# *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

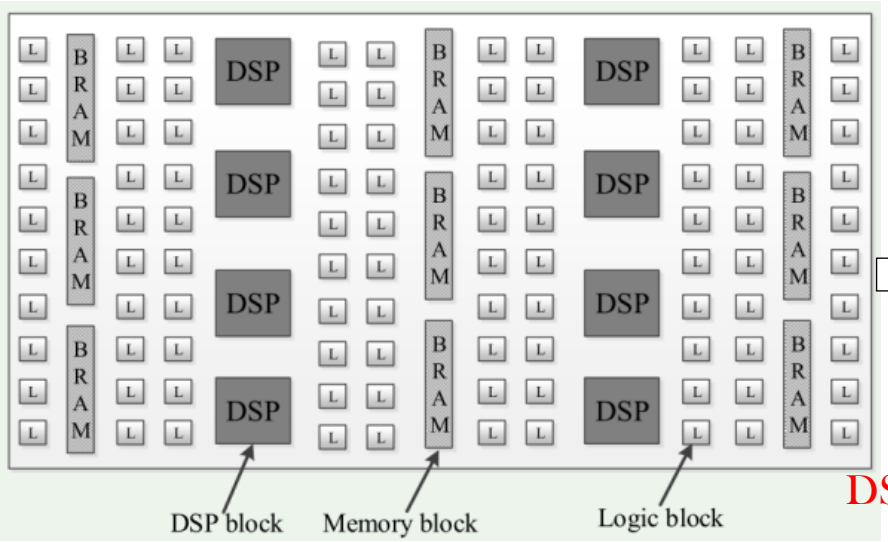
- Background and Motivations
- Our Solution
- Experiment
- Conclusion

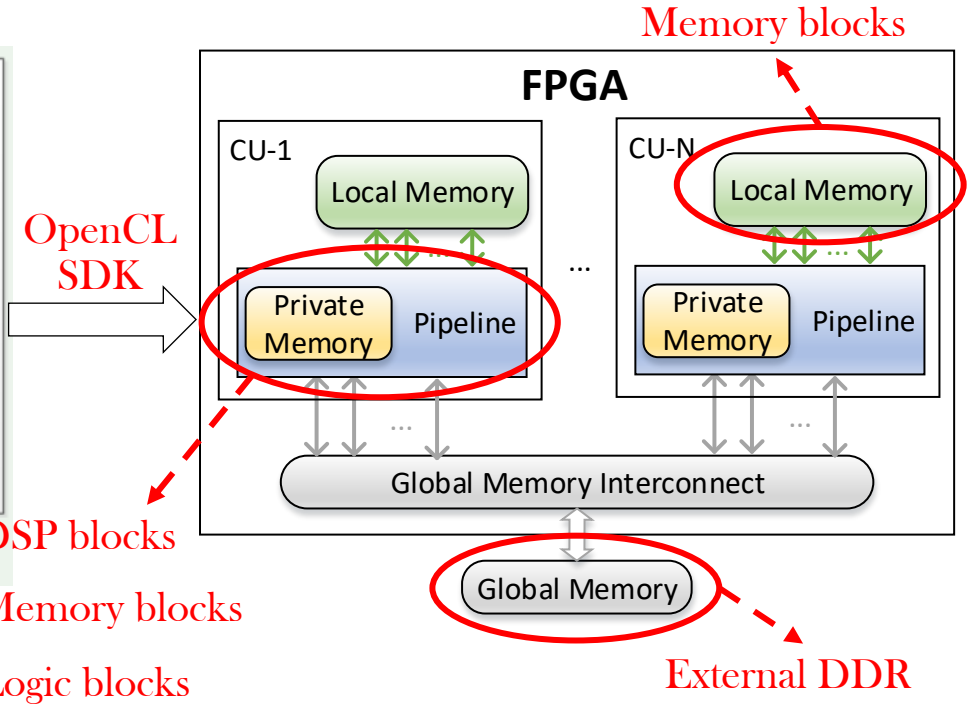
# What is OpenCL?

- 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

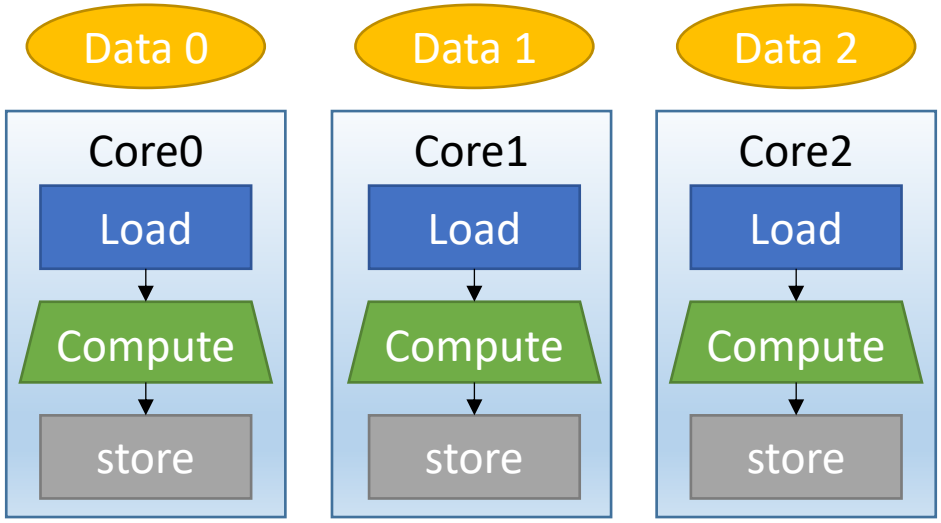



- Hardware-centric → fine-gained parallelism
- Users need to program with HDL.

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

# Conventional OpenCL: NDRange Kernel

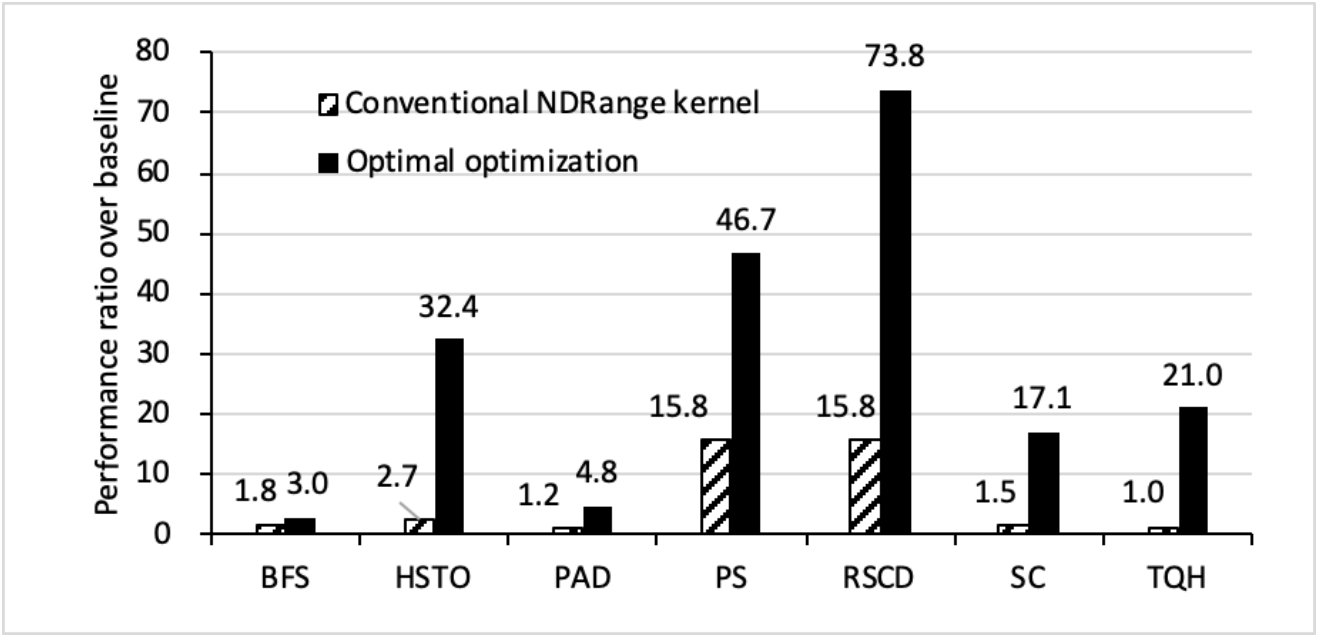



Explicit multi-threaded → executing the same operation on multiple data concurrently

- How conventional OpenCL performs on an FPGA?

# Issue of Conventional OpenCL

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



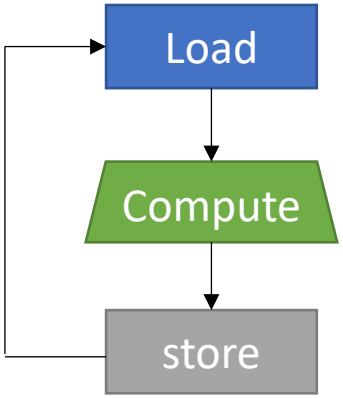

1.6x 12.1x 4.1x 3.0x 4.7x 11.3x 21.0x

**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.

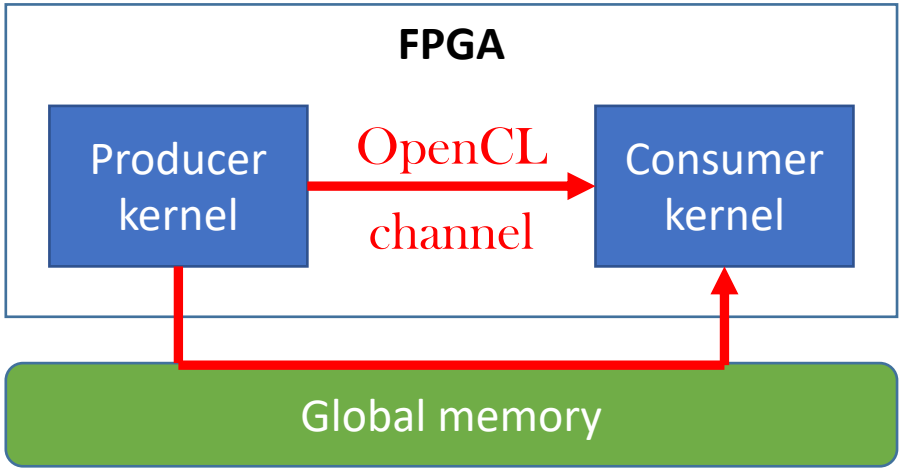

8 Data Points



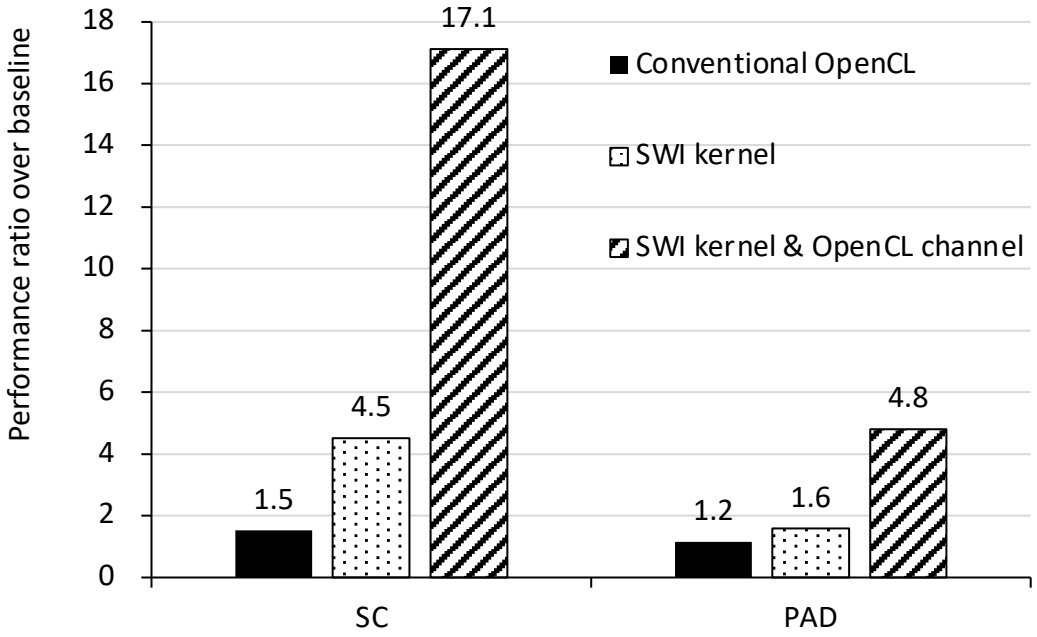

Pipelined parallelism→ No conflict among work items.

# 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



# Effect of Two OpenCL Features




An order of magnitude performance improvement over conventional OpenCL!

# 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.

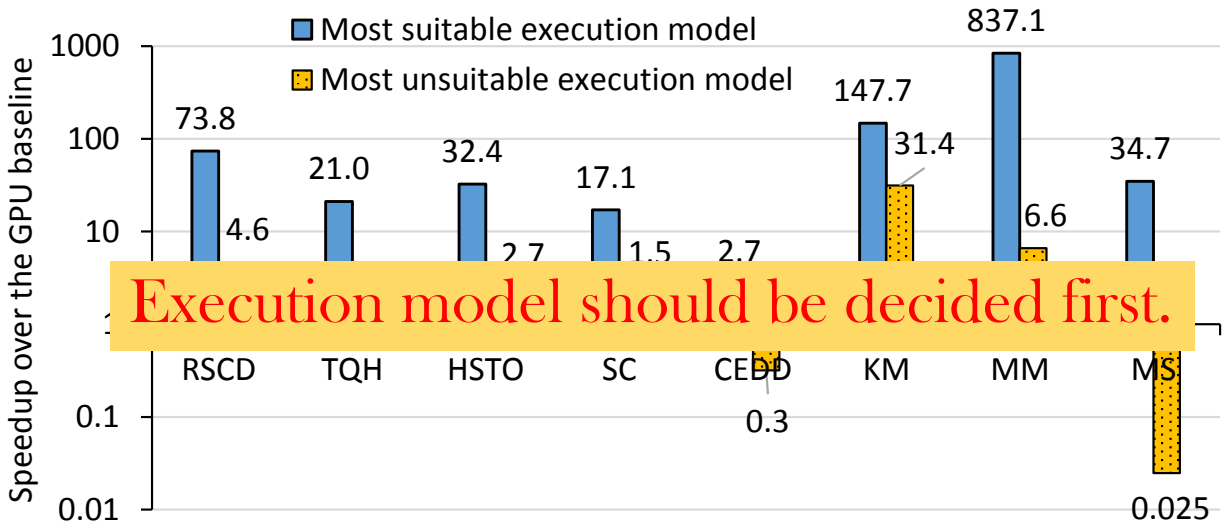
The compilation time is extremely long!

There is no systematic study to guide OpenCL programmers on efficiently leveraging these two OpenCL features.

# Observation

- Different execution models can significantly affect the performance.

NDRange | SWI | NDRange+Channel | SWI+Channel



# Our Goal

- 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?

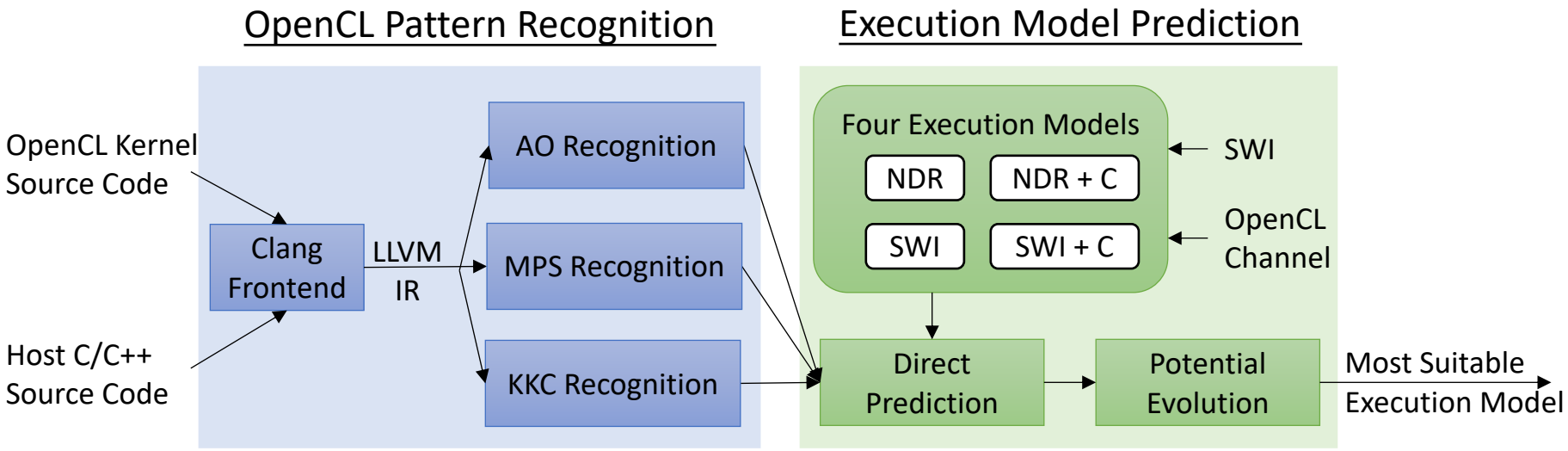
- We provide a systematic framework to make such automated decisions.

# 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

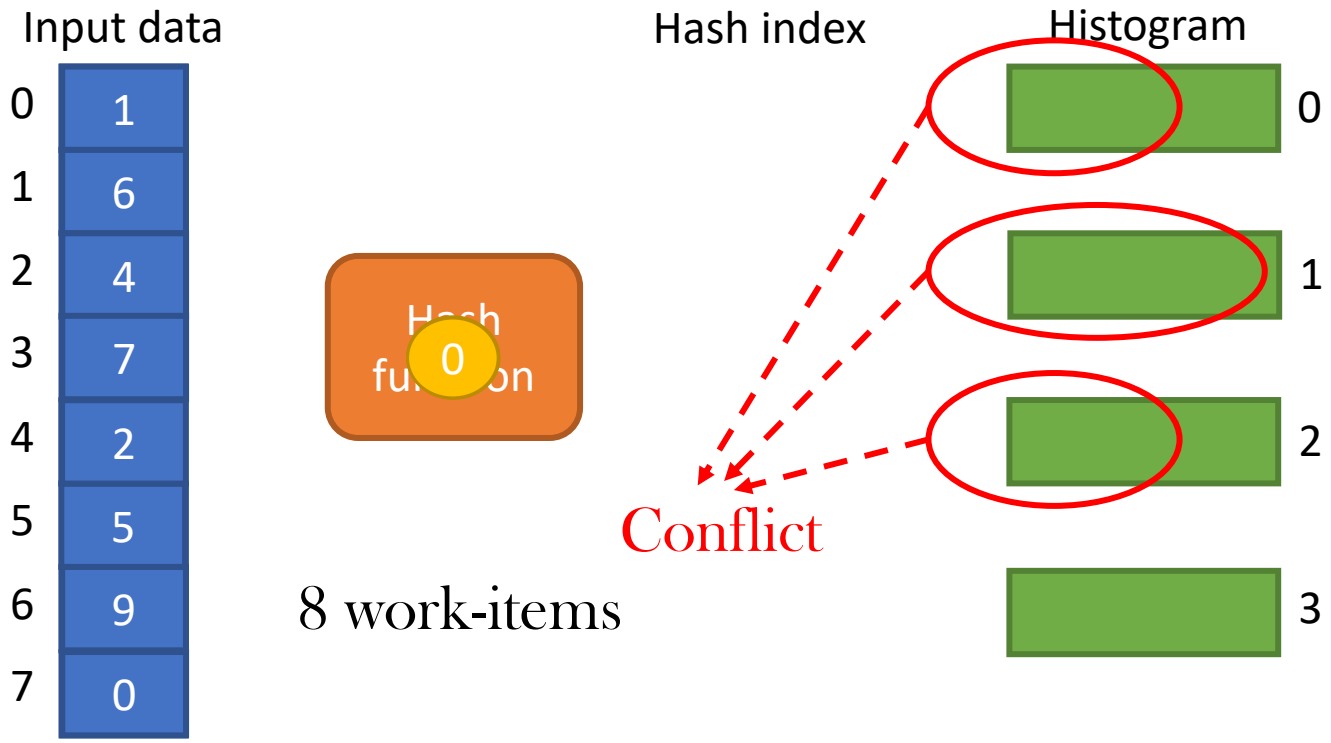



AO: Atomic Operation, MPS: Multi-Pass Scheme, KKC: Kernel-to-Kernel Communication

# Outline

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

# Pattern AO

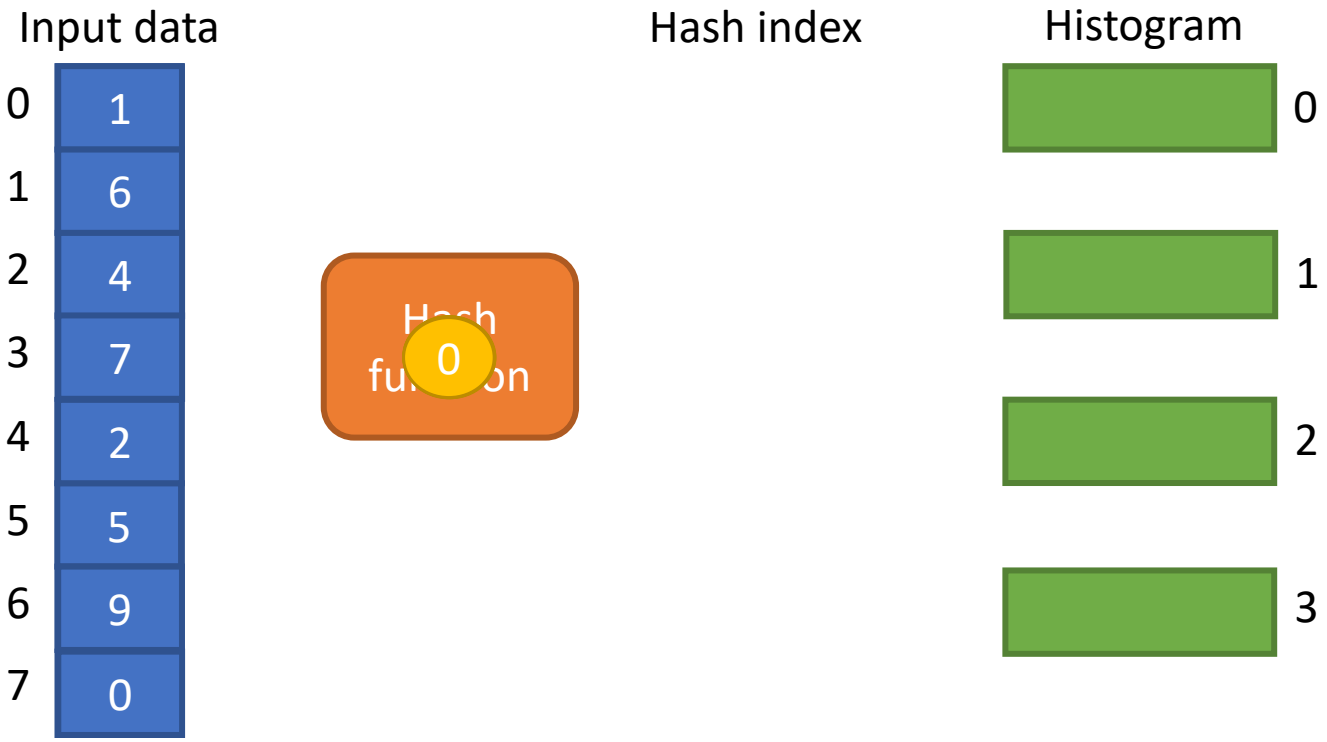



- Issues on FPGAs:
  - Noticeable resource overhead
  - Long latency and low bandwidth
  - Low frequency

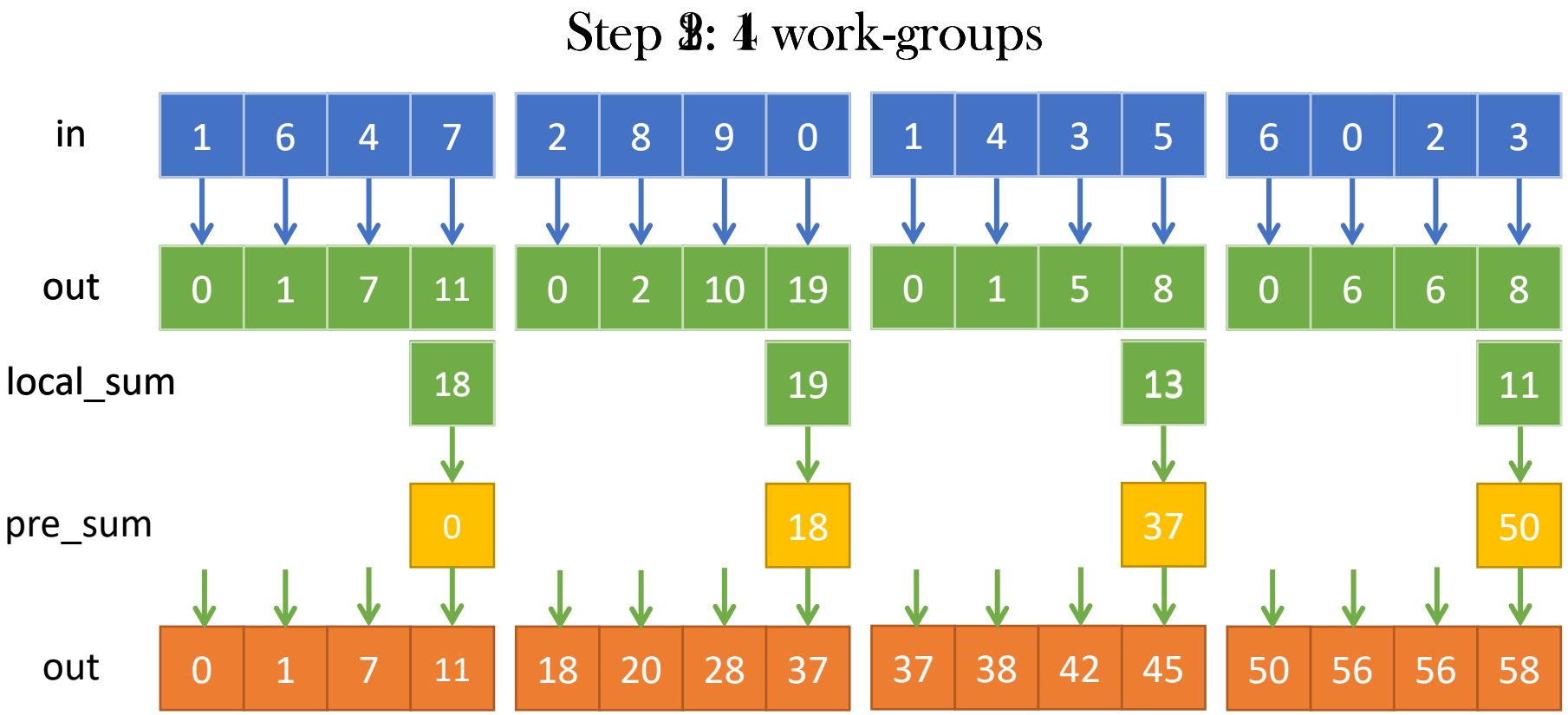
→ AO is not a good fit on FPGAs.

# Pattern AO

- Potential on FPGAs:



# Pattern MPS


Step 2: 4 work-groups
in
1 6 4 7
2 8 9 0
1 4 3 5
6 0 2 3
out
0 1 7 11
0 2 10 19
0 1 5 8
0 6 6 8
local_sum
18
19
13
11
pre_sum
0
18
37
50
out
0 1 7 11
18 20 28 37
37 38 42 45
50 56 56 58

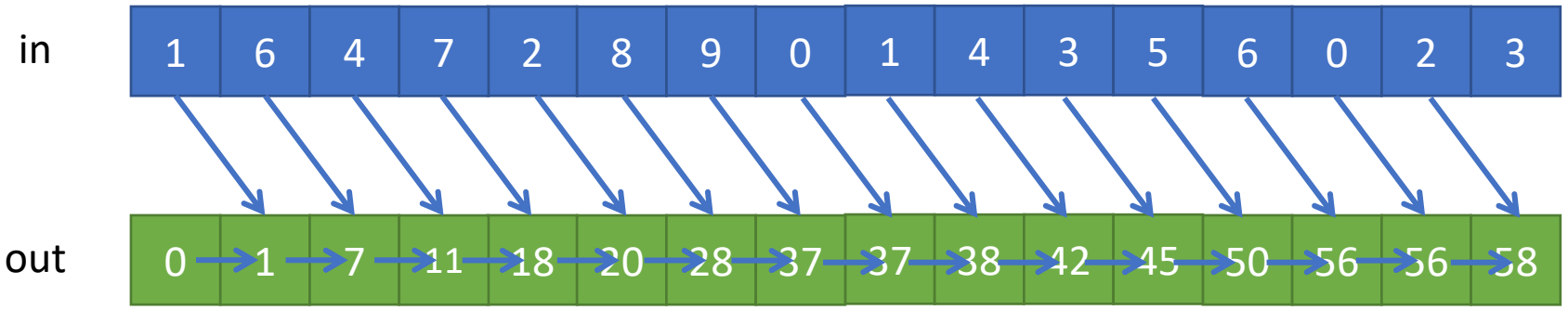
# Pattern MPS

- Issues on FPGAs:

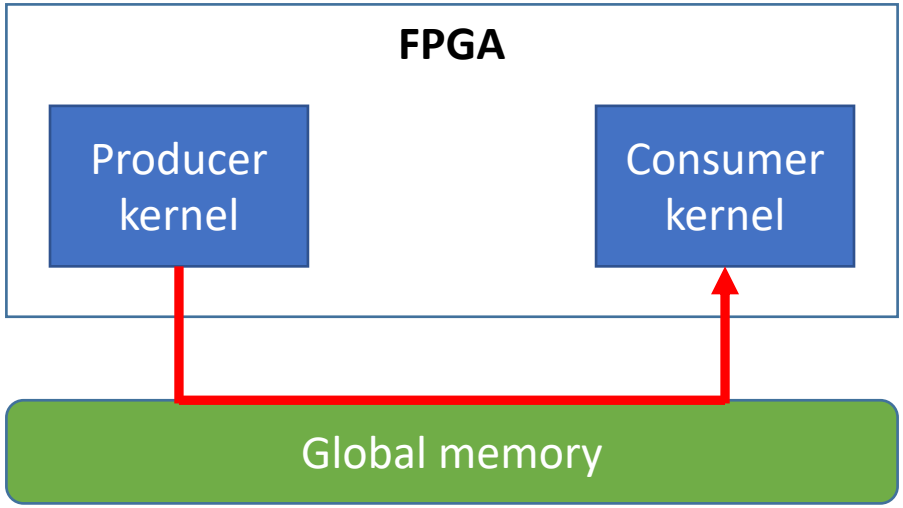
  More memory traffic

  → MPS is not a good fit on FPGAs.

- Potential on FPGAs:



# Pattern KKC




- Issues on FPGAs:

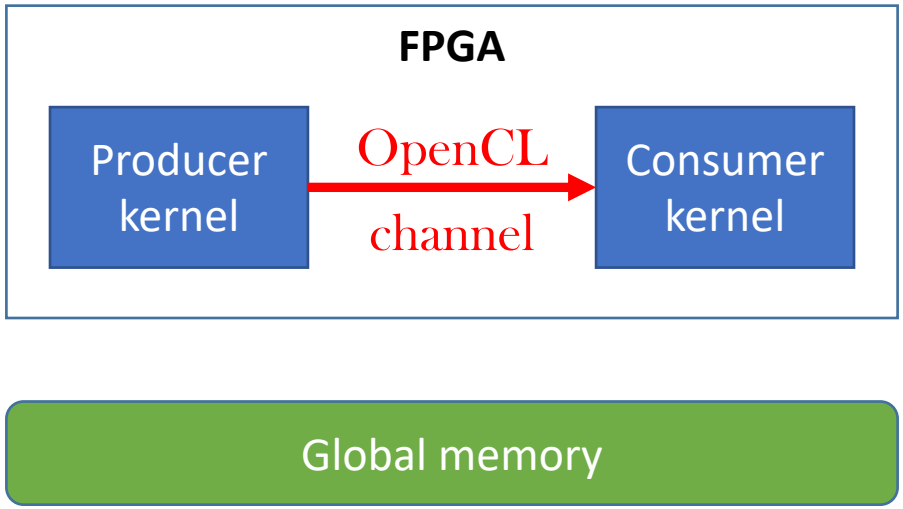
The communication via global memory is expensive.

# Pattern KKC

- 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

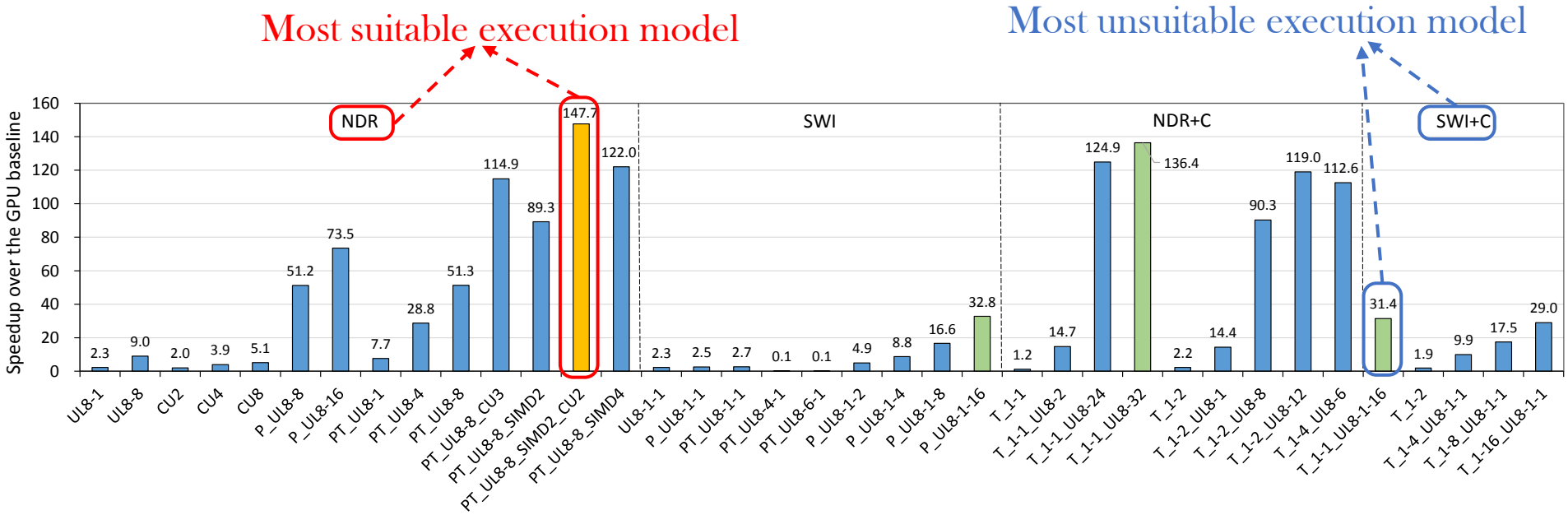
# Experimental Setup

- **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 |

# Performance Comparison

- Exploring optimization combinations




Performance speedup of a subset of optimization combinations over baseline for KM. “CUx” indicates x CUs, “SIMDx” indicates the kernel vectorization factor x, “ULx-y-z” indicates the loop unrolling factors x,y,z to the inner, middle and outer loop respectively. “1-x” indicates a multi-kernel design with one producer kernel and x consumer kernel(s). “P” indicates the use of private memory for input feature array and “T” indicates the transposition of input feature array for a better access pattern.

# Performance Comparison

- Comparison of 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.

significant performance differences.

# Performance Comparison

- Comparison of execution models

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

S. Huang et al., "Analysis and modeling of collaborative execution strategies for heterogeneous cpu-fpga architectures", *ICPE*, 2019.

# Prediction of Execution Model

| Application | AO | MPS | KKC | Actual | Prediction |
|---|---|---|---|---|---|
| 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 ※ |
| CEDD | N | | | | |
| KM | N | | | | |
| MM | N | N | N | NDR | NDR |
| MS | N | N | N | NDR | NDR |
| PS | N | Y | N | SWI | SWI |

The actual and predicted execution models roughly match.

SWI+C ※ indicates the potential evolution of SWI

# Outline

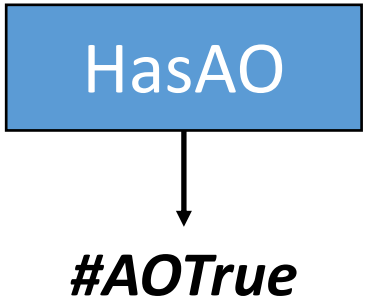
- Background and Motivations
- Our Solution
- Experiment
- Conclusion

# Conclusion

- Two OpenCL features are needed to exploit the performance potential of FPGA, since the architecture of FPGA is significantly different from that of GPU.
- We propose Boyi, a systematic framework for deciding the most suitable execution model of an OpenCL application.
- Our automatic prediction of execution model can roughly match the real experimental results.

# OpenCL Pattern Recognition

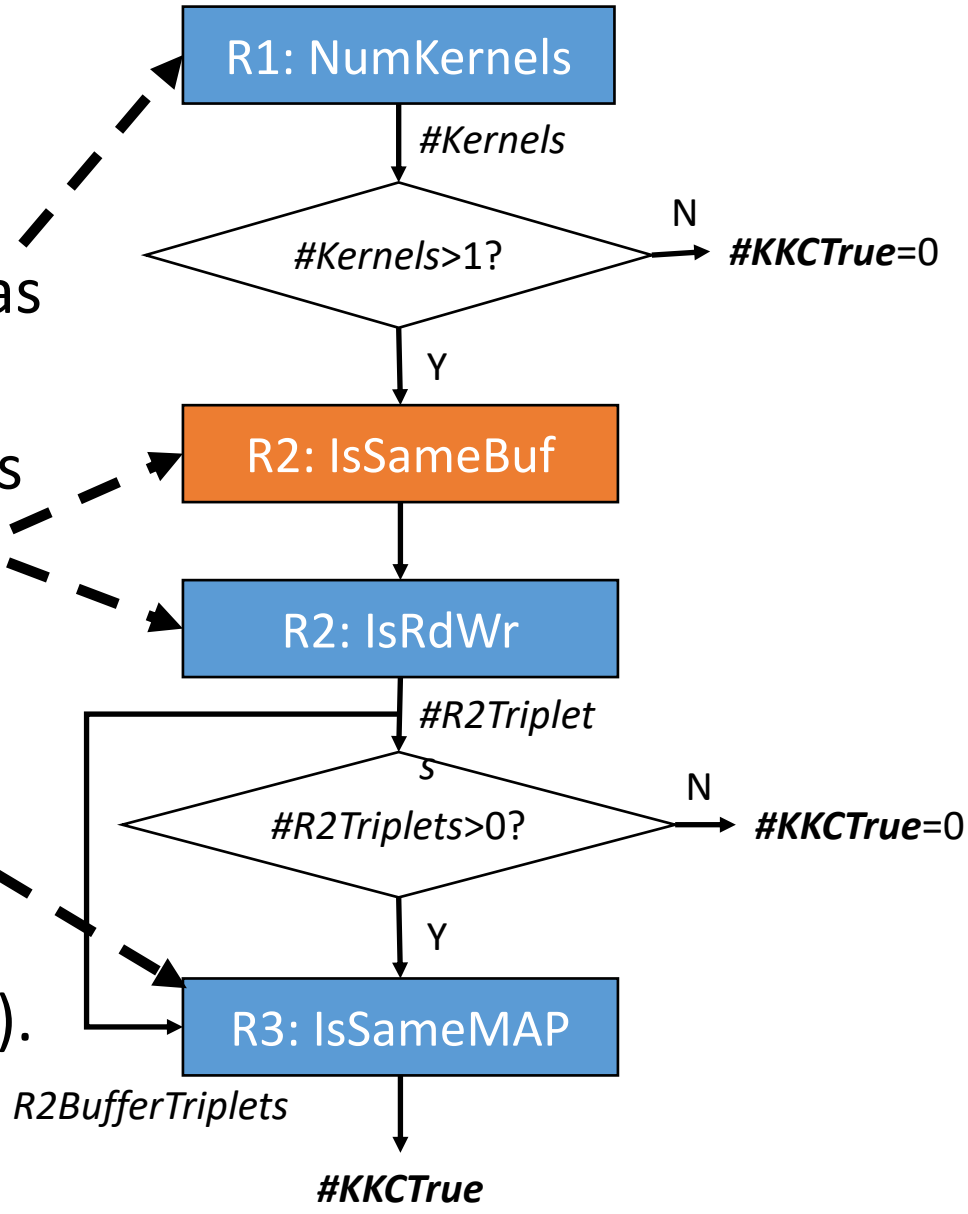
- An LLVM-based OpenCL pattern recognition mechanism
  - AO recognition




  - KKC recognition
  - MPS recognition

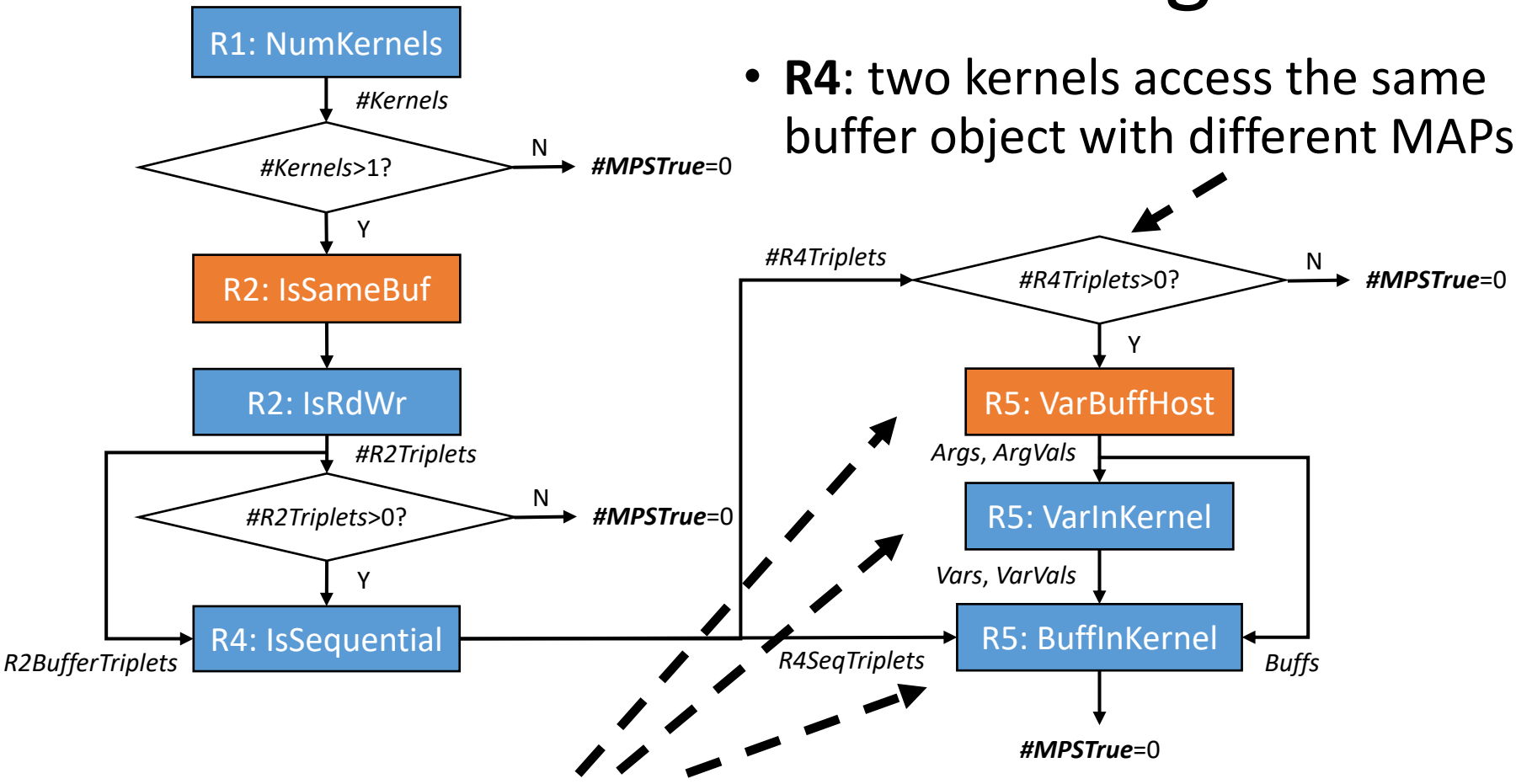
# KKC Recognition

- **R1**: the OpenCL application has two or more OpenCL kernels.
- **R2**: the consumer kernel reads from the same buffer object (in global memory) as a producer kernel writes to.
- **R3**: the producer and the consumer kernels access the buffer object with the same memory access pattern (MAP).



# MPS Recognition

- **R4**: two kernels access the same buffer object with different MAPs




- **R5**: the intermediate buffer objects are not needed when the OpenCL application is mapped to one work-group

# Four Execution Models

- General comparison of four execution models:

| | NDR | SWI | NDR+C | SWI+C |
|---|---|---|---|---|
| Programmability | Moderate | High | Low | Low |
| Computing parallelism | High | Low | High | Moderate/High |
| Memory traffic | High | Low | Moderate | Low |

- NDR can achieve higher computing parallelism.
- SWI is more easy to program and requires a lower memory traffic.
- OpenCL channel will increase the programming difficulty, reduce the memory traffic and lead to higher computing parallelism.