“Using OpenCL to Increase SCA Application Portability”

NordiaSoft: Steve Bernier, François Lévesque, Martin Phisel

Cobham: David Hagood

About this Paper

- **Techniques for Efficient Software Porting Between Heterogeneous Platforms**
  - The #1 innovation on the top 10 list of most wanted innovations compiled by the WInnF

- **This paper describes how OpenCL™ can be used to create more portable SCA Waveform Applications**

OpenCL

What are the problems to be solved?

- GPPs generally programmed in C++ or C
- DSPs generally programmed in C or limited C++
- Code often modified to get best performance
  - Extensions for accessing special instructions (SIMD [e.g. SSE, AVX, 3DNow!, Altivec, NEON], VLIW)
- Thus not portable between different CPU or DSP families

What are the problems to be solved?

- **FPGAs generally programmed in VHDL**
  - Different FPGA architectures require changes in code
  - VHDL directly supports notions of parallelism
- **System C allows “C like” semantics, but not same as CPU or DSP C**

The desired solution looks like….

- Holy grail would be one language that can target GPPs, DSPs, and FPGA
- Language should directly support concepts of parallelism and vector processing
- Could also support GPUs, as they are increasingly available on low cost SoCs
- We just described OpenCL™
  - See: www.khronos.org/opencl/

What is OpenCL (1)

- Language initially created to access shader engines in GPUs for general computation
- “C like” syntax, with direct support for vectorization
- Just in time compiler(s) to target different compute devices

What is OpenCL (2)

- **Libraries to manage non-uniform memory models**
  - GPU memory is fast and wide vs. GPP
  - Just like DSPs or FPGAs
- **Now being supported by large number of vendors:**

Running an OpenCL kernel (1)

- **Write C code to select an OpenCL device from the library**
- **Create a "context" within that device**
  - Devices can support many contexts to isolate apps
- **Load code for the kernel(s) into context**
  - Usually, code gets compiled at this time
  - Can also be precompiled to Standard Portable Intermediate Representation (SPIR)
  - Or can be precompiled for target

Running an OpenCL kernel (2)

- **How long does it take to create an OpenCL kernel?**

	Small		Large	
Format in kernel file	**CPU**	**GPU**	**CPU**	**GPU**
Source code	13149	391	142089	447
Native binary	968	378	4381	396
Binary in SPIR	923	--	4187	--

Small kernel is 16 LOC, Large kernel is 398 LOC, Figures in microseconds

Running an OpenCL kernel (3)

- **Define memory buffers**
  - Can be on target e.g. GPU
- **For each block of data:**
  - Move data from host to OpenCL buffer(s)
  - Run kernel(s)
    - One kernel's output can be another kernel's input
  - Move data from OpenCL buffer to host

Running an OpenCL kernel (4)

- How long does it take to move data?

Buffer size (KB)	CPU		GPU	
	H2D (µs)	D2H (µs)	H2D (µs)	D2H (µs)
4	5	9	10	12
32	7	12	19	19
320	32	42	101	104
640	67	75	191	312
960	112	105	406	464
1280	155	153	468	614
1600	193	161	520	694
1920	247	186	577	814
2240	274	209	653	903
2560	333	234	706	1020
2880	608	296	746	1194
3200	694	372	794	1307

H2D : Host to OpenCL Device
D2H : OpenCL Device to Host

Running an OpenCL kernel (5)


```
// Multiply a vector by a scalar
__kernel void VectorGain(__global const float *inputVector,
                         float const gain,
                         __global float *restrict outputVector )
{
    int index = get_global_id(0);
    outputVector[index] = gain * inputVector[ index ];
}
```

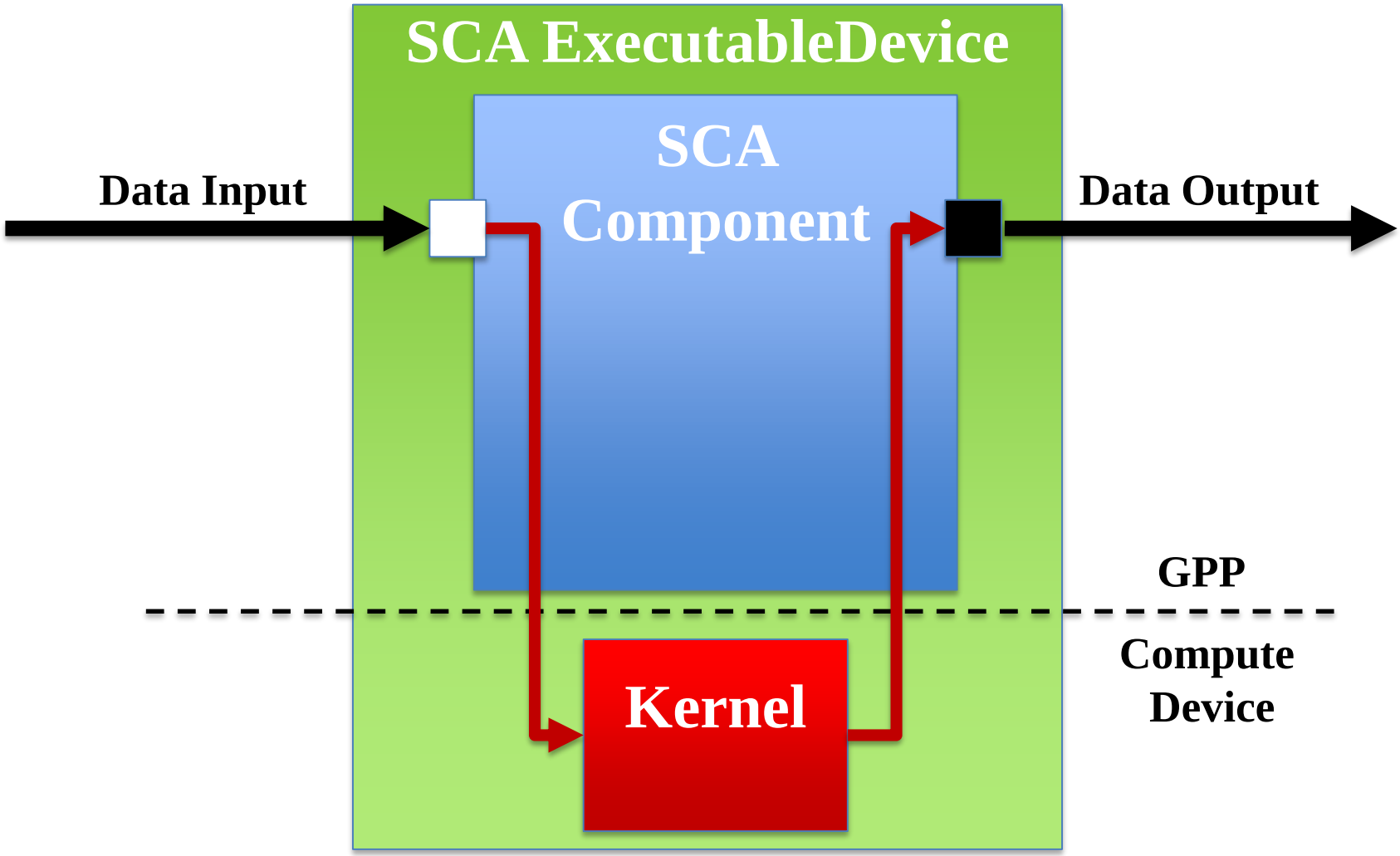
SCA and OpenCL (1)

- **NordiaSoft has an SCA ExecutableDevice that is OpenCL aware**
  - Works with any OpenCL compute device
  - Works with any number of Kernels per waveform component
  - Performs the setup of OpenCL via calls to drivers
  - Loads the kernels into the compute device

SCA and OpenCL (2)

- **NordiaSoft's SCA Architect™ performs code generation for OpenCL SCA components**
  - Defines dependencies between SCA Resource and OpenCL kernels
  - Generates code to copy the incoming input data from host to OpenCL Device
  - Generates code to copy the processed data from OpenCL Device to host
  - Generates code to push the output data to the next component(s) via the output port connection(s)

SCA and OpenCL (3)



– The End –

- **For more information, ask NordiaSoft for a demonstration**

- **Point of contact information:**
  - Steve.Bernier@NordiaSoft.com
  - David.Hagood@Aeroflex.com