On-The-Fly Parallel Data Shuffling for Graph Processing on OpenCL-based FPGAs

Xinyu Chen[1], Ronak Bajaj[1], Yao Chen[2], Jiong He[3],

Bingsheng He[1], Weng-Fai Wong[1], Deming Chen[4]

[1]National University of Singapore, [2]Advanced Digital Sciences Center,
[3]Alibaba Group, [4]University of Illinois at Urbana-Champaign

Graph processing on FPGAs

- **Graph processing is widely used in variety of application domains.**
  - Social networks
  - Cybersecurity
  - Machine learning

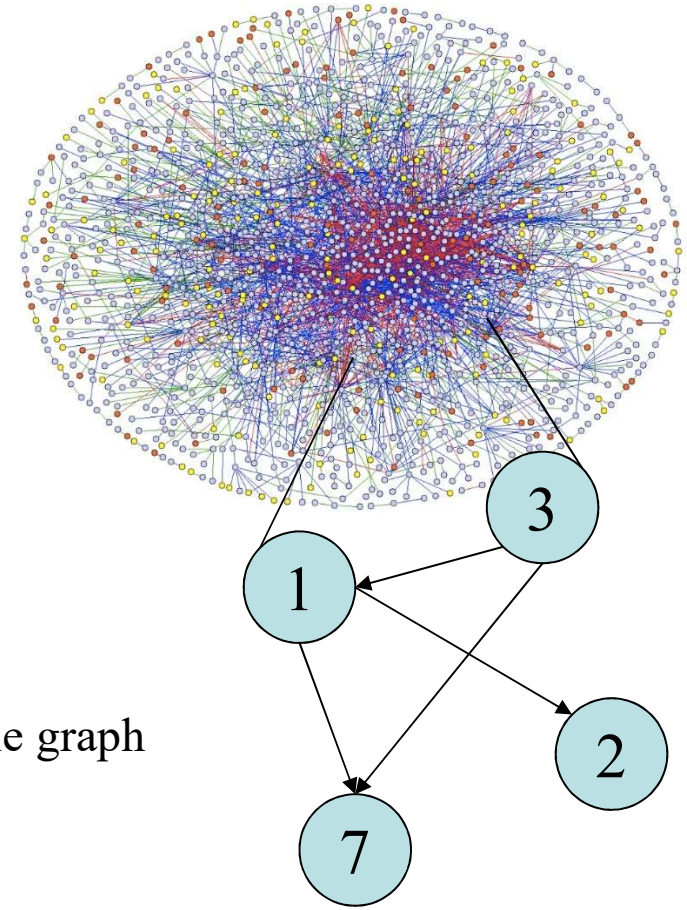
- **Accelerating graph processing on FPGA has attracted a lot of attention benefiting from:**
  - Fine grained parallelism
  - Low power consumption
  - Extreme configurability

Graph processing on HLS-based FPGAs

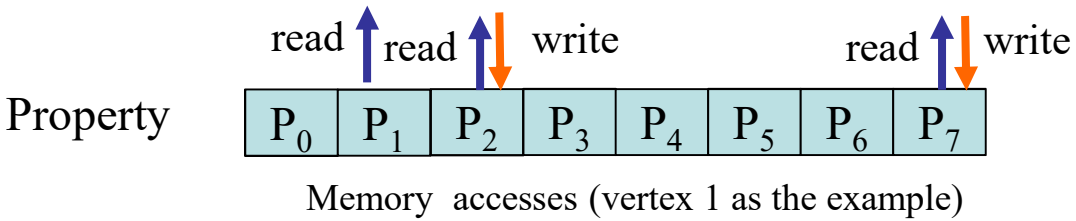
- **Previous RTL-based FPGAs development.**
  - Time-consuming
  - Deep understanding of hardware

- **To ease the use of FPGAs, HLS tools have been proposed.**
  - High-level programming model
  - Hide hardware details
  - Both Intel and Xilinx have HLS tools
- **Graph processing on OpenCL-based FPGAs.**

GAS model for graph processing



Example graph



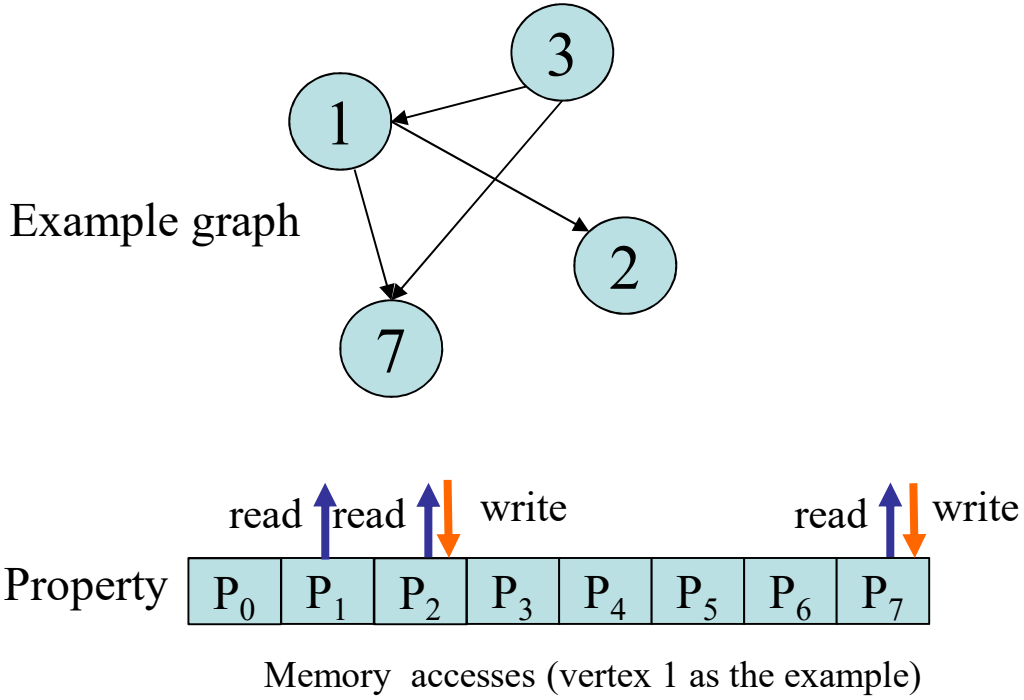
Memory accesses (vertex 1 as the example)

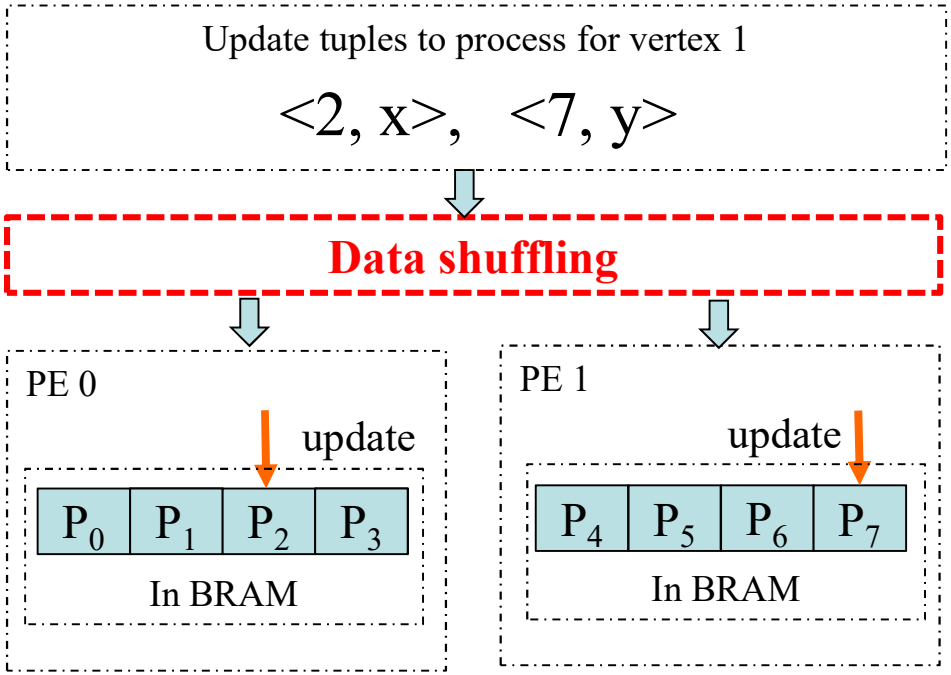
- **Scatter**: for each edge, an update tuple is generated with the format of *<destination, value>.*
  - E.g. <2, x>, <7, y> for vertex 1
- **Gather:** accumulate the value to destination vertices.
  - E.g. Op(P_2 , x), Op(P_7 , y)
- **Apply:** an apply function on all the vertices.

A. Roy, L. Bindschaedler, J. Malicevic, and W. Zwaenepoel, "Chaos: Scale-out graph processing from secondary storage," in *SOSP*, 2015

GAS model on FPGAs

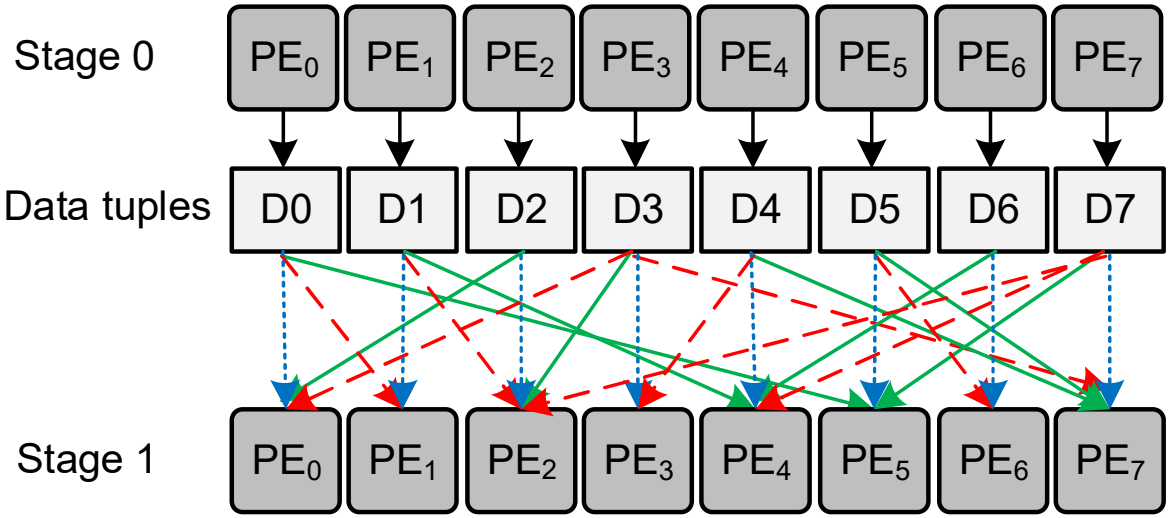
- **BRAM caching**
  - avoid random memory accesses to property array.
- **Multiple PEs**
  - each PE processes a part of cached data and runs independently.





Data shuffling

- **Widely used for irregular applications.**
- **The data generated with format of <*dst, value*> is dispatched to *'dst'* PEs to process.**
- **Challenges:**
  - Run-time data dependency
  - Parallelism



* Arrows with different colours show a few shuffling examples.

OpenCL does not natively support shuffling

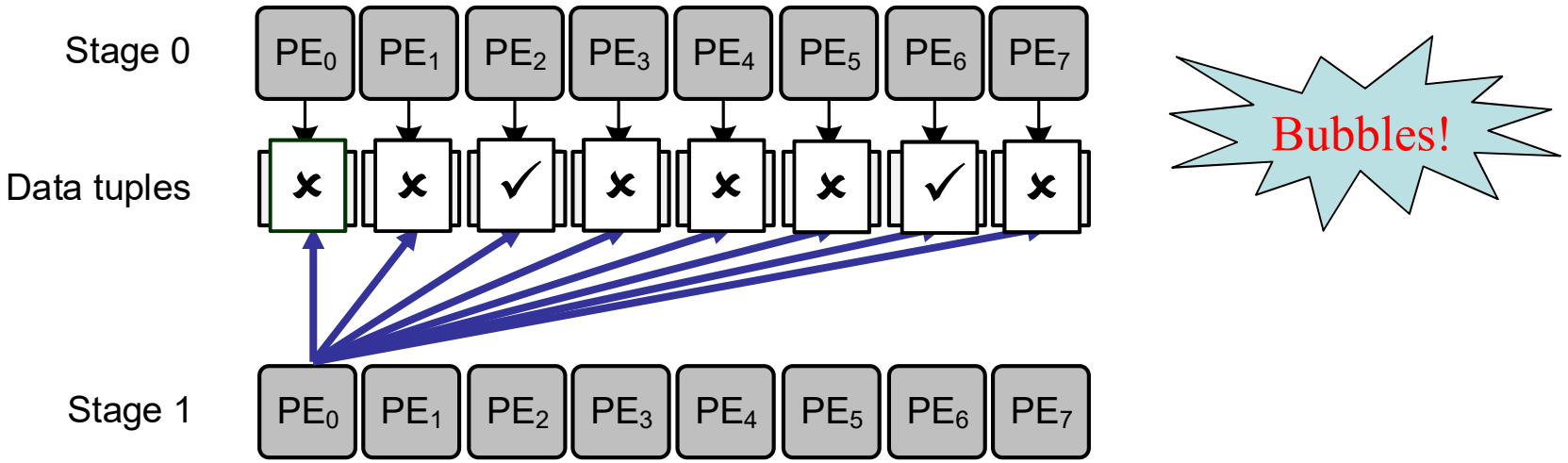
- **Fine-grained control logic is not available for OpenCL.**
- **No vendor-specific extension for shuffling [1].**
- **OpenCL only does static analysis at compile time, thus cannot extract parallelism in functions with run-time dependency [2].**

[1] Kapre, Nachiket, and Hiren Patel. "Applying Models of Computation to OpenCL Pipes for FPGA Computing." *Proceedings of the 5th International Workshop on OpenCL*. ACM, 2017.

[2] Z. Li, L. Liu, Y. Deng, S. Yin, Y. Wang, and S. Wei, "Aggressive pipelining of irregular applications on reconfigurable hardware," in *ISCA*, 2017.

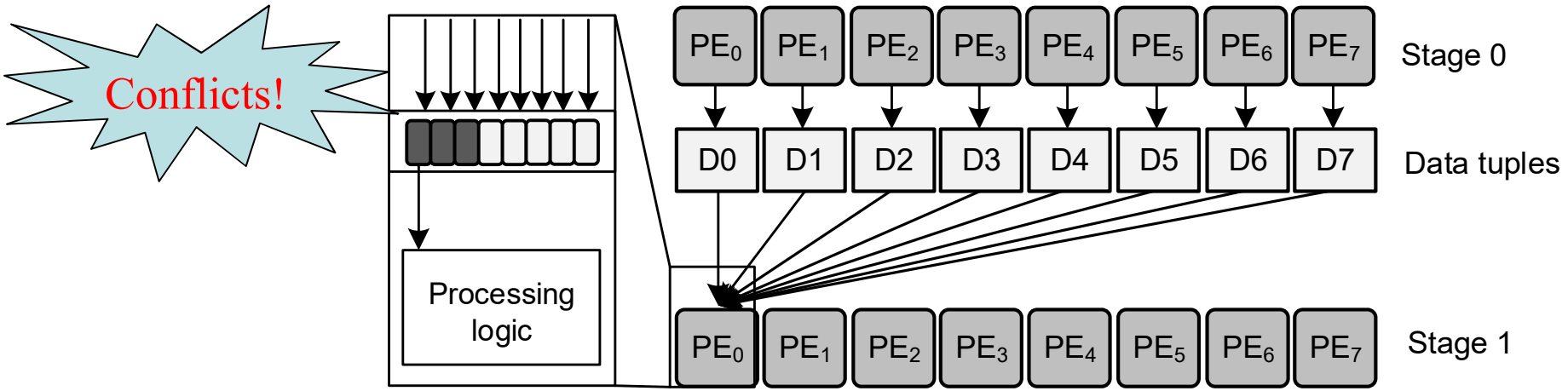
Potential shuffling solutions with OpenCL

- **Polling**
  - Each PE checks the tuples serially.
  - ‘Bubbles’ are introduced.
  - 8 cycles for dispatching a set of 8 tuples.



Potential shuffling solutions with OpenCL

- **Convergence kernel from [1]**
  - Each PE writes wanted tuples to local BRAM in parallel.
  - The run-time data dependency is not resolved.
  - Initiation interval (II) equals to 284 cycles.



[1] Wang, Zeke, et al. "Multikernel Data Partitioning With Channel on OpenCL-Based FPGAs." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 25.6 (2017): 1906-1918.

Insights

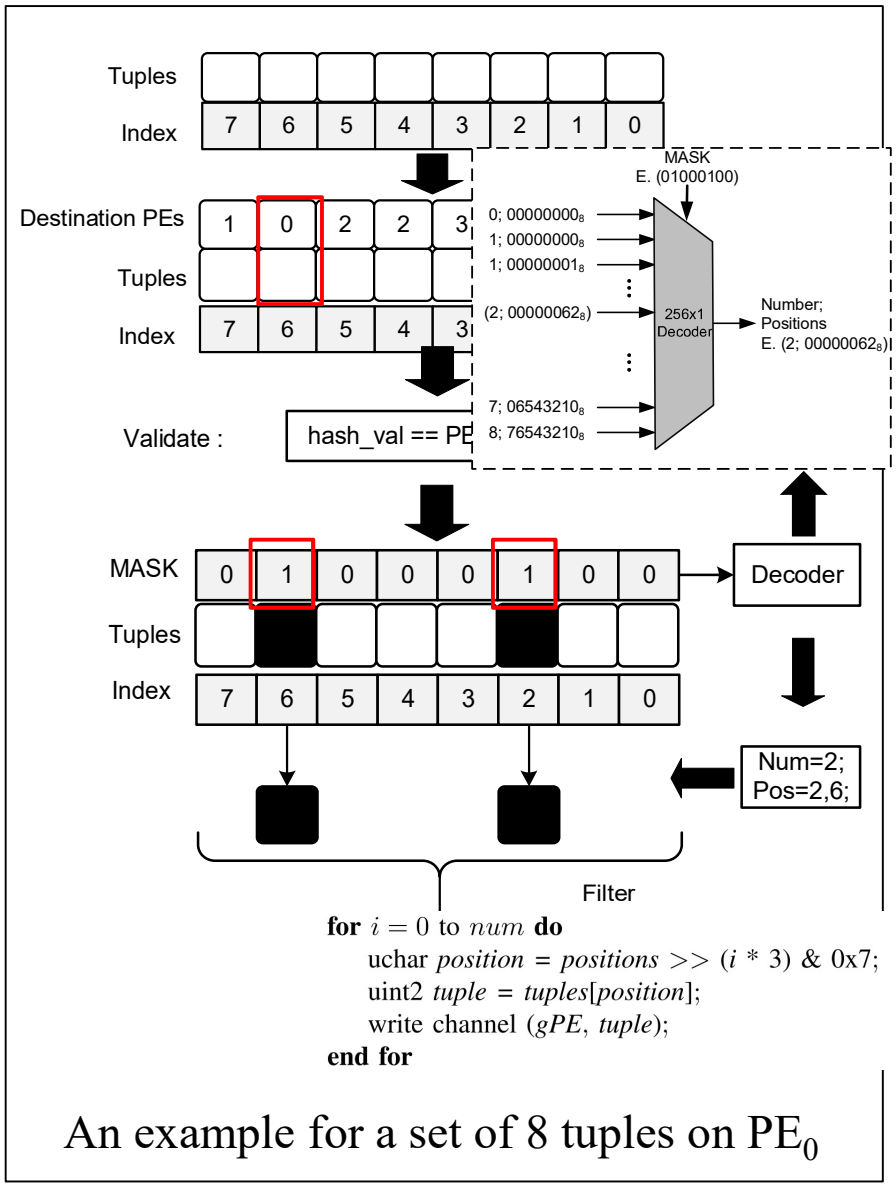
- **Polling: introduces ‘bubbles’.**
- **Convergence kernel: the run-time dependency is still there.**

- **What if we know the positions and number of wanted tuples?**
  - PEs can directly access the wanted tuples.
  - Cycles needed equal to number of wanted tuples.

- **How to know the positions and number of wanted tuples?**
  - Decoder based solution.
  - E.g. 2^8 possibilities, for a set of 8 tuples, since each tuple has two statuses only.

Proposed shuffling

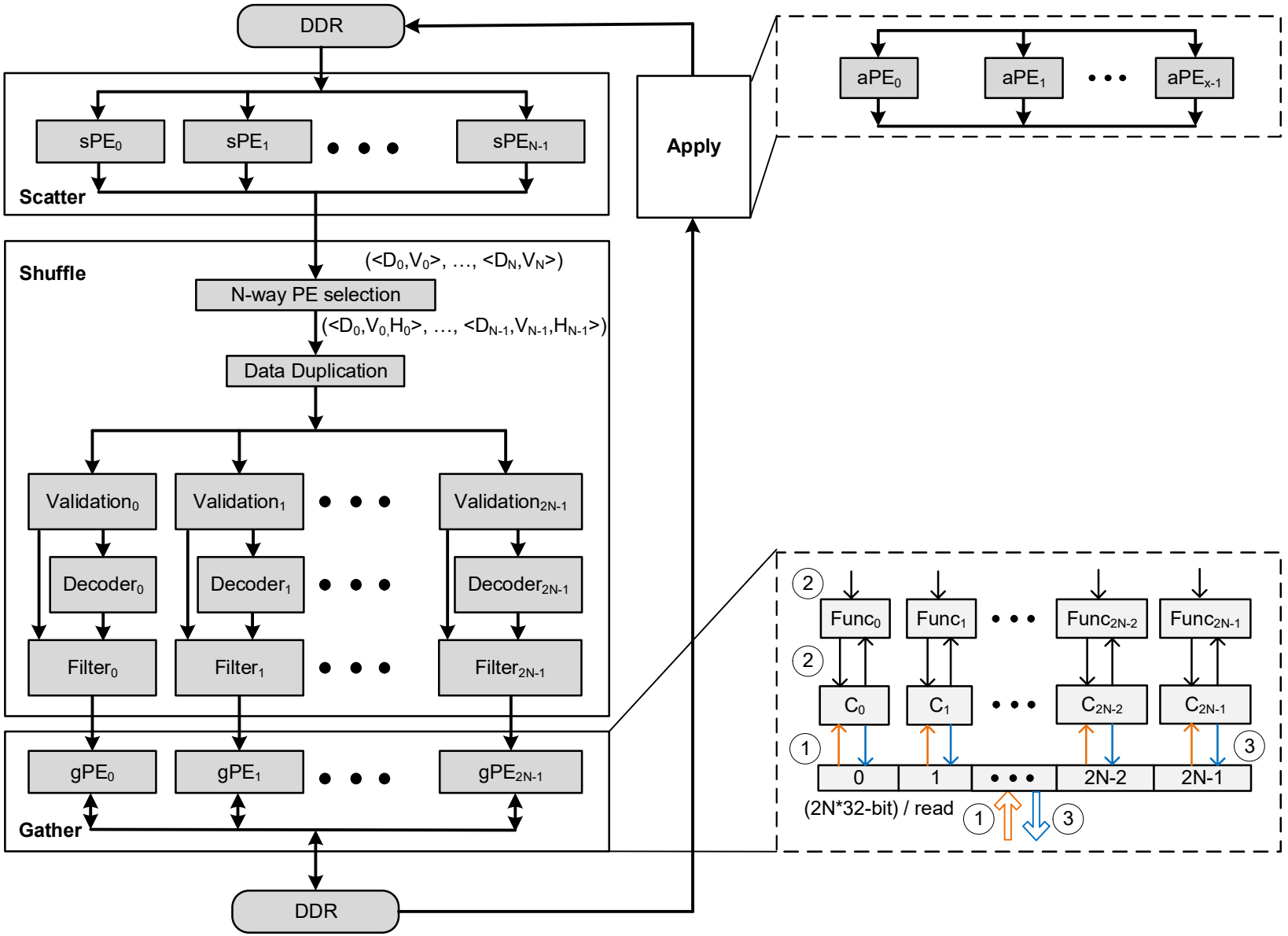
- Calculate the destination PEs.
- Compute an 8-bit MASK by comparing destination PEs with the id of current PE, 0.
- Decode the positions and number of wanted tuples.
- Collect the wanted tuples without “bubbles”.



Proposed shuffling

- No ‘bubbles’ - no cycle wasted on unwanted tuples.
- Resolve the run-time dependency.
- All the modules are pipelined.

Proposed graph processing framework with shuffle



Experimental configuration

- **Our experiments are conducted on a Terasic DE5-Net board.**
- **BFS, SSSP, PageRank and SpMV are used as applications.**

TABLE I: Details of hardware.

FPGA	Altera Stratix V GX
OpenCL	Intel FPGA SDK 16.1 for OpenCL
Memory Bandwidth	17GB/s (Peak); 12.5GB/s (at 200MHZ)

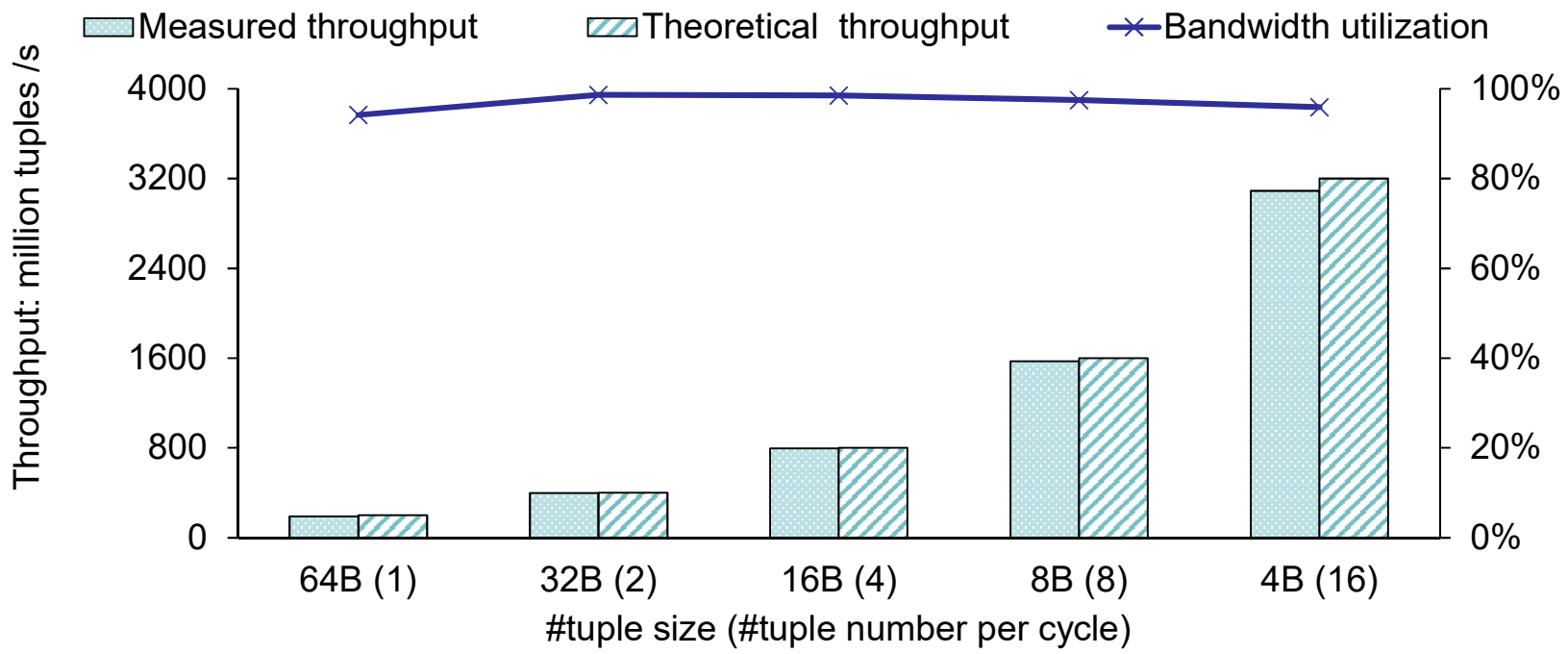
TABLE II: Graph dataset.

Graphs	$\|V\|$	$\|E\|$	D_{avg}	D_{max}
rmat-19-32 (R19) [34]	524K	17M	32	90K
rmat-21-32 (R21) [34]	2M	67M	32	211K
mouse-gene (MG) [35]	43K	14.5M	670	8K
web-google (GG) [35]	875K	5.1M	11	6.4K
pokec (PK) [35]	1.6M	31M	37	20K
wiki-talk (WT) [35]	2M	5M	4	100K
live-journal (LJ) [35]	4.8M	69M	13	3K
twitter-2010 (TW) [35]	41M	1.4B	35	770K

[34] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: An approach to modeling networks," JMLR, 2010.
[35] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in AAAI, 2015.

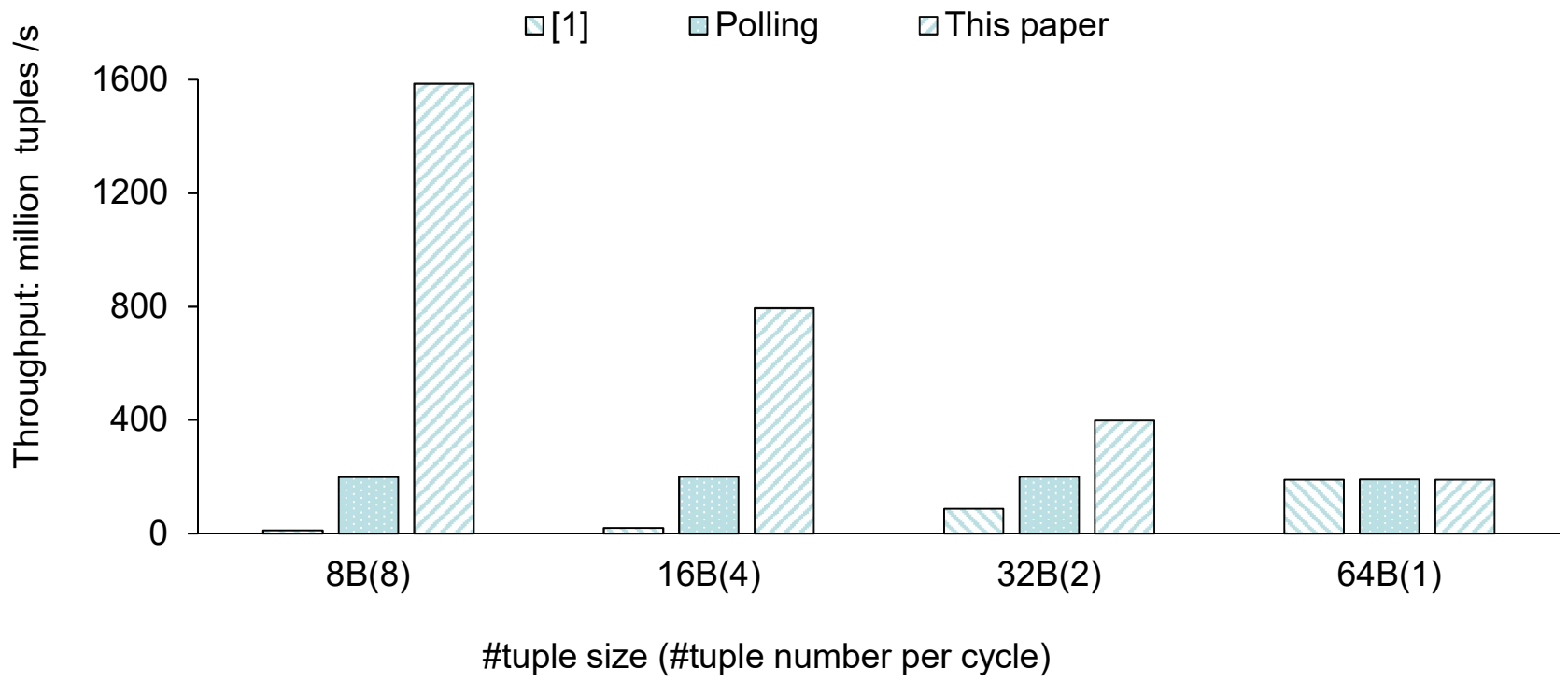
Efficiency of shuffle

- **Theoretical throughput = memory_bandwidth / tuple_size**
- **The performance is close the theoretical throughput.**



Efficiency of shuffle

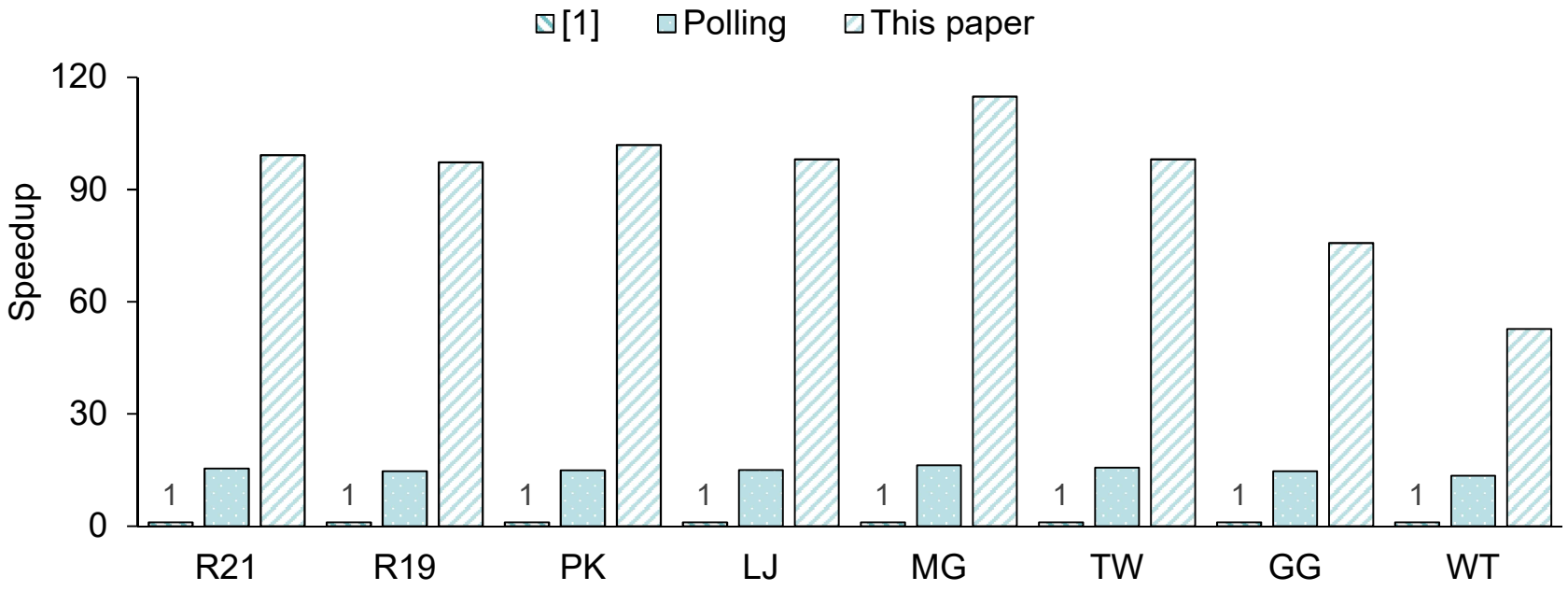
- **The throughput of our shuffle is much higher than existing shuffling solutions.**



[1] Wang, Zeke, et al. "Multikernel Data Partitioning With Channel on OpenCL-Based FPGAs." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 25.6 (2017): 1906-1918.

End to end performance

- **Compare the performance of graph frameworks with different shuffling solutions.**
- **Speedup of PageRank is up to 100× of [1], and 6× of Polling.**



[1] Wang, Zeke, et al. "Multikernel Data Partitioning With Channel on OpenCL-Based FPGAs." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 25.6 (2017): 1906-1918.

Resource utilization

- **BRAMs are well utilized for vertex caching.**
- **PR and SpMV consume DSPs.**

TABLE III: Resource utilization and frequency.

Algo.	Freq.(Mhz)	BRAM	Logic	DSP
PR	171.5	2,329 (91%)	125,811 (54%)	8 (3%)
SpMV	165.4	2,394 (94%)	125,854 (54%)	8 (3%)
BFS	172.6	2,030 (79%)	127,085 (54%)	0 (0%)
SSSP	170.6	1,926 (75%)	123,457 (53%)	0 (0%)

Compare with RTL-based works

- **Our approach achieves throughput that is comparable or even better than RTL-based graph processing designs.**

TABLE V: Comparison with state-of-the-art implementations.

Algo.	Graph	Others	Throughput	Ours	Impro.
PR	LJ	ForeGraph [14]	1193	1110	0.93×
	WT	[11]	279	584	2.09×
	GG	[11]	317	838	2.64×
SpMV	WT	GraphOps [38]	190	551	2.90×
SSSP	WT	[13]	657	618	0.94×
	LJ	[13]	872	1129	1.29×

[11] S. Zhou, C. Chelmis, and V. K. Prasanna, "Optimizing memory performance for FPGA implementation of pagerank." in *ReConFig*, 2015.

[13] S. Zhou, C. Chelmis, and V. K. Prasanna, "High-throughput and energyefficient graph processing on FPGA," in *FCCM*, 2016.

[14] G. Dai, T. Huang, Y. Chi, N. Xu, Y. Wang, and H. Yang, "Foregraph: Exploring large-scale graph processing on multi-FPGA architecture," in *FPGA*, 2017.

[38] T. Oguntebi and K. Olukotun, "Graphops: A dataflow library for graph analytics acceleration," in *FPGA*, 2016.

Conclusion

- Data shuffling on OpenCL-based FPGAs is challenging due to the run-time data dependency.
- We propose an efficient OpenCL-based data shuffling method.
- The performance of graph processing framework integrated our shuffling is comparable to state-of-the-art RTL based works.

Acknowledgement

- This work is supported by a MoE AcRF Tier 1 grant (T1 251RES1824) and Tier 2 grant (MOE2017-T2-1-122) in Singapore. This work is also partly supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme, and the SenseTime Young Scholars Research Fund.

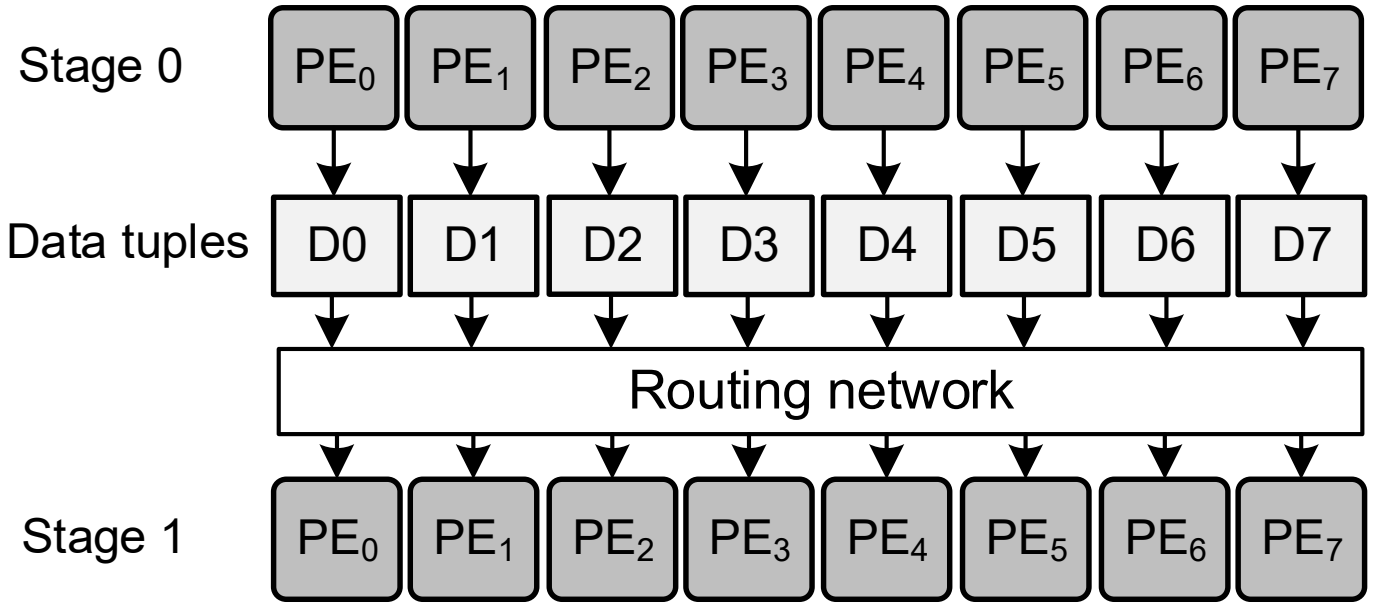
- We also thank Intel for hardware accesses and donations.

NUS National University of Singapore

Thanks

Data shuffling on RTL-based FPGAs

- **Fine-grained control logic based NoCs.**



Outline

- **Introduction to data shuffling**
- **Data shuffling on OpenCL-based FPGAs**
  - Motivations
  - Design and implementation
- **Graph processing framework with proposed shuffling**
- **Evaluation**
- **Conclusion**
- **Acknowledgement**