CAPRI

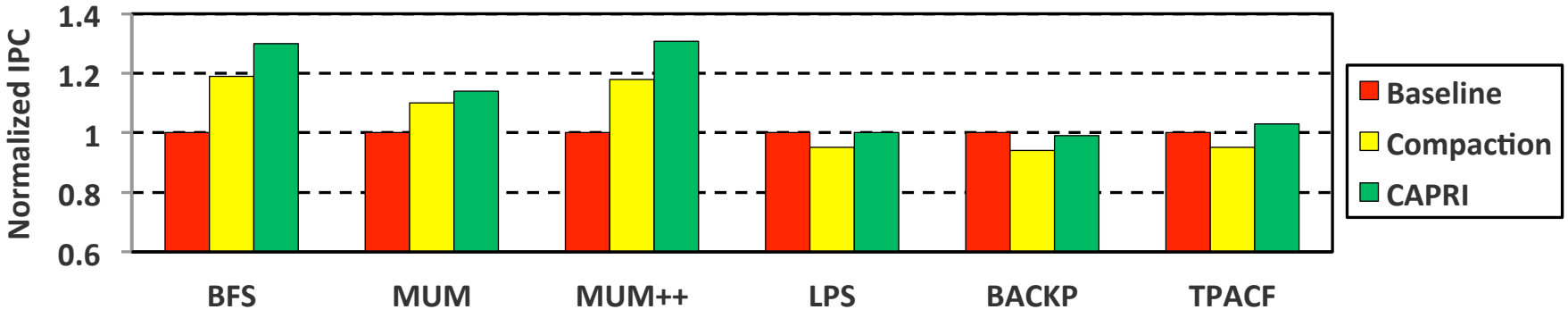
Prediction of Compaction-Adequacy for Handling Control-Divergence in GPGPU Architectures

Minsoo Rhu and Mattan Erez

The University of Texas at Austin
Electrical and Computer Engineering Dept.

CAPRI: Compaction-Adequacy Prediction

- Dynamic SIMD-lane compaction
  - Enhances SIMD lane utilization from control divergence
  - Fills idle lanes in one SIMD group with threads from others
- Problem
  - Compaction often ineffective
  - Always imposes compaction-barrier at all branches
- Our solution
  - Predict whether compaction will be beneficial or not
  - Mitigate the detrimental impact of extra barriers

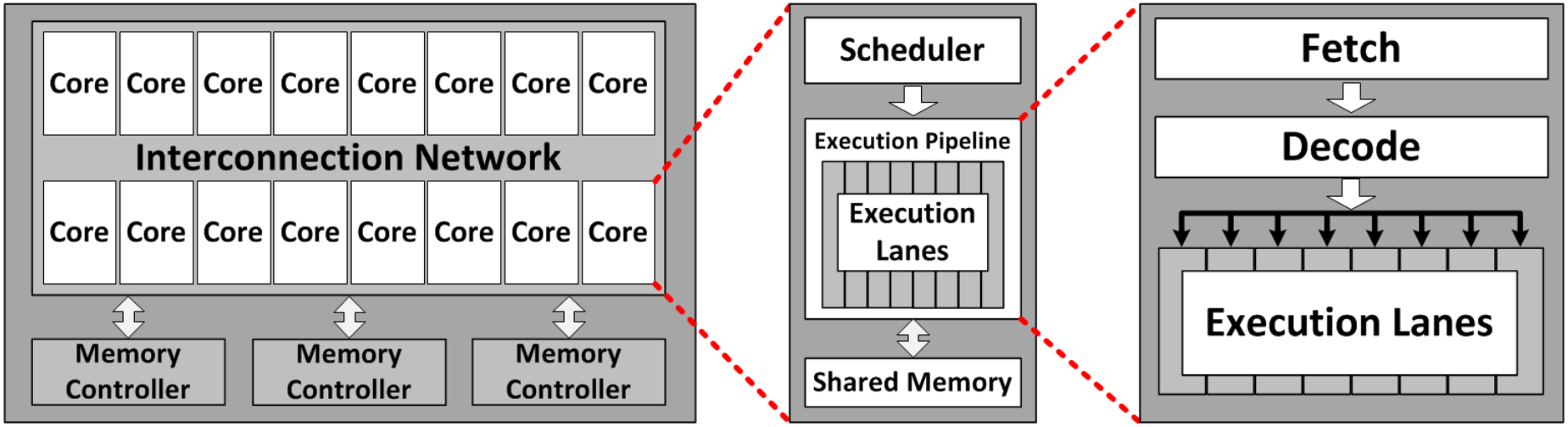


Outline

- GPU and SIMD compaction background
- Compaction can degrade performance
- CAPRI – **c**ompaction **a**dequacy **pr**edi**i**ction
- Evaluation

Graphic Processing Units (GPUs)

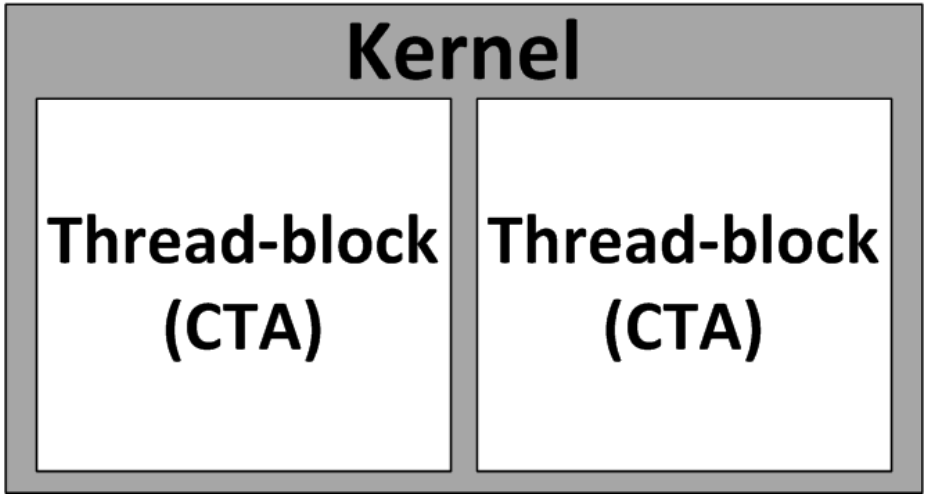
- General-purpose many-core accelerators
  - Supports non-graphics APIs (e.g. CUDA, OpenCL)
- Scalar frontend (fetch & decode) + parallel backend
  - Amortizes the cost of frontend and control



CUDA exposes hierarchy of data-parallel threads

- **SPMD model**: single ***kernel*** executed by all threads

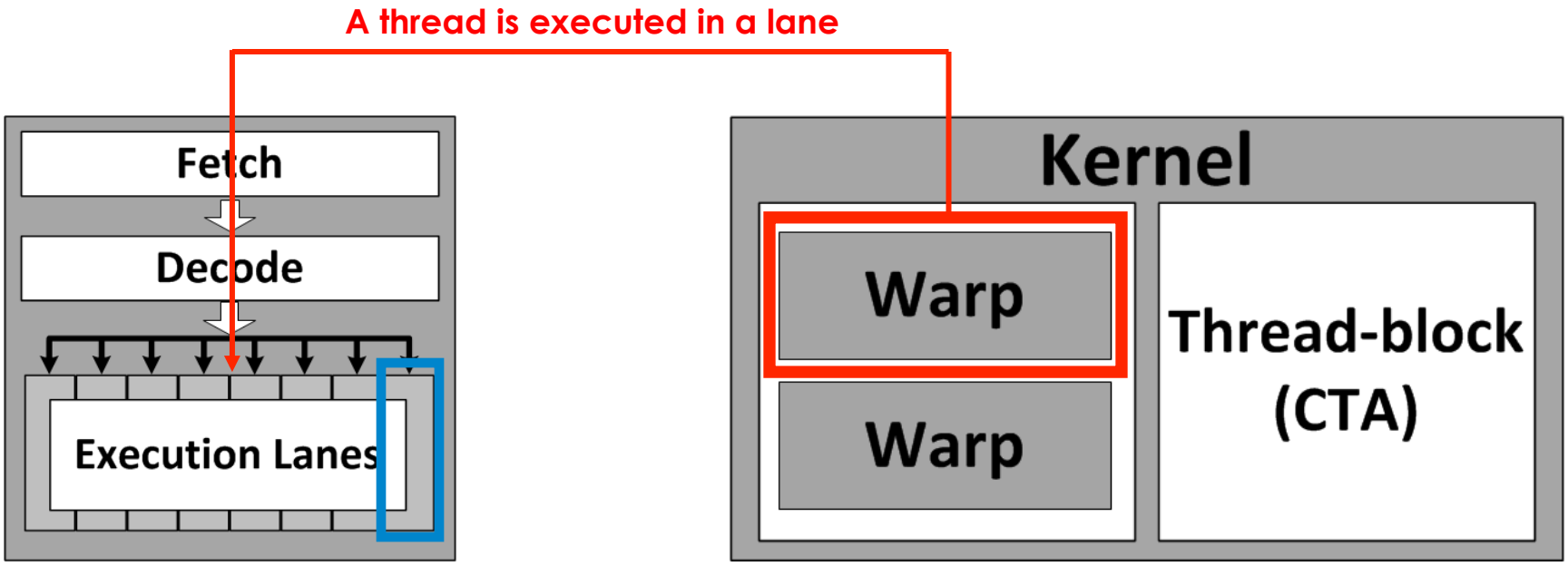
- **Kernel / Thread-block**
  - Multiple **thread-blocks** (concurrent-thread-arrays(**CTA**s)) compose a **kernel**



CUDA exposes hierarchy of data-parallel threads

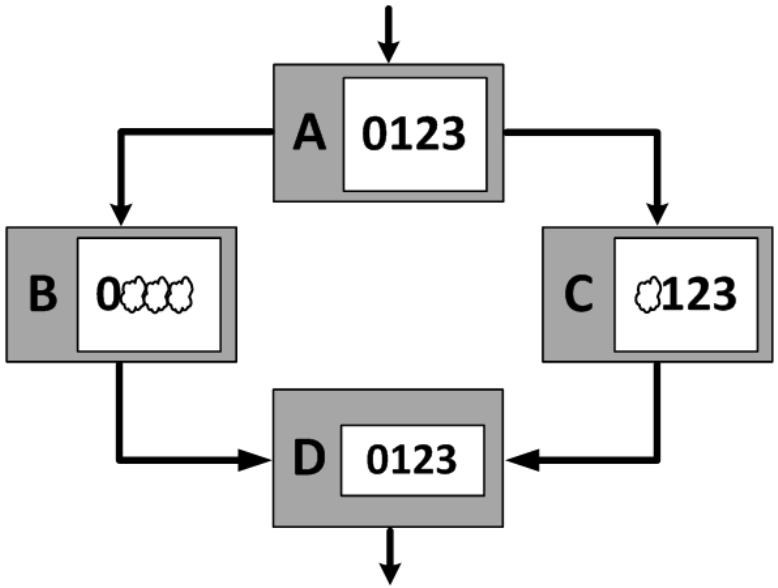
- **SPMD model**: single ***kernel*** executed by all threads

- **Kernel / Thread-block / *Warp***
  - Multiple **warps** compose a **thread-block**
  - Multiple **threads (32)** compose a warp



GPUs have HW support for conditional branches

- Control-divergence: threads in warp branch differently
  - **Predicate-registers** : only a subset of the warp commits results
  - Stack-based re-convergence model
  - Always execute ***active***-threads at the ***top-of-stack* (TOS)**.
  - Reconvergence PC (RPC) derived at compile-time



(a) Example control flow graph

TOS →

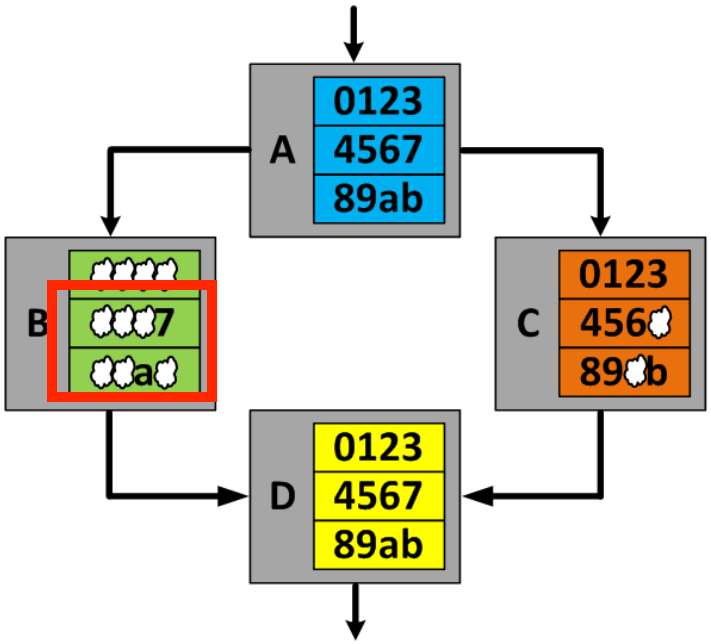
HW-Stack		
RPC	Active Mask	Next PC
-	0123	D
D	123	C
D	0	B

(b) Per-warp re-convergence stack

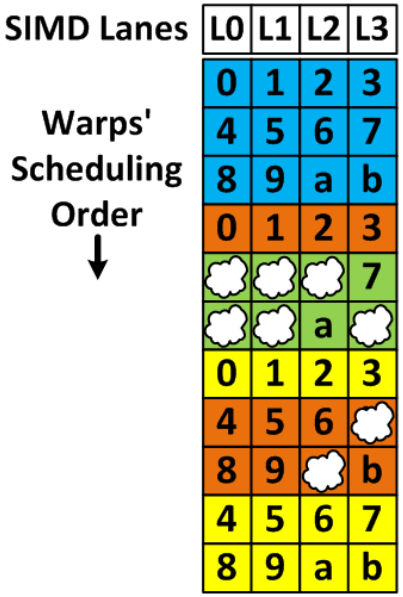
: Threads (lanes) masked out from exeuction

Thread-block compaction (TBC) [Fung11]

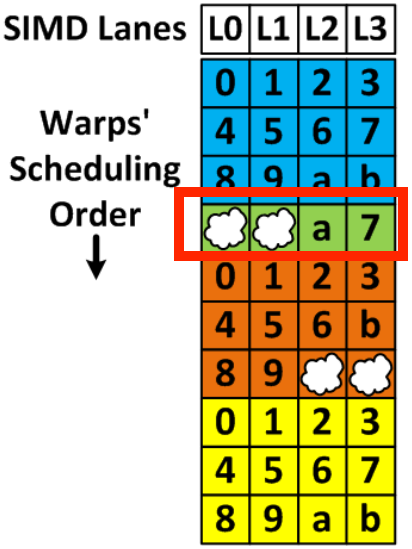
- Dynamically ***compact*** warps within a ***thread-block***
  - ***Synchronize*** all warps at conditional-branches and reconvergence points
  - Adopts a ***CTA-wide*** re-convergence stack for sync
  - Similar approach in [Narasiman11] – **TBC+**



(a) Example control flow graph



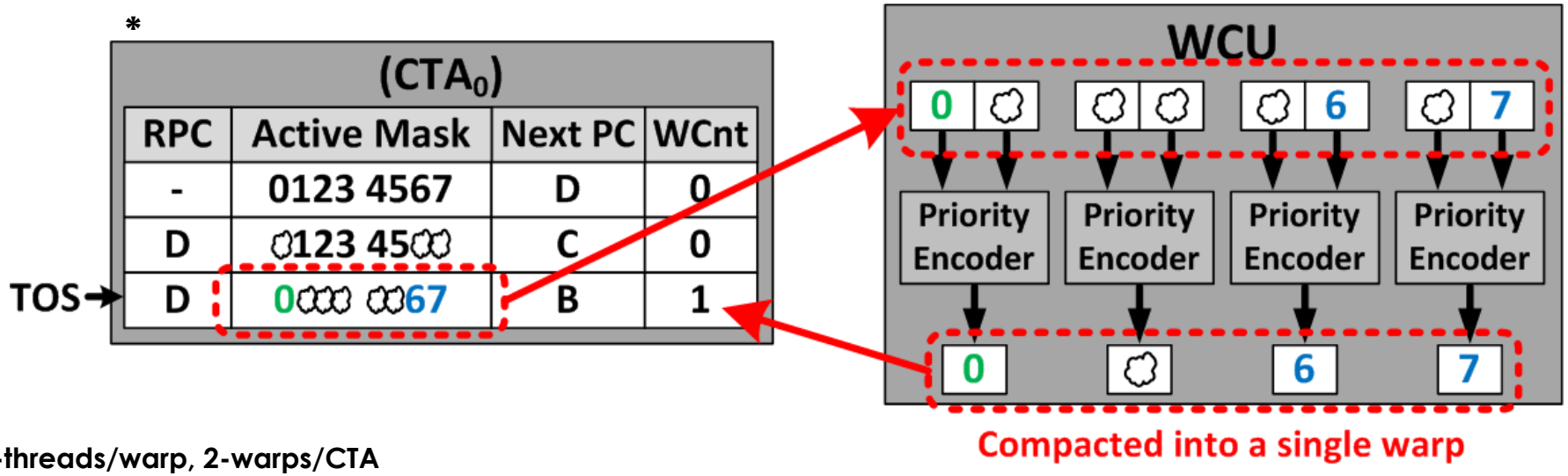
(b) Execution flow without compaction



(c) Execution flow with compaction

Thread-block compaction (TBC)

- CTA-wide re-convergence stack
  - Each warp incrementally updates the active bitmasks
  - Enforces HW-generated compaction-barriers
- Warp compaction unit (WCU)
  - Composed of a set of priority encoders
  - Generates one valid warp per cycle (after compaction)
  - Only threads not executing in a common lane are compacted into the same warp



* 4-threads/warp, 2-warps/CTA

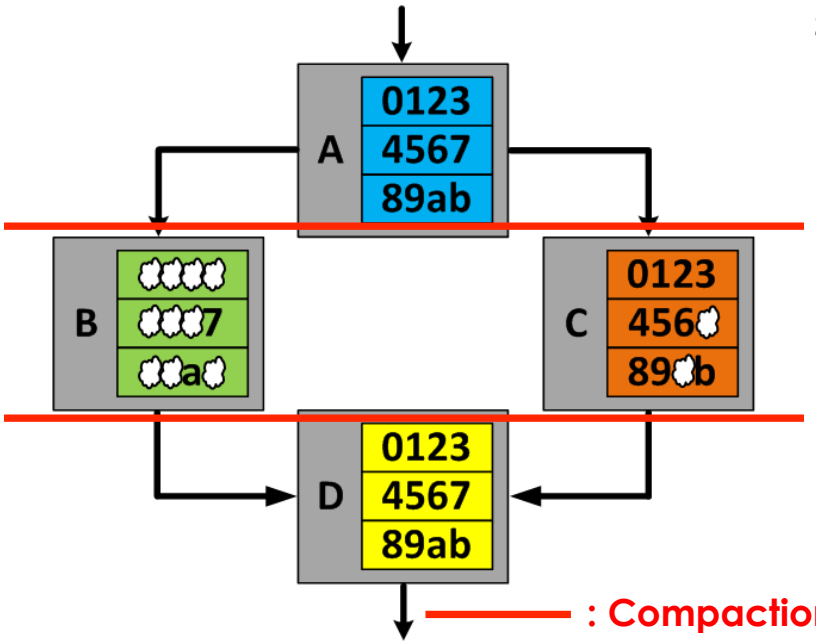
Outline

- GPU and SIMD compaction background
- **Compaction can degrade performance**
- CAPRI – **c**ompaction **a**dequacy **pr**ed**i**ction
- Evaluation

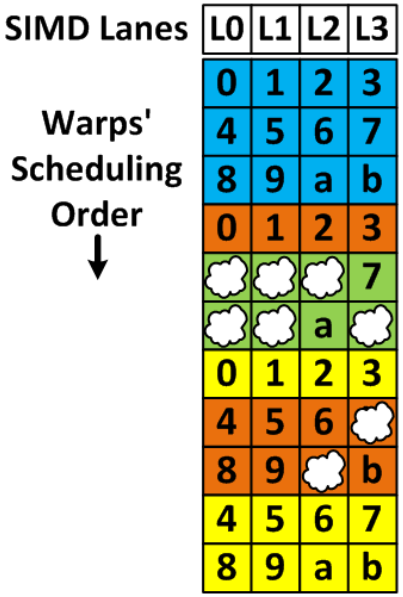
Problem 1: extra synchronization overhead

- Compaction is applied at ***all*** branching points
  - A branch's divergence is determined after execution
  - Not all branches are amenable to compaction
  - Synchronization is imposed regardless

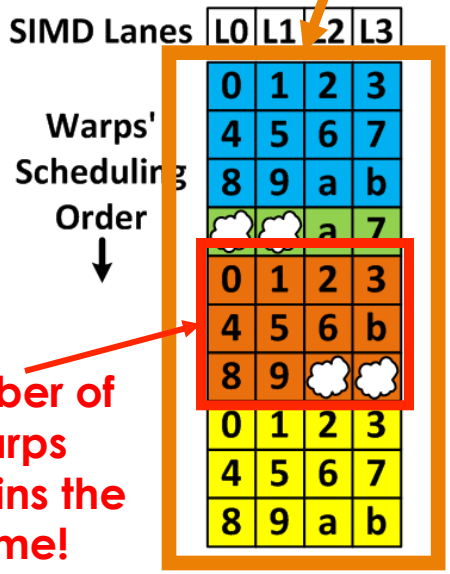
Warps are always forced to execute only within a basic-block



(a) Example control flow graph



(b) Execution flow without compaction

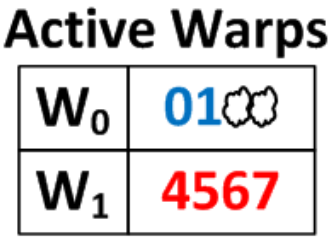


(c) Execution flow with compaction

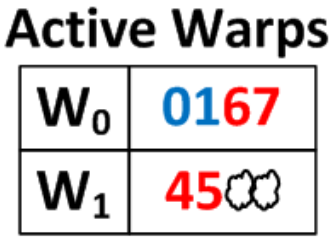
Problem 2: increased memory divergence

- Threads within a warp shuffled after compaction
  - Compaction unit "slides" threads up
  - May break coalesced memory accesses in original order

- Compaction is worthwhile only when the application is highly irregular



(a) Total number of warps stalled: *one* (Without compaction)



(b) Total number of warps stalled: *two* (With compaction)

Outline

- GPU and SIMD compaction background
- Compaction can degrade performance
- **CAPRI – compaction adequacy prediction**
- Evaluation

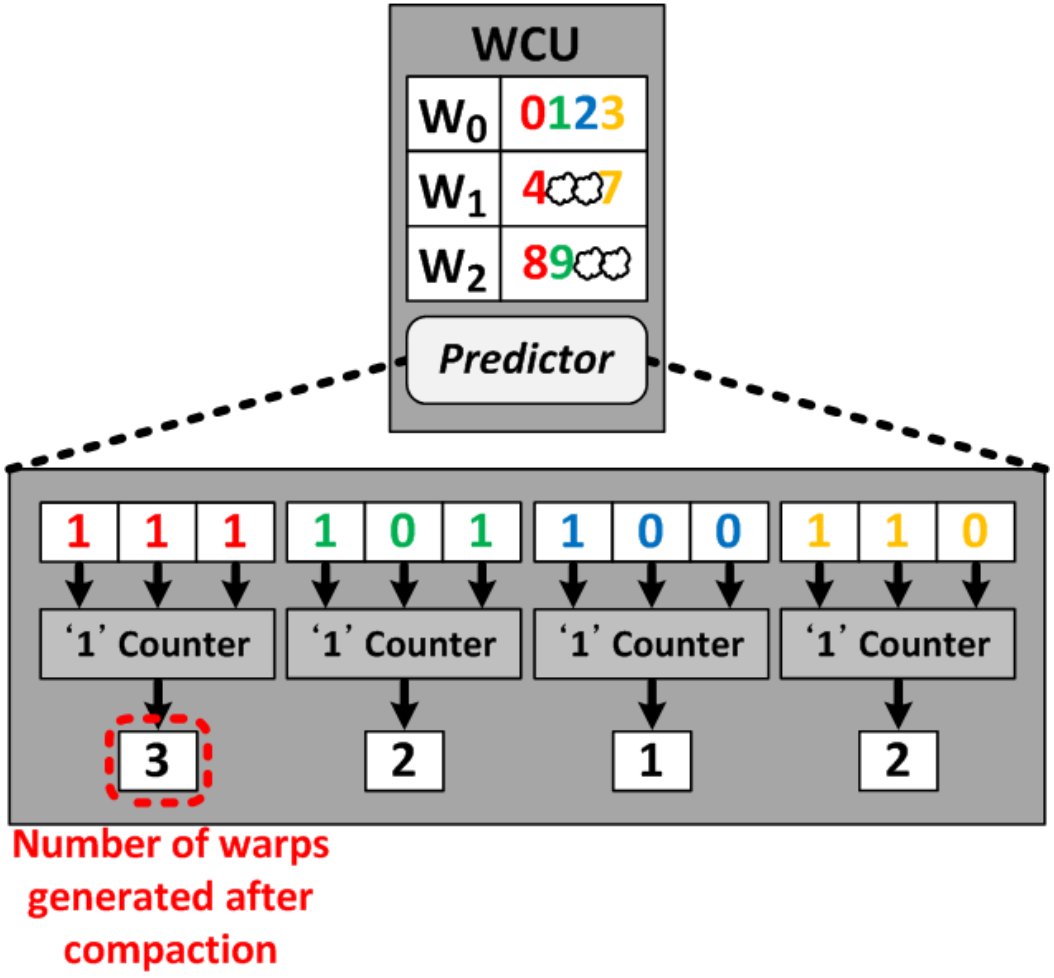
CAPRI: Compaction-Adequacy PRedIction

- Intuition
  - Not all branch points are likely to be compactable
  - Activate compaction ***only when*** past history indicates compaction was adequate
  - ***Bypass*** warps from compaction-barrier when inadequate

- Compaction-adequacy
  - Compaction is ***adequate*** when the number of executing warps is reduced

Compaction-adequacy assessment

- Check if compaction was beneficial
  - Number of warps before/after compaction are compared



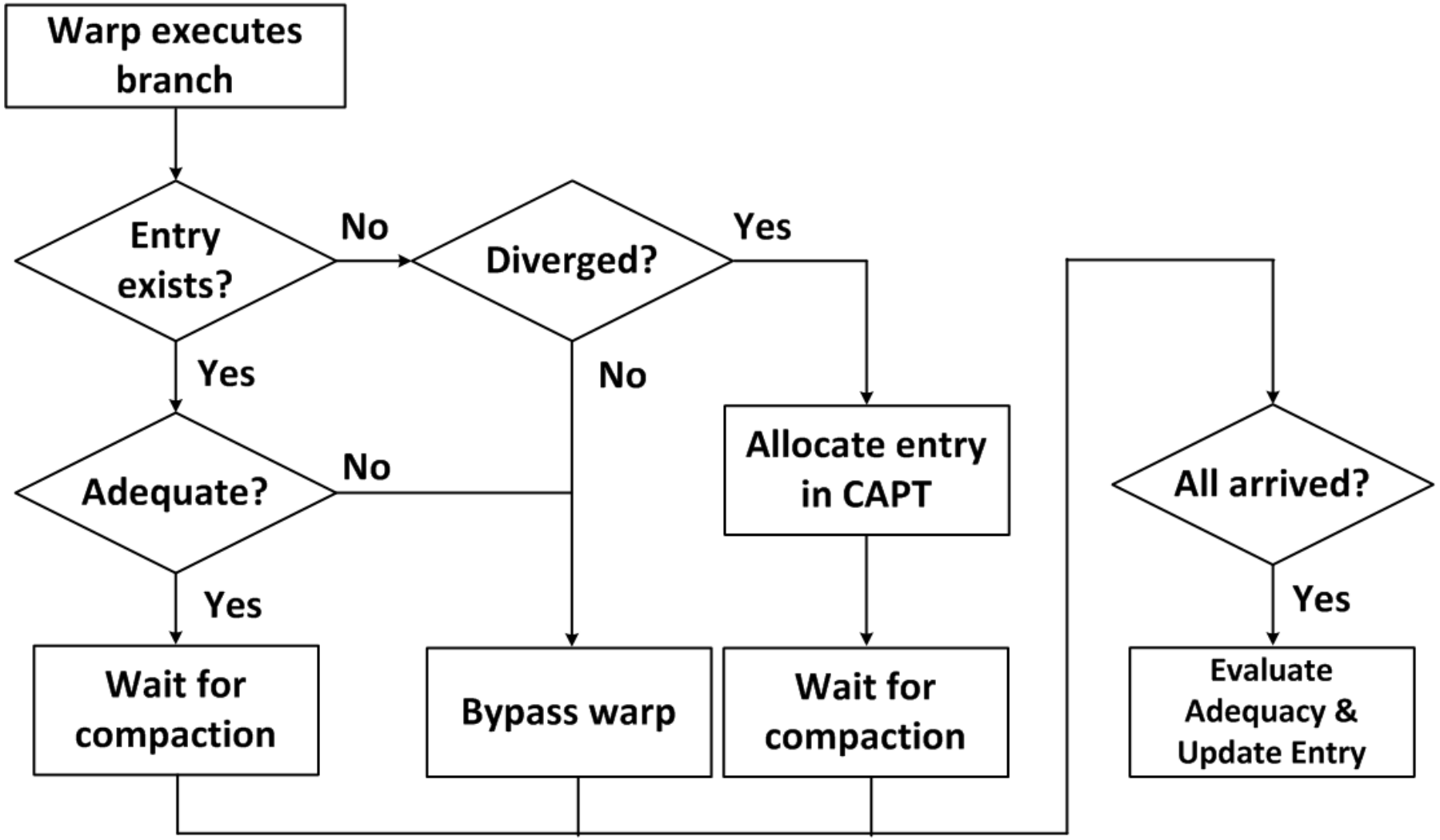
Microarchitecture of CAPRI

- Design
  - Compaction-Adequacy Prediction Table (CAPT)
  - Single CAPT per shader core (shared among CTAs within core)
  - Fully-associative structure (32-entries)
    - 8-entries typically sufficient
  - Each entry contains a history-bit for compaction-adequacy
  - Tag: PC of Branch instruction (BADDR)

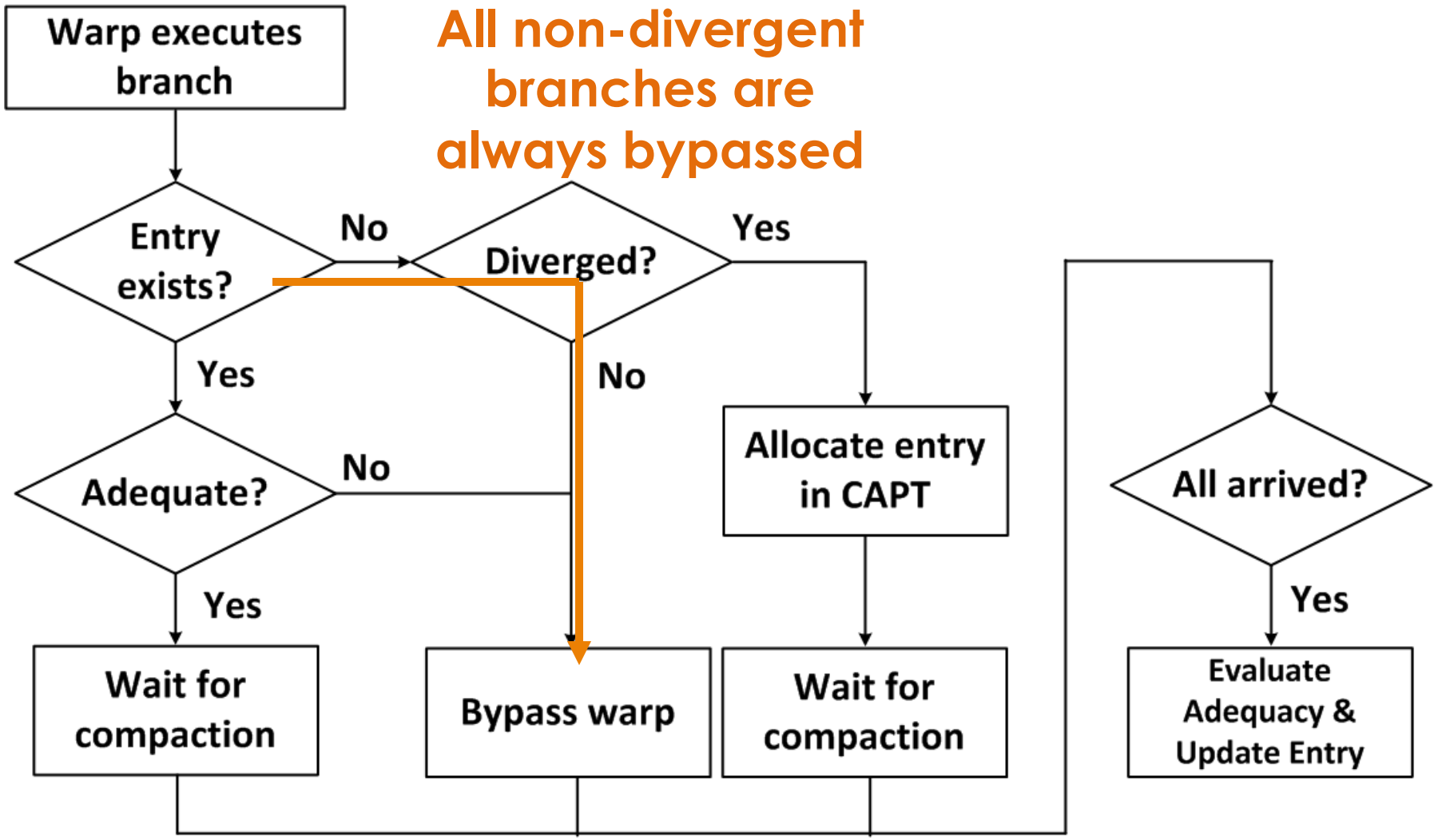
- History-bit configuration
  - Most recently evaluated compaction-adequacy

CAPT		
Valid	History	BADDR
1	1	BR_{BC}
0	0	-
0	0	-

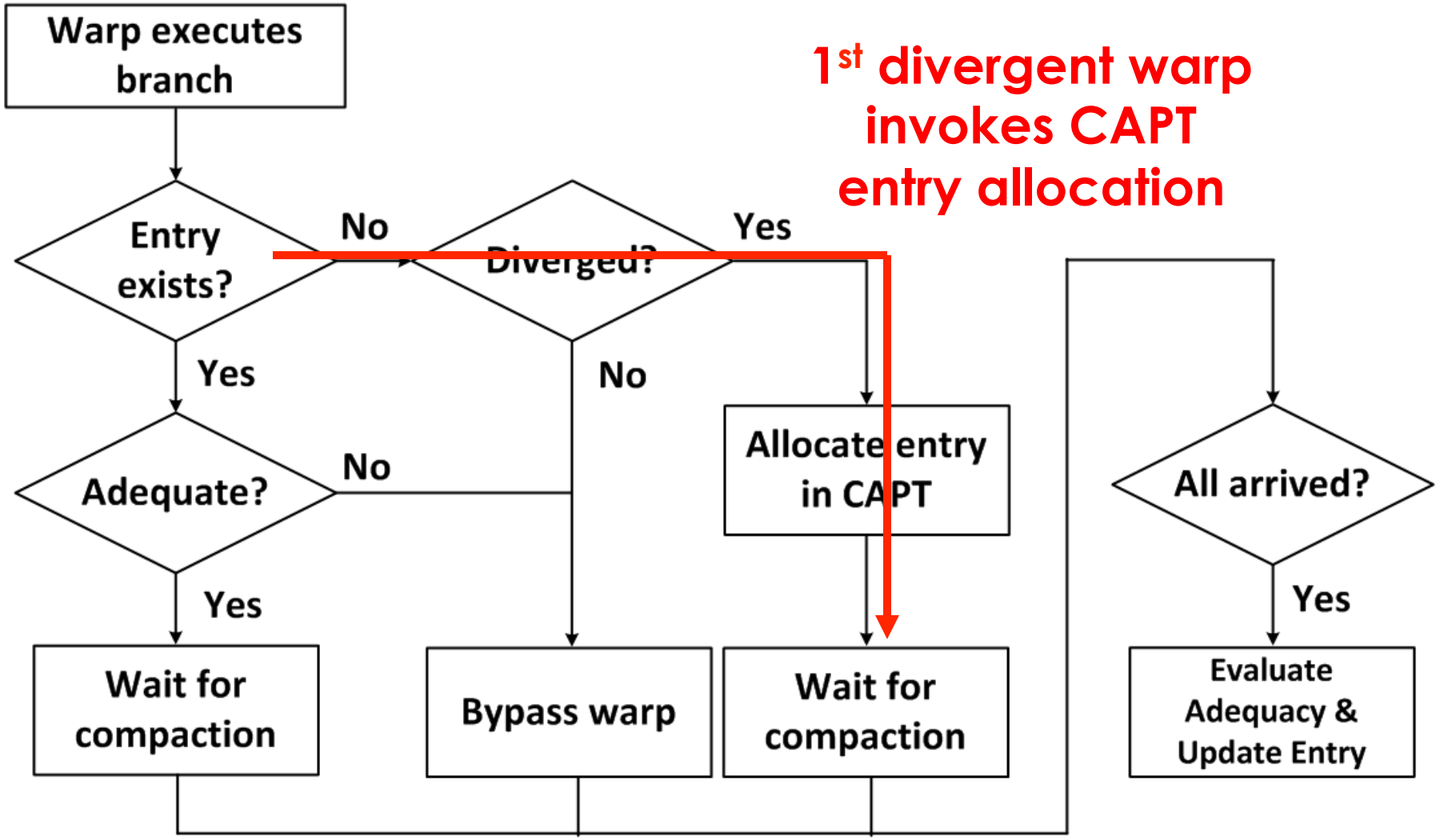
CAPRI Algorithm



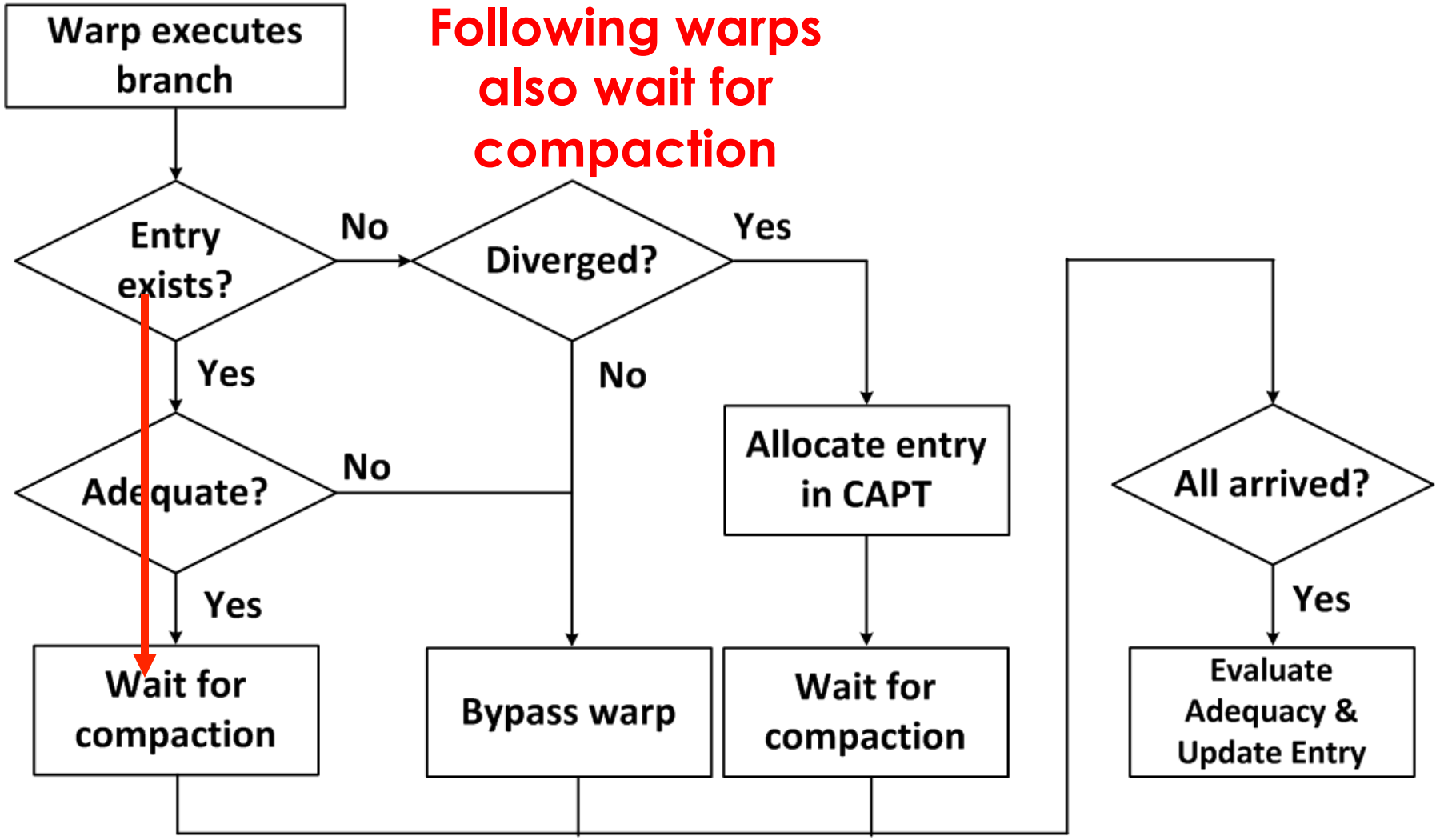
CAPRI Algorithm



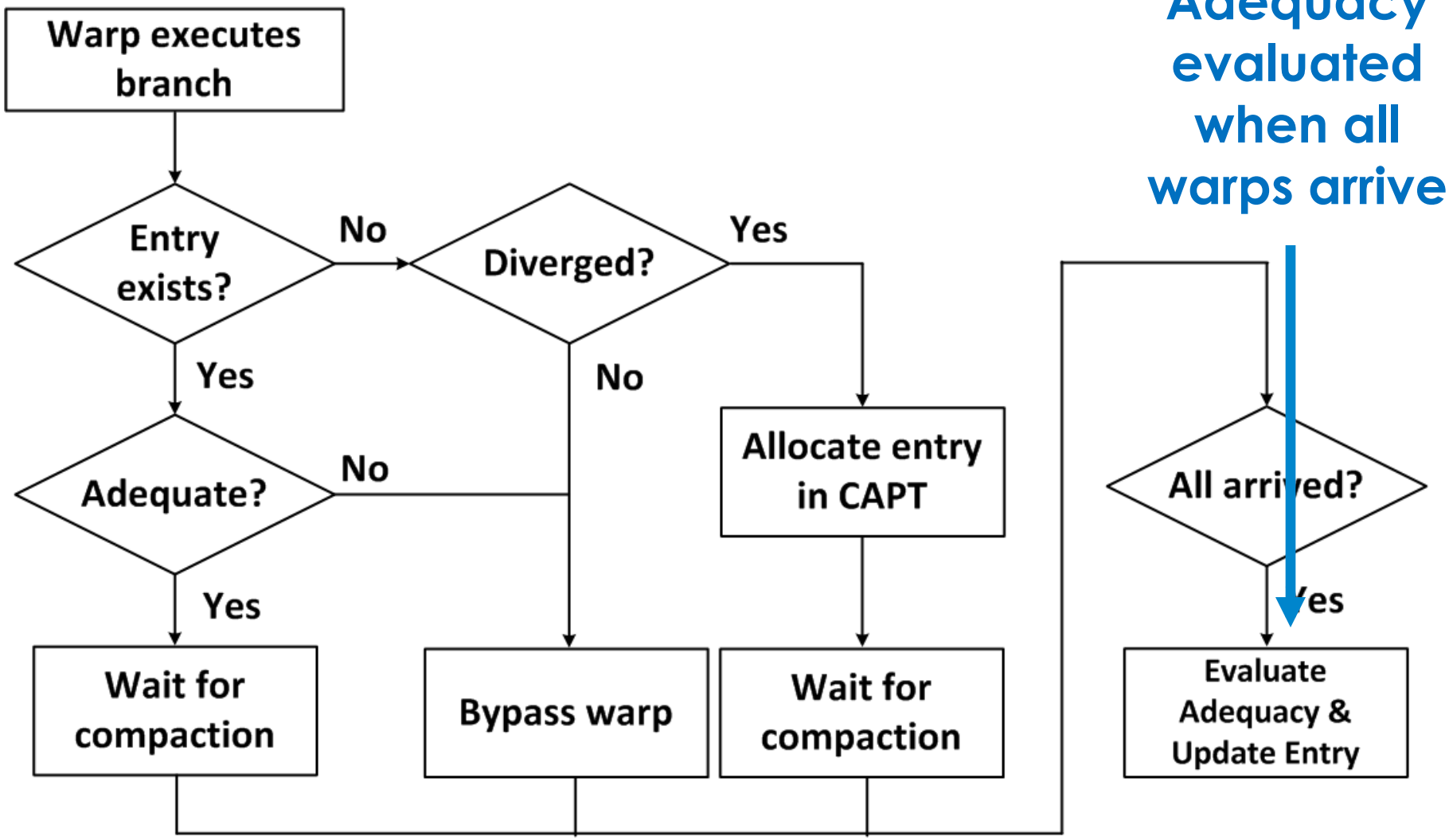
CAPRI Algorithm



CAPRI Algorithm



CAPRI Algorithm



Compaction Adequacy evaluated when all warps arrive

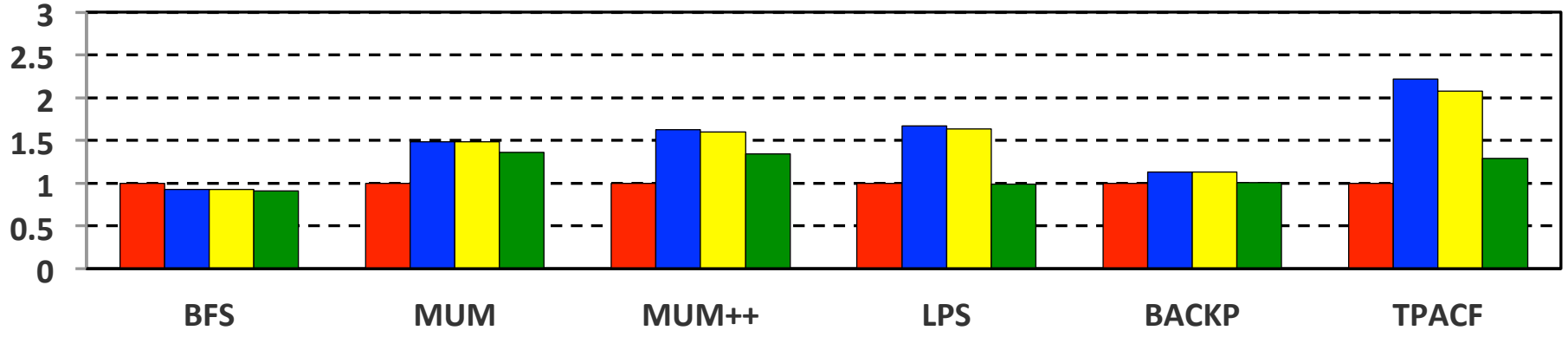
Outline

- GPU and SIMD compaction background
- Compaction can degrade performance
- CAPRI – **c**ompaction **a**dequacy **pr**ed**i**ction
- **Evaluation**

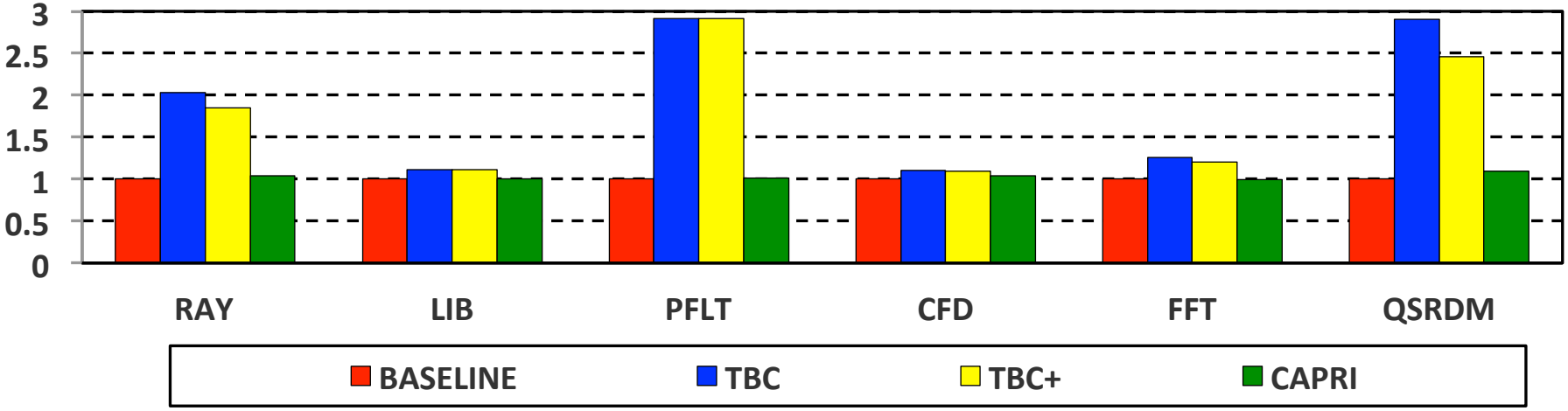
Simulation Environment

- GPGPU-Sim (v2.1.1b)
  - 30 shader cores (Streaming Multiprocessors)
  - 1024 threads per core, 16K registers per core
  - Cache: 32kB L1, 1024kB Unified L2
  - Warp scheduling policy: Round-Robin
  - Memory Controller: FR-FCFS
- Workloads
  - Regular/Irregular apps with various input-sets
  - Chosen from CUDA-SDK(v2.2), Rodinia, Parboil, etc

Idle Cycles (Normalized)

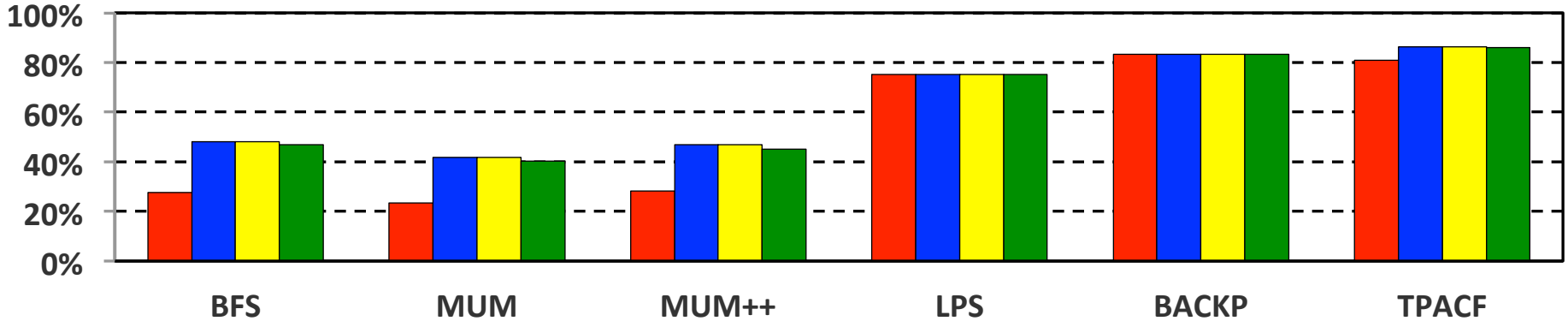


(a) Divergent Benchmarks

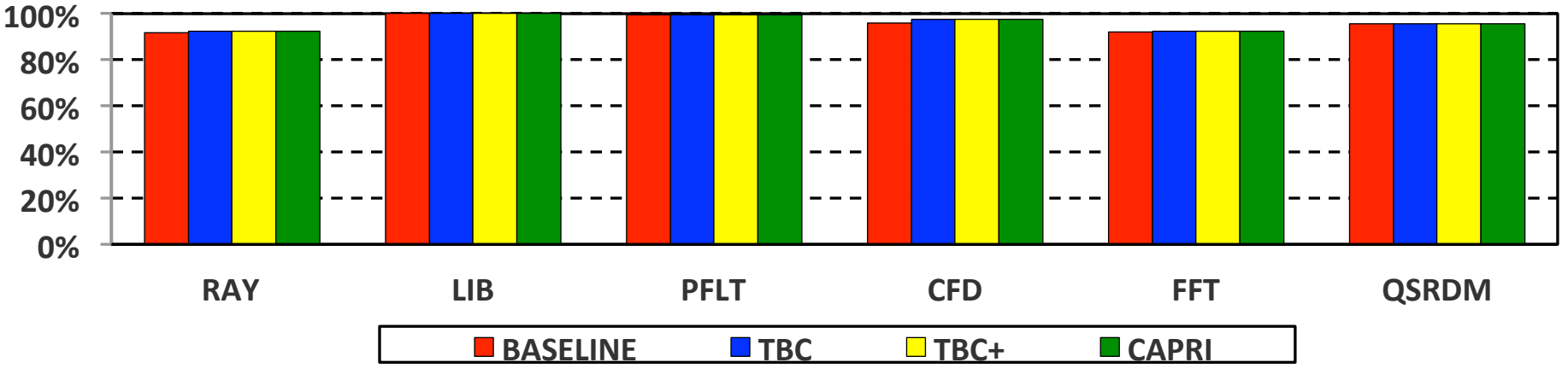


(b) Non-divergent Benchmarks

SIMD lane utilization*



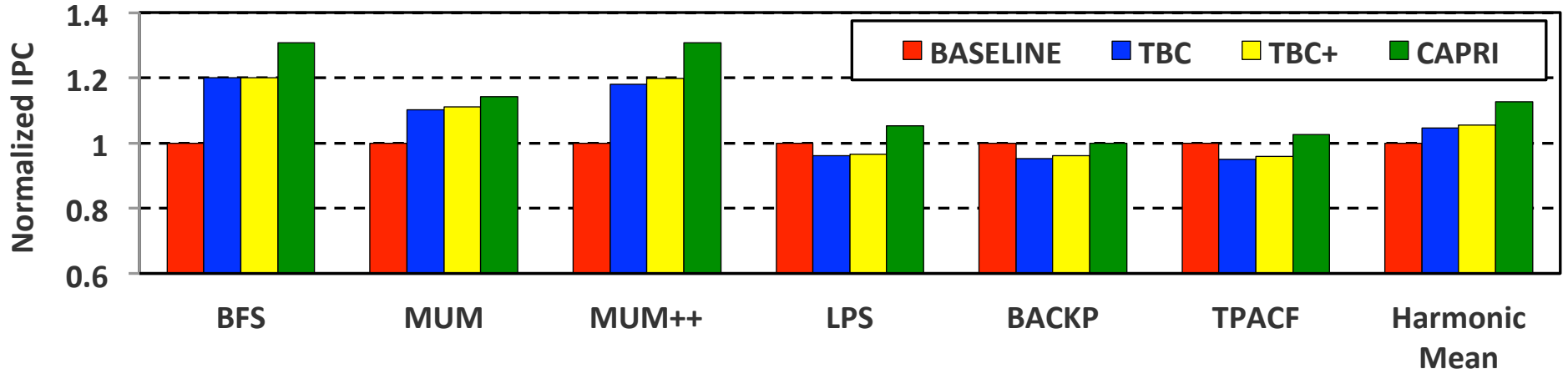
(a) Divergent Benchmarks



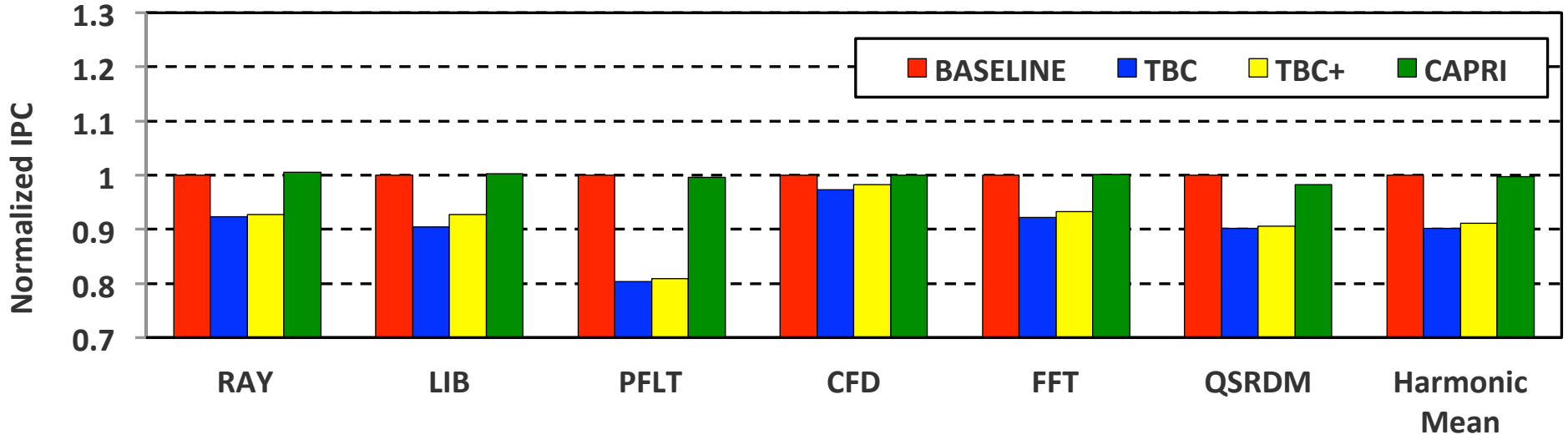
(b) Non-divergent Benchmarks

* Average number of lanes utilized when a warp is issued for execution

Overall Performance



(a) Normalized IPC (Divergent)



(b) Normalized IPC (Non-divergent)

Conclusions

- CAPRI provides the best of both baseline and TBC
  - Higher resource utilization and minimized synchronization
- Throughput improvements
  - **Divergent**: 7.6% (max 10.8%) improvements on top of TBC
  - **Non-divergent**: Avoids the average 10.1% performance degradation of TBC
  - Robust to scheduling policy thanks to bypassing
    - Unlike TBC and TBC+
- Implementation cost is negligible
  - 32-entry prediction-table per shader core