A GPU-BASED DVC TO H.264/AVC TRANSCODER

Alberto Corrales-García, Rafael Rodríguez-Sánchez, José Luis Martínez, Gerardo Fernández-Escribano, José M. Claver and José Luis Sánchez

Advanced Communication and Computer Architectures

UNIVERSIDAD POLITECNICA DE VALENCIA
UCLM UNIVERSIDAD DE CASTILLA-LA MANCHA
UNIVERSIDAD DE MURCIA
VNIVERSITAT DE VALÈNCIA

GOBIERNO DE ESPAÑA MINISTERIO DE CIENCIA E INNOVACIÓN
PROGRAMA INGENIO 2010 Consolider

Overview

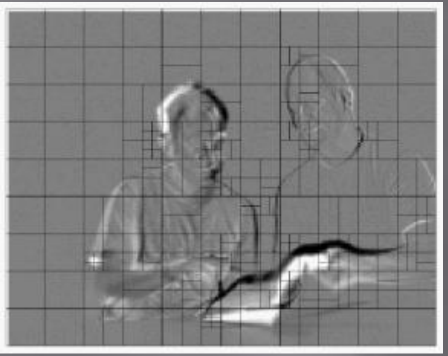
1. Introduction
2. Technical Background
3. Proposed DVC to H.264/AVC GPU-based Video Transcoder
4. Results
5. Conclusions

Overview

1. Introduction
2. Technical Background
3. Proposed DVC to H.264/AVC GPU-based Video Transcoder
4. Results
5. Conclusions

1. Introduction

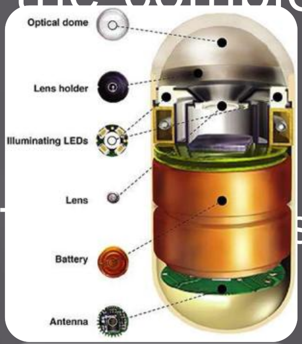
- Traditional coding: encoders more complex than decoders.



- Tipical scene: digital television.

1. Introduction

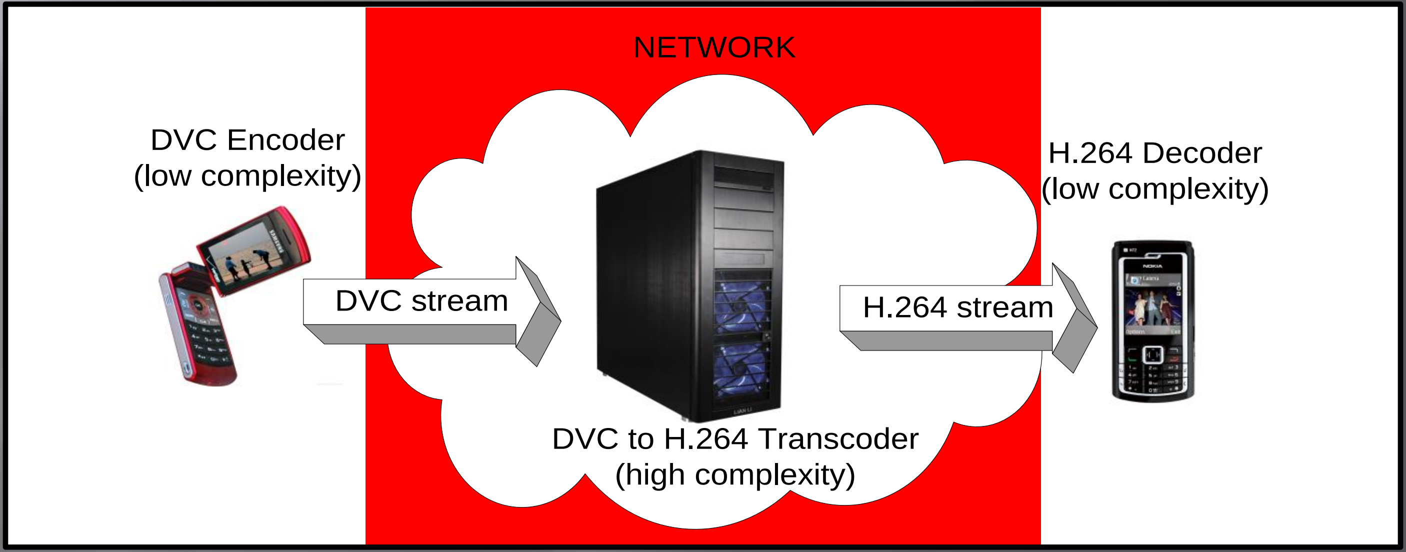
- Recently, new applications have been introduced. These applications have few resources.
- Examples: Surveillance Wireless Networks, Sensors Networks, Micro cameras , Mobile Devices or PDAs.
  - Devices which need low consumption and low complexity
- Main idea of *Distributed Video Coding (DVC)*: the complexity is dis ncoder t oder
  - The emporal
  - Son ansmitted coder. s are esti r side :



ble frames (Side Information) + Correlation Model + Parity Bits

1. Introduction

- Traditional coding : H.264
  - Encoder with high complexity
  - Decoder with low complexity
- DVC:
  - Encoder with low complexity
  - Decoder with high complexity
- Idea:
  - Taking advance of the low complexity of both paradigms for mobile-to-mobile video communications
- Disadvantage:
  - The transcoder joins the highest complexity of both paradigms

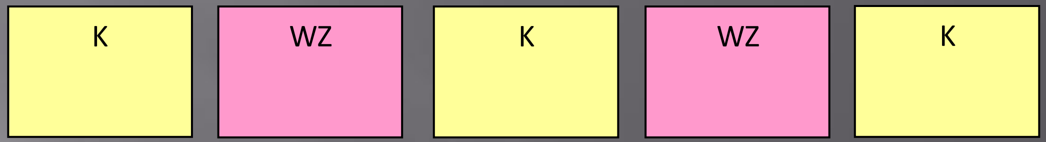


Overview

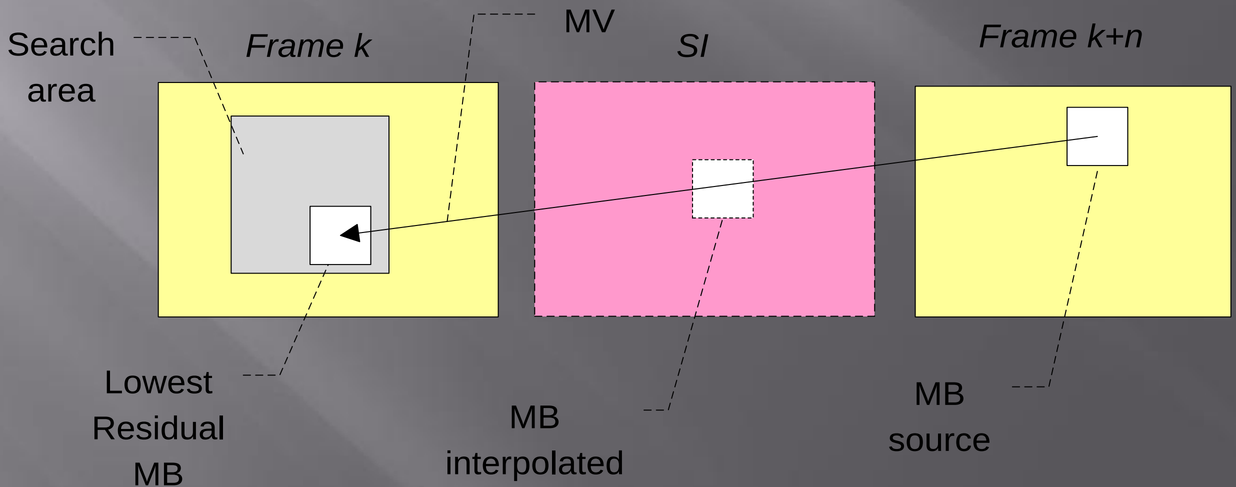
1. Introduction
2. Technical Background
3. Proposed DVC to H.264/AVC GPU-based Video Transcoder
4. Results
5. Conclusions

2. Technical Background

- DVC encoder:
  - Key Frames
  - Wyner-Ziv frames →Parity Bits

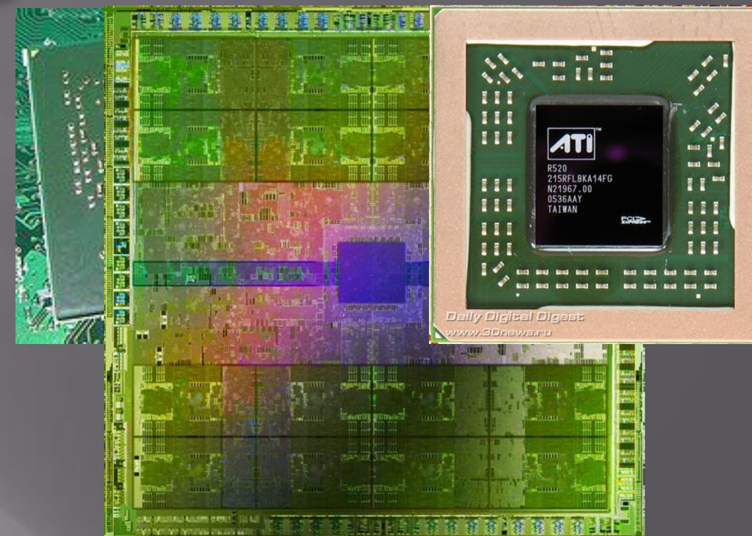


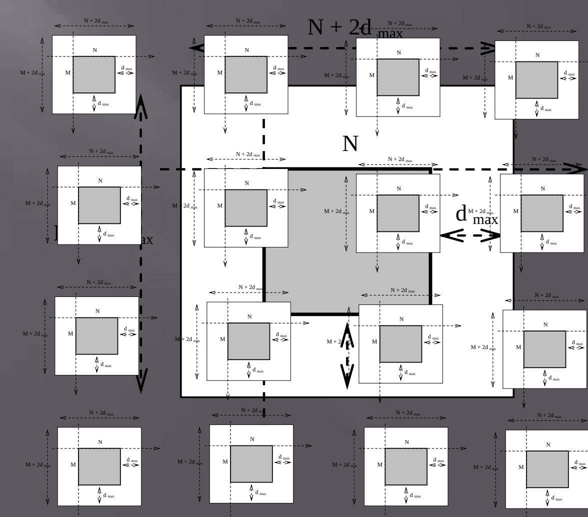
- DVC decoder:
  - Side Information → Motion Vectors (MVs)



2. Technical Background

- A Graphic Processing Unit, or GPU, is a specialized processor that was originally designed for offloading 3D graphics rendering from the microprocessor.
- Recently, nVIDIA © has developed a powerful GPU architecture denominated Compute Unified Device Architecture (CUDA), which is accessible to software developers through industry standard programming languages, such us C, Python, Fortran, Java and Matlab.
- Hence, the ME algorithm developed in the H.264/AVC encoding algorithm fits well in the GPU philosophy, and offers us a new challenge for the GPUs.
- The main goal is how to efficiently distribute all the computations over the GPU and how DVC can improve the process.

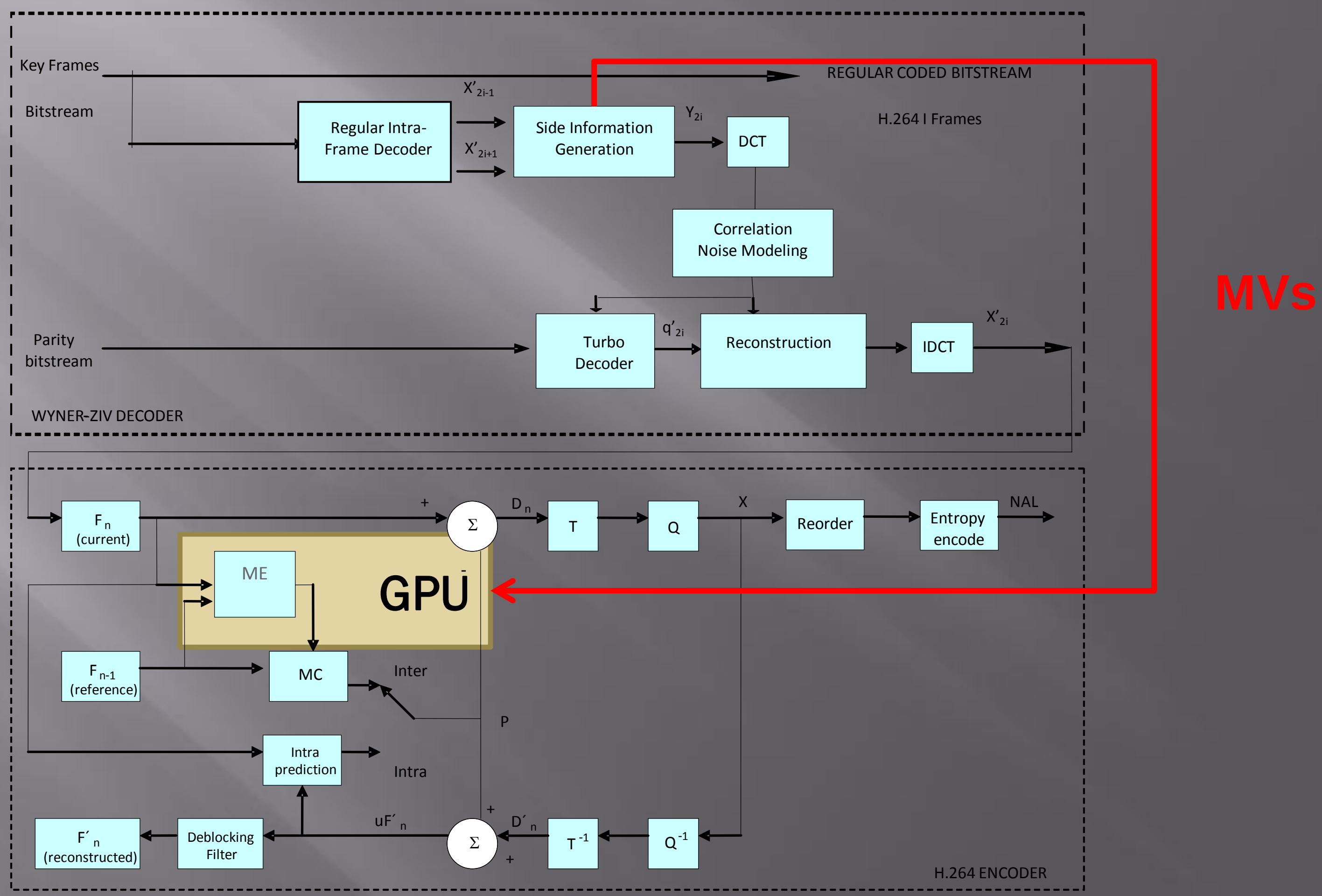




Overview

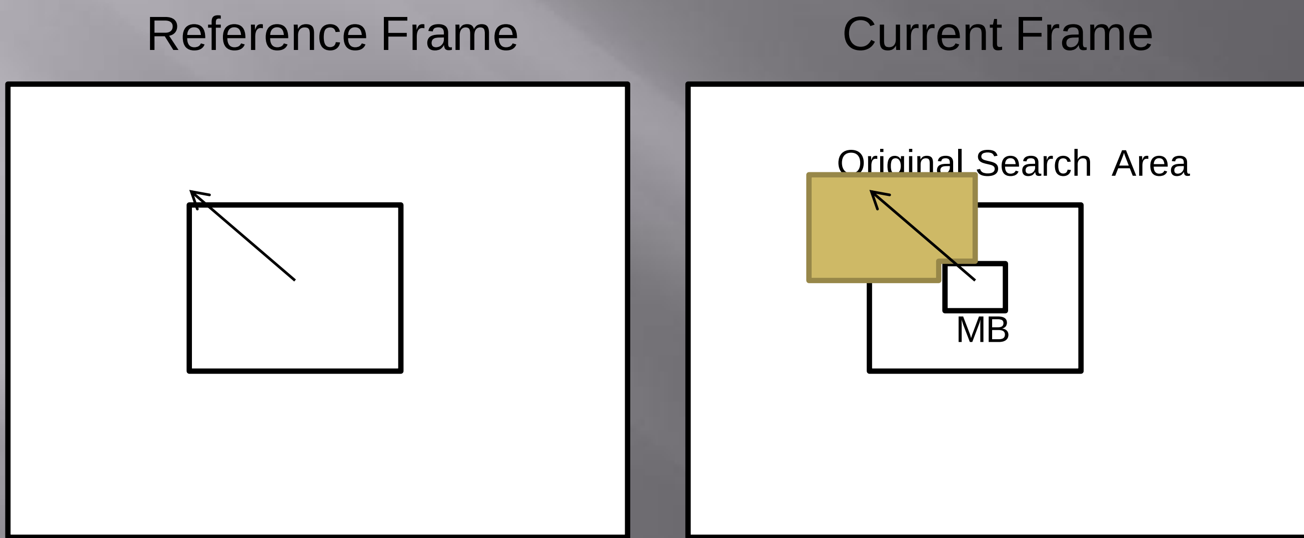
1. Introduction
2. Technical Background
3. Proposed DVC to H.264/AVC GPU-based Video Transcoder
4. Results
5. Conclusions

3. Proposed DVC to H.264/AVC GPU-based Video Transcoder



3. Proposed DVC to H.264/AVC GPU-based Video Transcoder

- H.264 Motion Estimation



Search area

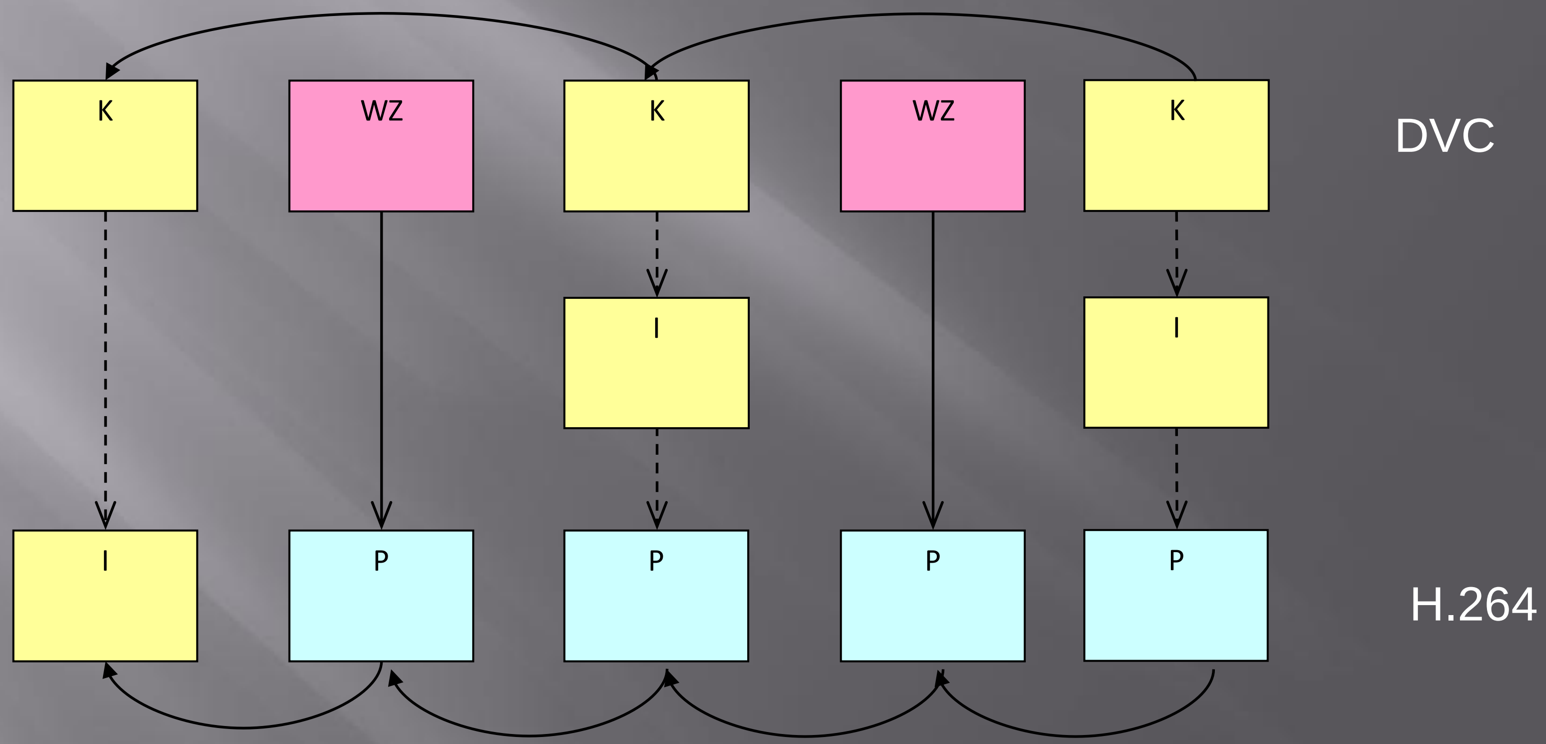
Predictors

Predictors are replaced by DVC MVs

1. *Sum Absolute Differences* (SAD) calculation between the current MB (split into sixteen 4x4 partitions)
2. SAD costs for the different sub-partitions
3. SAD reduction cost to one SAD cost for each one

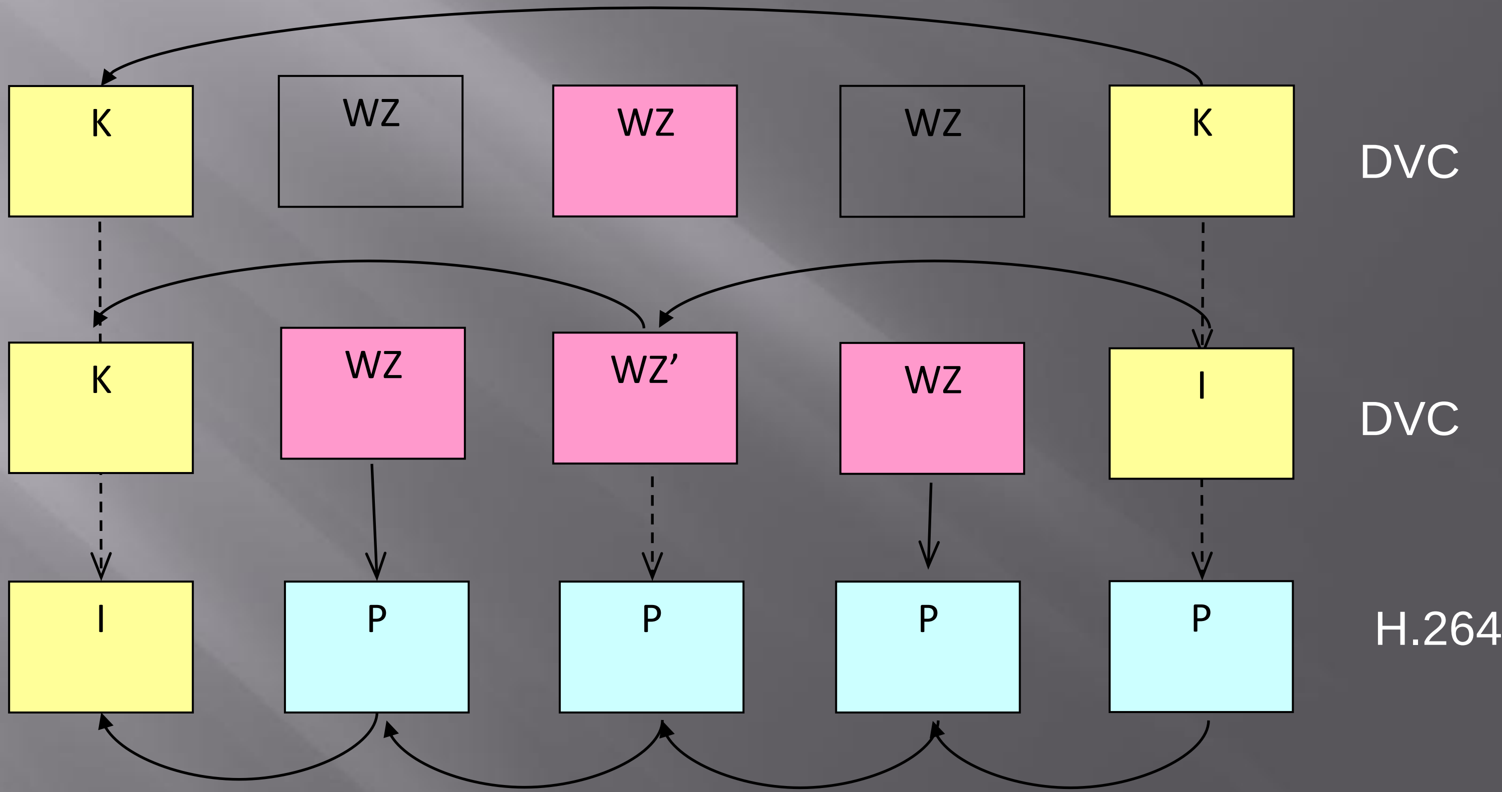
3. Proposed DVC to H.264/AVC GPU-based Video Transcoder

- Mapping Motion Vectors (GOP 2)



3. Proposed DVC to H.264/AVC GPU-based Video Transcoder

- Mapping Motion Vectors (GOP 4)



- The process can be extended for all DVC GOPs

Overview

1. Introduction
2. Technical Background
3. Proposed DVC to H.264/AVC GPU-based Video Transcoder
4. Results
5. Conclusions

4. Results

- Simulation conditions
  - The decoder was implemented with VISNET-II.
  - The encoder was implemented using the H.264 JM reference software (JM15.1). The approach is compared with an unmodified H.264 JM reference software encoder
  - QP values 28, 32, 36 and 40 were use for testing the sequences.
  - GOP format:
    - DVC: 2, 4 and 8
    - H.264: I11(P)

Feature	GTX285
Compute capability	1.3
Global Memory	1 GB
Number of multiprocessors	30
Number of cores	240
Constant memory	64 KB
Shared memory per block	16384
Registers per block	16 KB
Max. active threads per multiprocessor	1024
Clock rate	1.48 GHZ

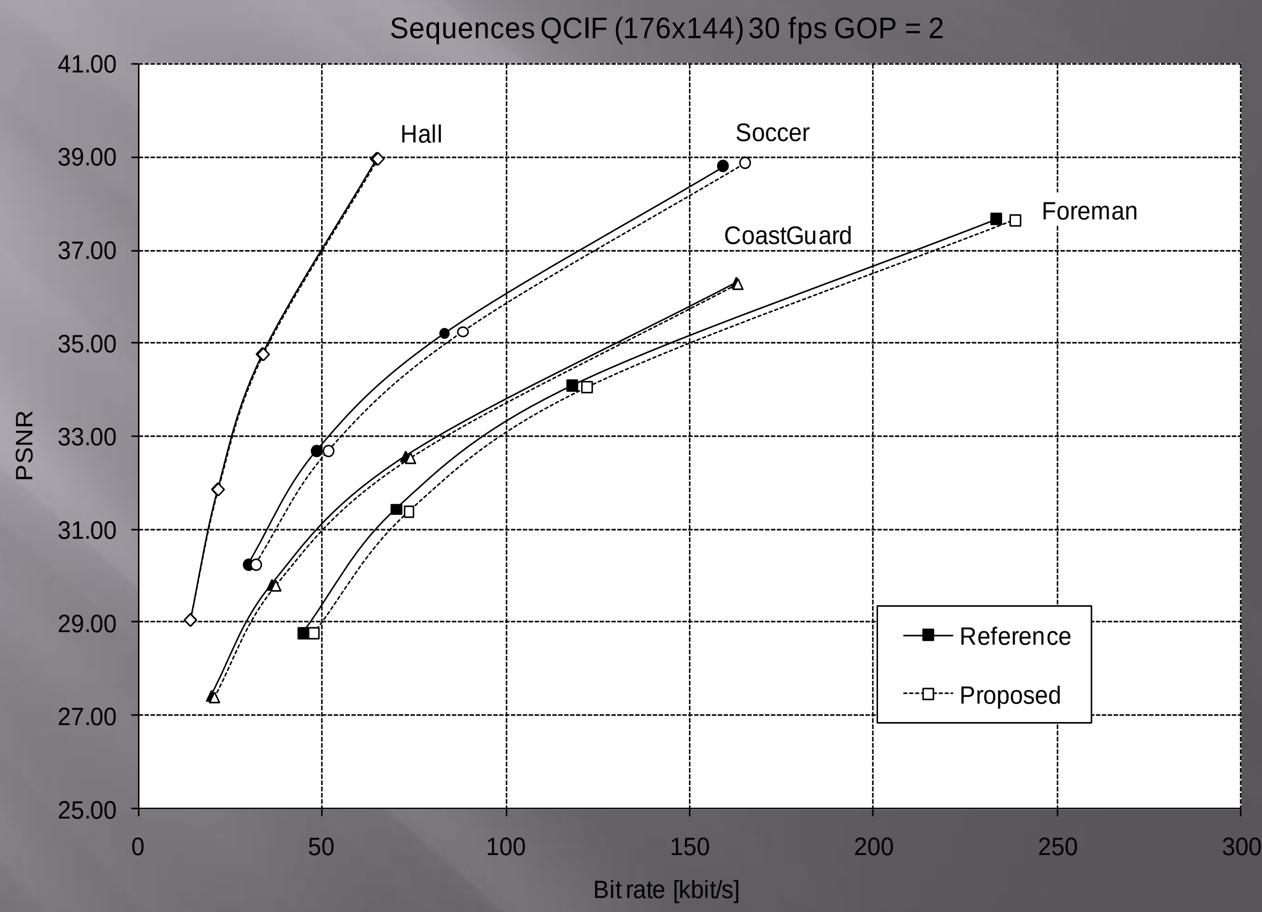
- Tested Sequences: Foreman, CoastGuard, Hall and Soccer at 30 fps

4. Results

RD performance of the WZ/H.264/AVC video transcoder – 30fps					
Sequence	GOP	PSNR (dB)	Bitrate (%)	TR (%)	fps
Foreman	2	-0.191	4.60	80.53	27,12
	4	-0.217	4.80	80.54	26,95
	8	-0.161	4.12	80.88	26,83
Hall	2	-0.055	1.21	72.97	28,96
	4	-0.042	0.92	73.09	28,99
	8	-0.036	0.82	73.18	29,05
Coastguard	2	-0.118	2.95	82.47	27,11
	4	-0.102	2.33	81.90	27,07
	8	-0.105	2.37	81.71	27,13
Soccer	2	-0.216	5.11	81.31	26,57
	4	-0.213	5.26	81.97	26,84
	8	-0.201	5.08	82.11	26,65
mean		**-0.138**	**3.297**	**79.39**	**27,44**

4. Results

- RD performance for QCIF sequences



Overview

1. Introduction
2. Technical Background
3. Proposed DVC to H.264/AVC GPU-based Video Transcoder
4. Results
5. Conclusions

5. Conclusions

- DVC to H.264 transcoding provides a suitable framework to support mobile-to-mobile video communications.
- However, the transcoder has high complexity and it should be reduced.
- Motion Estimation is the most complex task of H.264 and GPUs can help to accelerate it by using MVs generated by DVC.
- Experiments show that the complexity of ME is reduced about a 79% without significant RD penalty.
- This proposal provides a first step in DVC to H.264 parallel GPU-based transcoding.

Thank you

Any question?

DVC: albertocorrales@dsi.uclm.es

GPU/H.264: rrsanchez@dsi.uclm.es