

# *Architecture and Compiler Design Issues in Programmable Media Processors*

---



**Jason Fritts**

*Princeton University*

*Department of Electrical Engineering*

*Advisor: Prof. Wayne Wolf*

## *Overview*

---

- **Why Programmable Media Processors (PMPs)?**
  - Present and future of multimedia and media processing
- **Design Methodology**
- **Multimedia Workload Evaluation**
- **Architecture Evaluation**
  - Datapath architecture
  - Memory architecture
- **Parallelism Extraction by Compiler**
  - Types and granularity of parallelism
  - Supporting data parallelism in multimedia
  - Speculative Broadcast Loop (SBL) run-time method
- **Conclusions and Contributions**
- **Future Research**

2

## Multimedia Applications

- **Wide range of applications**

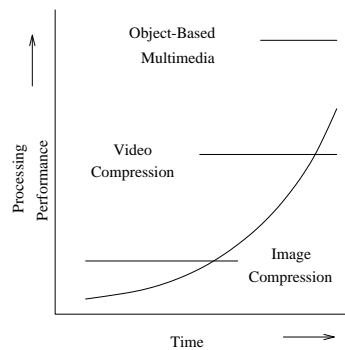
- *Entertainment*
  - games
  - chat rooms
  - home movies
- *Communication*
  - video conferencing
  - World Wide Web
  - digital libraries
  - videophones
- *Education*
  - interactive learning
  - virtual classrooms
- *Computer Vision*
  - image understanding
  - surveillance
  - tracking
- *Art and Architecture*

**Multimedia is primarily a communication media**

3

## Future of Multimedia

**Multimedia industry evolves with processor performance.**



**Multimedia is moving towards an object-oriented representation**

4

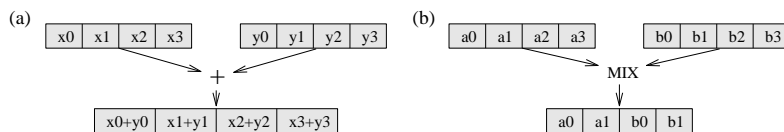
## Multimedia Processing Solutions

- **Application-specific processors**
  - good performance
  - low cost
  - limited flexibility
- **Multimedia extensions to general-purpose processors**
  - some speedup for SIMD parallel operations
  - low cost
  - some flexibility
- **Current “programmable” media processors (PMPs)**
  - good performance
    - special functional units: DCT, VBR coder, motion estimation
  - moderate frequency
  - flexibility
    - special programming libraries or paradigms

5

## Multimedia Extensions to General-Purpose Processors

- **Subword Parallelism:**
    - pack narrower data into larger words
    - special parallel operations process packed words
    - Introduced by Ruby Lee:
- [RLee95] “Accelerating Multimedia with Enhanced Microprocessors,” IEEE Micro, vol. 15, no. 2, April 1995.
- **Advantages**
    - little additional on-chip hardware => minimal cost
  - **Disadvantages**
    - only provides SIMD parallelism
    - packing words generates overhead
    - chip area optimized for general code



6

## *Existing Programmable Media Processors*

---

<i>Media Processor</i>	<i>Subword Parallelism</i>	<i>Specialized Hardware</i>	<i>ILP</i>	<i>Parallel Processing</i>	<i>Clock Frequency</i>
Texas Instruments MVP (C8x)	✓	-	-	✓	50 MHz
Texas Instruments VelocITI (C6x)		-	8-wide	-	300 MHz
Philips TriMedia TM-1000	✓	✓	5-wide	-	166 MHz
Philips TriMedia TM-2000	✓	✓	5-wide	-	~300 MHz
Equator/Hitachi MAP1000A	✓	✓	4-wide	-	220 MHz
NEC V830R/AV	-	-	2-wide	-	200 MHz
Chromatic Research Mpact-1	✓	✓	5-wide	-	?
Chromatic Research Mpact-2	✓	✓	6-wide	-	125 MHz
MicroUnity MediaProcessor	✓	✓	-	-	-
Samsung MSP-1	✓	✓	-	✓	-

7

## *Expectations for Future Media Processors*

---

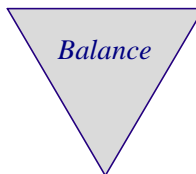
- **Greater Throughput**
- **Larger On-Chip Memory Hierarchies**
- **Increased Architecture Regularity**

**Throughput**

- fast clock speed
- high parallelism
- high utilization

**Storage**

- large on-chip memory
- large register file
- efficient memory I/O

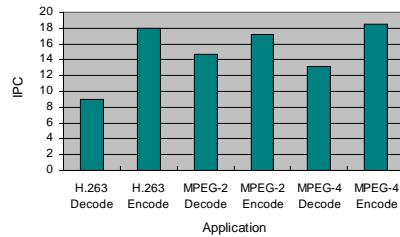


**Programmability**

- high connectivity
- regular arrangement
- optimizing compiler

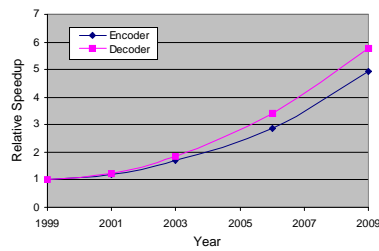
8

## Potential of Future Media Processors



- parallelism within video applications evaluated using trace-driven simulation

- relative speedup for MPEG-2 coder based on increasing VLSI technology over next decade



9

## Design Methodology

10

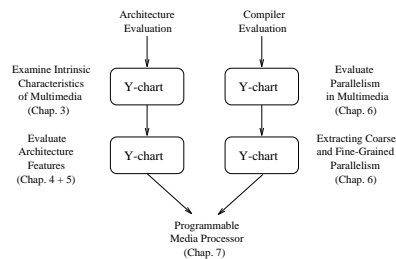
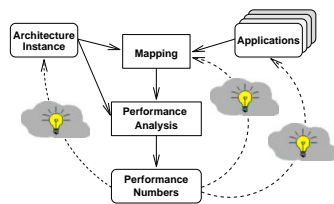
## *Design Methodology for Programmable Media Processors*

- **'Y-Chart' defines architecture design space exploration**

[Kienhuis99] "Design Space Exploration of Stream-based Dataflow Architectures," Ph.D. Thesis, Delft University of Technology, 1999.

- **Methodology for media processors focused on:**

- Behavioral and cycle-accurate simulation levels
- Separate architecture and compiler evaluation paths



11

## *MediaBench Benchmark Suite*

- **Developed at UCLA**

[Clee97] "MediaBench: A Tool for Evaluating and Synthesizing Multimedia Communication Systems," MICRO-30, 1997.

- **Excellent combination of applications**

- video: MPEG-2
- audio: ADPCM coder
- graphics: Mesa
- image: JPEG, EPIC, Ghostscript
- security: PGP, Pegwit
- speech: GSM, G.721, Rasta

- **Still in development stages**

- Only two input data sets
- Some small data sets and traces

- **Augmented for greater representation of future multimedia**

- MPEG-4 object-oriented video
- H.263 very-low bitrate video

12

## *IMPACT Compiler*

---

- **Aggressive ILP research compiler**
  - superblock (speculation)
  - hyperblock (predication)
  - loop unrolling
- **Three levels of optimizations**
  - Classical - classical optimizations only
  - Superscalar - adds loop unrolling and superblock formation
  - Hyperblock - adds hyperblock optimization
- **Architecture-independent evaluation**
  - large, generic instruction set
  - retargetable back-end
- **Performance analysis tools**
  - profiling
  - simulation for superscalar, VLIW architectures

13

---

## *Workload Evaluation*

14

## *Characteristics of Multimedia*

---

- **Compile with classical optimizations only**

- **Related Work**

- [Lee97] "MediaBench: A Tool for Evaluating and Synthesizing Multimedia Communication Systems," MICRO-30, 1997.
- [Lee98] "Media Architecture: General Purpose vs. Multiple Application-Specific Programmable Processors," DAC-35, 1998.
- [ZWu99] "Architecture Evaluation of Multi-Cluster Wide-Issue Video Signal Processors," Ph.D. Thesis, Princeton University, 1999.

### Characteristics

### Design Areas

Operation Frequencies	Functional Units
Integer Data Sizes	Datapath Width, Subword Parallelism
Branch Statistics	Branch Prediction
Working Set Sizes	Memory Hierarchy
Spatial Locality	Memory Hierarchy
Loop Statistics	Datapath, Compiler Design
Path Ratio	Datapath, Compiler Design
Parallelism	Datapath, Compiler Design

15

## *Basic Characteristics*

---

- **Comparison of operation frequencies with SPEC**

- (ALU, mem, branch, shift, FP, mult) => (4, 2, 1, 1, 1, 1)
- Lower frequency of memory and floating-point operations
- More arithmetic operations
- Larger variation in memory usage

- **Basic block statistics**

- Average of 5.5 operations per basic block
- Need global scheduling techniques to extract ILP

- **Static branch prediction**

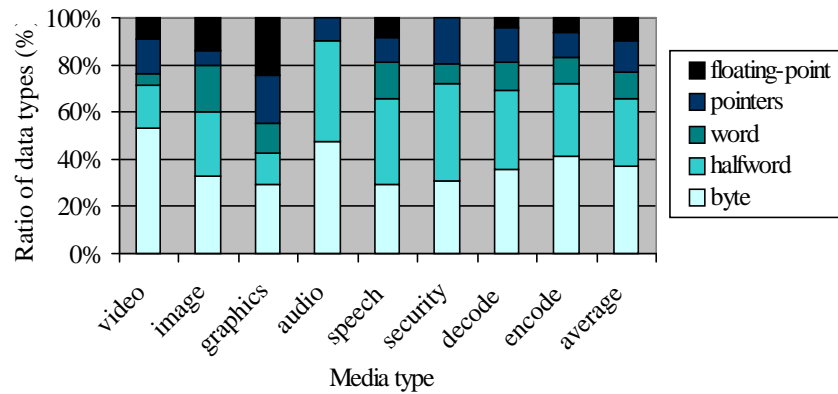
- Average of 89.5% static branch prediction on training input
- Average of 85.9% static branch prediction on evaluation input

- **Data types and sizes**

- Nearly 70% of all instructions require only 8 or 16 bit data types

16

## Breakdown of Data Types by Media Type



17

## Memory Statistics

- **Working set size**

- cache regression: cache sizes 1K to 4MB
- assumed line size of 64 bytes
- measured read and write miss ratios

$$\text{spatial locality} = \frac{(A - B)}{\left(\frac{A}{l_a / l_b}\right)}$$

- **Spatial locality**

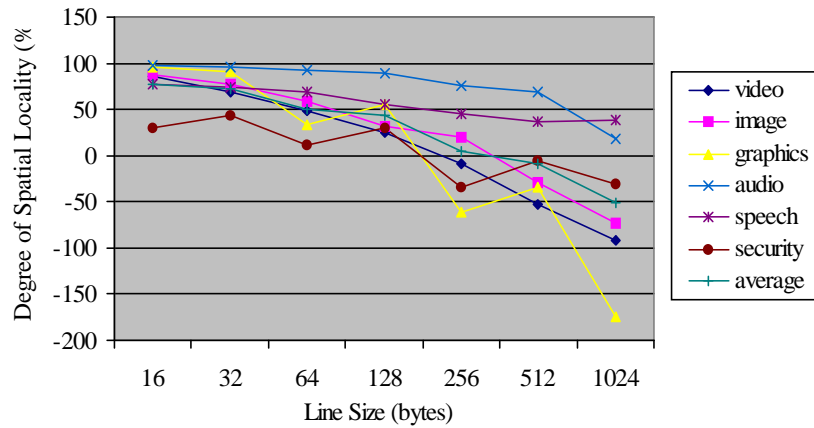
- cache regression: line sizes 8 to 1024 bytes
- assumed cache size of 64 KB
- measure read and write miss ratios

- **Memory Results**

- data memory: 32 KB and 60.8% spatial locality (up to 128 bytes)
- instruction memory: 8 KB and 84.8% spatial locality (up to 256 bytes)

18

## Data Spatial Locality



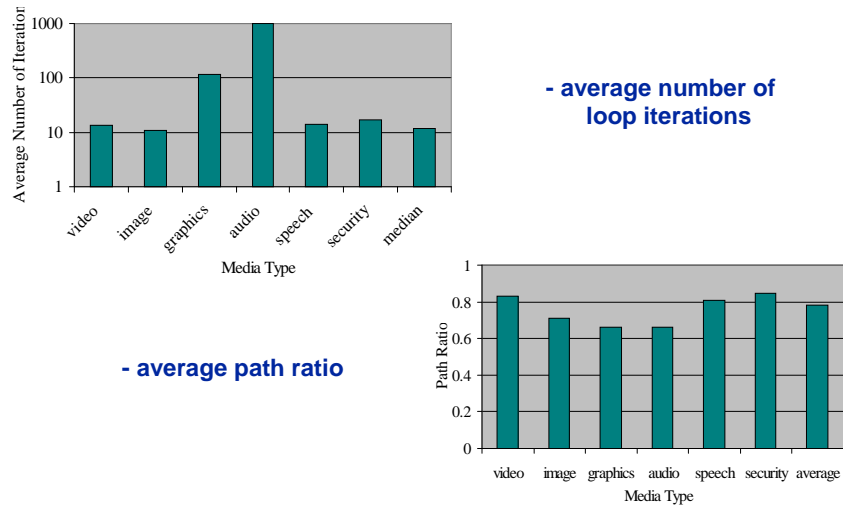
19

## Multimedia Looping Characteristics

- **Highly Loop Centric**
  - Nearly 95% of execution time spent within two innermost loop levels
- **Large Number of Iterations**
  - Significant processing regularity
  - About 10 iterations per loop on average
- **Path Ratio indicates Intra-Loop Complexity**
  - Computed as ratio of average number of instructions executed per loop invocation to total number of instructions in loop
  - Average path ratio of 78%
  - Indicates greater control complexity than expected

20

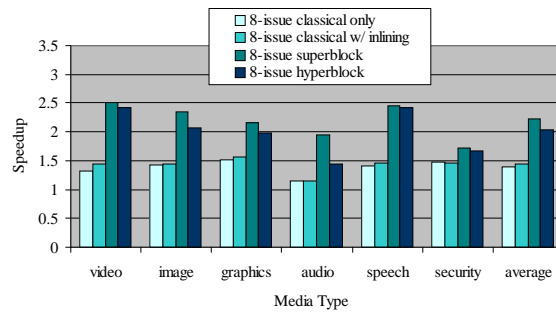
## Average Iterations per Loop and Path Ratio



21

## Instruction Level Parallelism

- **Instruction level parallelism**
  - base model: single issue using classical optimizations only
  - parallel model: 8-issue
- **Explores only parallel scheduling performance**
  - assumes an ideal processor model
  - no performance penalties from branches, cache misses, etc.



22

## *Workload Evaluation Conclusions*

---

- **Operation Characteristics**
  - More arithmetic operations; less memory and floating-point usage
  - Large variation in memory usage
  - (ALU, mem, branch, shift, FP, mult) => (4, 2, 1, 1, 1, 1)
- **Good Static Branch Prediction**
  - Multimedia: 10-15% avg. miss ratio
  - General-purpose: 20-30% avg. miss ratio
  - Similar basic block sizes (5 instrs per basic block)
- **Primarily Small Data Types (8 or 16 bits)**
  - Nearly 70% of instructions require 16-bit or smaller data types
  - Significant opportunity for subword parallelism or narrower datapaths
- **Memory**
  - Typically small data and instruction working set sizes
  - High data and instruction spatial locality
- **Loop-Centric**
  - Majority of execution time spent in two innermost loops
  - Average of 10 iterations per loop invocation
  - Path ratio indicates greater control complexity than expected

23

---

## *Architecture Evaluation*

24

## Architecture Evaluation

---

- **Determine fundamental architecture style**
  - **Statically Scheduled** => **Very Long Instruction Word (VLIW)**
    - allows wider issue
    - simple hardware => potentially higher frequencies
  - **Dynamically Scheduled** => **Superscalar**
    - allows decoupled data memory accesses
    - effective at reducing penalties from stall
- **Examine variety of architecture parameters**
  - Fundamental Architecture Style
  - Instruction Fetch Architecture
  - High Frequency Effects
  - Cache Memory Hierarchy
- **Related Work**
  - [Lee98] "Media Architecture: General Purpose vs. Multiple Application-Specific Programmable Processors," DAC-35, 1998.
  - [PChang91] "Comparing Static and Dynamic Code Scheduling for Multiple-Instruction-Issue Processors," MICRO-24, 1991.
  - [ZWu99] "Architecture Evaluation of Multi-Cluster Wide-Issue Video Signal Processors," Ph.D. Thesis, Princeton University, 1999.
  - [DZucker95] "A comparison of hardware prefetching techniques for multimedia benchmarks," Technical Report CSL-TR-95-683, Stanford University, 1995.

25

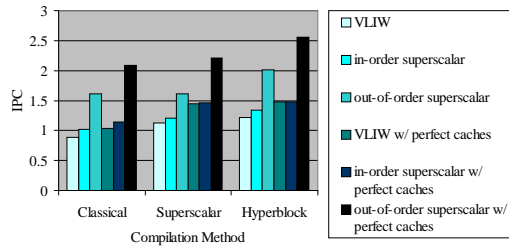
## Fundamental Architecture Evaluation

---

- **Fundamental architecture evaluation included:**
  - Static vs. dynamic scheduling
  - Issue width
- **Focused on non-memory limited applications**
  - Determine impact of datapath features independent of memory
  - Assume memory techniques can solve memory bottleneck
- **Architecture model**
  - 8-issue processor
  - Operation latencies targeted for 500 MHz to 1 GHz
  - 64 integer and floating-point registers
  - Pipeline: 1 fetch, 2 decode, 1 write back, variable execute stages
  - 32 KB direct-mapped L1 data cache with 64 byte lines
  - 16 KB direct-mapped L1 instruction cache with 256 byte lines
  - 256 KB 4-way set associate on-chip L2 cache
  - 4:1 Processor to external bus frequency ratio

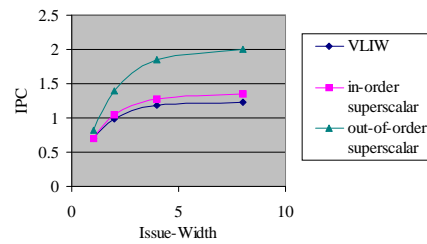
26

## Static versus Dynamic Scheduling



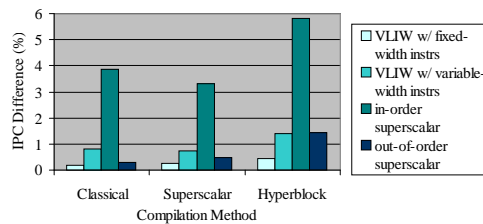
- static versus dynamic scheduling for various compiler methods

- result of increasing issue width for the given architecture and compiler methods



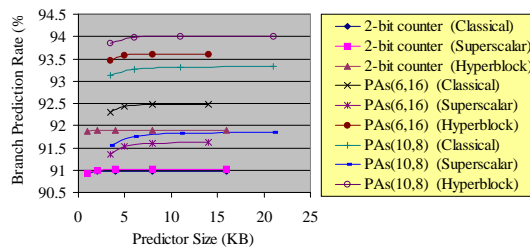
27

## Instruction Fetch Architecture



- aggressive versus conservative fetch methods

- comparison of dynamic branch prediction schemes



28

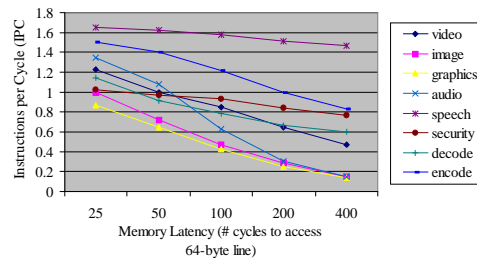
## *Impact of Higher Processor Frequencies*

- **Increased wire delay at higher frequencies may cause:**
  - Longer operation latencies
  - Delayed bypassing
- **Three processor models with different operation latencies**
  - 250 MHz – 500 MHz: stores – 1, loads – 2, FP – 3, mult – 3, div – 10
  - 500 MHz – 1 GHz: stores – 2, loads – 3, FP – 4, mult – 5, div – 20
  - 1 GHz – 2 GHz: stores – 3, loads – 4, FP – 5, mult – 7, div – 30
- **10% performance difference between processor models**
- **35% performance degradation for delayed bypassing**
- **Out-of-order scheduling and superscalar compilation least susceptible to high frequency effects**
  - 20-30% less performance degradation

29

## *Evaluation of Cache Memory Hierarchy*

- **L1 instruction and data cache**
  - Determined by working set size in workload evaluation
  - Good performance found with 32 KB data and 16 KB instruction caches
- **L2 cache**
  - Cache size
  - Miss latency
  - Line size
- **External Memory**
  - Miss latency
  - Bandwidth
- **Conclusions**
  - L2 cache has little impact on performance
    - useful for storing state during context switches
  - External memory miss latency is primary memory problem
    - Streaming data structures will help alleviate this
  - External memory bandwidth is second-most problem



30

## *Architecture Evaluation Conclusions*

---

- **Fundamental Architecture Style**
  - VLIW and In-order superscalar are comparable
  - Out-of-order superscalar has 70% better performance
  - Hyperblock is most effective compilation technique
  - Issue widths of 3-4 are sufficient
- **Instruction Fetch Architecture**
  - Small dynamic branch predictor provides good performance
  - Aggressive fetch provides little benefit
  - 2% performance degradation for additional pre-execute pipeline stages
  - Instruction fetch is not critical in media processors
- **High Frequency Effects**
  - 10% performance difference between processors with varying operation latencies
  - 35% performance degradation from delayed bypassing
  - Out-of-order superscalar and superscalar compilation least affected
- **Cache Memory Hierarchy**
  - L1 cache size has little effect on media processing
  - External memory latency and bandwidth are primary bottlenecks

31

---

## *Parallelism Extraction by Compiler*

32

## *Parallelism in Multimedia*

---

- **Studies indicate significant parallelism**
  - Trace-driven studies (Oracle experiments)
- **Instruction Level Parallelism (ILP)**
  - Performance similar to general-purpose applications
- **Subword Parallelism**
  - Has shown 2-4x speedup with hand scheduling
  - Compiler methods under development
- **Data Parallelism (Loop-based Parallelism)**
  - Coarser level of parallelism than ILP
  - Granularity of 250+ operations per parallel data element
  - Conventional compilers cannot extract data parallelism
  - Significant potential, but currently most underutilized means of parallelism
- **Task-Level Parallelism**
  - Typically requires explicit parallel programming by user
  - Enables best results

33

## *Hand-Scheduling Experiment - Traditional DCT*

---

```

for (m = 0; m < 8; m++)
  for (n = 0; n < 8; n++) {
    y[m][n] = 0;
    for (i = 0; i < 8; i++)
      for (j = 0; j < 8; j++)
        y[m][n] += y[m][n] + x[i][j] * c[i][m] * c[j][n];
  }

```

```

r0 = m      r10 = &x[0][0]
r1 = n      r11 = &x[i][0]
r2 = i      r12 = &c[0][n]
r3 = j      r13 = &c[0][m]
r4 = m * 4  r14 = &c[i][m]
r5 = n * 4  r15 = x[i][j]
r6 = i * 4  r16 = c[i][m]
r7 = j * 4  r17 = c[j][i]
r8 = i * 4 * 8  r20 = y[m][n]
r9 = j * 4 * 8  r21 = &y[m][n]

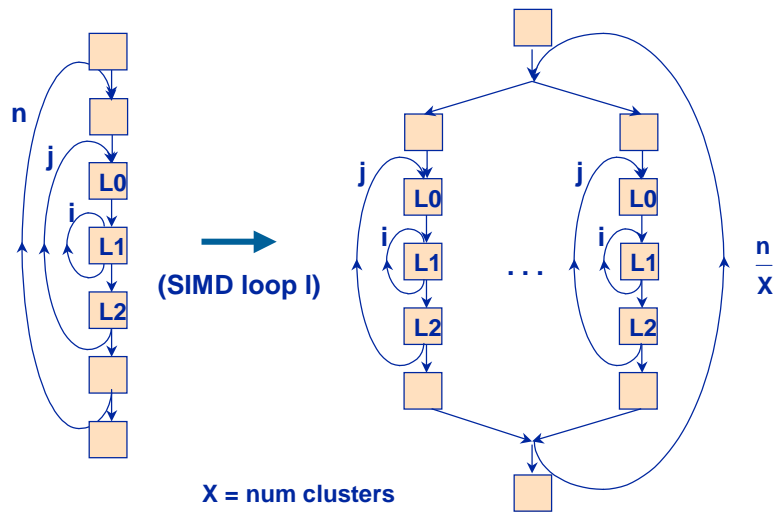
```

Two inner loops:

L0: mult r6, r2, 4	(1)
mult r8, r6, 8	(2)
add r11, r10, r8	(3)
load r14, r13, r8	(4)
add r20, 0, 0	(5)
L1: mult r7, r3, 4	(6)
mult r9, r7, 8	(7)
load r15, r11, r7	(8)
load r17, r12, r9	(9)
mult r18, r16, r15	(10)
mult r19, r18, r17	(11)
add r20, r20, r19	(12)
add r3, r3, 1	(13)
bge r3, 8, L1	(14)
L2: store r21, r20	(15)
add r2, r2, 1	(16)
bge r2, 8, L0	(17)

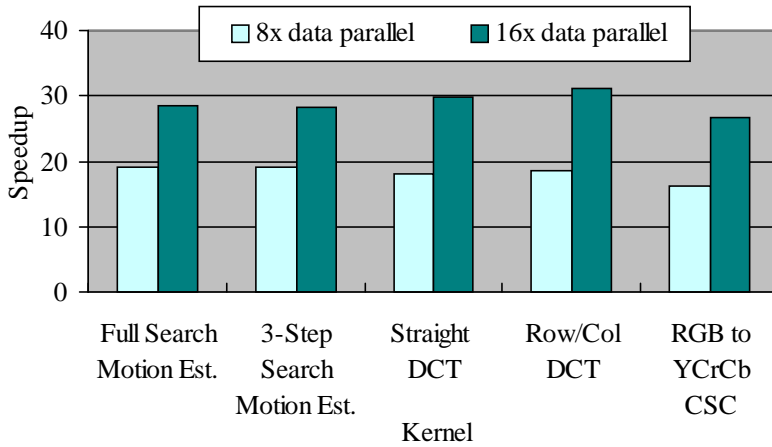
34

## *SIMD Parallelism Across Clusters*



35

## *Speedup from SIMD Parallelism Across Clusters*



36

## *Parallel Compiler and Architecture Methods*

---

- **Determine independence of loop iterations**

- Static parallelization by a parallel compiler
  - memory disambiguation may not be able to determine parallelism
  - must conservatively assume loop iterations are not parallel
- Run-time parallelization enables optimistic parallelization
  - non-speculative -> *inspector/executor* approach
  - speculative -> dynamic memory conflict checking

- **Speculative run-time methods**

- LRPD Test (Rauchwerger)
- Multiscalar Project (Sohi)
- Thread-Level Data Speculation (TLDS) (Mowry)
- Thread-Level Speculation (TLS) (Olukotun, Lam)

[LRauchwerger95] "The LRPD Test: Speculative Run-Time Parallelization of Loops with Privatization and Reduction Parallelism," PLDI, 1995.

[MFranklin93] "The Multiscalar Architecture," Ph.D. Thesis, University of Wisconsin at Madison, 1993.

[JGregory98] "The Potential for Using Thread-Level Data Speculation to Facilitate Automatic Parallelization," HPCA-4, 1998.

[JOplinger97] "Software and Hardware for Exploiting Speculative Parallelism with a Multiprocessor," Technical Report CSL-TR-97-715, Stanford University, 1997.

37

## *Speculative Broadcast Loop (SBL) Execution*

---

- **Speculative SIMD parallelism method for data parallelism**

- Simplified version of Multiscalar, TLDS, TLS approaches
- Supports fully and partially parallel loops
- Does not support parallelism across function boundaries

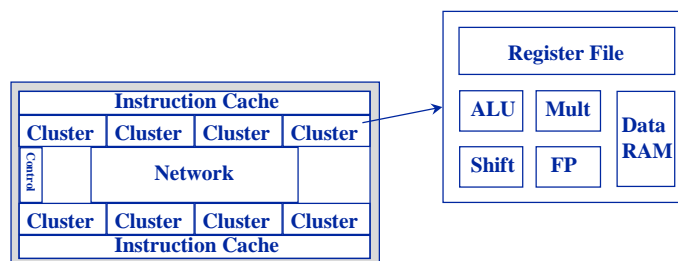
- **SBL method ideal for multi-cluster architecture**

- Single instruction control stream may be broadcast to all clusters for SIMD parallel execution
- Independence of clusters corresponds well with independence of data elements (loops)
- Processing regularity of data parallelism in multimedia makes it conducive for SIMD parallelism
- Control synchronization is implicit in single control stream
  - only need synchronization for ensuring correct memory image
- Separate clusters provides automatic scalar privatization

38

## Wide Issue => Clustered Architecture

- Global register file too slow for wide issue
- Separate datapath into disjoint clusters:
  - functional units (2-4 issue slots)
  - local register register file
  - local memories/caches
- Low latency network between clusters



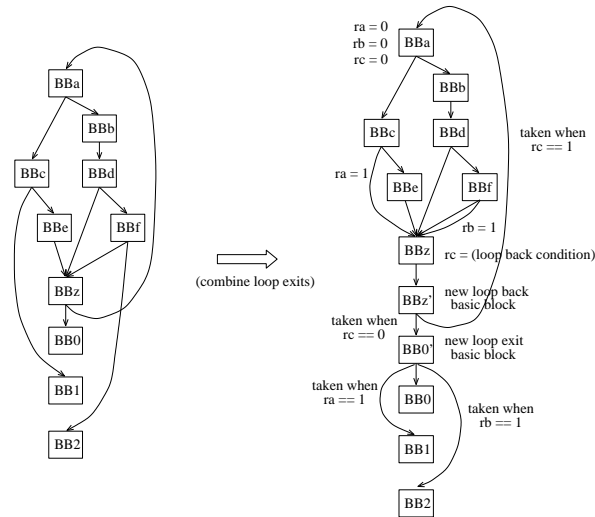
39

## Implementing SBL Execution

- Major aspects of SBL execution
  - Loop broadcast and multi-level loop scheduling for SIMD parallelism across a multi-cluster architecture
  - Hardware support for large-scale speculative parallel execution
- Steps to implement SBL execution
  - Find potentially parallel loops using profiling and register dependence analysis
  - Select loops for broadcast and schedule them to eliminate all unnecessary branches (Multi-Level If-Conversion)
  - Provide speculative hardware for support large speculative state
- Multi-Level If-Conversion
  - Inner loops: combine all control paths into single control path
  - Nested loops: recursively if-convert nested loop regions

40

## If-Conversion of Inner Loops



41

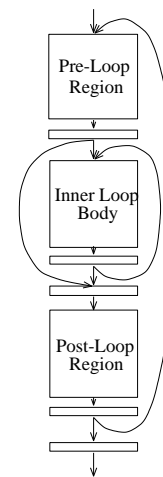
## If-Conversion of Nested Loop Regions

- Divide nested loop into inner loops, preloop regions, and post loop regions
- If-Conversion is similar for inner loops, preloop regions, and post loop regions
- Recursively apply nested loop if-conversion until no remaining nested loops

```

combine_loop_exits(main_loop) {
  if (main_loop has child loops) {
    for each (child_loop) {
      combine_loop_exits (child_loop);
      combine_pre_loop_exits (child_loop);
    }
    combine_post_loop_exits ( child_loop);
  }
  else
    combine_inner_loop_exits (child_loop);
}

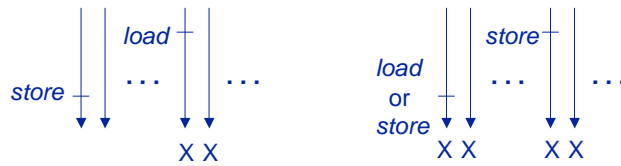
```



42

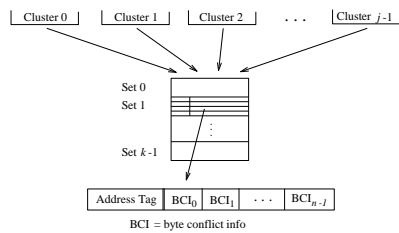
## Speculative Hardware Support for SBL Execution

- **SBL method must support speculation for**
  - Memory Independence Speculation
  - Control Flow Speculation
  - Parallel Loop Iteration Speculation
- **For large speculative state, use L1 data cache**
  - Also need checkpoints and a restorable register file
- **Loop Memory Conflict (LMC) cache monitors all memory accesses**
- **Memory conflict checking:**
  - Compares memory accesses for separate iterations
  - Clusters have left-to-right ordering
  - Store-load conflict
    - cancel iterations beyond load
  - Load-store/store-store conflict
    - cancel iterations beyond first load or store



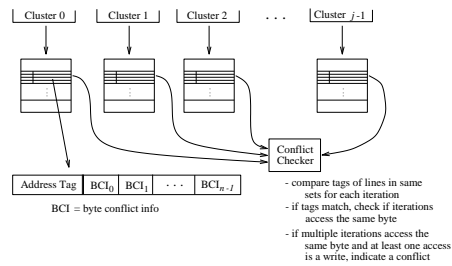
43

## Loop Memory Conflict (LMC) Cache



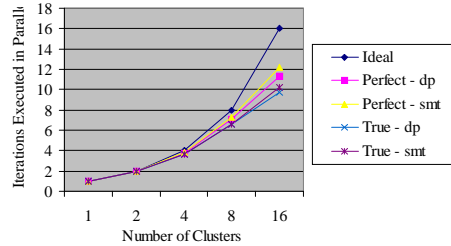
- global LMC cache

- distributed LMC cache



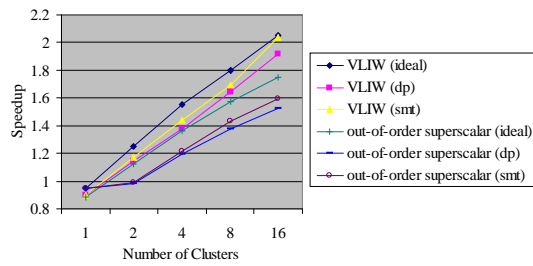
44

## Performance of SBL Execution



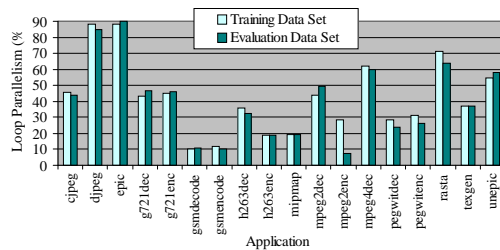
- speedup of SBL execution on parallel loops in MediaBench

- speedup of SBL method for full applications



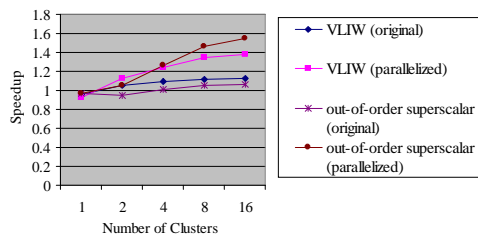
45

## Improving Performance of SBL Execution



- parallelism found by SBL execution

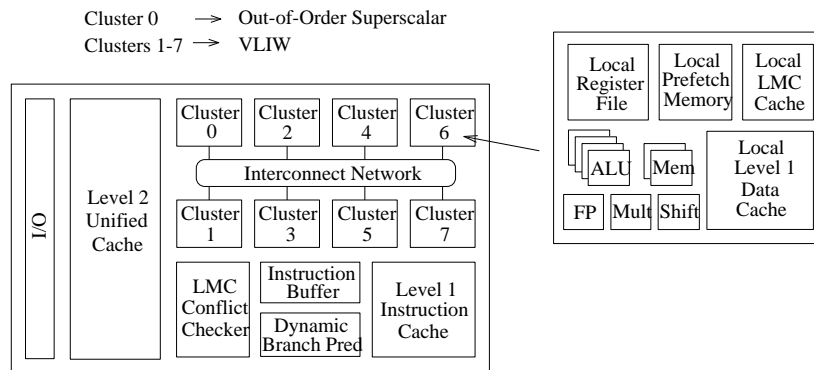
- speedup from manually applying parallel compiler optimizations



46

## *Proposal for High-Performance Programmable Media Processor*

---



47

## *Conclusions and Contributions*

---

- **Comprehensive Evaluation of Multimedia Characteristics**
  - Higher arithmetic and lower memory operation frequencies
  - Excellent static branch prediction
  - Small working set sizes and high spatial locality
  - Highly loop-centric, but greater intra-loop complexity than expected
- **Datapath Architecture Evaluation for Media Processors**
  - Dynamic scheduling has 70% better performance than in-order scheduling
  - Instruction fetch not critical in media processing
  - Longer operation latencies and delayed bypassing significantly affect IPC
- **Evaluation of Cache Memory Hierarchy**
  - External memory latency and bandwidth are primary bottlenecks
- **Investigation of Parallelism in Multimedia**
  - ILP offers moderate speedup
  - Subword parallelism requires further compiler research
  - Data parallelism offers promising parallelism
- **Speculative Broadcast Loop (SBL) Method**
  - Novel speculative run-time method for supporting data (SIMD) parallelism
  - Multi-level if-conversion scheduling to maximize SIMD parallelism
  - Achieved 2x speedup; parallel compiler optimizations promise further parallelism

48

## *Future Research in Programmable Media Processors*

---

- **Extend Multi-Level If-Conversion to Subword Parallelism**
- **Use Parallel Compiler to Extract Data Parallelism**
  - Available parallelism for static SIMD parallelism
  - Available parallelism for SBL execution method
  - Available parallelism for static multiprocessor parallelism
- **Evaluate SBL Method on Single-Chip Multiprocessor**
  - Available parallelism for SBL execution on multiprocessor
- **Multi-Level Prefetch Hierarchy**
  - Conservative prefetching on-chip; aggressive prefetching off-chip
- **Compiler Support for Specialized Functional Units**
  - Motion estimation, DCT, variable-bit rate coding, etc.
- **Evaluating DSP Features**
  - DSP operations: multiply-accumulate, saturation arithmetic, etc.
  - Low-overhead looping