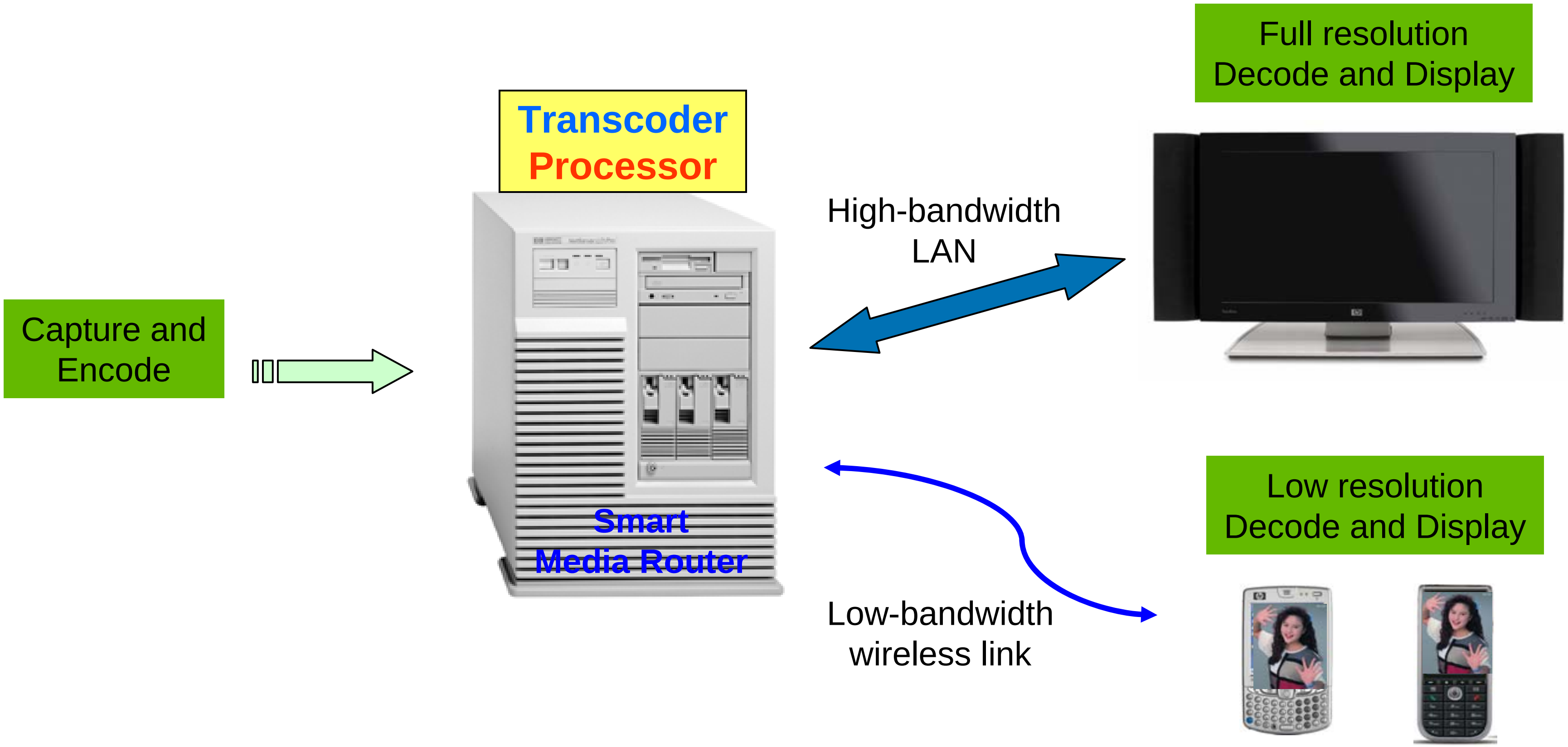
Compressed-Domain Video Processing and Transcoding

Susie Wee, John Apostolopoulos
Mobile & Media Systems Lab
HP Labs

Stanford EE392J Lecture

Compressed Domain Processing and Transcoding

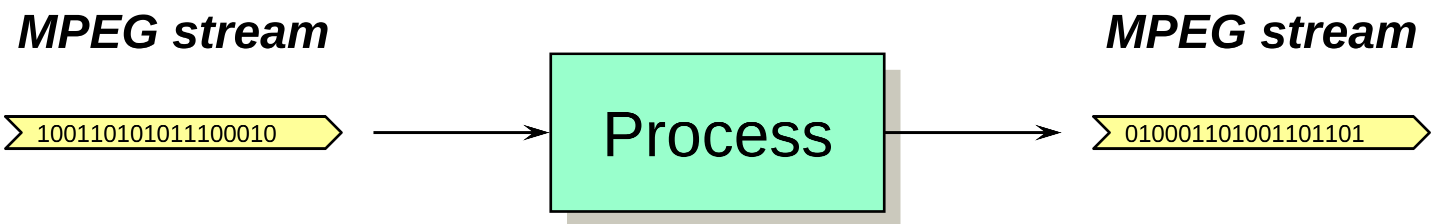


Transcoding: Media streaming over packet networks

Processing: Media processing in the compressed domain

Problem Statement

Goal: Efficiently process MPEG video streams

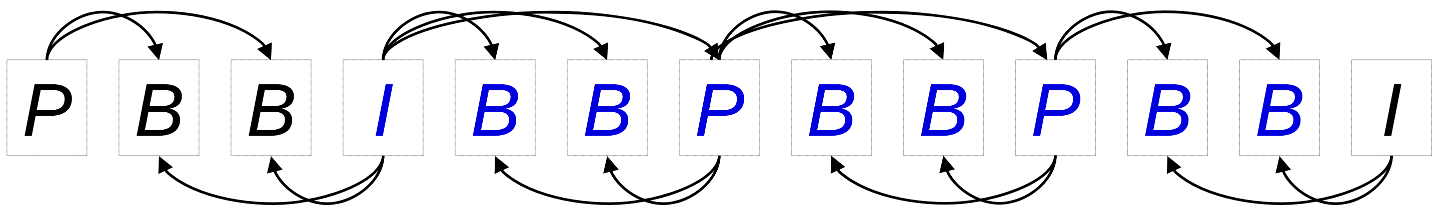


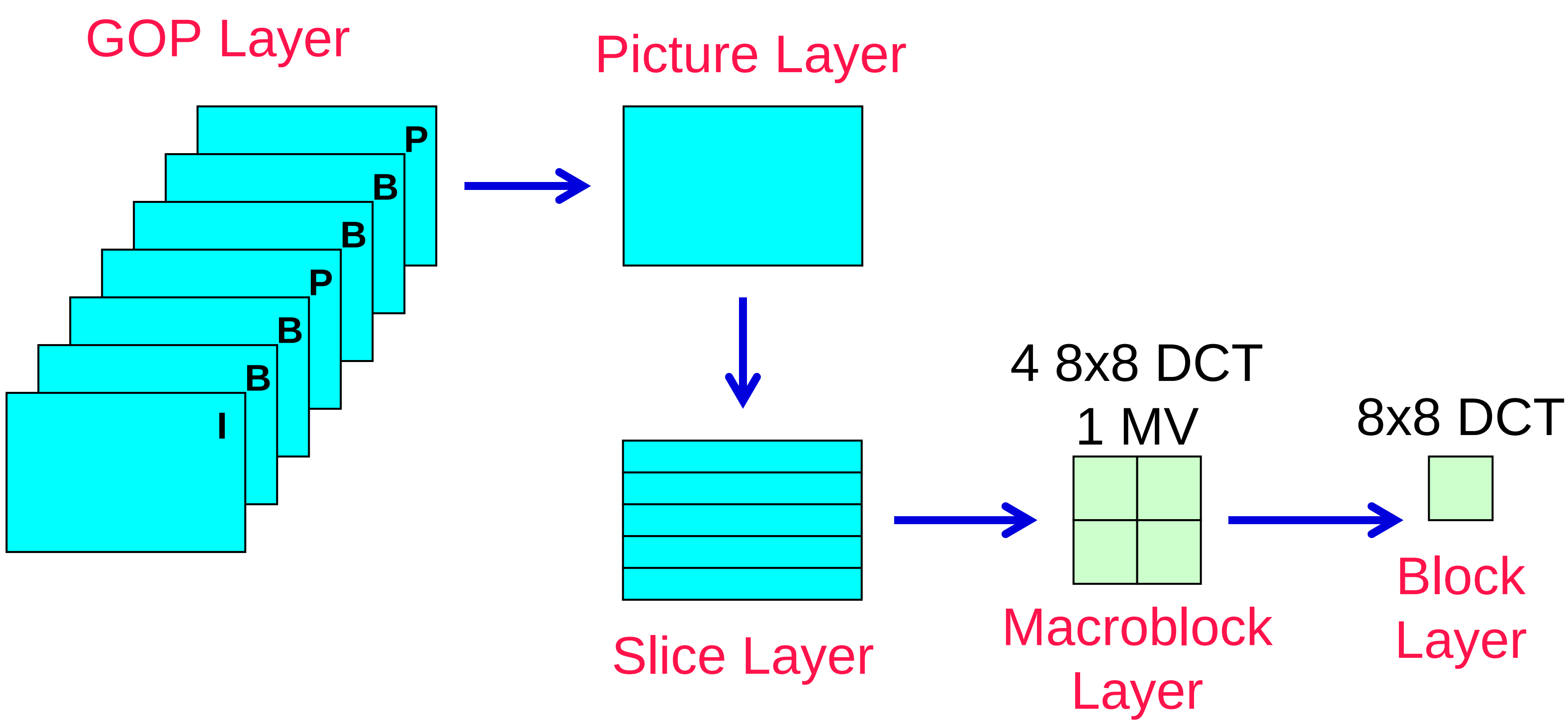
High-quality video processing with low computational complexity and low memory requirements.

Outline

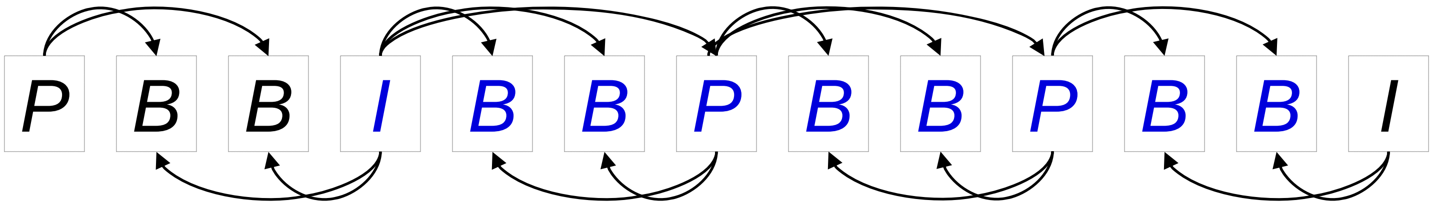
- Background
- Manipulating Temporal Dependencies
- Compressed-Domain Splicing
- Compressed-Domain Reverse Play
- MPEG2-to-H.263 Transcoding
- Overview and Demo

MPEG Structures





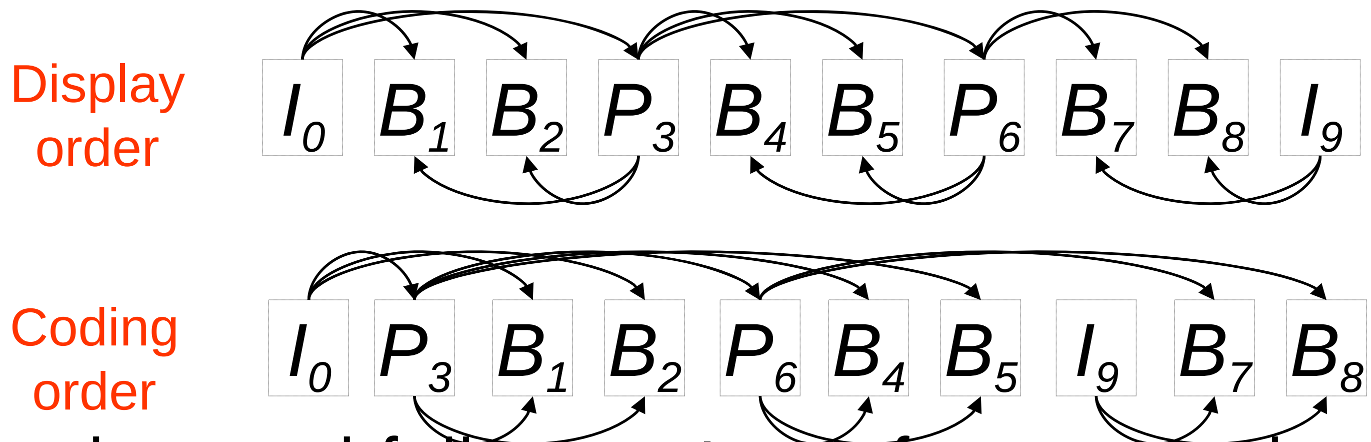
MPEG Structures



- GOPs*
- Pictures*: Intraframe (I frame), Forward predicted (P frame), Bidirectionally predicted (B frame)
- Slices*: (16x16n pixel blocks)
- Macroblocks (16x16 pixel blocks):
  - 1 MV and 4 DCT blocks per MB (plus chrominance)
    - I frame: Intra MBs
    - P frame: Intra or Forward MBs
    - B frame: Intra, Forward, Backward, or Bidirectional MBs
- Blocks (8x8 pixel blocks): 8x8 DCT

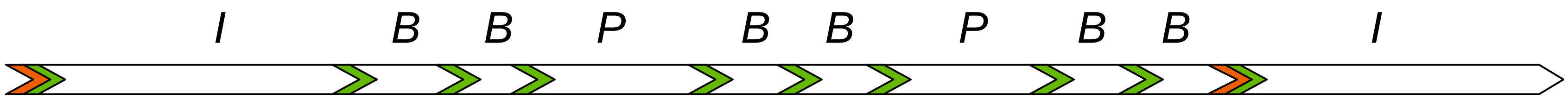
(* start codes)

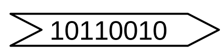
MPEG Coding Order



- Preceding and following I or P frames (anchors) are used to predict each B frame.
- "Anchor" frames must be decoded before B data. (Note: In coding order, all arrows point forward.)

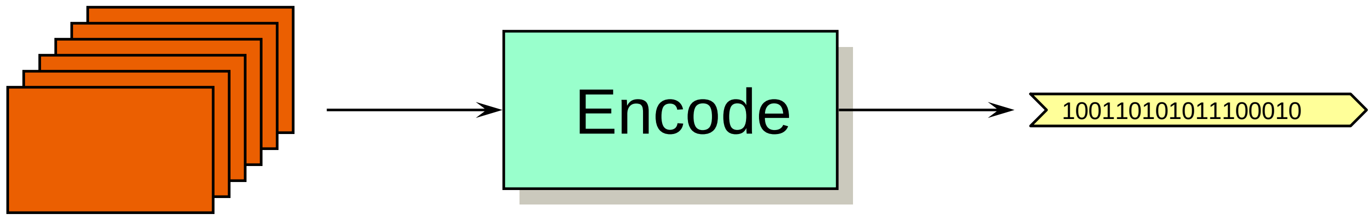
MPEG Bitstream



- GOP header
- Picture header
- Picture data 

- Pictures in coding order
- Start codes (seq, GOP, pic, and slice start codes; seq end code)
  – Unique byte-aligned 32-bit patterns
  – *0*x*000001nm* 23 zeros, 1 one, 1-byte identifier
  – Enables random access

Video Compression



Raw video

MPEG stream

HDTV quality

720 x 1280 pixels, 60 fps
~ 1 Gbps

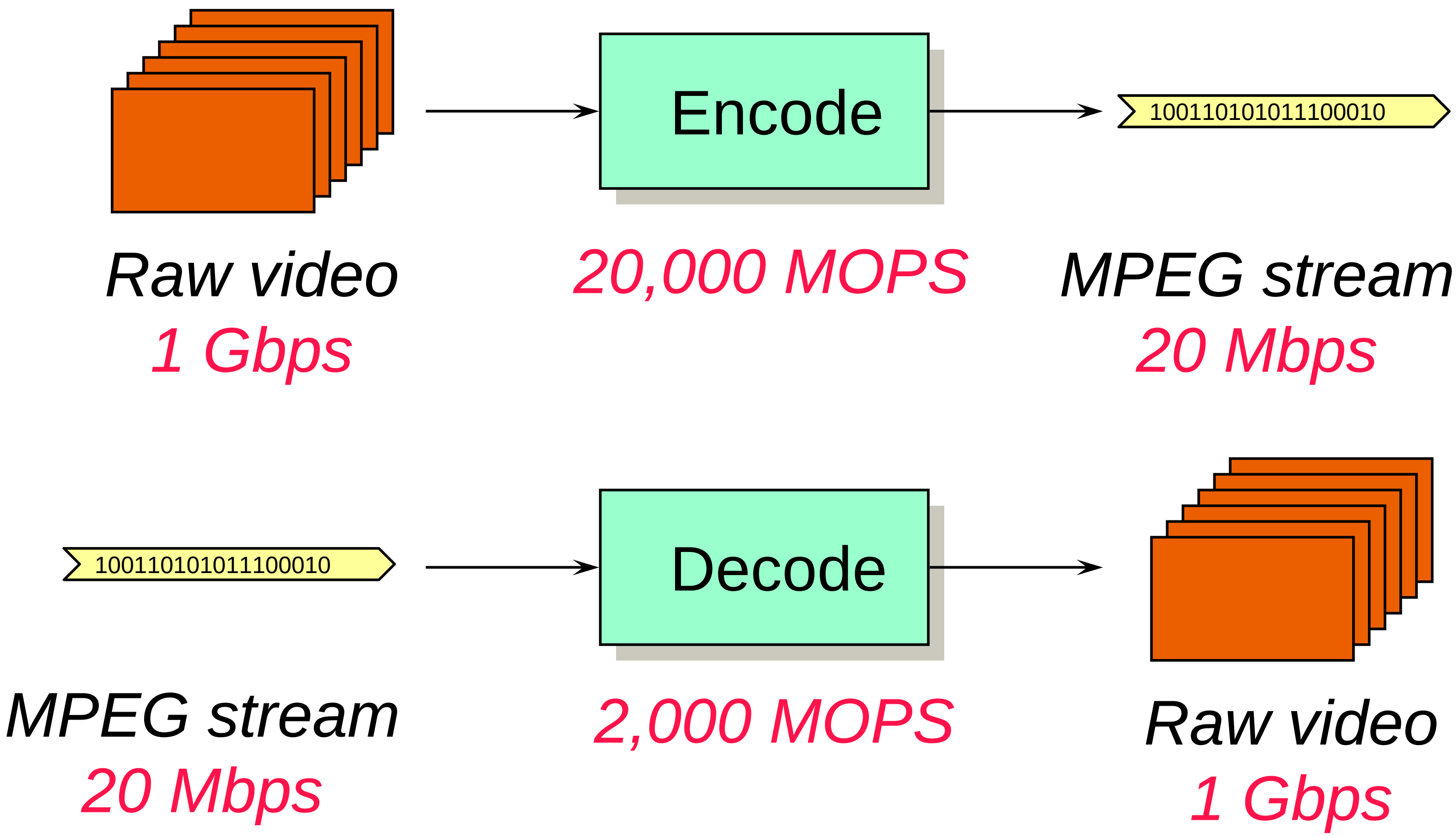
20 Gbytes / 2 hour movie
~ 20 Mbps

Broadcast quality

480 x 720 pixels, 30 fps
~ 250 Mbps

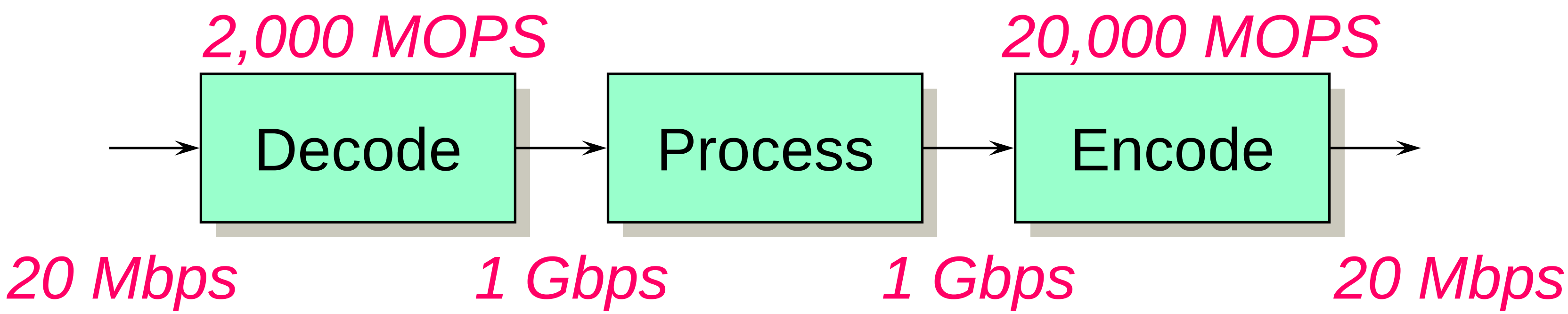
5 Gbytes / 2 hour movie
~ 5 Mbps

MPEG Video Compression



Problem statement

How do we process compressed video streams?



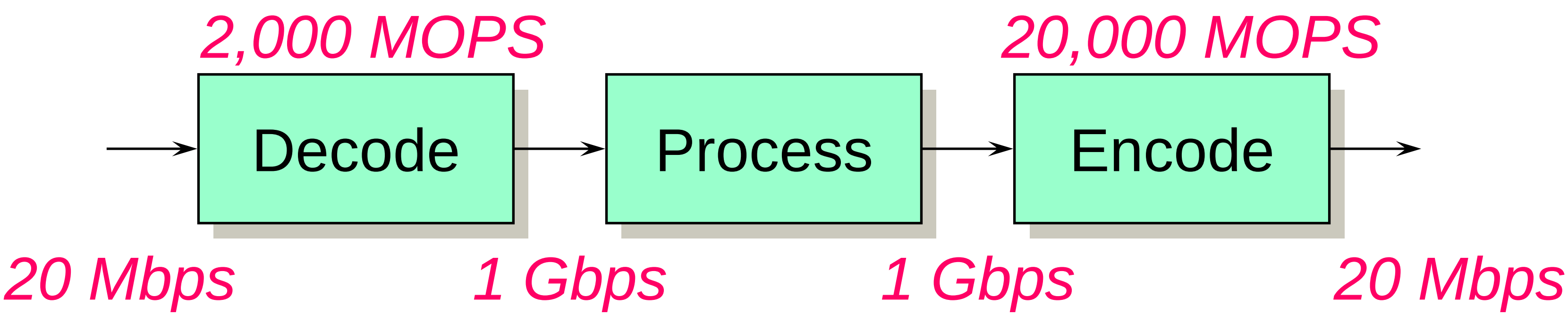
- Difficulties:
  - Computational requirements: 22,000 MOP overhead from decode and encode operations (plus additional processing)
  - Bandwidth requirements: Need to process uncompressed data at 1 Gbps
  - Quality issues: Even without processing, the decode/encode cycle causes quality degradation.

MPEG CDP

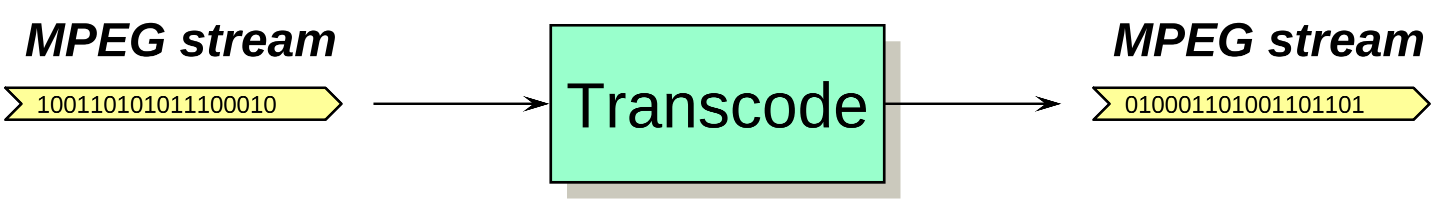
How do we process compressed video streams?

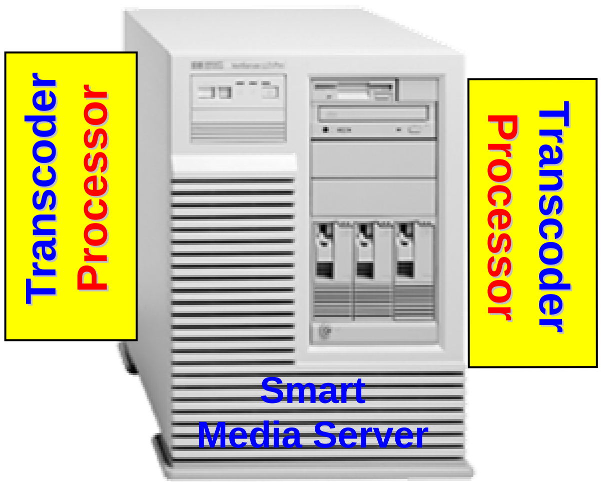
Use efficient compressed-domain processing (CDP) algorithms.

Naive Solution:



Ideal Solution:





Decode | SPATIAL-DOMAIN PROCESSING | Encode

2,000 MOPS | > 20,000 MOPS

Conventional Approach

Compressed-Domain Processing

Huffman Decode | MV+DCT-DOMAIN PROCESSING | Huffman Encode

200 MOPS | 200 MOPS

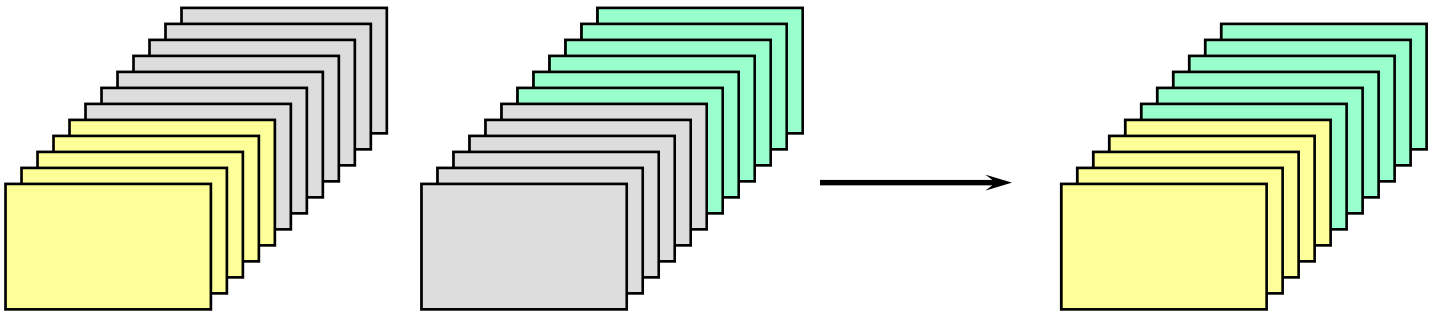
BITSTREAM PROCESSING

OUR APPROACH

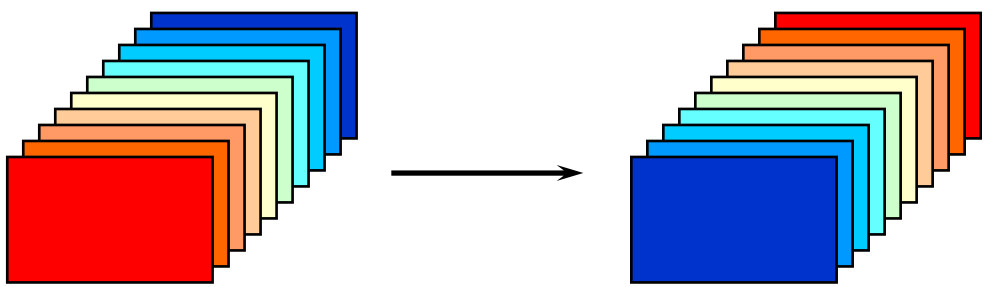
Develop algorithms to perform equivalent spatial-domain video processing tasks using fast algorithms that operate directly on the compressed-domain data.

Frame-Level Video Processing

Splicing



Reverse Play



Difficulties and Solutions

- High compression causes temporal dependencies
  - Frame conversion
- Coding order vs. display order
  - Data reordering
- Rate control / Buffer limitations
  - Requantization, Frame conversion
- MPEG header information (e.g. time stamps, vbv_delay)
  - Rewrite header bits

Outline

- Background
- ➢ Manipulating Temporal Dependencies
- Compressed-Domain Splicing
- Compressed-Domain Reverse Play
- MPEG2-to-H.263 Transcoding
- Overview and Demo

Manipulating Temporal Dependencies in Compressed Video

- Video compression uses temporal prediction
  - → Prediction dependencies in coded data
- Many applications require changes in the dependencies
- Manipulating (modifying) temporal dependencies in the compressed video [Wee]
  - Adding
  - Removing
  - Changing

Temporal Prediction

Original Video Frames: $\mathbf{F} = \{ F_1, F_2, \ldots, F_n \}$

Coded Video Frames: $\hat{\mathbf{F}} = \{ \hat{F}_1, \hat{F}_2, \ldots, \hat{F}_n \}$

Each frame F_i is coded with two components

1. Prediction $P_i(\hat{\mathbf{A}}_\mathbf{i}, S_i)$

   Anchor frames $\hat{\mathbf{A}}_\mathbf{i} \subseteq \{ \hat{F}_1, \hat{F}_2, \ldots, \hat{F}_{i-1} \}$

   Side information S_i

2. Residual $R_i = F_i - P_i(\hat{\mathbf{A}}_\mathbf{i}, S_i)$

Coded Residual $\hat{R}_i$

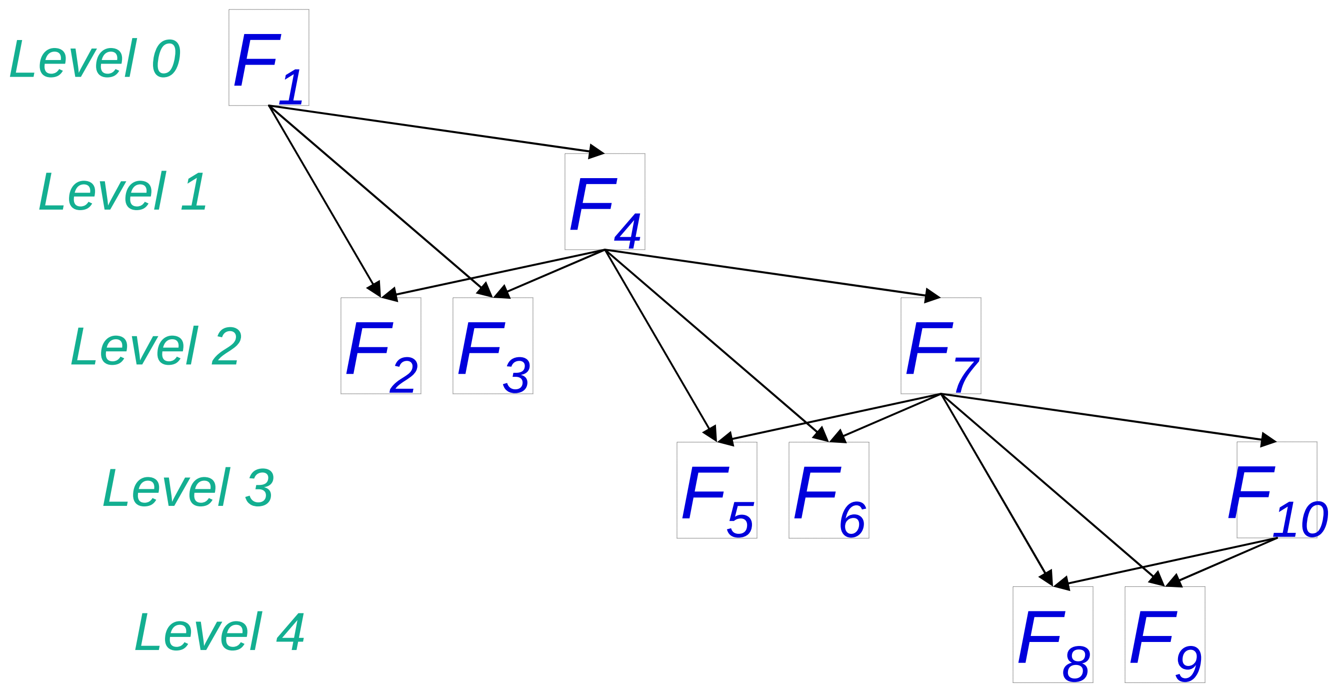
Reconstructed Frame $\hat{F}_i = P_i(\hat{\mathbf{A}}_\mathbf{i}, S_i) + \hat{R}_i$

Prediction Dependencies

Each coded frame is *dependent* upon its anchor frames.

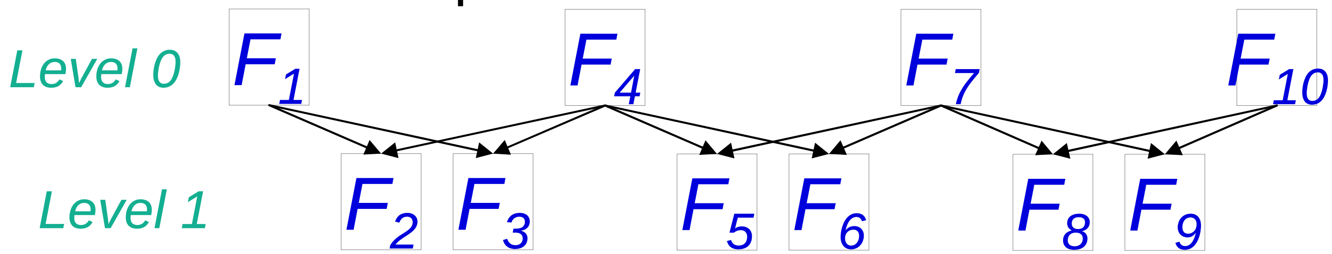
$$\hat{F}_i = P_i(\hat{\mathbf{A}}_i, S_i) + \hat{R}_i$$

Prediction Depth

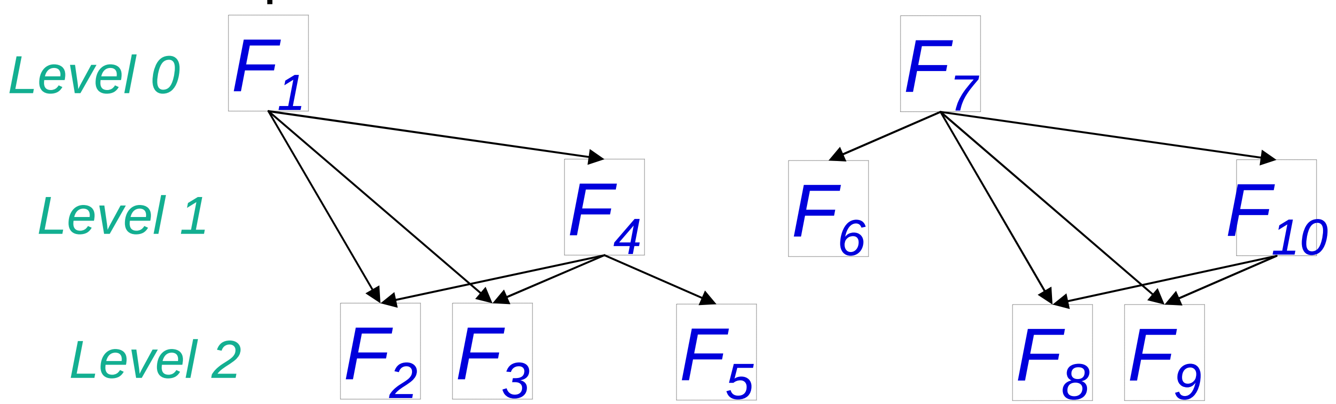


Manipulating Dependencies

Fewer levels of dependence



Remove dependence between frames



Problem Formulation

Compression algorithm has a set of prediction rules.

Choose prediction modes during encoding.

$$\hat{\mathbf{A}}_{\mathbf{i}}, S_i \qquad\qquad \hat{\mathbf{A}}_{\mathbf{i}}^{'}, S_i^{'}$$

$$\hat{F}_i = P_i\left(\hat{\mathbf{A}}_{\mathbf{i}}, S_i\right) + \hat{R}_i \qquad\qquad \hat{F}_i^{'} = P_i^{'}\left(\hat{\mathbf{A}}_{\mathbf{i}}^{'}, S_i^{'}\right) + \hat{R}_i^{'}$$

Each choice yields:

- different compressed representation of frame F_i ,
- different distribution between P & R components,
- different set of prediction dependencies.

Manipulating Dependencies

Given a compressed representation with dependencies $\hat{\mathbf{A}}_{\mathbf{i}}, S_i$

how do we create a new compressed representation with dependencies $\hat{\mathbf{A}}'_{\mathbf{i}}, S'_i$?

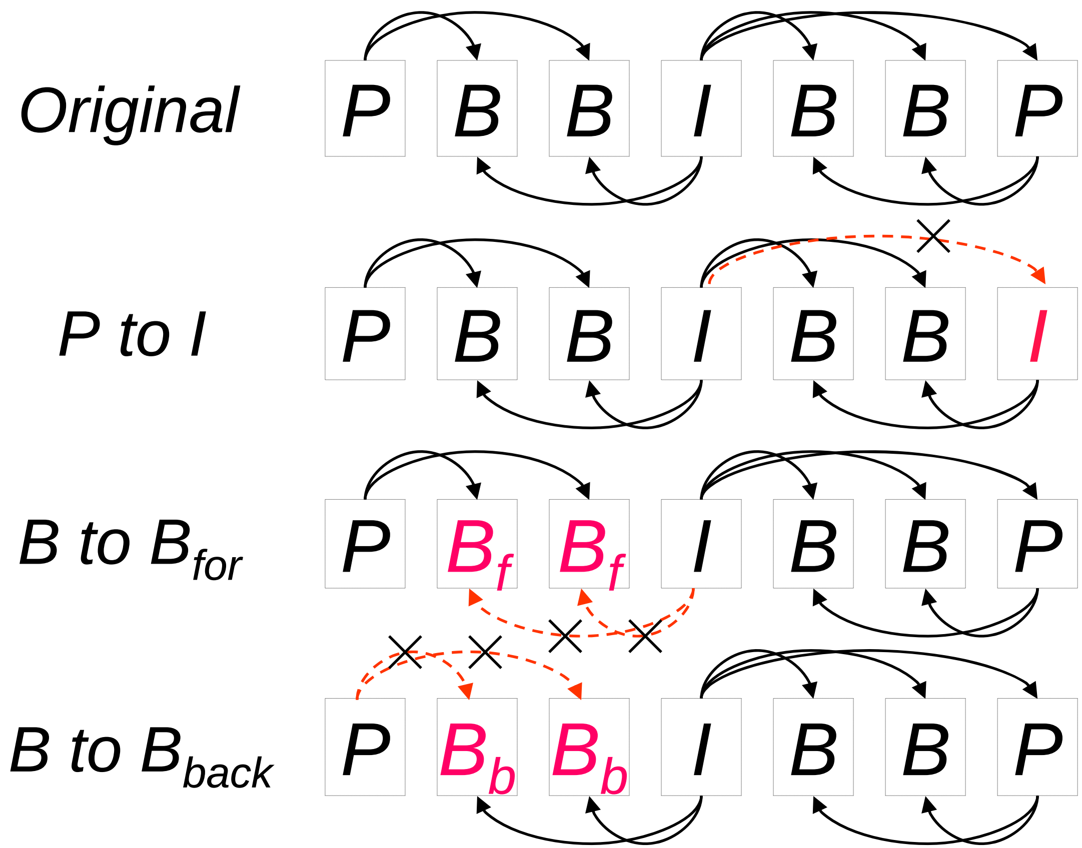
- Reconstruct frames $\hat{F}_i = P_i\left(\hat{\mathbf{A}}_{\mathbf{i}}, S_i\right) + \hat{R}_i$
- Compressed-domain approximation

$$\hat{F}_i \approx F_i \qquad R'_i = F_i - P'_i\left(\hat{\mathbf{A}}'_{\mathbf{i}}, S'_i\right)$$

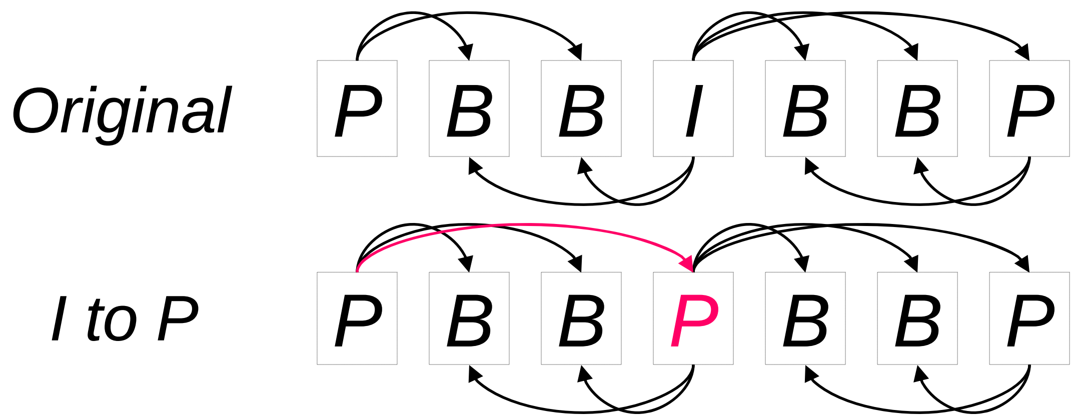
- Calculate new prediction and residual

$$R'_i \approx \hat{R}_i + P_i\left(\hat{\mathbf{A}}_{\mathbf{i}}, S_i\right) - P'_i\left(\hat{\mathbf{A}}'_{\mathbf{i}}, S'_i\right)$$

Frame Conversions: Remove Dependencies



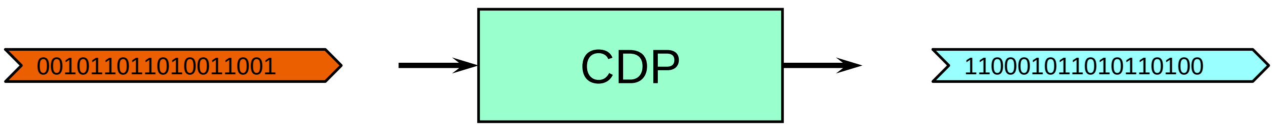
Frame Conversion: Add Dependencies



1. Find and code new MVs. *MV Resampling*
2. Calculate new prediction.
3. Calculate and code new residual.

Compressed-Domain Processing

MPEG standard addresses prediction rules, buffer requirements, coding order, and bitstream syntax.



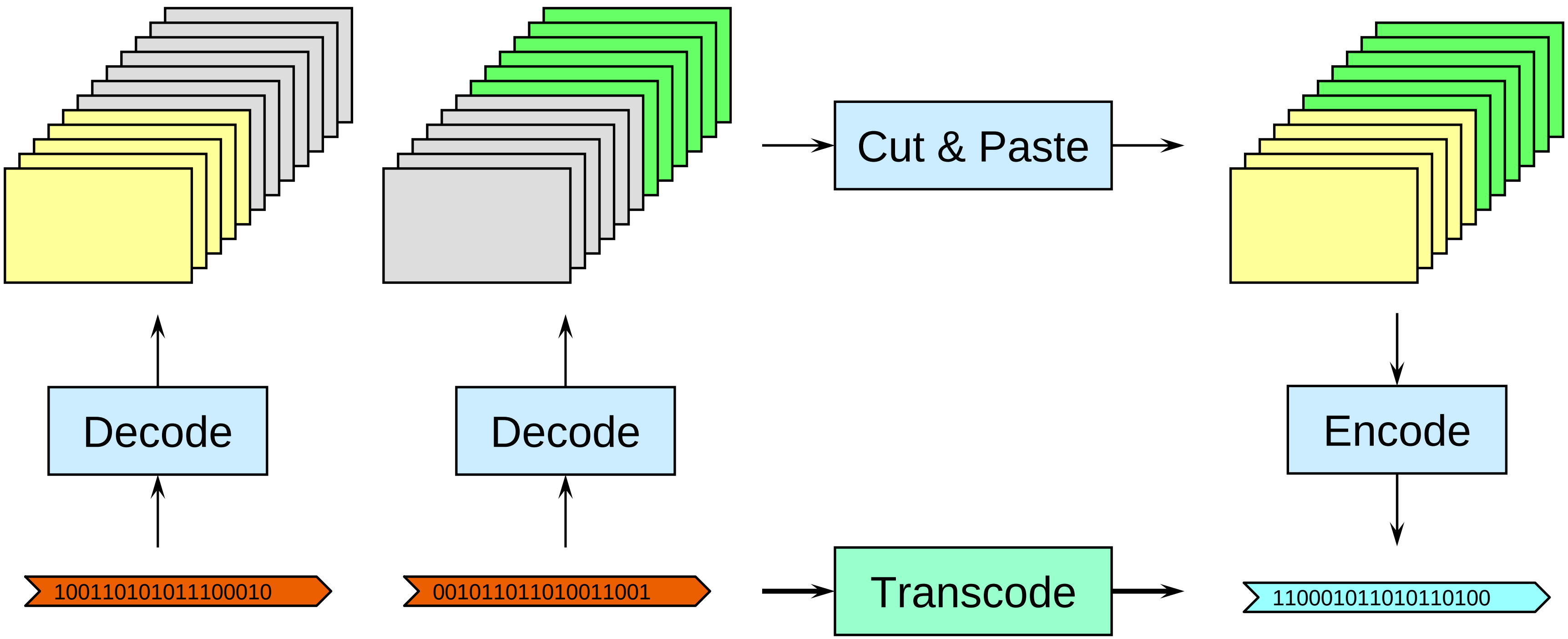
MPEG CDP steps

- Determine and perform appropriate frame conversions (i.e. manipulate dependencies).
- Reorder data.
- Perform rate matching.
- Update header information.

Outline

- Background
- Manipulating Temporal Dependencies
- ➢ Compressed-Domain Splicing
- Compressed-Domain Reverse Play
- MPEG2-to-H.263 Transcoding
- Overview and Demo

Compressed-Domain Splicing

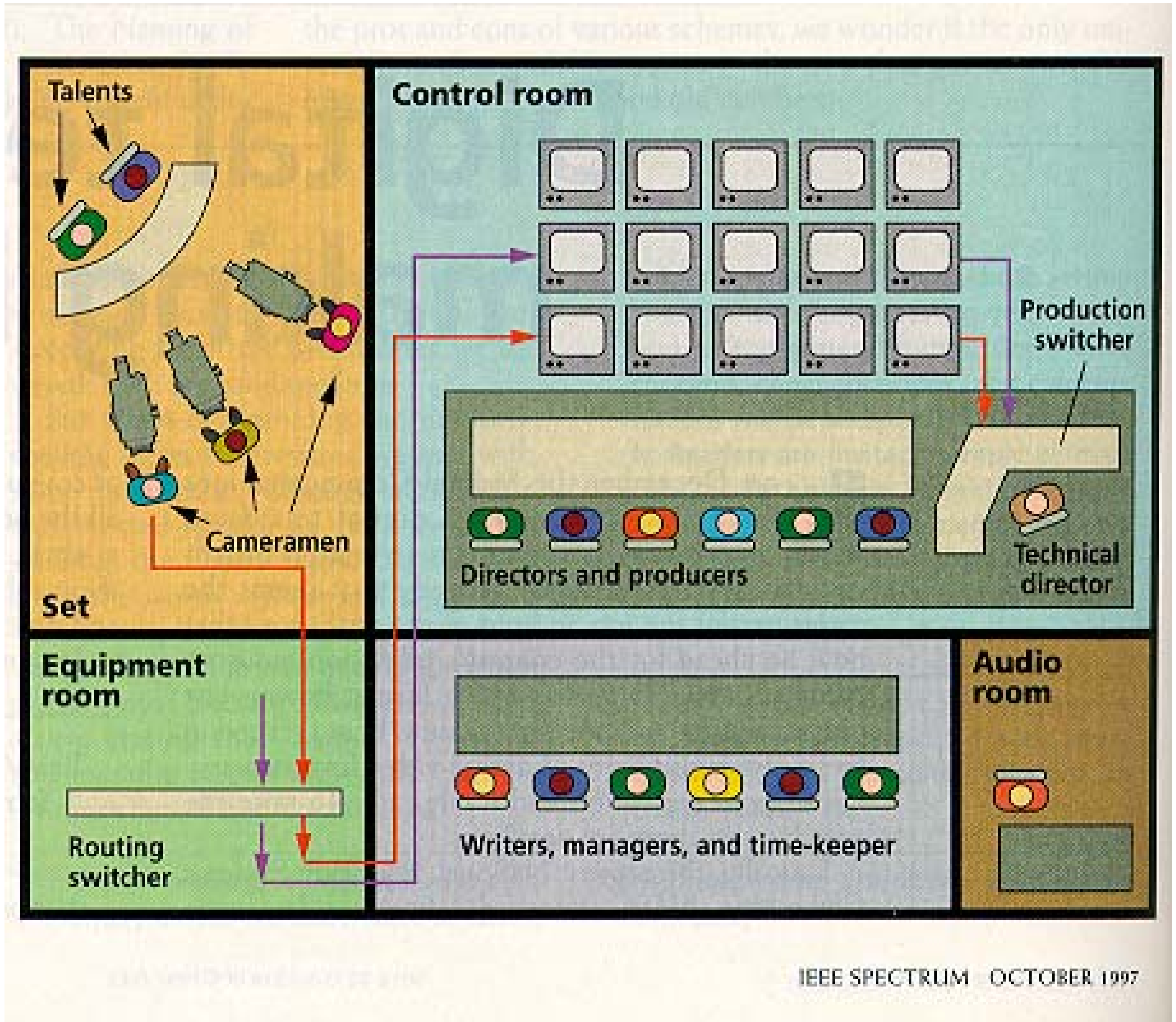


Application: Ad Insertion for DTV

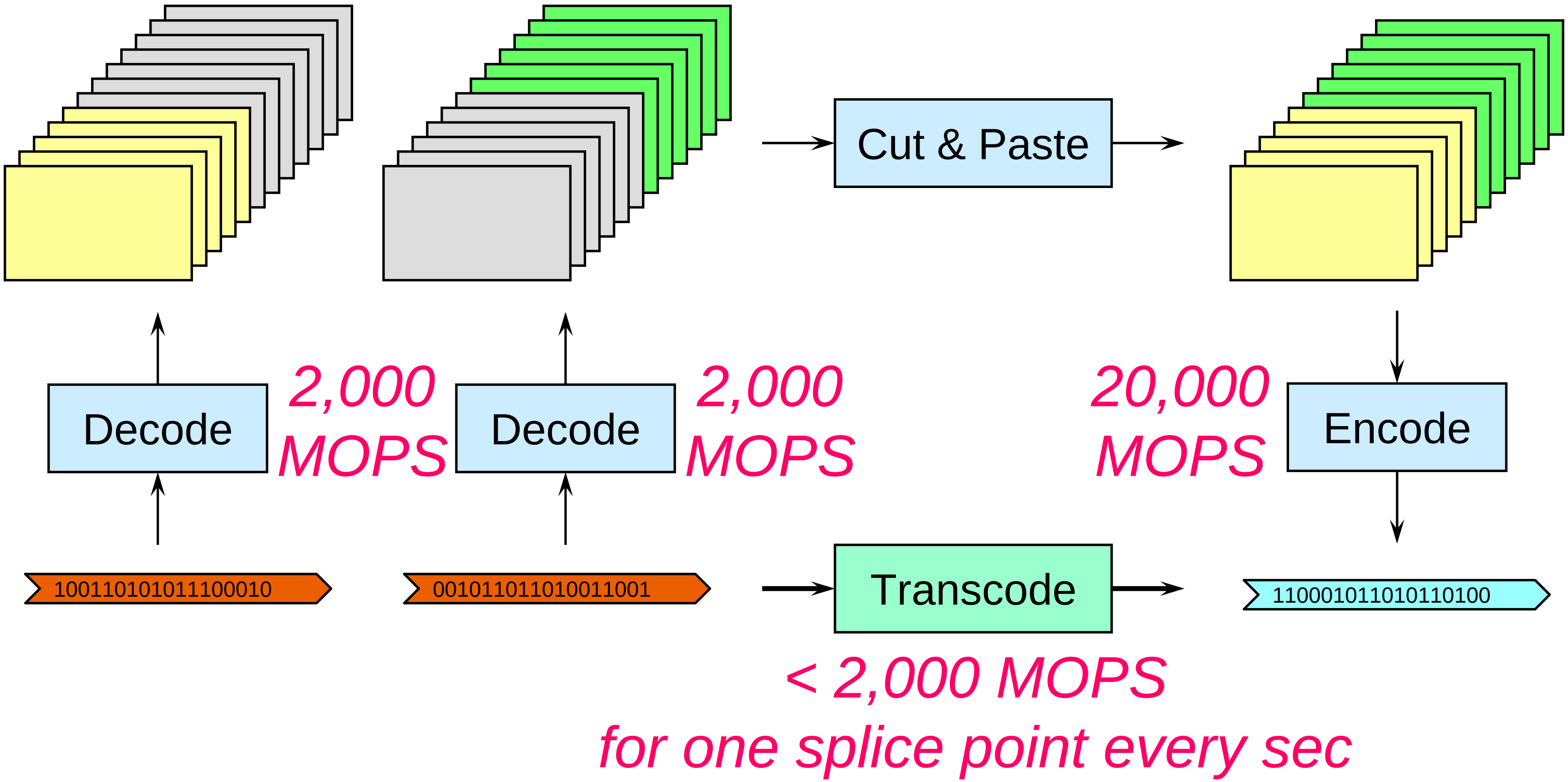
Splicing: SMPTE standard

- SMPTE formed a committee on splicing.
- *Disadvantages:*
  - User must predefined splice points during encoding
    - ⇒ Complicated encoders.
  - Splice points can only occur on I frames
    - Not frame accurate.
- *Advantages:*
  - Simple cut-and-paste operation.

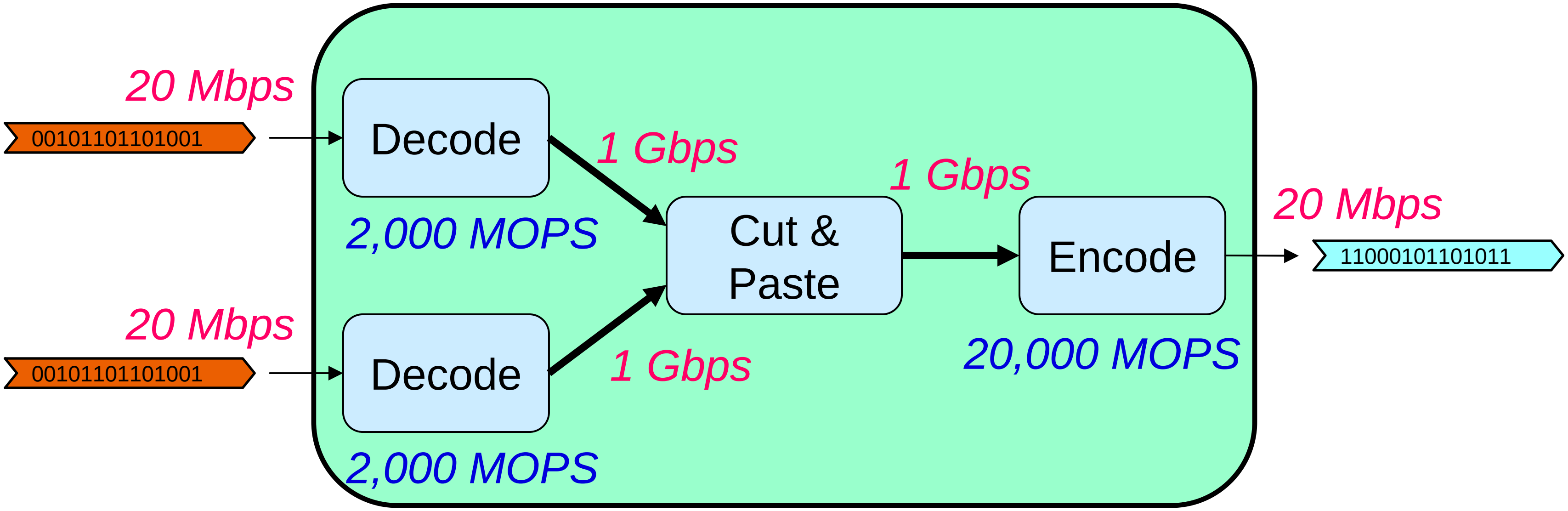
The TV Newsroom **IEEE Spectrum**



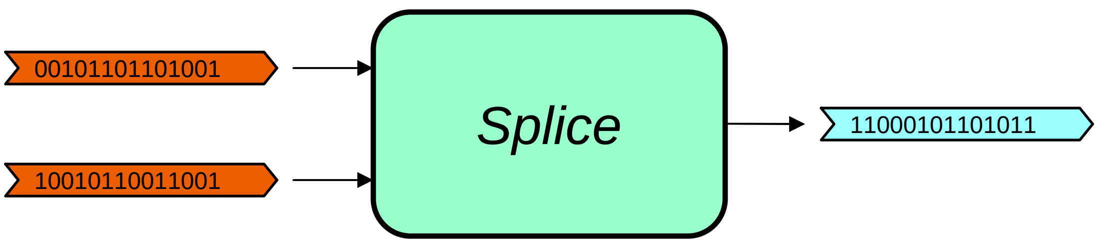
Compressed-Domain Splicing



Splicing: Conventional approach



> 24,000 MOPS *for one splice point every sec*



Compressed-Domain Processing

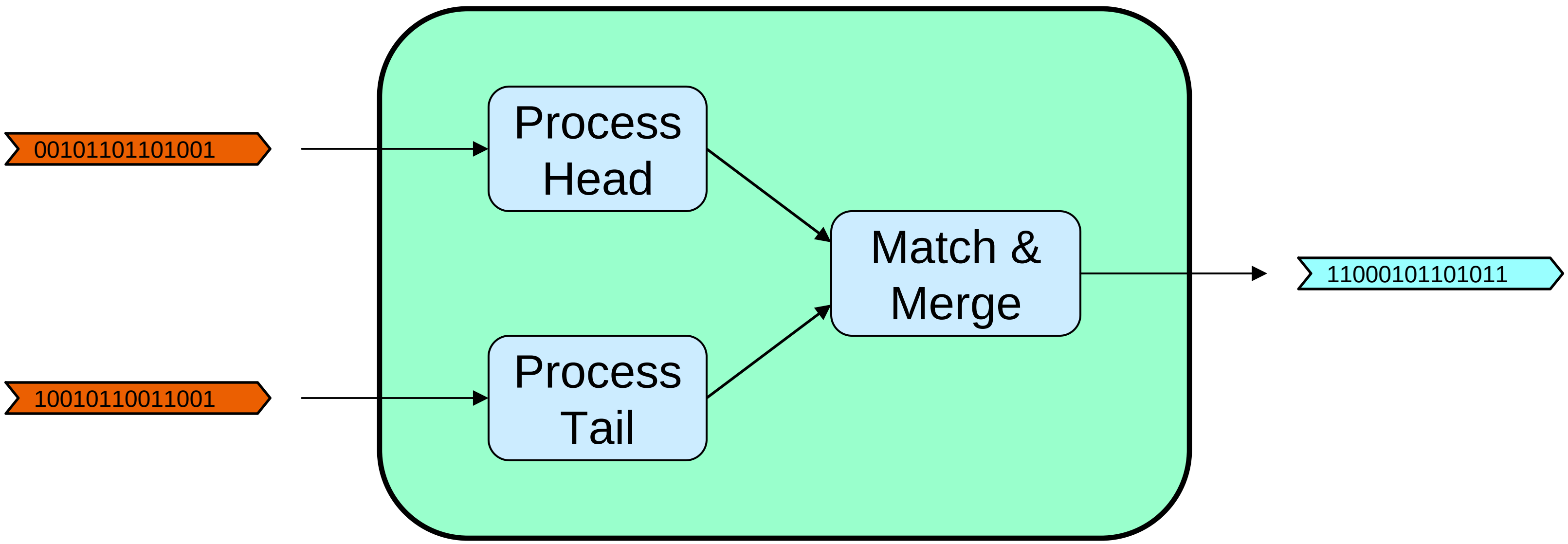
Can we do better by exploiting properties of

1) the MPEG compression algorithm

and

2) the splicing operation ?

Splicing: Our approach



< 2,000 MOPS *for one splice point every sec*

Splicing Algorithm (simplified overview)

Only process the GOPS affected by the splice.

Step 1: Process the head stream.

- Remove (backward) dependencies on dropped frames.

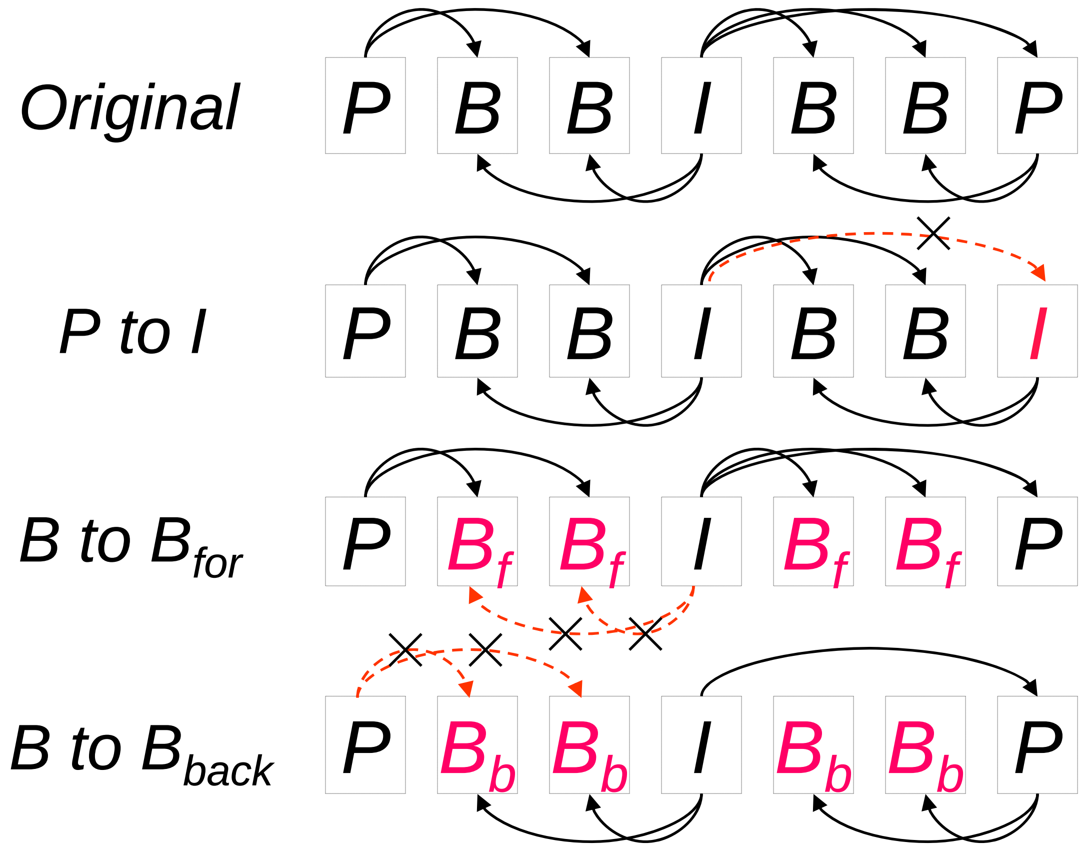
Step 2: Process the tail stream.

- Remove (forward) dependencies on dropped frames.

Step 3: Match & Merge the head and tail data.

- Perform rate matching.
- Reorder data appropriately.
- Update header information.

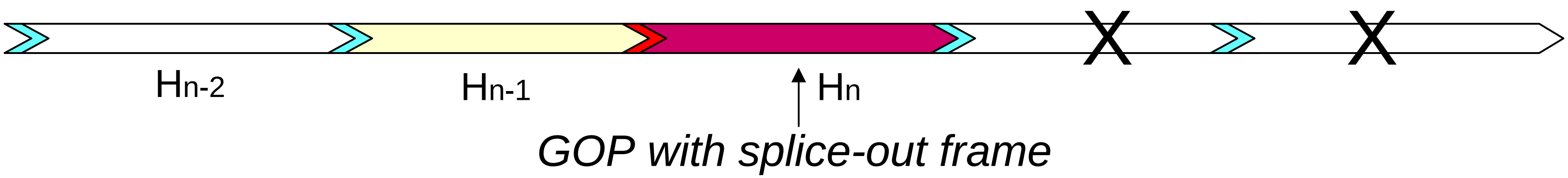
Frame Conversion: Remove Dependencies



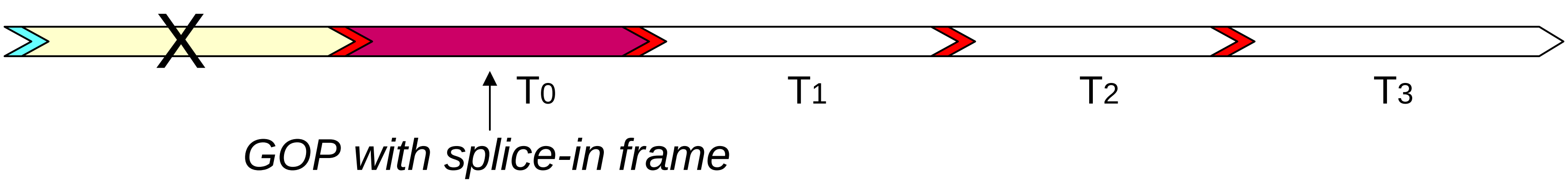
1. Eliminate temporal dependencies.
2. Calculate new prediction (DCT-domain).
3. Calculate and code new residual.

Splicing

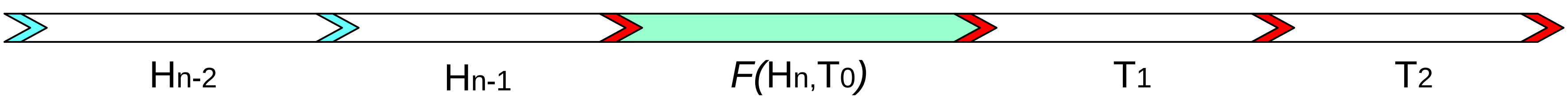
Head Input Stream



Tail Input Stream

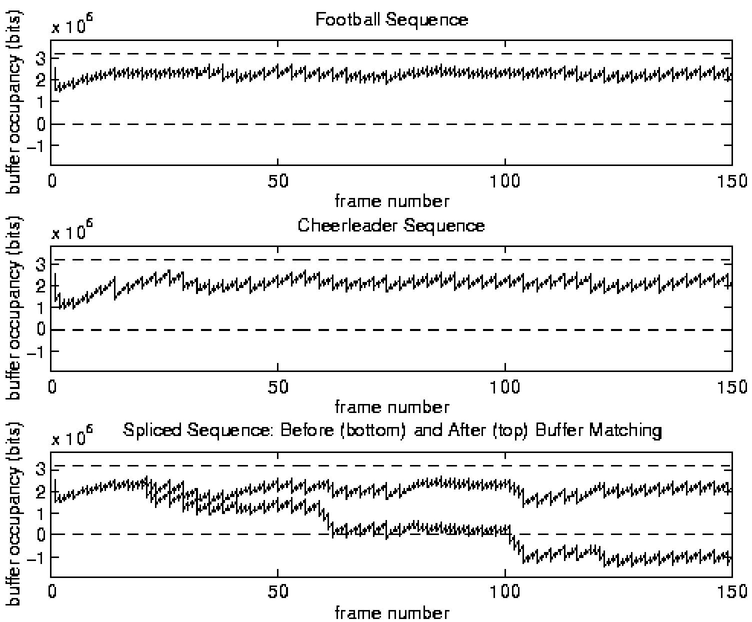


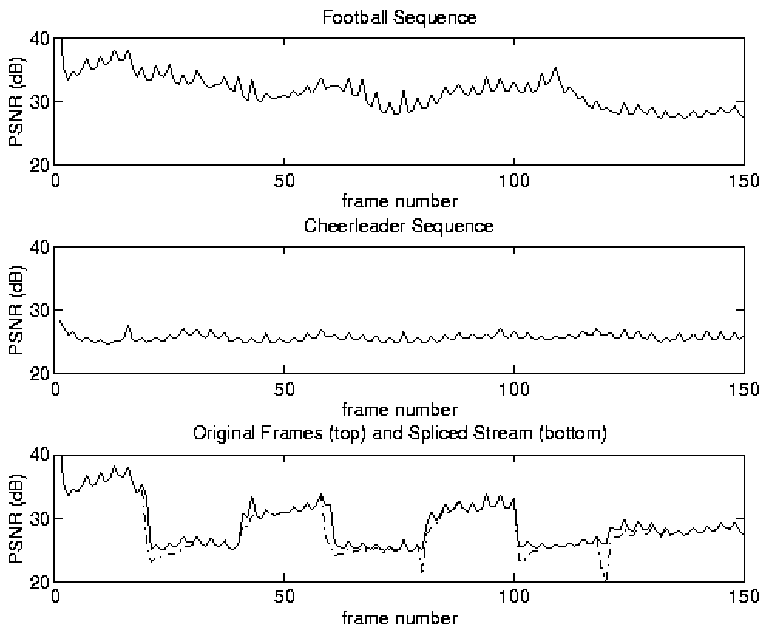
Spliced Output Stream



Results

Splicing between two sequences
with one splice every 20 frames





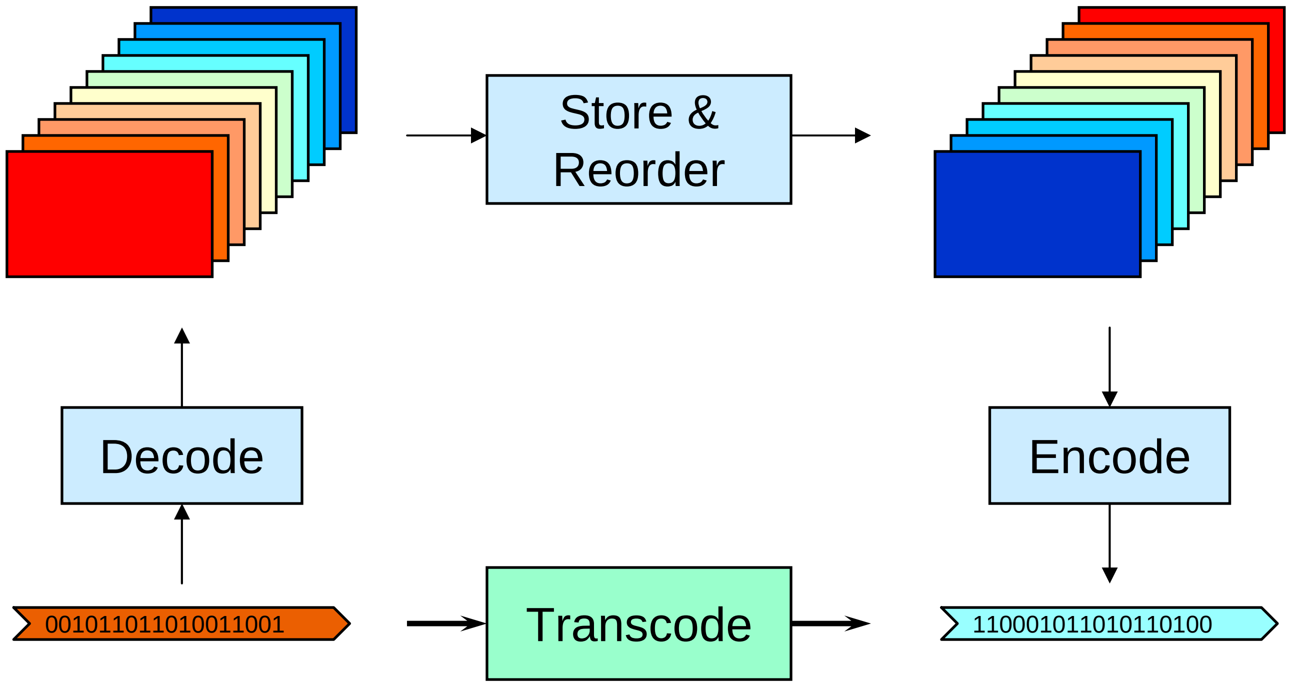
Splicing: Remarks

- Proposed Algorithm
  - Frame-accurate splice points
  - Only uses MPEG stream (Encoder is not affected.)
  - Frame conversions in MV+sparse DCT domain.
  - Rate control by requantization and frame conversion. (Do not insert synthetic frames.)
  - Quality only affected near splice points.
  - Computational scalability: Video quality can be improved if extra computing power is available.

Outline

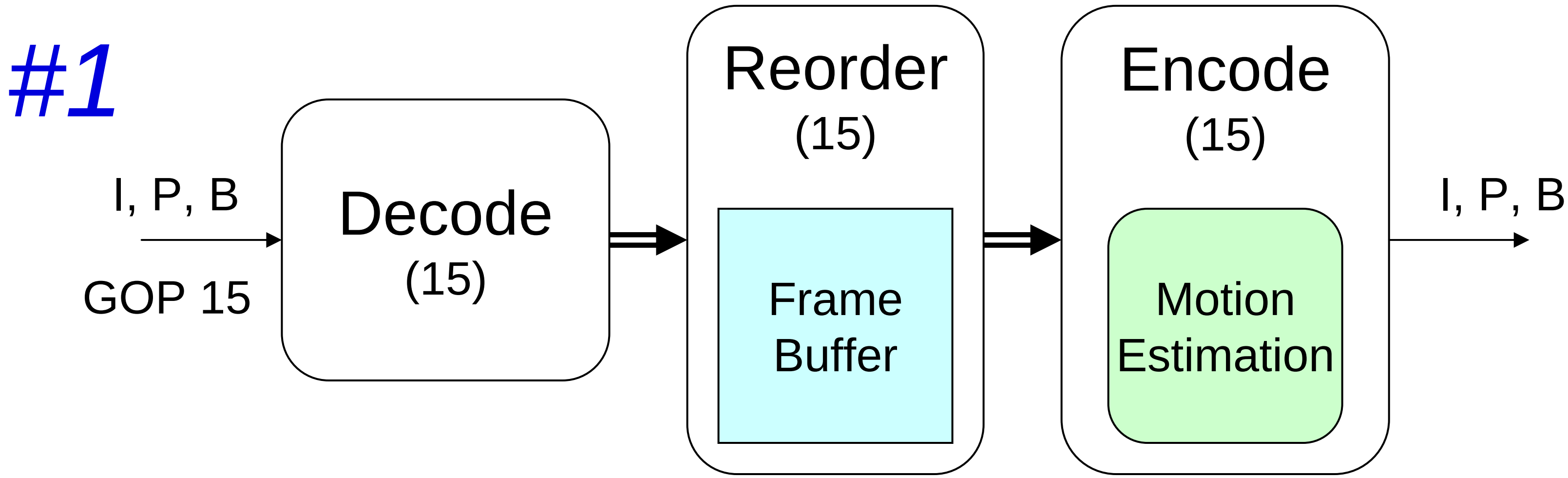
- Background
- Manipulating Temporal Dependencies
- Compressed-Domain Splicing
- ➢ Compressed-Domain Reverse Play
- MPEG2-to-H.263 Transcoding
- Overview and Demo

Reverse-Play Transcoding



Develop computation- and memory-efficient transcoding algorithms for *reverse play* of a given forward-play MPEG video stream.

Reverse-Play Architecture #1



- High memory requirements
  - Frame buffer must store 15 frames
- High computational requirements
  - Motion estimation dominates computational needs

Compressed-Domain Processing

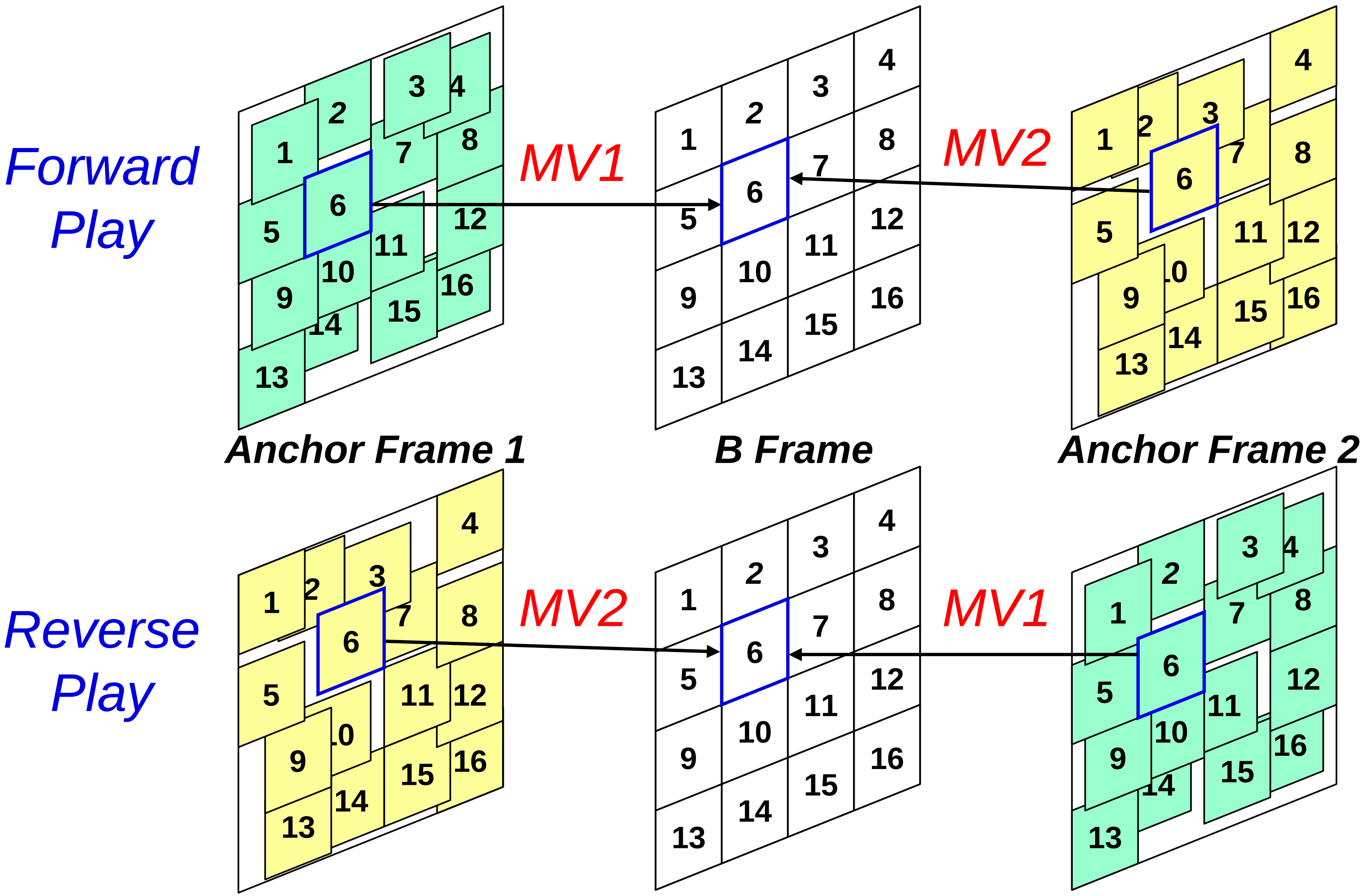
Can we do better by exploiting properties of

1) the MPEG compression algorithm

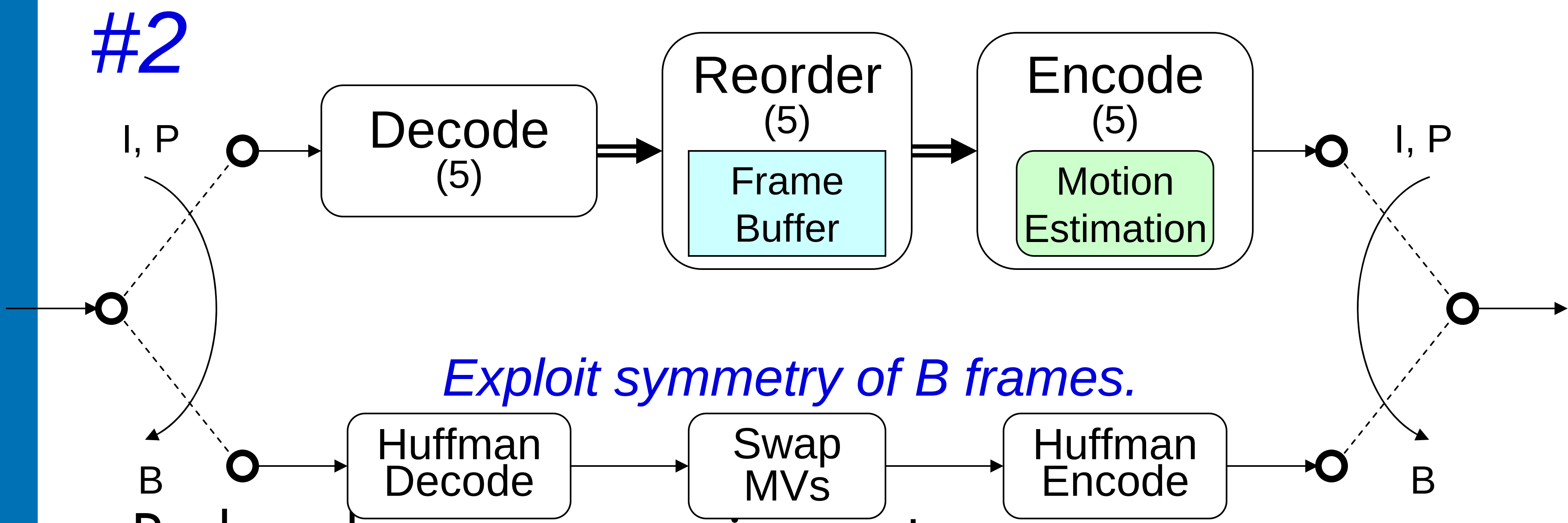
and

2) the reverse-play operation ?

B-Frame Symmetry: Swap MVs

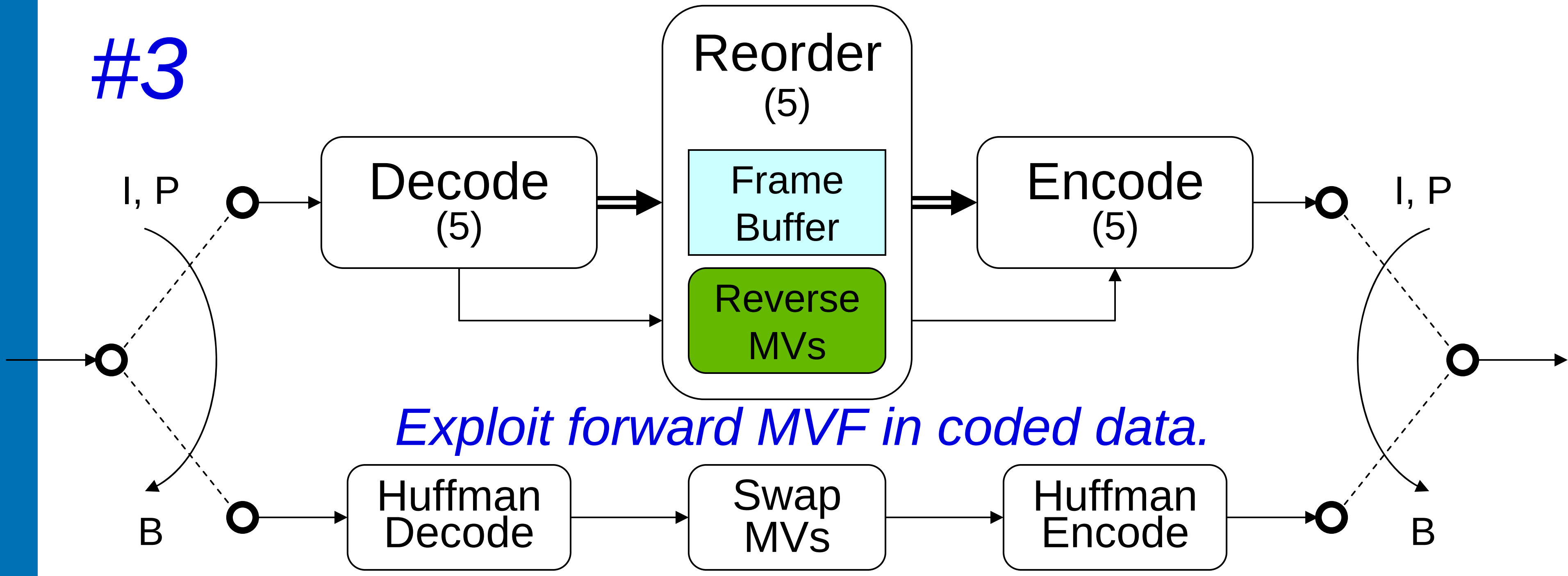


Reverse-Play Architecture #2



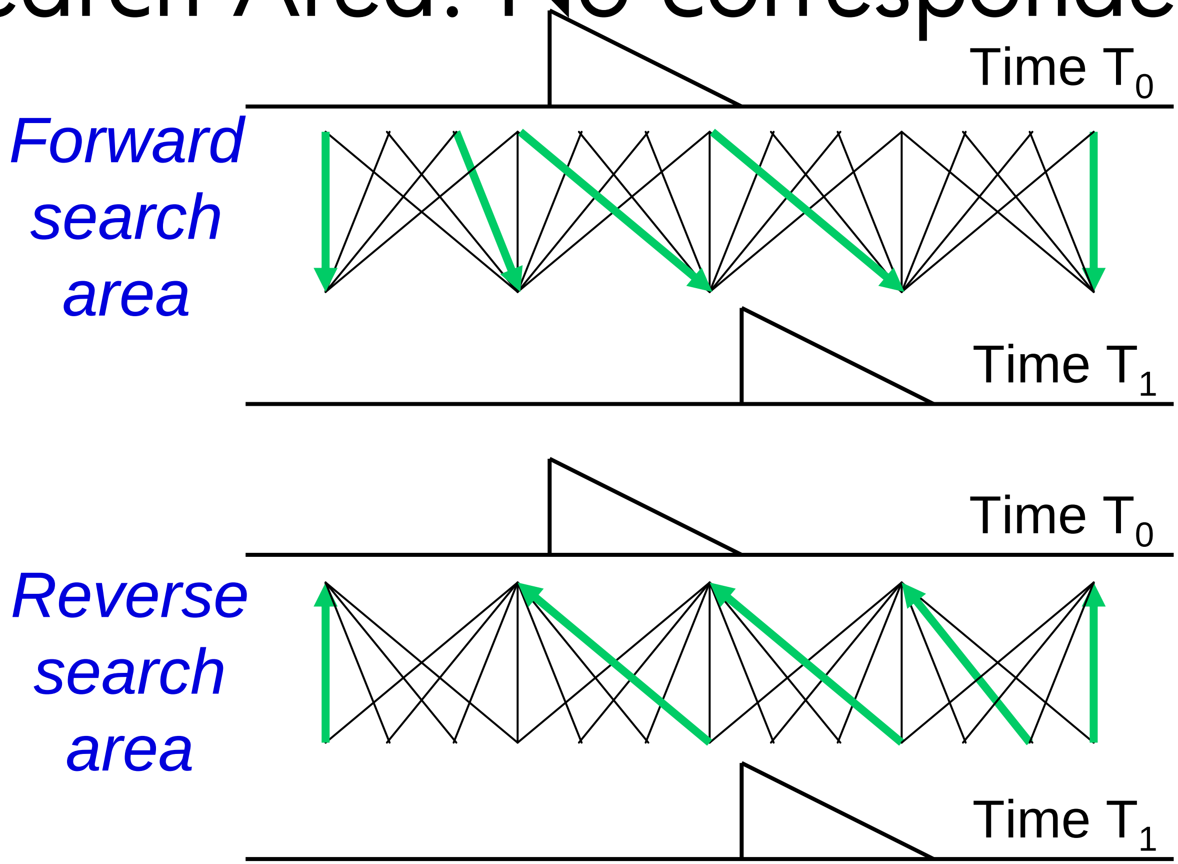
- Reduced memory requirements
  - Frame buffer must store 5 frames
- Reduced computational requirements
  - ME still dominates computational needs

Reverse-Play Architecture #3



- Reduced memory requirements
- Reduced computational requirements

Search Area: No correspondence



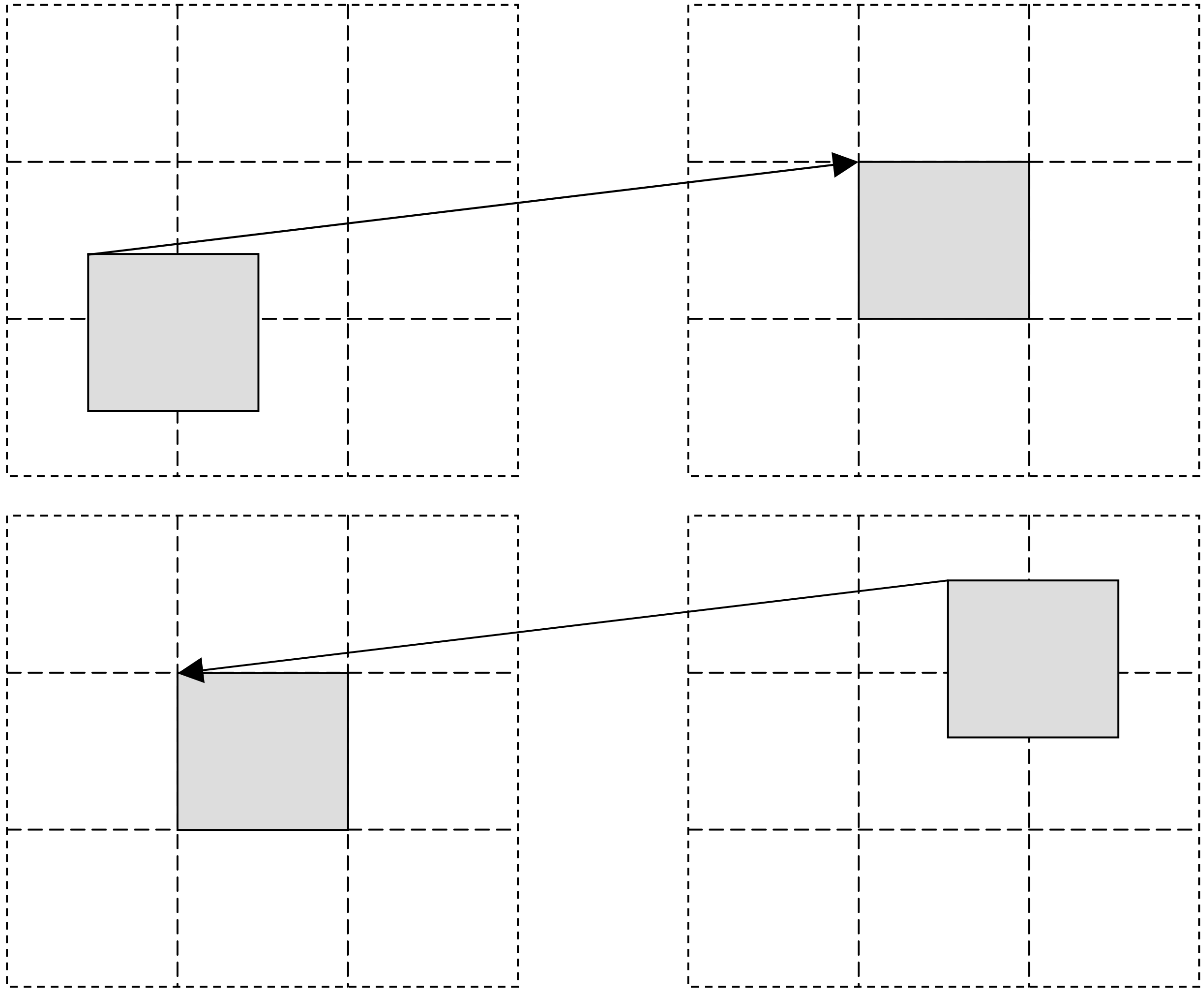
Interpret MVs as specifying a match between blocks.
Develop motion vector resampling methods.

In-place Reversal Method

Forward MV

Reverse MV

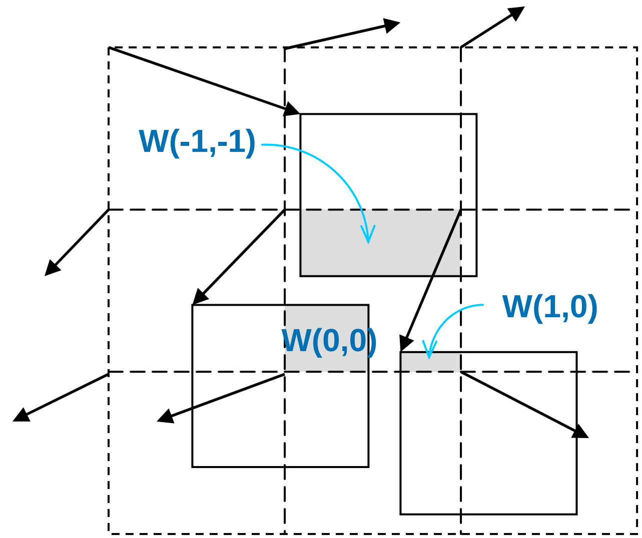
In-place reversal



Maximum Overlap Method

MV(-1,-1) W(-1,-1)	MV(0,-1) W(0,-1)	MV(1,-1) W(1,-1)
MV(-1,0) W(-1,0)	MV(0,0) W(0,0)	MV(1,0) W(1,0)
MV(-1,1) W(-1,1)	MV(0,1) W(0,1)	MV(1,1) W(1,1)

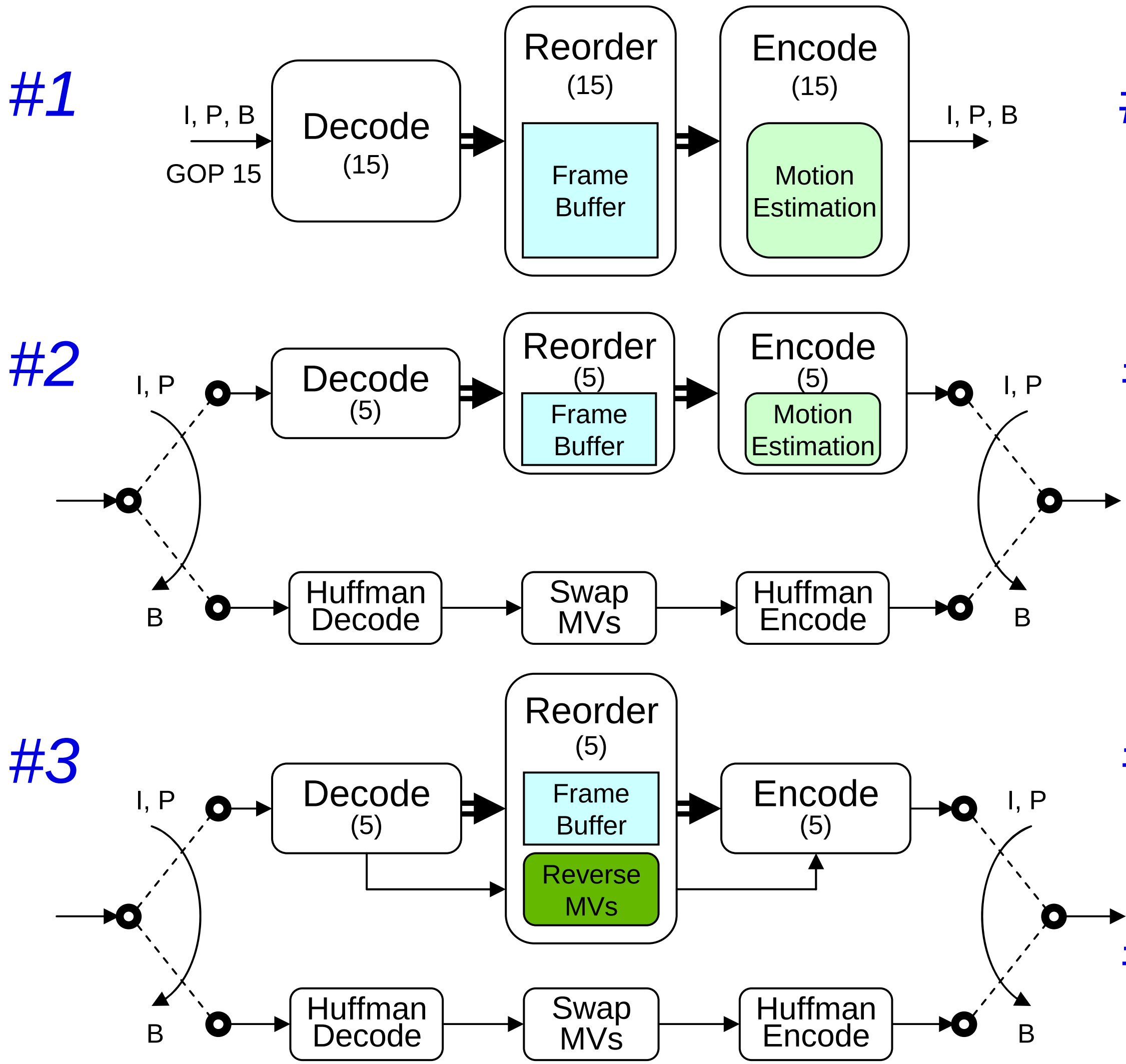
Local neighborhood



Overlap weights

Other MV resampling methods exist.

Computational Requirements



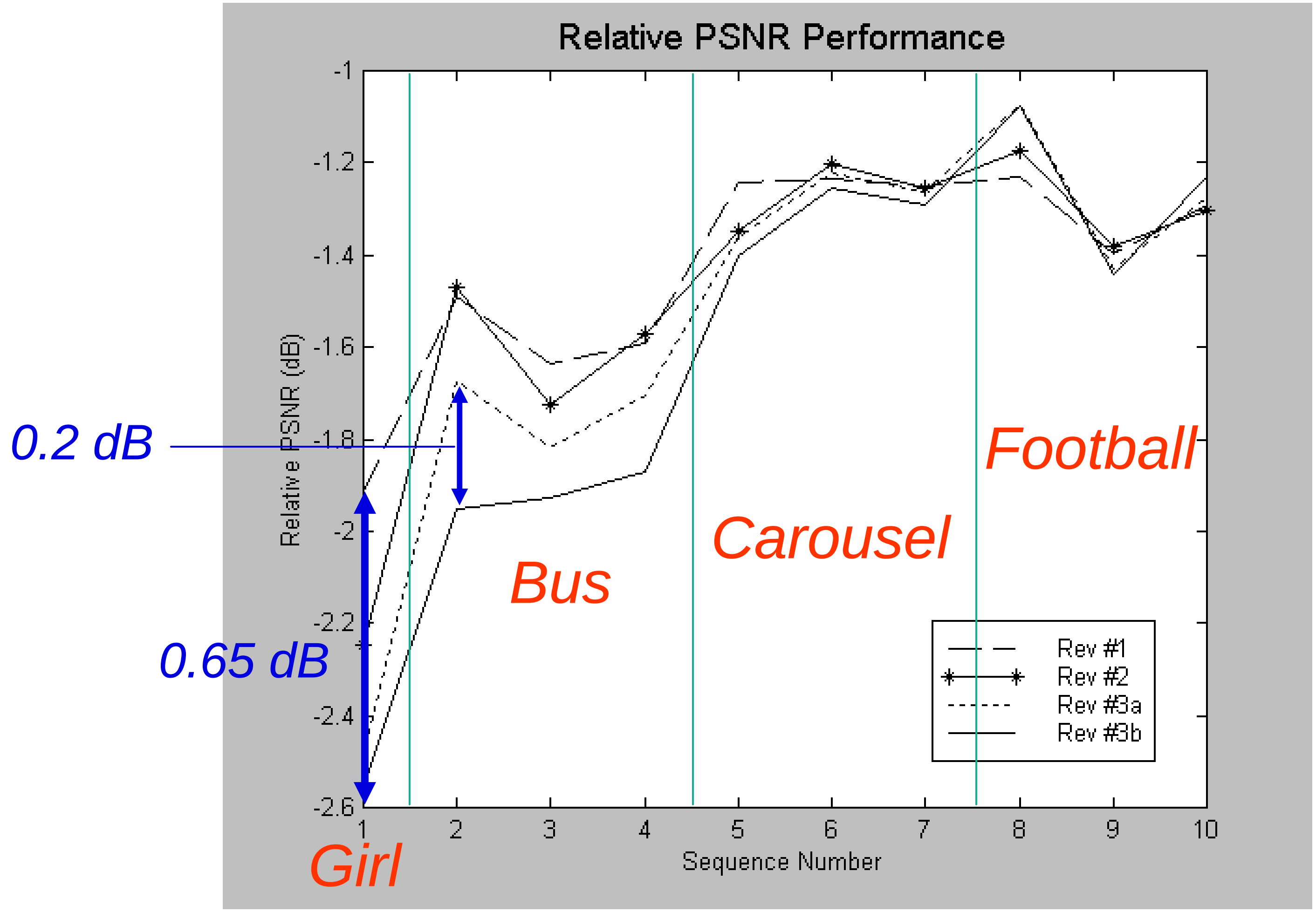
#1: 4200 Mcycles
Log Search ME

#2: 1030 Mcycles
Log Search ME

#3a: 420 Mcycles
Maximum Overlap

#3b: 330 Mcycles
In-place Reverse

Experimental Results



Observations

- Girl sequence showed largest PSNR loss (.65 dB) due to high detail and texture in source.
  - MV accuracy is important!
- Bus sequence benefits from maximum overlap because of large motions in source.
- Carousel has little performance loss because block MC does not match motion in source.
- Football has little performance loss due to blurred source.
  - When MV accuracy is not important, fast approximate methods do not siginificantly sacrifice quality.

Reverse Play Summary

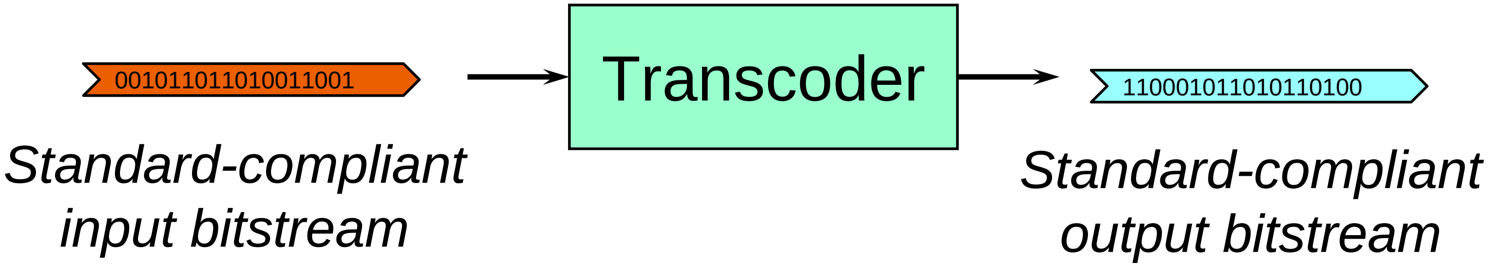
- Developed several compressed-domain reverse-play transcoding algorithms.
- CDP: Exploit properties of compression algorithm and reverse-play operation
  - Exploit symmetry of B frames.
  - Exploit MVF information given in forward stream.
- Order of magnitude reduction in computational requirements.
- Worst case 0.65 dB loss in PSNR quality over baseline transcoding.

Outline

- Background
- Manipulating Temporal Dependencies
- Compressed-Domain Splicing
- Compressed-Domain Reverse Play
- ➢ MPEG2-to-H.263 Transcoding
- Overview and Demo

Motivation

- Video communication requires the seamless delivery of video content to users with a broad range of bandwidth and resource constraints.
- However, the source signal, communication channel, and client device may be incompatible.
- Therefore, efficient transcoding algorithms must be designed



MPEG2-to-H.263 Transcoding

Media Server
DTV or DVD content

Low-bandwidth wireless link

MPEG-to-H.263 Transcoder

- Transcode MPEG video streams to lower-rate H.263 video streams.
- Interlace-to-progressive conversion.
- Order of magnitude reduction in computational requirements.

Stream DVD movies to portable multimedia devices.

Stream DTV program material over the internet.

Problem Statement

Develop a fast transcoding algorithm that adapts the bitrate, frame rate, spatial resolution, scanning format, and/or coding standard while preserving video quality

MPEG-2 input

- High bitrates
- DVD, Digital TV
- >1.5 Mbps, 30 fps
- Interlaced and progressive

H.263 output

- Low bitrates
- Wireless, internet
- <500 kbps, 10fps
- Progressive

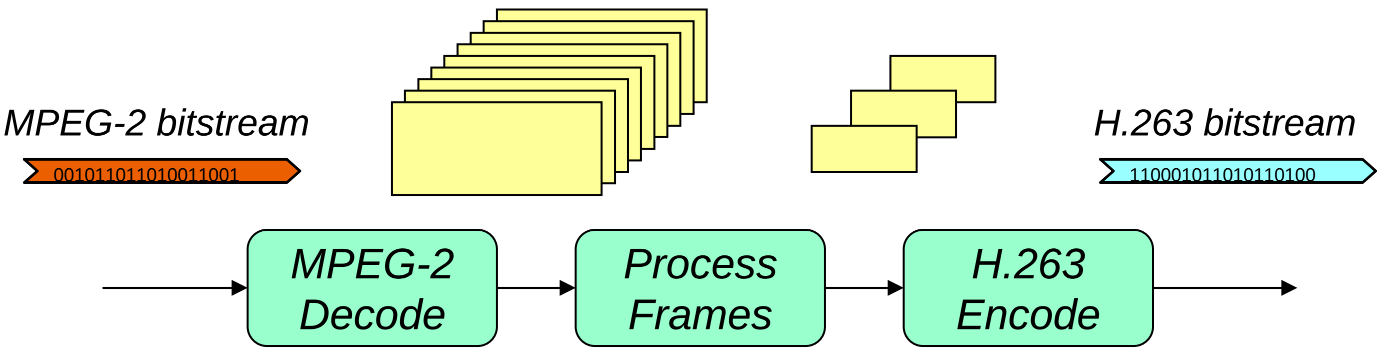
Important features:

- *Interlaced-to-progressive* transcoder.
- *Inter-standard* transcoder, e.g. MPEG-2 to H.263 (or MPEG-4)

Contributors: Wee, Apostolopoulos, Feamster (MIT)

Difficulties

- High cost of conventional transcoding



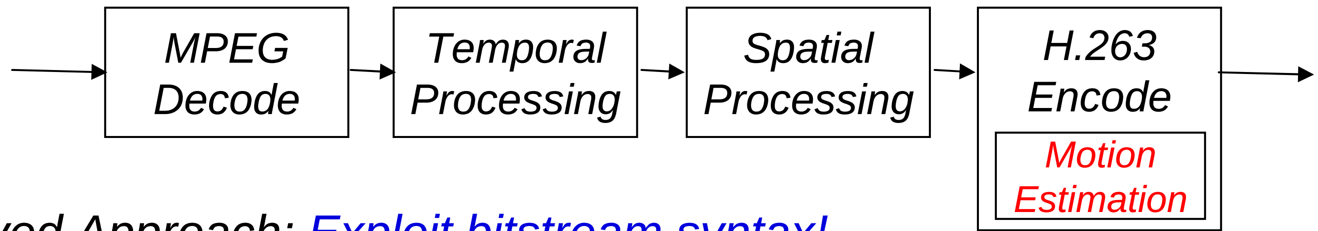
- Differences in video compression standards
- Interlaced vs. progressive formats

Issue: Standard Differences

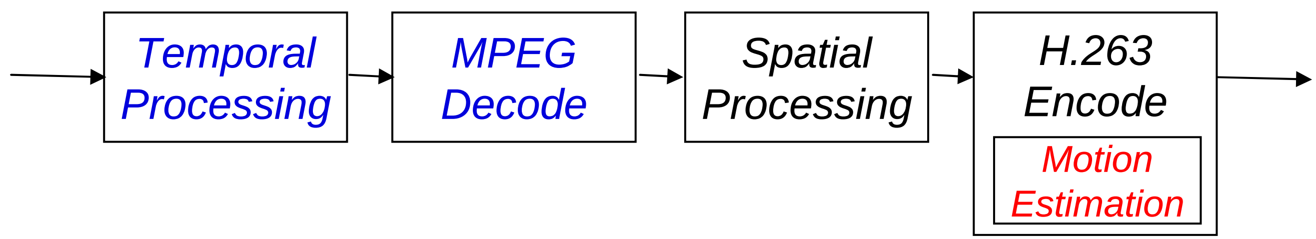
	MPEG-2	**H.263**
Video Formats	Progressive and Interlaced	Progressive Only
I frames	More (enable random access)	Fewer (compression)
Frame Coding Types	I, P, B frames	I, P, Optional PB frames
Prediction Modes	Field, Frame, 16x8	Frame Only
Motion Vectors	Inside Picture Only	Can point outside picture

Development of Proposed Approach

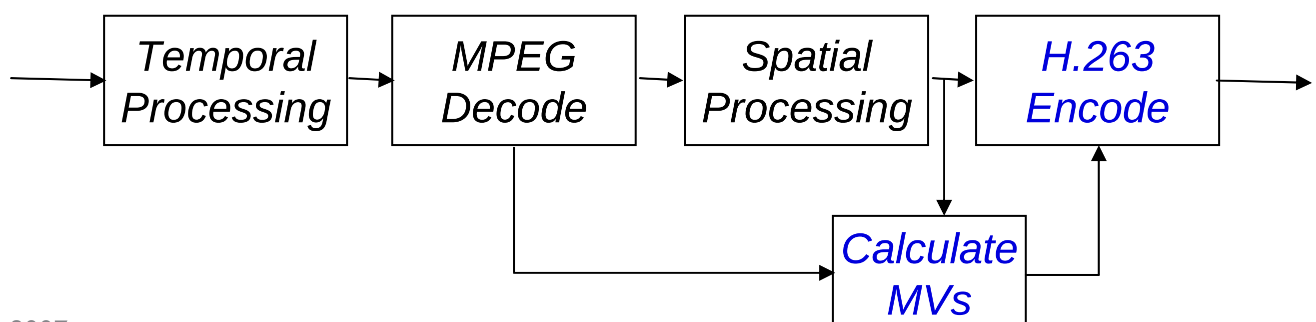
Conventional



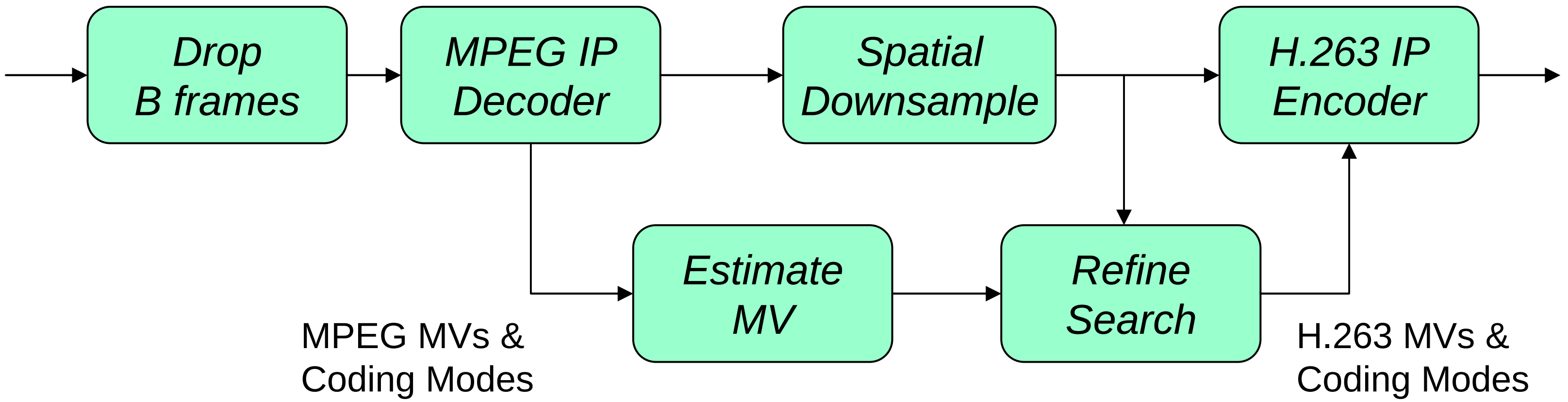
Improved Approach: *Exploit bitstream syntax!*



Proposed Approach: *Also exploit input coded data!*

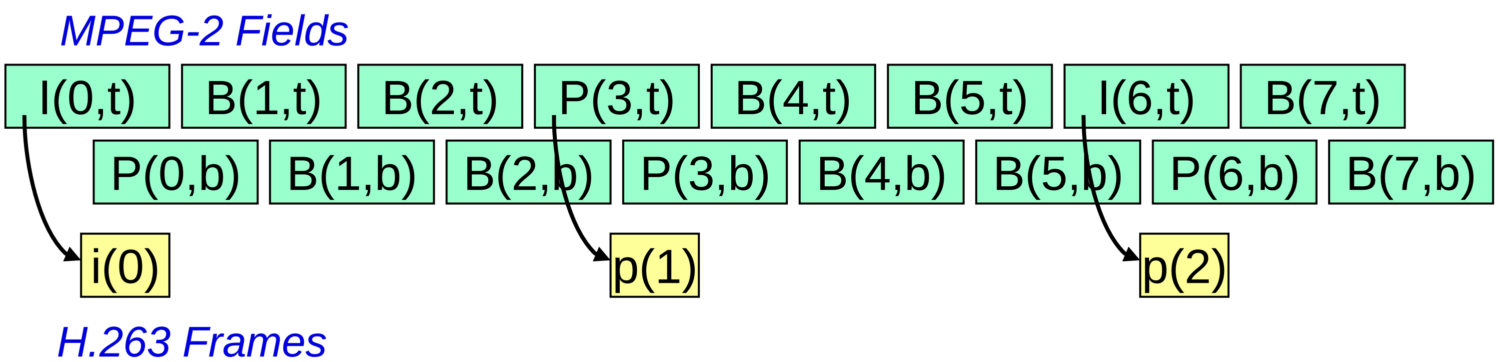


Block Diagram



Issue: Match Prediction Modes

- Match MPEG-2 Fields to H.263 Frames

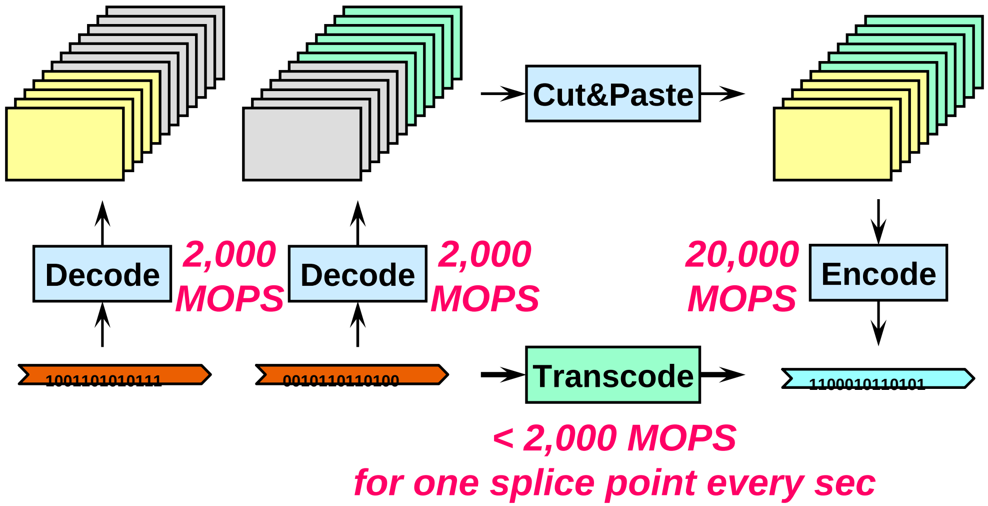


- Vertical downsampling => discard bottom field
- Horizontal downsampling => downsample top field
- Temporal downsampling => drop B frames
- Match MPEG-2 IPPPPIPPPP to H.263 IPPPPPPPPP
  - Problems
    - Convert MPEG-2 P fields to H.263 P frames
    - Convert MPEG-2 I fields to H.263 P frames

Outline

- Background
- Manipulating Temporal Dependencies
- Compressed-Domain Splicing
- Compressed-Domain Reverse Play
- MPEG2-to-H.263 Transcoding
- ➢ Overview and Demo

Compressed-Domain Splicing



Enables Ad Insertion for DTV

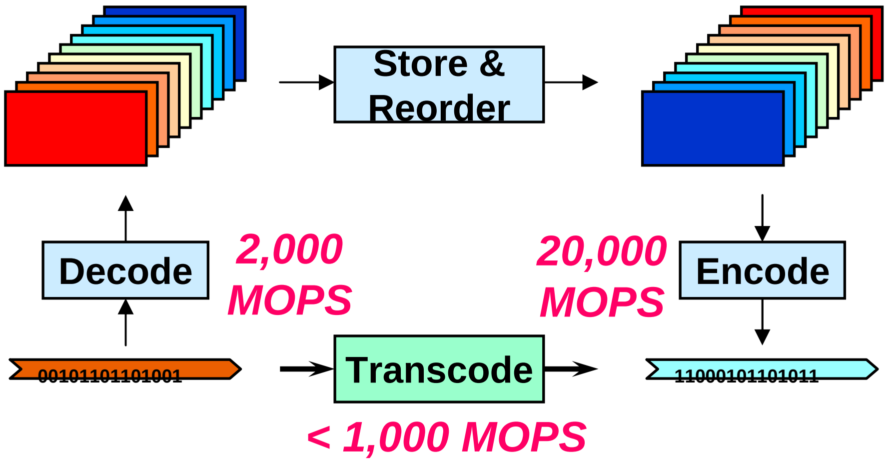
SMPTE splicing solution

- User must define splice points during encoding
  Specialized complex encoders.
- Restricted splice points

HP splicing solution

- Works with any MPEG stream.
- Frame-accurate splice points

Reverse-Play Transcoding



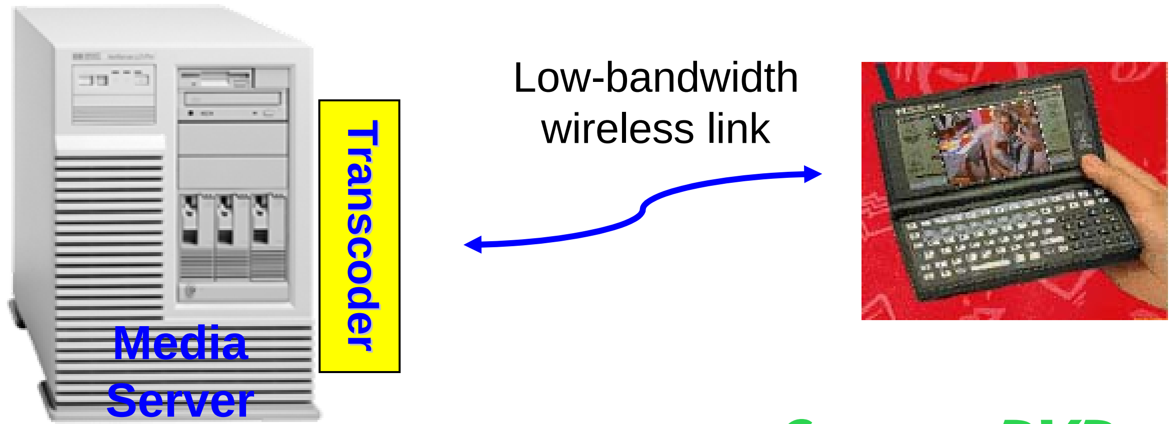
VCR functionality for DTV and Set Tops

<u>HP reverse-play solution</u>

- Frame-by-frame reverse play.
- Works with any MPEG stream.
- Order of magnitude reduction in computational requirements.

MPEG2-to-H.263 Transcoding

Media Server
DTV or DVD content



Stream DVD movies to portable multimedia devices.

Stream DTV program material over the internet.

MPEG-to-H.263 Transcoder

- Transcode MPEG video streams to lower-rate H.263 video streams.
- Interlace-to-progressive conversion.
- Order of magnitude reduction in computational requirements.

References

References:

- “Manipulating Temporal Dependencies in Compressed Video Data with Applications to Compressed-Domain Processing of MPEG Video”, S. Wee, ICASSP 1999.
- “Splicing MPEG Video Streams in the Compressed Domain”, S. Wee, B. Vasudev, IEEE Workshop on Multimedia Signal Processing, 1997.
- “Compressed-Domain Reverse Play of MPEG Video Streams”, S. Wee, B. Vasudev, SPIE Inter. Sym. On Voice, Video, and Data Communications, 1998.
- “Reversing Motion Vector Fields, S.J. Wee, IEEE International Conference on Image Processing, October 1998.
- “Field-to-frame Transcoding with Spatial and Temporal Downsampling”, S. Wee, J. Apostolopoulos, N. Feamster, ICIP 1999.
- “Compressed-Domain Video Processing”, S. Wee, B. Shen, J. Apostolopoulos, HP Labs Technical Report, 2002, CRC Handbook of Video Databases, 2004.

  http://www.hpl.hp.com/techreports/2002/HPL-2002-282.html

invent