國立清華大學
NATIONAL TSING HUA UNIVERSITY

Multi-Core and GPGPU Acceleration of Video Coding

-- 100x Speedup of Motion Estimation --

Wei-Chih Chen　　Youn-Long Lin

National Tsing Hua University

MPSOC, July 2, 2017
Annecy, France

Outline

- Introduction
- System and Software
- Proposed Methods and Implementations
- Experiment Results
- Conclusion
- Reference

Outline

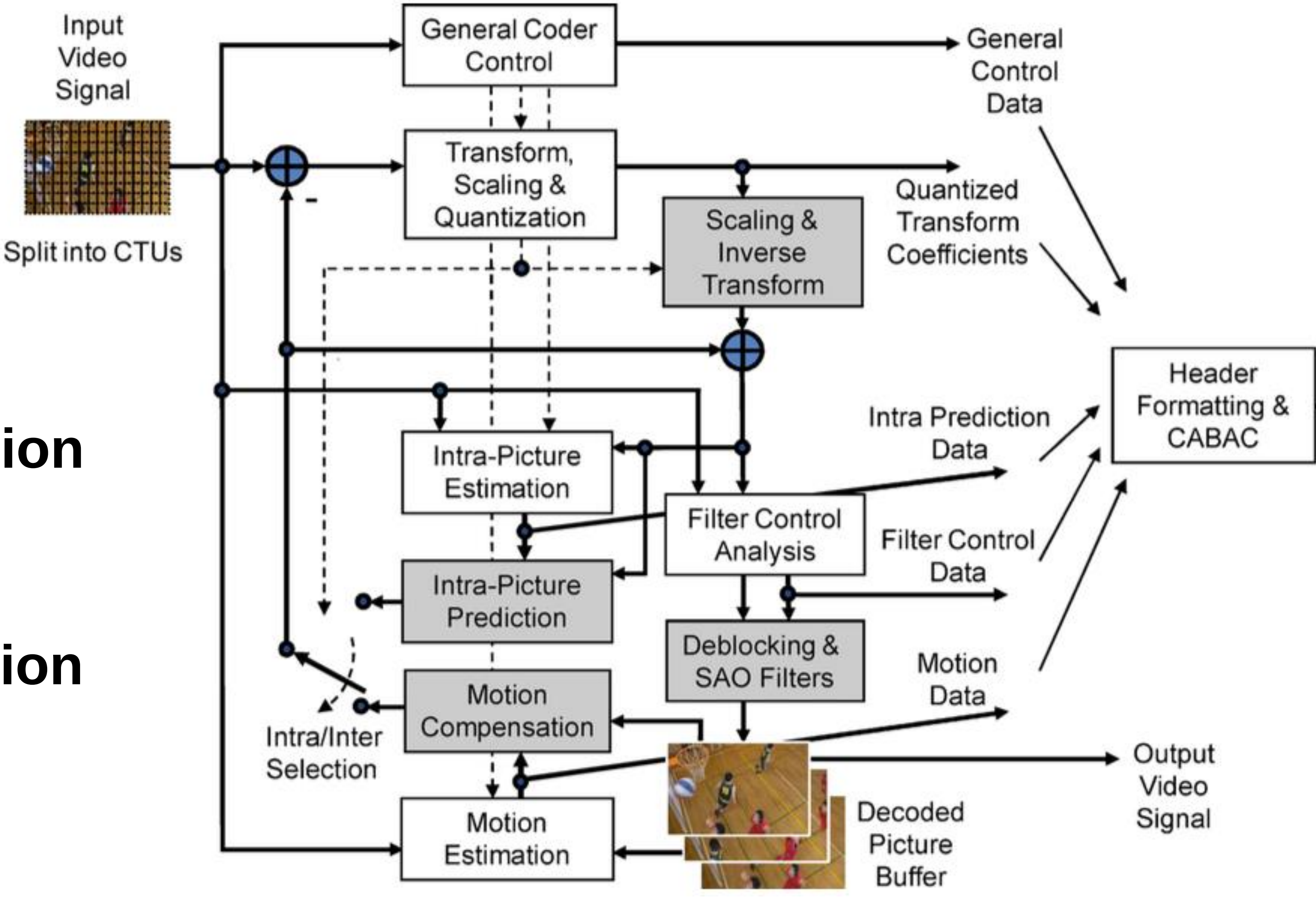
- **Introduction**
- System and Software
- Proposed Methods and Implementations
- Experiment Results
- Conclusion
- Reference

Block-Based Hybrid Coding

- **Contemporary**
- **video coding**
  - Uses the same “hybrid” approach starting from H.261

- **Intra-picture prediction**
  - Spatial correlation

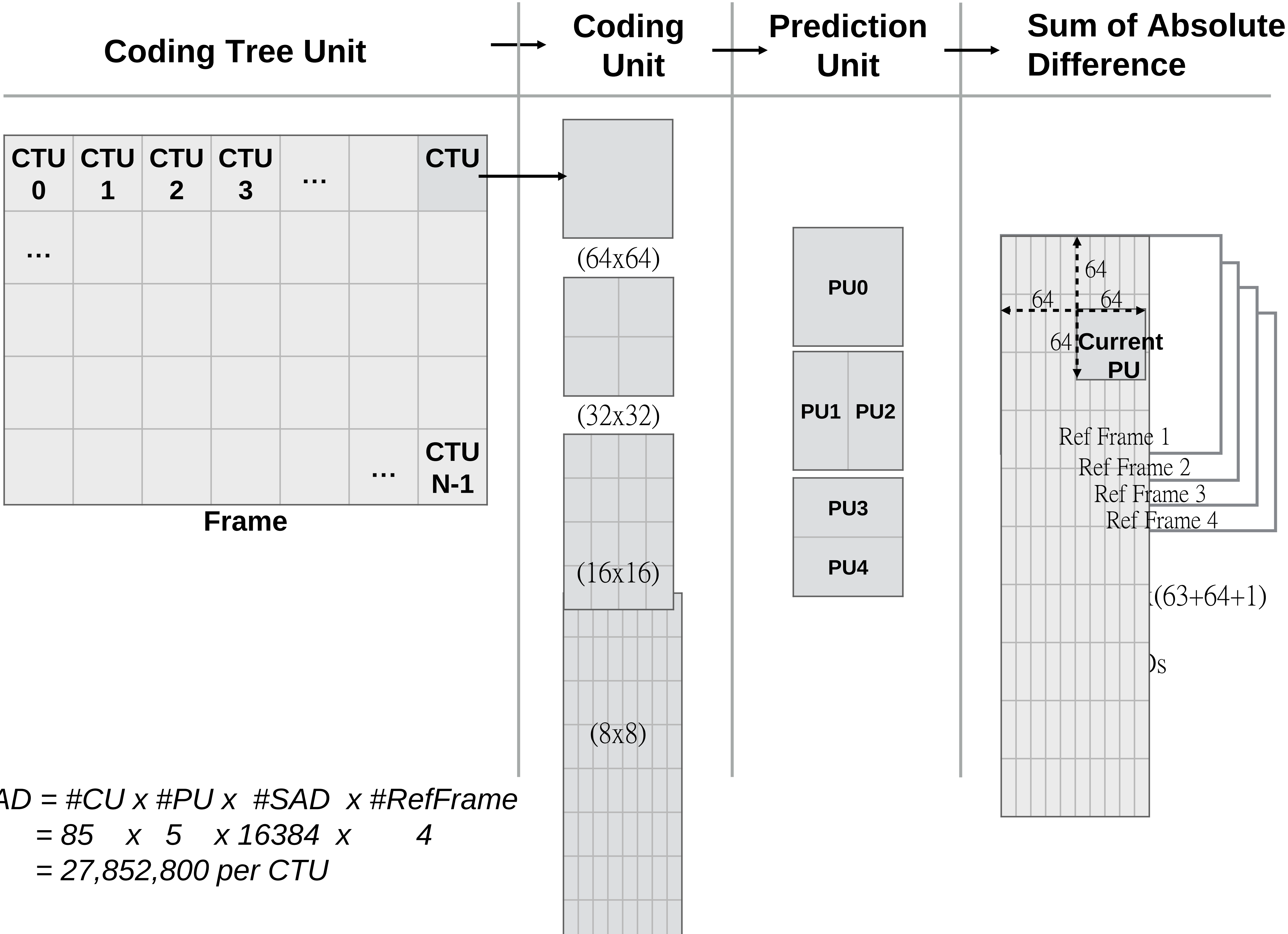
- **Inter-picture prediction**
  - Temporal correlation



HEVC video encoder / decoder.

SOURCE: Overview of the High Efficiency Video Coding (HEVC) Standard [1]

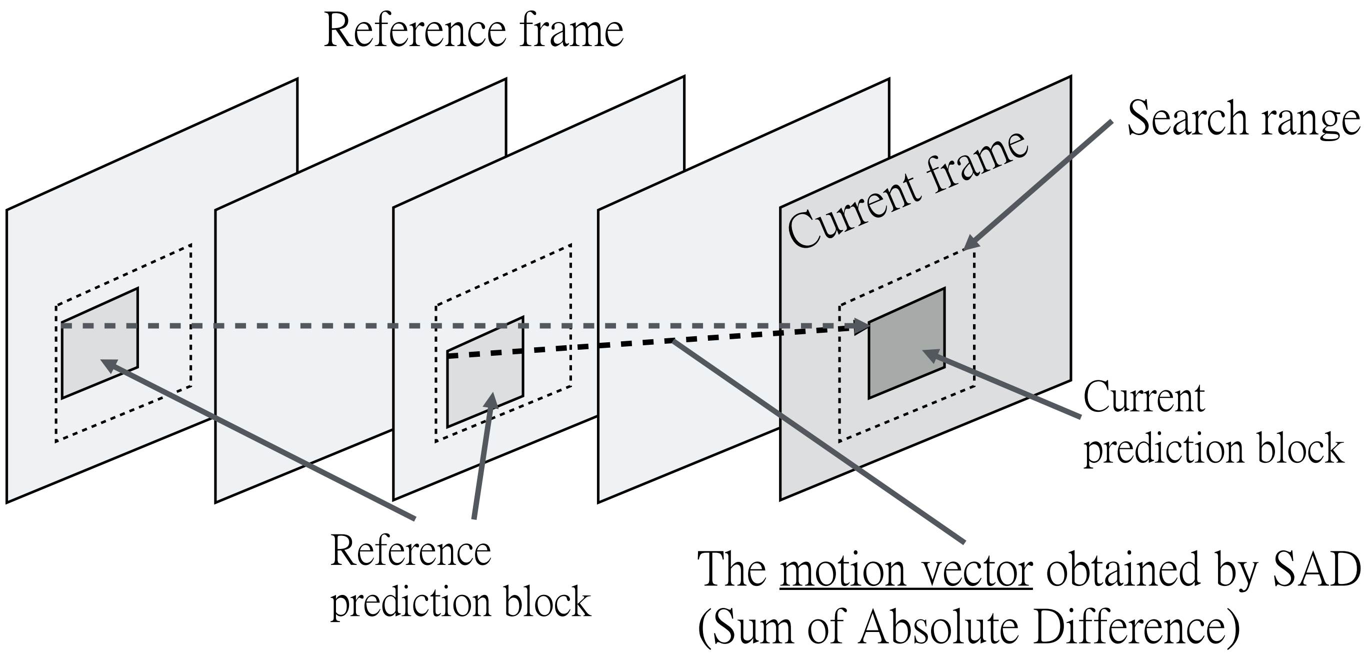
HEVC Coding Unit Structure



Total SAD = #CU x #PU x #SAD x #RefFrame
= 85   x   5   x 16384 x   4
= 27,852,800 per CTU

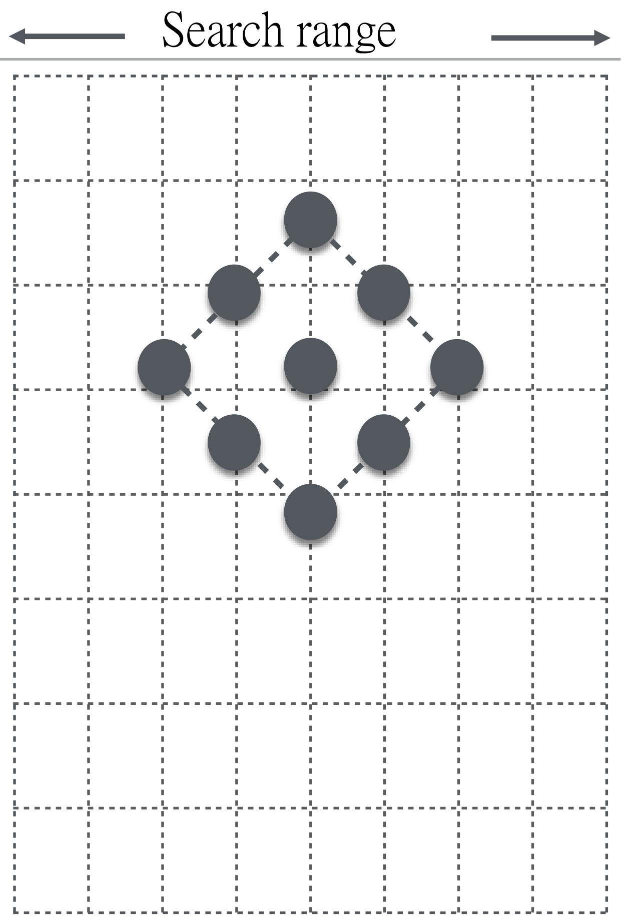
Motion Estimation

- **Multi-Reference Frame Prediction**
  - Uni-predictive ME (P-frame)
  - Bi-predictive ME (B-frame)



Inter-picture prediction

- **Block Matching Motion Estimation**
  - Full search
    - High coding efficiency
    - High computing complexity
  - Fast search
    - Low coding efficiency
    - e.g. Diamond search, TZ-search



Outline

- Introduction
- Related Work
- **System and Software**
- Proposed Methods and Implementations
- Experiment Results
- Conclusion
- Reference

System and Software

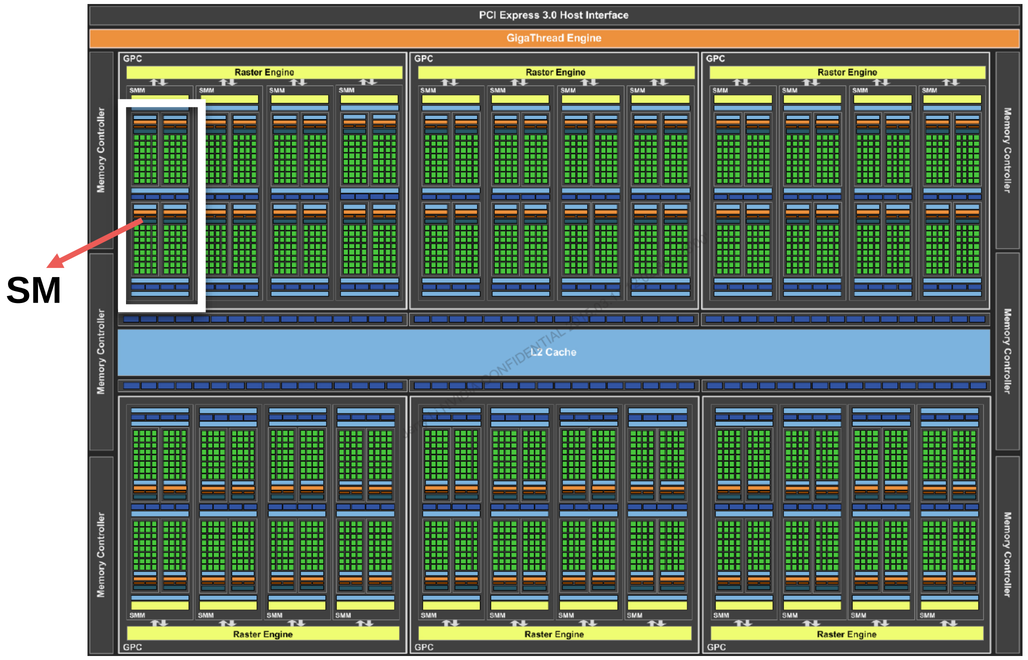
Machine	
CPU	Intel(R) Core(TM) i7-6700K, 4.00GHz, 4 cores with 8 threads
GPU	GeForce GTX **TITAN X Maxwell™** with 3072 CUDA cores, 12GB RAM
Memory	32GB
OS	Linux ubuntu 16.04

GeForce GTX TITAN X	
Architecture	Maxwell
CUDA Capability	5.2
Driver Version	8.0
Runtime Version	8.0
Streaming Multiprocessors	**24**
CUDA Cores	3072 cores
GPU Max. clock rate	1076 MHz
Shared Memory	48KB/CUDA block

HEVC Test Model （reference software）	
Version	16.9
Configure File	encoder_lowdelay_main_P.cfg
Uni-Prediction Frames	**4 (P-Slice)**
Block-Matching Algorithm	**Full Search**
Search Range	**64**
Quantization Parameter (QP)	22, 27, 32, 37
Frame Count	16
Fast Encoder Decision	DISABLE
Asymmetric Motion Partitions	DISABLE
Wavefront Parallel Processing	DISABLE

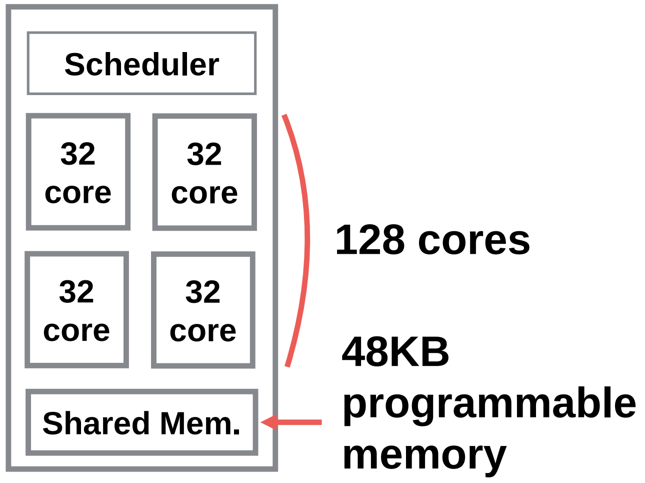
GPGPU

GPU Hardware Architecture (Physical)



- 24 Streaming Multiprocessors(SMs)
- 128 CUDA cores/SM (3072 cores)
- 12GB Global memory

- Streaming Multiprocessor (SM)



Scheduler

32 core

32 core

32 core

32 core

Shared Mem.

128 cores

48KB programmable memory

CUDA Programming Model (Logical)

SIMD_CUDA_Program.cu

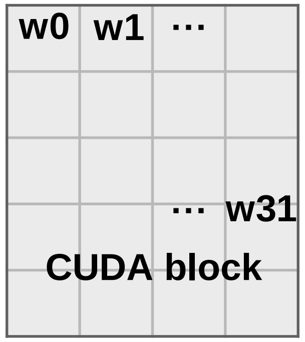
```
kernel<<< #CUDA block, #CUDA thread >>>(…);
{
    const Int Tid = threadIdx.x;
    const Int Bid = blockIdx.x;
    …
}
```

- Max. number of CUDA block: $(2^{31}-1)x(2^{16}-1)x(2^{16}-1)$
- Max. number of threads per CUDA block: 1024

Bid	0	1	2	…				…
Kernel	1024 thread	1024 thread	1024 thread	1024 thread	…			1024 thread

- CUDA Block

- CUDA block can run in any order
- Divide CUDA block into warps



w0 w1 …

… w31

CUDA block

1 warp = 32 threads

Video Sequences

Class	Sequence Name	Resolution	Source
4K	Marathon	3840x2160	The SJTU 4K Video Sequence Sequence Dataset **[5]**
	Wood	3840x2160	
	Runners	3840x2160	
	Library	3840x2160	
A	Traffic	2560x1600	HEVC Standard Standard Test Test Sequences Sequences
	PeopleOnStreetet	2560x1600	
B	BasketballDrivee	1920x1080	
	ParkScene	1920x1080	
	BQTerrace	1920x1080	
	Cactus	1920x1080	
	Kimono1	1920x1080	
	Tennis	1920x1080	

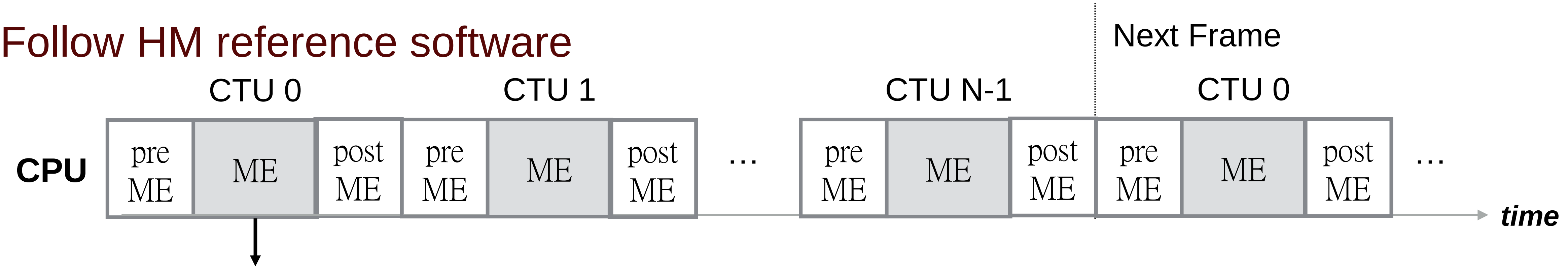
Outline

- Introduction
- System and Application
- **Proposed Methods and Implementations**
  - **Versions 1, 2, 3, 4**
- Experiment Results
- Conclusion
- Reference

Basic Idea

Sequential Execution

- Follow HM reference software



Next Frame

CTU 0 CTU 1 … CTU N-1 | CTU 0 …

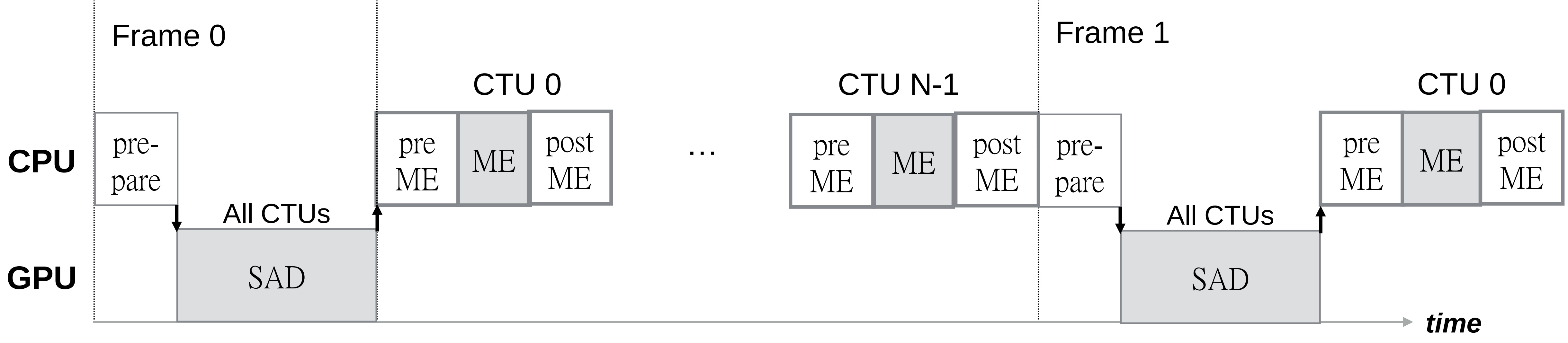
CPU: pre ME | ME | post ME | pre ME | ME | post ME | … | pre ME | ME | post ME | pre ME | ME | post ME | …

time

```
for all CU in a CTU do:
    for all PU in a CU do:
        for all RefFrame do:
            for all Pixel(x, y) in SrchRng do:
                $Distortion = SAD + \lambda_{pred} * (MVD_x + MVD_y)$
```

Proposed Methods

- Offload SAD Calculation to GPU



Frame 0 | Frame 1

CPU: pre-pare | CTU 0: pre ME | ME | post ME | … | CTU N-1: pre ME | ME | post ME | pre-pare | CTU 0: pre ME | ME | post ME

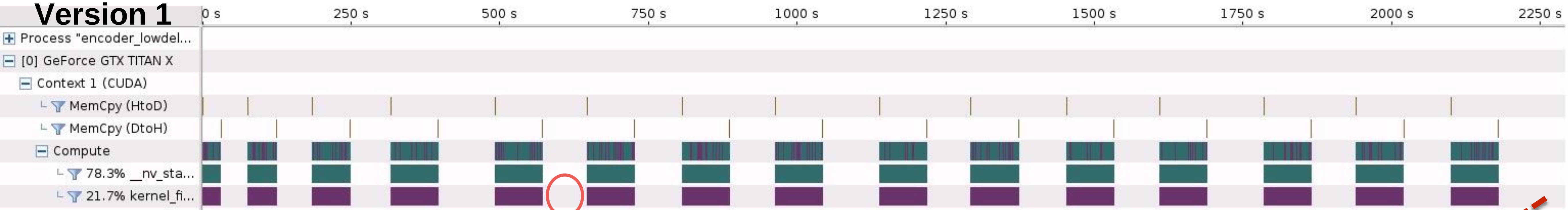
GPU: All CTUs SAD | All CTUs SAD

time

High Utilization by Scheduling and Allocation

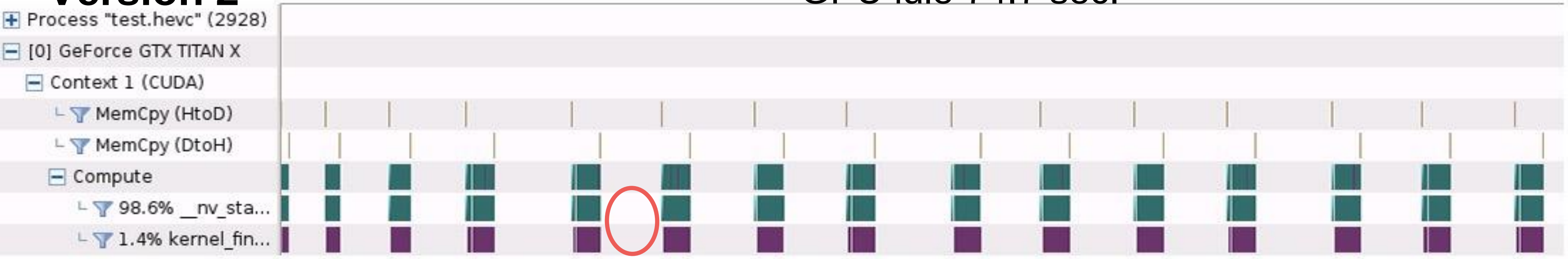
Test Sequence	**4K**: Marathon_3840x2160, **16** frames, QP=32

Version 1



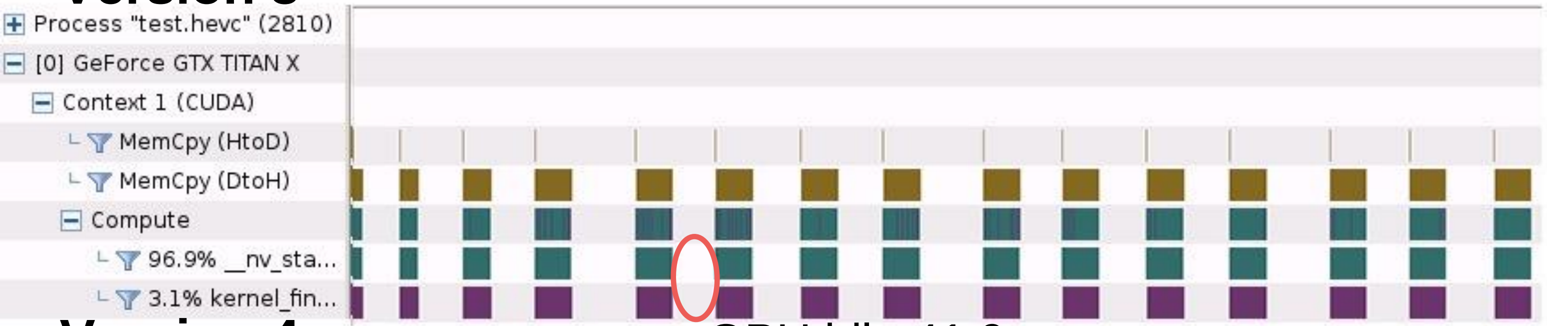
GPU idle 74.7 sec.

Version 2



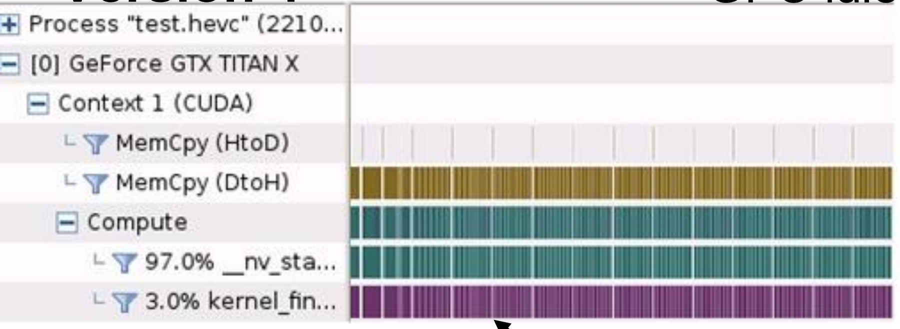
GPU idle 75.3 sec

Version 3



GPU idle 41.9 sec.

Version 4

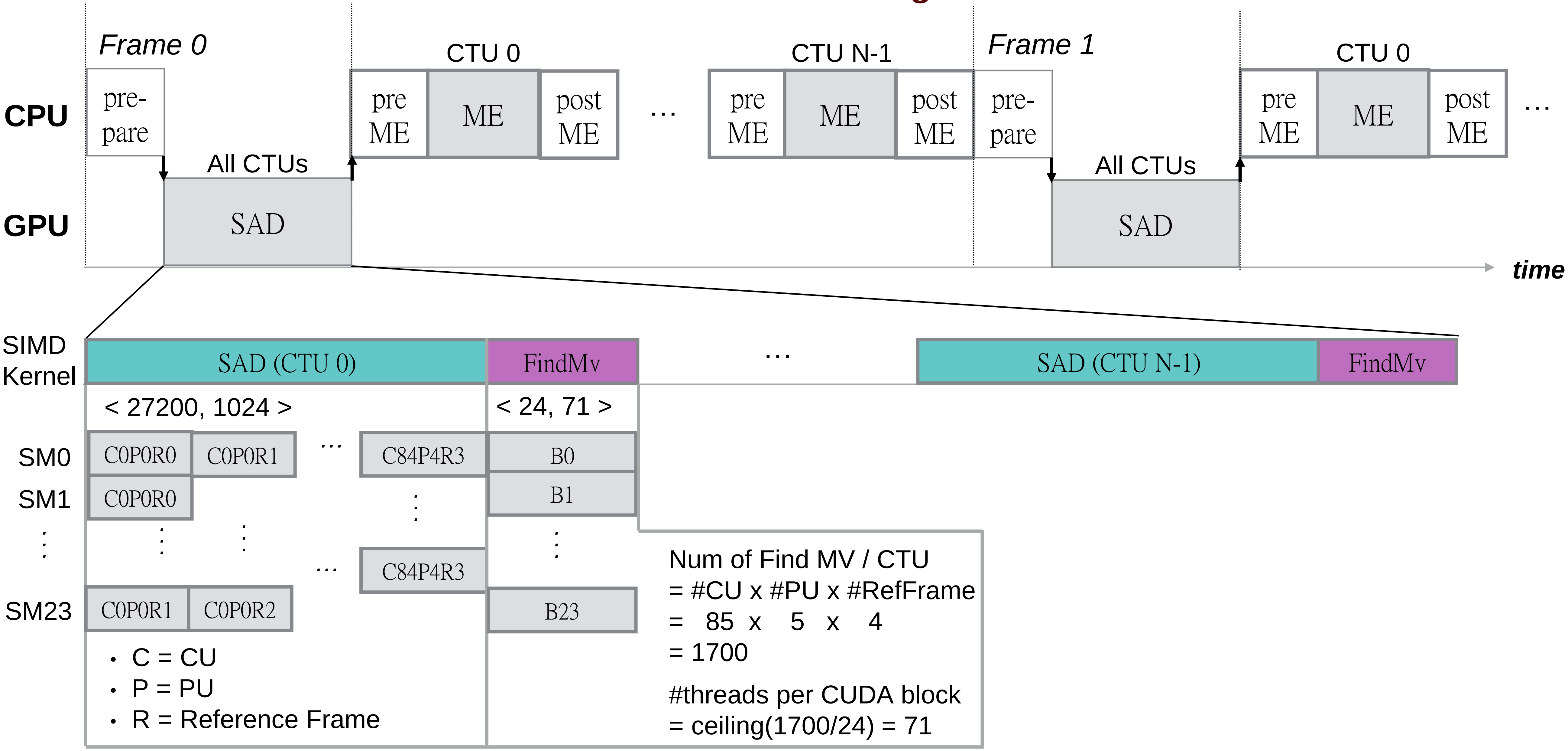


GPU idle 0.79 sec.

Proposed Method Ver. 1

Pixel-level Parallelism

- Create 27,852,800 threads each calculating one SAD



- C = CU
- P = PU
- R = Reference Frame

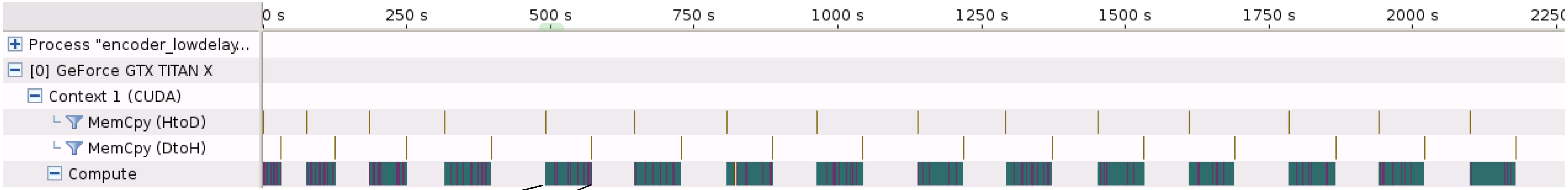
Num of Find MV / CTU
= #CU x #PU x #RefFrame
= 85 x 5 x 4
= 1700

#threads per CUDA block
= ceiling(1700/24) = 71

- Memory of SAD results = 27,852,800 x sizeof(Int) = 111.4 MB per CTU
- For 4K video (2040 CTUs), it's need 227GB
- FindMv kernel to find a best MV for each prediction unit (PU)

Timing Profile of Ver. 1

Test Sequence	**4K**: Marathon_3840x2160, **16** frames, QP=32



0 s 250 s 500 s 750 s 1000 s 1250 s 1500 s 1750 s 2000 s 225(

Process "encoder_lowdelay..

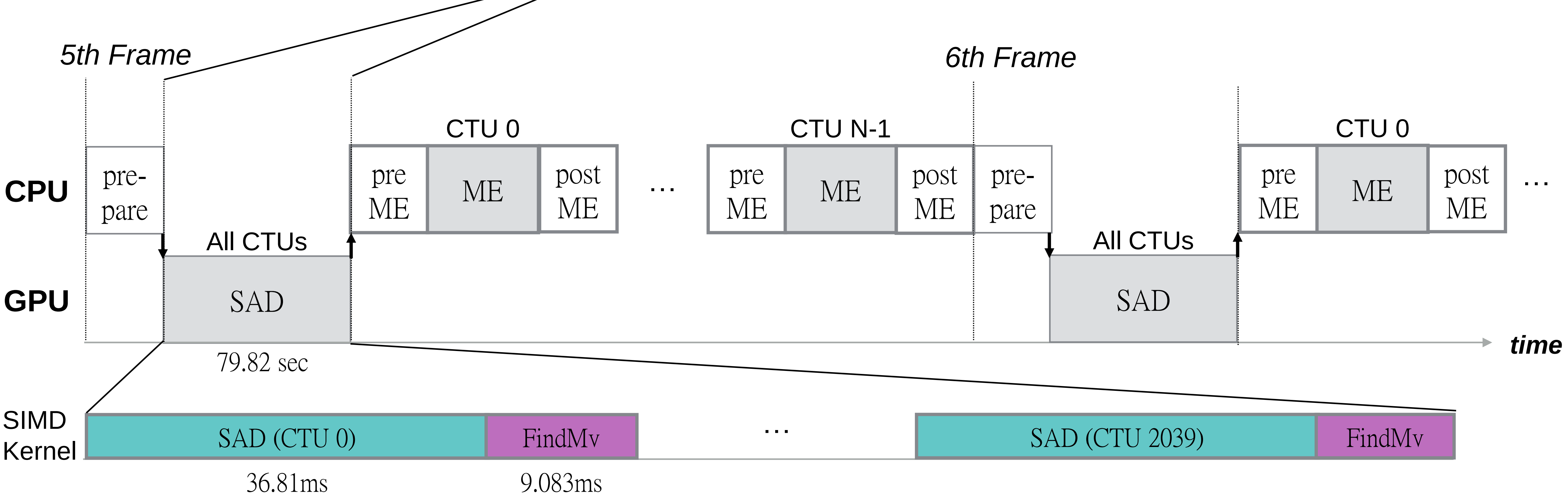
[0] GeForce GTX TITAN X

Context 1 (CUDA)

MemCpy (HtoD)

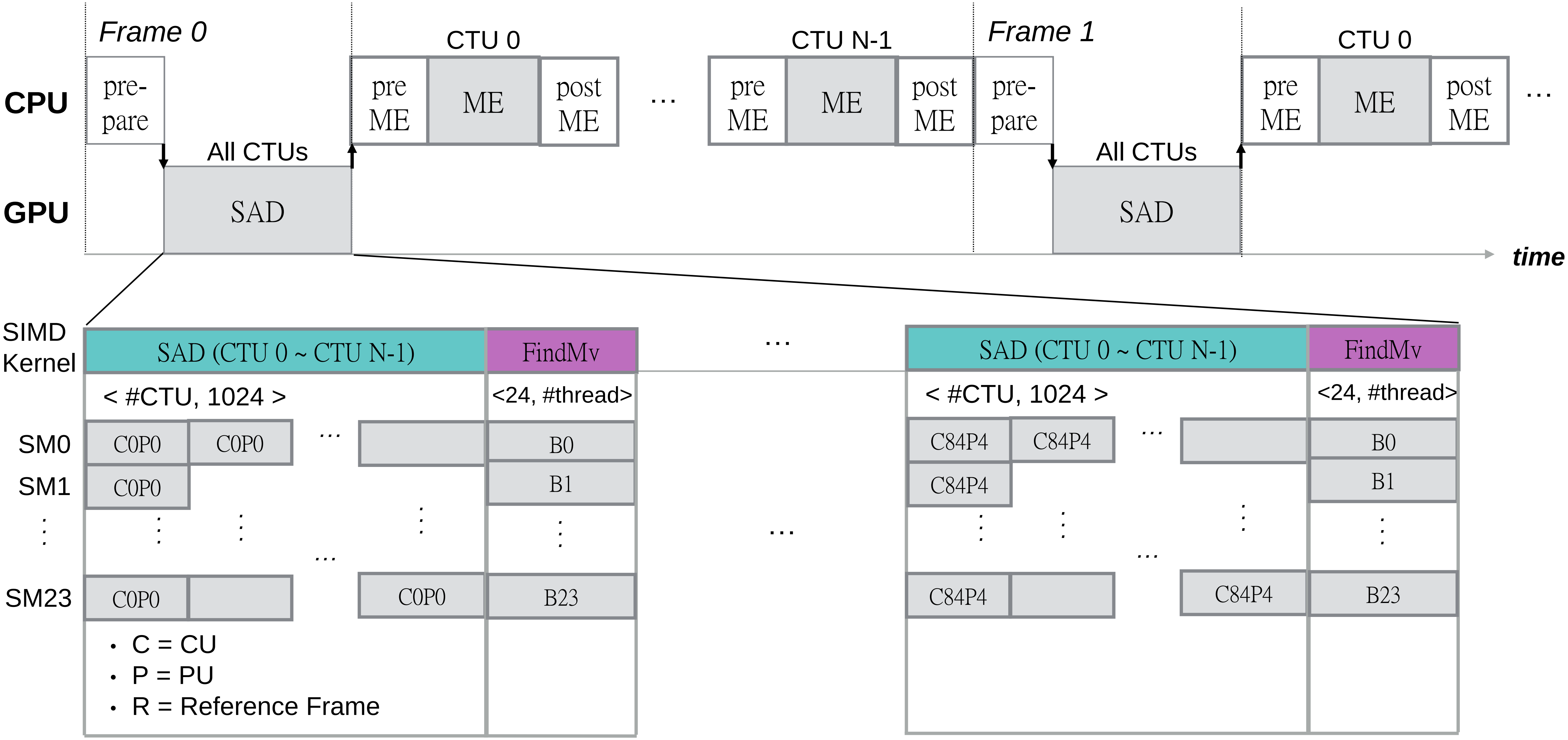
MemCpy (DtoH)

Compute



Proposed Method Ver. 2

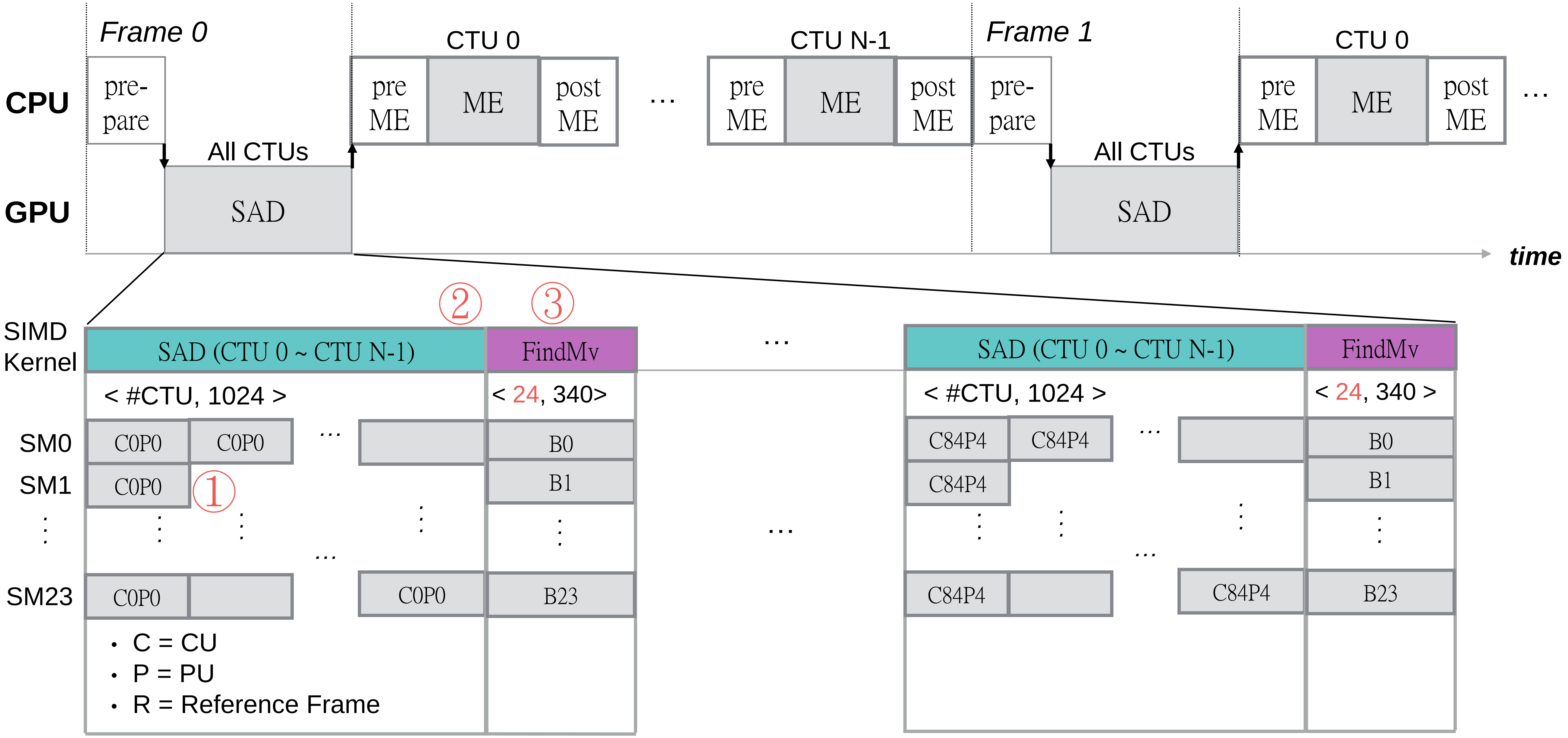
CTU-level Parallelism



- C = CU
- P = PU
- R = Reference Frame

Proposed Method Ver. 2(cont'd)

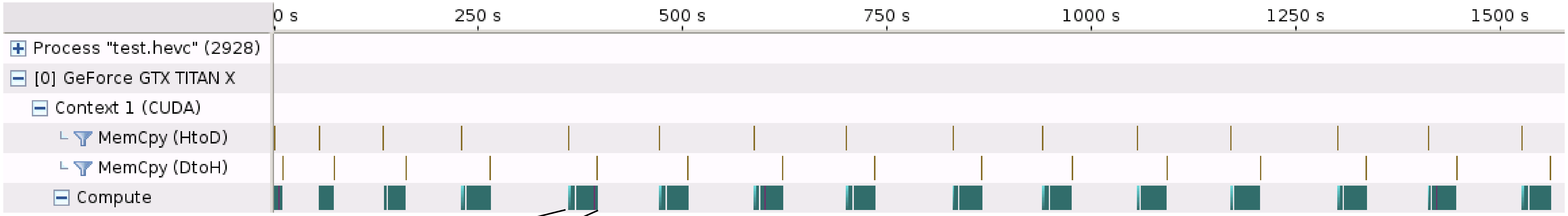
CTU-level Parallelism



① How to efficiently compute SADs using CUDA blocks?
② How much memory is required for each SAD kernel?
③ How to determine the number of threads for FindMv?

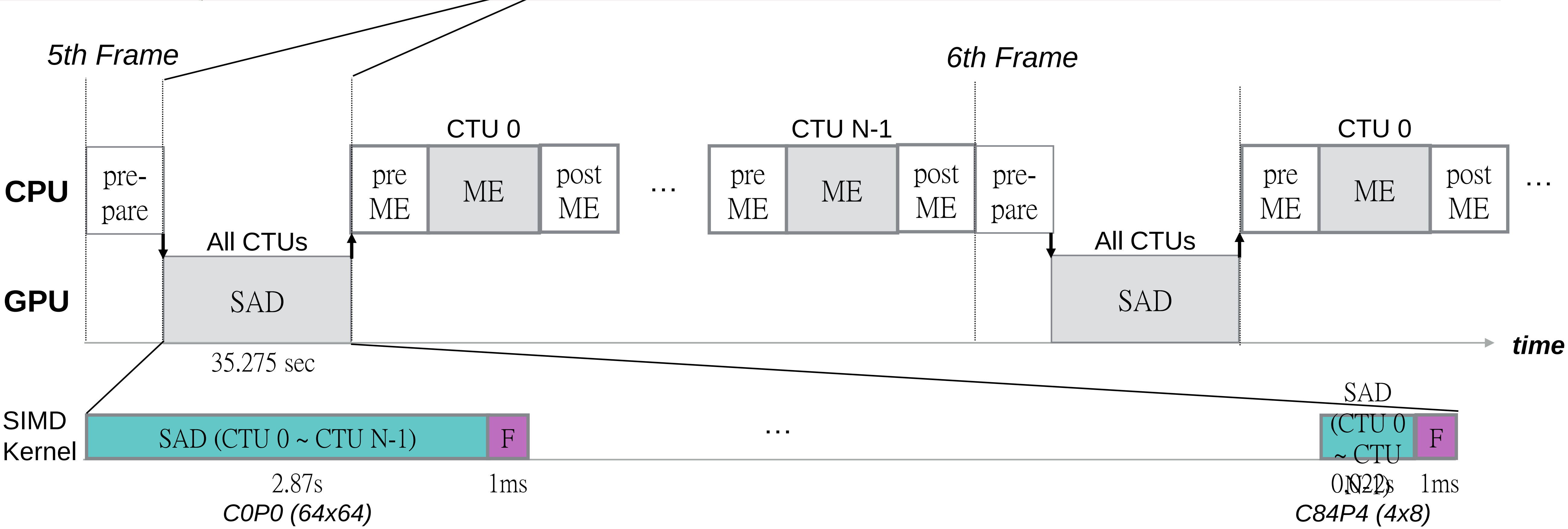
Timing Profile of Ver. 2

Test Sequence	**4K**: Marathon_3840x2160, **16** frames, QP=32



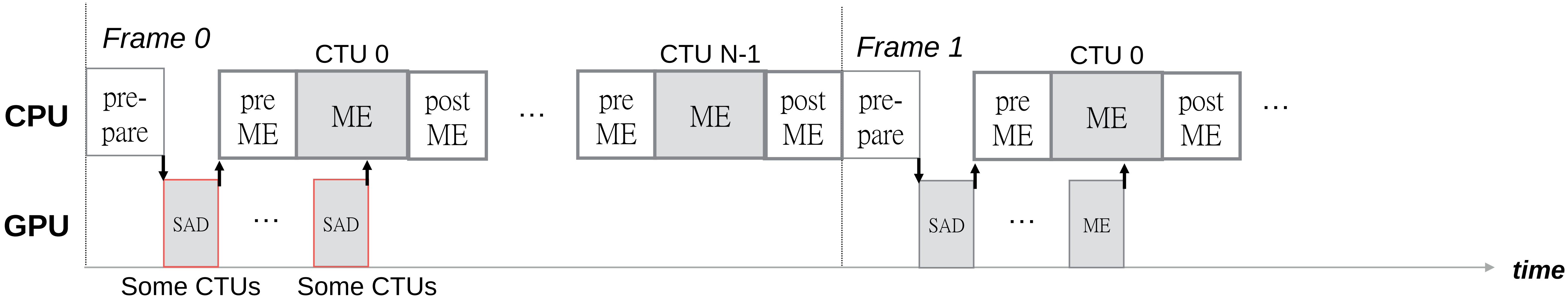
0 s 250 s 500 s 750 s 1000 s 1250 s 1500 s

- Process "test.hevc" (2928)
- [0] GeForce GTX TITAN X
- Context 1 (CUDA)
- MemCpy (HtoD)
- MemCpy (DtoH)
- Compute

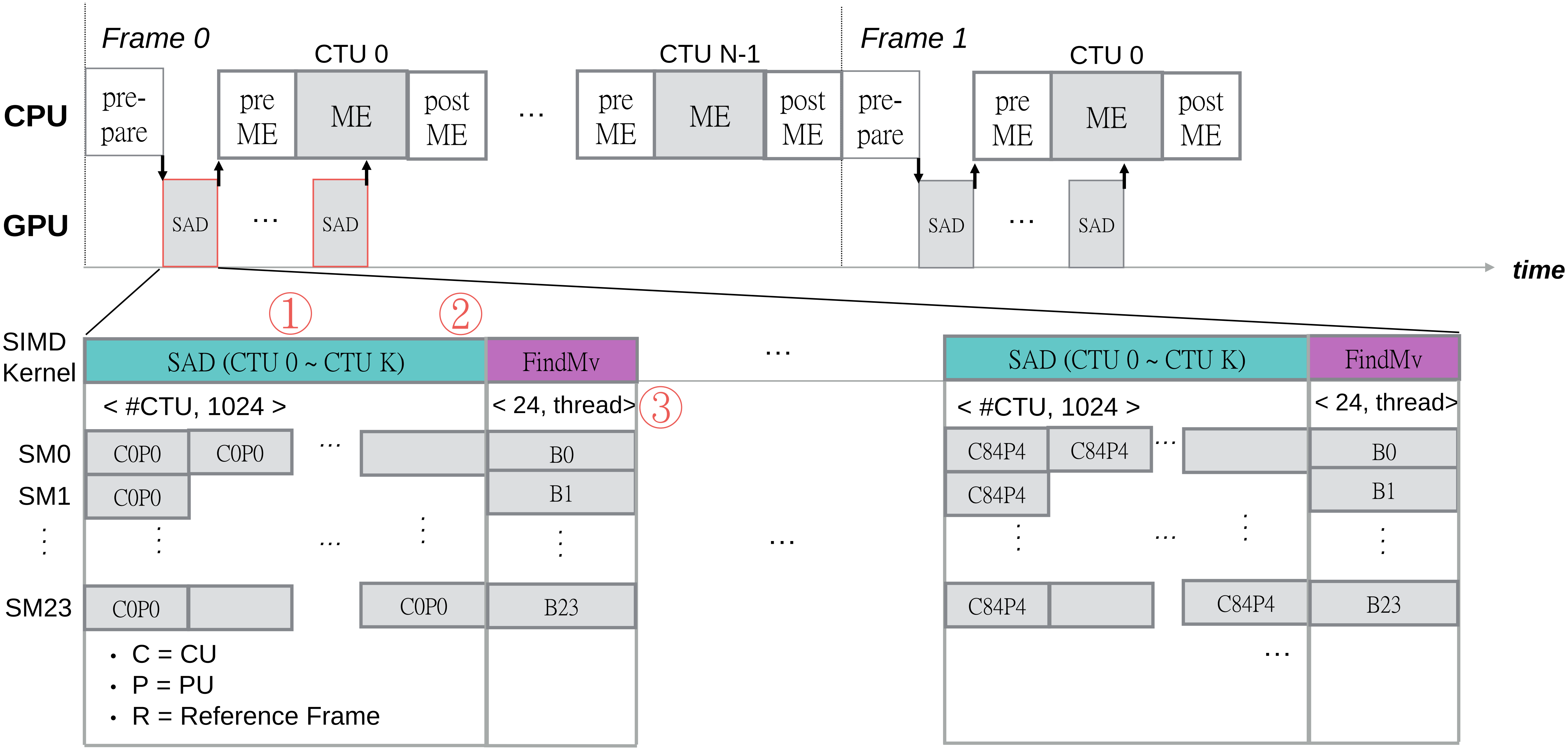


Proposed Method Ver. 3

Concurrent CPU & GPGPU Execution



Proposed Method Ver. 3(cont'd)

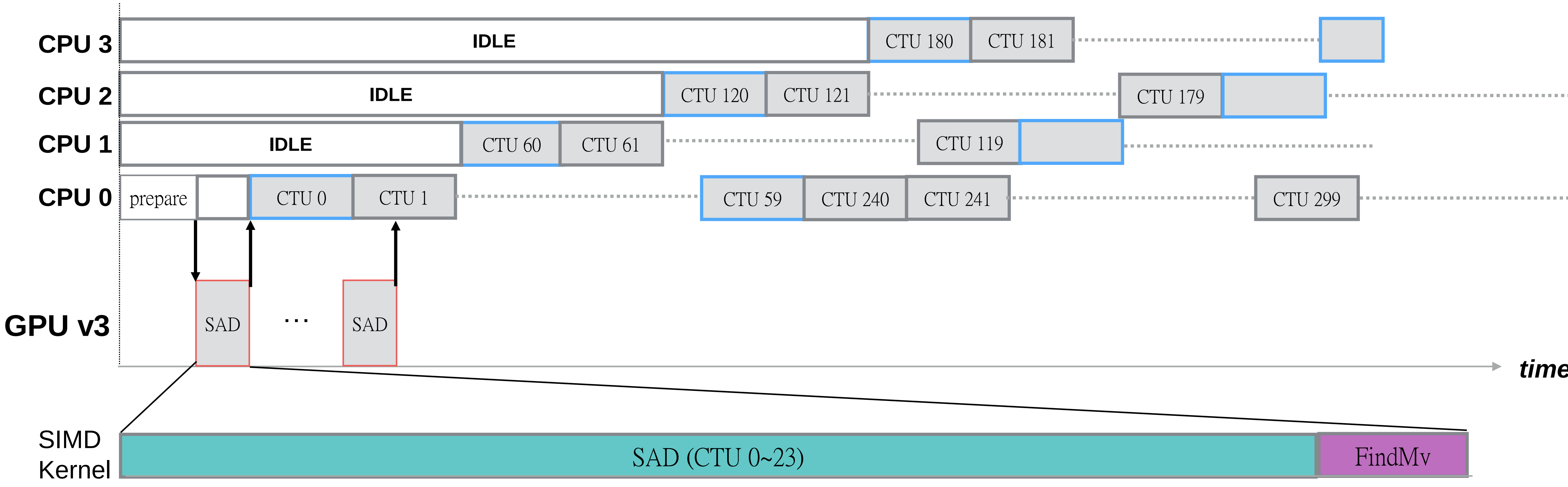


- C = CU
- P = PU
- R = Reference Frame

① **Number of CTUs per Batch?**
② **How much memory is required for each SAD kernel?**
③ **How to determine the number of threads of FindMv?**

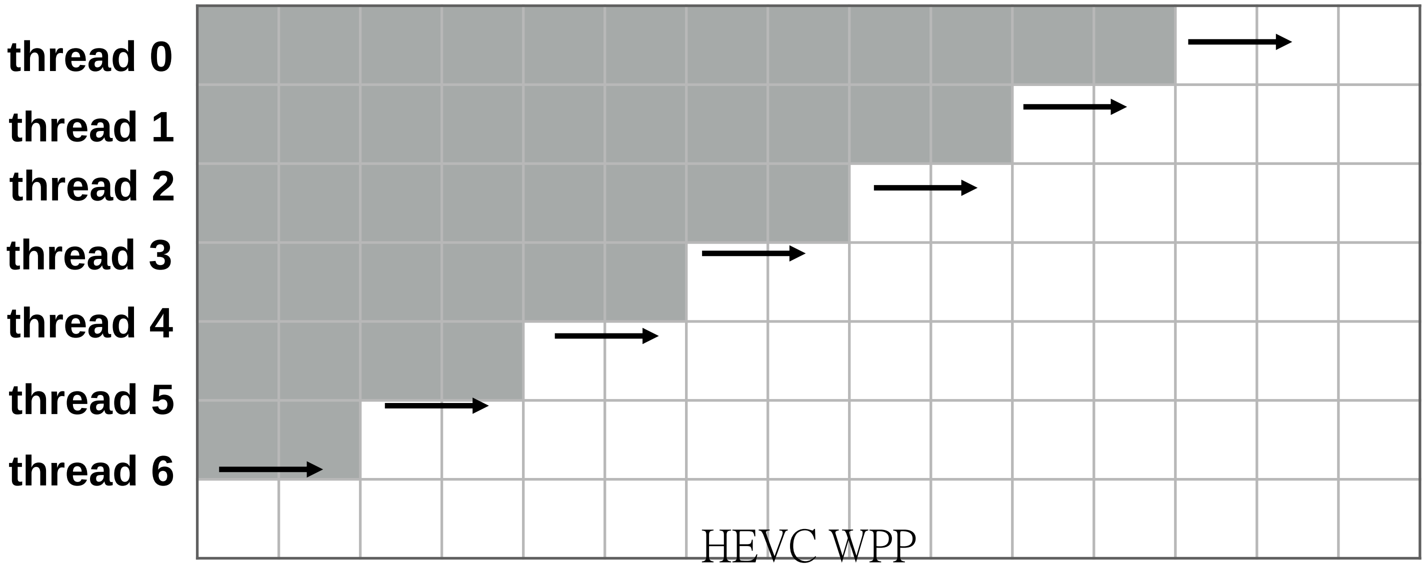
Proposed Method Ver. 4

Multi-Threaded CPUs



Proposed Method Ver. 4(cont'd)

- **Multi-threaded encoding based on Wave-front Parallel Processing (WPP)**



HEVC WPP

Problems:

① **HM reference software is not designed for full feature multi-threading**

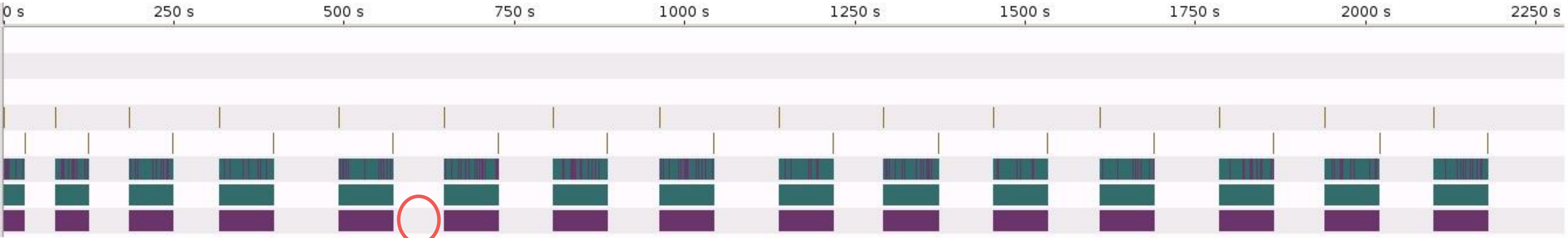
② **CTU-level dependency between CPU threads**

③ **How many active threads?**

Visual Profiler

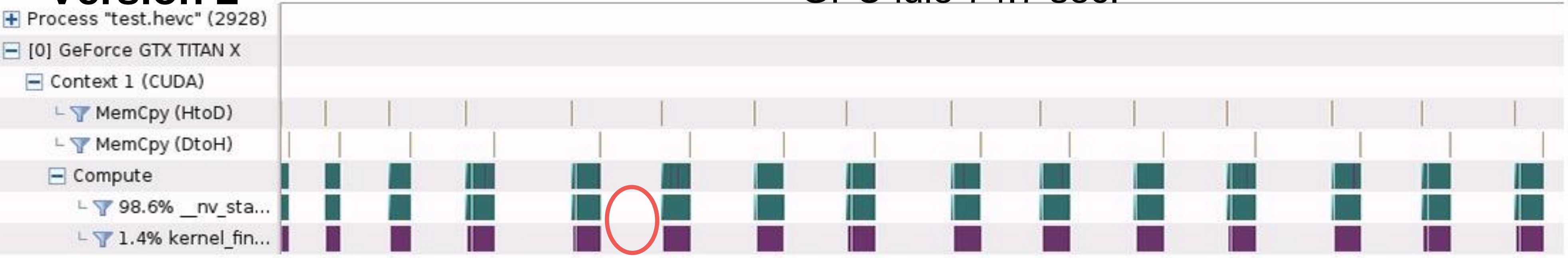
Test Sequence	**4K**: Marathon_3840x2160, **16** frames, QP=32

Version 1



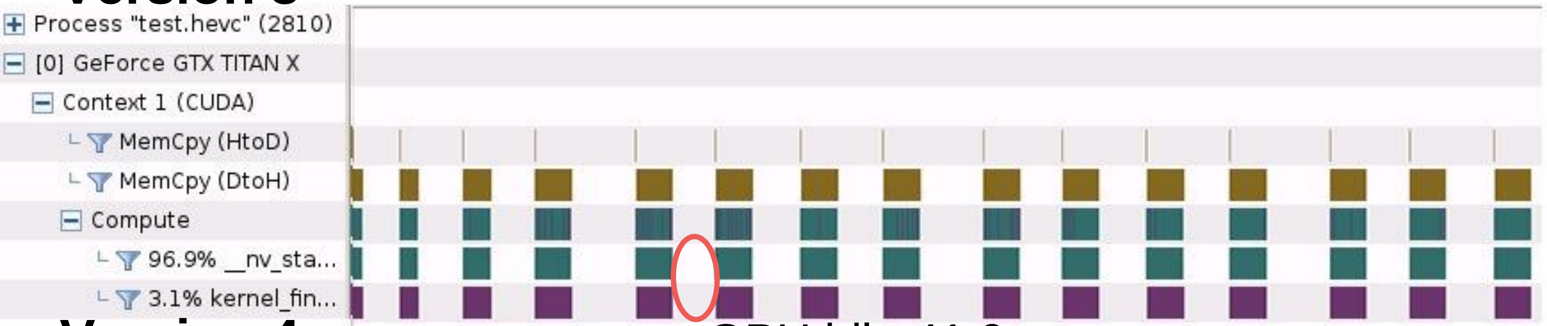
GPU idle 74.7 sec.

Version 2



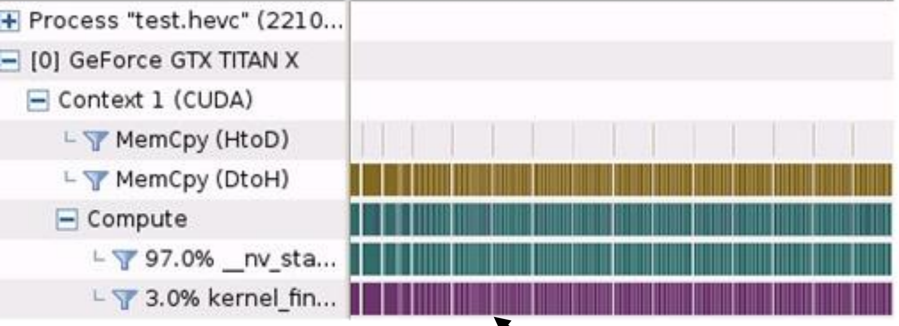
GPU idle 75.3 sec

Version 3



GPU idle 41.9 sec.

Version 4



GPU idle 0.79 sec.

Outline

- Introduction
- System and Software
- Proposed Methods and Implementations
- **Experiment Results**
- Conclusion
- Reference

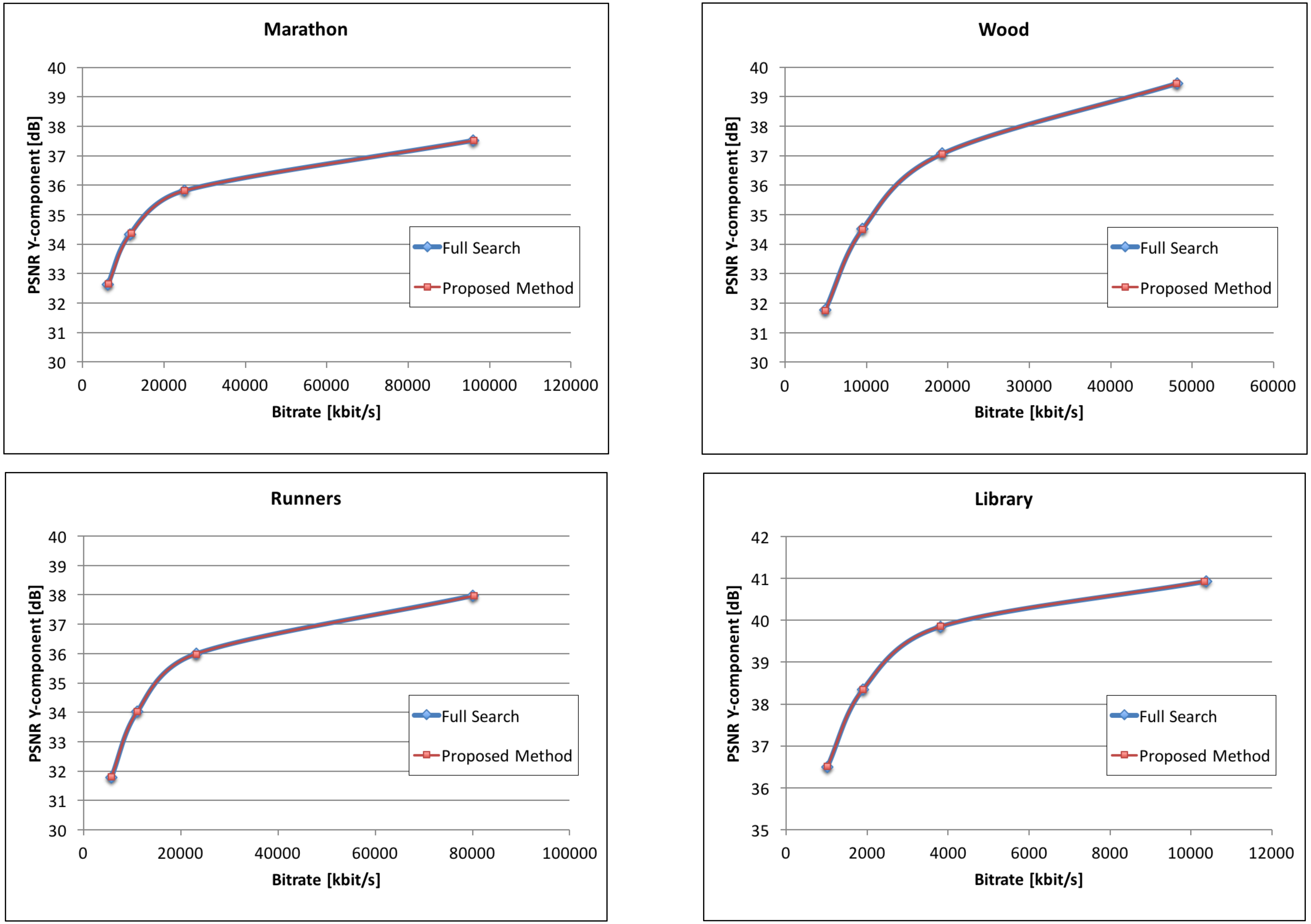
Experiment Results

Version	BD-BR [%]	BD-PSNR [dB]	Motion Estimation		Total Encoding	
			Time [s]	Speed-up	Time [s]	Speed-up
Sequential	-	-	25666.2	-	26196.8	-
1	0.138	-0.0037	550.56	45.41	1050.66	22.36
2	0.143	-0.0039	255.50	101.50	756.71	30.74
3	0.143	-0.0039	241.84	104.39	506.01	43.66
4	0.177	-0.0050	247.79	102.03	278.94	90.91

⁎ *Gray column is the results of Traffic_2560x1600 sequence at QP 32.*

48.07 s

Rate-Distortion Curves



The RD-curve of **4K** sequences (3840x2160)

Rate-Distortion Curves(cont'd)

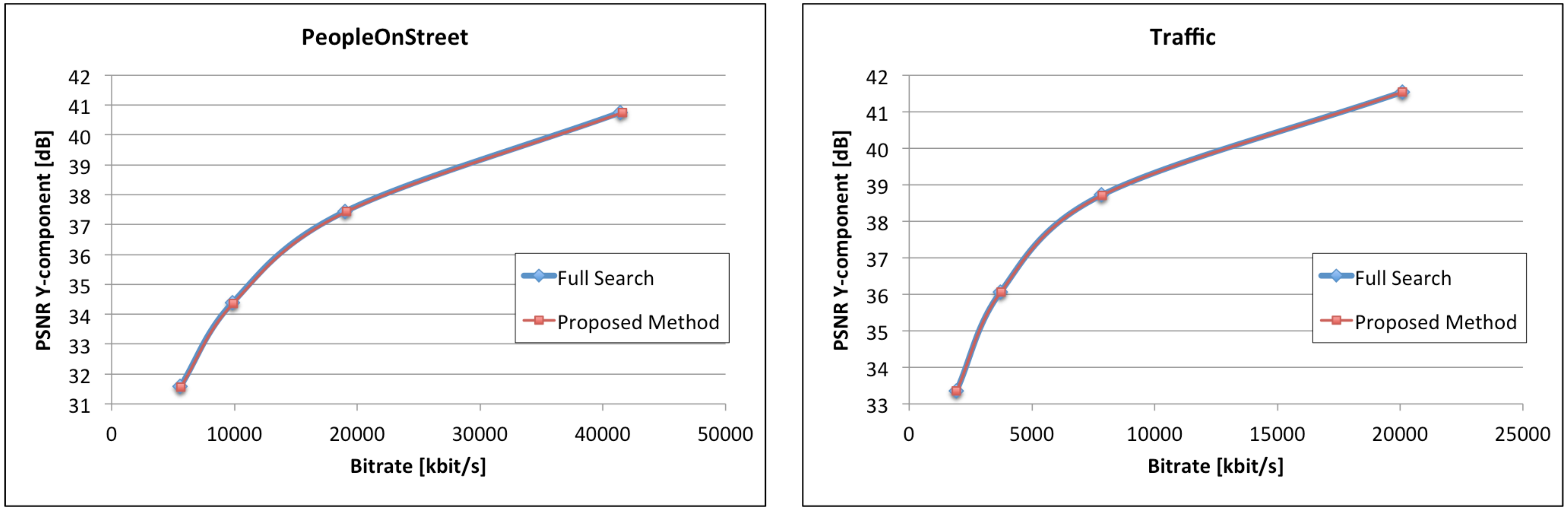
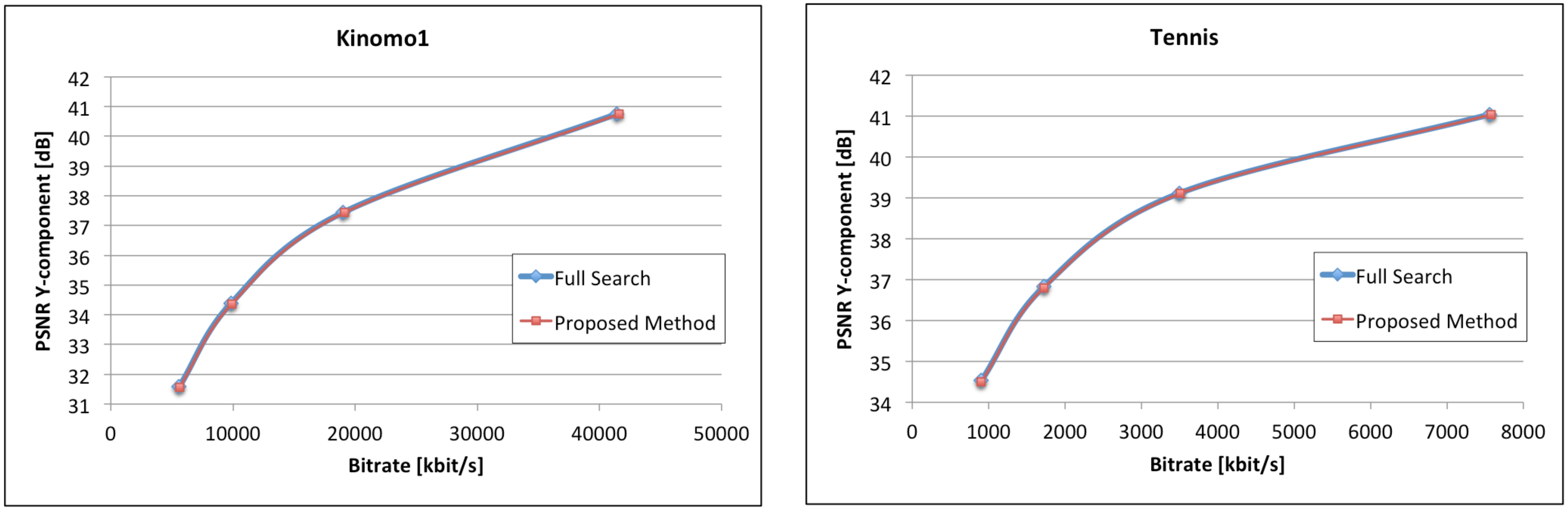
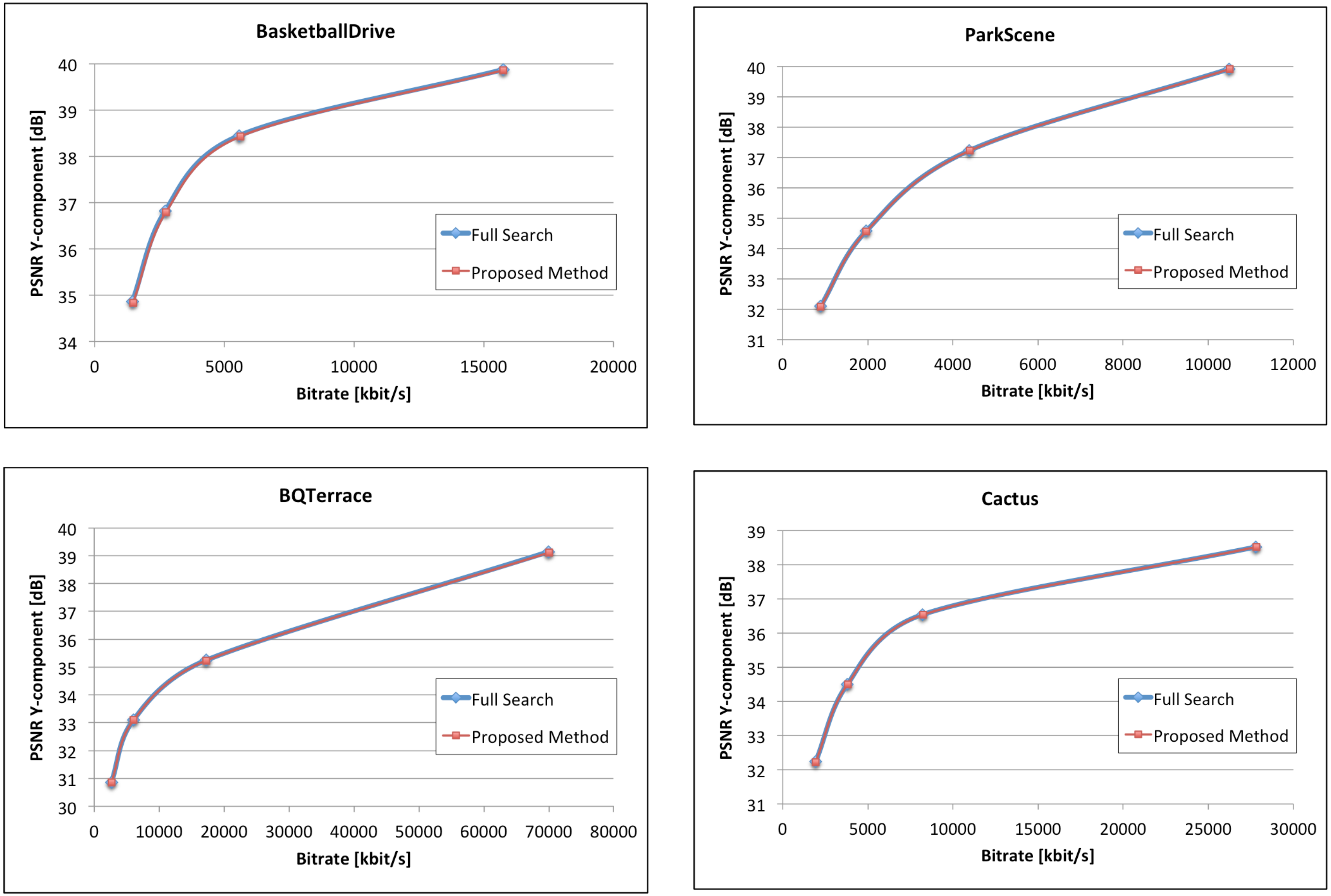


Figure. The RD-curve of class A sequences (2560x1600)



The RD-curve of **class B** sequences (1920x1080)

Rate-Distortion Curves(cont'd)



The RD-curve of **class B** sequences (1920x1080)

Conclusion

- GPGPU and Multicore
- 90X Speed-Up of HEVC Video Coding
- 102X Speed-Up of Motion Estimation
- 0.177% bit rate increase and 0.005db PSNR loss
- Utilization is the Key – Memory Allocation and Access

Thank You!!