1 Introduction

The eco16 cores are a family of 16-bit microprocessors for embedded application. The eco16b base ISA is targeted at general purpose embedded control and computing applications. The eco16d extension ISA adds advanced 16-bit DSP features to the base ISA. The eco16i extension ISA adds 4-way SIMD (Single Instruction Multiple Data) DSP features and is specifically targeted at image processing and video applications.

This MPEG4 video decoder test has been implemented to evaluate the video performance of the eco16i ISA and the eco16il implementation. MPEG4-ASP (Advanced Simple Profile) video was chosen because it is still widely used today based on the popular XviD and DivX codecs and because of the medium complexity compared to the older MPEG1/2 codecs and the newer MPEG4-AVC codec.

The MPEG4 video decoder is entirely written in assembly. A C compiler for the eco16 processor was not yet available at the time the decoder was implemented. The performance critical routines are highly optimized and can be reused for commercial video decoder implementations.

2 Decoder details

2.1 Features

- Decodes MPEG4 elementary streams with I,P and B-VOPs
- Supports picture width up to 720 pixels (limited by local memory buffers)
- Supports picture sizes up to 1620 macroblocks (SD video, limited by local memory buffers)
- Uses 2 external memory buffers per frame: 1. luma, 2. interleaved chroma
- Implements a 2-stage macroblock processing pipeline to compensate for the latency of DMA based prediction reads from external memory

A number of MPEG4-ASP features are not supported.

- Global motion
- Quarter PEL motion compensation
- Non rectangular shapes of the video object plane
- Interlaced video
- Overlapped block motion compensation
- Sprites
- Non 8-bit per pixel resolution
- Reduced resolution VOPs
- Scalability

These features are rarely used, most XviD and DivX encoded video streams can be decoded without them. The decoder has been tested with a number of video clips from different sources (encoders).

2.2 Concept

This software video decoder is optimized for highest performance efficiency (low processor MHz requirement). It is targeted at systems that have local memory with zero wait states access for macroblock decoding. Frame buffers are mapped into the main memory pool which can be external DRAM or on-chip SRAM/DRAM with some access latency. The required source data (e.g. temporal predictions) are transferred to the local memory by DMA operations. Decoded macroblock data are transferred to main memory frame buffers by DMA operations. Because the actual decoding accesses only the local zero wait state memory there are no performance penalties caused by main memory access latency.

2.3 Macroblock pipeline

DMA operations to read predictions from external memory frame buffers take some time to complete. A software decoder that processes one macroblock at a time would have to wait for these DMA operations before performing the remaining steps of macroblock decoding (prediction filtering, forward/backward interpolation, residual add).

To avoid waiting for prediction reads this decoder implements a 2-stage macroblock processing pipeline for P-VOPs and B-VOPs. There are two macroblock data structures with type, motion vectors, prediction buffers, and block coefficient buffers. The decoder switches between the two structures with every macroblock. For each new macroblock the decoder first reads the header, calculates motion vectors and sets up prediction read DMA operations. Next the decoder reads the block coefficients and stores them in a buffer. Then it goes back to the preceding macroblock and waits until all prediction read DMA operations for

that macroblock have completed. In most cases there is no waiting and the prediction reads have completed already. The decoder then performs the remaining processing steps on the preceding macroblock (prediction filtering, prediction interpolation for bidirectional macroblocks, IDCT and residual add) and sets up a DMA operation to transfer the decoded macroblock to an external memory frame buffer. I-VOP decoding processes one macroblock at a time because there are no prediction reads from external memory.

2.4 Co-located motion vectors

MPEG4-ASP with I-VOPs and P-VOPs requires some buffer space for DC/AC and motion vector predictors. Data for a few more than one row of macroblocks need to be buffered which is a reasonably small space to be mapped into the local memory for fast access. However to support B-VOPs the motion vectors of all P-VOP macroblocks need to be buffered to be used as predictors in subsequent B-VOPs. For e.g. SD (Standard Definition) video with a maximum of 1620 macroblocks per frame the required buffer space would be ~26kBytes (1620 * 16 Bytes) which is too costly to be mapped into the local memory. The decoder implements a compromise between local memory size and fast access.

The main memory has a buffer for all motion vectors of one frame (four motion vectors with 2 x 16-bit each). A single cache buffer for one row of macroblocks is mapped into the local memory. For SD video the required buffer size is 45 * 16 bytes = 720 bytes. When a P-VOP is decoded the motion vectors are stored in the local buffer. At the end of each macroblock row the decoder sets up a DMA operation that transfers the local buffer content to the main memory buffer. When a B-VOP is decoded a DMA operation is set up at the beginning of each macroblock row to transfer the motion vectors from the corresponding row from the main memory to the local memory buffer. During B-VOP macroblock decoding collocated motion vectors from the preceding P-VOP are read from the local memory buffer.

2.5 Memory footprint

The table below shows the total memory space requirements of the MPEG4-ASP video decoder.

ISA	Text (instruction code)	Variable data	Constant data
eco16i	12482 Bytes	10kBytes incl. stack	3000 Bytes

2.6 Performance Enhancements

The decoder implements a number of performance improvement features. The most important ones are listed below:

- Macroblock processing using local memory data only, main memory access latencies are hidden.
- Special IDCT routines are used for blocks with
  - only the DC coefficient non-zero
  - non-zero coefficients only in row 0 and
  - non-zero coefficients only in column 0
- For inter macroblocks IDCT and residual add are done in one step

3 Test Environment

To enable fair comparison with competing solutions the environment for performance measurements must be described especially any acceleration hardware but also any software functions that count into the measured numbers and are not necessarily required for a pure decoder software.

The MPEG4 decoder test environment has three hardware support functions:

- A bit string reader peripheral that allows the processor to read non-aligned bit strings from 1 to 16 bits length from the compressed video stream.
- A DMA engine that is used to transfer decoded macroblocks and collocated motion vectors from the processor's local RAM to the frame buffer memory. The input channel is used to read back collocated motion vectors from external memory
- A prediction reader (special DMA engine) to transfer prediction sample arrays from external memory frame buffers to local memory buffers. This engine is aware of the picture size and correctly handles out of frame references (edge pixel replication).

Besides the macroblock processing which takes most of the processor cycles the following software functions are included in the performance numbers:

- Parsing of the higher layers (video object layer, VOP, …)
- Assignment of frame buffers
- Setup of DMA operations
- Interrupt service routine for the prediction reader engine

- Test bench functions for the display of decoded macroblocks

4 Test Results

Performance has been measured on a number of video clips from various versions of the XviD and DivX encoders. The clips have different bit-rates, picture sizes and picture type sequences. The tests have been done in an environment with zero wait state instruction and local data memories. The latencies of DMA reads (collocated motion vectors) and prediction reads from the main memory are calculated using a realistic minimum cycle count plus a random number adder.
The table below shows the performance measurement results. For each stream performance numbers of the eco16i and the eco16il are given.
The first columns of the table contain video clip properties:

- Picture size in pixels horizontal (suffix h) and vertical (suffix v)
- Clip length, total number of VOPs (frames) in the first row and I/P/B VOPs in the second row
- Frame rate in frames per second
- Average bit rate in Kbits/s

Performance is expressed by two MHz numbers. The first row number is the average required MHz over the entire clip (suffix a). The second row number is the average MHz requirement over the worst frame of the clip (suffix w).

clip	size	VOPs	fps	Kbits/s	eco16i	eco16il
1	480h 224v	171 2/86/83	25	758	21.8a 37.6w	25.2a 43.0w
2	480h 224v	158 3/64/91	25	716	20.5a 47.7w	23.2a 53.4w
3	480h 272v	611 8/207/396	25	625	17.4a 77.8w	23.1a 103.8w
4	544h 416v	134 3/66/65	23,98	2278	45.5a 112.4w	51.8a 125.7w
5	720h 304v	48 6/38/4	25	4075	67.3a 131.6w	76.1a 146.2w
6	640h 272v	142 3/139/0	24	1013	25.4a 75.3w	29.3a 84.8w
7	640h 272v	76 15/43/18	25	3466	58.4a 93.9w	64.8a 104.8w
8	352h 256v	286 3/109/174	25	968	20.9a 61.8w	23.5a 67.7w
9	640h 256v	114 4/57/53	25	2467	44.1a 129.0w	49.6a 141.8w
10	704h 576v	57 1/56/0	25	3053	76.4a 147.0w	89.5a 168.9w

The test results show that the eco16i ISA and the eco16il implementation are well suited for MPEG4-ASP video decoding up to VGA (640x480) and SD (720x480) picture size with up to 30 frames/s. Typical average bit rates for streams with these parameters are ~2 Mbits/s. Some of the test clips have significantly higher bit rates to show the processor performance under difficult conditions.
It can be noted that for some test clips the processor load for the worst case frame is significantly higher than the average load. Especially for clip 3 the worst frame load is more than 3X higher than the average. This is because most of the test clips are difficult to decode clips. They have been encoded with 'constant high quality' (quantization) rather than constant bit rate to challenge the decoder and get worst case numbers. For real life software video applications, e.g. PMPs typically more decoder friendly video content is used to avoid overloading the processor.

4.1 Reasons for the relatively low IPC of the eco16il

It can be noted that the eco16il implementation takes some more cycles than the eco16i reference model. The average IPC (Instruction Per Cycle) for the eco16il is in the range of 0.75 – 0.9. This means that up to ~25% of the cycles are lost through processor stalls or pipeline bubbles. The following paragraphs give the main reasons for this relatively low IPC and describe how the MHz requirements can be decreased by adding hardware resources in systems where lowest MHz and power are most important.

4.1.1 Instruction Latency

With its large number of 64 x 16-bit registers that can be accessed as vectors and scalars the complexity and depth of the data dependency logic (to detect stall conditions) is significantly higher as for most other processor of the eco families. To keep the logic depth at a reasonable level (enable high clock rates) only a minimum number of operand forwarding cases is implemented. As a consequence some instruction groups have increased latency compared to most other eco processor implementations. E.g. the latency from load to computation/multiply/store instructions is 4 instead of 3 cycles. The latency from computation/multiply to store instructions is 2 instead of 1 cycle. The latency from computation to multiply is 2 instead of 1 cycle.

Especially in the symbol decode routines where load latency can't be compensated by instruction reordering the load latency creates a lot of lost cycles.

The situation can be improved with additional hardware acceleration for symbol decode, e.g. a module that autonomously decodes single symbols and symbol sequences and writes the into the processor data memory through DMA.

4.1.2 Loops with 32-bit opcode instructions only

Many of the inner loops of DSP processing consist of only extension instructions with 32-bit opcodes. Fetching of instructions is not faster than consumption in the execution units and the decoupling buffer between instruction fetch and execution cannot be filled. As a consequence branch instructions that jump back to the beginning of the loop take two cycles effective. With a filled decoupling buffer loop branches take zero cycle effective and help to compensate for cycle that are lost elsewhere.

This cycle loss could be avoided by implementing a single entry loop cache in the processor's instruction fetch unit. Such a cache enables zero cycle loop back branches from the second pass through the loop. Only the first branch back takes two cycles.

4.1.3 Blocking of the ALU

The decoding of P-VOPs and B-VOPs includes a step where residuals (IDCT result) are added to the prediction samples. For best performance the decoder does this in one step with the IDCT computation. The addition of the prediction samples is done using ADDVI instructions with one memory source operand. These instructions use the ALU three cycles after they have been issued (assuming no bus wait states). Subsequent instructions with no memory source operands that use the ALU are stalled as long is the ALU is block by instruction that have been issued previously.

A possible workaround is to send blocked instructions through the memory pipeline (as if they had memory source operands). Lost cycles can be saved only if at some point in the instruction sequence instructions that do not use the ALU follow, e.g. store instructions to store the computed results. The described workaround required additional control logic and data path registers in the processor.