2023 Design, Automation & Test in Europe Conference (DATE 2023)

# TURBULENCE: Complexity-effective Out-of-order Execution on GPU with Distance-based ISA

Reoma Matsuo, Toru Koizumi, Hidetsugu Irie, Shuichi Sakai, and Ryota Shioya
The University of Tokyo, Tokyo, Japan
matsuo@rsg.ci.i.u-tokyo.ac.jp, koizumi, irie, sakai@mtl.t.u-tokyo.ac.jp, shioya@ci.i.u-tokyo.ac.jp

***Abstract*—A graphic processing unit (GPU) is a processor that achieves high throughput by exploiting data parallelism. We found that many GPU workloads also contain instruction-level parallelism, which can be extracted through out-of-order execution to provide additional performance improvement opportunities. We propose the TURBULENCE architecture for very low-cost out-of-order execution on GPUs. TURBULENCE consists of 1) a novel ISA that introduces the concept of referencing operands by inter-instruction distance instead of register numbers and 2) a novel microarchitecture that executes the novel ISA. Our proposed ISA and microarchitecture enable cost-effective out-of-order execution on GPUs without introducing expensive hardware.**

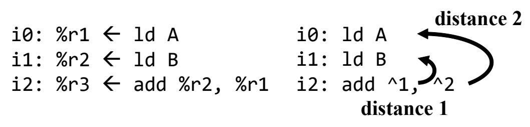
***Index Terms*—GPU, microarchitecture, instruction-level-parallelism, out-of-order execution, energy efficiency**

## I. Introduction

Throughput-intensive graphic processing units (GPUs) are widely used in various fields. A typical GPU has single-instruction multiple-data (SIMD) pipelines with many functional units while minimizing the control overhead [1], [2]. One of the key features of GPUs to efficiently use these many functional units is latency hiding through multi-threaded execution. GPUs can hide latency by executing other threads while instructions with long latencies are being executed. With this multi-threading, GPUs can effectively hide long latencies such as memory accesses and floating-point operations to achieve high throughput [3], [4].

On such GPUs, instructions in each thread are generally executed in-order and not out-of-order. Out-of-order execution is widely used in high-performance CPUs because it can effectively hide the latency of each instruction within a thread and improve performance. However, out-of-order execution requires very expensive circuits, such as renaming logic, reorder buffers (ROB), and load-store queues (LSQ) [5]–[7]. If out-of-order execution were implemented on GPUs, these costly units would be needed in proportion to the number of threads, making the circuits unrealistically large. As a result, GPUs generally focus on hiding latency through multi-threaded execution and do not introduce expensive out-of-order execution that hides latency within threads.

In contrast, we found that GPU workloads also contain available instruction-level parallelism, which can be extracted through out-of-order execution to provide performance improvement opportunities. We propose the TURBULENCE architecture, which introduces low-cost out-of-order execution to GPUs and improves performance without compromising energy efficiency. TURBULENCE consists of 1) a novel ISA that introduces the concept of referencing operands by inter-instruction distance instead of register numbers and 2) a novel

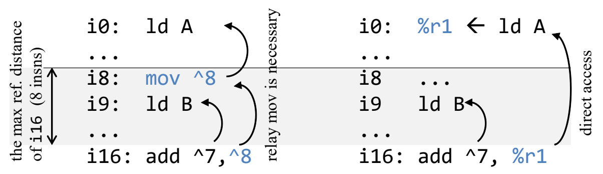


```
i0: %r1 ← ld A
i1: %r2 ← ld B
i2: %r3 ← add %r2, %r1
```

```
i0: ld A
i1: ld B
i2: add ^1, ^2
```

(a) Reg. Number-based ISA (b) Distance-based ISA

Fig. 1. (a) shows a conventional ISA that specifies source operands by register numbers, and (b) shows the distance-based ISA that specifies source operands by distance. In (b), ˆN indicates that the distance to the source operand is N. Both programs have the same meaning.



```
i0:  ld A
...
i8:  mov ^8
i9:  ld B
...
i16: add ^7,^8
```

```
i0:  %r1 ← ld A
...
i8   ...
i9   ld B
...
i16: add ^7, %r1
```

(a) Existing Distance-based ISA (b) Novel Hybrid ISA

Fig. 2. Instruction sequence when the maximum reference distance is eight. (a) The second operand of i16 refers to the result of i0, but the distance from i16 to i0 exceeds the maximum reference distance, so i8 is inserted to relay the result. (b) The result of i0 is directly referenced by i16 using register number %r1, while the result of i9, which is located closer, is referenced by distance.

microarchitecture that executes the novel ISA. This distance-based operand has the property that it does not cause any false dependencies. Using this property, we achieve cost-effective out-of-order execution on GPUs without introducing expensive hardware such as a rename logic and a load-store queue. The simulation results show that TURBULENCE improves performance by 17.6% without an increase in energy consumption compared to an existing GPU.

## II. TURBULENCE

### A. Hybrid ISA

The distance-based ISA is one originally proposed for high-performance CPUs, which specifies operands using inter-instruction distances instead of register numbers [6], [8]. Figure 1 shows an example of the distance-based ISA, along with that of an existing register number-based ISA. Because of the distance-based representation of operands, the distance-based ISA does not produce any false dependencies, and thus register renaming can be omitted [6], [8].

However, existing distance-based ISA can significantly increase the number of instructions, which in turn can reduce the performance. This increase is due to reasons such as the insertion of relay instructions to refer to results generated by distant instructions as shown in Figure 2 (a) [8].

We propose a novel hybrid ISA, which uses register numbers instead of distances for specific operands with long reference

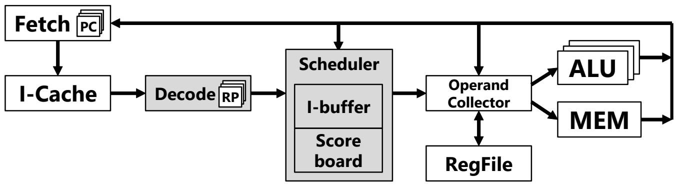


Fig. 3. The main modifications from existing GPUs are in the decoder and scheduler.

distances as shown in Figure 2 (b). Accessing operands through register numbers must be performed in-order, but GPUs can effectively hide the latency by multithreading. As a result, the number of instructions can be reduced while minimizing the performance impact.

## B. Microarchitecture

Our proposed TURBULENCE can achieve out-of-order execution without significantly changing the microarchitecture of existing GPUs as shown in Figure 3. The reasons for the low-cost out-of-order execution of our method are because 1) the GPU scheduler has a structure similar to that of the out-of-order execution CPU scheduler [9], and 2) the distance-based ISA enables omitting register renaming and recovering from exception simply. TURBULENCE also solves the memory ordering problem caused by out-of-order execution at low cost by using existing register number-based instructions as memory barrier instructions.

## III. Evaluation

### A. Methodology

We implemented and evaluated TURBULENCE in GPGPU-Sim 4.0.1 [10]. A GPU configuration used in the evaluation is based on NVIDIA RTX 2060. Eight applications from CUDA Samples [11] and Rodinia [12] were used for the evaluation. The PTX-format code compiled from each CUDA application was manually converted to TURBULENCE ISA. We used GPUWattch [13] to evaluate energy consumption.

### B. Results

Figure 4 shows the performance improvement of TURBULENCE over BASELINE for the entire execution in each application. TURBULENCE was measured by varying the instruction scheduling width per thread from 2 to 16. TURBULENCE outperforms BASELINE, which represents a baseline model based on RTX 2060, by more than 5% in all benchmarks and by 17.6% on geometric mean when the scheduling width is eight. In the following, we evaluate the performance with a scheduling width fixed at eight.

We evaluated the reduction in energy consumption of TURBULENCE compared to BASELINE. TURBULENCE consumes less energy in all benchmarks and consumes 6.1% less energy than BASELINE on average. TURBULENCE has a scheduler that is four times larger than that of BASELINE, but the scheduler consumes only 0.4% of the total processor energy consumption and does not have a significant impact. Although the scheduler in an out-of-order CPU can consume as much as 10% of the total processor energy [6], it has a

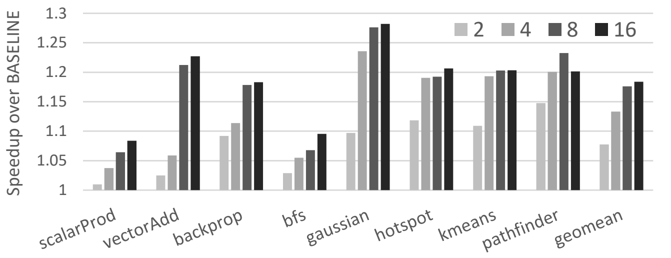


Fig. 4. Performance improvement with TURBULENCE when instruction window size is varied

very small impact on GPUs in our evaluation because the scheduler is shared among the 32-lane datapaths. In contrast, the performance improvement significantly reduces program execution time and static energy consumption, and thus, it reduces the total energy consumption.

## IV. Conclusion

We propose TURBULENCE that enables out-of-order execution on GPUs with little additional cost. TURBULENCE consists of 1) the novel ISA that overcomes the problems of existing distance-based ISA by utilizing the characteristics of GPUs and 2) the novel microarchitecture that uses almost the same mechanisms as existing GPUs. We implemented and evaluated TURBULENCE and showed that the proposed architecture improved performance by 17.6% and consumed 6.1% less energy compared to a GPU with the baseline configuration.

## Acknowledgment

This work was partilally supported by JSPS KAKENHI Grant Numbers JP20J23642, JP20H04153, and JP20J22752.

## References

[1] NVIDIA, "NVIDIA TURING GPU ARCHITECTURE," 2018. [Online]. Available: https://images.nvidia.com/aem-dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf

[2] AMD, "RDNA2 Instruction Set Architecture Manual," 2020. [Online]. Available: https://developer.amd.com/wp-content/resources/RDNA2_Shader_ISA_November2020.pdf

[3] S.-Y. Lee and C.-J. Wu, "Characterizing the latency hiding ability of gpus," in *ISPASS*, 2014, pp. 145–146.

[4] Y. Arafa, A.-H. A. Badawy, G. Chennupati, N. Santhi, and S. Eidenbenz, "Low overhead instruction latency characterization for NVIDIA gpgpus," in *HPEC*, 2019, pp. 1–8.

[5] A. Sembrant *et al.*, "Long term parking (LTP): Criticality-aware resource allocation in ooo processors," in *MICRO*, 2015, pp. 334–346.

[6] R. Shioya and H. Ando, "Energy efficiency improvement of renamed trace cache through the reduction of dependent path length," in *ICCD*, 2014, pp. 416–423.

[7] J. Shen and M. Lipasti, *Modern Processor Design: Fundamentals of Superscalar Processors*. McGraw-Hill Higher Education, 2002.

[8] H. Irie *et al.*, "STRAIGHT: Hazardless processor architecture without register renaming," in *MICRO*, 2018, pp. 121–133.

[9] B. W. Coon, P. C. Mills, S. F. Oberman, and M. Y. Siu, "Scoreboard having size indicators for tracking sequential destination register usage in a multi-threaded processor," Patent US8 225 076B1, 2012.

[10] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt, "Analyzing cuda workloads using a detailed gpu simulator," in *ISPASS*, 2009, pp. 163–174.

[11] NVIDIA, 2007. [Online]. Available: https://github.com/nvidia/cuda-samples

[12] S. Che *et al.*, "Rodinia: A benchmark suite for heterogeneous computing," in *IISWC*, 2009, pp. 44–54.

[13] J. Leng *et al.*, "GPUWattch: Enabling energy optimizations in gpgpus," in *ISCA*, 2013, p. 487–498.