GPU-acceleration of Affine Prediction in Versatile Video Coding

Iago Storch, Daniel Palomino, Sergio Bampi

Graduate Program in Computing (PPGC)
Federal University of Rio Grande do Sul (UFRGS)

2022 IEEE International Symposium on Circuits and Systems
May 28- June 1, 2022 Hybrid Conference

INTRODUCTION

- **Video coding** applications are **highly computing-intensive**
- Many computing systems are **equipped with GPUs**
- GPUs have been used to accelerate different encoding stages
  - Motion Estimation
  - Intra Prediction
  - In-loop Filtering

INTRODUCTION

- **Versatile Video Coding (VVC)** standard introduced the **affine prediction**
- Similar to (translational) motion estimation, but allows **affine motion model**
- Responsible for most of the ME time > **most time-demanding tool***
- **Accelerating affine prediction** is prime to low delay applications

*I. Siqueira, G. Correa and M. Grellert, "Complexity and Coding Efficiency Assessment of the Versatile Video Coding Standard," @ ISCAS 2021

INTRODUCTION

- Most **GPU techniques** designed for ME are **not efficient for affine** prediction:
  - Testing a predefined set of MVs has prohibitive complexity
  - Pre-computing the distortion of all blocks demands unreasonable memory
  - It is not possible to merge the distortion of adjacent blocks

INTRODUCTION

- Most **GPU techniques** designed for ME are **not efficient for affine** prediction:
  - Testing a predefined set of MVs has prohibitive complexity
  - Pre-computing the distortion of all blocks demands unreasonable memory
  - It is not possible to merge the distortion of adjacent blocks

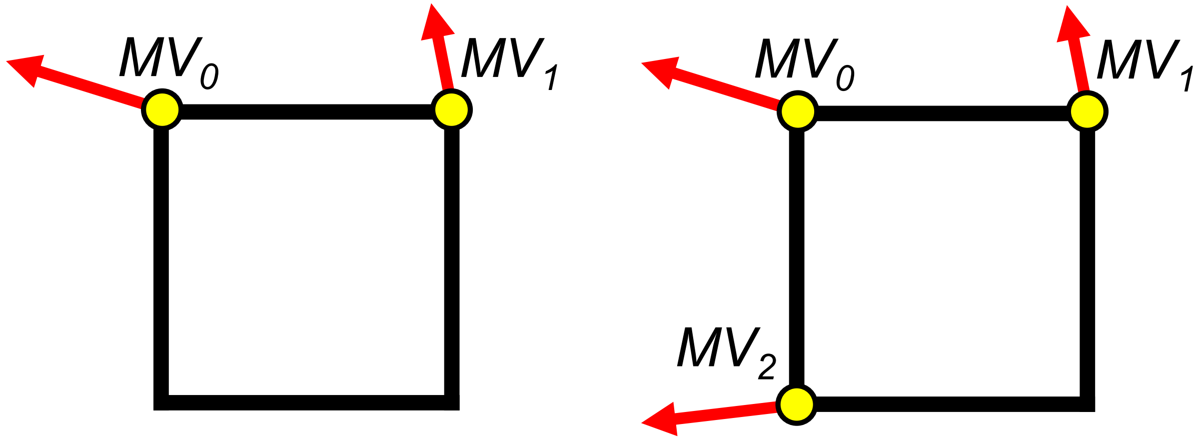
This work proposes to remodel the affine prediction aiming to explore GPU platforms

VVC AND AFFINE PREDICTION

- Frame is divided into **Coding Tree Units** (**CTUs**, 128x128 samples)
- Each CTU is the root of a **Coding Unit (CU)** tree, which follows recursive partitioning
- **Affine** prediction is available for **CUs 16x16 or larger**

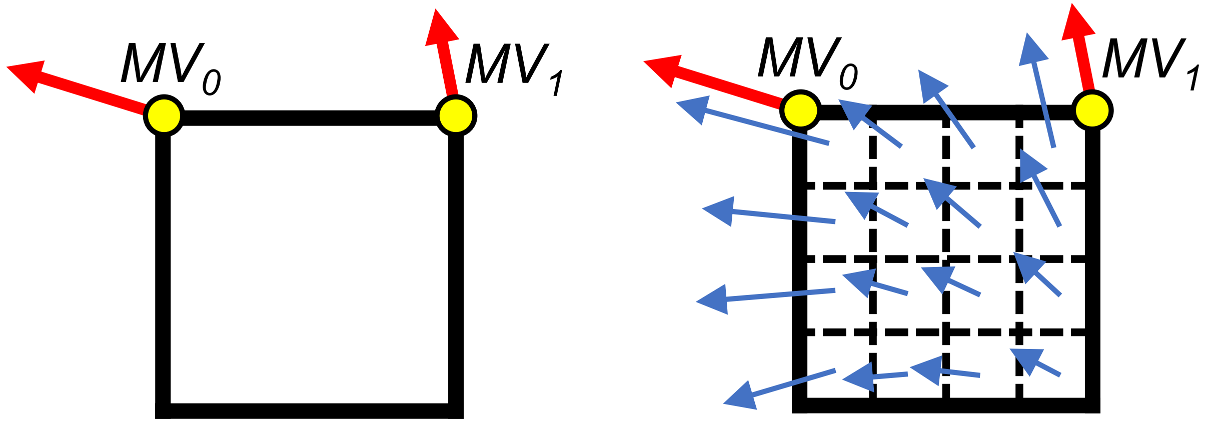
VVC AND AFFINE PREDICTION

- **Affine** can employ **two or three control points (CPs)**
- **Each CP** is assigned **a CP motion vector (CPMV)**



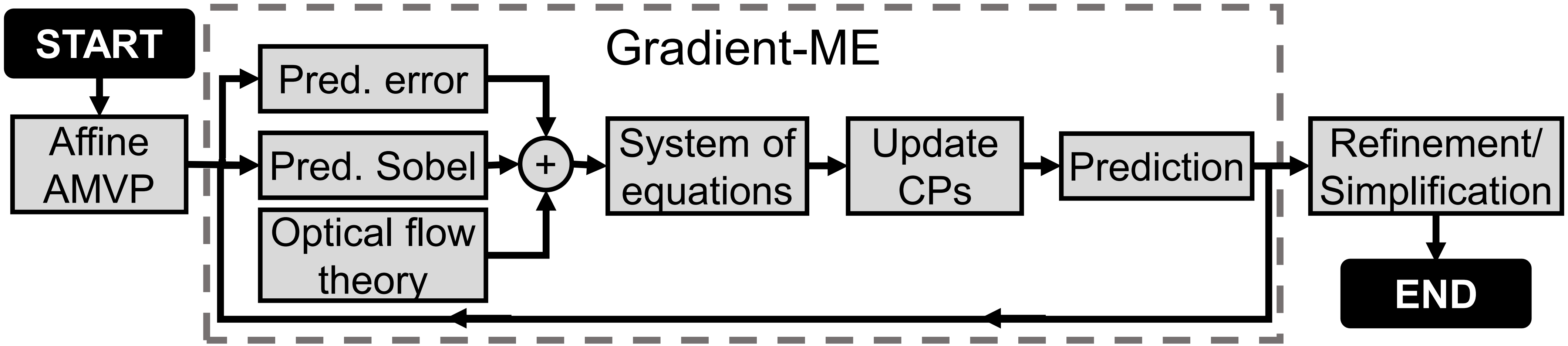
VVC AND AFFINE PREDICTION

- **Affine** can employ **two or three control points (CPs)**
- **Each CP** is assigned **a CP motion vector (CPMV)**
- CU **divided into sub-blocks 4x4**, each one is assigned a MV based on: CPMVs, CU dimensions, Sub-block position in CU



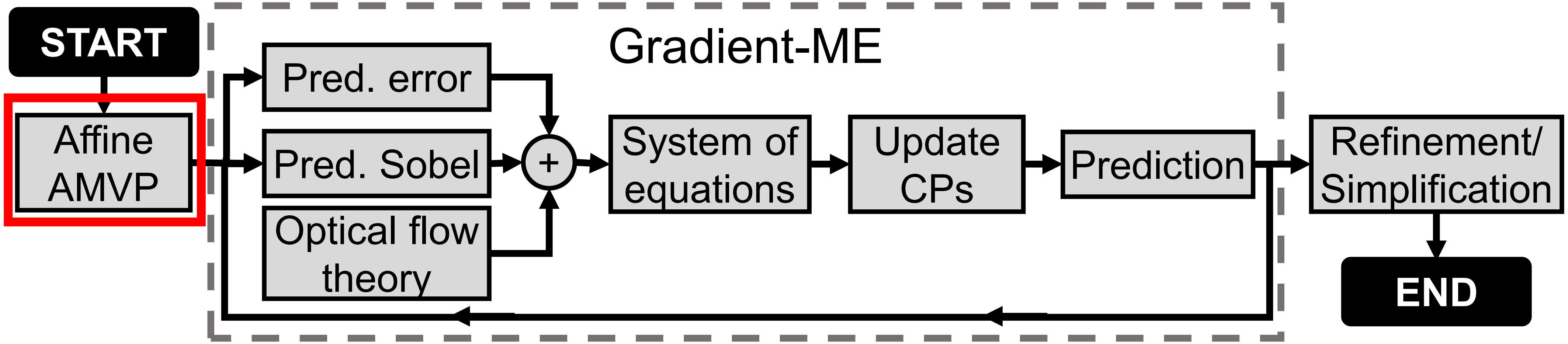
VVC AND AFFINE PREDICTION

- **Classical** ME **algorithms** are **not suitable** for **affine** prediction
- VVC reference software introduces a **Gradient-ME** algorithm



VVC AND AFFINE PREDICTION

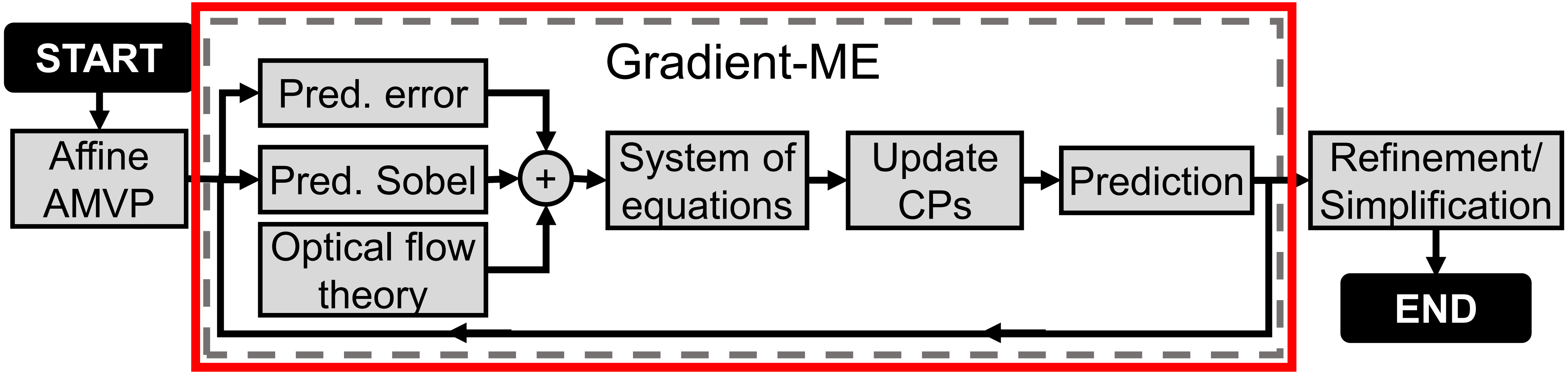
- **Classical** ME **algorithms** are **not suitable** for **affine** prediction
- VVC reference software introduces a **Gradient-ME** algorithm



Generates a set of predicted CPMVs

VVC AND AFFINE PREDICTION

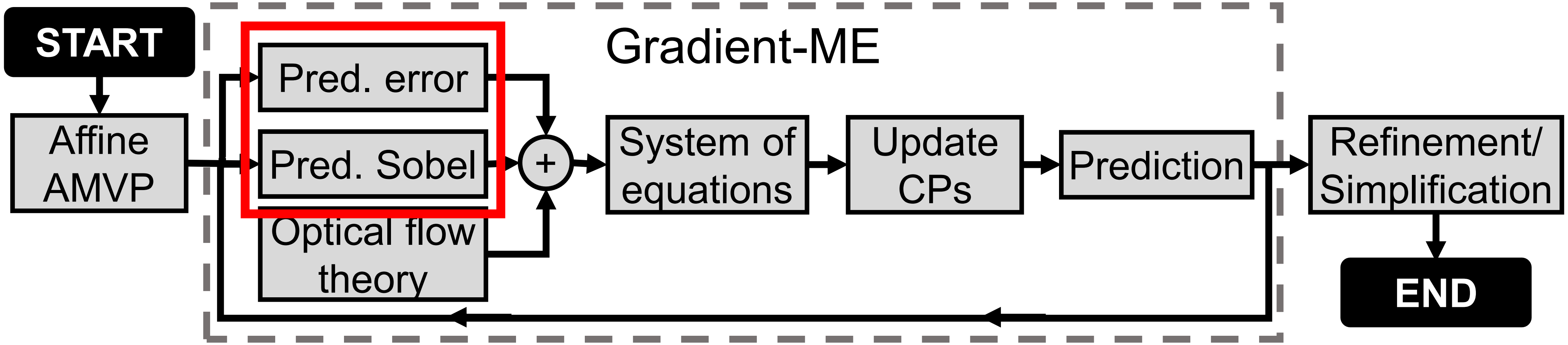
- **Classical** ME **algorithms** are **not suitable** for **affine** prediction
- VVC reference software introduces a **Gradient-ME** algorithm



Optimizes two/three CPMVs simultaneously

VVC AND AFFINE PREDICTION

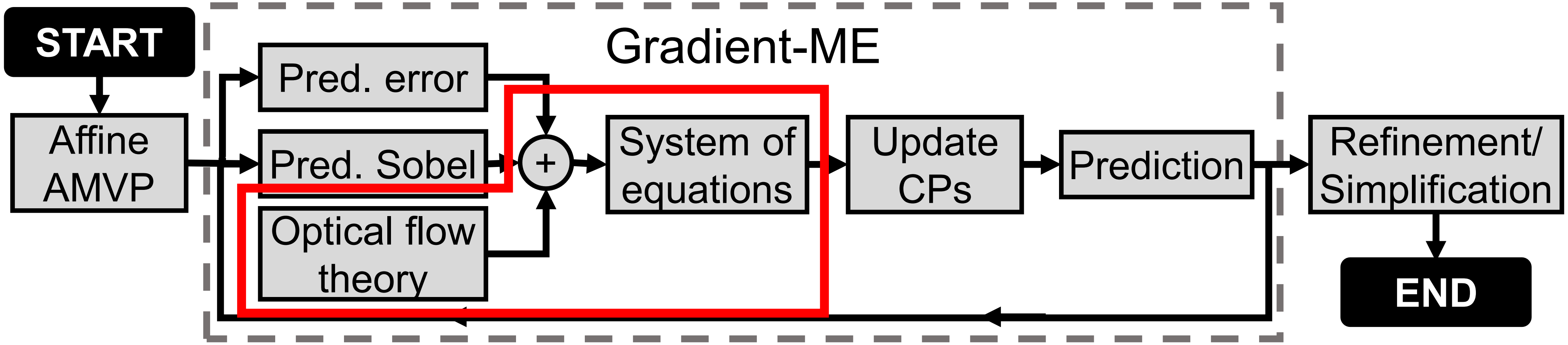
- **Classical** ME **algorithms** are **not suitable** for **affine** prediction
- VVC reference software introduces a **Gradient-ME** algorithm



Prediction error is computed sample-wise
Prediction gradient computed with Sobel op.

VVC AND AFFINE PREDICTION

- **Classical** ME **algorithms** are **not suitable** for **affine** prediction
- VVC reference software introduces a **Gradient-ME** algorithm

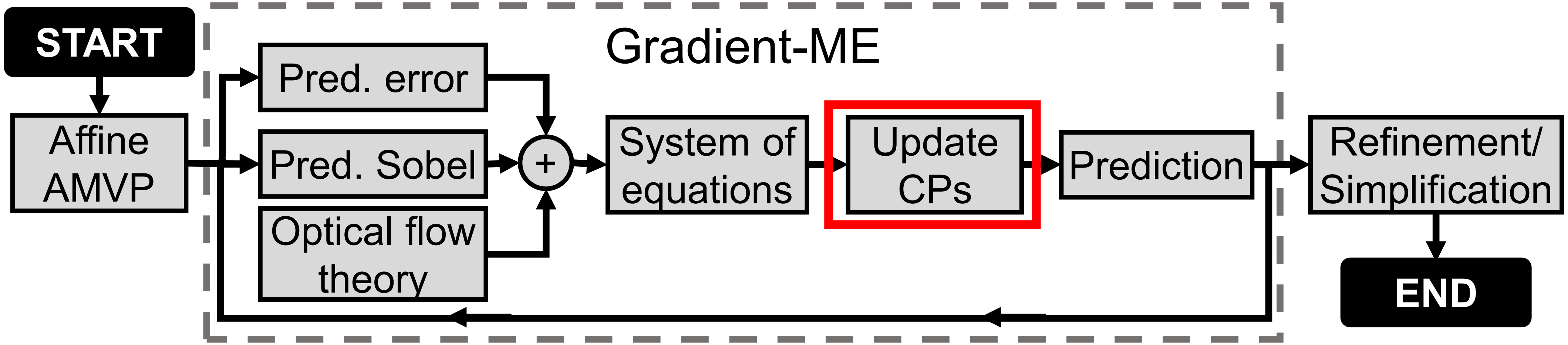


Pred error and gradient combined into system of equations
It updates the affine parameters, minimizing pred error

UFRGS ViTech VIDEO TECHNOLOGY RESEARCH GROUP

VVC AND AFFINE PREDICTION

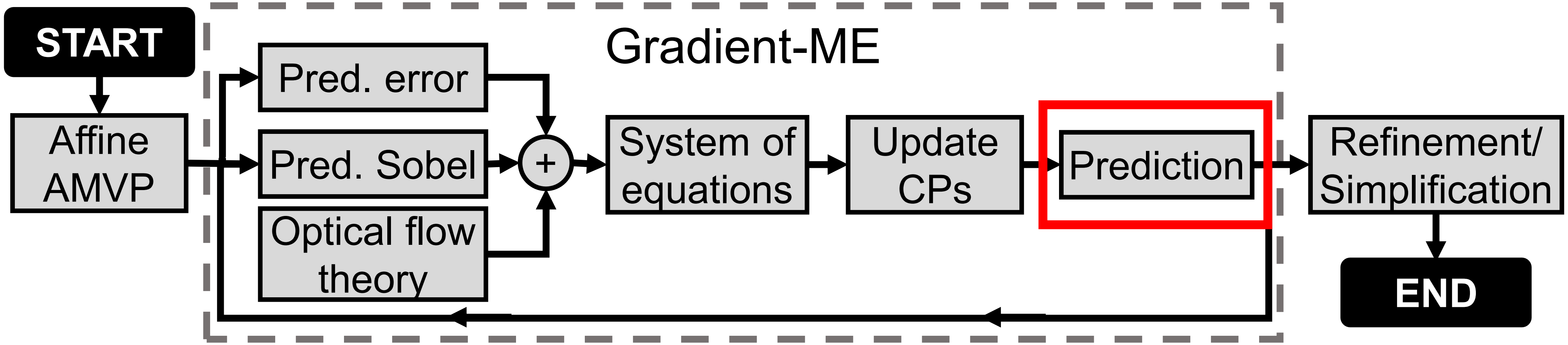
- **Classical** ME **algorithms** are **not suitable** for **affine** prediction
- VVC reference software introduces a **Gradient-ME** algorithm



Affine parameters are converted to ΔCPs and the CPs are updated
"Fractional" ME embedded into Gradient-ME

VVC AND AFFINE PREDICTION

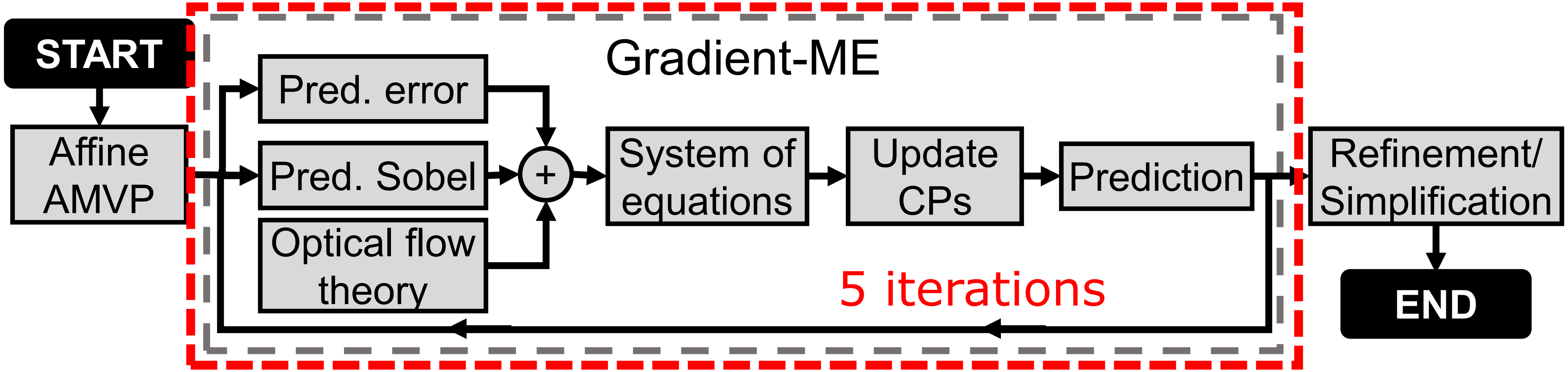
- **Classical** ME **algorithms** are **not suitable** for **affine** prediction
- VVC reference software introduces a **Gradient-ME** algorithm



Updated CPs are used to conduct a new prediction

VVC AND AFFINE PREDICTION

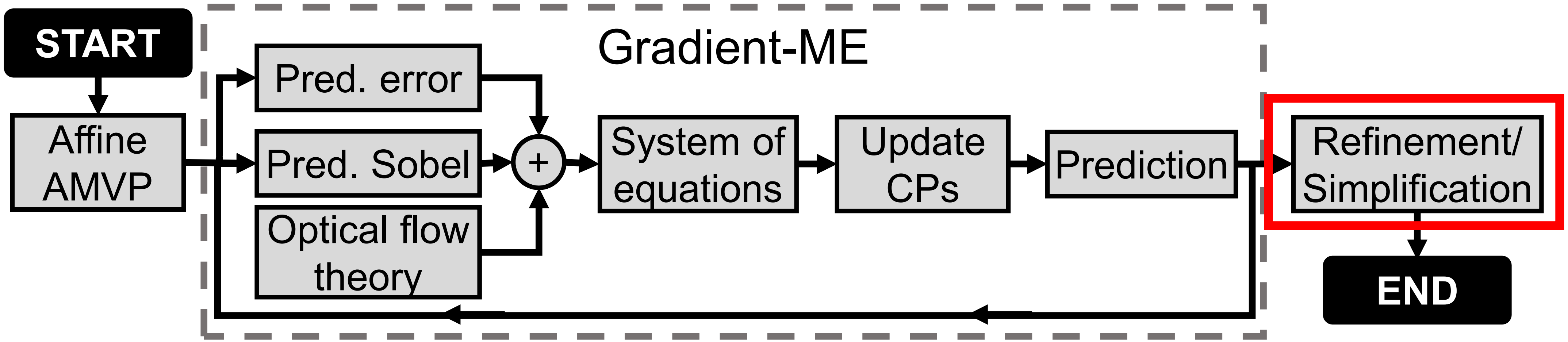
- **Classical** ME **algorithms** are **not suitable** for **affine** prediction
- VVC reference software introduces a **Gradient-ME** algorithm



Gradient-ME is repeated 5 times
on reference encoder

VVC AND AFFINE PREDICTION

- **Classical** ME **algorithms** are **not suitable** for **affine** prediction
- VVC reference software introduces a **Gradient-ME** algorithm



Small variations over the best CPMVs

VVC AND AFFINE PREDICTION

- **Affine** prediction raises novel **parallelization challenges and opportunities**:
  - Compute the **gradient** of multiple samples concurrently
  - Compute the **prediction error** of multiple samples concurrently
  - Build the system of equations in parallel, **multiple partial systems** concurrently
  - Compute the **distortion of multiple sub-blocks** concurrently

PROPOSED MODEL - OVERVIEW

- **Input**: original and ref frame
- **Output:** best CPMVs and costs
- Gradient-ME is performed 5 times
- Remaining encoding stages carried by CPU

PROPOSED MODEL - OVERVIEW

- Affine AMVP depends on CPMVs on adjacent blocks
  - Incurs data dependencies between CUs
  - **Our simplified AAMVP** always **produces zero CPMV** (0,0)
  - Allows affine prediction of all blocks (**Inter-CTU parallelism**)
  - Gradient-ME converges quickly → **Final CPMVs are similar**

PROPOSED MODEL - OVERVIEW

- Refinement/simplification is burdensome and not very efficient
  - Around 65% of Gradient-ME time
  - Small coding efficiency gains*
  - **Proposed work discards these stages**

*Y. He, X. Xiu, Y. Ye, "CE4-related: Affine motion estimation improvements," @ Document JVET-L0260
*X. Xiu, Y.-W. Chen, T.-C. Ma, H.-J. Jhu, X. Wang, "CE4-related: Motion estimation improvements," @ Document JVET-O0592

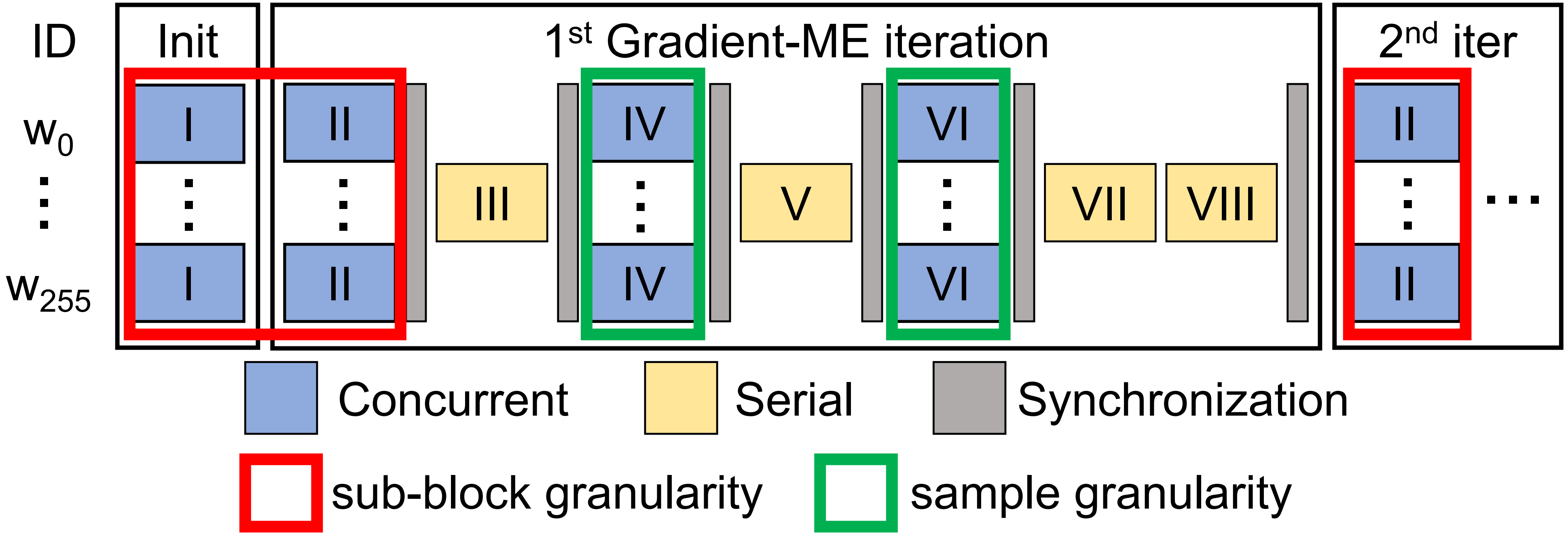
PROPOSED MODEL - OVERVIEW

- Original SATD computed with variable size HAD matrices
  - Creates a dependency **between all sub-blocks** of a CU
  - **Proposed work only uses 4x4 matrices** aligned with sub-blocks
  - Distortion of a sub-block is computed directly after its prediction
  - Less accurate but **highly parallel**
  - Final distortion obtained by adding partial values

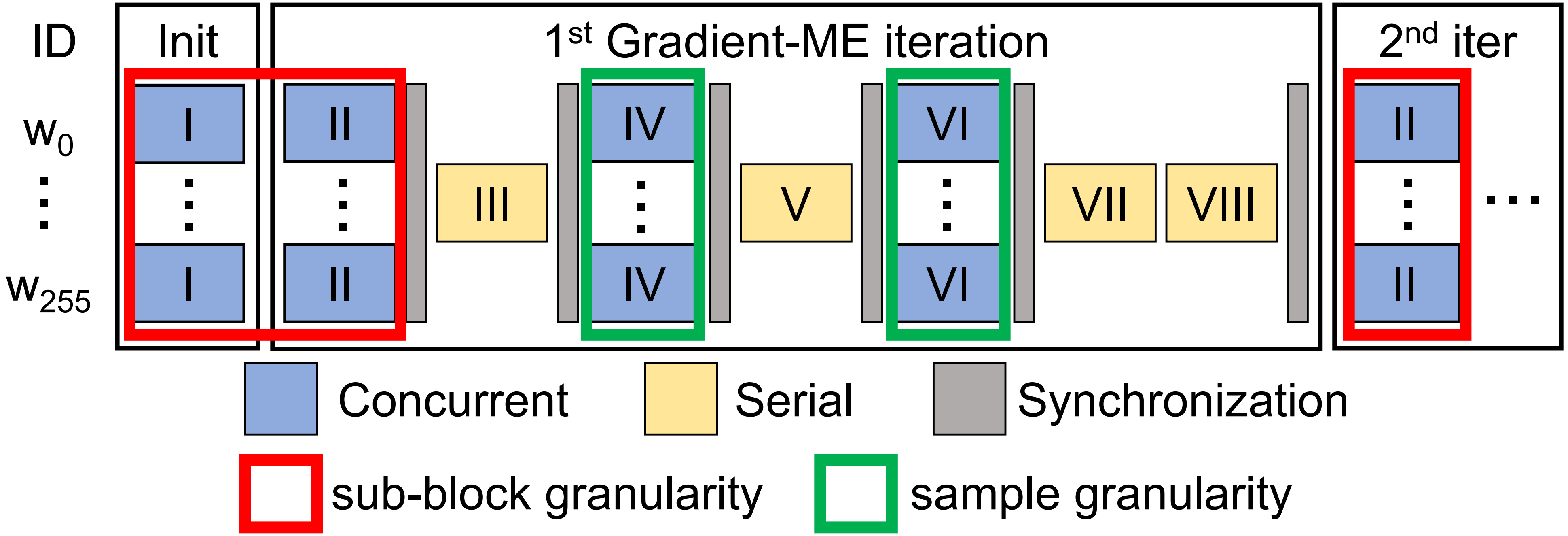
PROPOSED MODEL - DETAILS

- Currently supports 2 CPs and CUs 128x128
- Implemented in **OpenCL**
  - 1 workgroup (WG) per CTU
  - 256 workitems per workgroup (items on the same CTU share data)
  - **All CTUs predicted concurrently**
  - Prediction of a CTU broken into 8 stages (**Intra-CTU parallelism**)

PROPOSED MODEL - DETAILS



PROPOSED MODEL - DETAILS

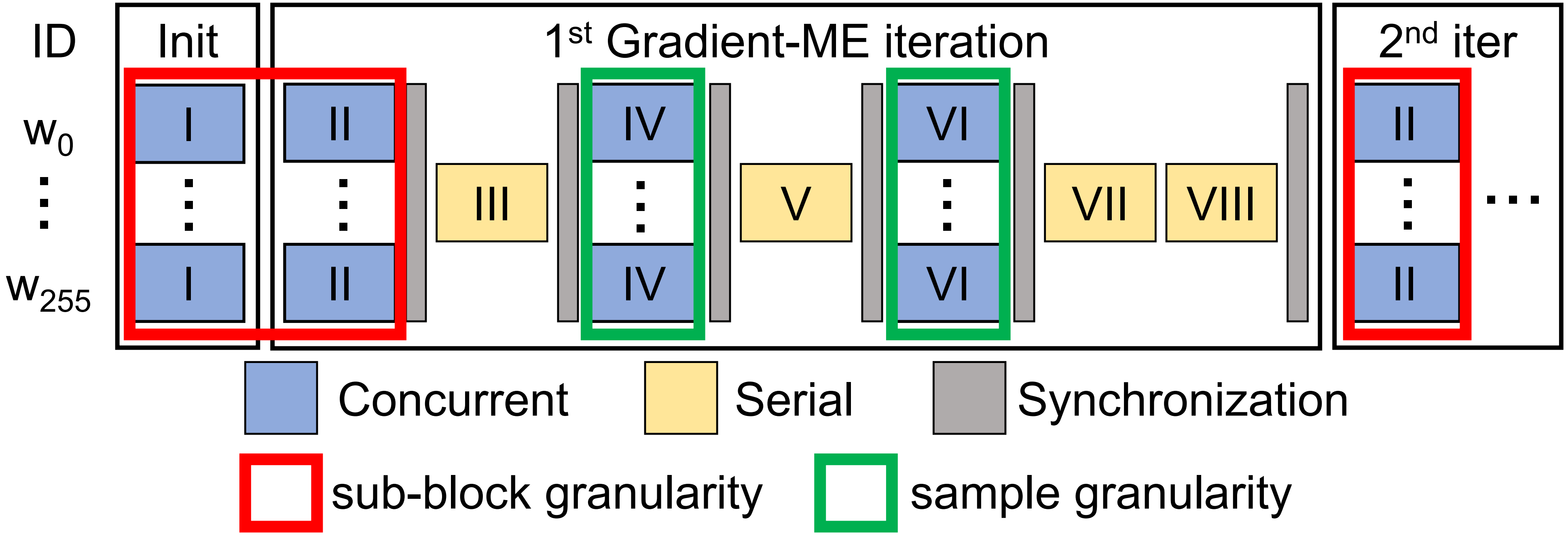


Stage I – SB Concurrency

Fetch the current CTU from global memory into private memory

Each item fetches 4 sub-blocks

PROPOSED MODEL - DETAILS



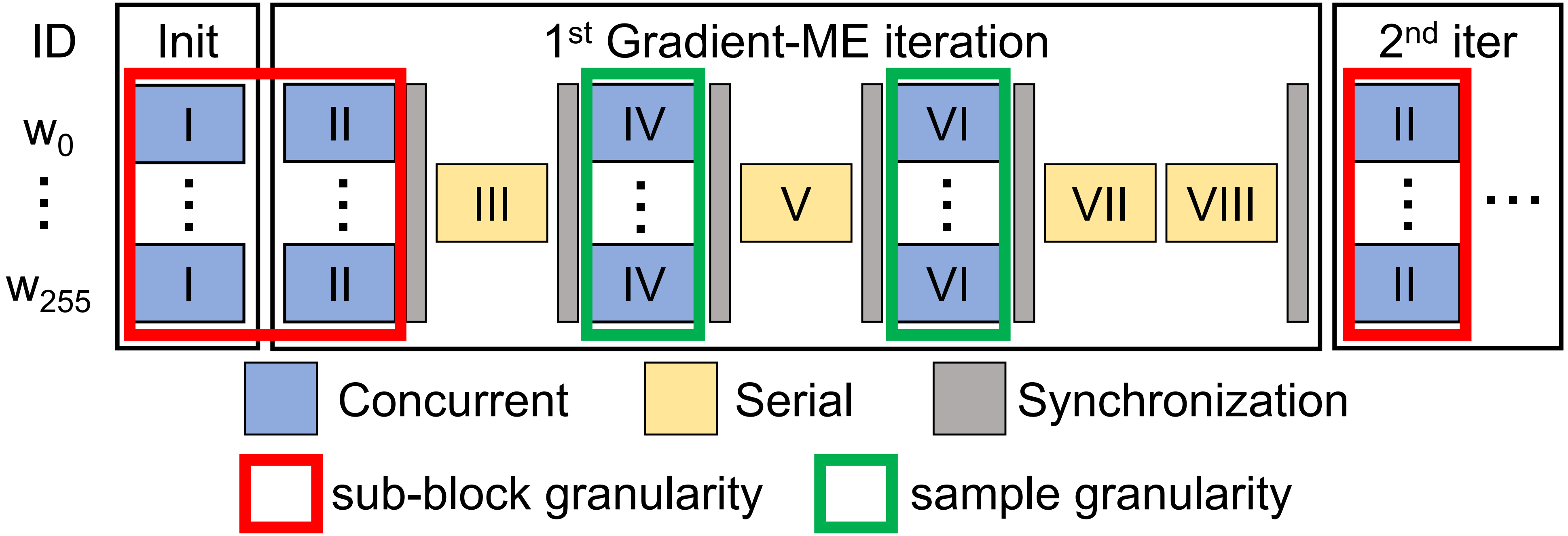
Stage II – SB Concurrency

Each item predicts its 4 sub-blocks and computes their distortion

Predicted samples stored in shared memory

Distortion stored in shared memory

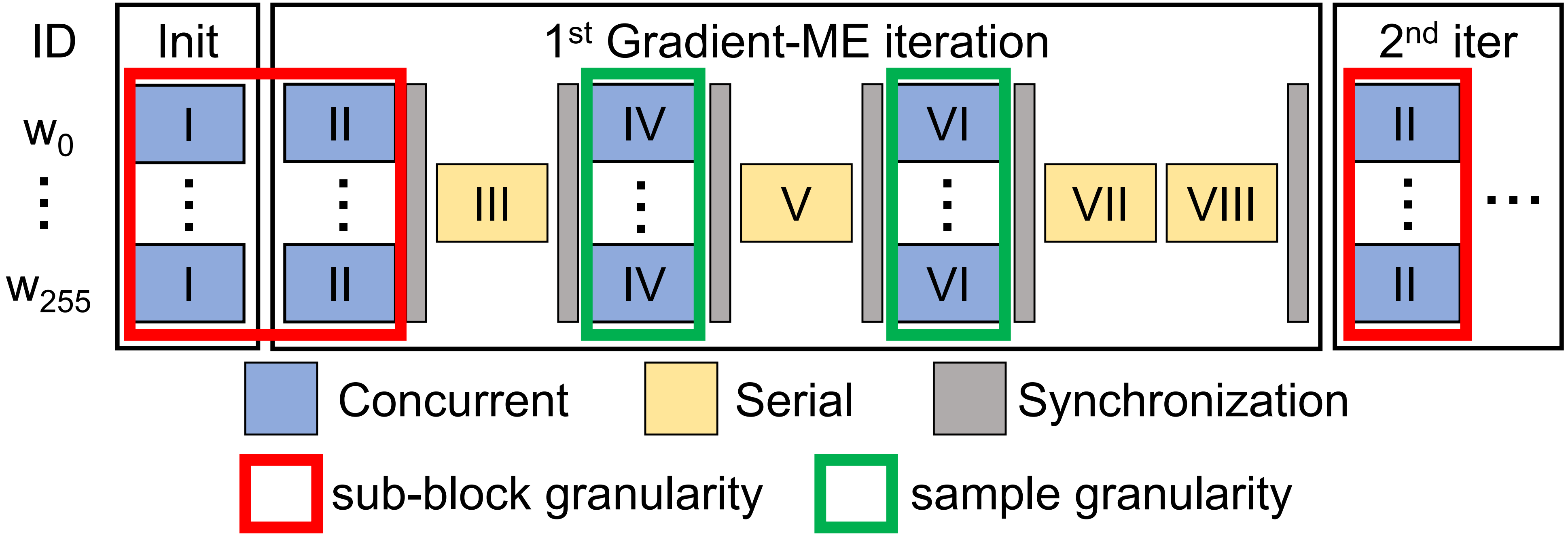
PROPOSED MODEL - DETAILS



Stage III - Serial

W_0 reduces all partial distortions into a single value

PROPOSED MODEL - DETAILS

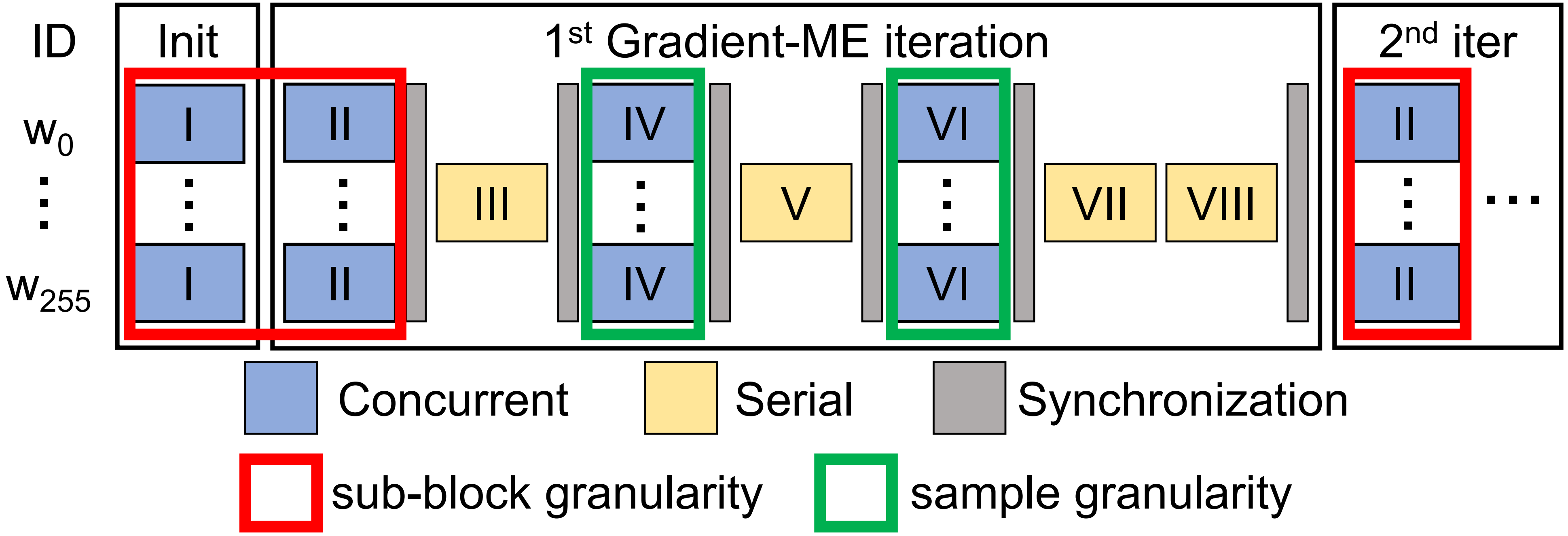


Stage IV – Sample parallelism

Each item computes the gradient for 64 samples

Gradient stored in global memory (large memory requirements)

PROPOSED MODEL - DETAILS

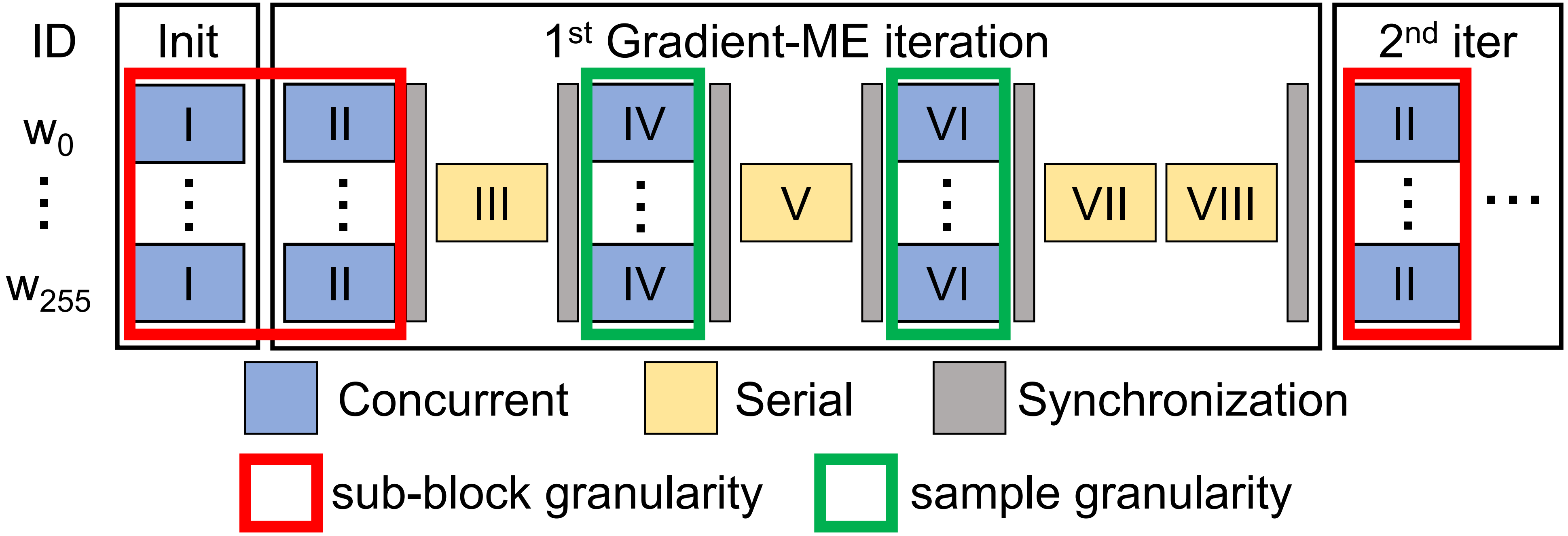


Stage V – Serial

W_0 computes the gradient for samples at the CU edges

Inherited from inner samples

PROPOSED MODEL - DETAILS



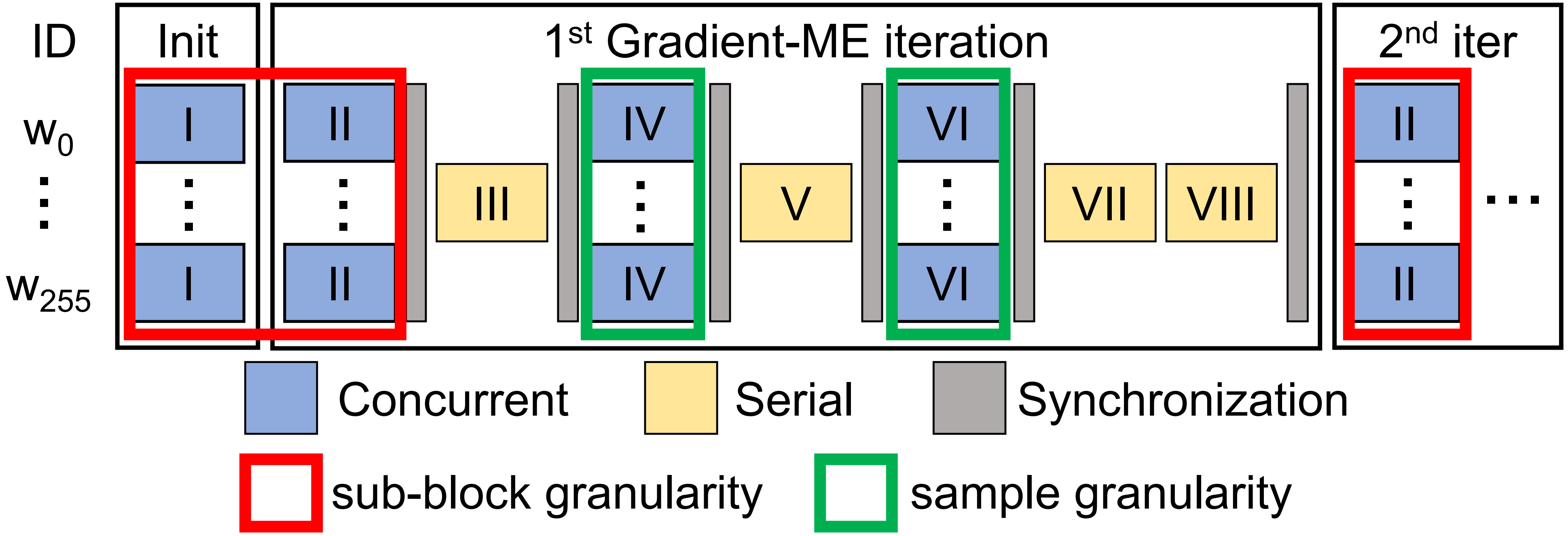
Stage VI – Sample parallelism

Each item computes the prediction error for 64 samples

Error overwrites prediction signal in shared memory (memory reuse)

Each item computes 64 partial systems, stored in global memory

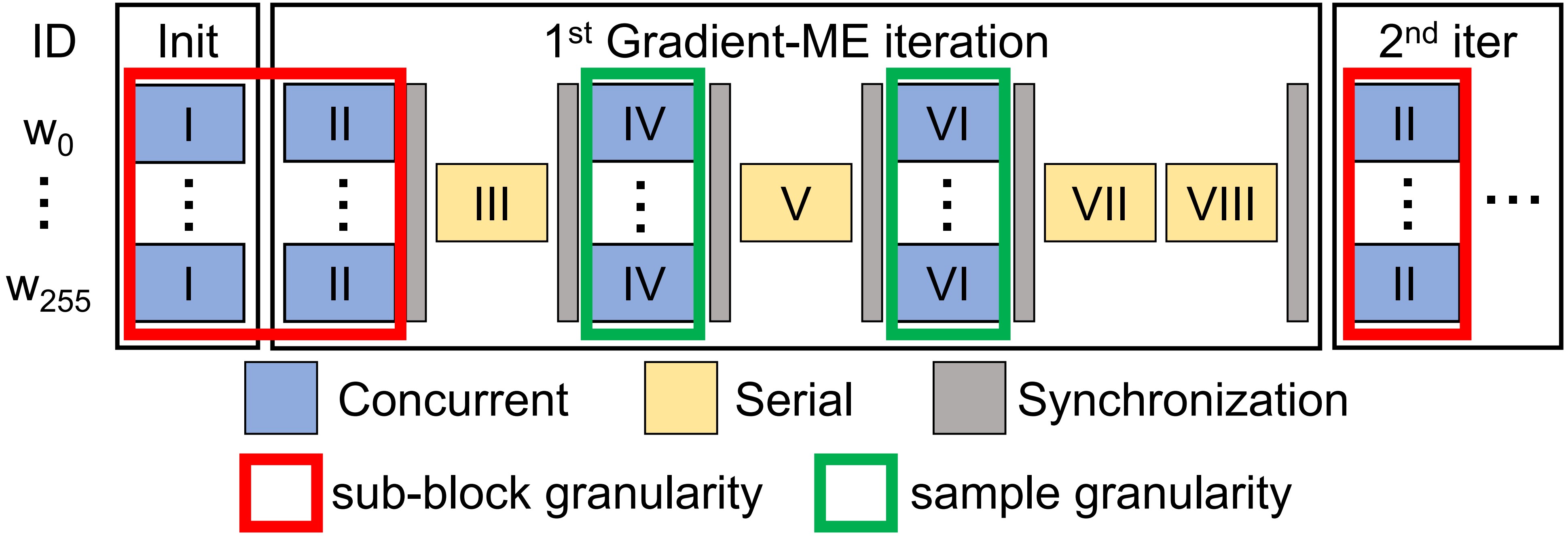
PROPOSED MODEL - DETAILS



Stage VII – Serial

W_0 reduces all partial systems into a single one

PROPOSED MODEL - DETAILS

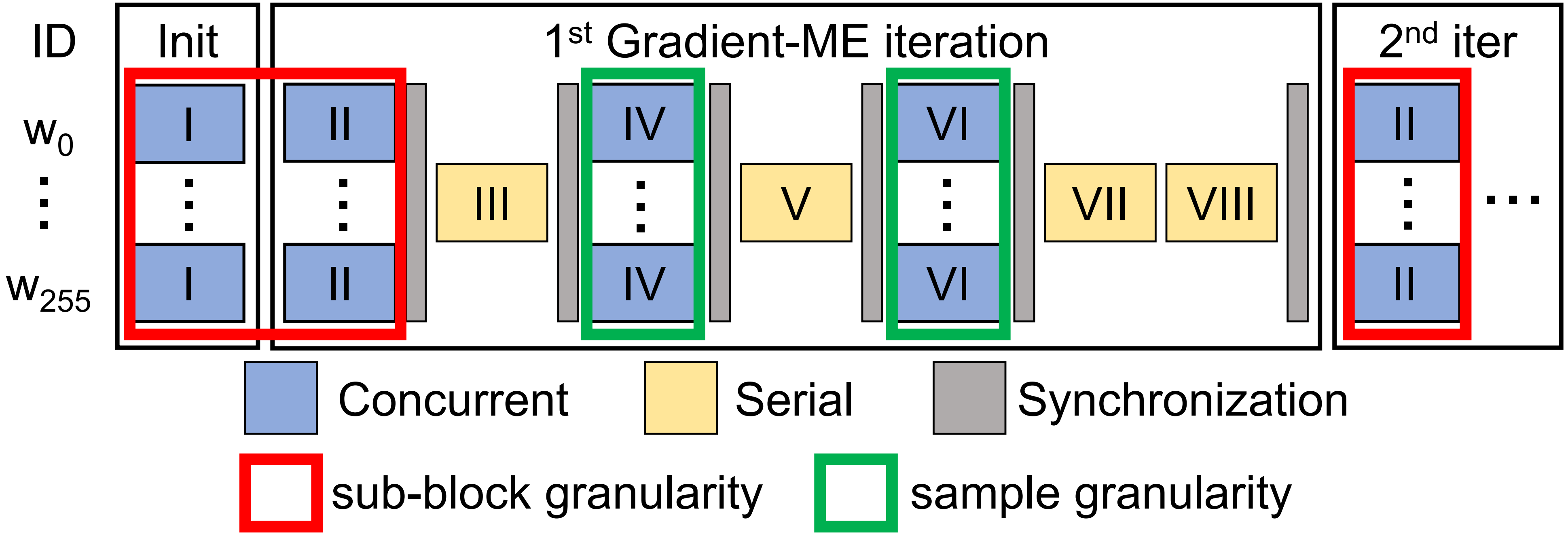


Stage VIII – Serial

W_0 solves the system by least-squares minimization

System solution used to update the CPMVs

PROPOSED MODEL - DETAILS



NOTEWORTHY

Stages II and III and conducted a sixth time at the end to verify the distortion of the last CPMVs

EXPERIMENTAL RESULTS

- **Five 1080p videos** → 1 workgroup per CU, 120 workgroups
- **Speedup**
  - Measured in relation to **AMVP + Gradient-ME + Refinement**
  - Versus **VTM with and without SIMD** optimizations
  - Considers only time of **128x128 CUs and 2 CPs**
- **Coding efficiency** → BD-BR in relation to VTM-CPU encoder
  - **SIMD** does not interfere on **coding efficiency**

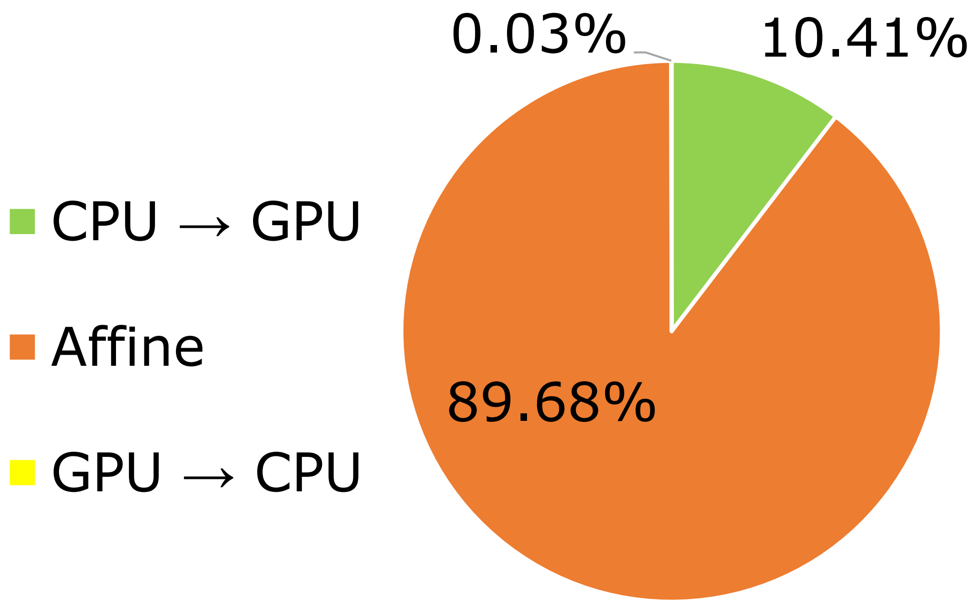
EXPERIMENTAL RESULTS

Speedup and coding efficiency results

Sequence	**Speedup (vs Serial)**	**Speedup (vs SIMD)**	**BD-BR**
BasketballDrive	63.24	33.92	0.07%
BQTerrace	42.22	23.40	0.32%
Cactus	43.81	25.40	0.02%
MarketPlace	70.30	36.27	0.31%
RitualDance	66.46	34.67	0.06%
Average	**57.21**	**27.58**	**0.16%**

EXPERIMENTAL RESULTS

- **Proposed** method comprises **3 shares**
  - **CPU → GPU:** Move original and reference frames from CPU to GPU
  - **Affine:** Conduct affine prediction on GPU
  - **GPU → CPU:** Return the CPMVs and costs from GPU to CPU



EXPERIMENTAL RESULTS

Comparison to related works

Work	Stage – Encoder	Speedup	BD-BR
Proposed	**Affine – VVC**	**57.21[ac], 27.58[bc]**	**0.16%**
Luo @ IEEE TMM2019	ME – HEVC	12.71[ac]	0.52%
Xiao @ IEEE TMM2019	ME – HEVC	11.26[ad]	0.10%
Grossi @ JRTIP 2018	ME – VP9	2.3[bd]	N.A.
Park @ IEEE Access 2019	Affine – VVC	1.6[bc]	0.10%

[a]Speedup vs Serial [b]Speedup vs SIMD [c]Speedup of Affine/ME [d]Speedup of whole encoder

EXPERIMENTAL RESULTS

Comparison to related works

Work	Stage – Encoder	Speedup	BD-BR
Proposed	**Affine – VVC**	**57.21[ac], 27.58[bc]**	**0.16%**
Luo @ IEEE TMM2019	ME – HEVC	12.71[ac]	0.52%
Xiao @ IEEE TMM2019	ME – HEVC	11.26[ad]	0.10%
Grossi @ JRTIP 2018	ME – VP9	2.3[bd]	N.A.
Park @ IEEE Access 2019	Affine – VVC	1.6[bc]	0.10%

[a]Speedup vs Serial [b]Speedup vs SIMD [c]Speedup of Affine/ME [d]Speedup of whole encoder

CONCLUSION

- Affine prediction raises **novel parallelization challenges and opportunities**
- **Leveraging knowledge** of **application** and **GPU architecture** allows efficient modeling
- Significant speedup with minor coding efficiency losses
- Idea is **generic for any block size**, and both CPs

THANK YOU!

Iago Storch, Daniel Palomino, Sergio Bampi
icstorch@inf.ufrgs.br

Graduate Program in Computing (PPGC)
Federal University of Rio Grande do Sul (UFRGS)