CMRC: Comprehensive Microarchitectural Register Coalescing for GPGPUs

Ahmad M. Radaideh
Qualcomm Technologies, Inc.
Austin, TX, USA
ahmadr@qti.qualcomm.com

Paul V. Gratz
Texas A&M University
College Station, TX, USA
pgratz@gratz1.com

***Abstract*—Graphics processing units (GPUs) deploy a large register file (RF) to achieve high compute throughput. This RF, however, consumes a large portion of the total dynamic power in the GPU. Additionally, the RF banks and operand collectors (OCs) are designed with limited number of ports causing access serialization and negatively impacting performance. In this work, we introduce CMRC, a coalescing-aware RF organization that takes advantage of frequent narrow-width data present in general purpose applications to increase performance and reduce energy for GPGPUs. CMRC is a low-cost comprehensive approach to register coalescing capable of combining narrow-width read and write accesses from same or different warp instructions into fewer accesses, reducing port contention and access pressure. On general purpose applications, CMRC reduces RF accesses by 31.8%, achieves a performance speedup of 16.5%, and reduces overall GPU energy by 32.2% on average, outperforming best of class prior work by ∼1.8x without the requirement of compiler support.**

I. Introduction

Graphics processing units (GPUs) are thread parallel processors that concurrently run thousands of hardware threads, originally designed for graphics applications. General-purpose GPUs (GPGPUs) apply GPUs to achieve high compute throughput on general compute applications and become ideal accelerators for data-parallel workloads. In these systems, scaling performance is primarily accomplished by integrating more compute resources and deploying a large register file (RF) to promote higher numbers of parallel threads in the GPU.

Power-hungry RF: To support massive thread level parallelism (TLP) and fast context switching between active threads, GPUs provide a large number of execution units to execute threads in parallel and a large RF to hold the execution state (context) of each thread. The RF size has been almost doubling for every new generation of the Nvidia GPUs, recently reaching 20MB in Tesla VG100 [17]. Prior power analysis done on the Fermi GPU, showed that the RF is one of the most power consuming components, contributing 16%-18% of the total chip dynamic power [14]. A percentage that is more likely increased on recent GPUs.

Serialized RF access: To avoid the high cost of multi-ported RF design, GPUs deploy a multi-banked structure with physical banks constructed from 6T SRAM arrays, each having a single read/write access port. The 6T arrays have a significant area benefit over dual-ported 8T arrays, at the cost of reducing the number of ports from two to one. Additionally, GPUs use single-ported operand collector (OC) units (smaller SRAM arrays) to capture and buffer the data read out of the RF banks and organize it for use in the functional units. The OCs and RF banks operate in parallel to support high access demand and provide high bandwidth. Due to access port limitations on the banks as well as the OC units, however, multiple access requests that target the same bank or OC unit at the same time experience *port conflicts* and their access is serialized. As a result, RF access latency increases, negatively impacting overall GPU performance and energy efficiency.

Wasted RF bandwidth and energy: Per-thread data values used in compute applications vary in size and some values can be represented by 8, 16, or 24 bits with the upper most-significant bits being either all zeros or all ones (carrying the sign bit). However, these frequent narrow-width values found in general purpose applications are still being treated as *full-width* when read and written into the RF, wasting RF bandwidth as well as access energy on *unneeded* bytes.

Contribution: We present CMRC, a new RF microarchitecture that supports read and write access coalescing in the GPU RF banks and OCs, to improve performance and reduce overall energy for GPGPU applications. CMRC addresses the RF access serialization issue by leveraging coalescing opportunities which arise from the frequent narrow-width data found in general purpose applications with low cost and complexity. As we show, read and write requests of narrow-width data can be combined into fewer number of bank accesses, yield higher bandwidth utilization for both RF banks and OCs, reduce RF and OC port contention, and as a result, improve overall performance and energy efficiency for GPGPU workloads.

II. Motivation

We motivate our approach to coalescing-aware RF by examining compute-intensive GPGPU benchmarks from Rodinia [5] and obtaining results by running those benchmarks on the GPGPU-sim v3.2 simulator [4].

Impact of limited access ports: With limited access ports on RF banks and OCs, read or write requests can experience one of the following types of *port conflicts*, causing their access to be serialized: (1) *Write-write* conflict. (2) *Read-write* conflict. (3) *Read-read* conflict. (4) *OC write* conflict. Fig. 1 shows the potential benefit of reducing RF port conflicts. In the figure we show the performance improvement achieved by removing all port conflicts (as if the banks had infinite ports) is ranging from 7% to 40%. From the figure we see that a large performance improvement is possible if port conflicts can be removed.

Wasted RF bandwidth: Narrow-width read and write accesses do not fully utilize the RF bandwidth as they carry unneeded bytes having only the sign-bit in their value. We found that only 29% of RF accesses, reads and writes, require a full register width on average. Fig. 2 shows that the wasted

978-3-9819263-5-4/DATE21/©2021 EDAA

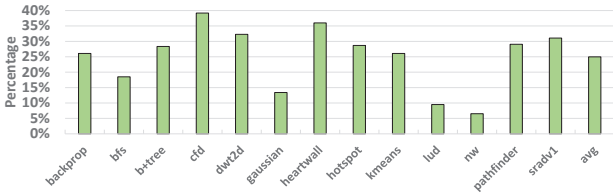


Fig. 1. A limit study on the potential speedup of reducing RF port conflicts.

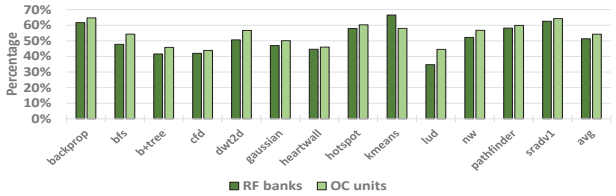


Fig. 2. Wasted RF bank and OC unit bandwidth.

bandwidth on RF banks and OCs due to narrow-width accesses is more than 50%. In addition, the unneeded bytes in narrow-width values waste dynamic power on every read and write to the RF and when propagate down into the OCs.

CMRC comprehensively addresses each form of conflict by coalescing *conflicting* narrow-width requests, reads or writes, targeting an RF bank or an OC unit into a single access, taking advantage of wasted RF bandwidth to reduce port contention and access pressure. With register coalescing, the number of RF accesses and the average access latency are reduced which lead to improving overall performance and energy.

III. Related Work

A. Non-coalescing Optimization Techniques

Significant work exists on RF power reduction for GPUs. RF caching [6], hierarchical RF [7], partitioned RF [2], RF renaming [9], RF with different power states [1], compiler-assisted RF [10], and an operand staging unit [12] have been proposed as alternative solutions to reduce dynamic power.

Data-dependent mechanisms have been proposed to reduce RF power. Lee et al. used a data compression technique to reduce power [13]. Liu et al. proposed another form of compression to handle scalar execution in GPGPU [16]. Li et al. proposed a mechanism to reduce SRAM memory power when reading and writing bit-1 data [15]. RF packing has been proposed by Wang et al. [19] to to reduce the number of register entries used and gate off unused entries.

Several mechanisms have been proposed to improve GPU performance. Gilani et al. took advantage of narrow-width values that can only be represented by 16 bits [8] to increase performance. Khorasani et al. proposed time-sharing a subset of physical registers between executing warps [11]. Oh et al. proposed increasing the number of concurrent thread blocks to improve performace [18]. All of these proposed power reduction and performance improvement techniques are orthogonal and potentially complementary to CMRC.

TABLE I
Register coalescing support in CORF/CORF++ [3] vs. CMRC.

Coalescing Support	Bank	Warp Instr.	CORF/CORF++	CMRC
2 reads	Same	Same	Yes	Yes
2 reads	Same	Different	No	Yes
2 writes	Same	Different	No	Yes
1 read and 1 write	Same	Different	No	Yes
2 reads	Different	Same	No	Yes

B. Register Coalescing Techniques

Esfeden et al. [3] introduced a Coalescing Operand Register File (CORF) for GPUs in two design flavors, a limited CORF (CORF from here forward) and an enhanced CORF++, to improve performance and reduce energy for GPGPU applications. This work is an important milestone, introducing the concept of GPU register coalescing. However, as the first such work, there are a number of limitations and overheads to the software-hardware approach taken there that we examine as key goals for improvement in our proposed CMRC technique.

CMRC represents a significant advance versus CORF/CORF++. First, CMRC is purely microarchitectural, with no need for compiler changes, unlike CORF/CORF++. The hardware-only approach removes the need for compile-time hints and software changes and allows the system to leverage per-workload/program behavior. Compile-time approaches lack information about actual data width as well as dynamic program behavior and add more hardware overhead and complexity (register packing and renaming in CORF/CORF++) that we wish to avoid. Further, CMRC's approach takes advantage of narrow-width data not only within warp registers but also across different warps that software approaches can not do. Second, CMRC supports a much wider variety of different coalescing types, (2-5) as listed in Table I, that CORF/CORF++ can not support. With more coalescing capabilities, our design is able to achieve a 1.8x speedup vs. the enhanced CORF++, with much lower area overhead (see Sections IV-C and V-F).

IV. Comprehensive Microarchitectural Register Coalescing Design

A. Design Overview

As shown in Fig. 3, the baseline RF for GPUs[1] is divided equally into number of banks. Each bank holds multiple register entries each of which is 128B wide. Physically, the bank is constructed using four narrow sub-banks where each sub-bank holds a 32B slice of every register. Each bank offers a single 128B access port to either read or write a full register in a cycle. OC units are required to capture source operands as they are read out from the RF banks over multiple cycles. Each OC offers a single 128B write port that can be used by one bank at a time. The RF banks are connected to the OCs using a crossbar network with 128B wide links.

[1]Here we focus on the Fermi architecture for easier comparison with prior works [3], the techniques developed can easily be applied to more modern GPU microarchitectures with minor changes.

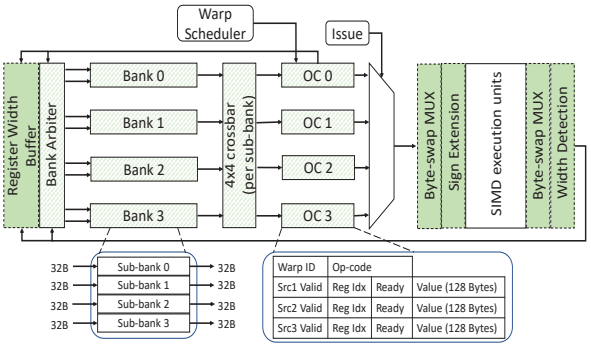


Fig. 3. GPU RF design. Shaded regions highlight our proposed additions.

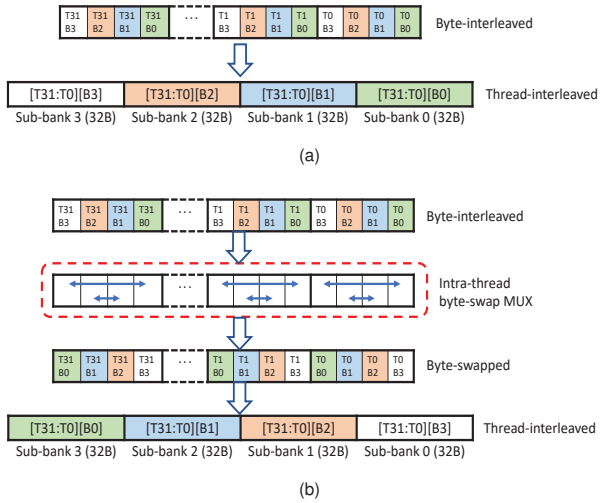


Fig. 4. Coalescing-friendly format and alignment for warp data: (a) Default right alignment and (b) Left alignment using intra-thread byte-swap MUXing.

Highlighted in Fig. 3 are the microarchitectural enhancements and additions of CMRC. The key feature of CMRC is enabling *finer control* on RF banks, crossbar network, and OC units such that the available 128B wide ports and data buses are *shared* among coalesed RF accesses. This allows for CMRC to provide comprehensive register coalescing at a very low cost and complexity.

B. Implementation Details

1) Register Data Format: Fundamental to register coalescing, is the ability for two narrow-width requests to be able to access different sub-banks within a bank at the same time. Therefore, instead of the *byte-interleaved* format used in baseline GPUs which poorly utilizes the effective sub-bank bandwidth, as shown on top in Fig. 4a, we use a coalescing-friendly data format, as shown at the bottom in Fig. 4a, that we refer to as a *thread-interleaved* format. In this format, register data starts with B0 for all 32 threads in the warp, then B1 for all of the threads, and so on. This allows for a narrow-width request to consume and fully utilize only a subset of the sub-banks and allow for the *free* sub-banks to be utilized by another *coalesced* access. We use the *thread-interleaved* format only within the RF (banks, interconnect, and OCs).

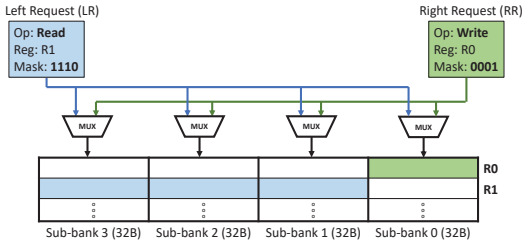


Fig. 5. RF bank with sub-bank controls to support register coalescing.

2) Register Data Alignment: Representing narrow-width data in the *thread_interleaved* format would make it right-aligned, *by default*, and allows only *lower* sub-banks within a bank to be used to hold the data. To support access coalescing, however, other narrow-width registers would need to be left-aligned or mapped to *upper* sub-banks such that a coalesced request would access non-overlapping sub-banks across the two targeted registers within a bank.

CMRC addresses registers left-alignment in a cheap way based on the fact that warp data does not have to be stored in the RF in a specific order (*ie.* Little-endian). Therefore, as shown in Fig. 4b, instead of shifting data across the entire warp-width when writing to the RF, CMRC swaps the data bytes (locally within each thread) such that, when data is represented in the *thread_interleaved* format, all B0's would map to sub-bank 3, B1's would map to sub-bank 2, and so on. When data is read from the RF, the same byte-swapping technique is used to revert data to its original order. Fig 3 shows the byte-swap MUXing placed on the write and read sides of the RF.

CMRC adopts a hardware-only solution to guide data alignment of warp registers with the goal of avoiding the high complexity and overhead associated with dynamic allocation and register renaming. Registers in CMRC are aligned based on their entry numbers with even entries being right-aligned and odd entries being left-aligned.

3) RF Banks with Sub-bank Controls: With the support of data formatting and alignment, it can be possible for two registers, even and odd, to be accessed at the same time given that their data reside in non-overlapping sub-banks. Therefore, CMRC provides each sub-bank with a separate read/write and address (entry number) control to allow for two narrow-width requests, reads or writes, targeting even and odd entries, to be coalesced and access the sub-banks at the same time, as shown in Fig. 5. Each narrow-width request uses a 4-bit mask (determined based on effective width) to indicate the sub-banks to be accessed. The ANDing of the two coalesced requests masks must be zero which means they do not conflict on any sub-bank. Each request provides its read/write and address controls to the targeted sub-banks separately.

4) RF Interconnect and OC Writes: To support more coalescing scenarios, we updated the RF crossbar that connects the banks with the OC units to have separate controls for every 32B slice of the data. Instead of using a crossbar with 128B

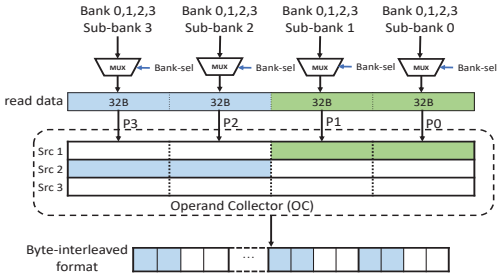


Fig. 6. OC unit with 32B write ports and per 32B input MUXing.

TABLE II
OVERHEAD OF CORF/CORF++ [3] AND CMRC OVER THE BASELINE.

Design Overhead	CORF/CORF++	CMRC
Rename table	3.7KB	0
Free register map	512B	0
Register width (sub-bank enables) buffer	1.47KB	384B
Code size increase	1.3% average	0
Data alignment	10 cross-thread shifters	5 thread-local MUXes

links, we split the crossbar into four narrower crossbars each having 32B links with same per-bit MUXing cost. This allows each 32B crossbar to connect a given sub-bank (across the 4 banks) into a 32B slice of the OC write port. Fig. 6 shows the separate crossbars (MUXing) on one OC.

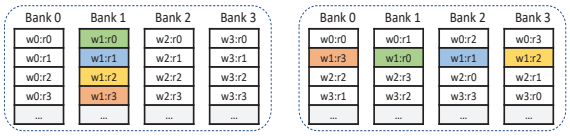
Fig. 6 also highlights the control updates on the OC unit to handle coalesced reads. Instead of capturing the coalesced read data in a single buffer entry and *unpacking* it as it is read out which would cost additional MUXing, CMRC unpacks the coalesced data as it is written into the OC, in a low-cost way, by having separate write controls for every 32B of the OC data buffers. The unpacking approach we use falls naturally as the *single* 128B write port on the OC now behaves as four 32B write ports where each narrow port writes to a 32B slice of the buffer entries separately.

5) Register Width Detection (Sub-bank Mask Generation): Width detection is added at the end of the execution pipeline (in write-back stage). We detect positive and negative narrow-width values by using reduction-OR and NAND gates on all bits with same byte-position, respectively. This is applied on bytes 1, 2, and 3 for all warp-threads and produces a 3-bit (sub-bank) mask (byte 0 mask-bit is implicitly set). The mask is captured in a buffer for the destination register being written. The total buffer space needed is only 384B (3-bit$\times$1024 warp registers). The generated mask is used on current register write and future reads to enable register coalescing.

6) Control Divergence: Here we note that the proposed solution has a limitation on handling control divergence of threads within the same warp. This limitation also exists in similar proposals [3]. We plan on addressing such limitation in our future work.

TABLE III
GPU CONFIGURATION PARAMETERS.

Parameter Name	Value
Number of SMs	16
Number of Warps	48 per SM
Number of warp threads	32
Register file size	128KB per SM
Register file banks	4 per SM
Register file bank width	128B
Operand collectors (OCs)	4 per SM
L1 cache size	16KB per SM
Shared memory size	48KB per SM
L2 cache size	768KB



(a) wid_layout (b) wshift_layout

Fig. 7. Different RF layouts (register-to-bank mapping).

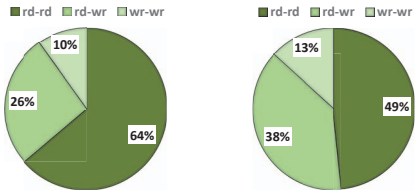
C. Design Overhead

Table II summarizes the overhead of CORF/CORF++ and CMRC over the baseline. CMRC avoids most of the complexity and overhead found in the software-hardware approach, including register packing and renmaing. The majority of updates in CMRC are enabling fine-grain control on RF banks, crossbar interconnect, and OC units which have very low overhead. CMRC uses a small buffer to capture the sub-bank enables for every warp register. It also uses cheaper thread-local multiplexers to perform warp data alignment. Both designs require a small amount of logic to detect registers width and perform sign extension when recovering narrow-width registers into their full width. We estimated the power overhead of CMRC to cost less than 2% of the RF dynamic power and less than 1% of the total GPGPU power.

V. EVALUATION

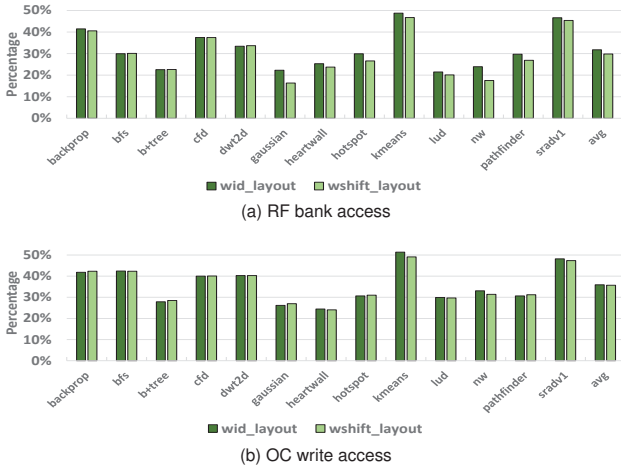
To evaluate our CMRC design, we implemented it into the GPGPU-Sim v3.2 simulator [4] and measured its dynamic power consumption using GPU-Wattch [14]. We used a Fermi-like Nvidia GPU model with configuration parameters listed in table III[1]. For all evaluations we used the Rodinia benchmark suite [5] which consists of general purpose benchmarks that target GPU platforms with the standard input sets that shipped with these benchmarks.

Fig. 7 shows two possible layouts for mapping warp registers into RF banks. In Fig. 7a, all registers for a given warp are mapped into the same bank which we call *wid_layout*. The other layout in Fig. 7b shows registers for a given warp are interleaved across the banks and we call this *wshift_layout*. Unlike CMRC, CORF/CORF++ has the requirement to use *wid_layout*. Here we implement CMRC with both layouts for comparison.



(a) wid_layout (b) wshift_layout

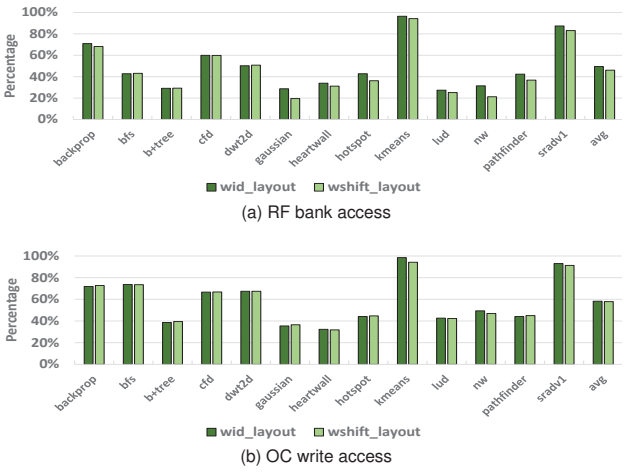
Fig. 8. Breakdown of RF banks coalesced accesses for two RF layouts.



(a) RF bank access

(b) OC write access

Fig. 9. Percentage of access reduction for two RF layouts.



(a) RF bank access

(b) OC write access

Fig. 10. Percentage of bandwidth utilization increase for two RF layouts.

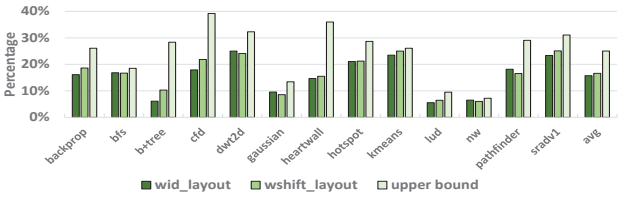


Fig. 11. Percentage of overall speedup for two RF layouts.

A. RF Coalesced Accesses

Fig. 8 shows the breakdown of coalesced accesses on RF banks for the two layouts. In Fig. 8a, for *wid_layout*, we see the majority of coalesced accesses are reads, representing 64% of all coalesced accesses while coalesced writes represents only 10%. Fig. 8b, for *wshift_layout*, shows that more writes are getting coalesced as the percentage of coalescing writes is increased to 13% and also the percentage of coalescing read and write requests significantly increased to 38%.

B. RF Access Reduction

CMRC reduces the number of accesses made into RF banks as well as the number of writes to the OC units by combining narrow-width registers accesses into fewer physical accesses. Bank accesses are reduced by coalescing read and write accesses from same or different warp instructions into fewer accesses. OC writes are reduced by coalescing registers read data from same or different banks into fewer accesses. Fig. 9a and Fig. 9b show the percentage of RF banks access reduction and OC units write reduction compared to the baseline, respectively.

With the low-cost alignment scheme, CMRC reduces RF bank accesses by ∼32% and reduces OC write accesses by ∼36% on average. Reduction in RF accesses improves bandwidth utilization as well as reduce RF dynamic energy consumption. Interestingly, we find very small difference between the *wid_layout* and *wshift_layout* register layouts, with the *wid_layout* register layout requiring ∼1.5% fewer accesses for both the banks and OCs.

C. RF Bandwidth Utilization

As CMRC is capable of combining narrow-width accesses into fewer accesses, RF banks and OCs bandwidth is consequently improved. Fig. 10a and Fig. 10b show the percentage of bandwidth utilization improvement on RF banks and OCs compared to the baseline, respectively. CMRC is able to increase the bandwidth utilization on the RF banks by ∼49% and on the OCs by ∼58% on average. The difference between the *wid_layout* and *wshift_layout* register layouts are small, with the *wid_layout* register layout showing ∼2.5% more bandwidth for the RF banks and ∼1% more bandwidth for the OCs.

D. Performance

One of the primary objectives of CMRC is to improve performance for general purpose applications on the GPU. CMRC enables register coalescing that reduces the number of RF read and write accesses, reduces RF pressure and port contention, and improves bandwidth utilization which contribute to an overall IPC performance speedup.

Fig. 11 shows the speedup of CMRC on the two RF layouts. The speedup achieved is ∼15.3% and ∼16.5% on the *wid_layout* and *wshift_layout* register layouts, respectively, compared to baseline. These speedups achieved are more than 50% of the average *upper bound* value of 25% which represents the microarchitectural limit register coalescing can reach,

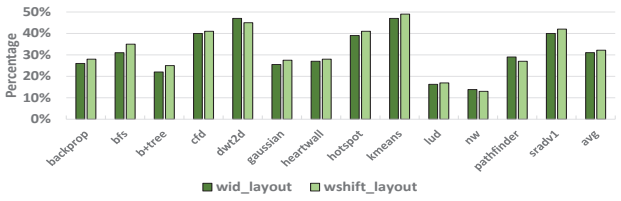


Fig. 12. Percentage of overall dynamic energy reduction for two RF layouts.

TABLE IV
RESULTS ACHIEVED BY CORF/CORF++ [3] VS. CMRC.

Metric	**CORF**	**CORF++**	**CMRC *wid_layout***	**CMRC *wshift_layout***
IPC speedup	4%	9%	15.3%	16.5%
Dynamic energy reduction	8.5%	17%	31.1%	32.2%
RF access reduction	10%	23%	31.8%	30.5%

given that RF requests are all coalescable and no coalescing opportunity is lost due to registers size or alignment.

E. Dynamic Energy Reduction

Improving energy efficiency in the GPU is another primary objective of CMRC. RF access coalescing improves performance by making kernels have faster runtimes and also reduces the dynamic energy of the RF by reducing the number of accesses and improving bandwidth utilization. These improvements lead to a more energy efficient GPU for general purpose applications. Fig. 12 shows the percentage reduction of overall dynamic energy in the GPU using our coalescing-aware design for two RF layouts. With a *wid_layout* RF, CMRC is able to reduce overall dynamic energy by ∼31.1% on average compare to the baseline. The achieved energy reduction with *wshift_layout* RF is ∼32.2% on average.

F. Comparison Against CORF/CORF++

Table IV summarizes the speedup and energy efficiency CMRC achieves versus the prior work CORF/CORF++. As shown together with the overhead comparisons listed in Table II, we see that CMRC significantly outperforms the prior work designs, while requiring lower hardware overheads and no direct software changes.

VI. CONCLUSION

In this work, we introduced the new Comprehensive Microarchitectural Register Coalescing (CMRC) design to improve performance and energy efficiency in GPGPUs. CMRC supports register coalescing for narrow-width reads and writes, taking advantage of wasted RF bandwidth to reduce RF banks and OC units port contention. Compared to prior work, our design provides higher register coalescing capabilities at a lower cost and complexity. Our design provided a 16.5% performance speedup and a 32.2% dynamic energy reduction on average for GPGPU benchmarks.

REFERENCES

[1] M. Abdel-Majeed and M. Annavaram, "Warped register file: A power efficient register file for gpgpus," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2013, pp. 412–423.

[2] M. Abdel-Majeed, A. Shafaei, H. Jeon, M. Pedram, and M. Annavaram, "Pilot register file: Energy efficient partitioned register file for gpus," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2017, pp. 589–600.

[3] H. Asghari Esfeden, F. Khorasani, H. Jeon, D. Wong, and N. Abu-Ghazaleh, "Corf: Coalescing operand register file for gpus," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 701–714.

[4] A. Bakhoda, G. L. Yuan, W. W. Fung, H. Wong, and T. M. Aamodt, "Analyzing cuda workloads using a detailed gpu simulator," in *2009 IEEE International Symposium on Performance Analysis of Systems and Software*. IEEE, 2009, pp. 163–174.

[5] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *2009 IEEE international symposium on workload characterization (IISWC)*. IEEE, 2009, pp. 44–54.

[6] M. Gebhart, D. R. Johnson, D. Tarjan, S. W. Keckler, W. J. Dally, E. Lindholm, and K. Skadron, "Energy-efficient mechanisms for managing thread context in throughput processors," in *2011 38th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2011, pp. 235–246.

[7] M. Gebhart, S. W. Keckler, and W. J. Dally, "A compile-time managed multi-level register file hierarchy," in *2011 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2011, pp. 465–476.

[8] S. Z. Gilani, N. S. Kim, and M. J. Schulte, "Power-efficient computing for compute-intensive gpgpu applications," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2013, pp. 330–341.

[9] H. Jeon, G. S. Ravi, N. S. Kim, and M. Annavaram, "Gpu register file virtualization," in *Proceedings of the 48th International Symposium on Microarchitecture*. ACM, 2015, pp. 420–432.

[10] N. Jing, H. Liu, Y. Lu, and X. Liang, "Compiler assisted dynamic register file in gpgpu," in *International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 2013, pp. 3–8.

[11] F. Khorasani, H. A. Esfeden, A. Farmahini-Farahani, N. Jayasena, and V. Sarkar, "Regmutex: Inter-warp gpu register time-sharing," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 816–828.

[12] J. Kloosterman, J. Beaumont, D. A. Jamshidi, J. Bailey, T. Mudge, and S. Mahlke, "Regless: Just-in-time operand staging for gpus," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2017, pp. 151–164.

[13] S. Lee, K. Kim, G. Koo, H. Jeon, W. W. Ro, and M. Annavaram, "Warped-compression: enabling power efficient gpus through register compression," in *ACM SIGARCH Computer Architecture News*, vol. 43, no. 3. ACM, 2015, pp. 502–514.

[14] J. Leng, T. Hetherington, A. ElTantawy, S. Gilani, N. S. Kim, T. M. Aamodt, and V. J. Reddi, "Gpuwattch: enabling energy optimizations in gpgpus," in *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3. ACM, 2013, pp. 487–498.

[15] A. Li, W. Zhao, and S. L. Song, "Bvf: enabling significant on-chip power savings via bit-value-favor for throughput processors," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2017, pp. 532–545.

[16] Z. Liu, S. Gilani, M. Annavaram, and N. S. Kim, "G-scalar: Cost-effective generalized scalar execution architecture for power-efficient gpus," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2017, pp. 601–612.

[17] Nvidia, "Whitepaper: Nvidia tesla v100 gpu architecture," 2017. [Online]. Available: http://nvidia.com

[18] Y. Oh, M. K. Yoon, W. J. Song, and W. W. Ro, "Finereg: fine-grained register file management for augmenting gpu throughput," in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2018, pp. 364–376.

[19] X. Wang and W. Zhang, "Gpu register packing: Dynamically exploiting narrow-width operands to improve performance," in *2017 IEEE Trustcom/BigDataSE/ICESS*. IEEE, 2017, pp. 745–752.