Efficient Implementation of Video Postfiltering on the TI DSC25 Platform

Murali Babu, Amitava Deb, Sankaranarayanan Parameswaran, Dileep Tamia & Sriram Sethuraman

Ittiam Systems Pvt. Ltd.

Presented by: Sriram Sethuraman
TI DSP Conference, 5 December 2002, Bangalore
Outline of Presentation

→ Video Postfiltering for artifact reduction
  - Coding artifacts (blocking/ringing)
  - State of the art algorithms
    • Deblocking
    • Deringing
  - Complexity analysis

→ DSC25 considerations
  - Ittiam’s MuVPlayer™ on DSC25

→ Proposed Postfiltering algorithm
  - Improving data access patterns
  - Reducing conditional execution
  - Multithreading on DSP, iMX, and DMA

→ Results / Demo
Video Coding Artifacts

- Block-based coding (8x8 blocks in MPEG-4)
  - Continuity not maintained at block boundaries
- Quantization
  - At low bit-rates, high frequency coefficients get coarsely quantized
  - Spatially adaptive quantization (adjacent blocks have different QPs)
- Motion compensation
  - Compensates from a reference with coding artifacts
- Resulting in...
Blocking artifacts
Ringing Artifacts
State of the Art Postfiltering

Postfiltering
- Outside the scope of a standards based codec
- Essential at low bit-rates

Spatial domain (more common)

DCT domain

Filtering is adapted based on coding parameters (e.g. quantizer)

Extent of filtering is adapted to spatial details
- To preserve edges/details

Independent filtering in luma and chroma

Good Baseline from the quality perspective
- MPEG-4 VM algorithms
**Spatially adaptive modes**

- **DC offset mode**
  - Smooth except for a discontinuity due to blockiness

- **Non DC offset mode**
  - Has local details or edges
Steps

- Detection of the mode
  - DC offset: 6 out of 9 differences are below $T_{pd}$
  - Else, non DC offset mode

- Correction for the mode
  - DC offset: If $|\text{MAX}(v_i) - \text{MIN}(v_i)| < 2*QP$,
    - Apply $[1 \ 1 \ 2 \ 2 \ 4 \ 2 \ 2 \ 1 \ 1]/16$ filter with boundary padding
  - Non DC offset:
    - Apply $[2 \ -5 \ 5 \ -2]/8$ kernel to $S_0$, $S_1$, $S_2$ to get $a_{30}$, $a_{31}$, $a_{32}$
    - Let $a_{3,0}' = \text{SIGN}(a_{3,0}) \cdot \text{MIN}(|a_{3,0}|, |a_{3,1}|, |a_{3,2}|)$
    - If $|a_{30}| < QP$,
      - $d = (v_4 - v_5)/2$ [clipped by $5 \cdot (a_{3,0}' - a_{3,0})/8$]
      - $v_4 = v_4 - d$ and $v_5 = v_5 + d$
MPEG-4 VM Deringing

Spatially Adaptive Index Acquisition:
- \( \text{Thr} = (\text{MAX} + \text{MIN} + 1)/2 \)
- \( \text{Index} = 1 \) if \( P_{ij} \geq \text{Thr} \); Else 0

Correction
- Apply a 3x3 LPF if all indices in that 3x3 window are all ones or all zeros; clip the difference by QP/2
TI TMS320DSC25

Reproduced from TI’s Reference Manual, © Texas Instruments
TI TMS320DSC25

→ ARM7
  - 40 MHz, 32-bit RISC, 32 kB RAM

→ C54x
  - 94 MHz, 16-bit DSP, 32 kW memory
  - Access to SDRAM only thro’ image buffers
  - 2 cycle access to image buffers

→ Image buffers (16kW)
  - BUF A and BUF B (shared between iMX and DMA)

→ Imaging Extension (iMX) coprocessor
  - 4 MAC units (parallel reads/writes from image buffer)
  - TI provided APIs for array op.s, transforms, etc.
  - Support for saturation
  - Cannot do conditional execution

→ VLD and Quant/DeQuant coprocessors
Ittiam’s **MuVPlayer™** on DSC25

- Implements MPEG-4 Video Decoding
  - Simple Profile compliant
  - Multithreaded to maximally utilize DSP, iMX, VLD/Dequant units, and DMA
  - To use the motion compensation iMX APIs, the reconstructed frame is kept in 4:2:2 (YCbYCr) interleaved format
- Optionally implements MP3 audio decoding
  - Optimized to use DSP and iMX effectively
- ARM7 is used for RTOS, networking, and A/V sync
- Goal is to add video postfiltering to this player
  - MOPS required for CIF@30fps (ignoring branching)
    - DC offset: ~200 MOPS
    - Non DC offset: ~120 MOPS
  - Processing on DSP alone is not possible; need to use iMX
VM Deblocking on DSC25

- Data Access Issues

- Frame Data
- Fully processed MB
- Partially processed MB
- Load and unpack 3 MBs
- Partially processed MB
- Store back horizontally
- Partially processed MB

SDRAM

- Processed MB write back
- Store back vertically
- Partially processed MB

iMX Buffer1

iMX Buffer2
VM Deblocking on DSC25

- Adaptive Processing Issues

→ DC/NonDC decision and corresponding corrections are independently performed for every row/column of a block
  - iMX has to compute one decision for 8 pixels
  - DSP has to schedule corrections for every 8 pixels
    • Setup overheads outweigh any iMX gains

→ DC mode
  - MAX and MIN cannot be done using iMX efficiently
  - iMX cannot pad with v0/v1 and v8/v9 for 9-tap filtering

→ Non DC mode
  - 4-point inner products are needed; inefficient on iMX
  - Touches only 2 pixels (not good enough smoothing)

→ 2-3 conditions per row/column
  • Branching penalty on DSP is 5 cycles
  • This alone would be around 12-18 MCycles/s

→ Frequent interaction between iMX and DSP makes multithreaded scheduling hard
## Proposed Approach
### - Solution to Data Access Issues

<table>
<thead>
<tr>
<th>Issue</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal and vertical deblocking happen on different set of pixels</td>
<td>Perform both on the same set of pixels; exclude v0 and v9. For each MB in frame, leads to:</td>
</tr>
<tr>
<td></td>
<td>• 1 MB load vs. 2MB loads</td>
</tr>
<tr>
<td></td>
<td>• 1 MB store vs. 2 MB stores and a copy within image buffer</td>
</tr>
</tbody>
</table>
## Proposed Approach

- **Solutions to Adaptive Processing Issues**

<table>
<thead>
<tr>
<th>Issue</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive decision per row/column</td>
<td>Combined decision for each block edge</td>
</tr>
<tr>
<td>Adaptive correction per row/column</td>
<td>Combined correction for each block edge</td>
</tr>
<tr>
<td>DC offset mode issues</td>
<td>Simplified DC offset smoothing</td>
</tr>
<tr>
<td>Non-DC offset mode issues</td>
<td>Adaptively correct up to 6 pels</td>
</tr>
<tr>
<td></td>
<td>Extent of filter computed at the block level</td>
</tr>
<tr>
<td>Conditional scheduling on iMX for each row/column</td>
<td>Need to schedule only one per block edge</td>
</tr>
<tr>
<td>Multithreading is hard</td>
<td>Reduced DSP/iMX interaction</td>
</tr>
</tbody>
</table>
Scheduling & Multithreading

DMA MB

Unpack & Detect Block Mode (iMX)

Set up Corrections (DSP)

Apply Corrections & Pack (iMX)

DMA Processed MB

→ Need to strike a balance between DSP and iMX processing times

→ iMX operations are sensitive to the buffers in which the source and destination data are kept

→ Using BUF-A and BUF-B, efficiently pipeline and hide the DSP and DMA operations by performing them in parallel with iMX
Deblocking - Results
Deringing

➔ Since deringing need not be performed in flat regions, we can skip deringing for DC offset mode blocks

➔ Data from DC mode detection process is used to perform edge detection and index acquisition
  – instead of computing 2D MAX and MIN and then thresholding as in VM

➔ iMX can be used to efficiently perform 3x3 filtering

➔ iMX can be used to efficiently sum indices on 3x3 area
Deringing - Results
Conclusions

→ Proposed approach
  - results in visual quality comparable to VM’s
    • PSNRs were also comparable
  - Considerably simplifies porting to DSC25
  - effectively utilizes DSP, iMX, and DMA resources
  - enables postfiltering to be added to MuVPlayer™
END
VM Debloking - Results