Design and Performance Evaluation of Optimizations for OpenCL FPGA Kernels

Anthony M. Cabrera[*†] (he/him) and Roger D. Chamberlain[*]
[*]McKelvey School of Engineering, Washington University in St. Louis
[†]Future Technologies Group, Oak Ridge National Laboratory

September 24, 2020
Waltham, MA, USA
HPEC ‘20

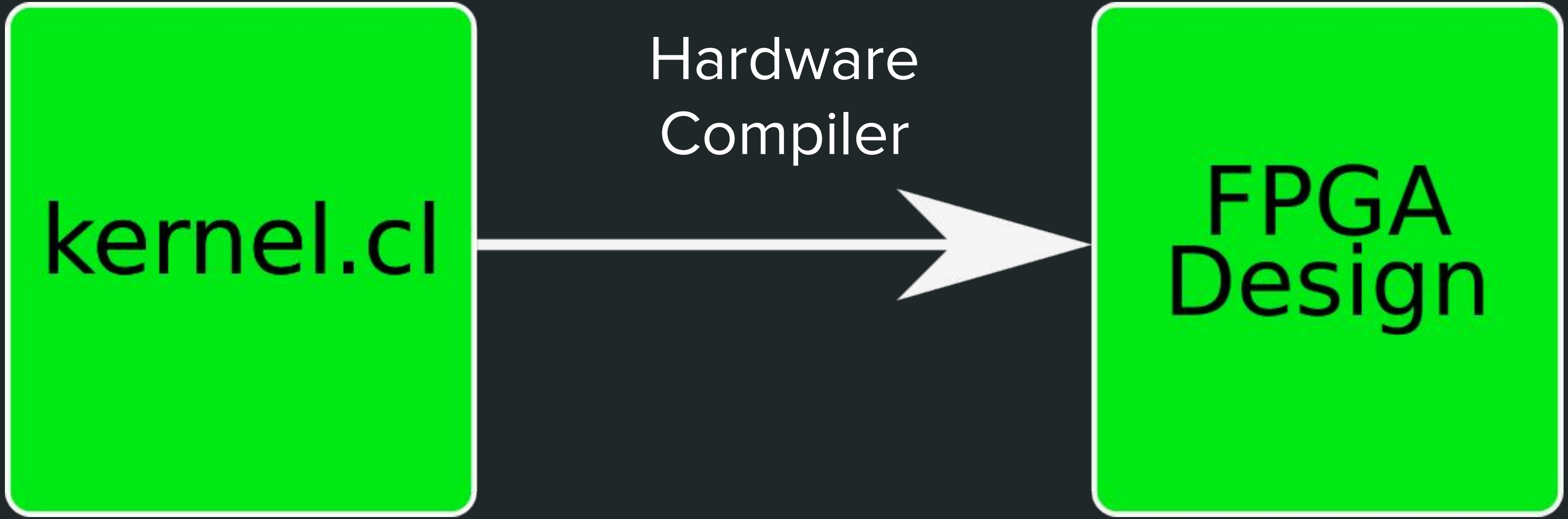
FPGAs Gaining Traction

Bloomberg

Deals

Intel's $16.7 Billion Altera Deal Is Fueled by Data Centers

OpenCL to the Rescue!



Our Contribution

OpenCL FPGA design methods for:

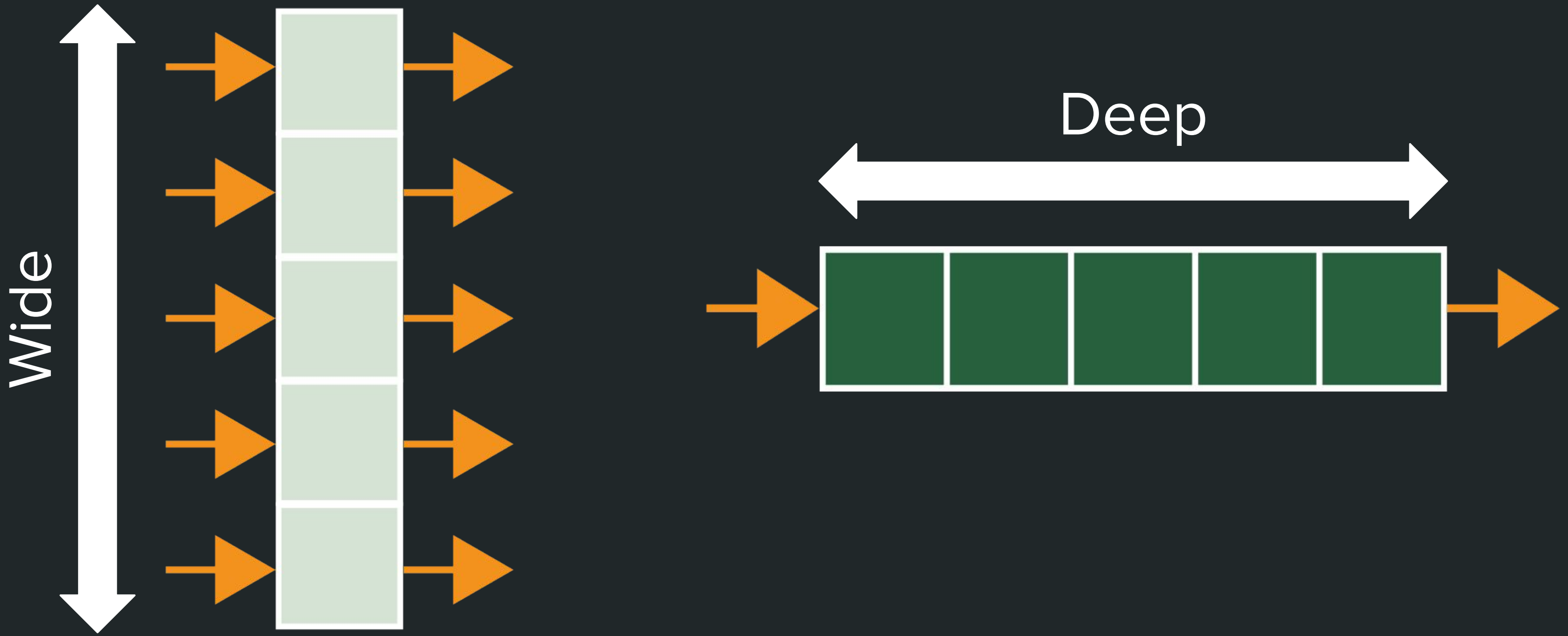
1) execution model selection

2) CDFGs to inform design choices

3) building on top of the best execution model

Width vs. Depth

The two OpenCL FPGA Design Paradigms



Case Study: `ebcdic_txt` Wide Kernel

3) Width Knobs

```
__kernel void e2a(
```

2) Kernel Arguments

```
{
```

1) Kernel Body

```
}
```

Case Study: `ebcdic_txt` Wide Kernel

3) Width Knobs

2) Kernel Arguments

```
__kernel void e2a(

{
    unsigned char e2a_lut[256] = { … };
    unsigned int i = get_global_id(0);
    uchar orig_char = src[i];
    uchar xformd_char;
    xformd_char = e2a_lut[orig_char];
    dst[i] = xformd_char;
}
```

Case Study: `ebcdic_txt` Wide Kernel

3) Width Knobs

```
__kernel void e2a(  __global const uchar* restrict src,
                    __global uchar* restrict dst)
{
    unsigned char e2a_lut[256] = { … };
    unsigned int i = get_global_id(0);
    uchar orig_char = src[i];
    uchar xformd_char;
    xformd_char = e2a_lut[orig_char];
    dst[i] = xformd_char;
}
```

Case Study: ebcdic_txt Wide Kernel

3) Width Knobs

```
__kernel void e2a(  __global const uchar* restrict src,
                    __global uchar* restrict dst)
{
    unsigned char e2a_lut[256] = { … };
    unsigned int i = get_global_id(0);
    uchar orig_char = src[i];
    uchar xformd_char;
    xformd_char = e2a_lut[orig_char];
    dst[i] = xformd_char;
}
```

Case Study: `ebcdic_txt` Wide Kernel

3) Width Knobs

```
__kernel void e2a(  __global const uchar* restrict src,
                    __global uchar* restrict dst)
{
    unsigned char e2a_lut[256] = { … };
    unsigned int i = get_global_id(0);
    uchar orig_char = src[i];
    uchar xformd_char;
    xformd_char = e2a_lut[orig_char];
    dst[i] = xformd_char;
}
```

Case Study: `ebcdic_txt` Wide Kernel

3) Width Knobs

```
__kernel void e2a(  __global const uchar* restrict src,
                    __global uchar* restrict dst)
{
    unsigned char e2a_lut[256] = { … };
    unsigned int i = get_global_id(0);
    uchar orig_char = src[i];
    uchar xformd_char;
    xformd_char = e2a_lut[orig_char];
    dst[i] = xformd_char;
}
```

What About the “Loose Ends”?

```
__kernel void e2a(    __global const uchar* restrict src,
                      __global uchar* restrict dst)
{
    unsigned char e2a_lut[256] = { … };
    unsigned int i = get_global_id(0);
    uchar orig_char = src[i];
    uchar xformd_char;
    xformd_char = e2a_lut[orig_char];
    dst[i] = xformd_char;
}
```

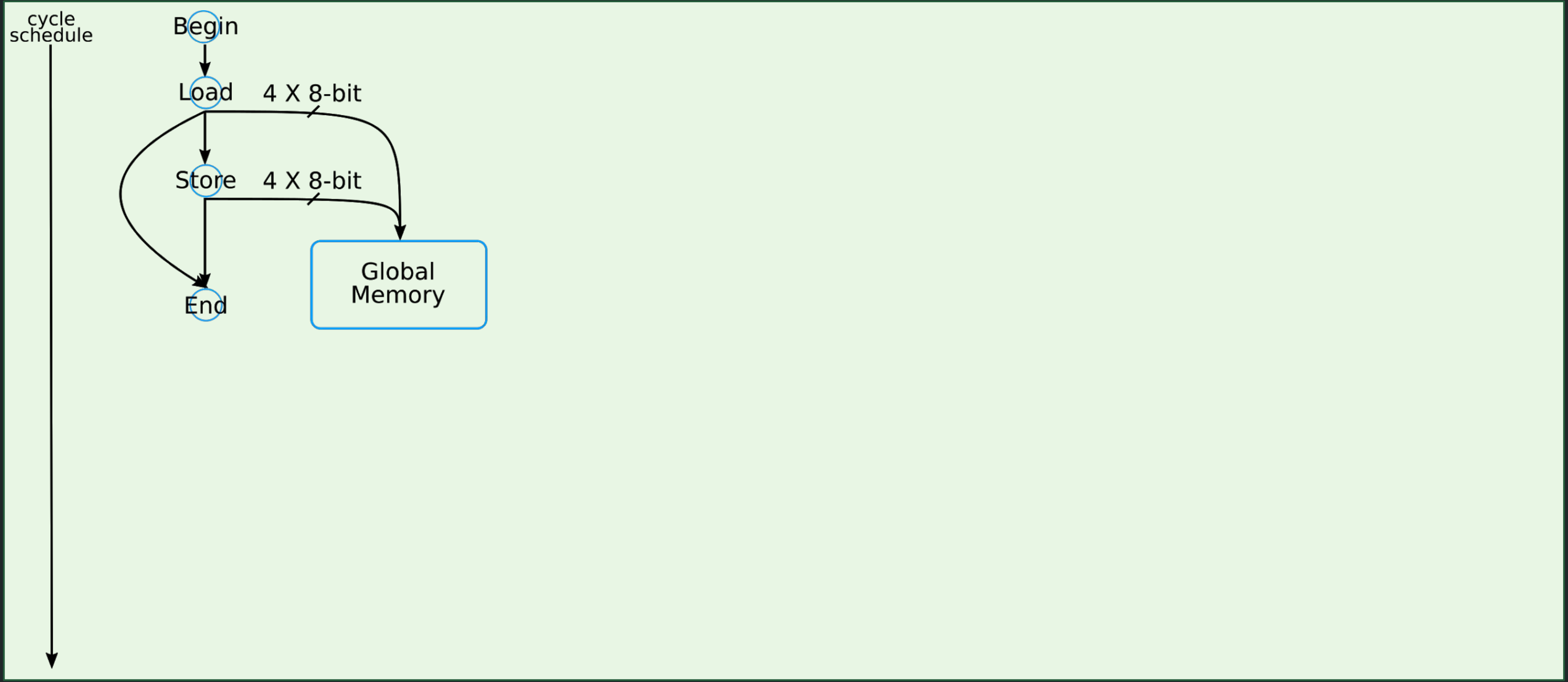
What About the “Loose Ends”?

```
__kernel void e2a(    __global const uchar* restrict src,
                      __global uchar* restrict dst,
                      ulong total_work_items)
{
    unsigned char e2a_lut[256] = { … };

    if (i < total_work_items) {
        unsigned int i = get_global_id(0);
        uchar orig_char = src[i];
        uchar xformd_char;
        xformd_char = e2a_lut[orig_char];
        dst[i] = xformd_char;
    }
}
```

Unbounded (left) vs. Bounded (right)

4 replicates

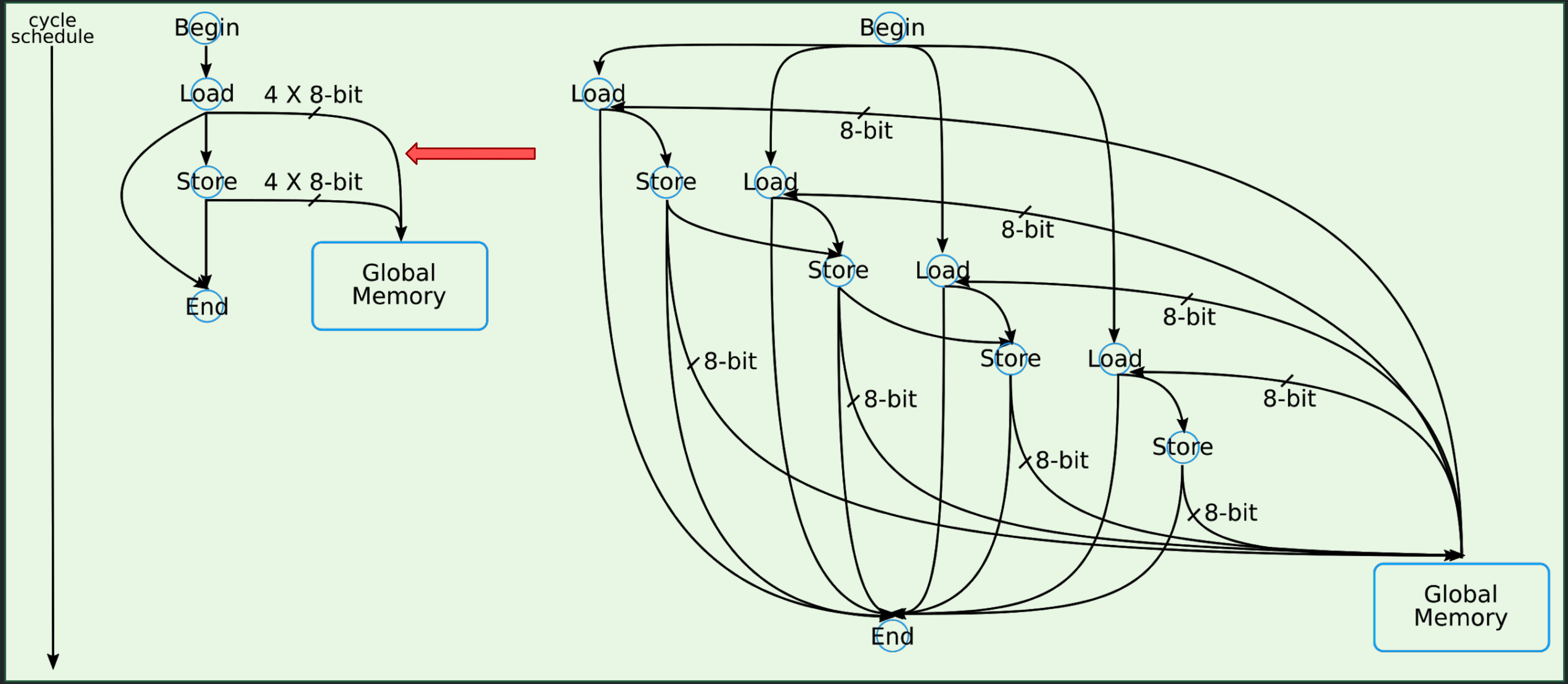


Unbounded (left) vs. Bounded (right)

4 replicates

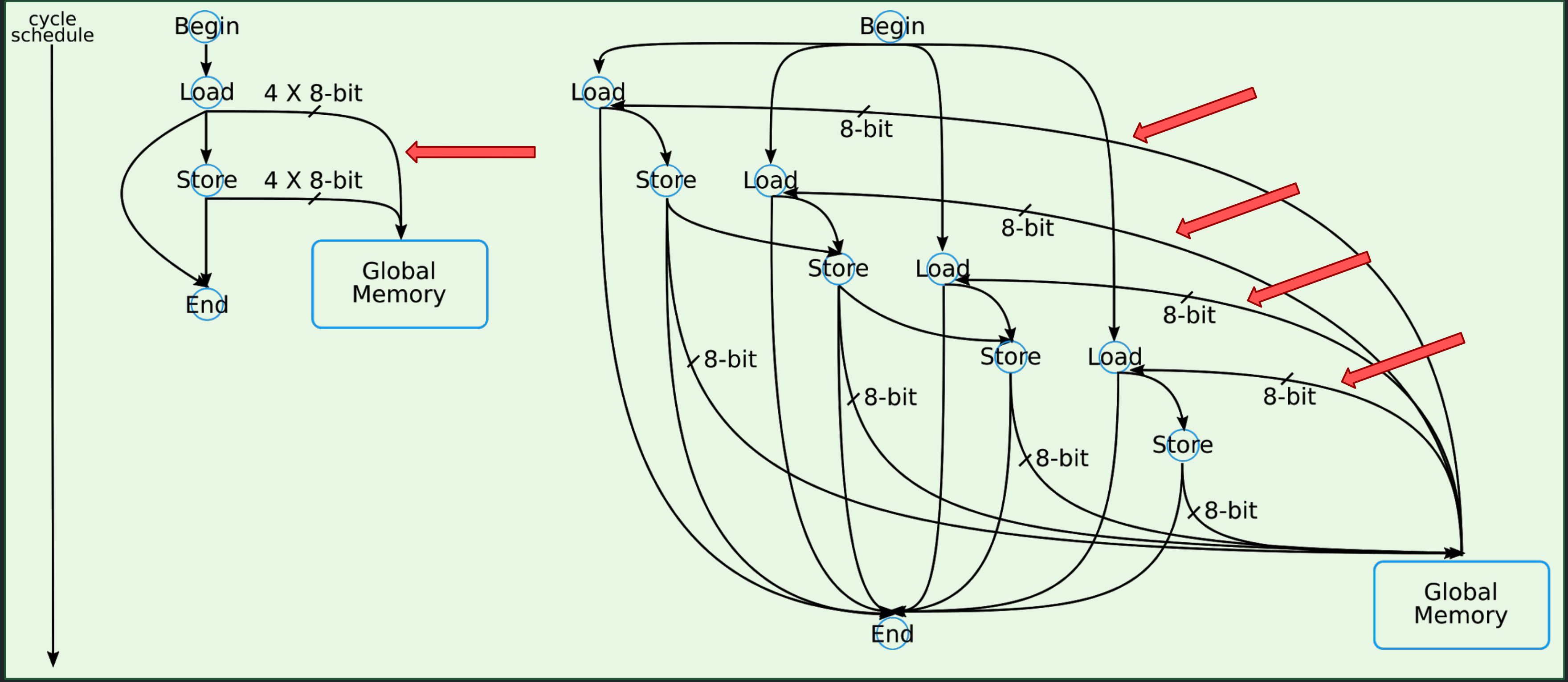
Unbounded (left) vs. Bounded (right)

4 replicates



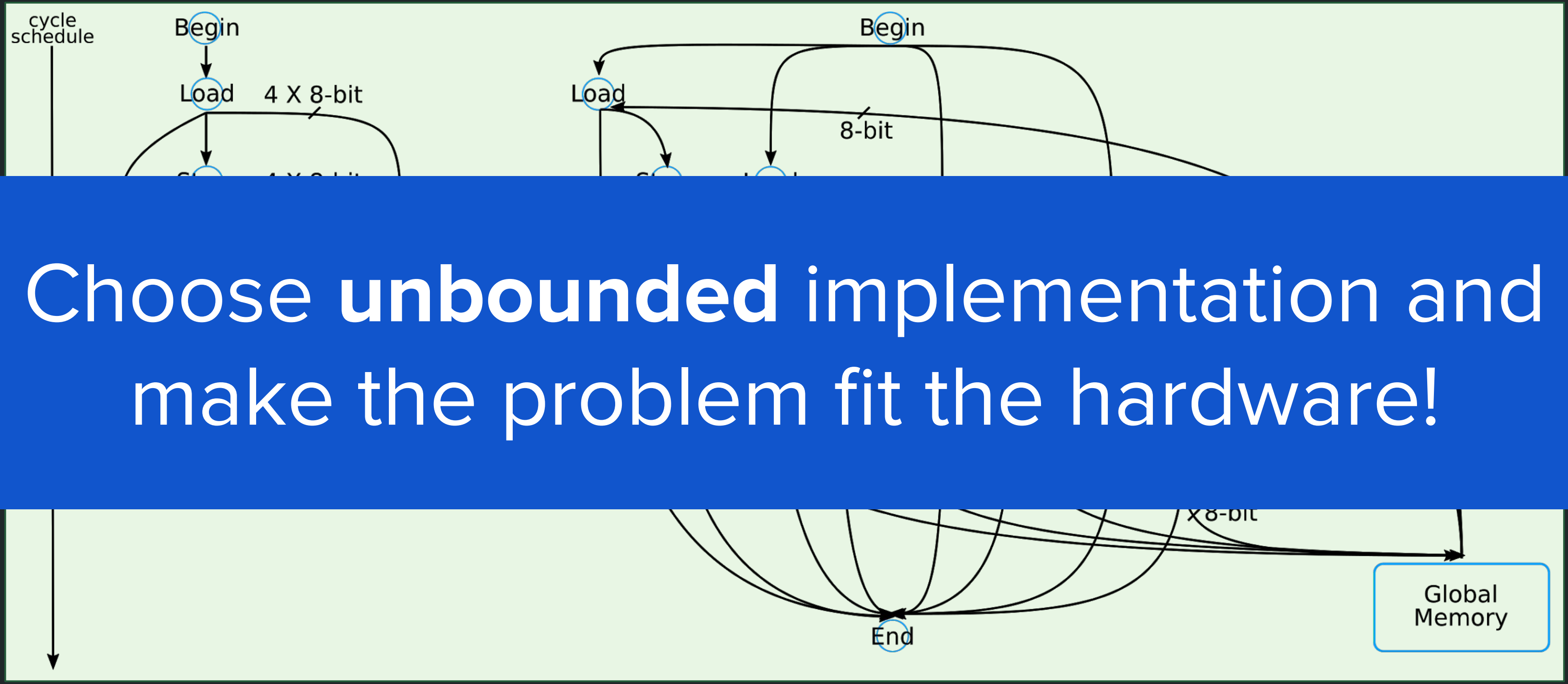
Unbounded (left) vs. Bounded (right)

4 replicates



Unbounded (left) vs. Bounded (right)

4 replicates



Choose unbounded implementation and make the problem fit the hardware!

Case Study: `ebcdic_txt` Wide Kernel

3) Width Knobs

```
__kernel void e2a(  __global const uchar* restrict src,
                    __global uchar* restrict dst)
{
    unsigned char e2a_lut[256] = { … };
    unsigned int i = get_global_id(0);
    uchar orig_char = src[i];
    uchar xformd_char;
    xformd_char = e2a_lut[orig_char];
    dst[i] = xformd_char;
}
```

`ebcdic_txt` Coarse-Grain Width Knobs

```
__attribute__((num_compute_units(NUMCOMPUNITS)))
```

NUMCOMPUNITS = # of replicated compute units

NUMCOMPUNITS ∈ {1,2,4,8}

`ebcdic_txt` Coarse-Grain Width Knobs

```
__attribute__((num_compute_units(NUMCOMPUNITS)))
__attribute__((reqd_work_group_size(WGSIZE,1,1)))
```

NUMCOMPUNITS = # of replicated compute units

WGSIZE = work-group size of compute unit

`NUMCOMPUNITS` ∈ {1,2,4,8}

`WGSIZE` ∈ {128, 256, 512, 1024}

`ebcdic_txt` Coarse-Grain Width Knobs

```
__attribute__((num_compute_units(NUMCOMPUNITS)))
__attribute__((reqd_work_group_size(WGSIZE,1,1)))
__attribute__((num_simd_work_items(NUMSIMD)))
```

NUMCOMPUNITS = # of replicated compute units

WGSIZE = work-group size of compute unit

`NUMCOMPUNITS` ∈ {1,2,4,8}

`WGSIZE` ∈ {128, 256, 512, 1024}

NUMSIMD = # of times data path is replicated within a compute unit

`NUMSIMD` ∈ {1, 2, 4, 8, 16}

Case Study: `ebcdic_txt` Deep Kernel

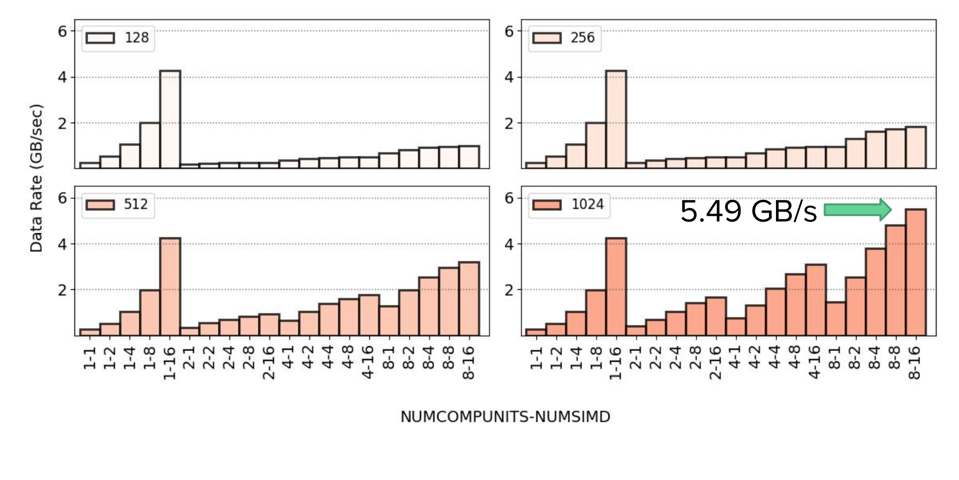
```
__kernel void e2a(  __global const uchar* restrict src,
                    __global uchar* restrict dst,
                    ulong num_elts)
{
    unsigned char e2a_lut[256] = { … };
    unsigned int i;
    #pragma unroll UNROLL
    for (i = 0; i < num_elts; ++i) {
        uchar xformd_char;
        xformd_char = e2a_lut[orig_char];
        dst[i] = xformd_char;
    }
}
```

Loop termination condition

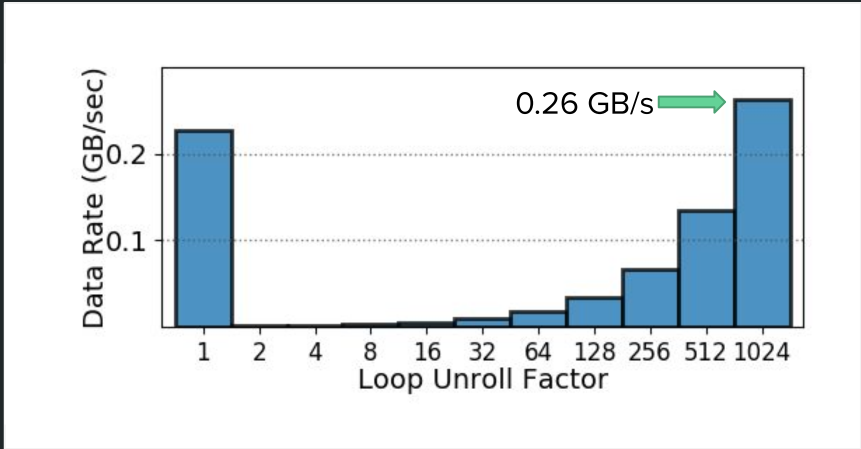
UNROLL = # of times to unroll the loop

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

`ebcdic_txt` Width vs. Depth Results



Wide Result



Deep Result

Widening the Data Type

$N = \{ 2, 4, 8, 16 \}$

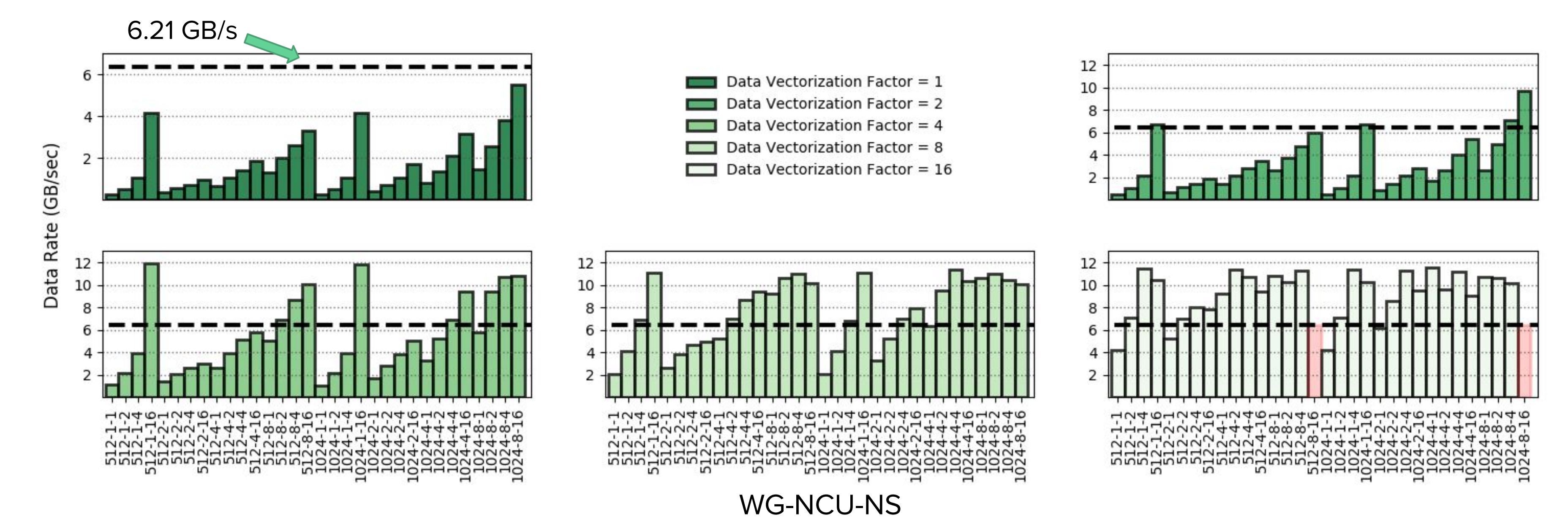
```
__attribute__(...)
__kernel void e2a(  __global const uchar* restrict src,
                    __global uchar* restrict dst)
{
    unsigned char e2a_lut[256] = { … };
    unsigned int i = get_global_id(0);
    uchar orig_char = src[i];
    uchar xformd_char;
    xformd_char = e2a_lut[orig_char];
    dst[i] = xformd_char;
}
```

Widening the Data Type

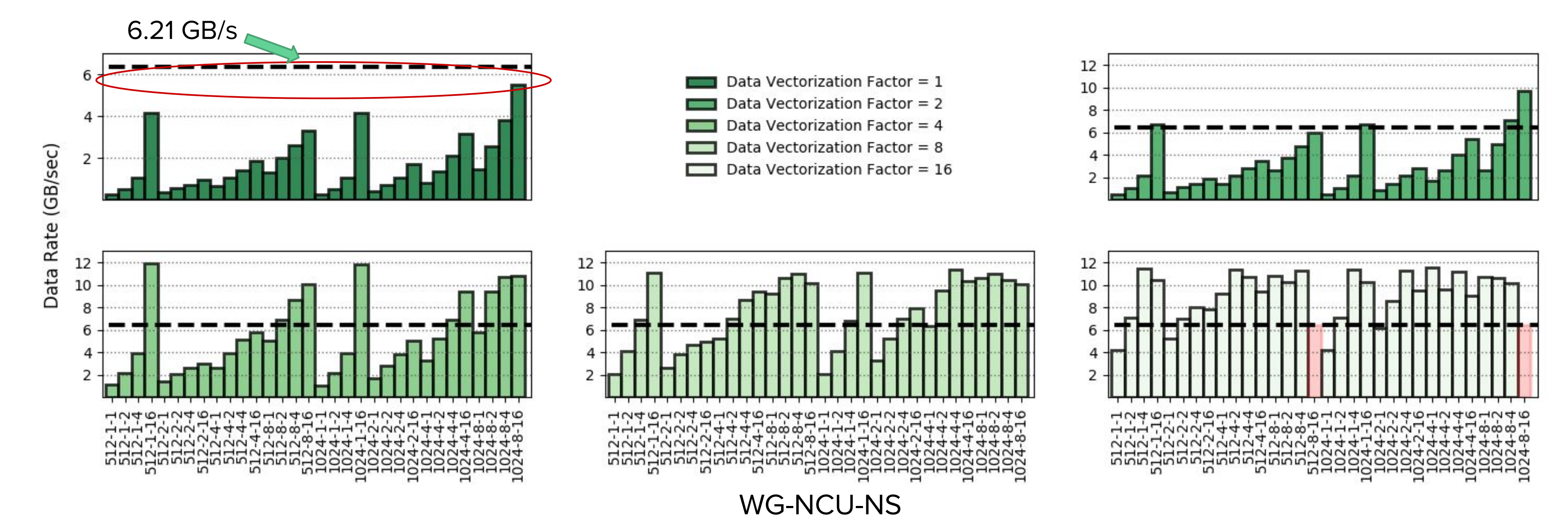
N = { 2, 4, 8, 16 }

```
__attribute__(...)
__kernel void e2a(  __global const ucharN* restrict src,
                    __global ucharN* restrict dst)
{
    unsigned char e2a_lut[256] = { … };
    unsigned int i = get_global_id(0);
    ucharN orig_char = src[i];
    ucharN xformd_char;
    xformd_char.s0 = e2a_lut[orig_char.s0];
    ...
    xformd_char.sN = e2a_lut[orig_char.sN];
    dst[i] = xformd_char;
}
```

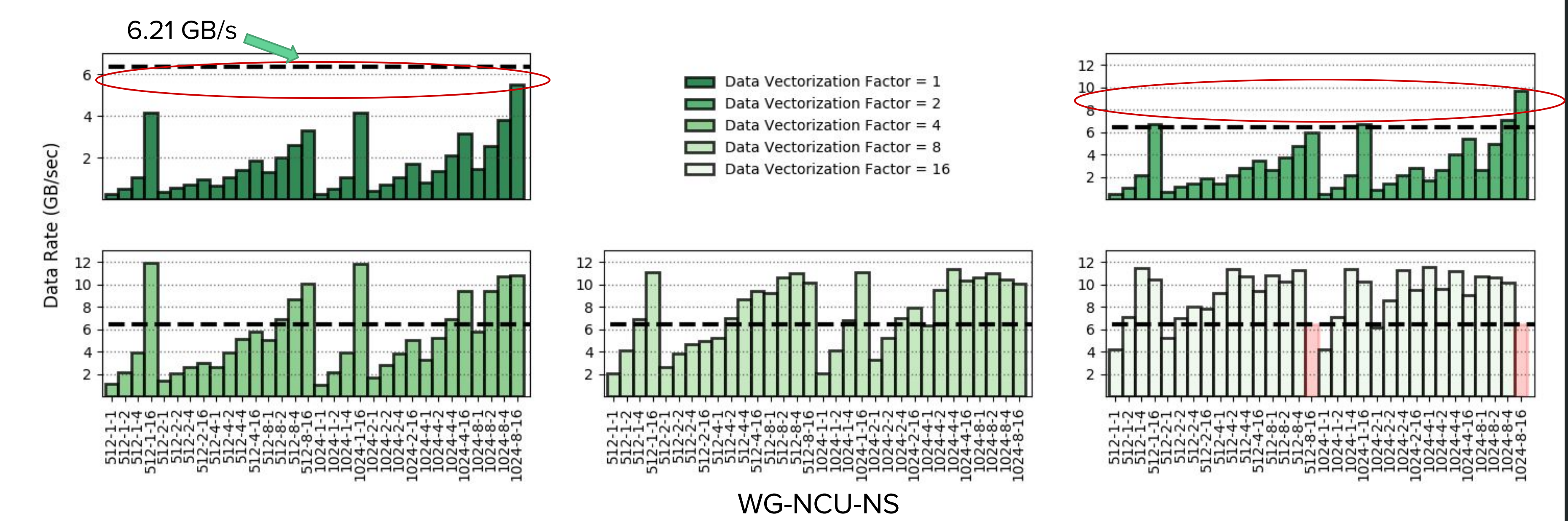
Widening the Data Type Results



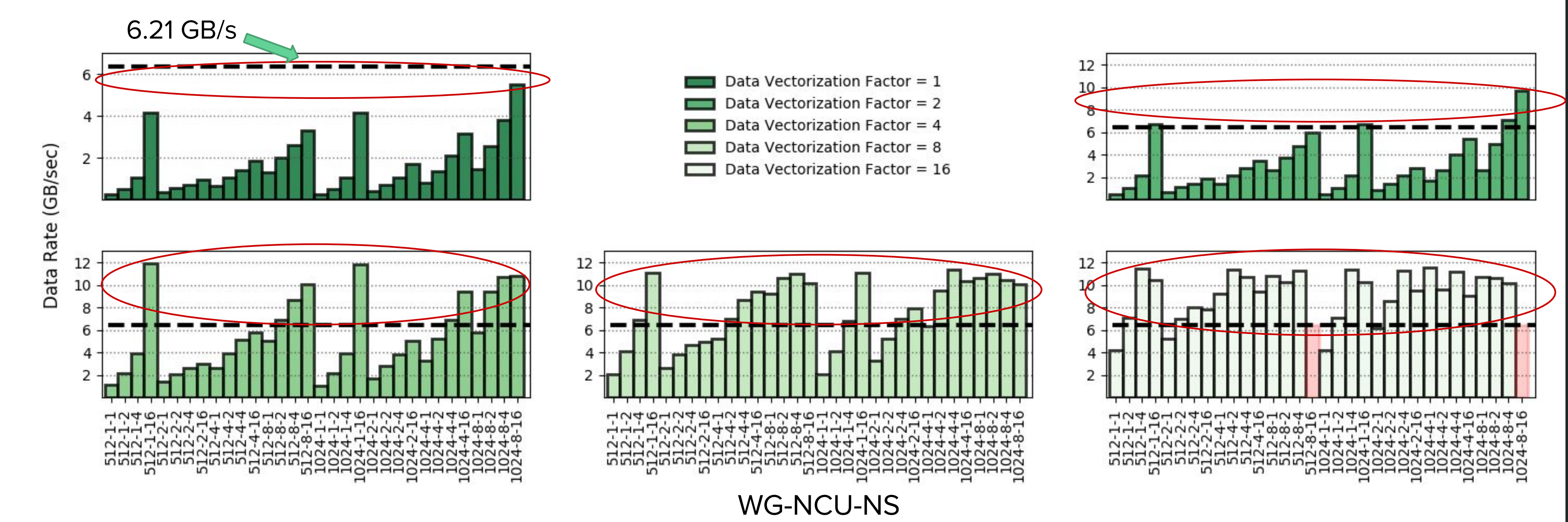
Widening the Data Type Results



Widening the Data Type Results



Widening the Data Type Results



Conclusion

We present OpenCL FPGA design methods for:

1) selecting a “wide” or “deep” execution model

2) informing design choices using CDFGs

3) evaluating additional knob interactions with best execution model

Future Work

More complex applications

Tuning decisions made by the tool-chain

We present OpenCL FPGA design methods for:

1) selecting a “wide” or “deep” execution model

2) informing design choices using CDFGs

3) evaluating additional knob interactions with best execution model

Contact Info

cabreraam AT ornl DOT gov