Design and Preliminary Evaluation of OpenACC Compiler for FPGA with OpenCL and Stream Processing DSL

Yutaka WATANABE 1),

Jinpil LEE 2), Kentaro SANO 2), Taisuke BOKU 1)3), Mitsuhisa SATO 1)2)

1) Graduate School of Systems and Engineering, University of Tsukuba

2) Riken Center for Computational Science

3) Center for Computational Sciences, University of Tsukuba

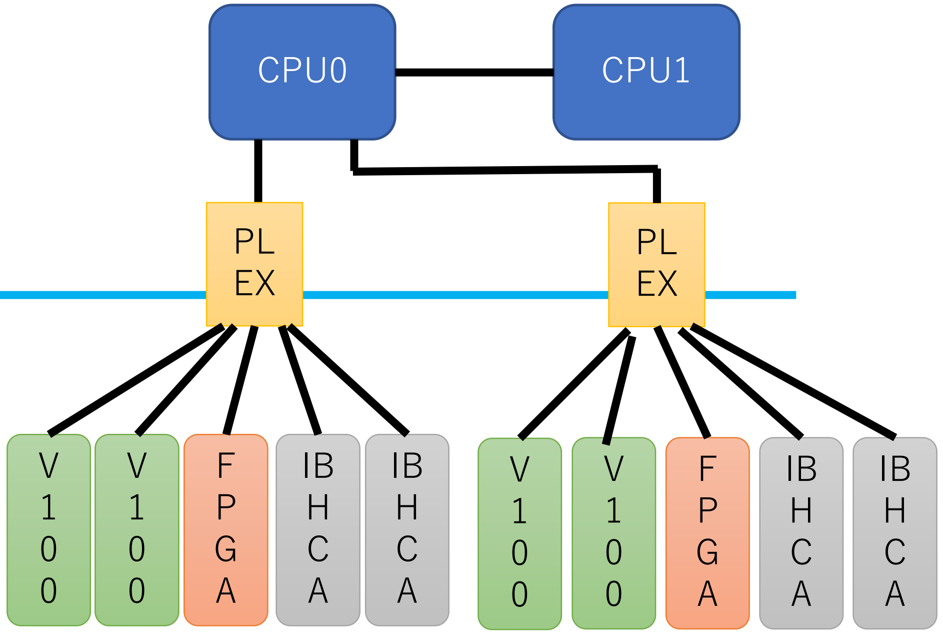
Outline

Background

- Increasing the demand of reconfigurable computing with FPGA
  - Better watt performance
  - Application specific hardware
  - Inter-FPGA direct communication

- FPGA clusters are operated in several institutions.
  - Noctua @ Paderborn Univ, Cygnus @ Univ. Tsukuba
  - Catapult Project @ Microsoft

- Programming is one of the problem to make FPGA used widely in HPC
  - HLS is still difficult for application developers



Cygnus @ CCS, Univ. of Tsukuba

The programming for FPGAs

- HDL (Hardware Description Language)
  - Able to specify register-transfer level operation
  - Optimal design may not be created unless by FPGA expert
  - High cost

- HLS by OpenCL
  - Relaxed the complexity of programming with HDL
  - Easy to write than HDL, but code generation mechanism is almost hidden
    - We need to write codes practically

  - Programming with OpenCL for FPGA is completely different from that for GPU

OpenCL

To use FPGAs used widely in HPC

- Need to make the programming for FPGA easy
  - To make easier for application developers
  - Need more easier way with performance portability

- Solution: Supporting directive-based programming model
  - Such as OpenACC or OpenMP
  - Previous reseaches: OpenARC for FPGA, OmpSs@FPGA, etc

```
#pragma acc kernels …
for (i=0; i<SIZE; i++) {
      sum += i;
}
```

Motivation

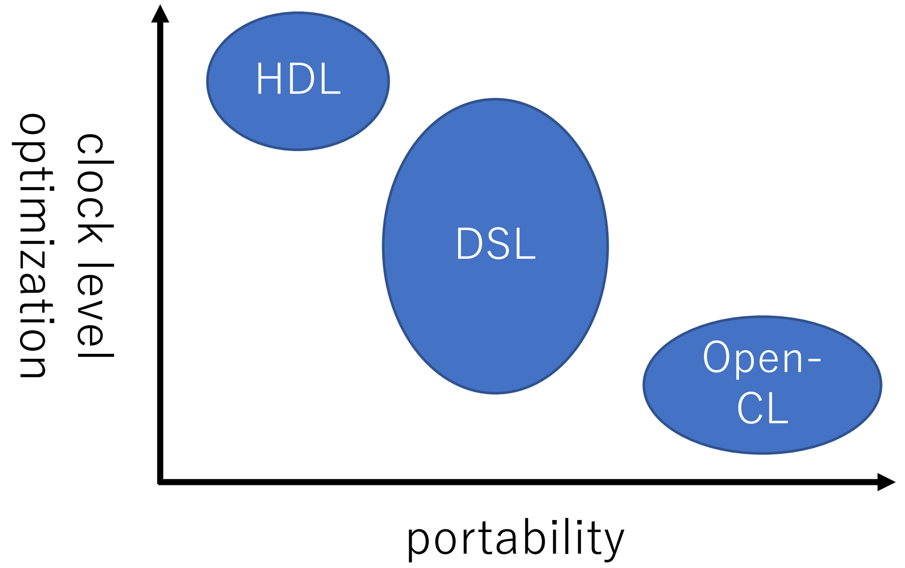
- Directive-based programming model should be supported even FPGAs
  - For end-users/application developers can easily use FPGA
  - OpenMP or OpenACC are widely used in HPC
- Can OpenCL be used as an intermediate code in the directive-based compiler?
  - It has good portability, but it is not clear what hardware is generated with OCL
  - It could be better to use more low-level language

OpenACC -> xxx -> FPGA: What could be used as xxx?

- There are many languages for FPGA

- HDL
  - ◯ : Cycle-level explicit optimization
  - ✗ : Low portability between boards, high dev cost
- OpenCL
  - ◯ : High portability
  - ✗ : The code gen mechanism is hidden

- DSL for FPGAs
  - ◯ : low-level optimization than HLS
  - ✗ : Portability, higher dev cost than HLS



Integration of OpenCL and DSL:
=> Will enable lower level optimization with DSL in the OpenACC compiler

SPGen (Stream Processing Generator)

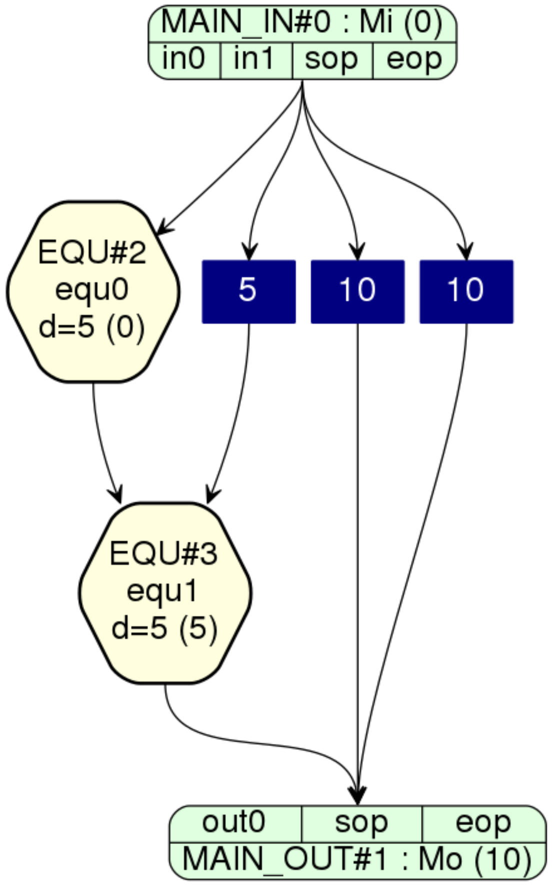
- Example of DSL for FPGAs (and we will use in our compiler)
- A stream processing framework for FPGA
  - User writes an SPD program
    - Compiler analyze the program into dataflow, and translate into HDL
  - User writes formulas, HDL module inclusions, data in/out
    - Loop structure or indirect access cannot be described

```
Name saxpy;

Main_In {Mi::in0, in1, sop, eop};
Main_Out {Mo::out0, sop, eop};

EQU equ0, tmp = 3.1337*in0;
EQU equ1, out0 = tmp + in1;

DRCT (Mo::sop, Mo::eop) =  (Mi::sop, Mi::eop);
```



SPGen (Cont'd)

- Fluid Simulation with SPGen
  - 519GFLOPS (SP), 9.67GFLOPS/w with Arria10 FPGA
  - (estimation) 6149GFLOPS (SP) could be archived with Stratix10
- C2SPD: a C frontend for SPGen
  - An original directive-based compiler for SPGen
  - Including optimization like loop transformation with Polly (LLVM)

Sano, Kentaro, and Satoru Yamamoto. "FPGA-Based Scalable and Power-Efficient Fluid Simulation using Floating-Point DSP Blocks." IEEE Transactions on Parallel and Distributed Systems 28, no. 10 (2017): 2823-2837.

Lee, Jinpil, Tomohiro Ueno, Mitsuhisa Sato, and Kentaro Sano. "High-productivity Programming and Optimization Framework for Stream Processing on FPGA." In Proceedings of the 9th International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies, p. 5. ACM, 2018.

The overview of OpenACC compiler with OpenCL and DSL

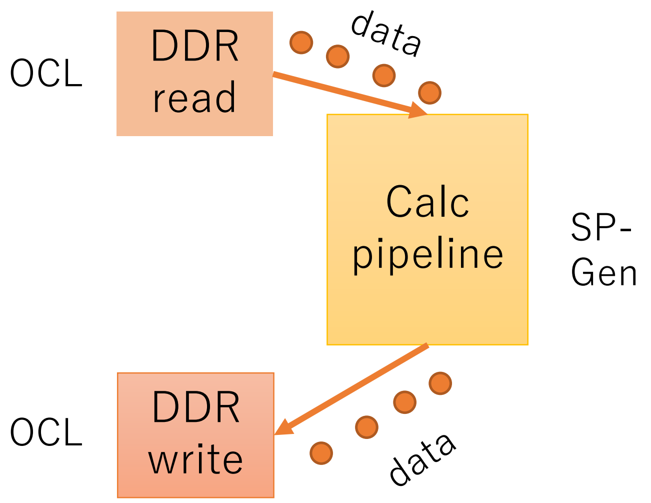
OpenACC program

```
#pragma acc kernels …
{
  for (int i=0; i<SIZE; i++) {
   … // to be SPGen kernel
  }
}
```

compiler

OpenCL host program

```
createBufferOnFPGA(…);
writeDataToFPGA(…);
enqueueTaskToFPGA(…);
…
if (endTaskOnFPGA()) {
  readDataFromFPGA(…);
}
```



- User writes OpenACC
- Compiler translates into SPGen and OpenCL
- OpenCL is used for offloading interface and memory access

OpenCL device program

```
kernel void kern(
  global float *a, global float *b
) {
  for (int i=0; i<SIZE;i++) {
    b[i] = spgen_kern(a[i]]);
  }
}
```

SPGen program

```
Name spgen_kern
Main_In {a, sop, eop};
Main_Out {b, sop, eop};

…
```

prototype implementation on omni-compiler

- Implemented by using the omni-compiler by RIKEN and Univ. Tsukuba
  - Extending existing OpenACC to OpenCL transpliration for GPU or PEZY-SC

- Implementations：
  - Support single work item model for FPGA
    - => OpenACC to naive OpenCL
  - SPGen code generation
    - Initial implementation of vectorization, reduction
    - Optimizations are not implemented yet

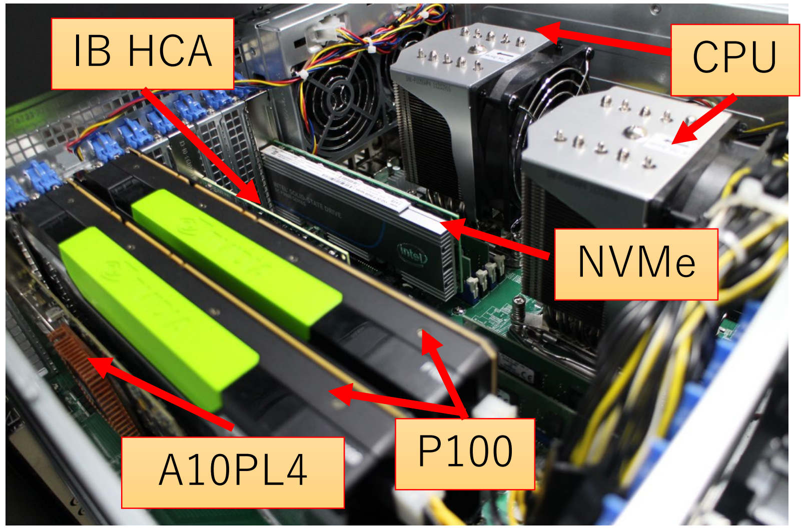
Compilation steps

1. Finding loops in the kernel
2. Vectorization if vector clause is specified
3. Reduction if reduction clause is specified
4. Decoupling the memory access from kernel body
5. Code generation
   1. Mem access -> OpenCL device kernel
   2. Computation body => SPGen program

$\left.\vphantom{\begin{matrix}1\\2\\3\\4\end{matrix}}\right\}$ In the intermediate expression of the Omni-compiler

Experimental Settings

- PPX (Pre-PACS X) system @ Univ. Tsukuba
  - Arria 10 FPGA, a former HPC-use FPGA
- Used kernel3, 9, 12 from the livemore kernels for
  - A kernel set to measure the vectorization performance
- Compare two method:
  - OpenACC to OpenCL only
  - OpenACC to OpenCL + DSL
- Measured execution time only in FPGA
- Used single floating precision



Hardware configuration of PPX	
CPU	Intel Xeon E5-2660 v4 x 2
Host DRAM	DDR4-2400 16GB x 4
FPGA Board	BittWare A10PL4 (Intel Arria10 GX1150) PCIe Gen3 x8
FPGA chip	10AX115N3F40E2SG
FPGA DRAM	DDR4-2133 4GB x 2

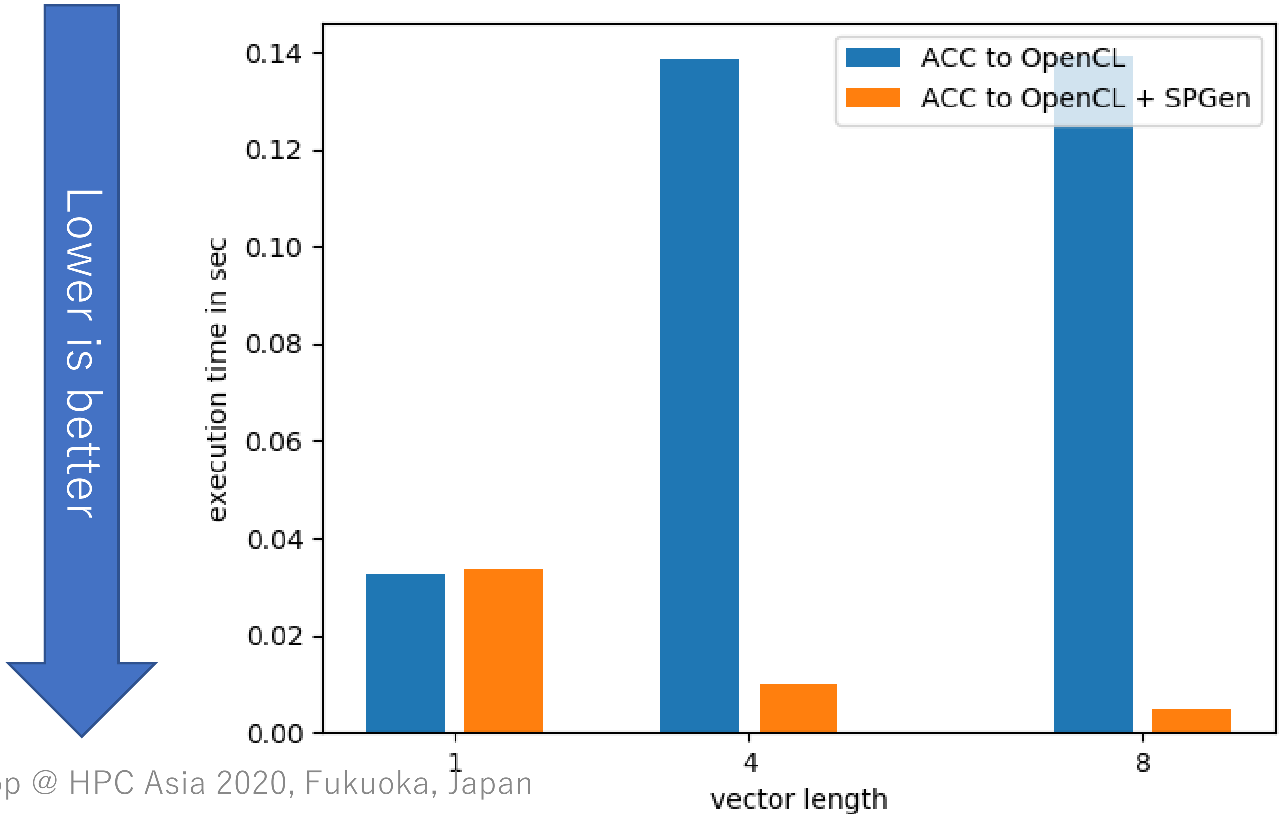
Software configuration of PPX	
OS	CentOS 7.3 x64
Host Compiler	GNU C Compiler 4.8.5
FPGA compiler	Intel FPGA SDK for OpenCL 17.1.2.304

Evaluation: kernel3

	\Vector Length	1	4	8
frequency (MHz)	ACC to OpenCL + SPGen	296.64	259.6	272.75
	ACC to OpenCL	309.98	289.77	288.68
ALMs	ACC to OpenCL + SPGen	11%	11%	11%
	ACC to OpenCL	10%	10%	11%
DSPs	ACC to OpenCL + SPGen	2	8	16
	ACC to OpenCL	1	4	8
M20K	ACC to OpenCL + SPGen	12%	16%	17%
	ACC to OpenCL	12%	14%	14%

```
q = 0.0;
for ( k=0 ; k<n ; k++ ) {
    q += z[k]*x[k];
}
```

- Resource Usage:
  - w/ SPGen > OpenCL only
    - Consumes 2x DSPs
- Performance：
  - w/ SPGen > OpenCL only
    - Lack optimization for OpenCL only



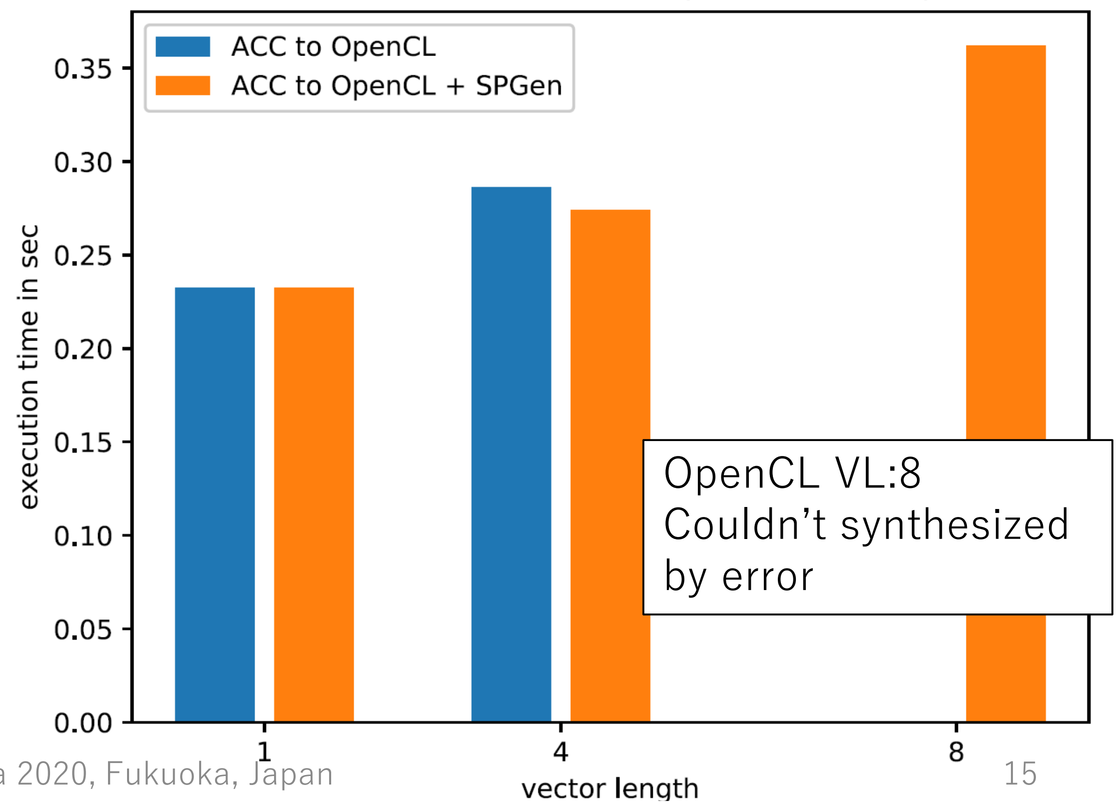
Evaluation: kernel9

```
for ( i=0 ; i<n ; i++ ) {
  px[i][0] =
      dm28*px[i][12] + dm27*px[i][11] +
      dm26*px[i][10] +  dm25*px[i][ 9] +
      dm24*px[i][ 8] + dm23*px[i][ 7] +
      dm22*px[i][ 6] +
      c0*( px[i][ 4] + px[i][ 5]) + px[i][ 2];
}
```

- Resource usage：
  - w/ SPGen > OpenCL only
    - Big difference on DSP, M20K
- Performance：
  - Almost same if Vector len <=4
  - Memory bound
    - Lack of memory optimization

	\Vector Length	1	4	8
frequency (MHz)	ACC to OpenCL + SPGen	263.64	193.72	195.61
	ACC to OpenCL	284.81	271.58	
ALMs	ACC to OpenCL + SPGen	11%	15%	20%
	ACC to OpenCL	12%	14%	
DSPs	ACC to OpenCL + SPGen	21	84	168
	ACC to OpenCL	12	48	
M20K	ACC to OpenCL + SPGen	17%	33%	53%
	ACC to OpenCL	14%	19%	

Lower is better



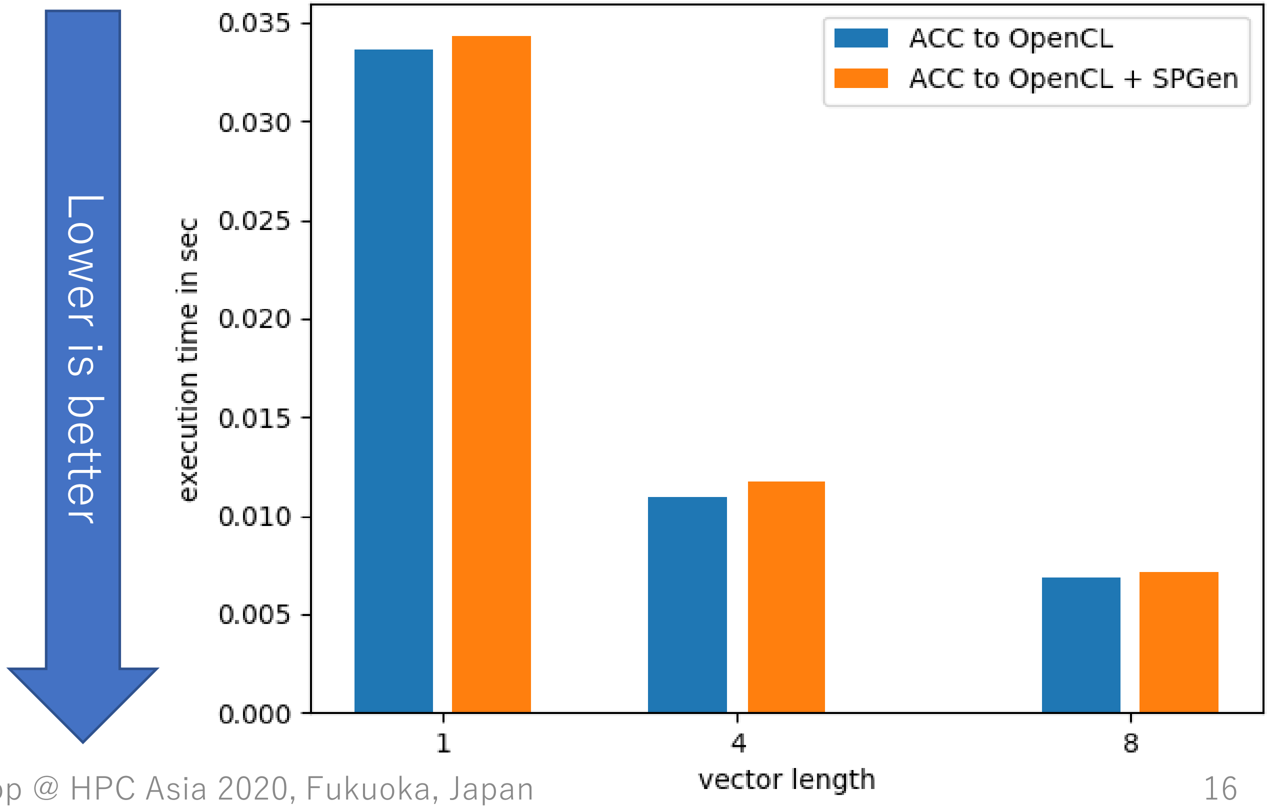
OpenCL VL:8
Couldn’t synthesized
by error

Evaluation: kernel12

```
for ( k=0 ; k<n ; k++ ) {
    x[k] = y[k+1] - y[k];
}
```

	\Vector Length	1	4	8
frequency (MHz)	ACC to OpenCL + SPGen	292.22	277.16	265.45
	ACC to OpenCL	299.04	291.88	277.46
ALMs	ACC to OpenCL + SPGen	11%	11%	11%
	ACC to OpenCL	11%	11%	11%
DSPs	ACC to OpenCL + SPGen	1	4	8
	ACC to OpenCL	1	4	8
M20K	ACC to OpenCL + SPGen	12%	14%	14%
	ACC to OpenCL	12%	13%	13%

- Resource Usage:
  - Almost same
- Performance：
  - Almost same



Discussion: Resource requirement

- OpenCL+SPGen consumes more resources
  - M20K： SPGen uses M20K to adjust pipeline size
    - => could be improved by using shift register instead of M20K
  - Redundant code generation in the OpenACC compiler
    - => could be improved by better code generation
  - DSP ： FMA mode or dot-product mode is not used
    - => Improved by better code generation in the OpenACC compiler

Discussion: Performance difference

- OpenCL+SPGen method shows lower kernel frequency
  - Critical Path seems to be FIFO module for adjusting pipeline size
    => Check if improved by using shift register
  - Lack of optimizations in the compiler
    => Check if improved by redaundant code elimination or code optimization

Related works

- OpenARC for FPGA by ORNL
  - Extend OpenACC + original directive
  - OpenCL is used as intermediate code for FPGA
    - Easier to write than OpenCL directly
- OmpSs@FPGA by BSC
  - OpenMP related programming model supporting Xilnix SoC platform FPGA
  - HLS C is used for intermediate code. However, user should optimize the target code with HLS pragma

Lee, Seyong, Jungwon Kim, and Jeffrey S. Vetter. "Openacc to fpga: A framework for directive-based high-performance reconfigurable computing." In 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 544-554. IEEE, 2016.

Lambert, Jacob, Seyong Lee, Jungwon Kim, Jeffrey S. Vetter, and Allen D. Malony. "Directive-Based, High-Level Programming and Optimizations for High-Performance Computing with FPGAs." In *Proceedings of the 2018 International Conference on Supercomputing*, pp. 160-171. ACM, 2018.

Filgueras, Antonio, Eduard Gil, Daniel Jimenez-Gonzalez, Carlos Alvarez, Xavier Martorell, Jan Langer, Juanjo Noguera, and Kees Vissers. "Ompss@ zynq all-programmable soc ecosystem." In Proceedings of the 2014 ACM/SIGDA international symposium on Field-programmable gate arrays, pp. 137-146. ACM, 2014.

Conclusion and future work

- Designed an OpenACC compiler for FPGA with OpenCL and DSL
  - DSL will enable lower-level optimization in the OpenACC compiler
- Performance evaluation and comparison with OpenACC to OpenCL-only method
  - Lower level code generation helps better pipelining
  - Found disadvantage in resource consumption and performance in some case
- Performance may be improved by better code generation or SPGen itself
  - Disadvantage in resource consumption and performance in some case
  - Think to use OpenCL only if OpenCL can perform well without SPGen
- Future Work
  - Better code generation including redundant-code elimination
  - Code optimization with loop transformation

Design and Preliminary Evaluation of OpenACC Compiler for FPGA with OpenCL and Stream Processing DSL

Yutaka WATANABE 1),

Jinpil LEE 2), Kentaro SANO 2), Taisuke BOKU 1)3), Mitsuhisa SATO 1)2)

1) Graduate School of Systems and Engineering, University of Tsukuba

2) Riken Center for Computational Science

3) Center for Computational Sciences, University of Tsukuba