XDF XILINX DEVELOPER FORUM

SDAccel Development Environment: FPGA Acceleration Performance and Ease of Use Aren't Mutually Exclusive

Presented By

BLACKLYNX

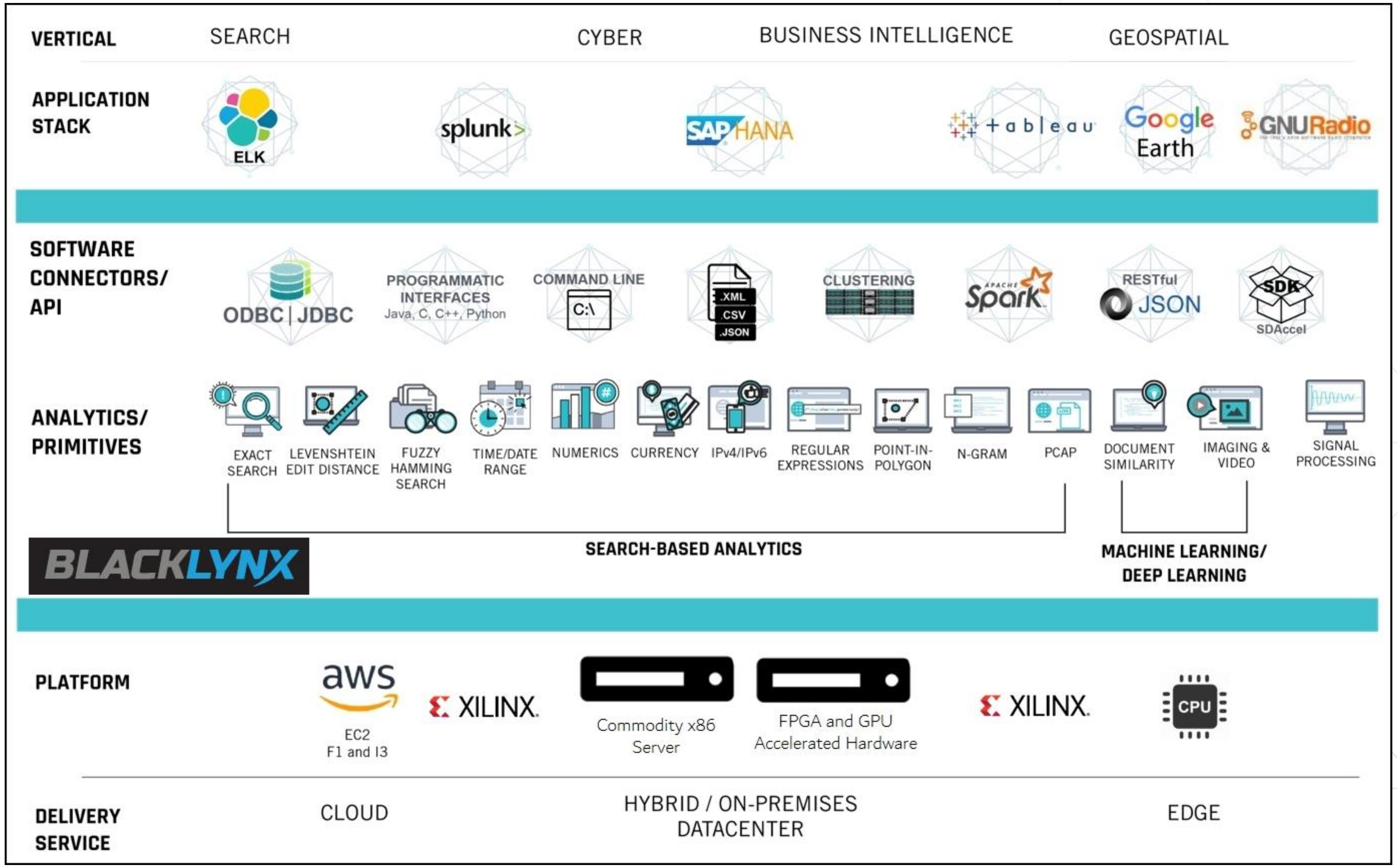
Pat McGarry, BlackLynx, *VP Engineering*
Neil Tender, BlackLynx, *Senior FPGA Engineer*
www.BlackLynx.tech
October 2, 2018

XILINX

BlackLynx

Who are we? What do we do?

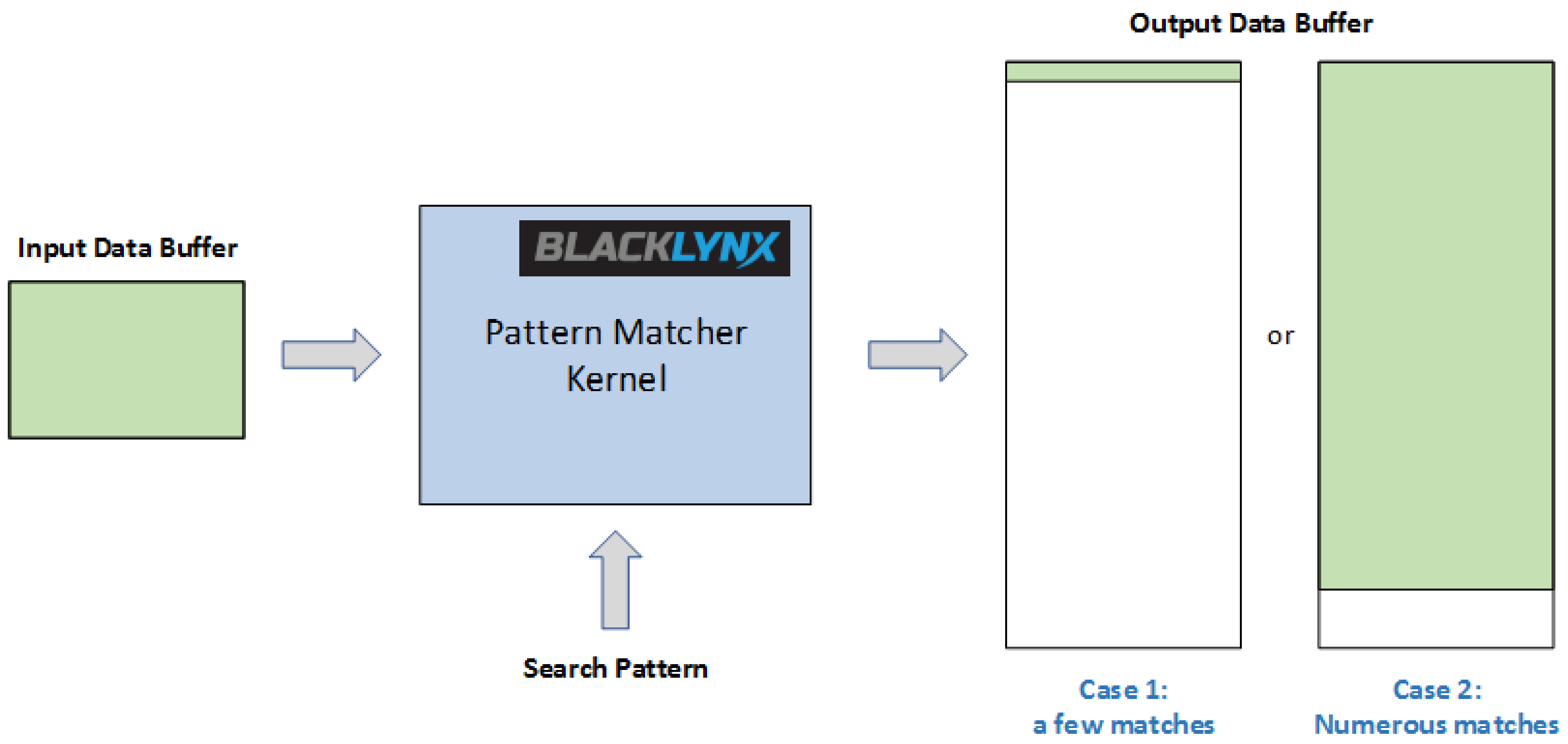
- High Performance Analytics at any scale, for any data type
- Seamless integration to high-level application stacks leveraging open software APIs (C, C++, Java, Python, ODBC, JDBC, REST, Spark, and more)
- Novel software-parallel control framework for automated, intelligent heterogeneous acceleration (for example, via FPGA offload by way of SDAccel)
- Deployment platform agnostic: in the cloud, data center, and/or at the edge
- AI/ML-based observation anywhere in the pipeline
- Customer sectors are national security, global telecommunications, and large financials



BlackLynx SDAccel Platforms Supported

Amazon EC2 F1 Instances

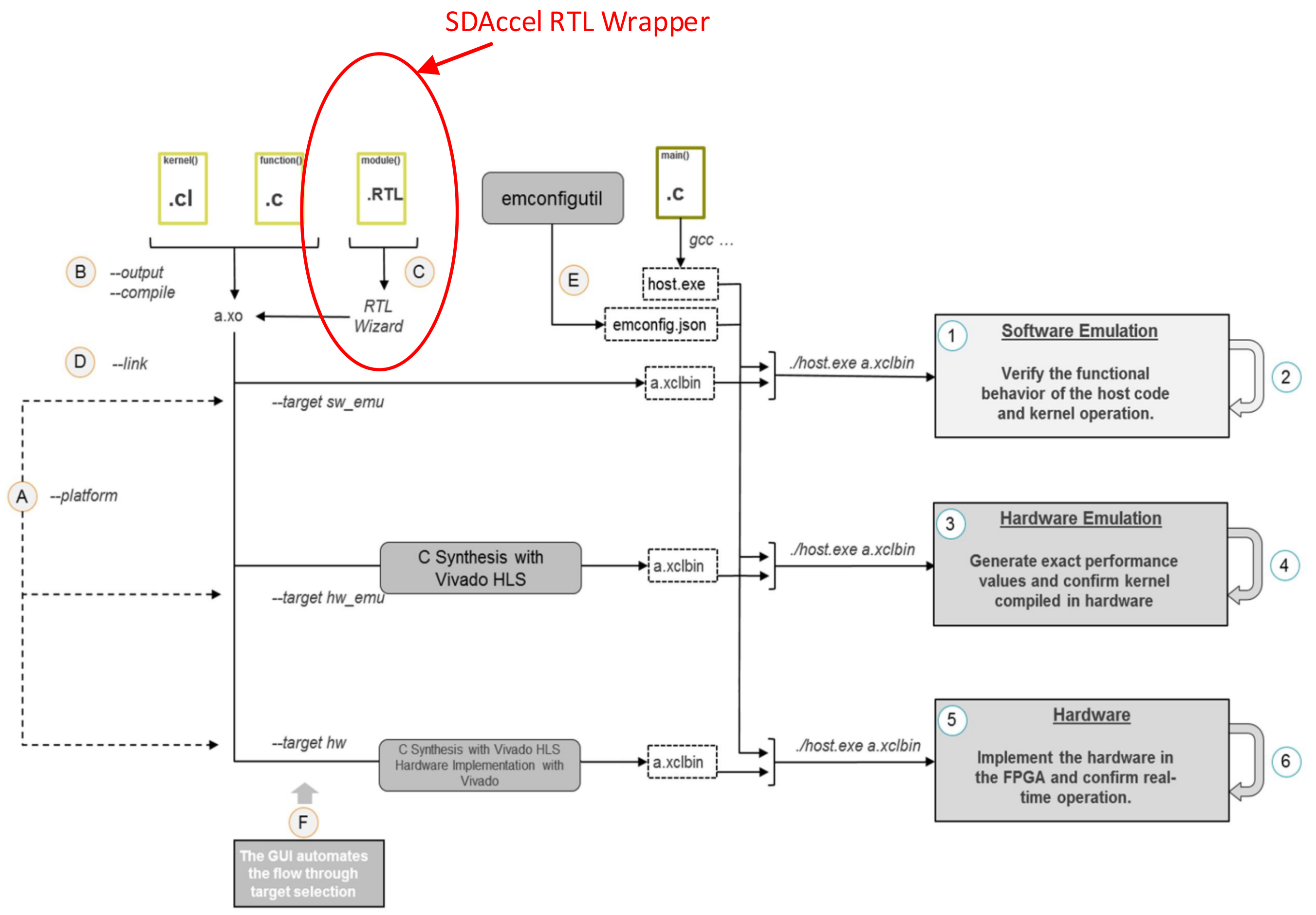
Example BlackLynx Use Case – Pattern Matching



> **Fixed-size input data, variable-size output data**
> **Several different pattern matching algorithms**
> **Legacy RTL code**

SDAccel Execution Model

The SDAccel environment supports three flows:



SDAccel RTL Wrapper
kernel()
.cl
function()
.c
module()
.RTL
emconfigutil
main()
.c
gcc ...
B --output --compile
C
RTL Wizard
a.xo
E
host.exe
emconfig.json
D --link
a.xclbin
--target sw_emu
./host.exe a.xclbin
1 Software Emulation
Verify the functional behavior of the host code and kernel operation.
2
A --platform
C Synthesis with Vivado HLS
a.xclbin
--target hw_emu
./host.exe a.xclbin
3 Hardware Emulation
Generate exact performance values and confirm kernel compiled in hardware
4
--target hw
C Synthesis with Vivado HLS Hardware Implementation with Vivado
a.xclbin
./host.exe a.xclbin
5 Hardware
Implement the hardware in the FPGA and confirm real-time operation.
6
F
The GUI automates the flow through target selection

Software Emulation –
OpenCL C kernels

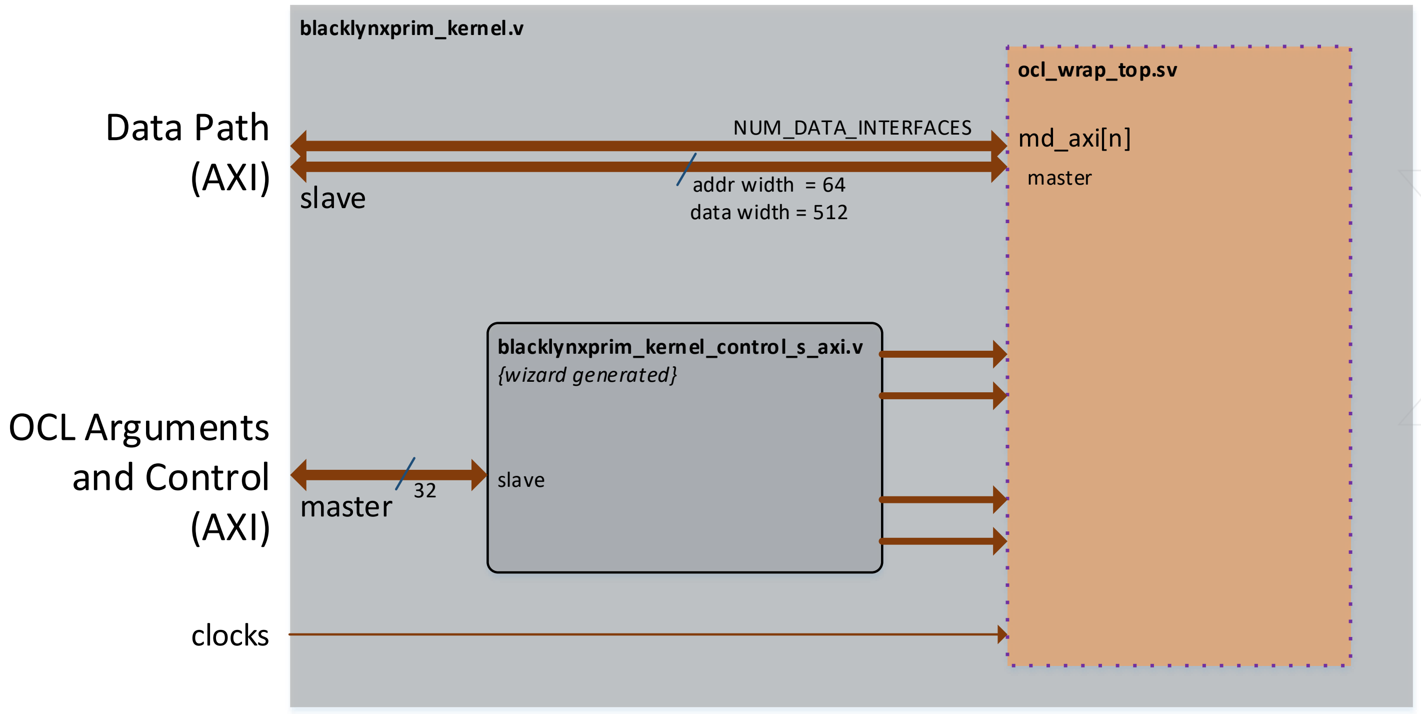
Hardware Emulation
compiled or native RTL kernels (RTL simulation)

Hardware
actual implementation (synthesis/place and route) to run on target hardware

RTL Kernels

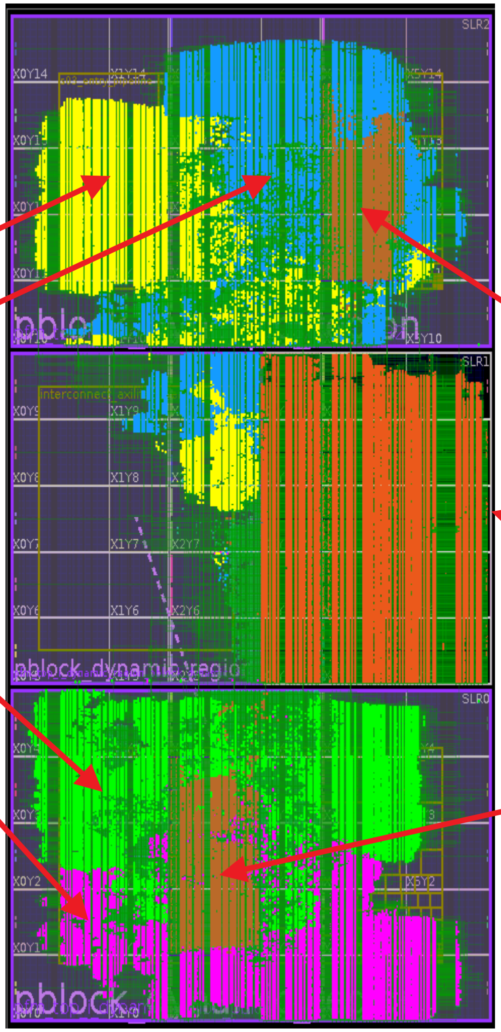
> To support legacy RTL primitives, we built a dedicated RTL kernel and wrapper to provide an adaption layer
> RTL Kernel Wizard automatically creates example design (RTL source and test bench)

BlackLynx RTL primitives can now be seamlessly dropped into the RTL kernel



Notes :
(1) Gray blocks are automatically generated by the RTL Kernel Wizard
(2) Orange Block is custom code

SDAccel Build Example



kernels
(dynamic region)

shell
(static region)

DDR bank
interfaces

Xilinx XCVU9P

Data Transfer Performance (between host and kernels)

> **Getting the best performance for variable-size output data required careful consideration of host/kernel memory transfer approach:**

<u>OpenCL memory transfer methods as implemented in SDAccel</u>

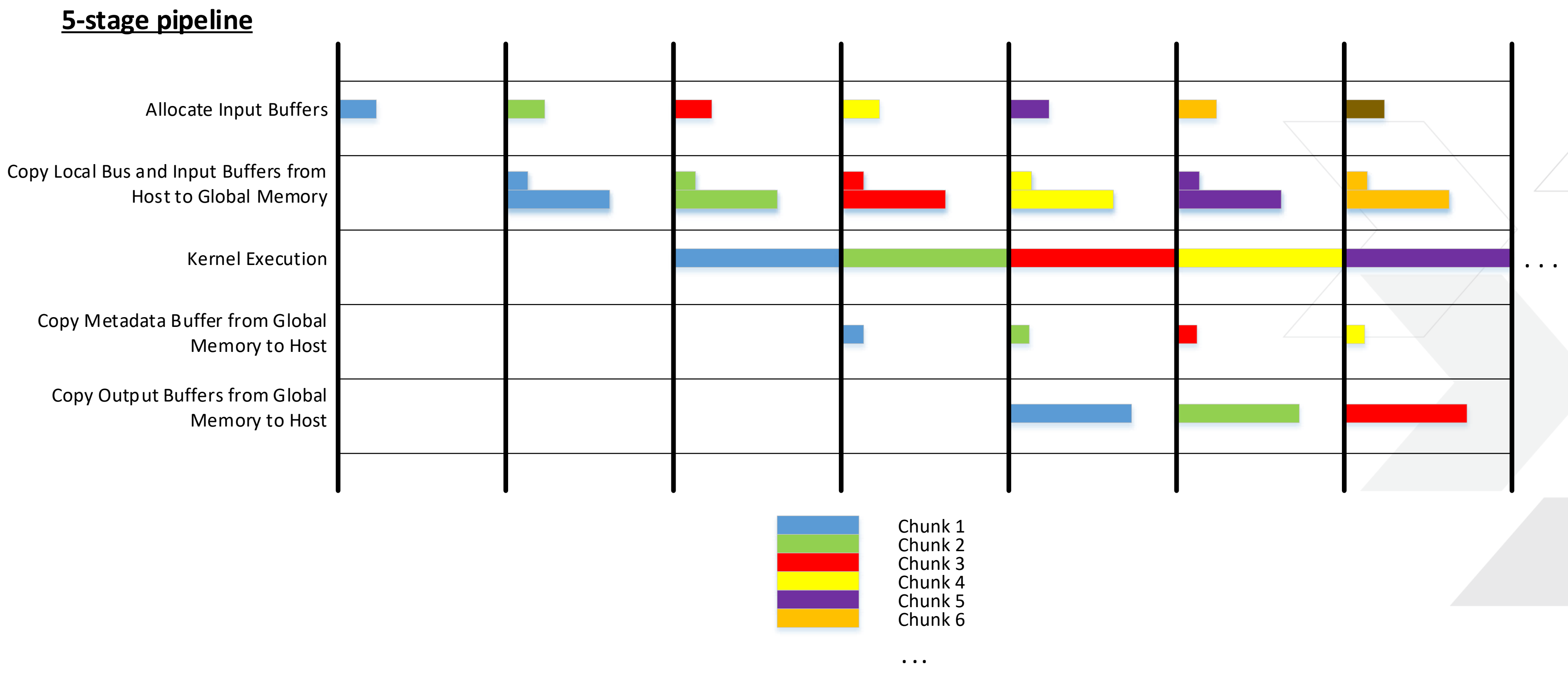
API call	Pros/Cons	Best for:
enqueueReadBuffer/ enqueueWriteBuffer	Flexible but slow	small output data
enqueueMigrateMemObject	Fast, uses pinned memory, but requires allocating entire output buffer upfront	large output data
enqueueMapBuffer	Fast, uses pinned memory, but requires allocating entire output buffer upfront, OpenCL function returns host pointer	large output data

> **OpenCL mechanism for control and handshaking between host and kernels is restrictive**

- Output data size is not known upfront, so buffers must be allocated for maximum possible size
- Statistics and other metadata from the kernel (e.g. output data size) must themselves be sent via buffers

Data Transfer Performance (cont'd)

To get best I/O performance, use pipelining to overlap memory data transfers and kernel execution:



Data Transfer Performance (cont’d)

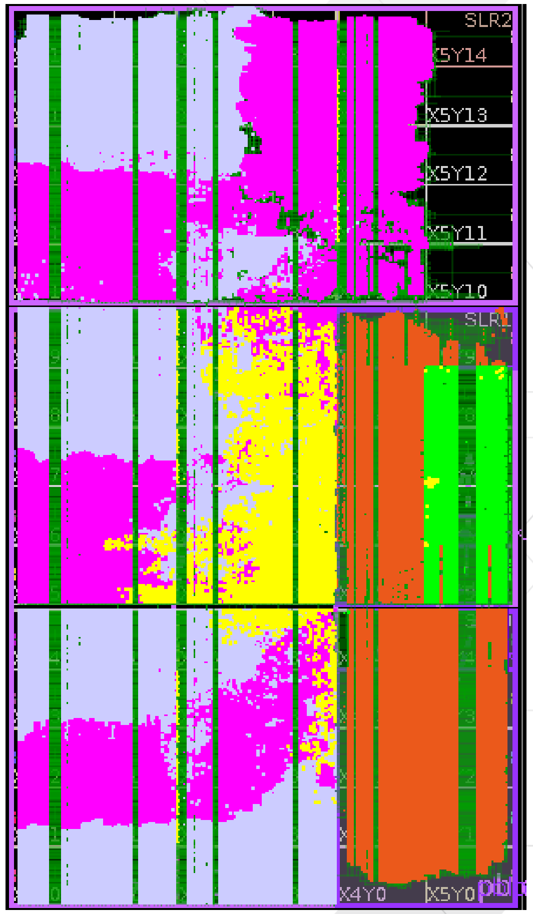
> **SDAccel is best suited for applications that are compute-bound, not I/O bound.**
> **In the best case, the PCIe bus caps out at around 9GB/s. (lower in practical uses)**

<u>Xilinx host_global_bandwidth example running on VCU1525 board:</u>

```
OpenCL migration BW host to device: 3.91702 MB/s for buffer size 64 with 1024 buffers
OpenCL migration BW device to host: 4.0294 MB/s for buffer size 64 with 1024 buffers
OpenCL migration BW host to device: 15.8509 MB/s for buffer size 256 with 1024 buffers
OpenCL migration BW device to host: 16.5981 MB/s for buffer size 256 with 1024 buffers
OpenCL migration BW host to device: 32.2082 MB/s for buffer size 512 with 1024 buffers
OpenCL migration BW device to host: 35.1914 MB/s for buffer size 512 with 1024 buffers
OpenCL migration BW host to device: 63.8203 MB/s for buffer size 1024 with 1024 buffers
OpenCL migration BW device to host: 67.4536 MB/s for buffer size 1024 with 1024 buffers
OpenCL migration BW host to device: 256.492 MB/s for buffer size 4096 with 1024 buffers
OpenCL migration BW device to host: 257.848 MB/s for buffer size 4096 with 1024 buffers
OpenCL migration BW host to device: 974.066 MB/s for buffer size 16384 with 512 buffers
OpenCL migration BW device to host: 996.388 MB/s for buffer size 16384 with 512 buffers
OpenCL migration BW host to device: 6820.12 MB/s for buffer size 1048576 with 8 buffers
OpenCL migration BW device to host: 7759.46 MB/s for buffer size 1048576 with 8 buffers
OpenCL migration BW host to device: 6159.77 MB/s for buffer size 1048576 with 64 buffers
OpenCL migration BW device to host: 8160.14 MB/s for buffer size 1048576 with 64 buffers
OpenCL migration BW host to device: 6764.79 MB/s for buffer size 1048576 with 256 buffers
OpenCL migration BW device to host: 8677.38 MB/s for buffer size 1048576 with 256 buffers
OpenCL migration BW host to device: 6719.87 MB/s for buffer size 2097152 with 8 buffers
OpenCL migration BW device to host: 7376.67 MB/s for buffer size 2097152 with 8 buffers
OpenCL migration BW host to device: 6997.59 MB/s for buffer size 2097152 with 64 buffers
OpenCL migration BW device to host: 8177.87 MB/s for buffer size 2097152 with 64 buffers
OpenCL migration BW host to device: 7534.95 MB/s for buffer size 2097152 with 256 buffers
OpenCL migration BW device to host: 8438.96 MB/s for buffer size 2097152 with 256 buffers
OpenCL migration BW host to device: 7488.66 MB/s for buffer size 16777216 with 64 buffers
OpenCL migration BW device to host: 9095.59 MB/s for buffer size 16777216 with 64 buffers
OpenCL migration BW host to device: 7465.79 MB/s for buffer size 268435456 with 4 buffers
OpenCL migration BW device to host: 7596.66 MB/s for buffer size 268435456 with 4 buffers
OpenCL migration BW host to device: 7430.14 MB/s for buffer size 536870912 with 2 buffers
OpenCL migration BW device to host: 9009.48 MB/s for buffer size 536870912 with 2 buffers
```

Performance and Capacity

- **# Kernels**
  - Instantiate multiple parallel kernels to maximize performance
- **DDR Memory Interfaces**
  - Each kernel can be assigned to any of the 4 interfaces
  - Utilizing more than one DDR interface improves routability and can help with I/O performance
- **Clock frequency**
  - SDAccel attempts to achieve a user-specified target clock frequency, but backs down automatically if closure cannot be achieved
  - Best to experiment with achievable clock frequencies and choose a reasonable target



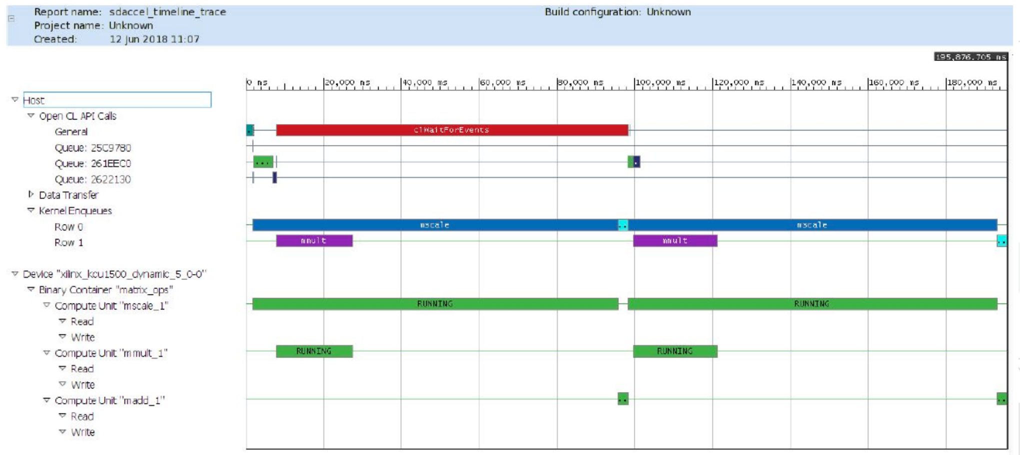
8-kernel design on AWS F1

Design Optimization and Debugging

SDAccel provides a number of features to help the developer optimize the design and debug issues:

- System estimate, profile summary, application timeline, waveform view, guidance

At BlackLynx, we created a test loopback primitive to help verify the integrity and performance of the control and dataflow, which proved extremely useful.



Report name: sdaccel_timeline_trace
Project name: Unknown
Created: 12 Jun 2018 11:07
Build configuration: Unknown

Application timeline

SDAccel Profiling and Optimization Guide, UG1207

Onboarding examples

Xilinx provides a full set of example reference designs

- Available for download from GitHub
- Complete source code, build files, scripts
- Covers a wide range of features SDAccel features

Examples:

- OpenCL acceleration algorithms
  - K-means, Smith-Waterman
- Data transfers
- RTL kernels
- OpenCL kernel optimization techniques
- Debugging/profiling

Summary

> SDAccel extends the relevance of heterogeneous computing to seamlessly target FPGA-enabled cloud instances, hardware in the datacenter, and edge devices

> Designs are portable across a number of available platforms and are easily scaled to provide flexibility in trading of performance with size

> SDAccel makes designing for FPGAs easier and faster by supporting high level languages (OpenCL/C++) while preserving the ability to use customized RTL designs

> We have shared our experiences and recommendations on how to achieve the best results when implementing SDAccel-based designs

BlackLynx is moving full speed ahead leveraging SDAccel to accelerate our heterogeneous computing acceleration flows.

XDF XILINX DEVELOPER FORUM

BLACKLYNX

http://www.BlackLynx.tech