

# *Parallel Media Processors in the Billion-Transistor Era*

---



**Jason Fritts**

*Princeton University*

*Dept. of Electrical Engineering*

*Co-Authors: Zhao Wu and Wayne Wolf*

## *Overview*

---

- **Why Parallel Media Processors (PMPs)?**
- **Architecture Issues**
  - datapath
  - memory hierarchy
  - potential performance over next decade
- **Compiler Issues**
  - extracting data parallelism
  - compiling to highly parallel architectures
  - potential parallel performance
- **Conclusion**

2

## *Next Decade of VLSI Technology*

---

- **Billion-transistor chips**
- **Multi-gigahertz processors**
- **Impact on multimedia processing industry?**
  - still evolving -- strongly dependent upon computing performance
  - must consider future of multimedia

**Multimedia processing industry evolves  
with processor performance**

3

## *Future of Multimedia*

---

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

- **Currently represented as video frames or audio channels**
  - data organized as fixed-sized blocks/groups of continuous data
  - processing regularity
- **Moving towards object-based representation**
  - objects represent real-world objects
    - objects have own video, audio, and graphical characteristics
    - advantages: greater compression, more interactive, content-based processing
  - greater processing demands
  - less processing regularity
  - requires flexibility of high-level language (HLL) programmability

4

## *Multimedia Processing Solutions*

---

- **Multimedia has distinctive workloads**
  - extensive data parallelism
  - high computation rates
  - large amounts of streaming data
  - small data types
- **Current multimedia processing support**
  - application-specific processors
  - multimedia extensions to general-purpose processors
  - current media processors
    - VLIW or DSP style architectures
    - performance achieved through specialized functional units
    - flexibility, but require special programming libraries or paradigms
- **Next generation of multimedia requires:**
  - flexibility of high-level language (HLL) programmability
  - computing power for all forms of multimedia

5

## *Future of Media Processors*

---

- **Evolve towards Parallel Media Processors (PMPs)**
  - increased parallelism
  - achieve throughput from both high parallelism and high frequency
- **Larger on-chip memory hierarchies**
  - accommodate large amounts of data
  - minimize penalties from long external memory latencies
- **Greater architecture regularity**
  - greater flexibility over a wide range of applications
  - better high-level language (HLL) programmability
    - no special libraries or programming paradigms

6

---

## *Architecture Issues*

7

---

## *What to do with a Billion Transistors?*

- **More functional units?**
  - extra silicon enables higher degrees of parallelism
  - however, functional units require minimal area
- **Larger memory hierarchies?**
  - many multimedia applications access memory intensively
    - especially video and graphics
  - larger on-chip memory helps reduce increasing processor-memory gap
- **Majority of extra resources likely used for memory**

8

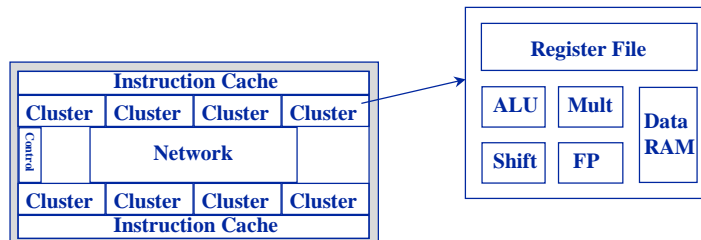
## *Datapath Issues*

- **Datapath architecture?**
  - variety of architectures exist:
    - superscalar, VLIW, single-chip multiprocessors, simultaneous-multithreading, vector processor, array processors, etc.
  - uncertain which alternative is most desirable
- **Throughput is primary datapath problem**
  - both high parallelism and high frequency necessary for multimedia computing demands
  - unfortunately, high parallelism and high frequency are counter-productive
    - high parallelism leads to excessive bypassing, memory ports, and register file ports
- **Distributed architecture necessary**
  - one possibility: clustered architecture

9

## *Wide Issue => Clustered Architecture*

- **Global register file too large, 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**



10

## *Memory Hierarchy Issues*

---

- **Memory hierarchy still undefined**
  - memory? with prefetching?
  - cache? with prefetching?
  - hybrids?
- **Must support multimedia memory access patterns**
  - high data rates
  - streaming data
  - high spatial locality
- **Streaming memory structures likely**
  - stream buffer
  - stride prediction table
  - hybrids
  - where in hierarchy will they be most effective?

11

## *Memory Hierarchy Issues, cont.*

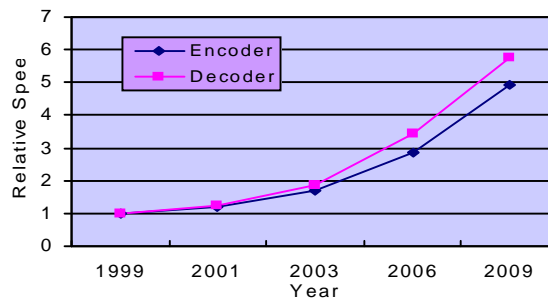
---

- **Must support parallel memory accesses**
  - large multi-ported memory too slow
  - require either:
    - highly banked memory
    - distributed memory
- **Primary truths about memory hierarchies**
  - need aggressive memory hierarchies for high data rates
  - increased silicon resources will enable many interesting alternatives over next decade

12

## Potential Performance of PMPs over Next Decade

- **Scaling based on National Semiconductor Roadmap:**
  - scaled processor frequency and number of transistors
  - additional transistors primarily applied to memory hierarchy
- **MPEG-2 evaluated with trace-driven simulator**
  - clustered architecture with 8 clusters of 4-issue slots/cluster
  - perfect branch prediction and memory disambiguation
  - scheduling window of one billion operations



13

## Additional Architecture Problems

- **Dynamic versus static scheduling**
  - static scheduling sufficient for media processing?
  - dynamic methods desirable?
    - out-of-order execution, dynamic branch prediction, dynamic memory disambiguation, etc.
- **Larger versus smaller datapath widths**
  - small data types amenable to narrower datapath
    - potential for higher frequency
    - support for pointers?
  - larger datapath conducive to subword parallelism (e.g. Intel's MMX)
- **Supporting mixed-signal multimedia**
  - designing media processors to support multiple types of multimedia
  - how to support simultaneous processing of multiple media?
    - context-switching, simultaneous multi-threading, multiprocessing, etc.

14

---

## *Compiler Issues*

15

---

## *Compilers for Media Processors*

- **Aggressive optimizing compiler**
  - multimedia offers extensive parallelism
  - throughput requires successful extraction of parallelism by compiler
- **Compiler and architecture balanced**
  - achieve high utilization of all architecture resources
  - allow each to tackle problems too difficult for the other
    - compiler: static optimizations
    - architecture: dynamic optimizations
- **Chief requirements**
  - extracting parallelism
  - scheduling to highly parallel (distributed) architectures

16

## *Parallelism in Multimedia*

---

- **Studies indicate significant parallelism**
  - trace-driven (Oracle-style) studies
  - parallelism has been utilized in many systems
- **Instruction Level Parallelism (ILP)**
  - fine-grained parallelism between individual instructions
  - multimedia contains no more ILP than general-purpose code
- **Data Parallelism**
  - exists between data elements whose processing are independent
  - coarser level of parallelism than ILP
  - generally separated by 500+ dynamic operations

17

## *Extracting Data Parallelism*

---

- **Conventional compilers cannot extract data parallelism**
  - good at scheduling for ILP
  - schedule over windows of typically less than 100 sequential instructions
  - scheduling windows too small for data parallelism
- **Compilation method**
  - execute separate iterations of outer loops in parallel
    - SIMD parallelism
  - perform any optimization on inner loops

18

## Example - Straightforward 2-D DCT

- **Two inner loops**
  - dependent on accumulation of  $y[k][l]$
- **Two outer loops**
  - independent
  - minimum distance between data parallel elements is  $9 \times 64 = 576$  instructions

```

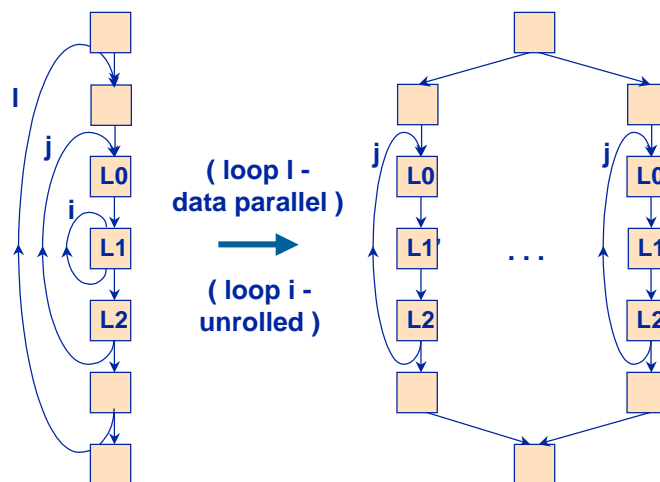
for (k = 0; k < 8; k++)
  for (l = 0; l < 8; l++)
  {
    y[k][l] = 0;

    for (i = 0; i < 8; i++)
      for (j = 0; j < 8; j++)
        y[k][l] += y[k][l] + x[i][j] * c[i][k] * c[j][l];
  }

```

19

## Exploiting Data Parallelism for Straightforward 2-D DCT



20

## *Compiling to Highly Parallel (Distributed) Architectures*

---

- **Communication in distributed architectures**
  - operation results not uniformly available to all clusters at same time
  - clusters interconnected by a low latency network
- **Cluster scheduling**
  - copy operations scheduled by compiler for inter-cluster communication
  - compiler attempts to schedule critical operations on same cluster to minimize performance degradation
- **Cluster scheduling for data parallelism**
  - independence of data parallelism matches well with independence of clusters
  - data parallel elements processing on separate clusters require little or no inter-cluster communication

21

## *Data Partitioning*

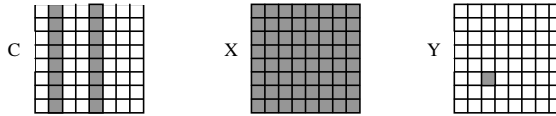
---

- **Memory for wide-issue architecture requires either:**
  - highly banked cache
  - separate cache or memory in each cluster
- **Evaluate data parallelism at various levels**
  - determine loop levels that offer data parallelism
  - examine memory requirements at each level
  - choose level with:
    - largest independent data set
    - acceptable granularity of independent data
    - least amount of copied data (read sharing)

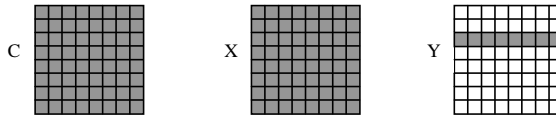
22

## Example - DCT

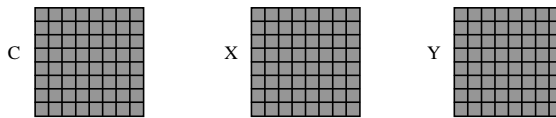
a) Single iteration of  $l$  loop (1/64 of block)



b) Single iteration of  $k$  loop (1/8 of block)



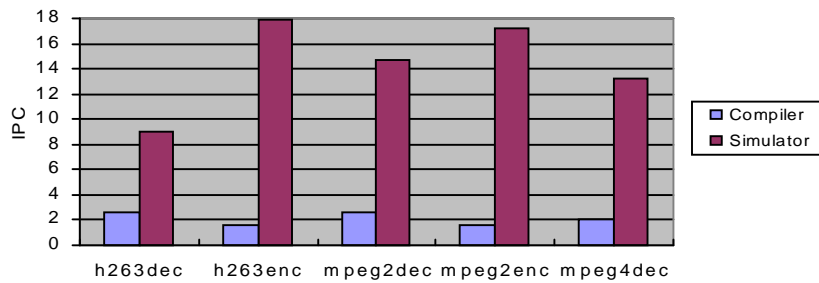
c) Full 8x8 block



23

## Performance of Compilation versus Trace-Driven Simulation

- **Trace-driven simulator**
  - clustered VLIW architecture with 8-clusters of 4-issue slots/cluster
  - assumes perfect branch prediction and memory disambiguation
  - scheduling window of one billion instructions
- **IMPACT compiler**
  - 32-issue statically-scheduled unclustered VLIW architecture

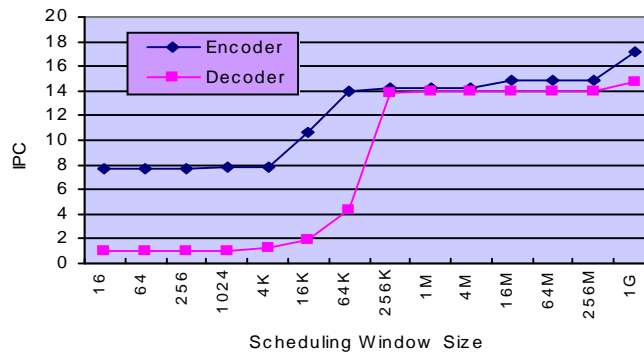


24

## *Granularity of Data Parallelism in MPEG-2*

- **Performance relative to scheduling window size**

- used trace-driven simulator
- varied size of scheduling window
- true data parallelism smaller than indicated
  - scheduler does not eliminate loop-carried dependencies from loop induction variables



25

## *Conclusions*

- **Much potential for PMPs over next decade**

- advances in VLSI technology will enable single-chip PMPs
- many research issues remain to realize full potential

- **Architecture Issues**

- majority of additional silicon resources applied to memory hierarchy
- wider datapaths
  - throughput requires high parallelism and high frequency
- larger memory hierarchies
  - streaming memory structures likely
  - must support many parallel memory accesses

- **Compiling Issues**

- extracting data parallelism
- compiling to highly parallel (distributed) architectures
  - cluster scheduling
  - data partitioning

26