Boyi: A Systematic Framework for Automatically Deciding the Right Execution Model of OpenCL Applications on FPGAs

Jiantong Jiang (*Northeastern University, China*),

Zeke Wang (*Zhejiang University, China*),

Xue Liu (*Northeastern University, China*),

Juan Gómez-Luna (*ETH Zürich, Switzerland*),

Nan Guan (*Hong Kong Polytechnic University*),

Qingxu Deng (*Northeastern University, China*),

Wei Zhang (Hong Kong University of Science and Technology),

Onur Mutlu (ETH Zürich *, Switzerland*)

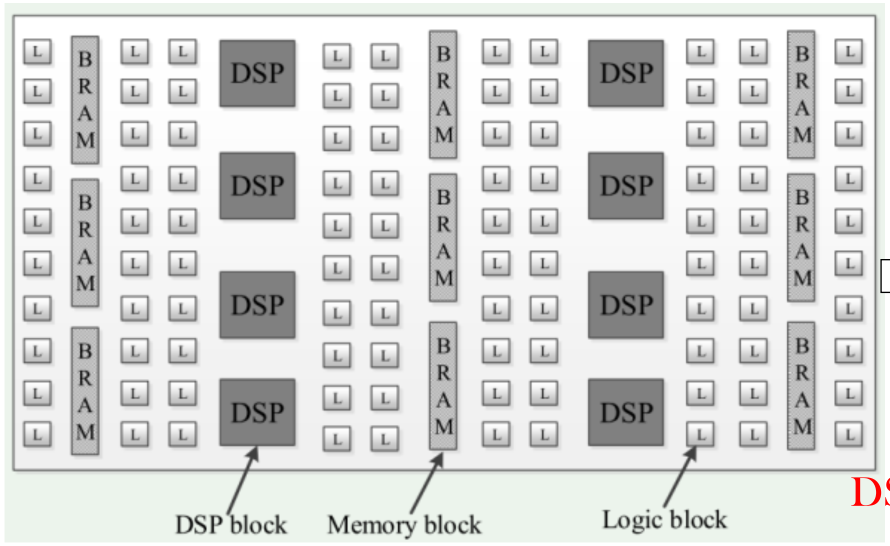
Outline

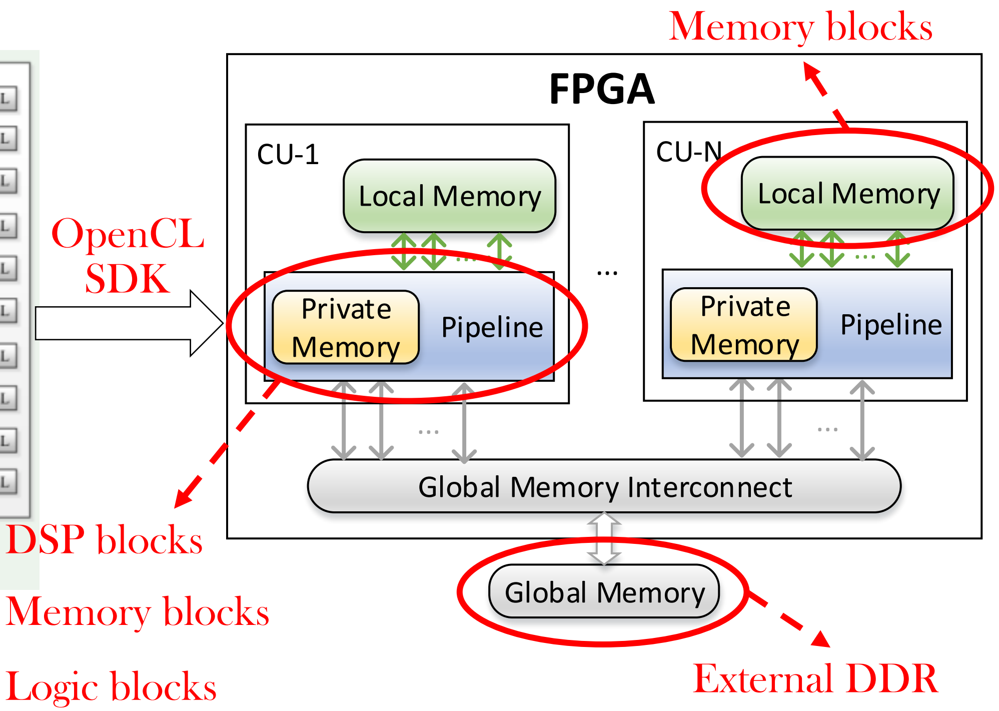
- Background and Motivations
- Our Solution
- Experiment
- Conclusion

What is OpenCL?

- OpenCL stands for *Open Computing Language*.
- OpenCL has been developed for heterogeneous computing environments with a host-accelerator execution model.
  - The CPU runs the control task.
  - The GPU/ FPGA runs the computing kernel.

OpenCL on FPGA

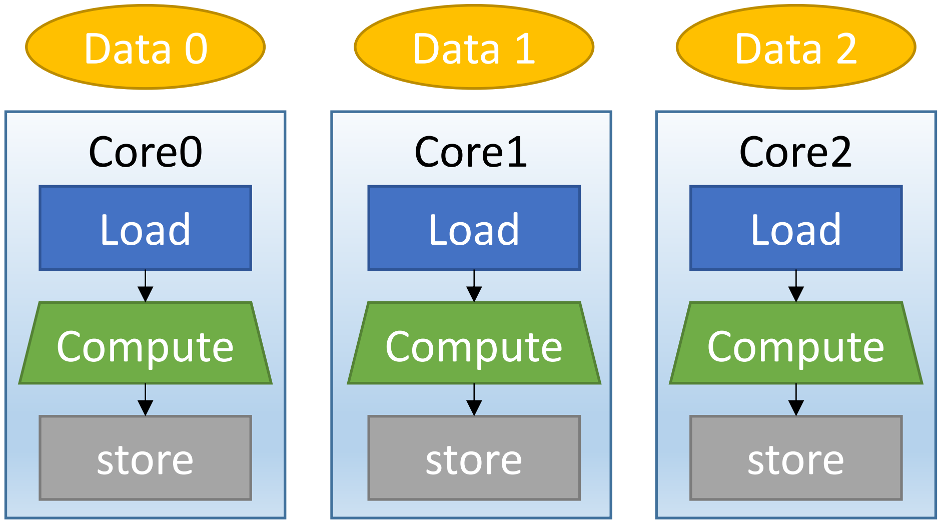




- Hardware-centric → fine-gained parallelism
- Users need to program with HDL.

- Software-centric → FPGA as a parallel architecture.
- Users can program with OpenCL.

Conventional OpenCL: NDRange Kernel

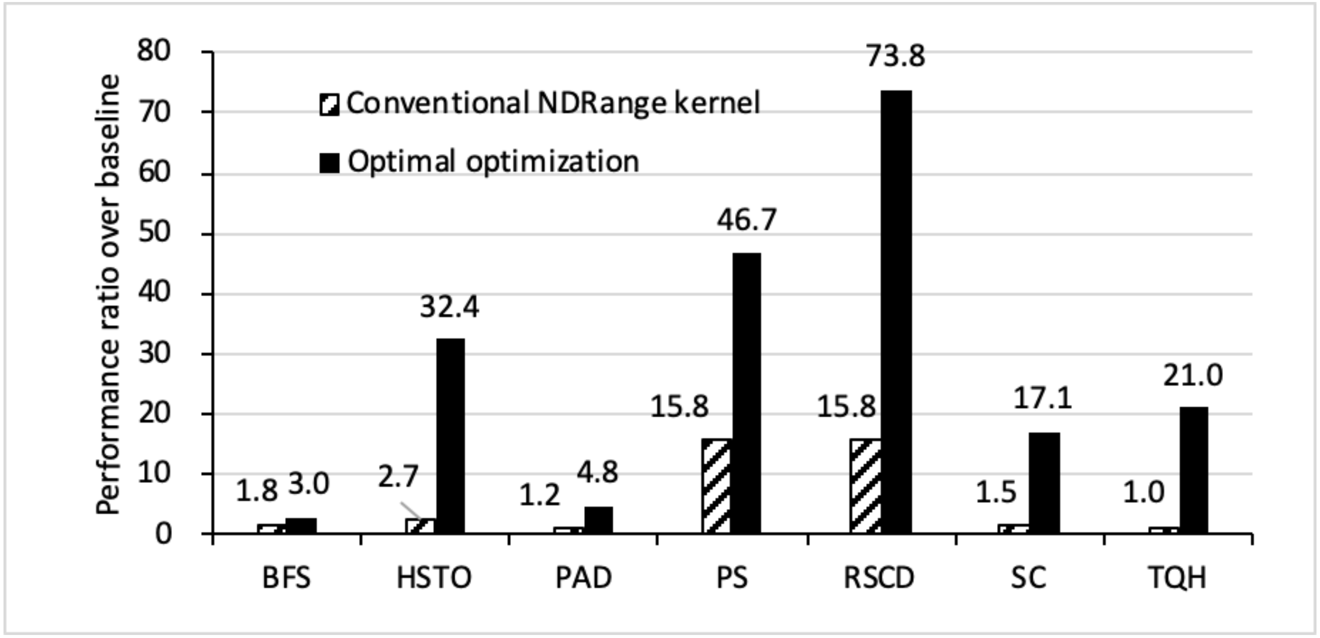


Explicit multi-threaded → executing the same operation on multiple data concurrently

- How conventional OpenCL performs on an FPGA?

Issue of Conventional OpenCL

- Conventional OpenCL cannot always represent FPGA architecture in an efficient manner.

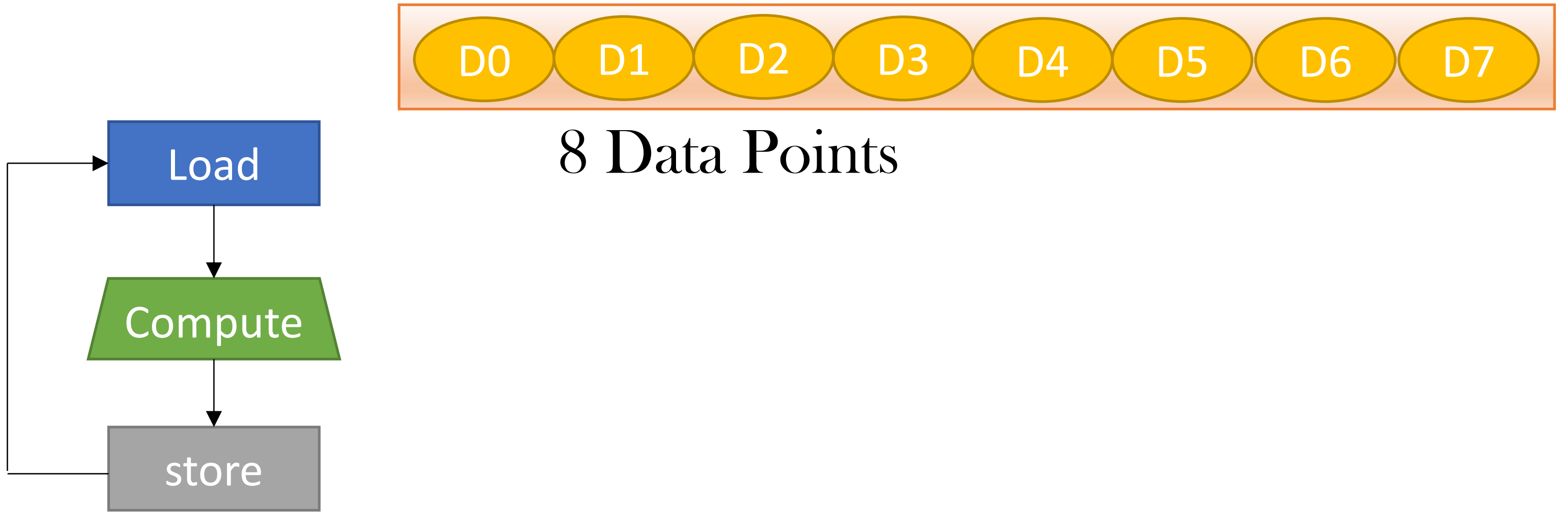


1.6x 12.1x 4.1x 3.0x 4.7x 11.3x 21.0x

The optimal performance is enabled by two OpenCL features!

OpenCL Feature 1: SWI Kernel

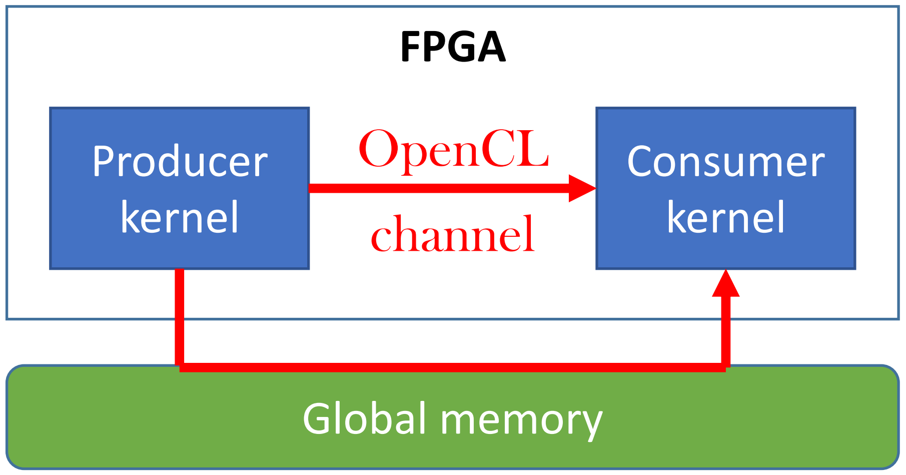
- The SWI model executes the kernel in only one CU that contains only one work-item.



Pipelined parallelism→ No conflict among work items.

OpenCL Feature 2: OpenCL Channel

- OpenCL channel can be used to pass data between two OpenCL kernels (typically SWI).
  - ➢Synchronizing the kernels
  - ➢Reducing the number of global memory accesses



Challenges

- Two OpenCL features exponentially increase design space.
  - Enabled by two features, we have four OpenCL execution models:

    NDRange | SWI | NDRange+Channel | SWI+Channel
  - For each execution model, we have at least six optimization methods:

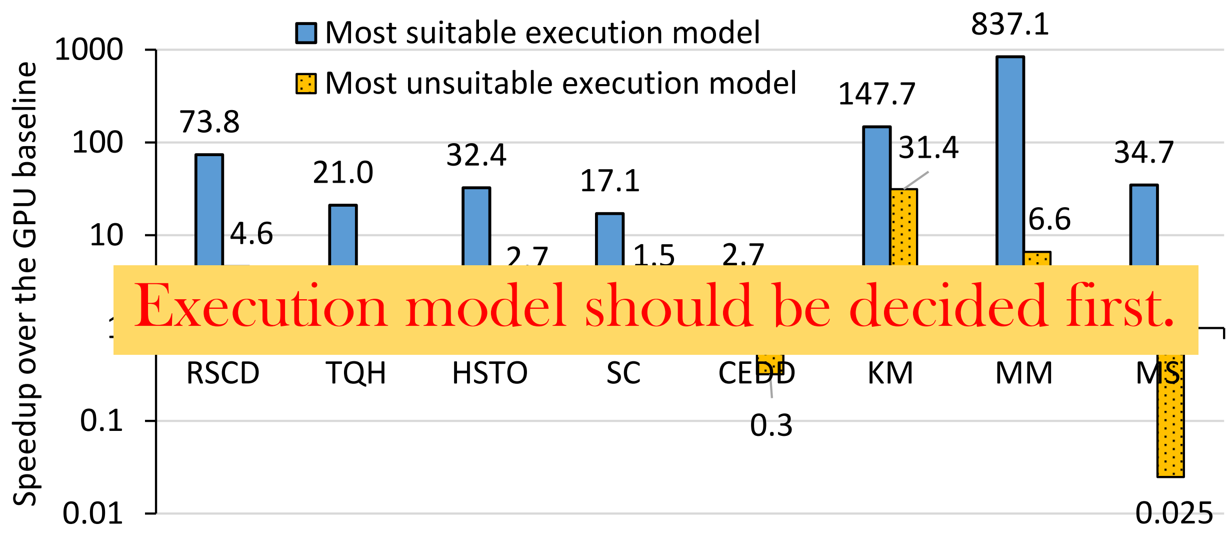
    SM | MC | PM | UL | SIMD | CU
  - For each optimization method, we have different pragmas.

- The compilation time is extremely long!

Effect of Four Execution Models

- Different execution models can significantly affect the performance.

NDRange | SWI | NDRange+Channel | SWI+Channel



Execution model should be decided first.

Can we explicitly determine the most suitable execution model (i.e., whether or not to use two OpenCL features) to optimize OpenCL programs on FPGAs?

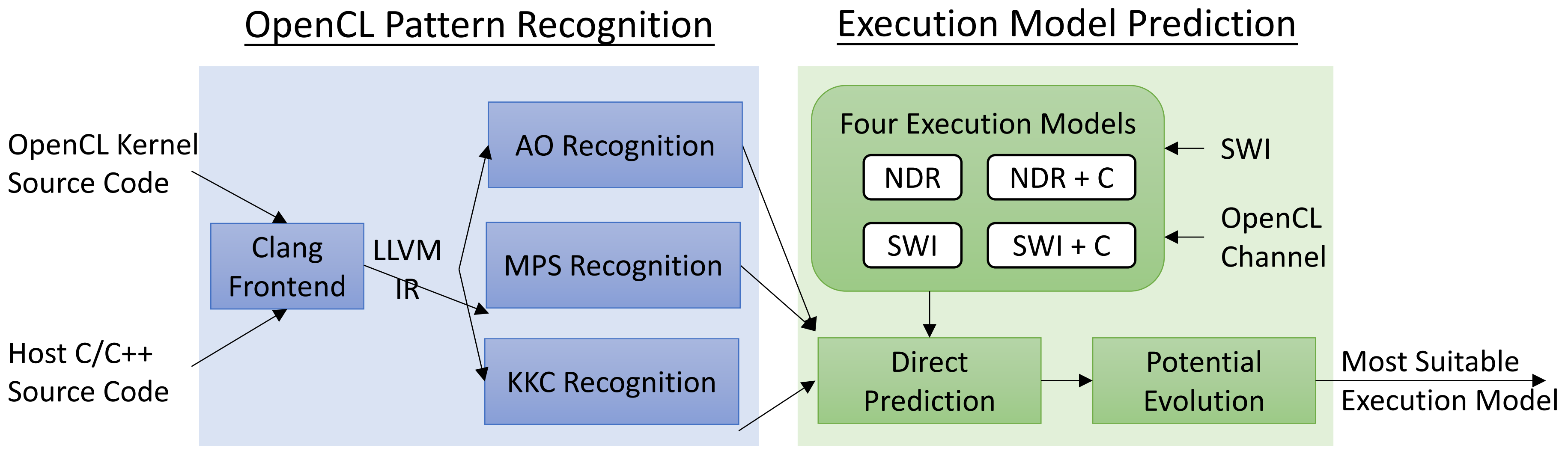
We provide a systematic framework Boyi to automatically determine the most suitable execution model.

Outline

- Background and Motivations
- Our Solution
    - OpenCL Pattern Recognition
    - Execution Model Prediction
- Experiment
- Conclusion

Architecture of Boyi

- Boyi explicitly determines the most suitable execution model to optimize OpenCL programs on FPGAs.
  - OpenCL Pattern Recognition
  - Execution Model Prediction



AO: Atomic Operation

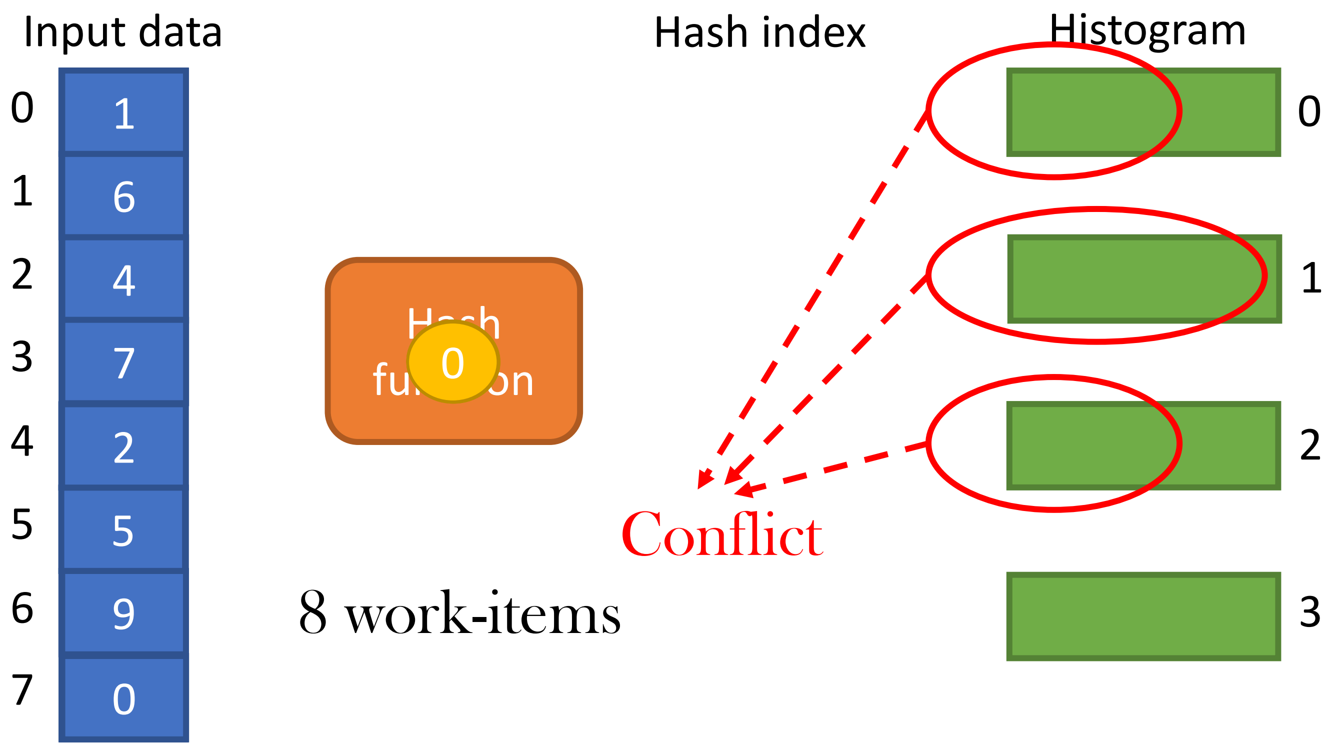
MPS: Multi-Pass Scheme

KKC: Kernel-to-Kernel Communication

Outline

- Background and Motivations
- Our Solution
  - ➢ OpenCL Pattern Recognition
  - ➢ Execution Model Prediction
- Experiment
- Conclusion

OpenCL Pattern: Atomic Operation



- Issues on FPGAs:

  Noticeable resource overhead

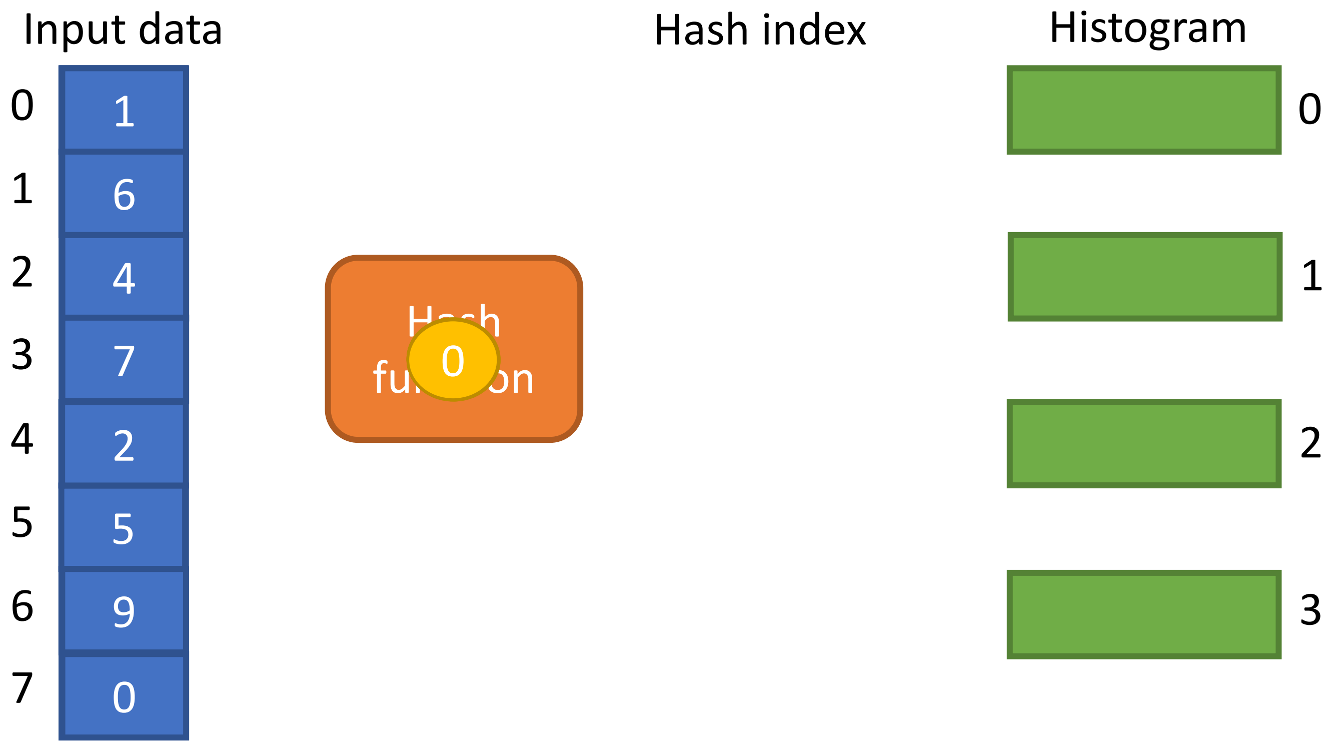
  Long latency and low bandwidth

  Low frequency

→ AO is not a good fit on FPGAs.

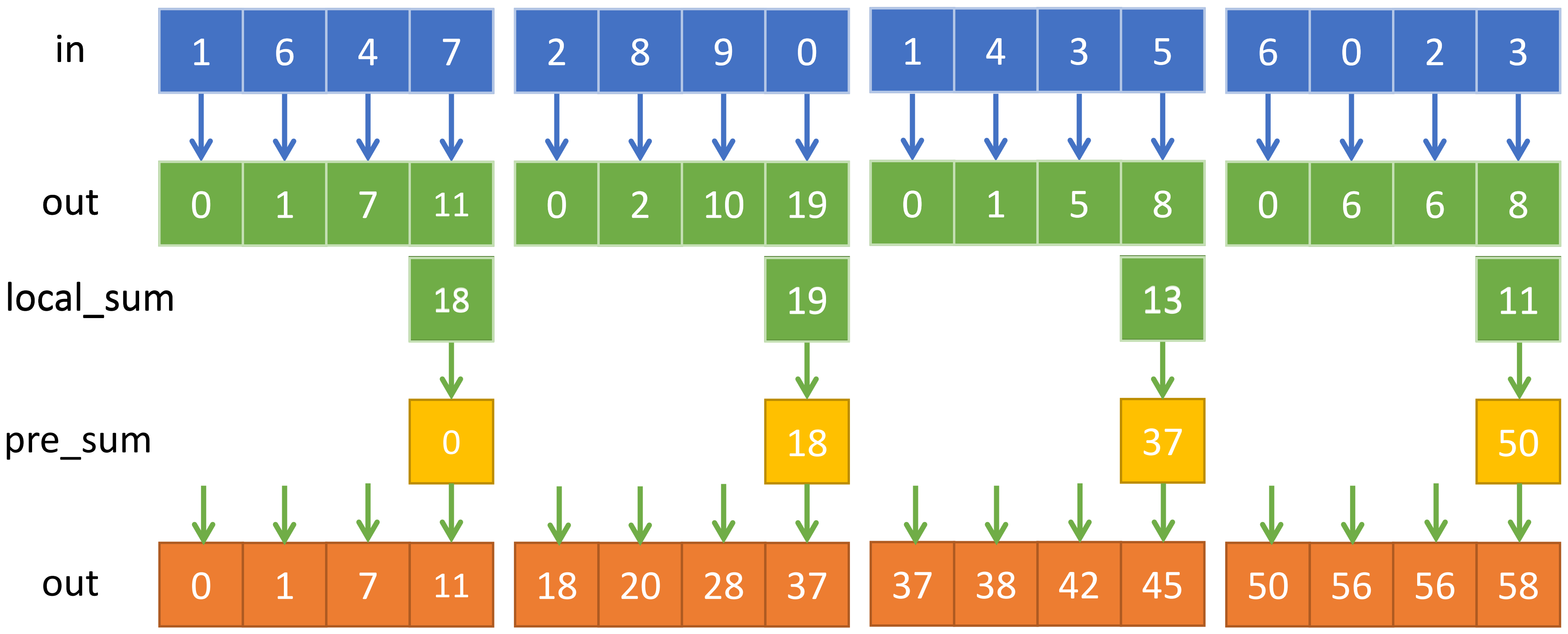
OpenCL Pattern: Atomic Operation

- Potential on FPGAs:



OpenCL Pattern: Multi-Pass Scheme

Step 3: 4 work-groups



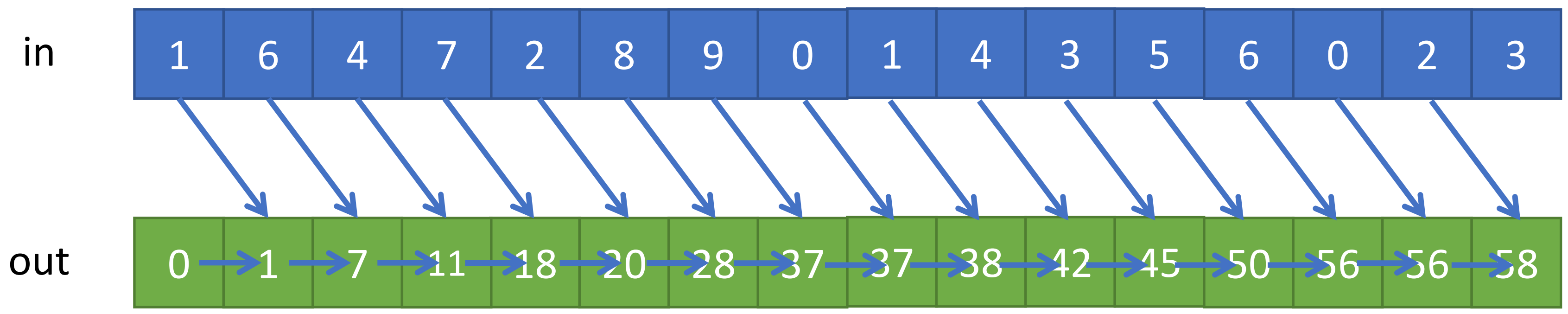
OpenCL Pattern: Multi-Pass Scheme

- Issues on FPGAs:

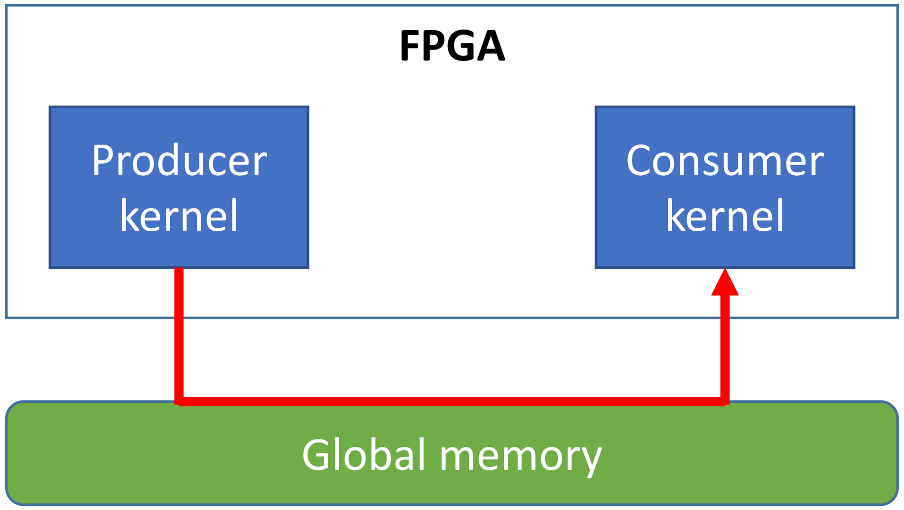
  More memory traffic

  → MPS is not a good fit on FPGAs.

- Potential on FPGAs:



OpenCL Pattern:
Kernel-to-Kernel Communication



- Issues on FPGAs:

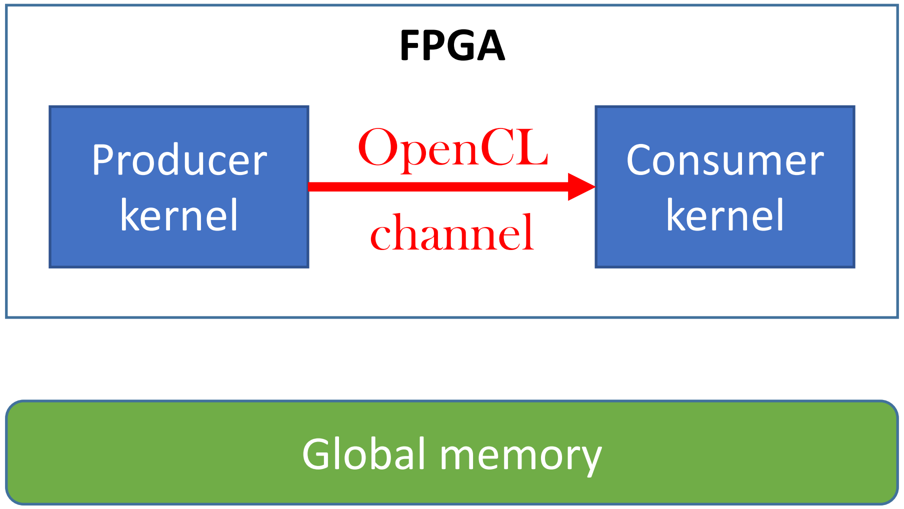
The communication via global memory is expensive.

OpenCL Pattern: Kernel-to-Kernel Communication

- Potential on FPGAs

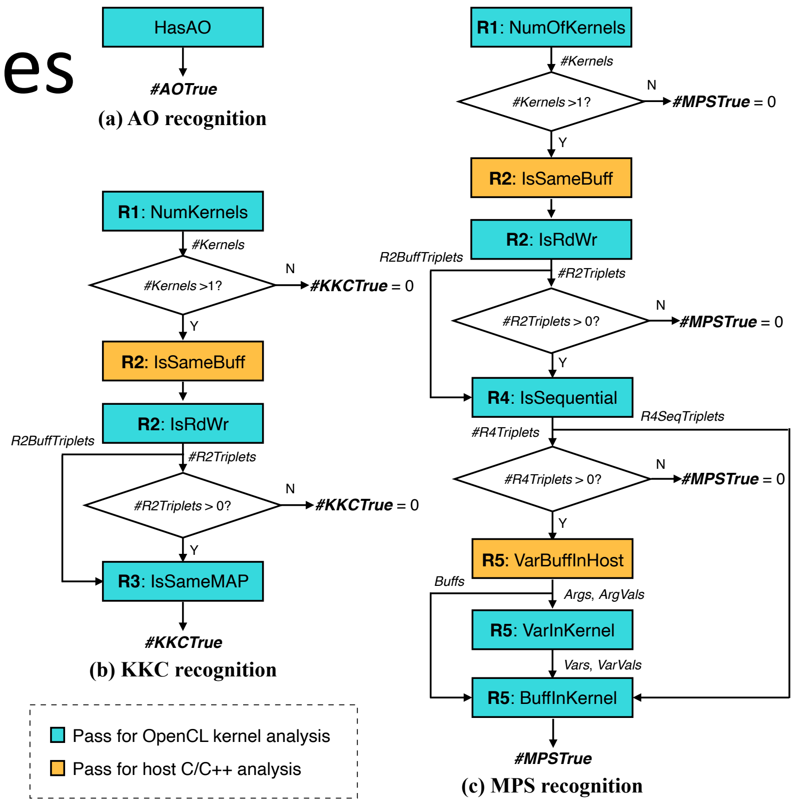
  Reducing the number of memory accesses

  Inter-kernel parallelism (i.e., concurrent kernel execution)



OpenCL Pattern Recognition

- We develop nine LLVM passes to recognize three OpenCL patterns.
  - AO recognition
  - KKC recognition
  - MPS recognition



The implementation details can be found in our paper!

Outline

- Background and Motivations
- Our Solution
  - ➢ OpenCL Pattern Recognition
  - ➢ Execution Model Prediction
- Experiment
- Conclusion

Execution Model Prediction

- Direct prediction

AO	MPS	KKC	Direct prediction
N	N	N	NDR
Y	N	N	SWI
N	Y	N	SWI
Y	Y	N	SWI
N	N	Y	NDR+C
Y	N	Y	SWI+C
N	Y	Y	SWI+C
Y	Y	Y	SWI+C

Kernels with AO and MPS benefit from the SWI kernel.
Kernels with KKC benefit from the OpenCL channel.

Execution Model Prediction

- Potential evolution of SWI

AO	MPS	KKC	Direct prediction	Potential evolution
N	N	N	NDR	NDR
Y	N	N	SWI	SWI+C
N	Y	N	SWI	SWI+C
Y	Y	N	SWI	SWI+C
N	N	Y	NDR+C	NDR+C
Y	N	Y	SWI+C	SWI+C
N	Y	Y	SWI+C	SWI+C
Y	Y	Y	SWI+C	SWI+C

➢Conditions: Sufficient FPGA resource.
The SWI kernel is compute-bound.

Outline

- Background and Motivations
- Our Solution
- Experiment
  - Experimental Setup
  - Effect of Execution Model
  - Prediction of Execution Model
- Conclusion

Experimental Setup

- **Platform**: Terasic DE5a-Net board: Altera Arria 10 GX FPGA and 8GB 2-bank DDR3, with Altera OpenCL SDK version 16.1.
- **Workloads**:

Benchmark	Source	Description	AO	MPS	KKC	Datasets
BFS	Chai	Breadth-First Search	Y	N	N	NY, NE, UT
RSCD		RANSAC	Y	N	Y	2000 iterations
TQH		Task Queue System	Y	N	N	Basket
HSTO		Histogram	Y	N	N	256bins
SC		Stream Compaction	Y	N	N	50%
PAD		Padding	Y	N	N	1000*999
CEDD		Canny Edge Detection	N	N	Y	Peppa, Maradona, Paw
KM	Rodinia	K-Means	N	N	N	25600 points, 8 features
MM	Intel demo	Matrix Multiplication	N	N	N	A: 2k*1k, B: 1k*1k
MS		Mandelbrot Set	N	N	N	640*800, 2000 iterations
PS	CUDA demo	Prefix Sum	N	Y	N	262144 points

Comparison Methodology

- Hypothesis 1: Different execution models lead to significant performance differences.
  - ➢Quantitative comparison among execution models
  - ➢Exploring optimization combinations

- Hypothesis 2: Boyi can predict the most suitable execution model for each OpenCL application.

Hypothesis 1: Different execution model --> different performance

- Quantitative comparison among execution models

App	Number of combinations				Maximum speedup			
	NDR	SWI	NDR+C	SWI+C	NDR	SWI	NDR+C	SWI+C
BFS	17	7	7	9	1.9	**3.1**	1.2	**3.1**
RSCD	25	10	24	46	15.8	4.6	**73.8**	39.7
TQH	9	15		23	1.1	1.3		**21.0**
KM	33	11	10	18	**147.7**	32.8	136.4	31.4
MM	25	9		6	**837.1**	13.3		6.6
MS	7	6		7	**34.7**	0.02		3.2
PS	26	20		12	15.8	**44.4**		**46.2**

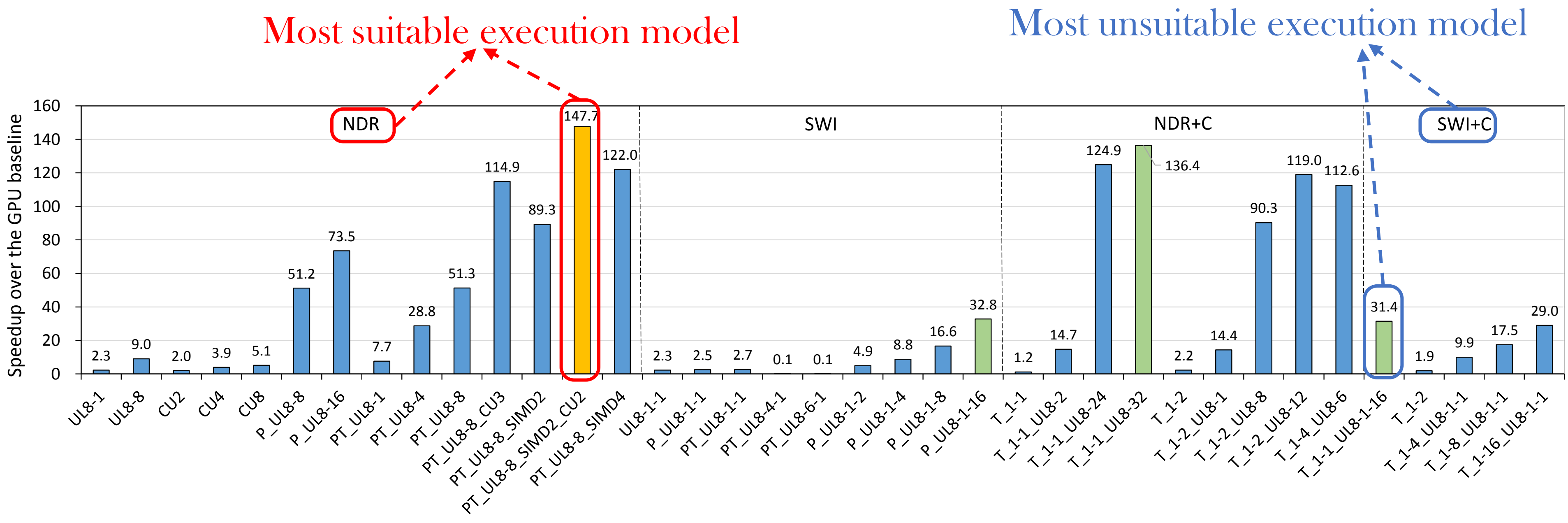
It is critical to decide the most suitable execution model when optimizing OpenCL applications on FPGAs.

models to achieve the best performance.

odels result in significant performance differences.

Hypothesis 1: Exploring optimization combinations for Each Execution Model

- We manually implement sufficient number of optimization combinations (subset) for KM, such that we reach the near-to-optimal optimization combination for each execution model.



Hypothesis 2: Boyi Predicts the Right Execution Model

Application	AO	MPS	KKC	Actual	Predicted
BFS	Y	N	N	SWI	SWI
RSCD	Y - - → N	N	Y	NDR+C	SWI+C - - → NDR+C
TQH	Y	N	N	SWI+C	SWI+C ※
HSTI	Y	N	N	SWI+C	SWI+C ※
SC	Y	N	N	SWI+C	SWI+C ※
PAD	Y	N	N	SWI+C	SWI+C ※

The actual and predicted execution models roughly match.

KM	N	N	N	NDR	NDR
MM	N	N	N	NDR	NDR
MS	N	N	N	NDR	NDR
PS	N	Y	N	SWI	SWI

SWI+C ※ indicates the potential evolution of SWI

End-to-end Performance Comparison

- Performance comparison to existing works

Application	Ours (ms)	Existing work (ms)	Our/Existing Speedup
RSCD [1]	0.8	28.9	38.3
TQH [1]	66.9	150.6	2.3
HSTO [1]	38.8	487.9	12.6
CEDD [1]	161.9	237.8	1.5
MM [2]	9.1	34.3	3.8
MS [2]	27.2	944.1	34.7

[1] S. Huang et al., “Analysis and modeling of collaborative execution strategies for heterogeneous cpu-fpga architectures”, *ICPE*, 2019.
[2] Intel. Intel SDK for OpenCL Design Examples. 2018

Outline

- Background and Motivations
- Our Solution
- Experiment
- Conclusion