Mapping the AVS Video Decoder on a Heterogeneous Dual-Core SIMD Processor

Nikolaos Bellas, Ioannis Katsavounidis, Maria Koziri, Dimitris Zacharis

University of Thessaly
Greece

Outline

• Introduction to AVS video standard
• Introduction to Tensilica 388VDO Diamond heterogeneous dual core architecture
• Porting the AVS decoder to the Tensilica architecture
  – Pure software optimizations
  – SIMD optimizations
  – Task level partitioning
• Experimental evaluation
• Conclusion
Proliferation of video standards

• Variety of standards for different applications
  – From Wireless, Low-rate to Broadcast High Definition quality
  – MPEG-1 (CD)
  – MPEG-2 (DVD)
  – MPEG-4 (wide range of applications)
  – H.264 (wider range of applications)

• Video compression standards historically have a major influence on computer architecture innovation
  – SIMD instruction sets (SSE, Altivec, Cell SPE)

Audio Video Standard (AVS)

• Development initiated by the Government of China in 2002
  – China’s national standard in 2005
  – Aim to reduce China’s royalty payments to foreign companies
  – IPTV drives AVS adoption

• AVS similar in structure to other video standards
  – Simpler than H.264, with very similar rate-distortion performance.
AVS Implementation

**Objective**
- Port the AVS video decoder to a dual core SIMD processor
- Provide a “reference port”, as a documented example for people wanting to port codecs to this core
- Performance target is 25 frames/sec for D1 resolution, NTSC format (720x576)
- OpenAVS is the starting code

**Main challenge**
- Enable real-time video decoding (25fps) starting from sequential OpenAVS code running at 1.73 GHz in the base Xtensa™ RISC processor
- Exploit all forms of parallelism available in the AVS decoder

*OpenAVS is an open source AVS decoder*
Tensilica 388VDO Diamond core

• Two heterogeneous processors
  – Based on the Xtensa™ RISC processor
  – Enhanced with video specific instructions
  – Around 400 such instructions
• Stream processor
  • Optimized for serial processing for bitstream parsing and Variable Length Decoding (VLD)
• Pixel processor
  • Optimized for SIMD processing for Motion Compensation, Deblocking, inverse DCT, and inverse Quantization

Tensilica 388VDO Diamond core

• Two heterogeneous processors
  – Tightly coupled Instruction and Data SRAMs in each processor to reduce memory latency
  – Multichannel DMA controller asynchronously transfers data between external DRAM and SRAMs
  – GCC-based cross-compiler, assembler, debugger, simulator, profiler
Support for SIMD operations

A, B, res: vector registers
res = xvd_add_16x12(A, B)

Vector register file extended with guard bits to enable extended precision during computation

res = xvd_add_8x24(A, B)

Code restructuring

Main loop for macroblock processing

```
for (MbIndex = 0; MbIndex < no_MBs; < MbIndex++) {
    ParseOneMacroblock; // Parsing, VLD
    McIdctRecOneMacroblock; // Inverse transform, Motion Comp
}
for (MbIndex = 0; MbIndex < no_MBs; < MbIndex++) {
    DeblockOneMacroblock; // Deblocking
}
```

- We apply loop fusion to replace the two loops with a single one
- This avoids spilling the frame pixels to external memory in the first loop and reading it back in the second
  - Memory spills degrade performance
- Additional data structures are needed to store intermediate data for deblocking filter
  - Three pixel rows and three pixel columns after Motion Comp
  - This extra information is stored in the local SRAMs
SIMD optimizations
Deblock filter

Low-pass filter across block boundaries to smooth block edges.
Horizontal and vertical filter

Deblock filter flow diagram for luma components

Only pixels p1,p0,q1,q0 affected when BS=2
Only pixels p0,q0 affected when BS=1

Horizontal 8x8 block edge

SIMD optimizations
Deblock filter Example

// Computing for Orange Boxes
// Speculatively, compute all possible results
// pr0, pr1 are 16x12 predicate vectors
// pr0[i] == 1, when (X==2, C==TRUE) for bit i
// pr1[i] == 1, when (X==2, C==FALSE) for bit i
// Compute V = P0 + Q0 + 2 only once and reuse
V = xvd_add_16x12(P0, Q0);
V = xvd_add_16x12_i(V, 2);

// (P1 + P0 + V) >> 2
T0 = xvd_add_16x12(V, P0);
T0 = xvd_add_16x12(T0, P1);
T0 = xvd_sra_16x12_i(T0, 2);

// (2*P1 + V) >> 2
T1 = xvd_add_16x12(V, P1);
T1 = xvd_add_16x12(T1, P1);
T1 = xvd_sra_16x12_i(T1, 2);

// Use predication to compute new P0, P1
// xvd_mvt is a conditional move instruction
xvd_mvt_16x12(P0, T0, pr0);
xvd_mvt_16x12(P0, T1, pr1);
xvd_mvt_16x12(P1, T1, pr0);
SIMD performance

- Global speed up due to SIMD optimizations
  - 2.26x
- Motion Compensation kernel speed up
  - 4.8x
- Deblock Filter kernel speed up
  - 3.35x

Task level parallelism

- Exploit the concurrent functionality of three cores:
  - Stream Processor
  - Pixel Processor
  - DMA controller
- Two separate instruction threads and data-set spaces
- User partitions both code and data among the two processors and orchestrates data communications
- 45% - 55% load balance between processors
Decoupled functionality

- Allow the two processors to work independently in successive macroblocks
  - Avoids excessive lockstep synchronization
- A processor can run ahead in execution
- Multi-buffering scheme necessary to keep intermediate data
  - Buffers are kept in local Data SRAMs
- Dual buffering scheme used for AVS decoder

Decoupled functionality

- Decoupling facilitates performance variability
- Different bitstreams can create corner-cases where the bottleneck shifts between the two cores
- Multi-buffering alleviates such corner cases
- Performance depends now on average load balance
Concurrent functionality

- Stream processor can proceed with the next MB(i+1) when:
  - it finishes VLD of MB(i)
  - there is enough space in the double buffer
- Overlap multiple macroblocks

Performance Evaluation

- Tensilica toolset used to compile, simulate and profile the code
  - Xtensa™ optimizing compiler (xt-xcc)
  - Cycle accurate multi-core simulator used for profiling
- Main Memory latency
  - 32 cycles for first read, 1 cycle after that
  - 64 cycles for first write, 1 cycle after that
- Benchmark
  - 4Mbps input bitstream, D1 720 x 576 output resolution
Performance Evaluation

Required 388VDO MHz frequency for 720 x 576 x 25 pix/sec (Speed Up in parenthesis)

Optimizations

Baseline | SW optimizations (VLD+Parsing) | SIMD (+MC+Intra+iT) | SIMD (+Deblock) | +Task Level Parallelism
---|---|---|---|---
1730 (1x) | 1461 (1.18x) | 1354 (1.28x) | 748 (2.31x) | 649 (2.67x) | 359 (4.8x)

The 388 VDO Diamond processor technology

TSMC 65 GP process technology

- Typical $F_{\text{max}}$: 448 MHz
- Typical Power (H.264 decode): 36 mW

Courtesy of Tensilica, Inc.
Conclusion

• We outlined the steps to extract parallelism from the AVS video decoding application
  – Data level parallelism (SIMD)
  – Task level parallelism
• and map it to Tensilica’s Diamond 388 VDO heterogeneous dual-core
• Total speed up almost 5x starting from a sequential software version
  – 1.18x from software optimizations
  – 2.26x from SIMD
  – 1.8x from task level parallelism