MPEG-2 to H.264 Transcoder on DM64x

Sriram Sethuraman
Senior Member of Technical Staff
Ittiam Systems (Pvt.) Ltd.
Sriram.sethuraman@ittiam.com

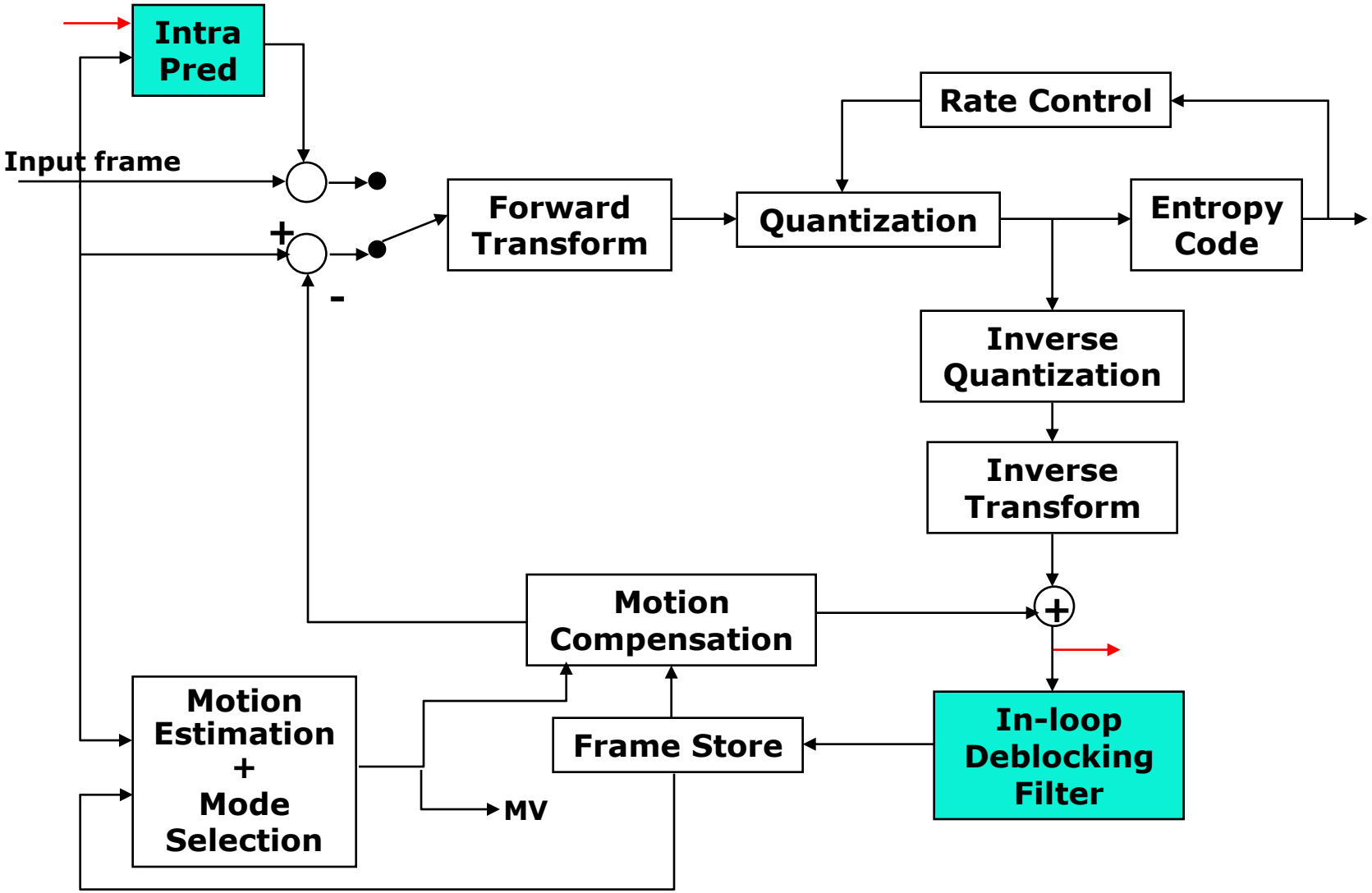
Outline of Presentation

- Overview of H.264
- MPEG-2 to H.264 Transcoding
  - Application Scenarios
- Complexity and Control flow Challenges
- Overview of DM64x Architecture
- Design Methodology
  - Pipeline design considerations
  - Exploiting parallelism
  - EDMA specific considerations
- Transcoder variants
- Conclusions

H.264 - Introduction

- Joint ITU-T and ISO/IEC standard
  - ISO/IEC 14496-10 (aka MPEG-4 Part 10)
  - Compression target: twice as much
- Status
  - Now an official international standard
  - Corrigendum being considered to fix a few errors
  - Via licensing and MPEG LA have formed patent pools
- Addresses diverse applications
  - Video telephony
  - Streaming
  - Storage (contender for the new DVD format)
  - Digital Cinema

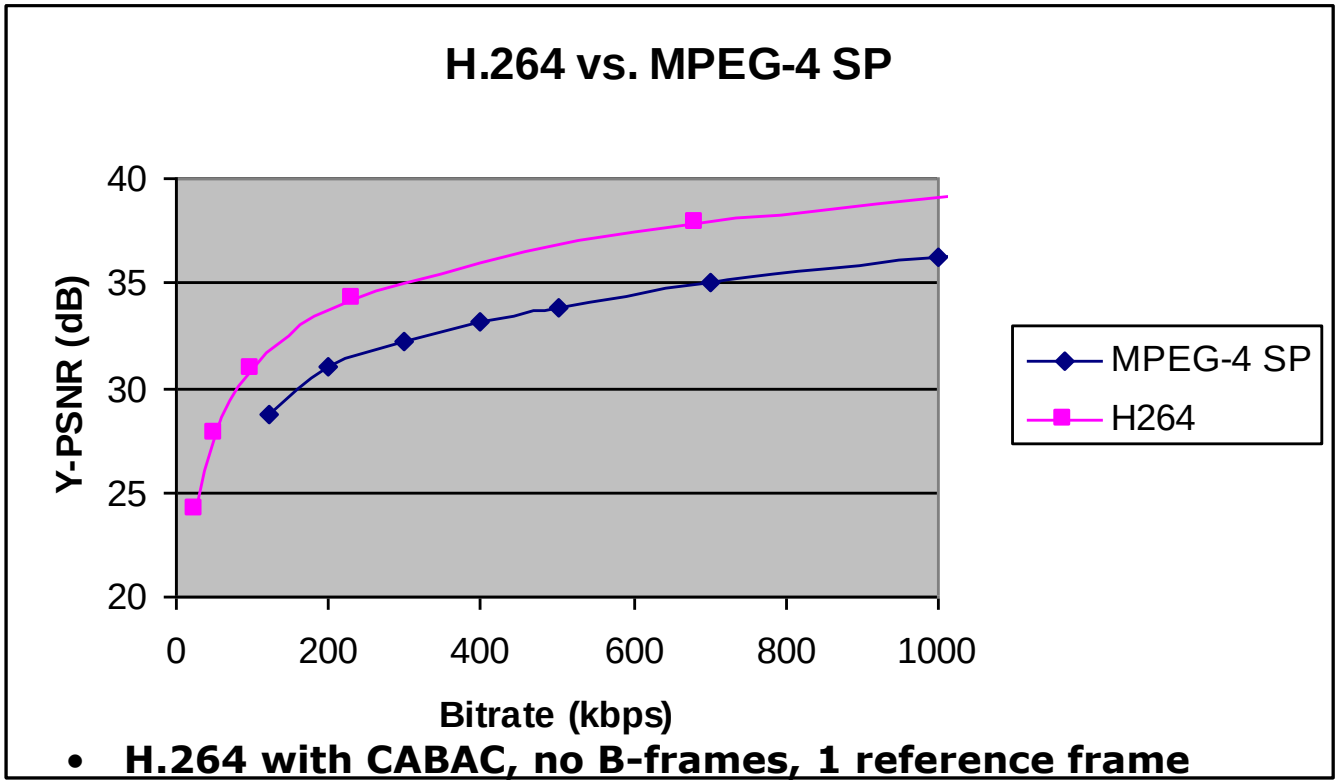
H.264 Encoder



H.264 - Features

- Improved intra prediction
  - Compression efficiency on par with JPEG-2000
- Enhanced temporal prediction
  - Improved quarter pixel motion compensation
  - Variable block size motion compensation
  - Multiple reference frames (beyond 2)
  - Weighted prediction
- Efficient entropy coding
  - Context based VLC (CAVLC)
  - Context based binary arithmetic coding (CABAC)
- Interlaced support
  - Field pictures, MB level adaptive field/frame
- Integer transform
  - No mismatch between encoder and decoder
- Low complexity, wider range quantization
- Superior in-loop deblocking filter

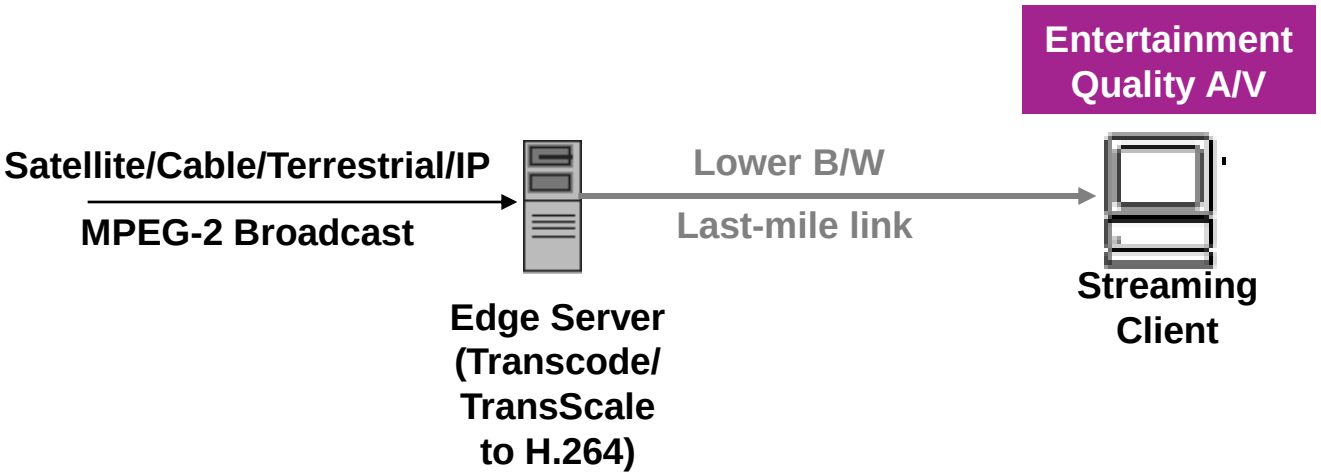
H.264 Advantage



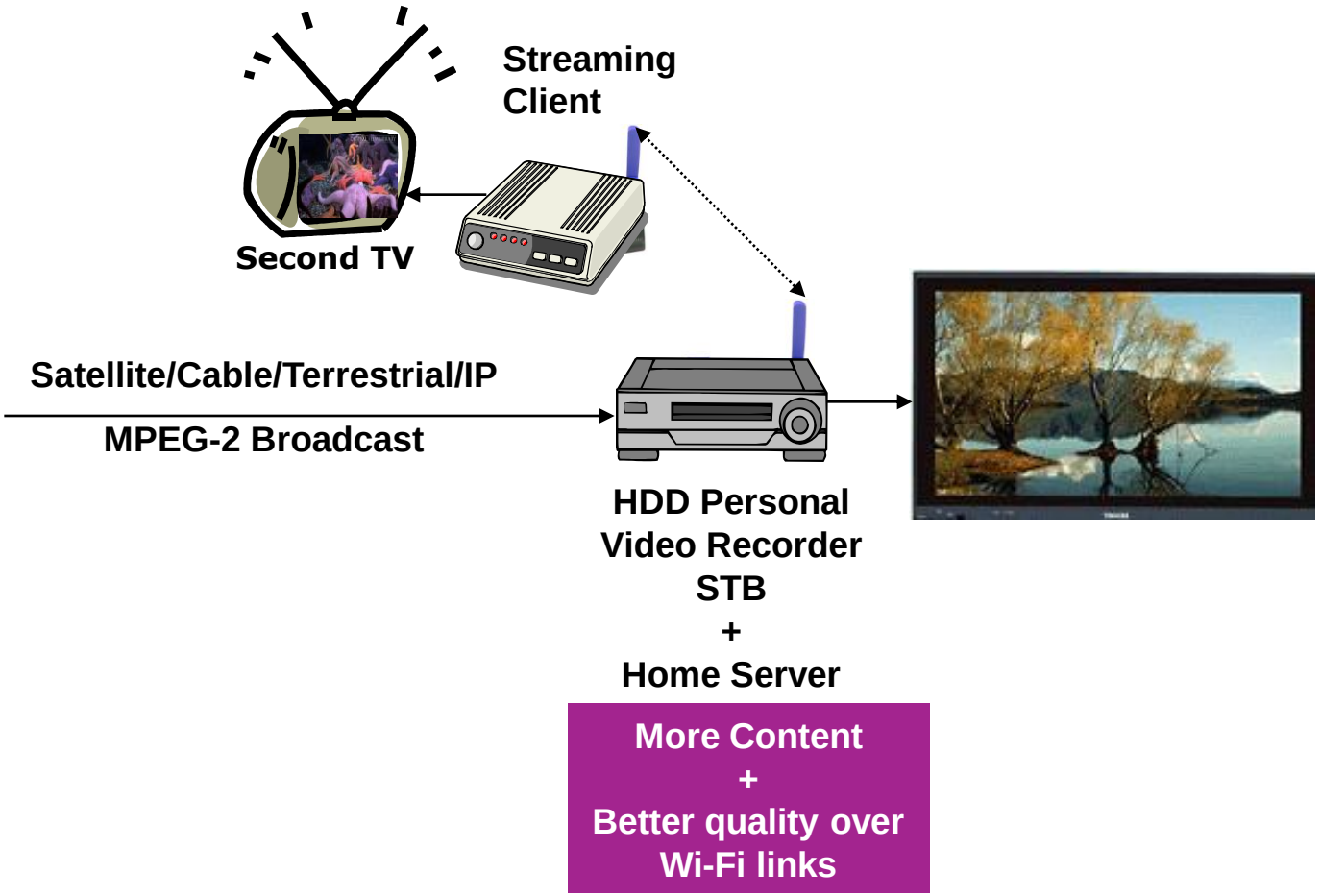
- **H.264 with CABAC, no B-frames, 1 reference frame**

- **~50% bit-rate reduction compared to MPEG-4**
- **~67% bit-rate reduction compared to MPEG-2**

Application Scenario 1

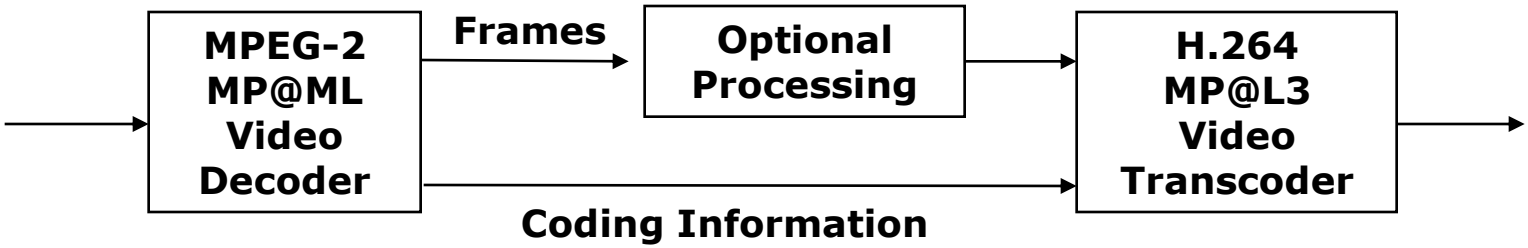


Application Scenario 2



MPEG-2 to H.264 Transcoder

- Transcoder design options
  - Transform domain
  - Minimal drift
  - Drift free
- Minimal overlap in texture coding between MPEG-2 & H.264
- Low complexity transcoder
  - Re-uses information obtained from the MPEG-2 stream
    - Motion, modes, residual energy, bit allocation, …
  - Drift-free as re-encoding is not done in the decoding loop



Compression Efficiency Gain

- **Transcoded quality**
  - can never be the same as the MPEG-2 stream quality
    - unless losslessly coded
  - Details in video cannot be distinguished from artifacts
- **Similar visual quality is possible with coding tools such as**
  - Better Intra prediction
    - In broadcast, I-frames occur every half a second
      - consume ~35% of the total bits
  - In-loop filter
    - Removes artifacts present in the input stream
  - CABAC
    - Better entropy coding
  - Quarter sample and segmented MC
    - Reduces residuals to code
  - 25-30% reduction is possible at similar quality with above tools
  - Further reduction with multiple reference frames, weighted pred, etc.

Challenges

- **Computational complexity is significantly higher**
  - **While transform and quantization are simpler, the following add a lot of complexity**
    - **Deblocking filter in the loop**
      - **Adaptive; lots of conditional code**
        - » **Computing boundary strength; signal-based thresholds**
      - **Multiple filters (3 to 5 taps; variable coefficients)**
    - **Half/Quarter-sample interpolation with a 6-tap filter**
      - **rather than bilinear interpolation**
    - **Quarter pixel motion refinement**
    - **Supporting inter prediction modes not available in MPEG-2**
    - **Intra prediction**
      - **Planar mode (takes ~10 operations per pixel)**
      - **Others (~5 operations per pixel)**
    - **Context-adaptive entropy coding**
      - **Multiple VLC tables or arithmetic coder contexts**
      - **Renormalization complexity in arithmetic encoding**

Challenges (Contd.)

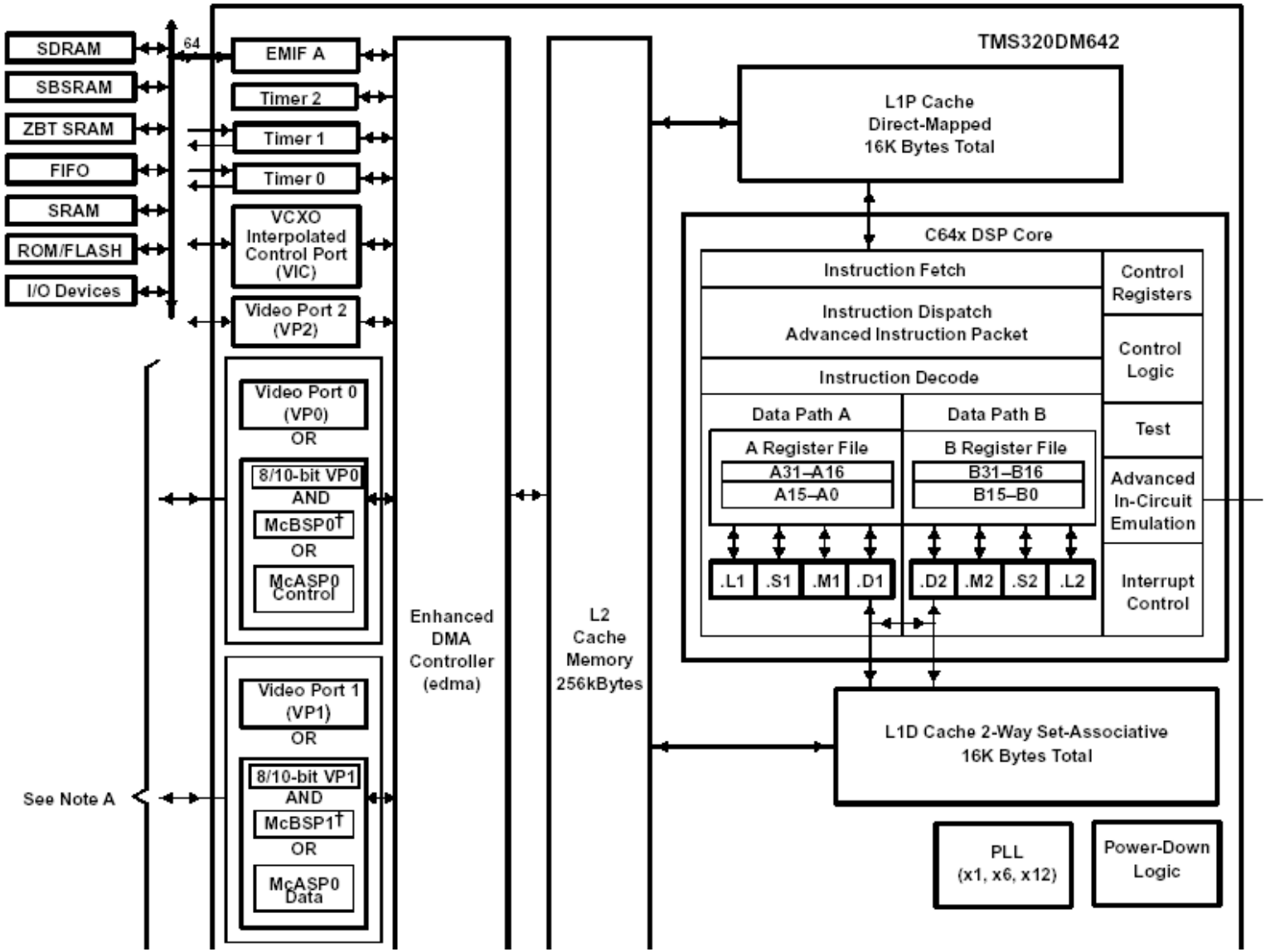
- Control flow issues
  - Granularity of interaction between decoder and re-encoder
  - Intra (4x4) prediction requires reconstructed samples
  - Deblocking cannot be performed immediately following the decoding of a macroblock
- Memory bandwidth problems
  - Decoder bandwidth + Re-encoder bandwidth
  - Display buffer (4:2:2) transfer bandwidth
  - Optional Intermediate processing stage bandwidth
- Code size
  - I-Cache misses depend on granularity

Programmable Processor vs. ASIC

- In a PVR/STB,
  - Multiple format support on the same device
    - E.g. MPEG-2, MPEG-4, H.264, WMV
    - Newer standards (AVS, SMPTE WMV)
- Time to market
- Flexibility to change specifications
  - Application specific
  - Standard specific (e.g. corrigendum to H264)
- Ability to do field upgrades

- TI DM64x is one of the select few DSPs to offer enough power to realize worthwhile single-chip configurations

TI TMS320DM642



TI C64x DSP Core

- VLIW architecture (VelociTI)
  - 256-bit instructions (8 32-bit)
  - Dual data paths
    - 4 functional units (L, M, S, D) each
      - L, S – arithmetic/logical; M – multiply/shift; D – Data load/store
    - 32 32-bit registers each
    - Cross path access to registers
  - Can load and store up to 64-bits
    - Non-aligned load/stores are supported
  - All instructions can be conditionally executed
  - Packed 8-bit and 16-bit operations
    - E.g. AVGU4, DOTPU4, ADD2, MIN2, MAX2, etc.
  - Instructions to pack and unpack
- Transfer crossbar with 4 queues of varying priority
- 64-channel enhanced DMA (TCInt, Chaining, linking)
- 2-level cache (16 kB I and D – L1; 256 kB L2)

TI DM642 Features

- 2 External Memory Interfaces
  - 1 64-bit (EMIFA); 1 16-bit (EMIFB)
- 3 video ports
  - Can be configured as video input, video output, or transport stream interface
- Multi-channel Audio Serial Port (Data/Control)
- 2 Multi-channel Buffered Serial Ports
- Ethernet MAC
- PCI interface
- Host Post Interface
- I2C Interface

TI DM642 EVM Features

- Composite/S-video input/output
- Line In/Out, Mic In
- 10/100 Mbps Ethernet Controller
- 32 MB SDRAM + 4 MB Flash
- PCI interface (boot option + HPI)
- EMIF Daughter Card Interface
- Video port Daughter Card Interface

Design Steps

1. Decide on granularity of interaction between Dec. & re-Enc.
2. Estimate rough MCPS and code size of various modules
   - Optimize leaf modules before estimating
3. Group the various processing stages suitably to develop a pipeline design
   - Decides code and data buffer requirement in ISRAM
   - Develop strategy for DMA of data and code (if necessary)
4. Implement the pipeline design by suitably modifying code
5. Adjust code placements and buffer placements in ISRAM

Granularity of Interaction

- **Options**
  - **Frame level synchronous**
  - **Frame level asynchronous**
  - **Multiple slice level**
- **Governing factors**
  - **MB pair in H.264 MB-AFF**
  - **Display while recording requirement**
  - **I-cache thrashing**
  - **EDMA bandwidth requirements**
  - **ISRAM requirements**

Optimizing Key Modules

- **Develop optimized assembly code for the leaf modules**
  - **Ensure that leaf modules have a clean interface**
    - **To avoid having to re-write ASM modules later**
  - **Optimization guidelines**
    - **Analyze whether module is I/O limited or process limited**
    - **Maximize the utilization by packing as many slots as possible**
      - **Reduce slots by using appropriate SIMD instructions**
      - **Evaluate alternate ways of reducing the # of execution packets**
      - **Balance the load between data-paths A and B in a clean way to reduce cross-path accesses**
    - **Explore unrolling possibilities**
      - **balance code-size, register pressure and speedup**
    - **Minimize stalls due to branches and delay slots**
    - **Use conditional execution to avoid shallow branches**
    - **Avoid memory bank stalls**

Exploiting Parallelism

- **Algorithmic Parallelism (High in video processing)**
  - **Each element of a group can be worked on independently. Enables**
    - **Packed data operations on multiple elements**
      - **E.g. AVGU4 - Compute four $(p_{i,j} + p_{i,j+1} + 1) >> 1$ operations in one instruction**
      - **E.g. DOTPU4 – $\Sigma w_i * p_i$ (4 8x8 multiplications and 3 adds)**
      - **Other instructions that come in handy**
        - » **DOTP2, SHRMB, SHLMB, UNPCK, PACK, SPACK, MIN2, MAX2, MPY4, SUBABS4**
    - **Scheduling of operations on multiple elements in slots of the same execution packet**
    - **Loop unrolling**
    - **Multiple ways of performing the same operation**
- **Instruction level Parallelism**
  - **Dissimilar operations in a code segment that have no dependencies and hence can be scheduled in parallel**
    - **Works out well even for control code**
    - **Can be used to fill empty slots in an execution packet**
    - **Only catch will be register pressure**

Pipeline Design

- Group the various processing stages such that
  - I-cache thrashing is minimized
  - Suitable pipelining can be established to minimize the cycles spent by the DSP waiting for data (Keep pipeline granularity flexible)
    - Goal is to maximally utilize DSP and DMA
  - Required buffering can be easily accommodated in the ISRAM
    - Re-use of scratch buffers minimizes L1 D-cache misses
- Processing stages in this case are:
  - Decoder's syntax parsing, motion compensation, texture decoding
  - Re-Encoder's motion refinement, encoding loop, entropy coding, deblocking
  - Optional intermediate processing & display related processing (if any)
- Verify the pipeline design
  - Use memcpys as placeholders for DMAs with suitable sync points in software (to verify completion of DMA)
  - Select the type of triggering for the various DMA channels
  - Implement the DMA and replace the memcpys

Impact of Grouping: Example

MB

Motion Refinement

Motion Compensation

Encode/Reconstruction

Entropy Coding

Deblocking

- Code Size beyond 16kB
- Every fetch packet thrashes L1P

MB Row

Motion Refinement

Motion Compensation

Encode/Reconstruction

Entropy Coding

Deblocking

- Code for each group is within 16kB
- But output of each group needs to be buffered
- L1D thrashes

Code Placement

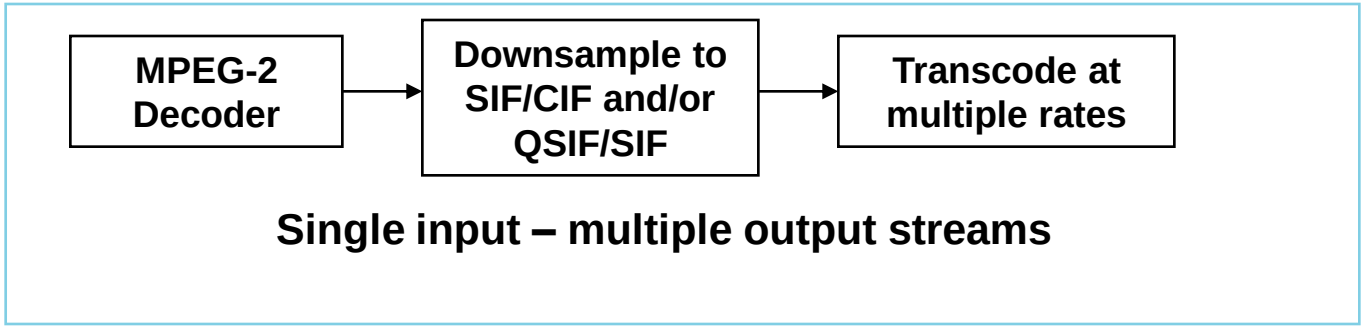
- Code for decoder + re-encoder
  - Much larger than L1 P-cache
  - Larger than even L2 cache
- Cache miss penalty is very high
  - 8 cycles for L1 miss
  - ~40 cycles for L2/ISRAM miss
- Prefetching of code is vital - requires
  - Careful code overlay at link time
  - Use of EDMA to prefetch required code
  - Suitable sync points in software to ensure proper operation

EDMA considerations

- Use QDMA whenever possible for short bursts of transfer
- Use chaining whenever possible to automate triggering
- Optimize the triggering done under an ISR
- Balance the load across the four priority queues and according to the criticality of the data
- Plan the buffer structures according to the restriction that 2D to 2D transfer is possible only if both source and destination strides are equal
- Ensure that DMAed area is not cached

Transcoding Variants

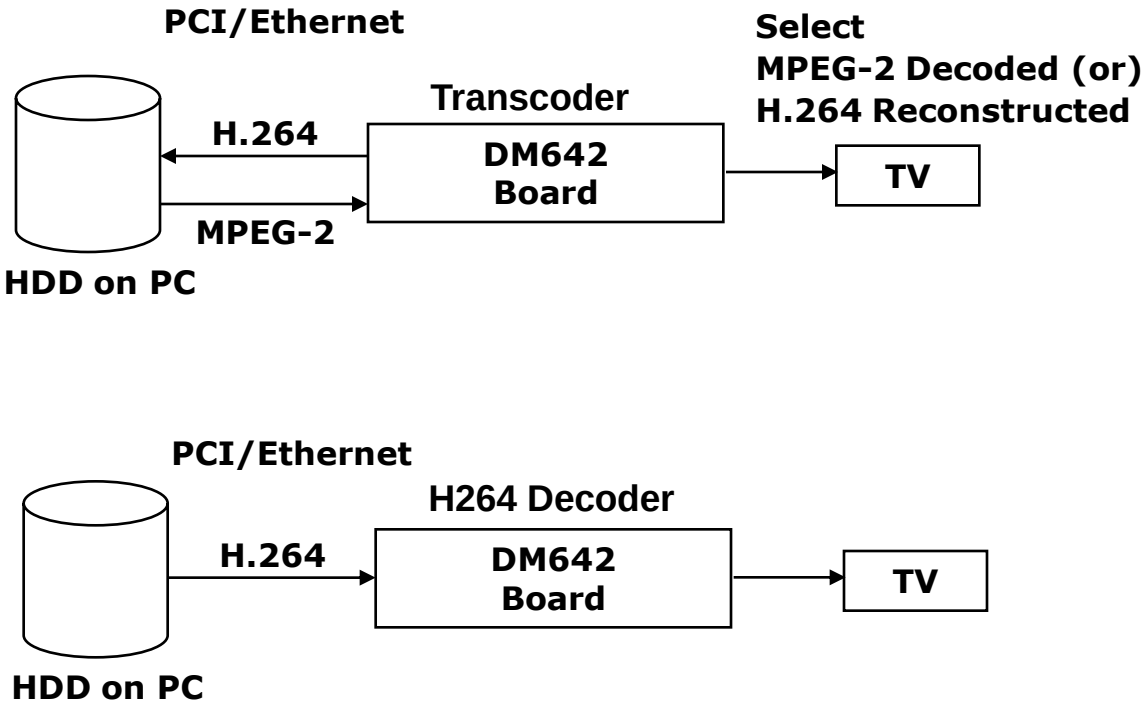
- **Options**
  - **Nearly the same quality as the input bitstream's**
    - **At ~30-40% lower bit-rates**
    - **Can use bit-allocation information of MPEG-2**
  - **Lower quality at even lower bit-rates**
    - **MPEG-2 bit allocation information not very relevant**
  - **Lower resolution at very low bit-rates - TransScaler**
    - **Optional Intermediate processing becomes a downsampler**
    - **Examples: D-1 to QCIF or CIF or half-D1 or 2/3-D1 or 3/4-D1**



Performance

- In real-time on a DM642@600MHz
  - Can transcode D-1 MPEG-2 MP@ML to half-D1 MP@L3
- Speed-ups on leaf modules up to 40x compared to –O3 compiled C code
  - Typical speed-ups in the range of 10-16x

Demonstration Setup



Conclusions

- **Real-time transcoder realized on DM64x**
  - **Complexity challenges**
    - **Addressed by exploiting ALP, ILP**
  - **Control flow issues**
    - **Handled through suitable pipeline design**
  - **Memory bandwidth problems**
    - **Mitigated using a multithreaded design between DSP and DMA**
  - **Code size problems**
    - **Mitigated using overlay with EDMA, code sectioning & positioning**
- **Further Challenges**
  - **Getting at least 2/3-D1 or 3/4-D1 transScaling**
    - **Future versions may clock higher as well (allowing even D-1)**
  - **Supporting time-shifted playback with recording on a PVR**
  - **Multi-channel transcoding**

Video and Imaging Track

MPEG-2 to H.264 Transcoding on DM64x

Sriram Sethuraman
Senior Member of Technical Staff
Ittiam Systems (Pvt.) Ltd.
Sriram.sethuraman@ittiam.com