HLS Portability from Intel to Xilinx: A Case Study

Zhili Xiao & Roger Chamberlain, Washington University in St.Louis

Anthony Cabrera, Oak Ridge National Lab

Washington University in St.Louis

Kernel Selection: Needleman-Wunsch Algorithm

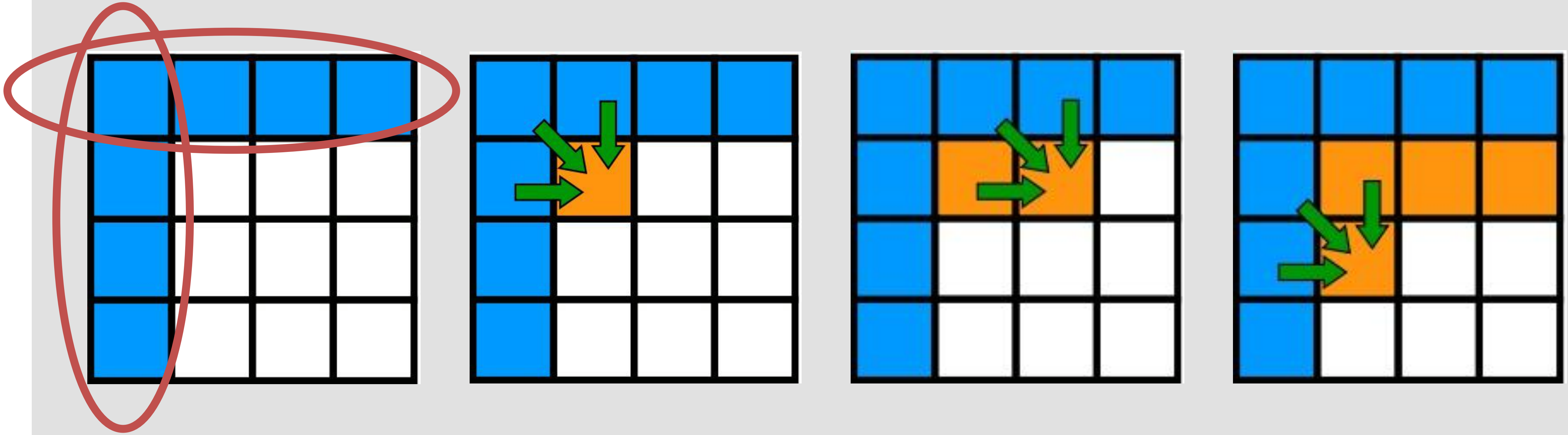
- From the Rodinia[1] benchmark suite
  - Originally for benchmarking OpenCL performance on multicore CPUs and GPUs
  - Extended for Intel FPGAs by Zohouri[2]

- A dynamic programming algorithm used in bioinformatics for a global alignment of protein or nucleotide

- We used Vitis 2020.1 development tool flow to port the baseline and the best versions of the NW kernels to Xilinx Alveo U250 Data Center accelerator card
  - an XCU250 FPGA of the Xilinx UltraScale+ architecture
  - a Gen3 x16 PCIe interface and 64 GB of DDR4 off-chip memory

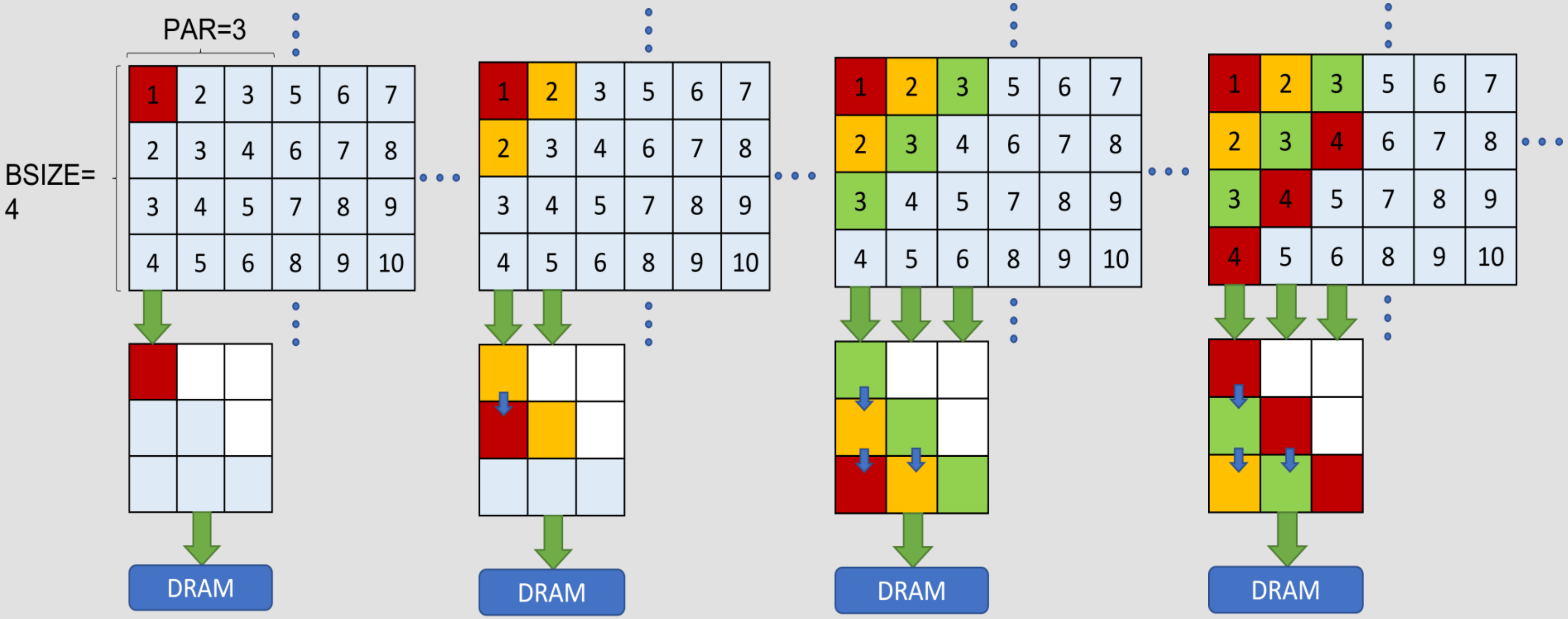
[1] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, “Rodinia: A benchmark suite for heterogeneous computing,” in Proc. of IISWC, 2009, pp. 44-54
[2] H. R. Zohouri, N. Maruyama, A. Smith, M. Matsuda, and S. Matsuoka, “Evaluating and Optimizing OpenCL Kernels for High Performance Computing with FPGAs,” in Proc. of SC. IEEE, 2016, pp. 409–420

Description of The Basic Kernel



Description of The Optimized Kernel

- Hardware design space the optimized kernel
  - BSIZE = {256,512,1024,2048,4096}, PAR = {8,16,32,64}



Porting the Baseline Kernel

- The baseline kernel does not involve any FPGA optimizations

```
inline void foo ( a, b, c, d ) {
  ...
}
```

```
//Xilinx OpenCL C
__attribute__((always_inline))
  void foo ( a, b, c, d ) {
  ...
}

//Xilinx C/C++
void foo ( a, b, c, d ) {
    #pragma HLS inline
  ...
}
```

Version	FPGA	Runtime(sec)
Baseline	Intel Stratix V GX A7, PCIe	204
	Intel Arria 10 GX 1150, HARP	830
	Xilinx Alveo u250, PCIe	**322**

Porting Optimizations for The Optimized Kernel: Direct Porting Efforts

- 1D shift registers
  - Complete partition of arrays to ensure shift register inference

Intel OpenCL C

```
int shift_reg[SR_SIZE];
int i;
//new input
shift_reg[SR_SIZE-1] = input;

//shift
#pragma unroll SR_SIZE - 1
for (i = 0; i < SR_SIZE - 1; ++i)
{
    shift_reg[i] = shift_reg[i+1];
}
```

Xilinx OpenCL C

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

int i;

//new input
shift_reg[SR_SIZE-1] = input;

//shift
__attribute__((opencl_unroll_hint(SR_SIZE-1)))
for (i = 0; i < SR_SIZE - 1; ++i)
{
    shift_reg[i] = shift_reg[i+1];
}
```

Porting Optimizations for The Optimized Kernel: Direct Porting Efforts

- 1D shift registers
  - Complete partition of arrays to ensure shift register inference

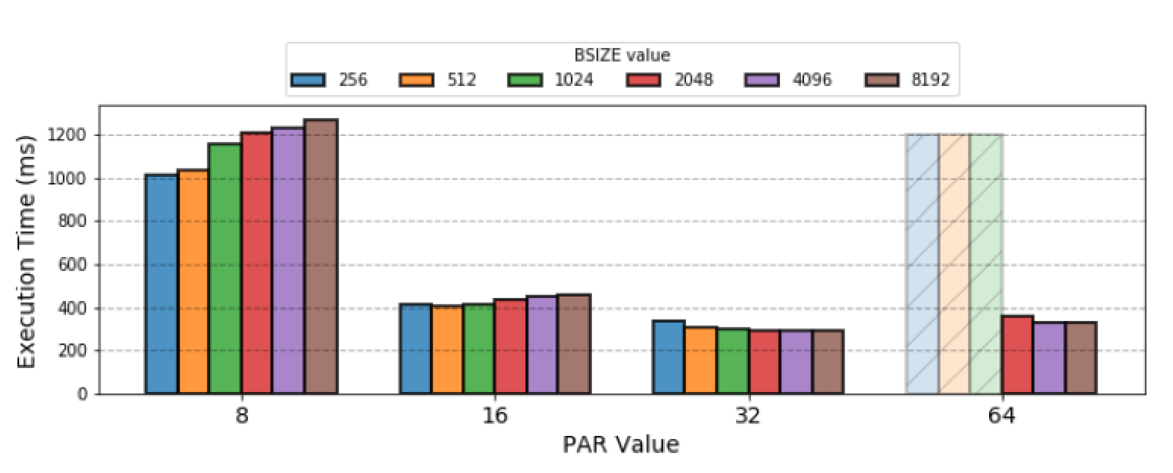
Intel OpenCL C

```
int shift_reg[SR_SIZE];
int i;
//new input
shift_reg[SR_SIZE-1] = input;

//shift
#pragma unroll SR_SIZE - 1
for (i = 0; i < SR_SIZE - 1; ++i)
{
    shift_reg[i] = shift_reg[i+1];
}
```

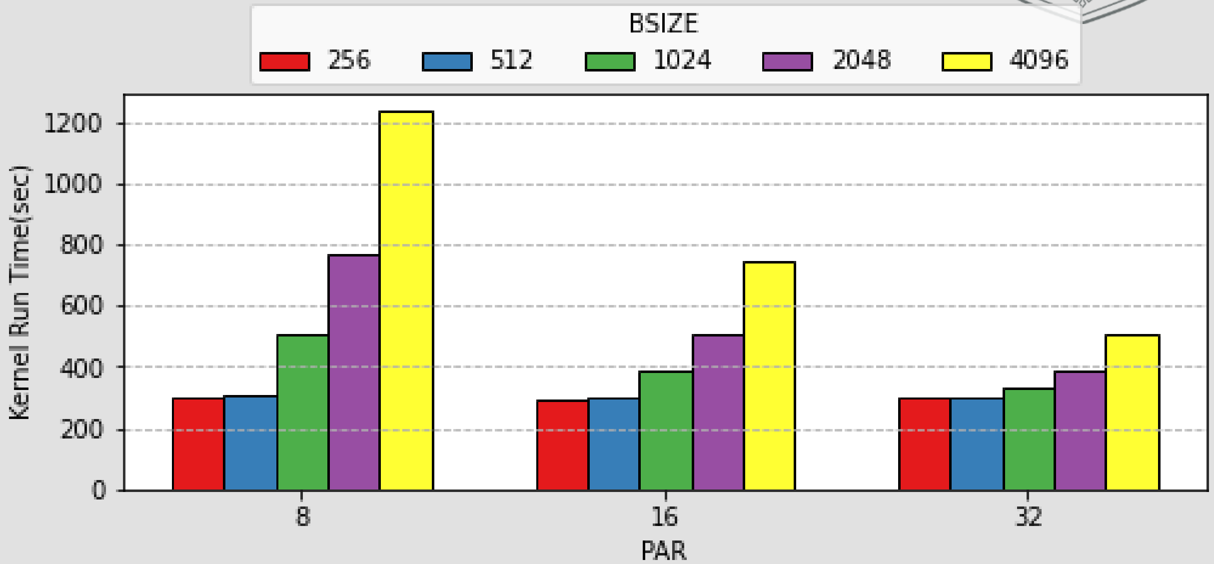
Xilinx C/C++

```
int shift_reg[SR_SIZE];
#pragma HLS ARRAY_PARTITION variable=shift_reg
complete

int i;
//new input
shift_reg[SR_SIZE-1] = input;

//shift
for (i = 0; i < SR_SIZE - 1; ++i)
{
    #pragma HLS unroll
    shift_reg[i] = shift_reg[i+1];
}
```

One-to-one Porting Results of the Best Kernel



A. M. Cabrera and R. D. Chamberlain, "Exploring Portability and Performance of OpenCL FPGA Kernels on Intel HARPv2," in Proc. of International Workshop on OpenCL. ACM, 2019.

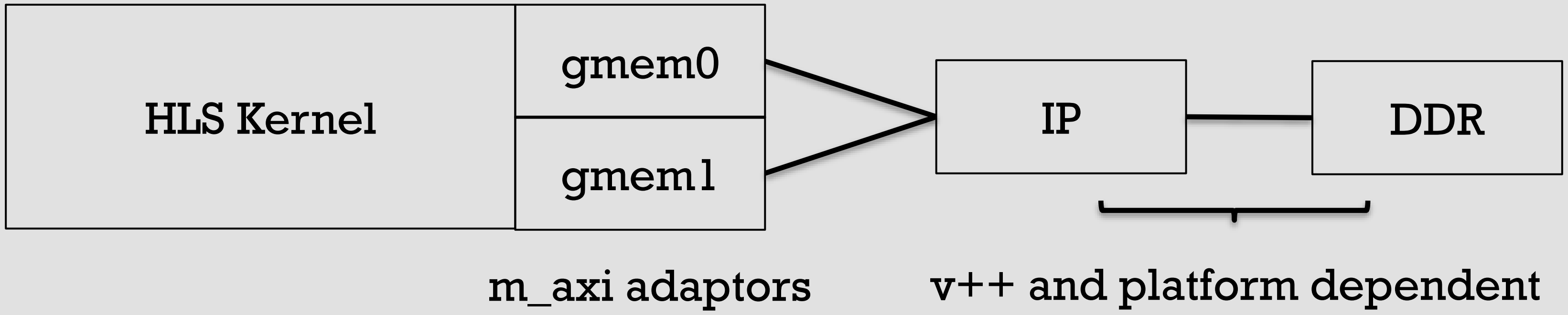


Our results.

Version	FPGA	Runtime(sec)
Best	Intel Stratix V GX A7, PCIe	0.26
	Intel Arria 10 GX 1150, HARP	0.29
	Xilinx Alveo u250, PCIe	294

Effective Porting Efforts

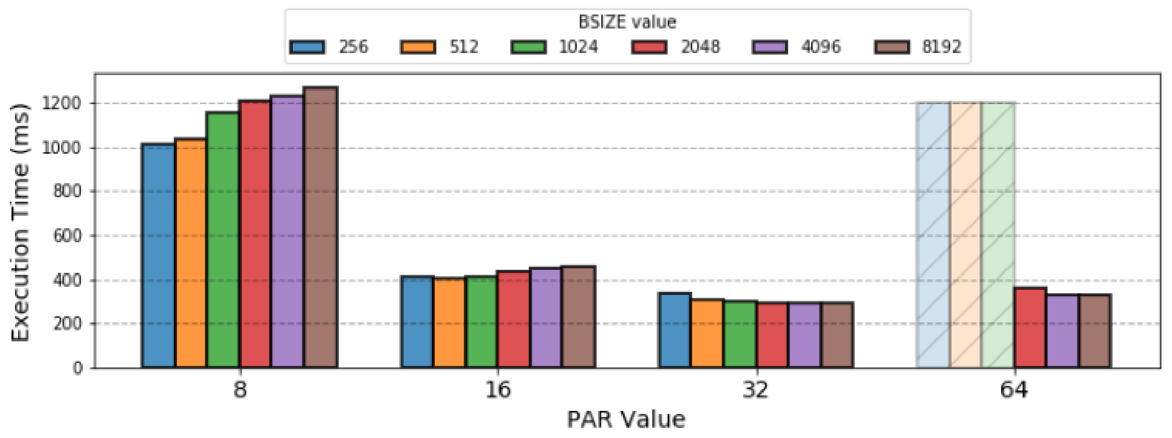
- Enable Burst Transfer
  - Isolated global memory access loops from other operations in the computation loop.
  - Changed `i--` to `i++` because one of the precondition for burst transfer in Xilinx is continuous monotonically increasing order
  - Loop unswitching technique to reduce pipeline initiation interval (II) to 1
- Pipeline Computation Loop
  - `#pragma HLS PIPELINE`
- Non-interleaving memory accesses by memory mapping and bundle assignment



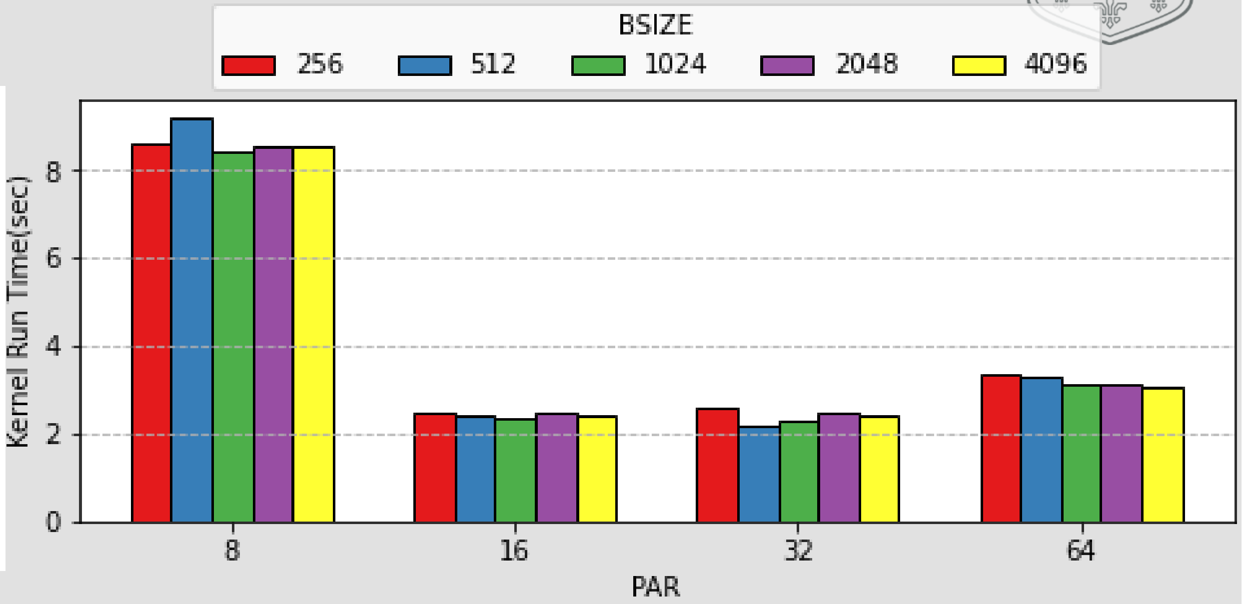
Efforts and their effects

Efforts	Effects
Enable burst transfer	Increase average transaction size. Reduce global memory access latency.
#pragma HLS PIPELINE	Reduce loop II.
Bundle option in HLS interface	Reduce the memory contention. Reduce loop II.
Memory banks mapping	Reduce memory stall time. Potentially increase the clock rate.

Performance Analysis



A. M. Cabrera and R. D. Chamberlain, "Exploring Portability and Performance of OpenCL FPGA Kernels on Intel HARPv2," in Proc. of International Workshop on OpenCL. ACM, 2019.



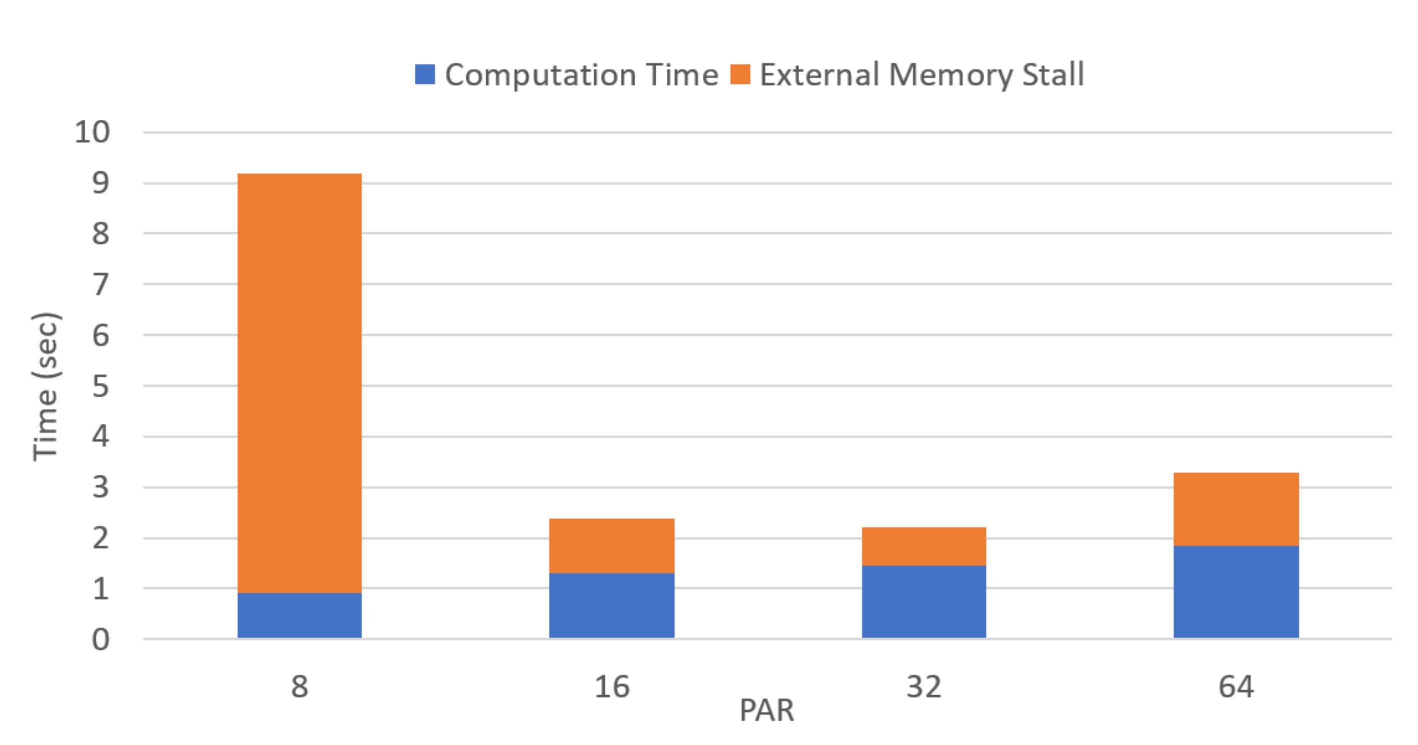
Our results.

Version	FPGA	Runtime(sec)
Best	Intel Stratix V GX A7, PCIe	0.26
	Intel Arria 10 GX 1150, HARP	0.29
	Xilinx Alveo u250, PCIe	2.2

Performance Analysis

- II always equals to PAR

- Tradeoff between parallelism, clock rates and memory access latency



Conclusion

- One-to-one optimization porting is not sufficient.
- Xilinx has many canonical rules and rigorous constraints.
- Trade off between the ability of more fine-grained control and simplicity in coding and expressions.
- Noticeable differences between the Intel and Xilinx tool flows

Further Work

- Focus on closing the performance gap.
- New functionalities in Vitis 2020.2.
  - Automatic port width widening