler

Outline

- Introduction
- The Block-Atomic Execution Model
- Implementation
- Evaluation
- Design Alternatives
- Conclusion

Introduction

- Microprocessor performance has improved at a rate of 50-60% per year over the past decades
  - In 70's, wider datapath and hardware support for memory management are main contributors
  - In 80's, memory hierarchies, speculation and superscalar execution are main contributors
  - Since then, performance growth mainly from fast clock rates. (in 90's, 4/5 growth from CR)

Introduction - Problems Facing

- Clock rates growth slow down soon
  - Clock rate comes from technical scaling and deeper pipelines, more from the latter, however the deeper pipelines reach limits on the number of gates per stage.
  - Gates rate estimated to improved by 12-19%
  - Further performance improvements from ILP, TLP

Introduction - Problems Facing

- Increasing wire resistance will make achieving high ILP in conventional architecture more difficult
  - Signal transmission need more CCs
  - Limiting number of devices useful
  - Wire delays make memory-oriented architecture slow.

Introduction - GPA and Main Features

- GPA will achieve faster clock rates and higher ILP
- No central instruction issue window
- A routed P2P network other than broadcast bypass network
- Like VLIW, compiler detects the parallelism and statically schedules instructions

Introduction - GPA and Main Features

- Few large structures reside on the critical execution path
- Large instruction blocks are mapped onto nodes as single units of computation, amortizing overheads over a large number of instructions

The Block-Atomic Execution Model

- Instructions are placed into groups by the compiler
- A group has no internal control transfer
- Three types of data: group inputs; group temporaries; group outputs
  - Inputs must read when the group execute
  - Temporaries forward from producers to consumers; no written back to central storages
  - Outputs written back central storages when the group commit

The Block-Atomic Execution Model

- Each instruction in a group assigned to one of the name ALU, no ALU has more than one instruction.
- *Move* instruction read the group inputs and forward to appropriate ALUs
- A group instructions fetched and mapped to substrate once

Simple Example of Block-Atomic Mapping

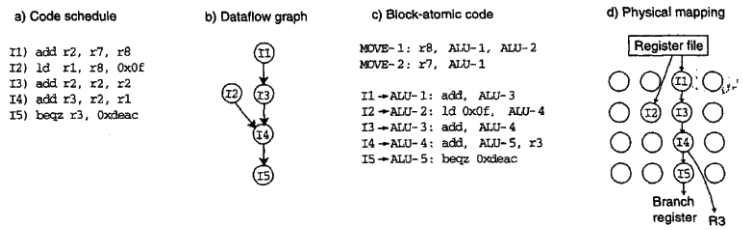


Figure 1: Simple Example of Block-Atomic Mapping

Key Advantages of this Model

- No centralized associative issue window
- No register renaming table
- Fewer register read and write
- Can execute in dynamic order without hazards checking or a broadcasting bypassing and forwarding network
- Producer to consumer can take place along P2P
- Instructions off critical path can afford longer communication delay
- The scheduler can minimize the critical path

Implementation

- Terminology
  - Node: function unit
  - Frame: A frame consists of a single instruction slot in all of the grid nodes. virtual grid
  - Hyperblock: A set of predicated basic blocks in which control may enter from the top, but may exit from one or more location

Implementation - High-level Grid Processor Organization

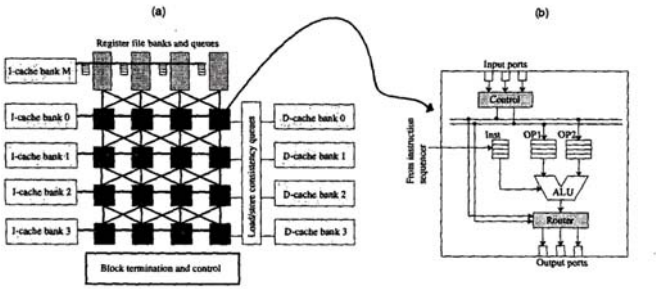


Figure 2: High-level Grid Processor organization

Implementation

- Instruction fetch and map
  - I-cache has multiple rows
  - A row's worth of instruction indicate the row position of inst in the grid
  - After a hyperblock mapped, branch and target predictors in the block sequencer predict the succeeding hyperblock, and begin fetching and mapping it onto the grid prior to the completion of the previous hyperblock

Implementation

- Instruction execution
  - The *move* Instructions mapped to register file banks
  - When a operand arrives the node, control logic wakeup, select and issue the correspond instruction
  - If all operands ready, the inst issued to the ALU
  - If no new operands arrives at a node for a given circle or must wait more operands, any other ready instruction is selected and issued

Implementation - Operand routing

- In GPA-1, every node has 3 inputs and 3 outputs ports
- If more than 3 consumers, *split* Instruction insert
- Design trade-off, instruction size, routing delay, complexity
- Statically showed, 70% producers have 3 or less consumers

Implementation - Inter-node Network

- Four kinds of delay
  - Routing delay, transmission/wire delay, instruction wakeup delay, and delay induced by contention for the wires/ports at the node
  - Routing delay and wire delay are most important factors in overall performance of GPA-1

Implementation - Hyperblock Control

- Predication
  - GPA-1 uses an execute-all approach, but only one path delivers a result to the common instructions
  - Special instruction set *cmove*
  - See code example

Implementation - Predication Code Example

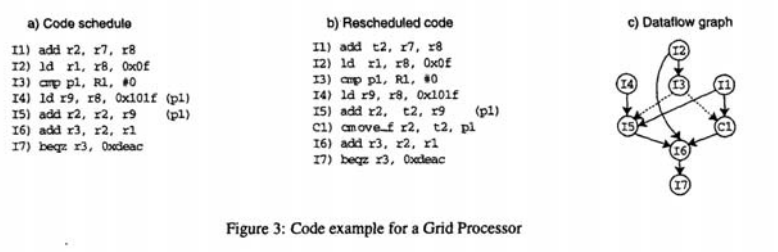


Figure 3: Code example for a Grid Processor

Implementation - Hyperblock Control

- Early exits
  - GPA-1 uses predication to enforce the sequentiality
  - Extra-predication is necessary when the same register name is to be produced by multiple instructions in the block and not for every output instruction
  - Those results executed before a prior branch should filter out by block commit logic using index (position of static program order)

Implementation - Hyperblock Control

- Block commit
  - Distributed execution make global control complicated
  - Additional logic is needed in block commit control
  - GPA-1 employs a count of output values associated with each hyperblock

Implementation - Hyperblock Control

- Block stitching
  - Concurrent execution of multiple hyperblocks
- Memory access
  - The primary data cache resides on the right hand side of array
  - To maintain the load-store order, use traditional load-store queues

Evaluation

- SPEC CPU2000 floating-point benchmarks
  - equake, ammp, and art
- SPEC CPU2000 integer benchmarks
  - parser, gzip, and mcf
- Three Mediabench benchmarks
  - adpcm, dct, and mpeg2enc
- Compiled by Trimaran tool set
- Custom instruction scheduler and custom event-driven timing simulator

Evaluation - Application Characteristics

- The characteristics of benchmark compiled by trimaran compiler

	Block size		Register usage					
Name	Static	Dynamic	Inputs	Temp reads	Temp writes	Outputs	Branch exits	Memory conflicts(%)
adpcm	51.1	30.7	6.5	26.7	22.4	5.7	5.7	0
dct	187.9	172.1	14.7	199.8	163.6	7.3	1.7	0
mpeg2	109.7	94.1	11.9	98.5	84.4	7.4	3.4	1.6
gzip	66.2	37.1	11.6	29.8	28.4	7.1	4.4	9.3
mcf	40.1	28.8	4.7	33.6	25.7	2.3	1.7	0.9
parser	20.3	16.2	3.4	13.7	12.7	2.8	1.8	4.8
ammp	82.5	72.3	6.6	67.8	58.1	3.8	8.7	0.6
art	82.7	79.6	25.6	54.2	54.0	20.0	5.3	54.9
equake	50.2	44.1	10.0	34.4	32.7	9.5	2.3	11.9

Table 1: Program characteristics

Register bandwidth reduced by 30-90%

Evaluation - Application Characteristics

- Overhead instructions, only *cmove* and *split* consume the instruction slot

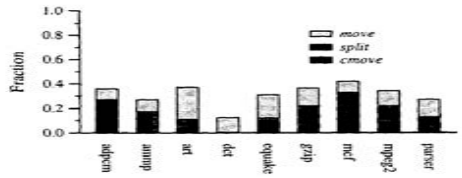


Figure 4: Overhead of Block-Atomic Execution

Overall 35% of all instructions, 20% instructions scheduled on the grid

Evaluation - Performance Evolution

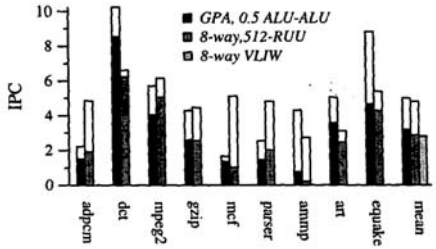


Figure 5: Performance Comparison of GPA with superscalar

Left bar: GPA-1; right bar: SS; white portion: perfect memory and branch

Evaluation - Block Stitching

Name	Perfect Mem + BP IPC No stitch	Perfect Mem + BP Stitching Speedup	Realistic Mem + BP IPC No stitch	Realistic Mem + BP Stitching Speedup
adpcm	1.4	1.4	1.1	1.2
dct	4.0	2.5	3.7	2.2
mpeg2	3.0	1.8	2.7	1.4
gzip	1.9	2.2	1.4	1.7
mcf	1.0	1.5	0.6	1.9
parser	1.0	2.4	0.8	1.7
ammp	2.3	1.8	0.2	3.1
art	2.5	1.9	1.5	2.2
equake	2.5	3.4	2.0	2.2

Table 2: Speedup achieved by stitching

Block stitching provided about a factor of 2 speedup

Evaluation - Routing Delay

Name	Average hops per data value Input	Temporary	Memory
adpcm	2.8	1.8	1.9
dct	2.9	2.4	3.7
mpeg2	3.3	1.5	3.8
gzip	3.0	2.1	2.7
mcf	2.1	2.8	1.9
parser	2.6	1.8	1.8
ammp	2.4	1.7	4.9
art	3.8	1.7	3.2
equake	3.8	2.0	2.8

Table 3: Average hops for different types of data operands

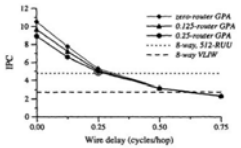
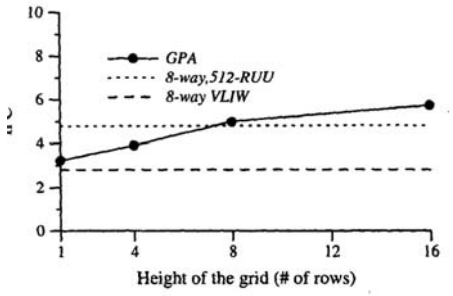


Figure 6: Sensitivity to wire delays

•3 most significant component: number of hops; inter-node wire delay and router delay at each hop

•Wire delay affects performance more than the router delay

Evaluation - Grid Dimension



•Some benchmarks performs best with 8 rows

•Programs with high available ILP and large block size benefits from the increase in the number of rows

Evaluation - GPA Effectiveness

Figure 7: Sensitivity to grid height

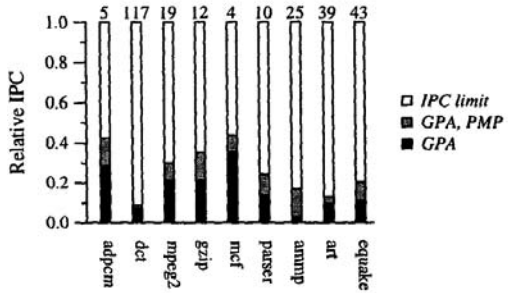


Figure 8: GPA effectiveness

Design Alternatives

- Grid network design
  - To reduce the logic and wire delay
    - Larger degree router decreases the number of hops but increases the delay per hop
    - Reduce handshaking overhead
    - Express channel
- Predication strategies
  - GPA-1: less efficient use of power
  - Or: send predicate bits to all instructions in PR
  - Or: send to the root of sub-graph.
  - Both alternatives limit performance

Design Alternatives

- Memory system
  - Compressed format of program codes below L1
  - Date memory, speculative and conservative strategies
  - The store-load pairs communicate via point-to-point, bypassing the memory system
- Grid speculation
  - Load speculatively, misprediction only trigger the dependence from the load to the end of the block

Design Alternatives

- Frame management
  - The frames speculatively mapped and executed the hyperblocks in a sequential program
  - The frames can support a multithreaded execution
- ALU control
  - Add more logic control to each ALU, each ALU as a simple microprocessor

Conclusion

- GPA intent to continue scaling both clock rate and instruction throughput.
  - Mapping dependence chains onto an array of ALUs
  - Conventional large structures can be distributed throughout the ALU array, permit better scalability of the processing core
  - Mitigate the growing global wire and delay overhead by P2P communication
  - Competitive with idealized superscalar, exceeding VLIW

Conclusion

- Drawbacks
  - Data cache far away from many of ALUs. Thus the delay between dependent operations can be significant
  - The complexity of frame management and block stitching is significant and may interfere with the goal of fast clock rate