Analyzing Scalability Limits of H.264 Decoding due to TLP Overhead

Mauricio Alvarez[1], Arnaldo Azevedo[2], Cor Meenderinck[2], Ben Juurlink[2], Andrei Terechko[3], Jan Hoogerbrugge[3] and Alex Ramirez[4].

[1]Universitat Politècnica de Catalunya (UPC). Spain

[2] Delft University of Technology. The Netherlands

[3]NXP. The Netherlands.

[4]Barcelona Supercomputing Center. Spain

HiPEAC Computing Systems Week.

Thales, Nov 26th, 2008, France

Outline

1. Motivation
2. Brief introduction to parallelization of H.264
3. Parallel scalability of H.264 decoding
4. Conclusions and future work

Media Application Trends

- Towards better quality: H264: SD, HD, Quad HD
  - High computational complexity
- Towards mobile & integrated systems:
  - Embedded systems with real-time, power, and cost constraints
- Towards multiple formats and extensions:
  - Programmable processors instead of application specific hardware

Chip Multicores

- Multicore/manycore architectures
  - Massive Thread Level Parallelism
- Heterogeneous multicore architectures
  - Exploiting full potential performance requires specialization
- Different memory architectures
  - Different parallel versions of applications for different memory architectures

Video Application Challenges

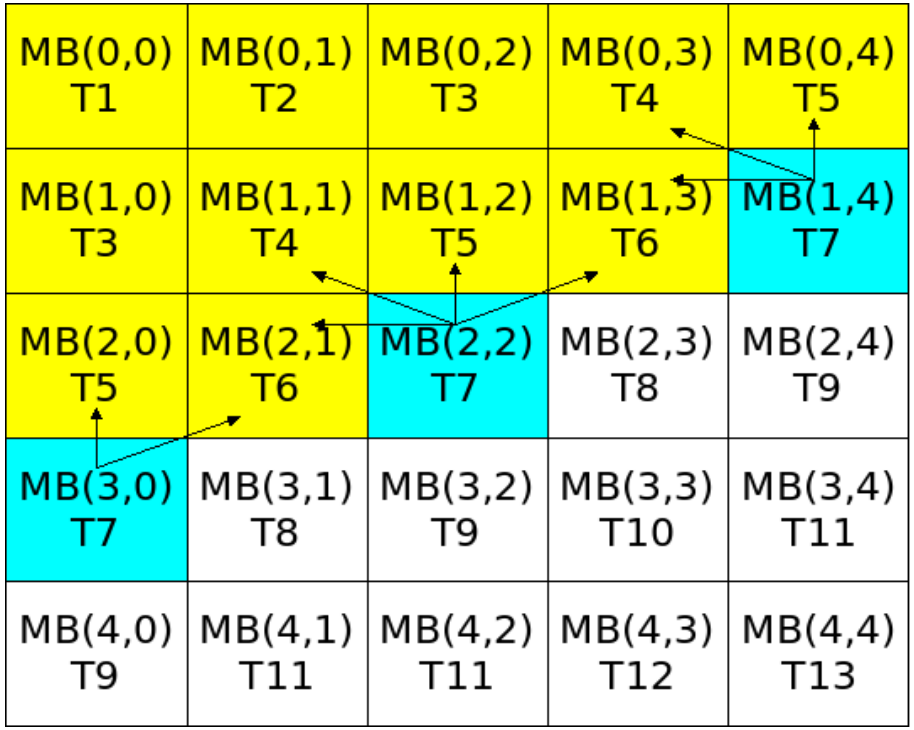
- From digital video trends
  - High quality translates in high computational complexity
  - Embedded applications impose real-time and power constraints.
  - Multiple video formats and new extensions requires programmability.
- Opportunities of new microprocessors:
  - Multicore architectures with hundreds of cores on a single die offer a huge performance potential.
- Main Goal:
  - Parallelize video applications in order to take advantage of multicore architectures in an efficient and scalable way
  - Identify bottlenecks and limiting factors for scalability in order to propose architecture enhancements.

Outline

1. Motivation
2. Brief introduction to parallelization of H.264
3. Parallel scalability of H.264 decoding
4. Conclusions and future work

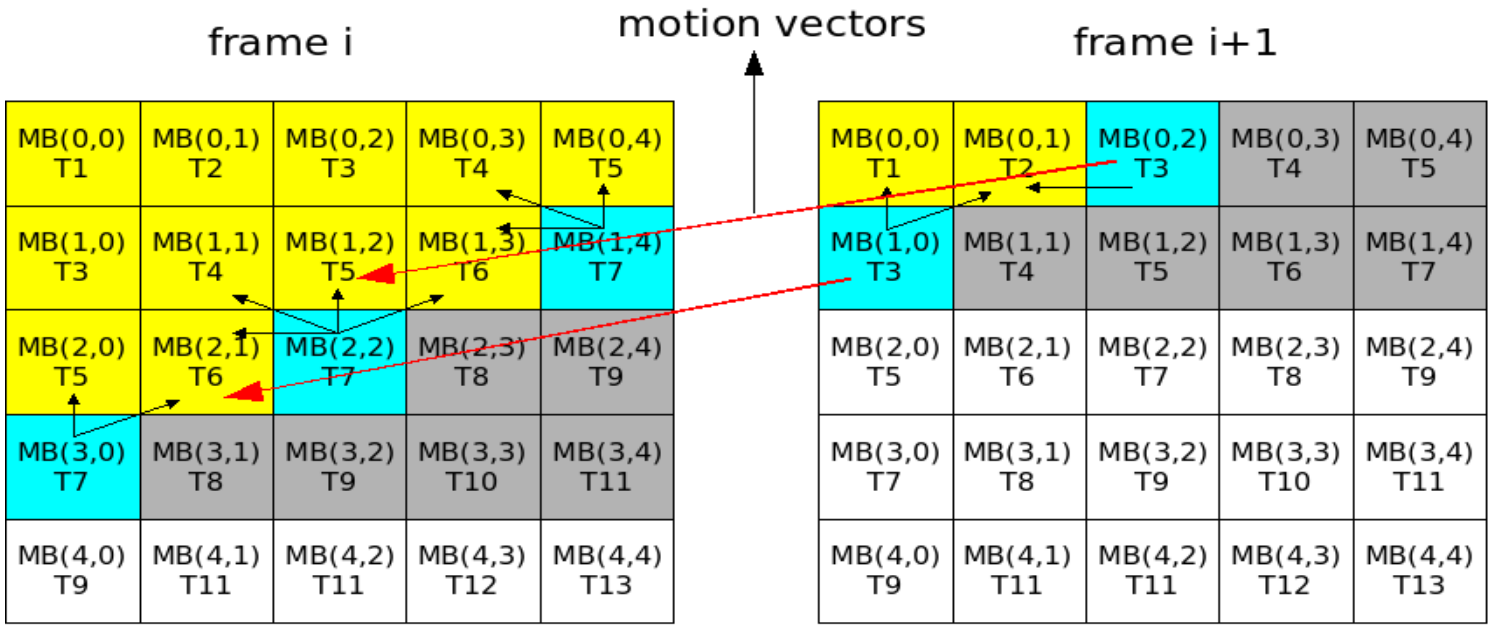
2D-wave: Intra-frame TLP

- Each macroblock (MB) requires data from neighbor MBs
- 2D-wave order allows parallel decoding of MBs in the same frame



3D-wave TLP

- MBs have inter-frame dependencies in the Motion Compensation kernel
- 3D-wave = Temporal parallelism + Spatial parallelism
  - 2D-wave decoding of each frame
  - Multiple frames decoded in parallel

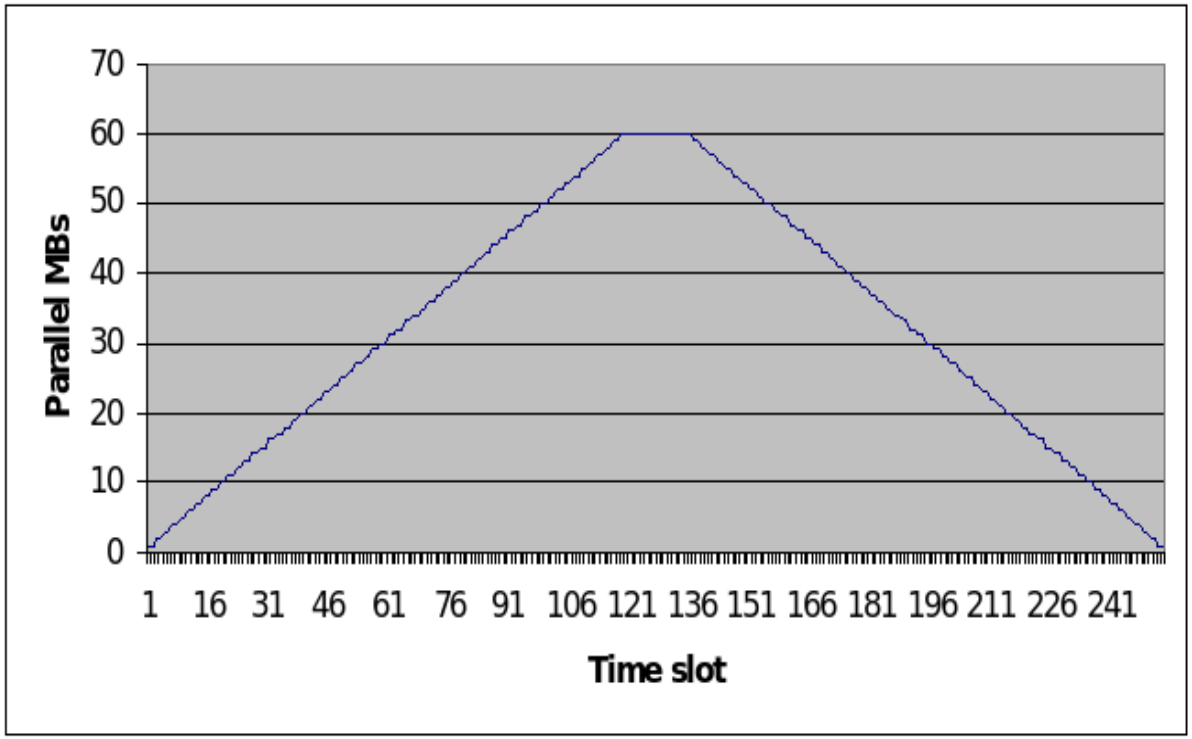


C. Meenderinck et al. Journal of VLSI systems. 2008

Outline

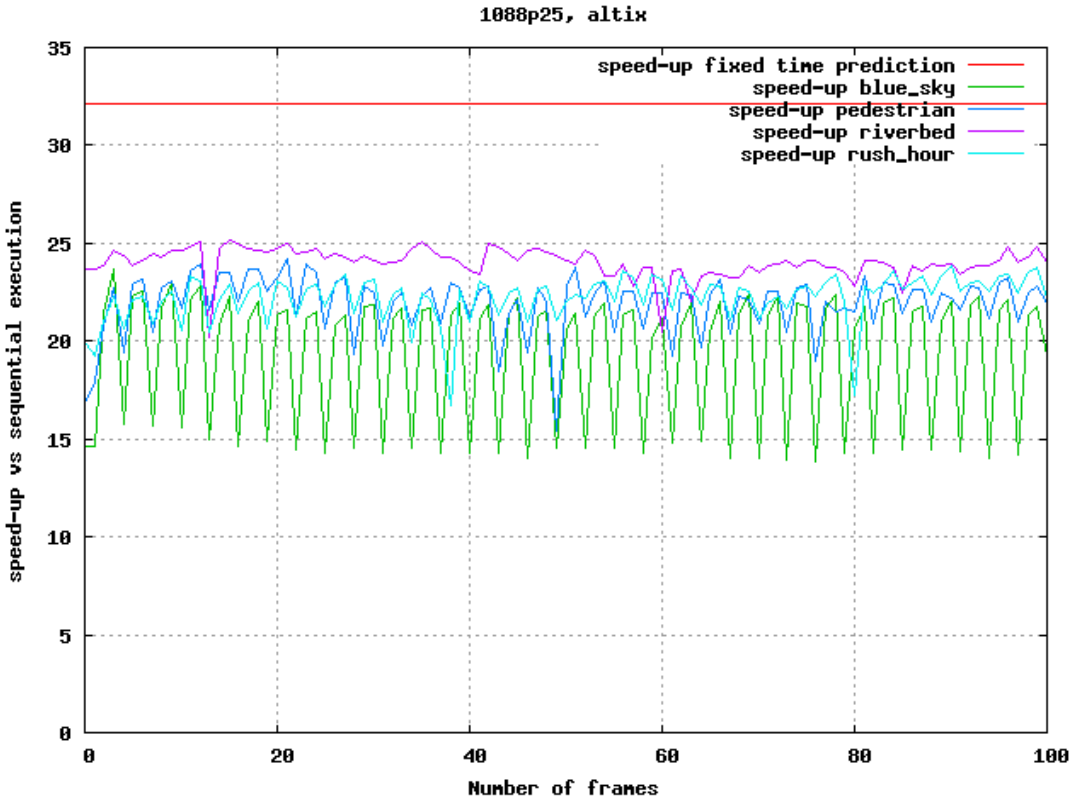
1. Motivation
2. Brief introduction to parallelization of H.264
3. Parallel scalability of H.264 decoding
   a) 2D-wave on a shared memory multiprocessor
   b) 3D-wave on an embedded multicore processor
4. Conclusions and future work

Scalability of 2D-wave



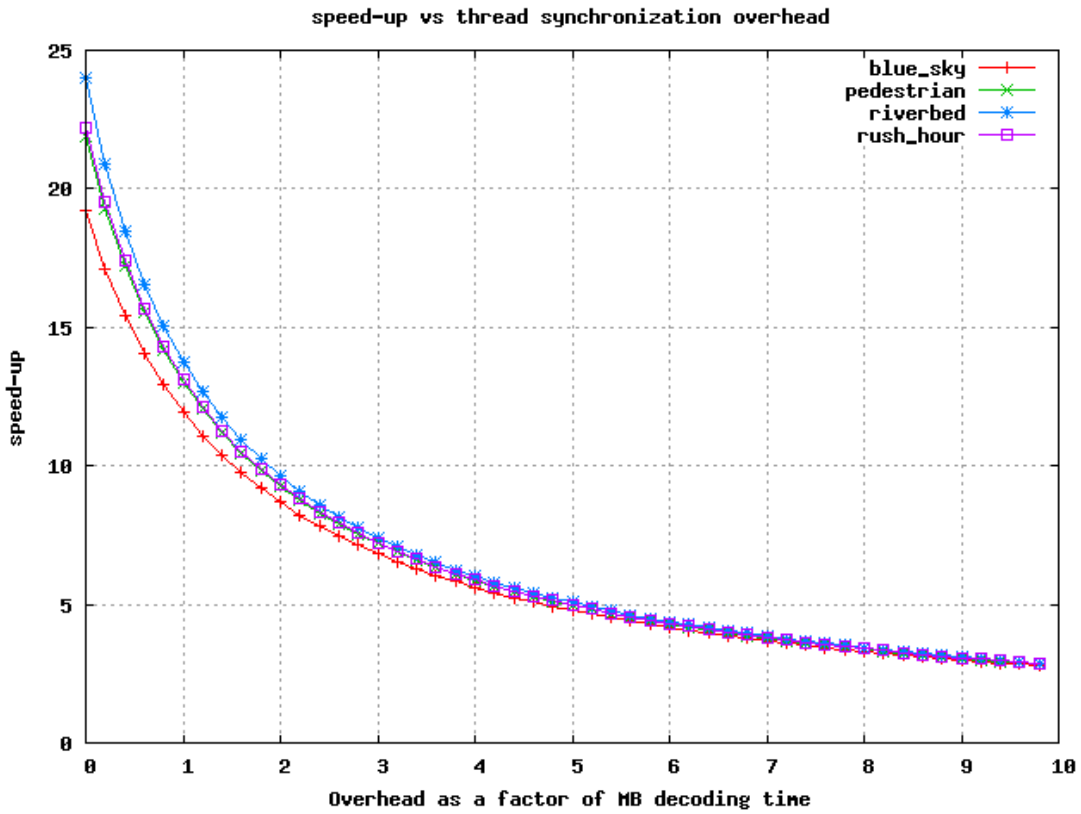
- FHD: Up to 60 independent MBs
- Theoretical maximum speedup: 32.13 X
  - Assuming constant processing time
  - Assuming no overhead from thread synchronization

Effects of variable processing time



- The processing time of each MB is input dependent.
- The operations applied to the image samples depend on the values of those samples.
- On average, maximum speedup is reduced from 32.1 to approx. 21.5

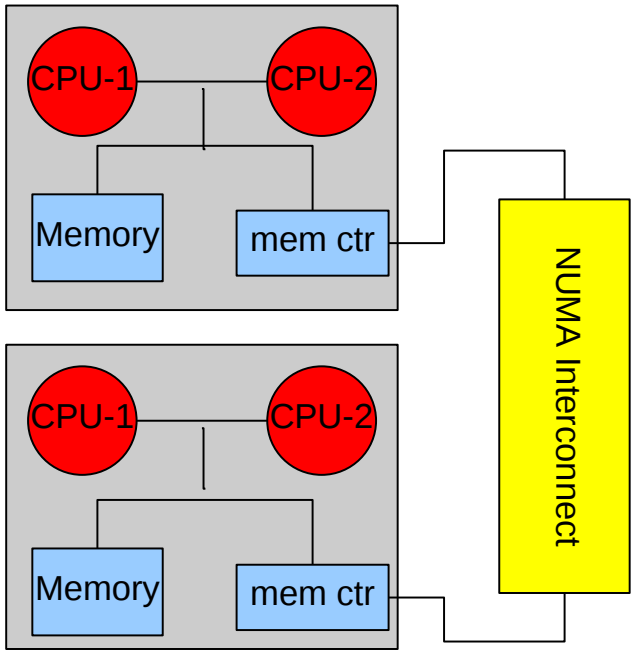
Effects of thread synchronization overhead



- Overhead as a factor of average MB decoding time
- Results obtained from a trace-driven “abstract” simulator
- For some implementations overhead ratio goes up to 12X

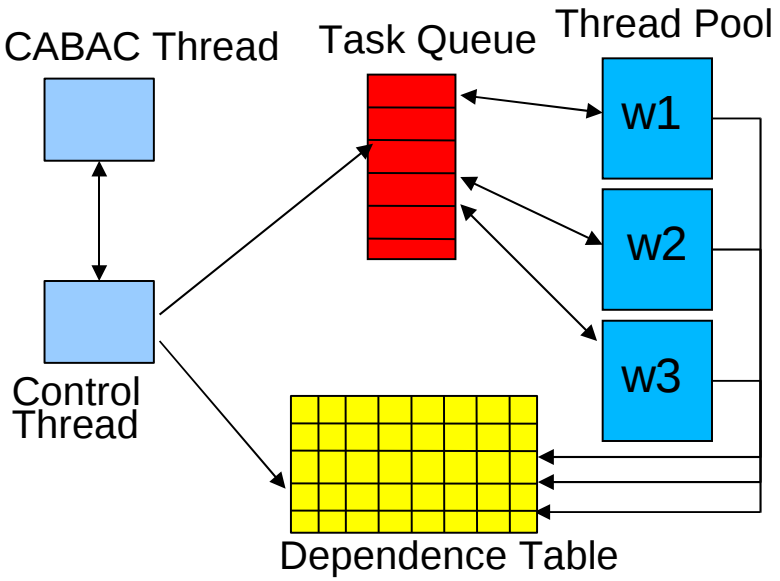
Implementation of 2D-wave on a multiprocessor

Architecture



- SGI Altix
- cc-NUMA architecture
- 64 dual core IA-64 processors
- Each core works at 1.6 GHz,
- Each core has 8MB L3 cache

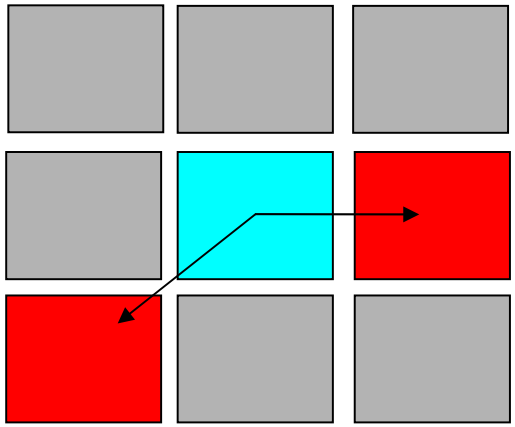
Programming model



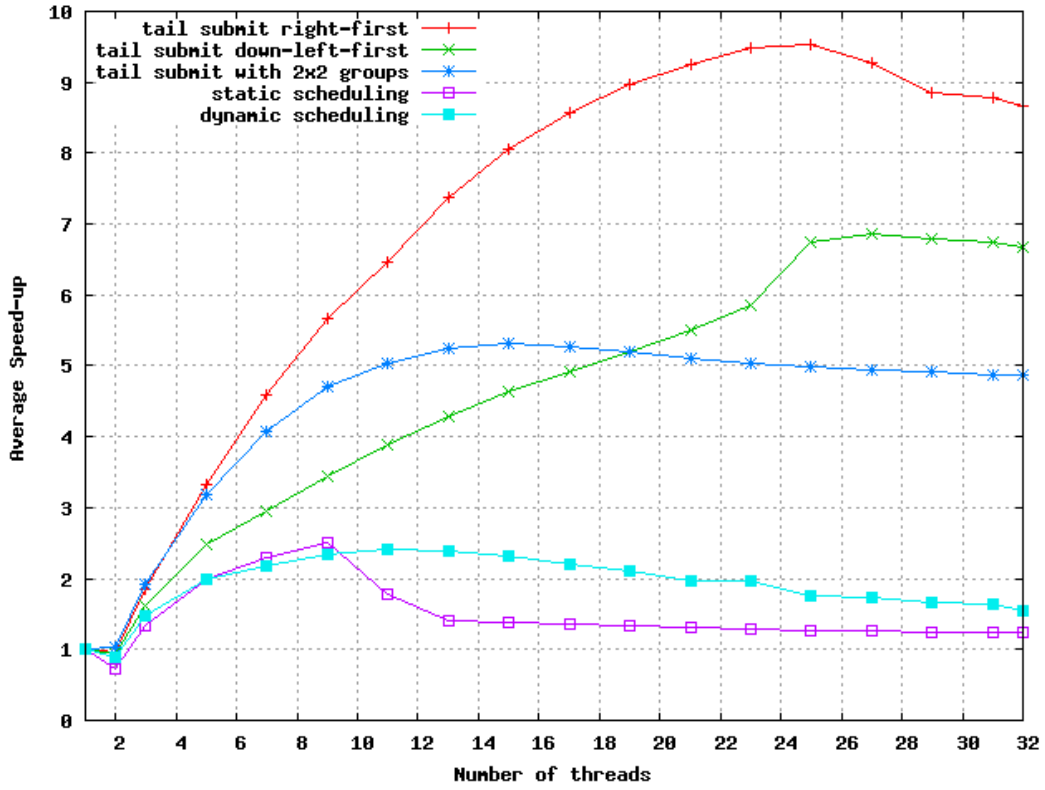
- 1 control thread
- N worker threads
- CABAC entropy decoding is done before parallel MB decoding
- POSIX threads and real-time semaphores

Speed-up on the Altix Machine

- Four different scheduling strategies
  - Static
    - Predefined scheduling and mapping
  - Dynamic
    - Centralized task queue
  - Dynamic with tail submit
    - Worker threads can assign work to themselves
    - Down-left-first MB order
    - Right-first MB order



Speed-up on the Altix Machine



- Dynamic scheduling with Tail submit with right first-order has best performance: reduces task overhead and exploits locality
- Efficiency is very low: 9.5 speed-up with 25 processors

Overhead/processing ratio in 2D-wave Altix implementation

- Dynamic scheduling is able to discover parallelism but suffers from contention in the centralized task queue
- Tail submit reduces the synchronization overhead but it is still significant

Threads	Dynamic sched	Tail submit
1	0.67	0.17
2	0.97	0.17
4	1.30	0.22
8	2.05	0.30
14	6.57	0.68
24	10.49	1.00
32	18.88	1.85

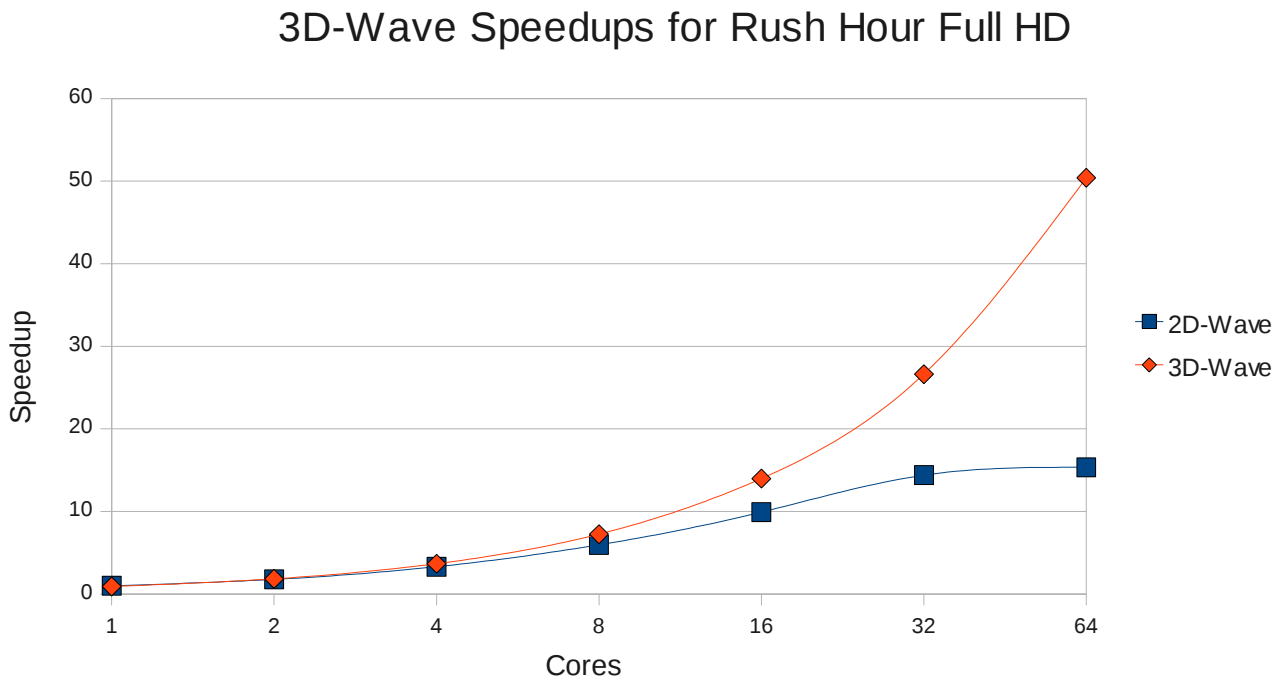
Outline

1. Motivation
2. Brief introduction to parallelization of H.264
3. Parallel scalability of H.264 decoding
   a) 2D-wave on a shared memory multiprocessor
   b) 3D-wave on an embedded multicore processor
4. Conclusions and future work

3D-Wave on an embedded multicore processor

- 3D-Wave was implemented on Trimedia TM3270
  - 4-way, VLIW, 64KB L1.
  - Private L1 D$, shared L2 D$ (not simulated)
  - Simulated embedded many-core
  - Hardware support for thread management
- Studies on bandwidth requirements, memory latency effects, L1 cache size were conducted
- Policies to reduce number of frames in flight and frame latency were developed
- Presented results are for 25 frames (1 second) of Rush Hour Full High Definition

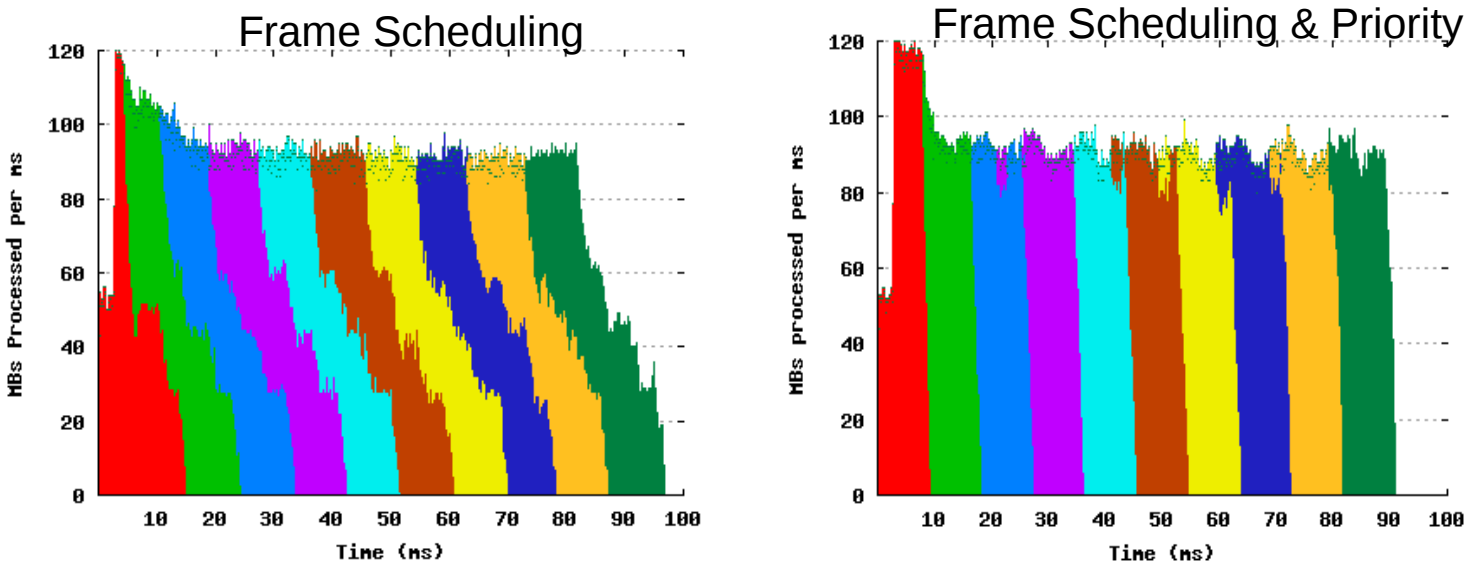
3D-Wave Scalability Results



- Efficiency of more than 80% for 64 cores
- Ramp-up and ramp-down times of short sequence decrease efficiency
- 64 cores is 16x faster than real-time for FHD

A. Azevedo et al. HiPEAC-09

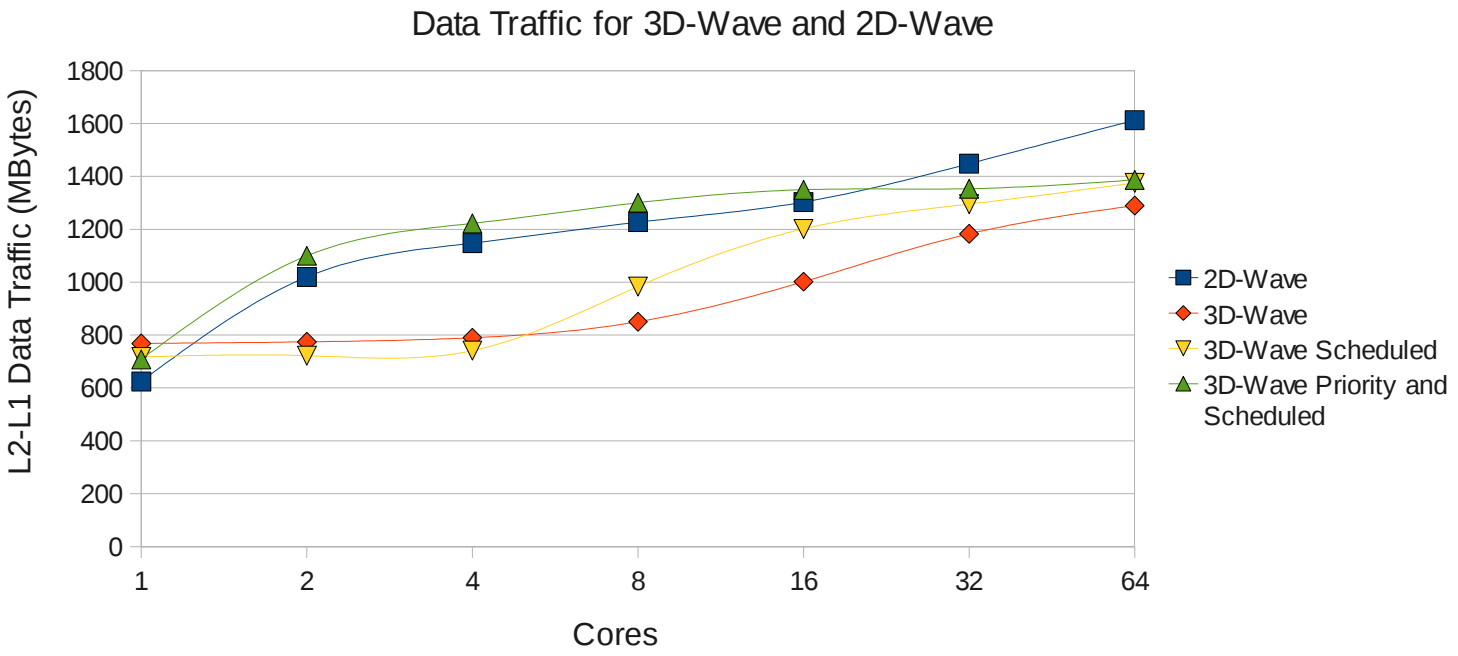
3D-Wave Frame Scheduling and Priority



FHD decoding on 16 cores

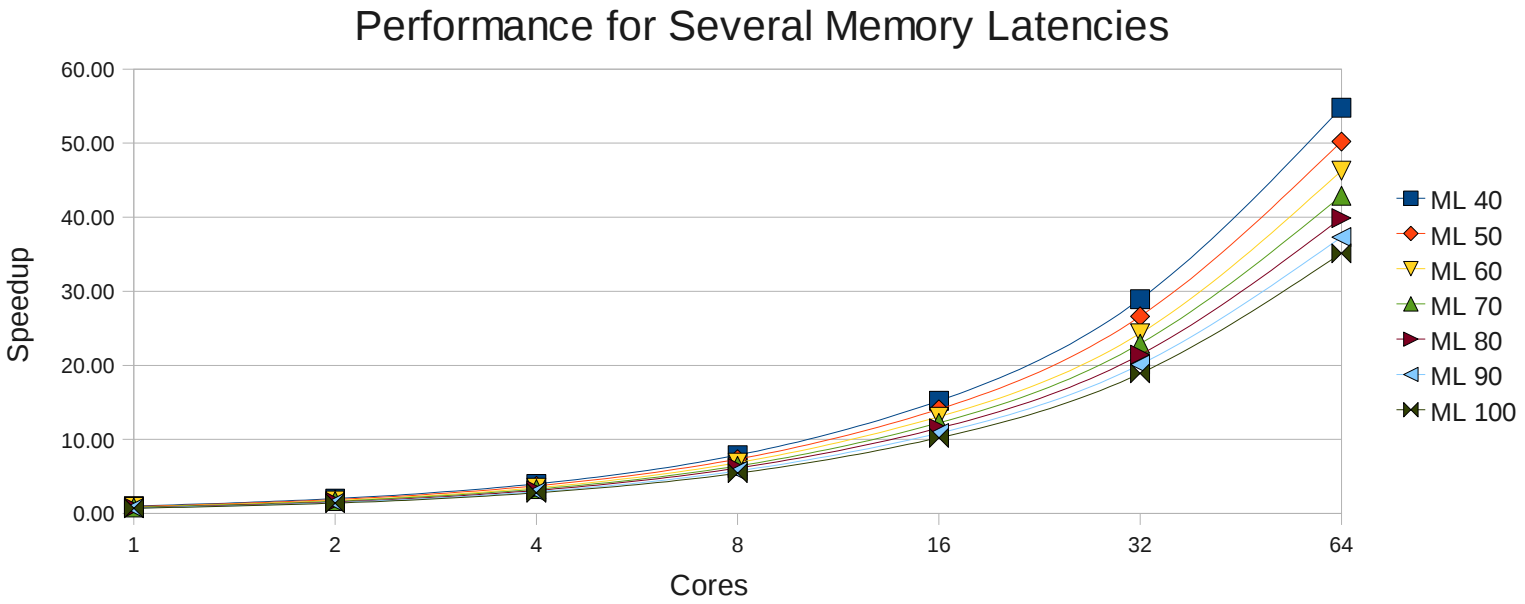
- Frame Scheduling limits the number of frames in flight
- Frame Priority reduces frame latency to the same as 2D-Wave (10ms)

Bandwidth Requirements



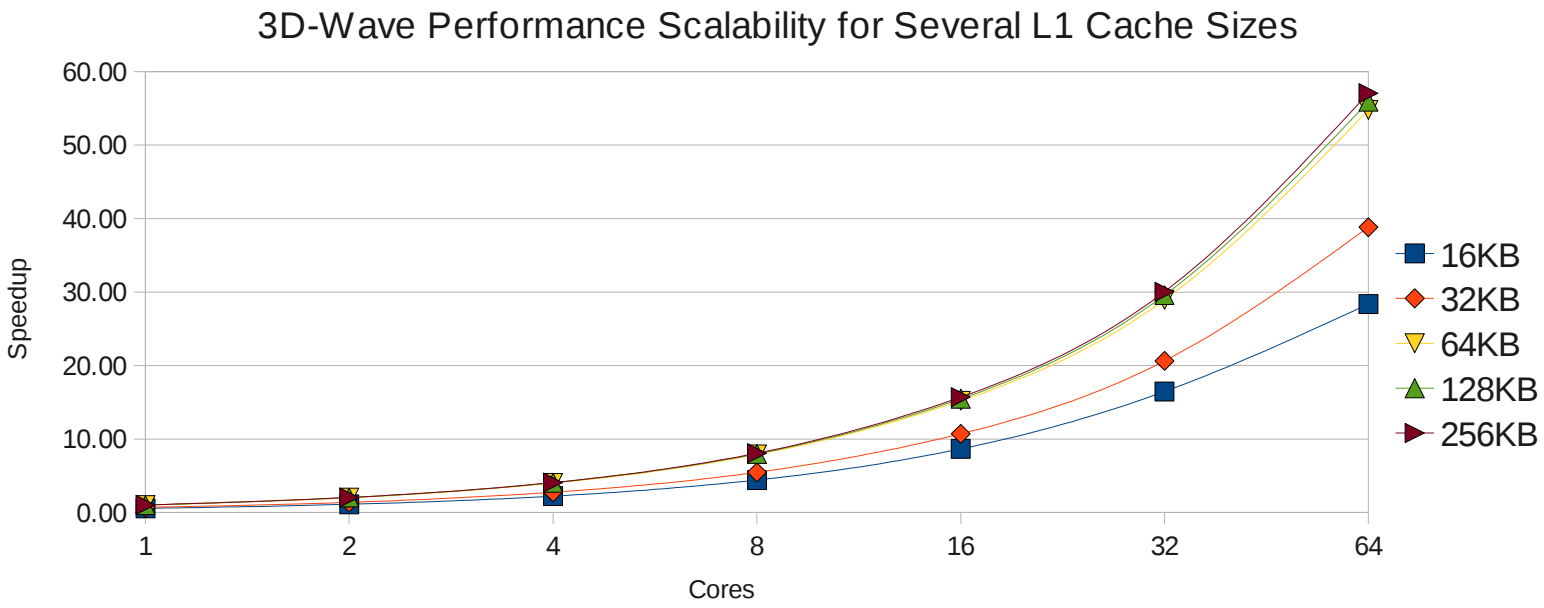
- Bandwidth required for real-time decoding is less than 2GB/s
- 3D-Wave is 20% more bandwidth efficient than 2D-Wave
- Scheduling and Priority reduce MC locality and increase bandwidth

Memory Latency (L1-L2)



- Increasing number of cores can increase latency to access data
- Speedup over single core ML 40
- Performance is highly affected by ML
- Doubling ML results in 10% scalability loss for 64 cores

L1 Cache Size



- Speedup over single core, 64KB
- 64KB L1 data cache sufficient for highest performance

Conclusions

- Parallelization strategies:
  - 3D-wave is more scalable than 2D-wave
    - 3.28X faster than 2D for 64 cores at FHD
    - In 2D-wave the number of independent MBs is variable over time
    - In 3D-wave frames are processed in parallel maintaining a high processor utilization
  - 3D-Wave is more bandwidth efficient than 2D-Wave

Conclusions

- MB parallelization is a fine grain form of TLP.
  - Thread synchronization can jeopardize the parallelization performance
    - Existing blocking synchronization APIs, like POSIX threads, incur large overhead
  - Both 2D- and 3D-wave require efficient and scalable thread synchronization primitives
    - Hardware support for scheduling and synchronization (e.g. Al-Kadi and Terechko. HiPEAC-09)

That's all

Questions, comments and suggestions are welcome:

Delft University of Technology

Backup slides

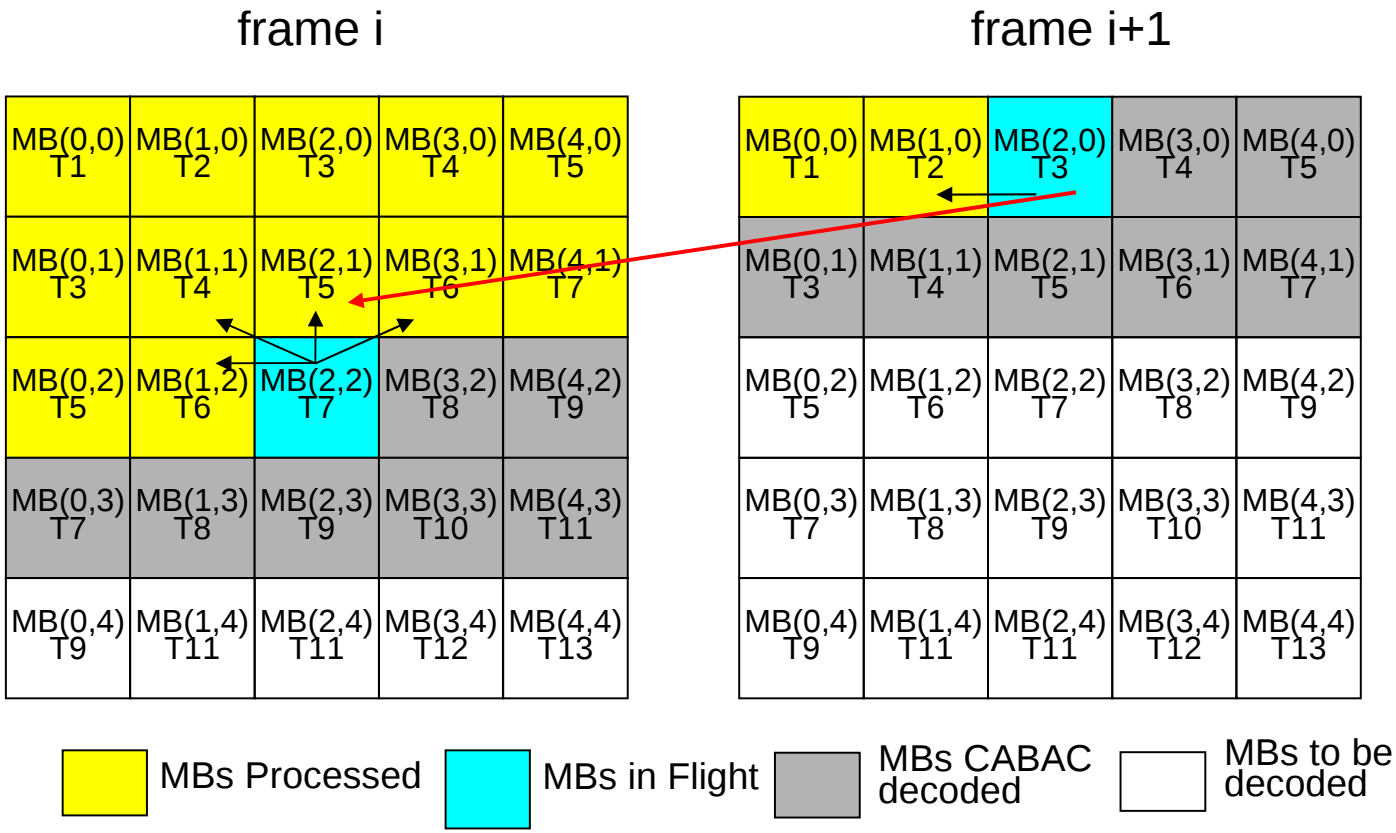
TU Delft
Delft University of Technology

H.264 MB Parallelization

- Macroblock (MB) Thread Level Parallelism
  - H.264 kernels are applied to small tails of frames called Macroblocks (MBs)
  - MBs can be processed in parallel if MB dependencies, both intra- and inter-frame, are satisfied
- MBs have intra frame dependencies
  - Motion Compensation, Intra-prediction and De-blocking filtering use data from neighbor MBs
- MBs have inter frame dependencies
  - Motion Compensation requires data from reference frames.

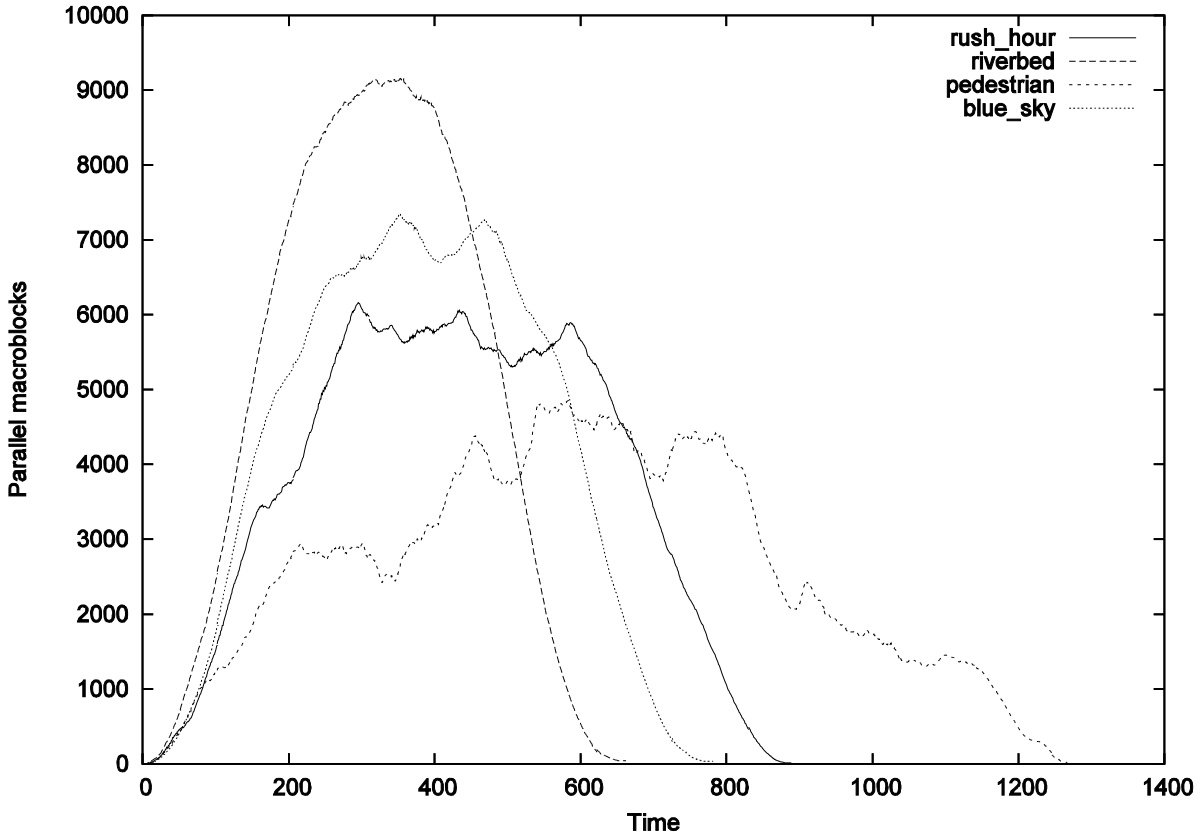
Inter-frame TLP

- It is possible to start decoding the next frame when the reference region has been decoded
- Multiple frames can be started in parallel



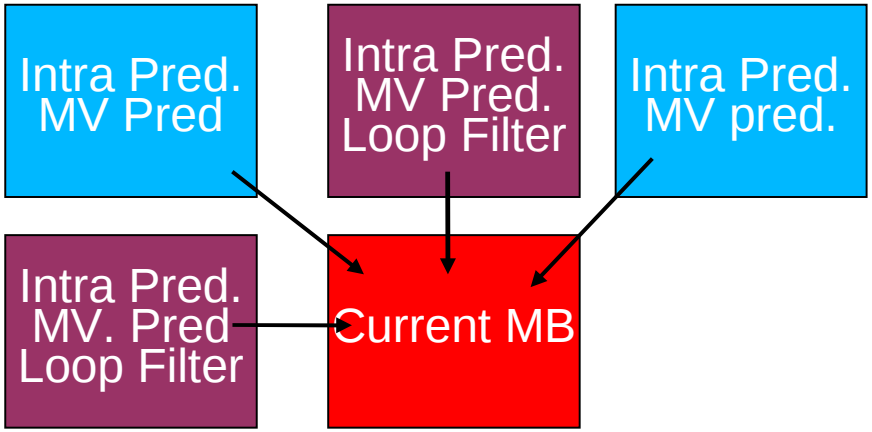
3D wave: theoretical maximum parallelism

- Maximum available parallelism ranges from 5000-9000 MBs!
- This requires >200 frames in flight.



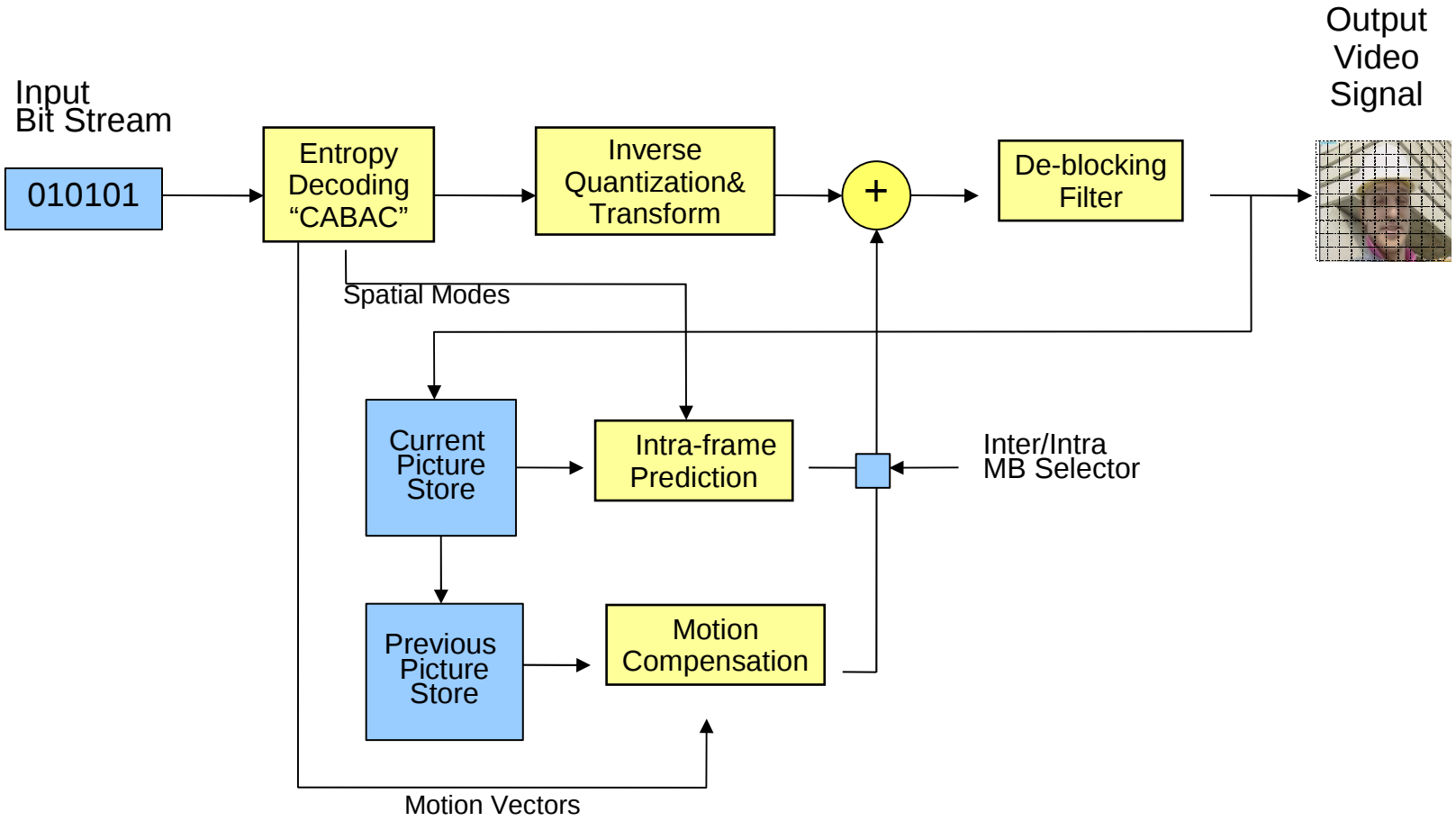
2D-wave: Intra-frame TLP

– H.264/AVC kernels have dependencies between neighbor Macroblocks



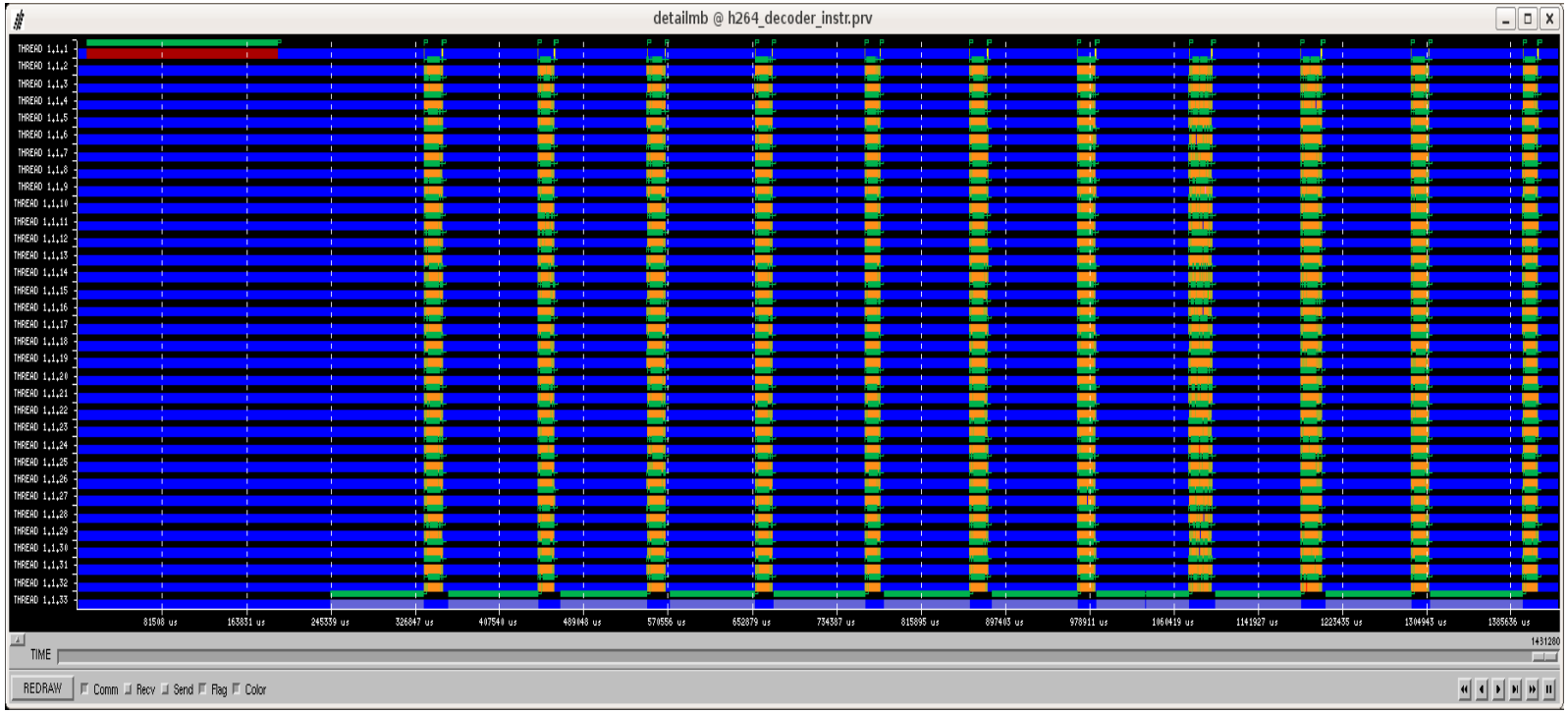
- **Intra-prediction**: pixel prediction from the current MB requires data from adjacent blocks
- **Motion vector prediction:** motion vectors can be predicted using data from adjacent macroblocks.
- **Deblocking filter:** deblocking of the current macroblock requires pixels from adjacent blocks

H.264 Decoder



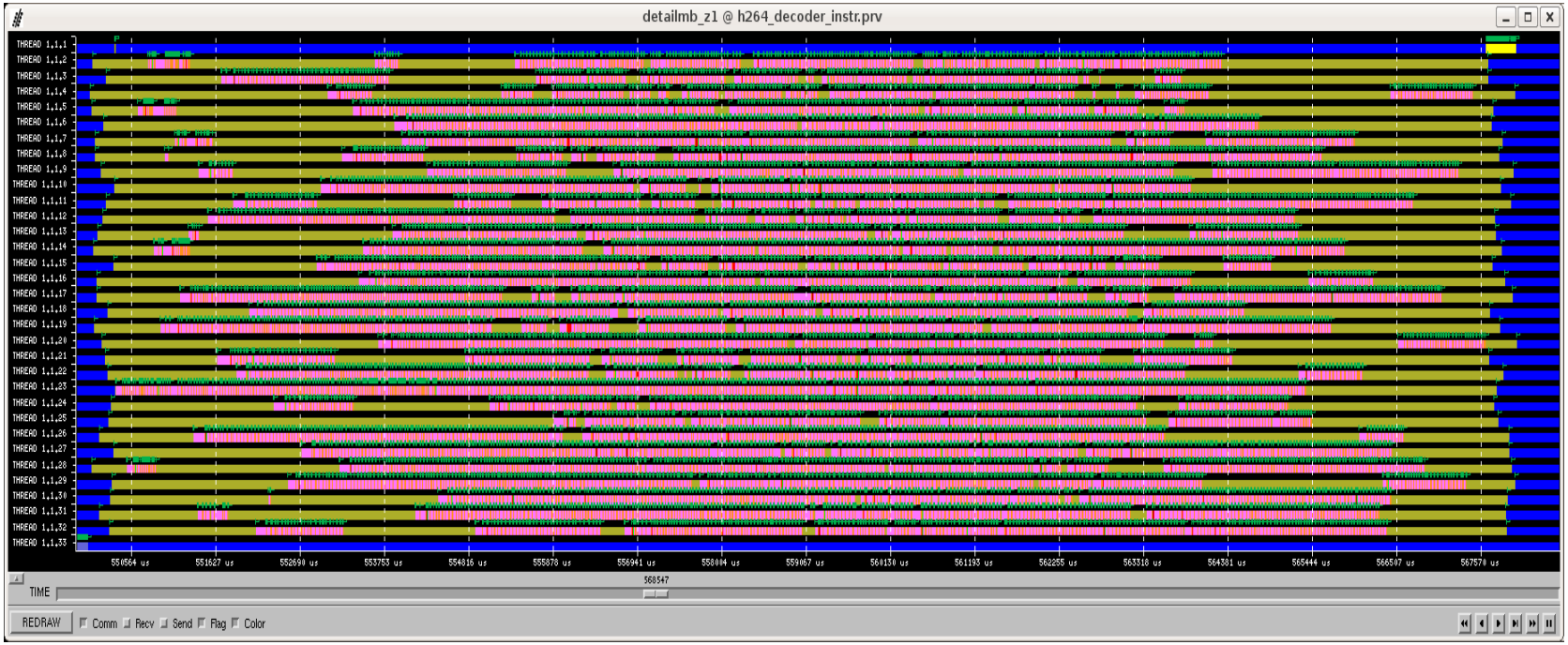
Execution on the Altix multiprocessor

Execution traces:
- 32 processors:
- complete frame sequence: 10 frames



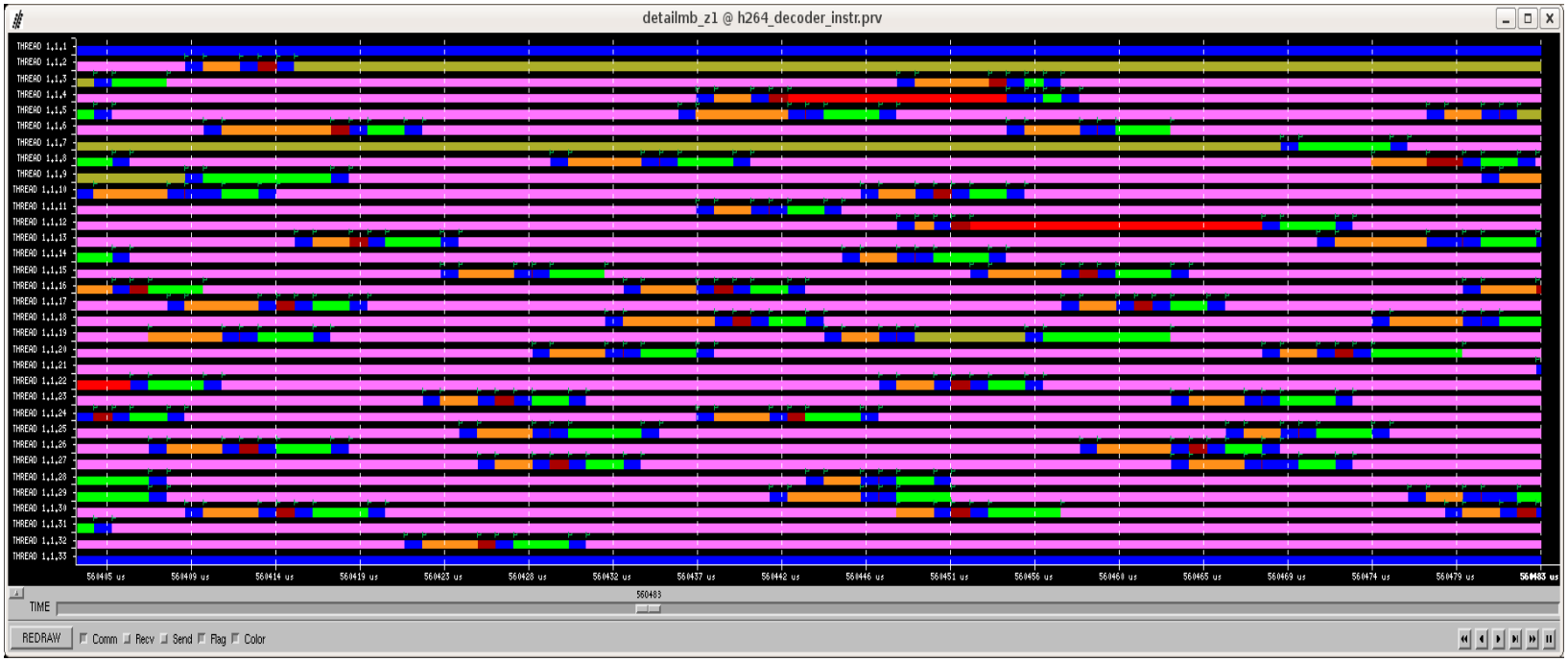
Execution on the Altix multiprocessor

Execution traces:
- 32 processors:
- 1 frame



Execution on the Altix multiprocessor

Execution traces:
•32 processors:
•parallel region inside a frame



H.264 Encoder

Transform& Quantization

Quant. Transf. coeffs

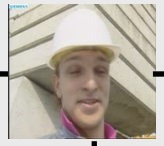
Inverse Quantization& Transform

Encoded bit-stream

Entropy Coding

De-blocking Filter

Intra-frame Prediction



Motion-Compensation

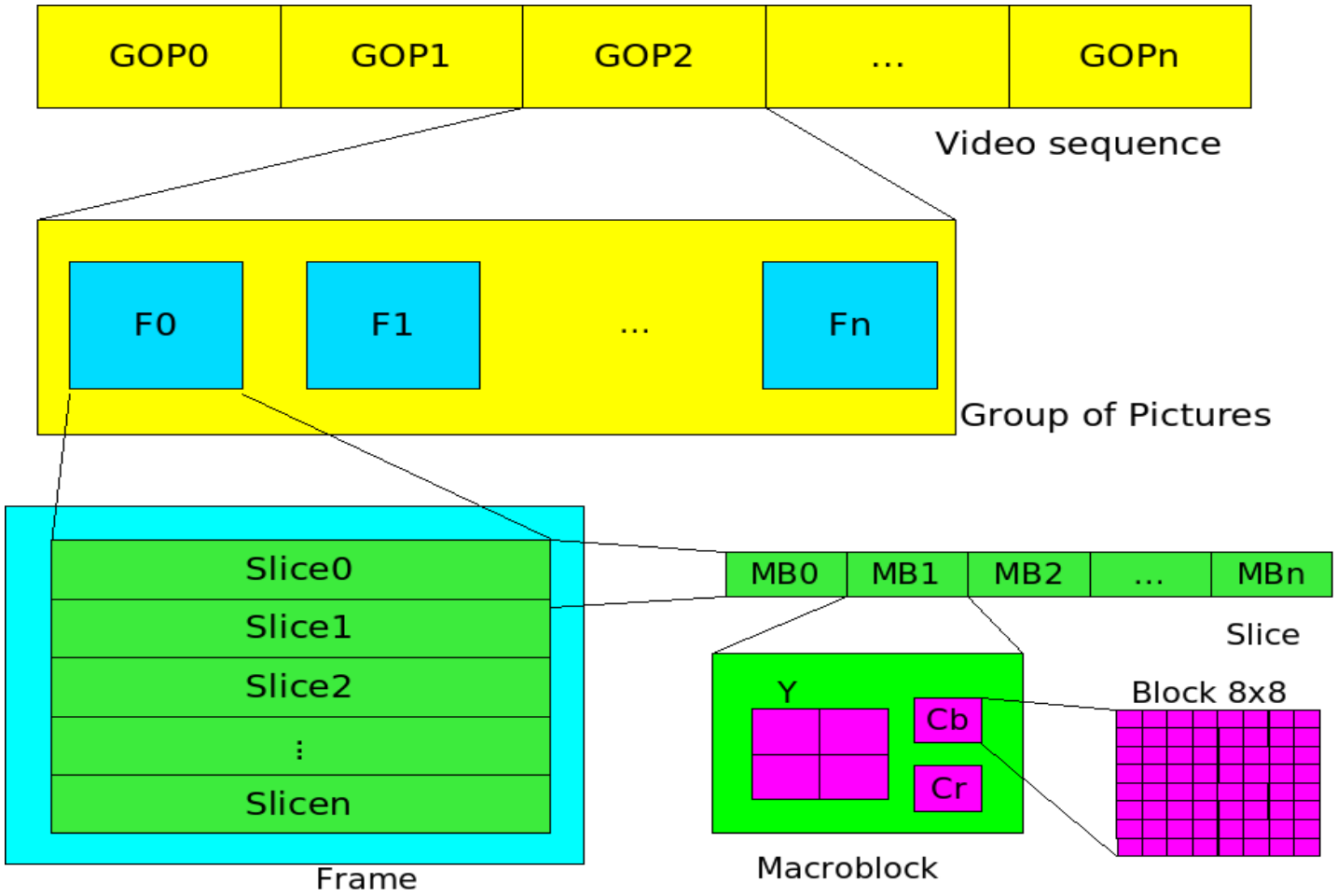
Intra/ Inter

Output Video Signal

Motion Data

Motion Estimation

Data Elements in a Compressed Video

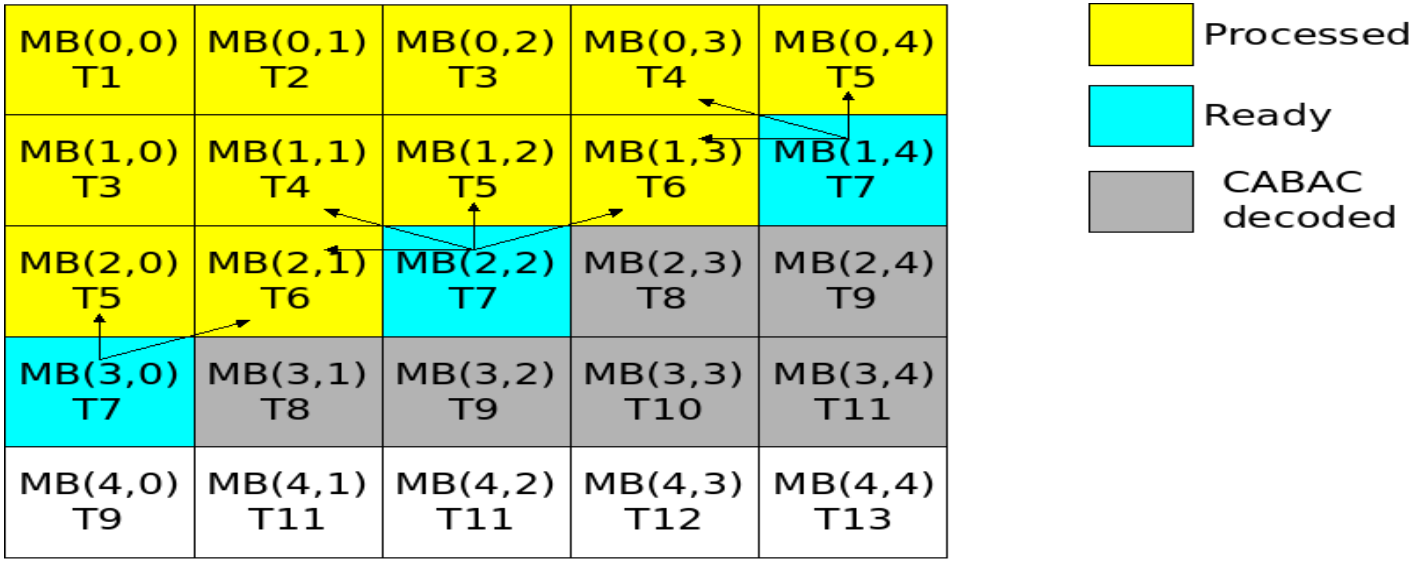


H.264 Decoder: Main Loops

```
for (all-frames-in-sequence){
   for (all-slices-in-frame){
         decode_slice_header()
         for(all-macroblocks-in-slice){
               decode_mb_cabac();
               mb_prediction();
               idct_add();
               filter_mb();
            }
      }
}
```

The CABAC Issue

- CABAC should be decoupled from macroblock decoding
  - 1 thread doing entropy decoding in sequential order
  - N-worker threads doing macroblock decoding in 2D wave-front order



Code transformation 1. CABAC decoupling

```
for(all-macroblocks-in-slice){
    decode_mb_cabac();
    copy_context_to_mb();
}

for(all-macroblocks-in-slice){
    copy_mb_to_context();
    hl_decode_mb();
}
```

Code transformation 2. 2D wave order

```
for(all-macroblocks-in-slice-in-sequential-
  order){
    decode_mb_cabac();
    copy_context_to_mb();
}

for(all-macroblocks-in-slice-in-2dwave-
  order){
    copy_mb_to_context();
    hl_decode_mb();
}
```

TU Delft
Delft University of Technology

Evaluation Platform

- Itanium 2 multiprocessor system
  - 1.6 Ghz processor nodes
  - ccNUMA memory architecture
- Software platform
  - SUSE Linux OS, kernel 2.6.16.27
  - Compiler: gcc-4.1.0
- H.264 decoder
  - Modified version of FFMPEG
  - HD inputs from HDVideoBench





Sample code of a worker thread

```
while (1) {
    wait_for_start_signal_from_master_thread();
    while (1){
        get_mb_from_taskq(fm, &mb_args);
        do {
            copy_mb_to_context(h_local, mb_args.block);
            hl_decode_mb(h_local);
            update_mb_dependencies(fm, h_local, &mb_right_ready,
                                   &mb_down_left_ready);
            submit_ready_mb(fm, h_local, &mb_right_ready,
                            &mb_down_left_ready);
        } while (mb_right_ready || mb_down_left_ready);
    }
}
```