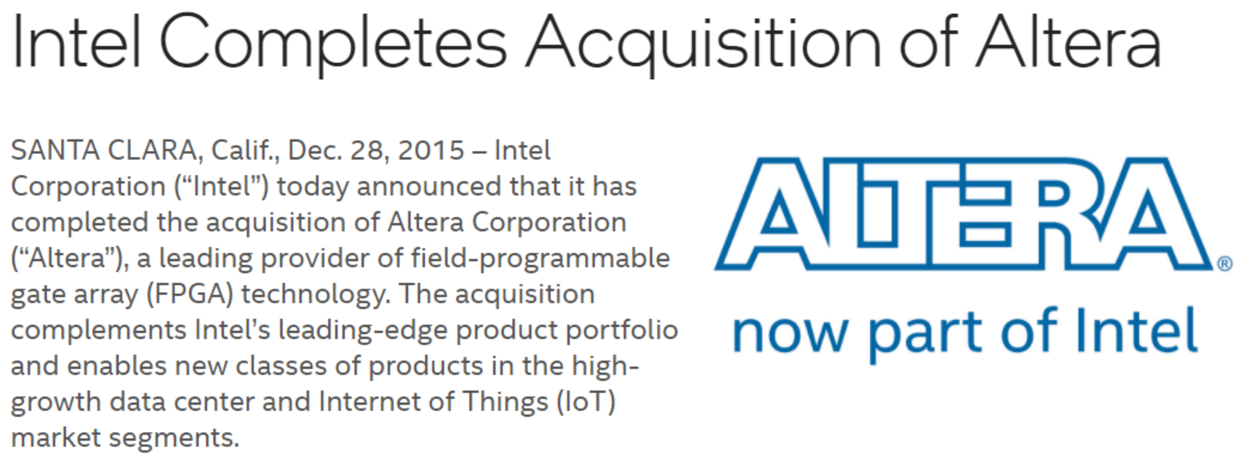
Toward Evaluating High-Level Synthesis Portability and Performance Between Intel and Xilinx FPGAs

Anthony Cabrera[1], Aaron Young[1], Jacob Lambert[2], Zhili Xiao[3], Amy An[3], Seyong Lee[1], Zheming Jin[1], Jungwon Kim[1], Jeremy Buhler[3], Roger Chamberlain[3], Jeffrey Vetter[1]

[1]Oak Ridge National Laboratory [2]University of Oregon [3]Washington University in St. Louis

IWOCL/SYCLcon 2021

FPGAs Are Gaining Traction as Moore's Law Wanes


Intel Completes Acquisition of Altera

SANTA CLARA, Calif., Dec. 28, 2015 – Intel Corporation ("Intel") today announced that it has completed the acquisition of Altera Corporation ("Altera"), a leading provider of field-programmable gate array (FPGA) technology. The acquisition complements Intel's leading-edge product portfolio and enables new classes of products in the high-growth data center and Internet of Things (IoT) market segments.

ALTERA® now part of Intel

AMD to Acquire Xilinx

Creating the Industry's High Performance Computing Leader

AMD | XILINX®

Microsoft

Project Catapult

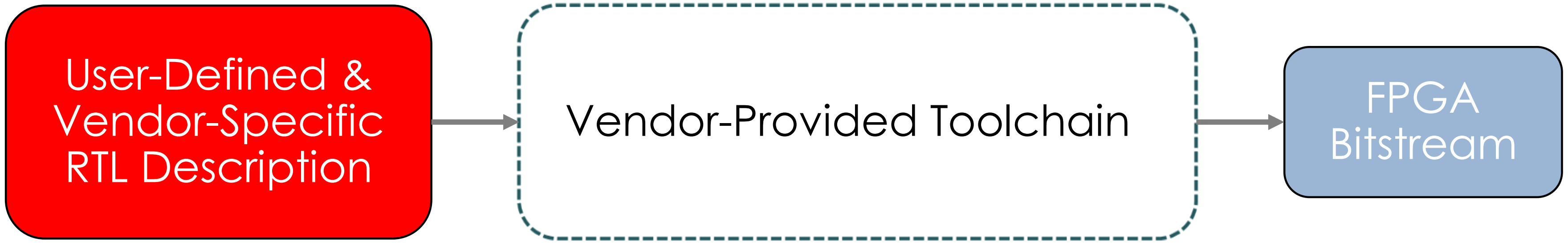
Project Brainwave

Amazon EC2 F1 Instances

Enable faster FPGA accelerator development and deployment in the cloud

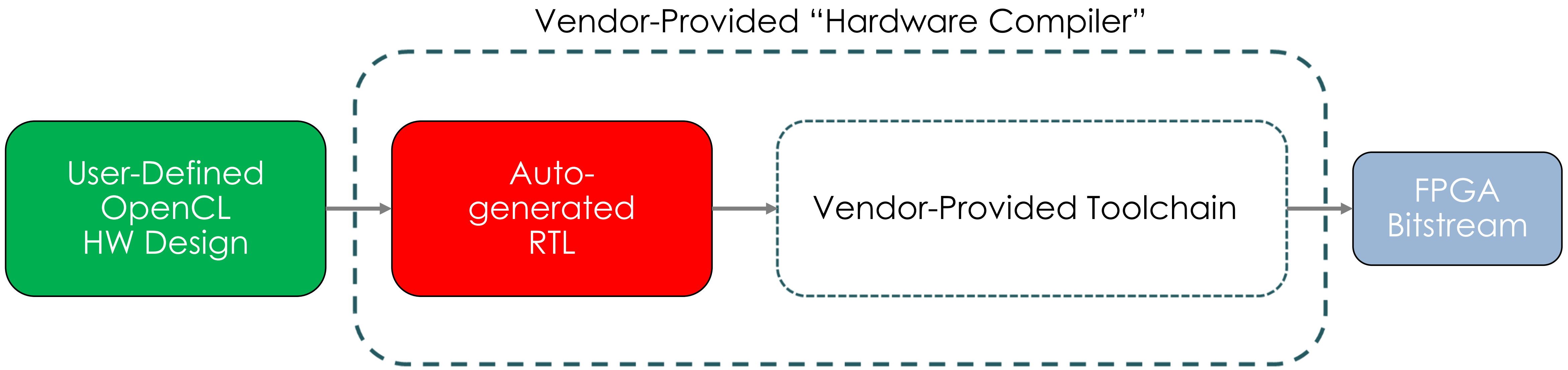
Making FPGAs More Programmable through High Level Synthesis (HLS)

Traditional Path to FPGA Bitstream


User-Defined & Vendor-Specific RTL Description
Vendor-Provided Toolchain
FPGA Bitstream

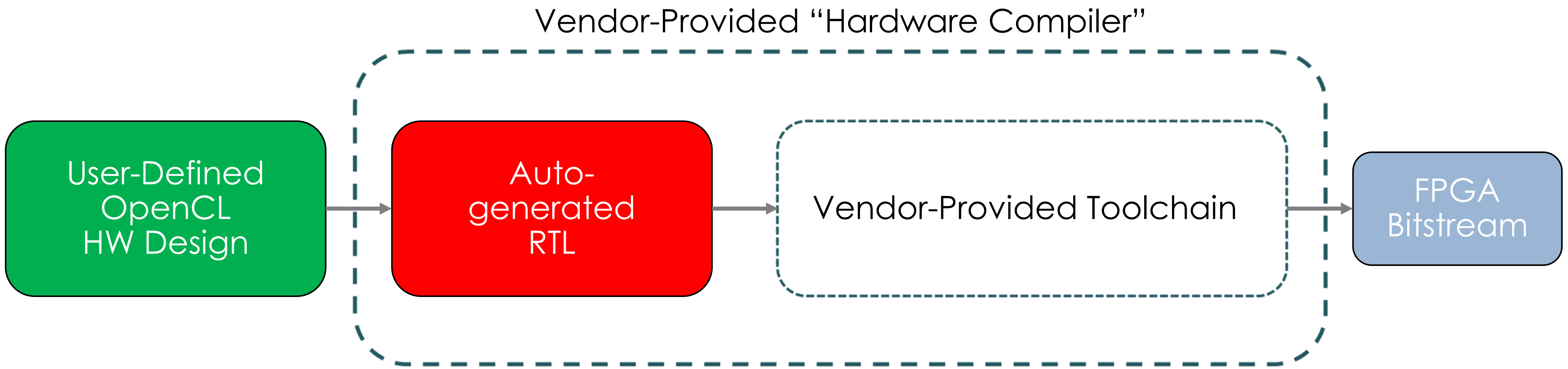
Making FPGAs More Programmable through High Level Synthesis (HLS)

Using HLS to Generate FPGA Bitstream


Vendor-Provided "Hardware Compiler"
User-Defined OpenCL HW Design
Auto-generated RTL
Vendor-Provided Toolchain
FPGA Bitstream

Making FPGAs More Programmable through High Level Synthesis (HLS)

Using HLS to Generate FPGA Bitstream


Vendor-Provided "Hardware Compiler"
User-Defined OpenCL HW Design
Auto-generated RTL
Vendor-Provided Toolchain
FPGA Bitstream

Intel and Xilinx support OpenCL C for designing hardware, but…

How portable and performant are HLS designs between Intel and Xilinx FPGAs?

How portable and performant are HLS designs between Intel and Xilinx FPGAs?

Our contribution:

- Detailing our port of a subset of FPGA kernel optimizations from an Intel OpenCL to a Xilinx OpenCL specification
- Evaluating OpenCL kernel portability and performance from the ported hardware kernels
- Presenting our experience of using Xilinx Vitis Tools with OpenCL C kernels
- Contributing to the sparse literature of using OpenCL C for Xilinx platforms

How portable and performant are HLS designs between Intel and Xilinx FPGAs?

Our contribution:

- **Detailing our port of a subset of FPGA kernel optimizations from an Intel OpenCL to a Xilinx OpenCL specification**
- **Evaluating OpenCL kernel portability and performance from the ported hardware kernels**
- Presenting our experience of using Xilinx Vitis Tools with OpenCL C kernels
- Contributing to the sparse literature of using OpenCL C for Xilinx platforms

Our Porting Approach

- Isolate kernels to port
- Modifications to Host Code
- Porting Intel OpenCL FPGA Optimizations to the Xilinx Platform

Kernel Selection

- We use a subset of the Intel OpenCL FPGA implementations[†] of the Rodinia Benchmark Suite[*]
- We port two versions of each kernel: the *baseline* and *best* versions of each kernel

Ported Applications
Pathfinder
Computational Fluid Dynamics (CFD)
Speckle-reducing Anisotropic Diffusion (SRAD)
HotSpot

[†] Zohouri et al. "*Evaluating and Optimizing OpenCL Kernels for High Performance Computing with FPGAs*" SC '16
[*] Che et al., "*Rodinia: A Benchmark Suite for Heterogeneous Computing*" IISWC '09

Our Porting Approach

- Isolate kernels to port
- Modifications to Host Code
- Porting Intel FPGA OpenCL Optimizations to the Xilinx Platform

Host-Side Code

- The host-side code is responsible for setting for setting and managing the OpenCL runtime resources
- Not much structural difference between prior host code and our work
- We do attempt to better organize the code and make the code less error prone by using C++ features

Our Porting Approach

- Isolate kernels to port
- Modifications to Host Code
- Porting Intel FPGA OpenCL Optimizations to the Xilinx Platform

Porting the Baseline Kernels

- All baseline kernels are implemented using the Single Work Item (SWI) execution model
- The baseline kernel versions for each application do not include any FPGA optimizations

Porting the Best Kernels

- Porting Goal: perform the minimum amount of effort possible to port an optimization from Intel to Xilinx
- Optimization porting difficulty varies

Porting FPGA Optimizations: Loop Unrolling

Intel

```
#define U_FACTOR 8
#pragma unroll U_FACTOR


unsigned int i;
for (i = 0; i < N; ++i)
{
  //do some work
}
```

Xilinx

```
#define U_FACTOR 8
__attribute__((
  opencl_unroll_hint(U_FACTOR)
))

unsigned int i;
for (i = 0; i < N; ++i)
{
  //do some work
}
```

Porting FPGA Optimizations: Shift Register

Intel

```
int shift_reg[SR_SIZE];



int i, n;

for (n = 0; n < N; ++n)
{
  shift_reg[SR_SIZE-1] =
    input_arr[n];

  #pragma unroll SR_SIZE-1


  for (i = 0; i < SR_SIZE-1; ++i)
    shift_reg[i] = shift_reg[i+1];

}
```

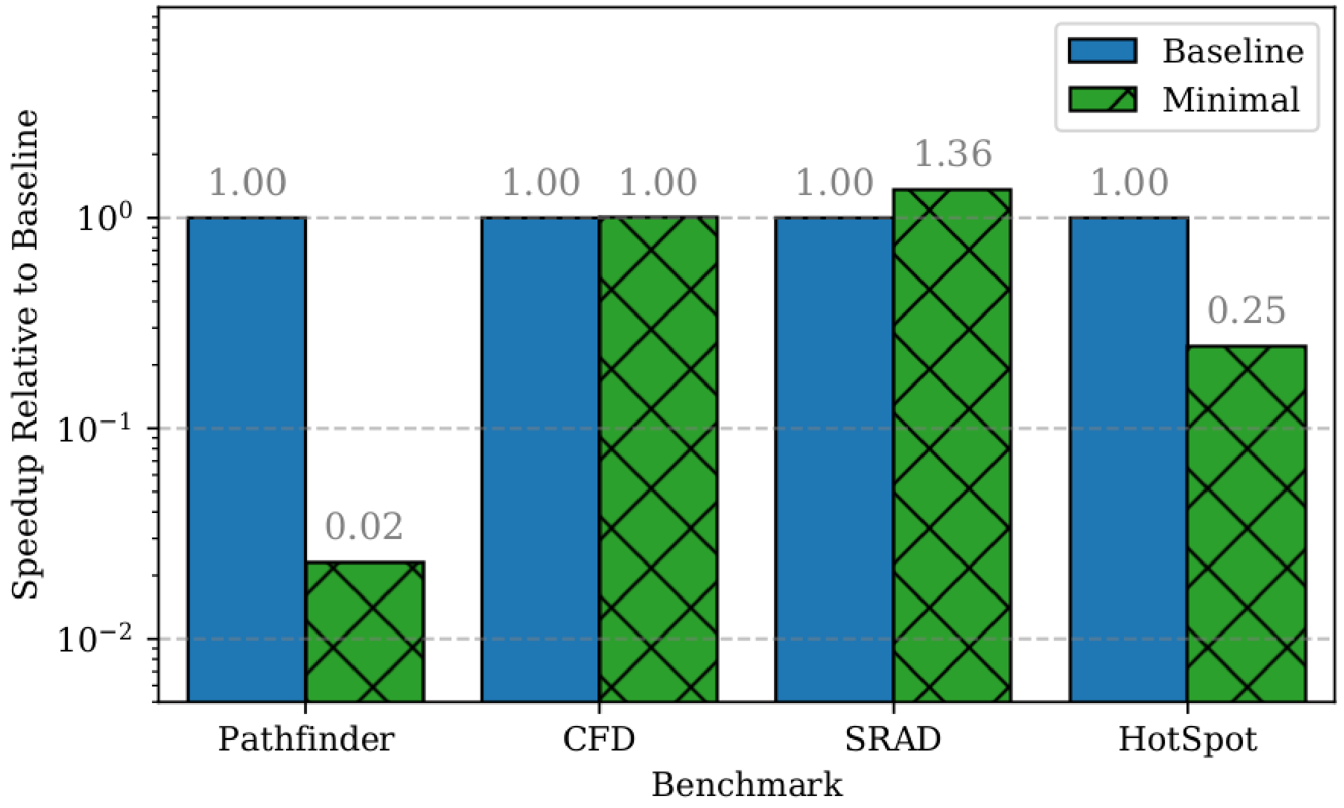
Xilinx

```
int shift_reg[SR_SIZE];
 __attribute__((
  xcl_array_partition(complete,0)
))
int i, n;

for (n = 0; n < N; ++n)
{
  shift_reg[SR_SIZE-1] =
    input_arr[n];

  __attribute__((
    opencl_unroll_hint(SR_SIZE-1)
  ))
  for (i = 0; i < SR_SIZE-1; ++i)
    shift_reg[i] = shift_reg[i+1];

}
```

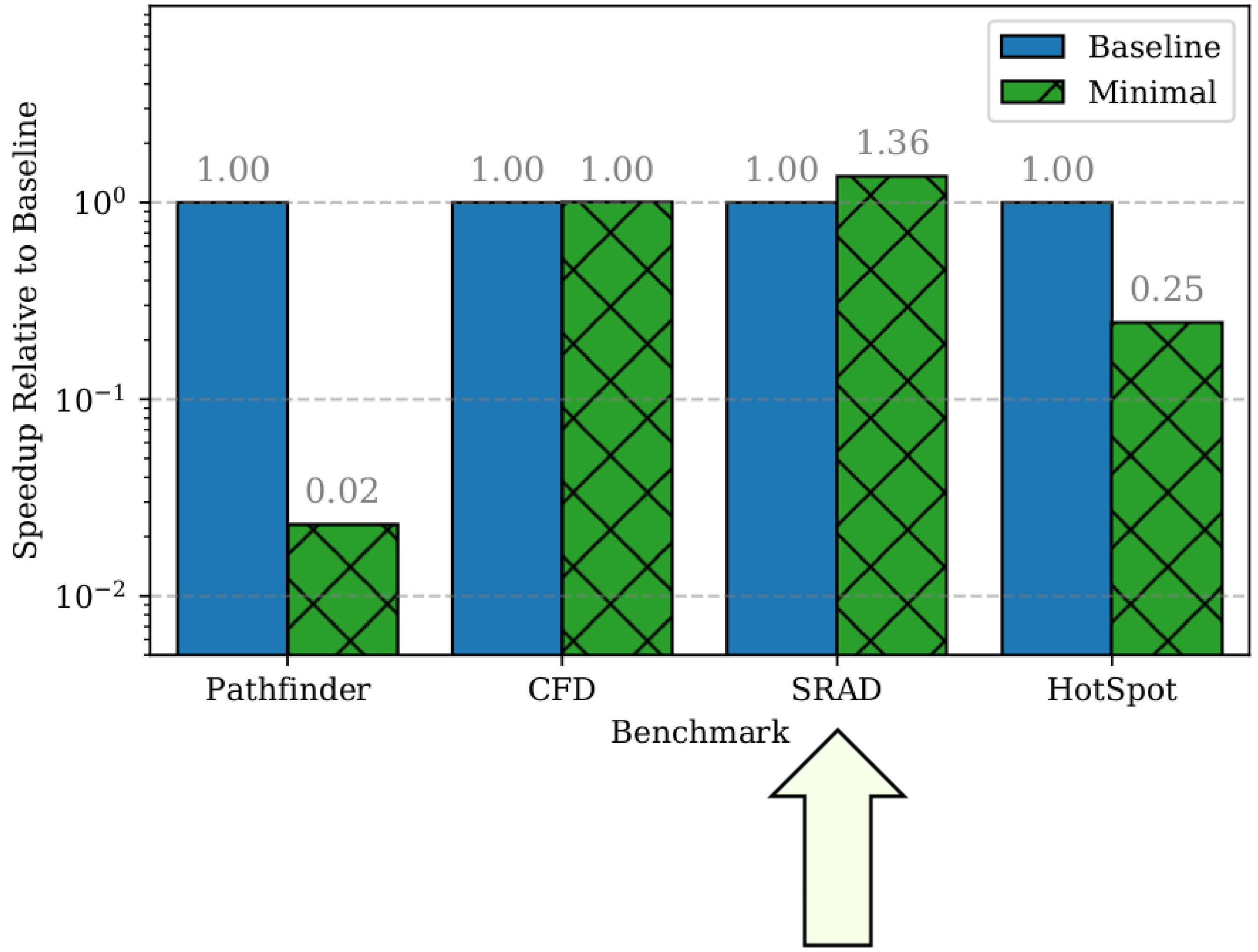
Evaluating Portability and Performance

- Results of the minimum effort ports
- Extracting more performance

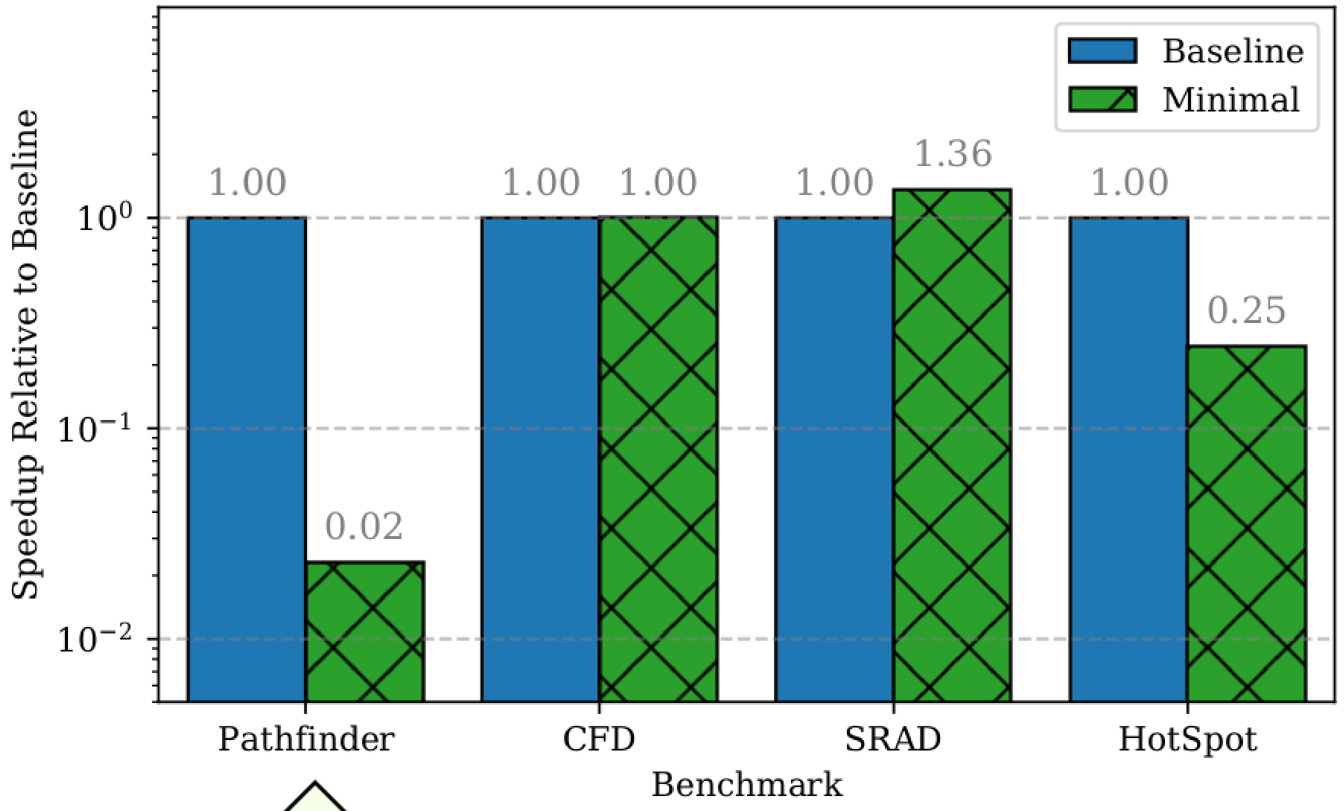
Minimum Effort Port Results



Minimum Effort Port Results



Minimum Effort Port Results

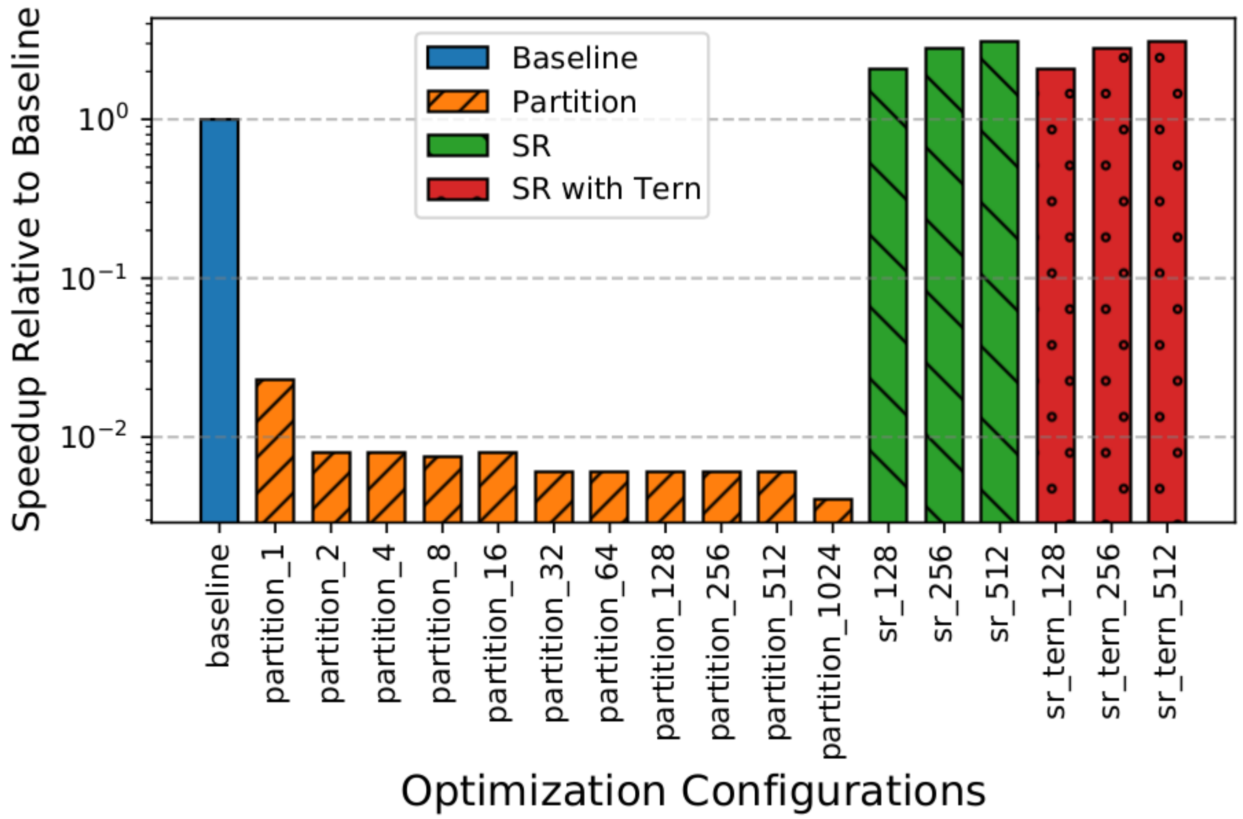


Evaluating Portability and Performance

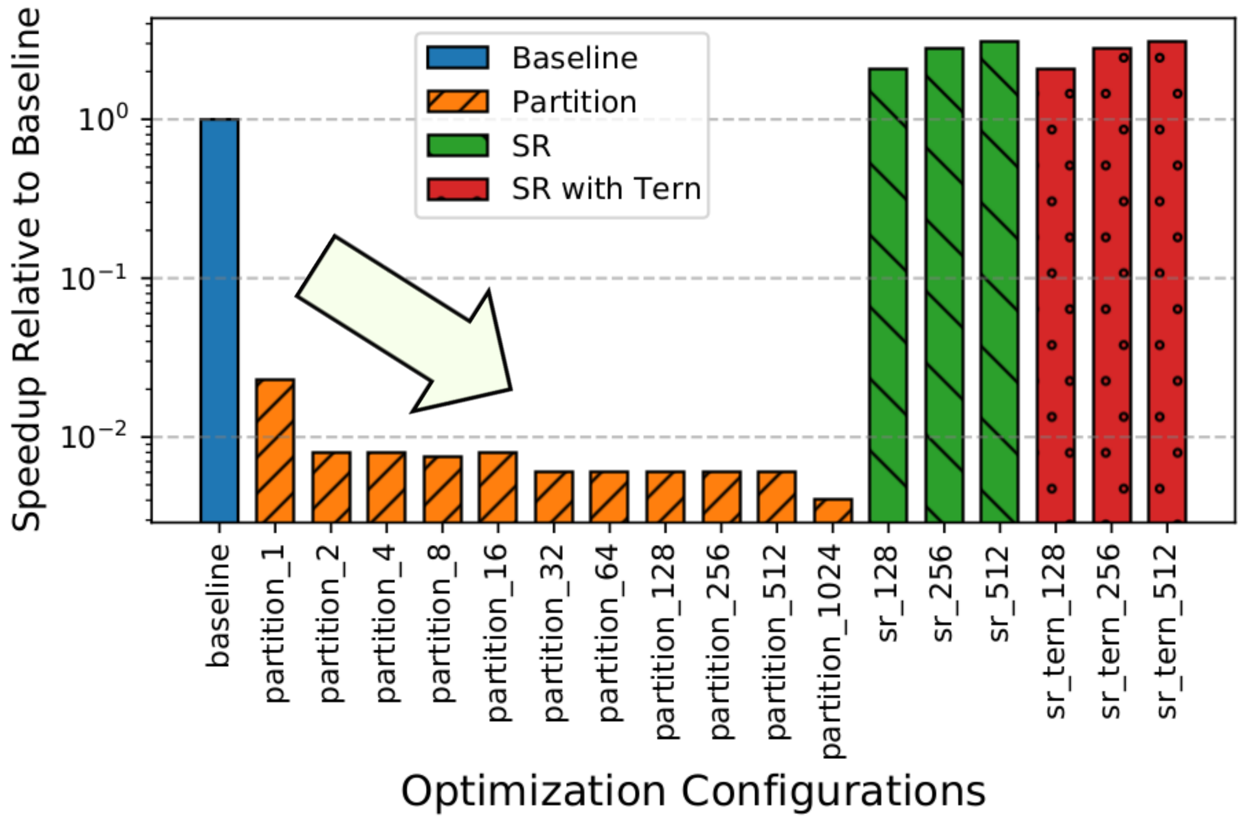
- Results of the minimum effort ports
- Extracting more performance

OAK RIDGE National Laboratory

Results of Optimization Exploration

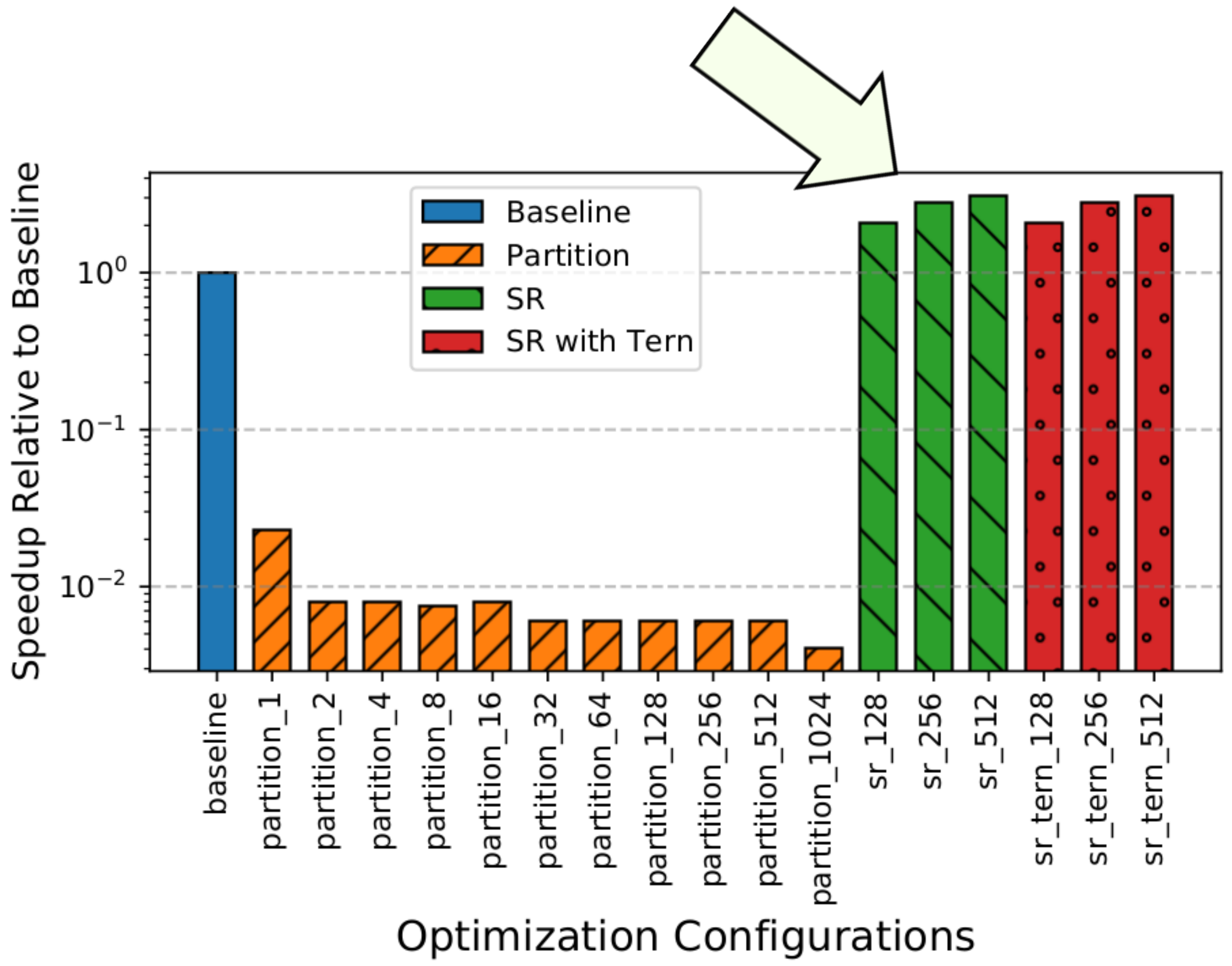


Results of Optimization Exploration



`PARTITION_ ∈ {1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024}`

Results of Optimization Exploration



SR_ ∈ {128, 256, 512}

Conclusion

- Initial effort toward evaluating portability and performance between Intel and Xilinx HLS kernels
- Ported Intel FPGA OpenCL implementations of the Rodinia Suite to a Xilinx perform this evaluation
- Varying degree of difficulty when porting optimizations
- Constructs that are known to perform well on an FPGA should perform well regardless of the platform, but may need non-trivial work to see good performance

Conclusion

- Initial effort toward evaluating portability and performance between Intel and Xilinx HLS kernels
- Ported Intel FPGA OpenCL implementations of the Rodinia Suite to a Xilinx perform this evaluation
- Varying degree of difficulty when porting optimizations
- Constructs that are known to perform well on an FPGA should perform well regardless of the platform, but may need non-trivial work to see good performance

Future Work

- Port more of the Rodinia applications to the Xilinx platform
- Explore the wider range of control afforded to the kernel designer by using C/C++ instead of OpenCL C
- Use lessons learned to automatically generate performant Xilinx HLS kernels

Contact: cabreraam AT ornl DOT gov