XDF XILINX DEVELOPER FORUM

Fundamentals of FPGA-based Acceleration

Presented By

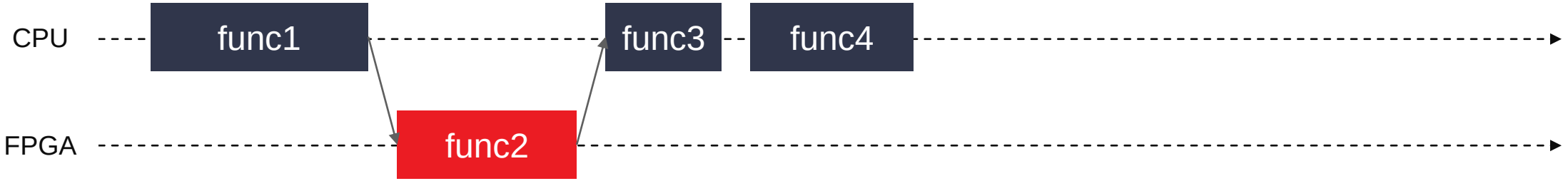
Thomas Bollaert
Senior Director, SW Applications
October 2nd, 2018

XILINX

FPGA Acceleration : Boosting Application Performance

Without acceleration

With FPGA acceleration



FPGA handles compute-intensive, deeply pipelined, massively parallel operations. CPU handles the rest

Customizable Architectures – The FPGA Advantage

CPUs and GPUs

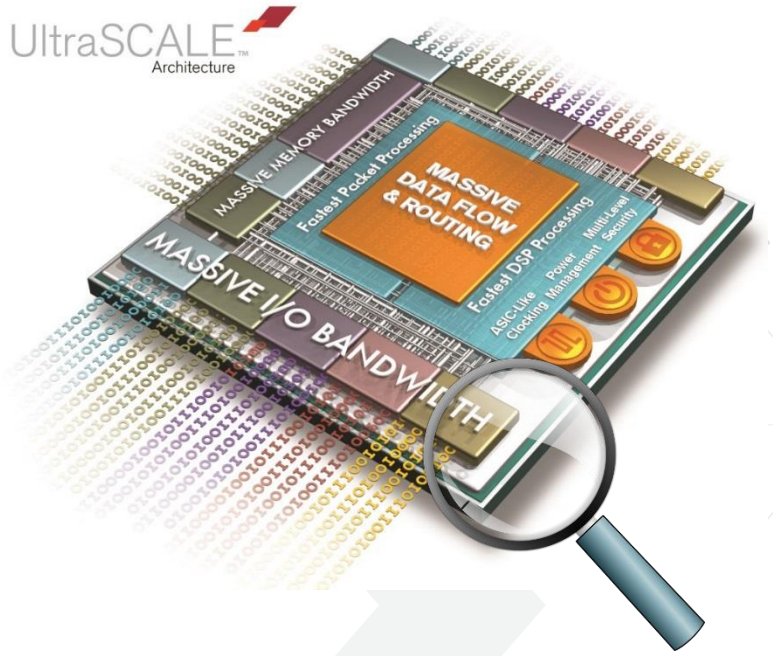
- Fixed instruction set and rigid memory hierarchy
- Developer adapts the program to the architecture

FPGAs

- Flexible, fully customizable architecture
- Developer adapts the architecture to the program

FPGAs – The Ultimate Parallel Processing Device

- **No predefined instruction set or underlying architecture**
- **Developer customizes the architecture to his needs**
  - Custom datapaths
  - Custom bit-width
  - Custom memory hierarchies
- **Excels at all types of parallelism**
  - Deeply pipelined (e.g. Video codecs)
  - Bit manipulations (e.g. AES, SHA)
  - Wide datapath (e.g. DNN)
  - Custom memory hierarchy (e.g: Data analytics)
- **Adapts to evolving algorithms and workload needs**

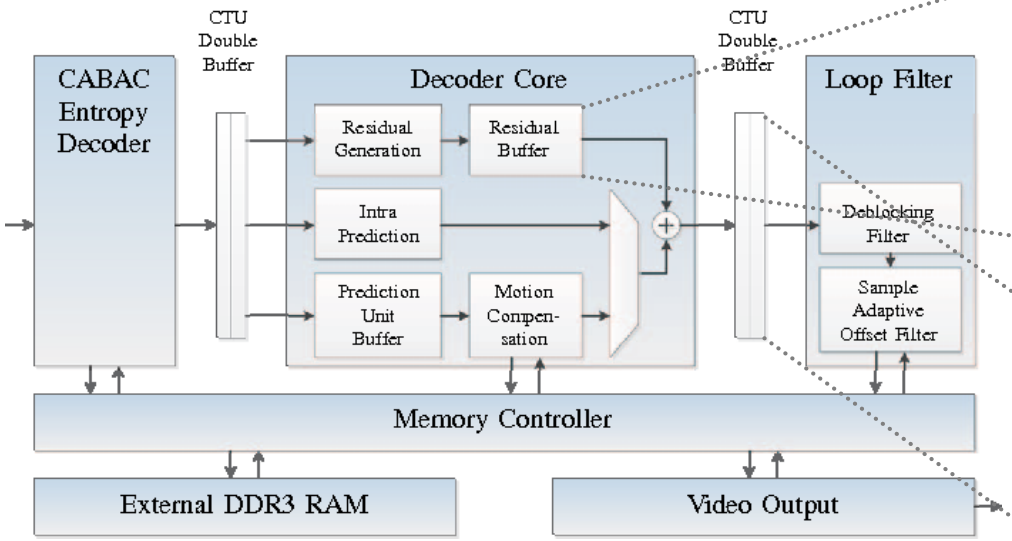


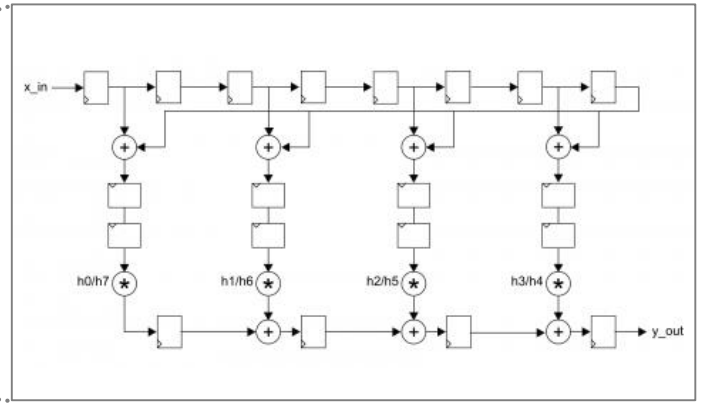
2.5 million system logic cells

6,800 DSP engines

345 Mb on-chip memory

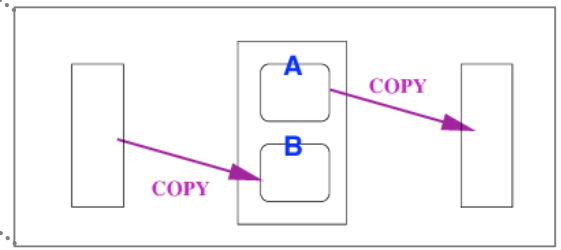
Custom Architectures – The Key to Acceleration





- Custom dataflow pipelines
- Multiple stages executing simultaneously
- Streaming programming model

- Custom datapaths, parallel and pipelined
- User-defined bitwidths



- Custom memory architectures
- Double-buffers, FIFOs, Shift-registers

FPGAs – High-Performance and Versatility

Speech Recognition

Song Han et al. "ESE: Efficient Speech Recognition Engine with Sparse LSTM on FPGA.“ International Symposium on FPGA 2017. [Link]

43x

Database Analytics

Jian Ouyang et al. "SDA: Software-Defined Accelerator for General-Purpose Big Data Analysis System." Hotchips 2014. [Link]

25x

Pattern Matching

Shreyas G Singapura et al. "FPGA Based Accelerator for Pattern Matching in YARA Framework.“ CENG 2015. [Link]

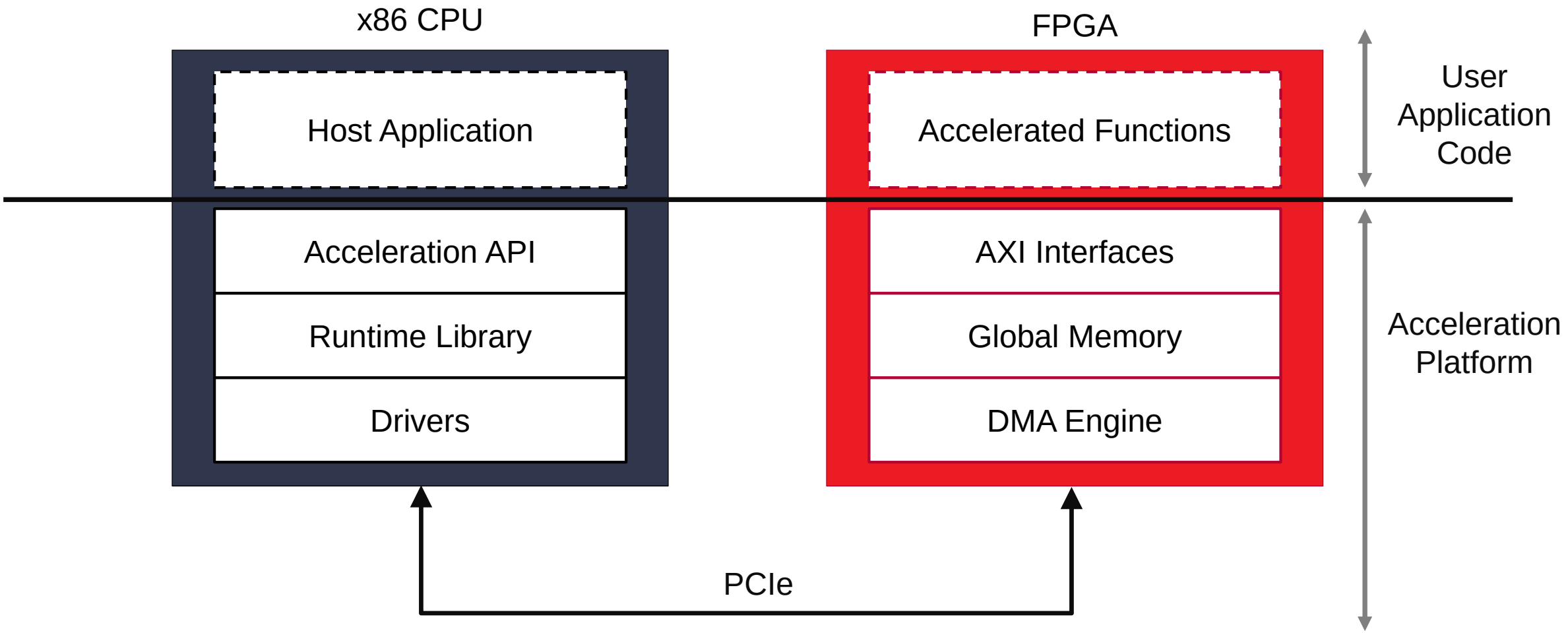
18x

Genomic Analysis

Edico Genome. "DRAGEN Genome Pipeline." Last accessed April 6, 2017. [Link]

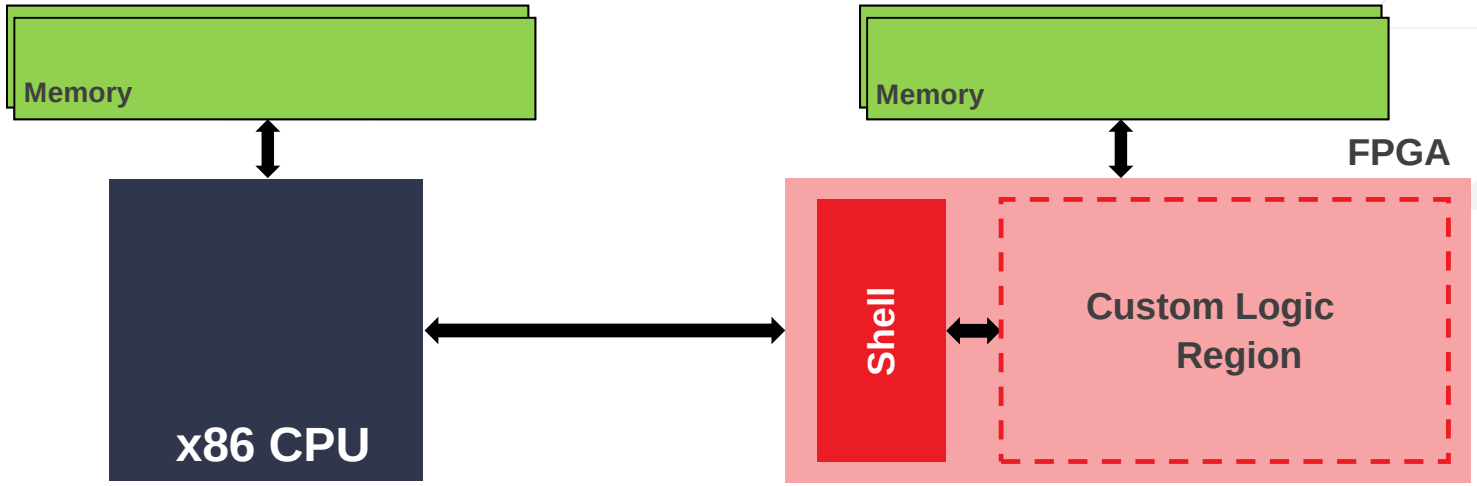
80x

Architecture of an FPGA Accelerated Application



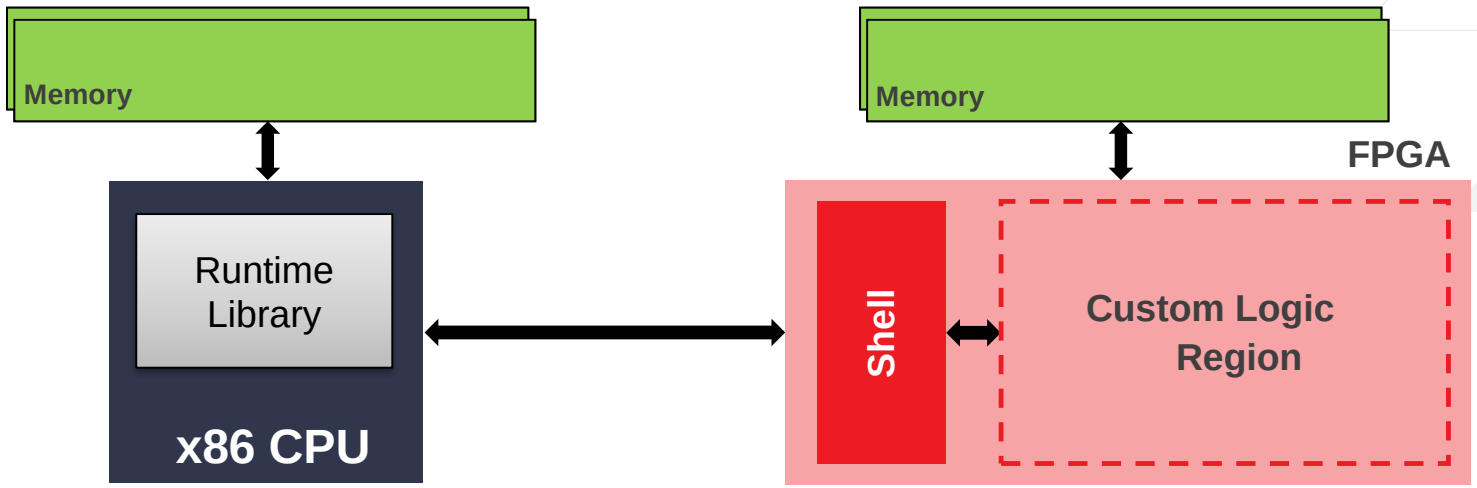
Initializing the Application

- **When the application starts, the FPGA device only contains the “Shell”**
  - The Shell will be managing the communications with the host



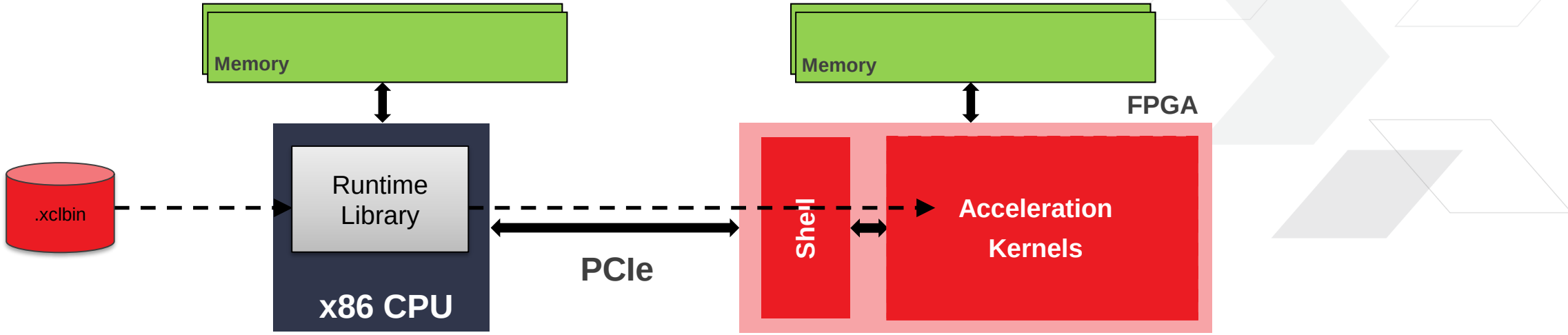
Initializing the Application

- **When the application starts, the FPGA device only contains the "Shell"**
  - The Shell will be managing the communications with the host
- **First, the host initializes the runtime**



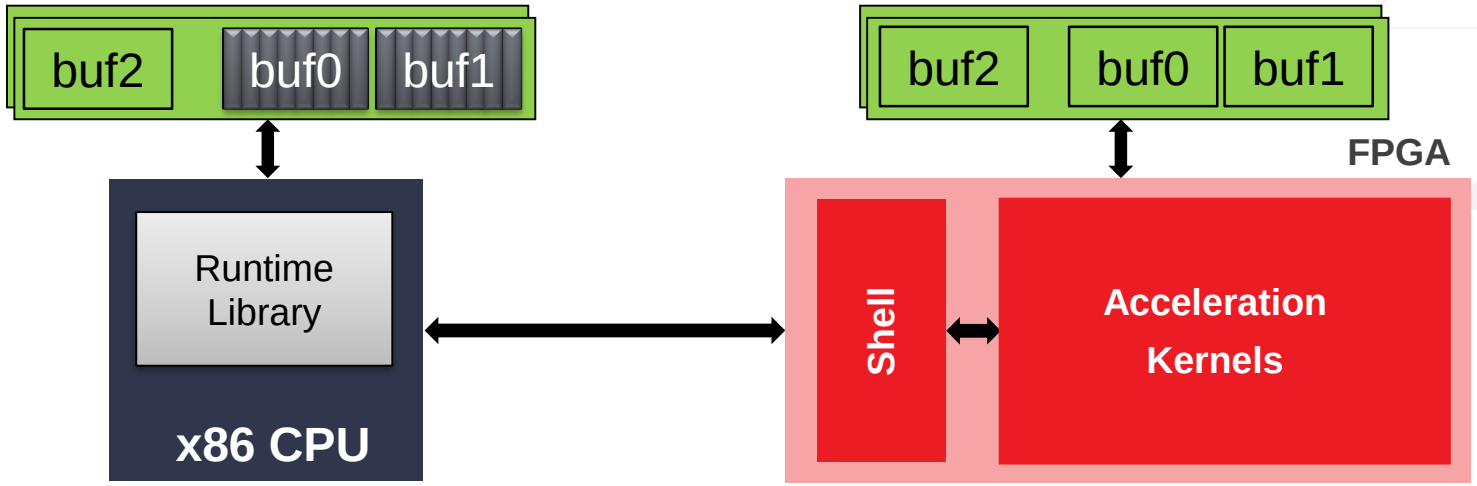
Initializing the Application

- **When the application starts, the FPGA device only contains the "Shell"**
  - The Shell will be managing the communications with the host
- **First, the host initializes the runtime**
- **Then programs the device with the desired FPGA binary**



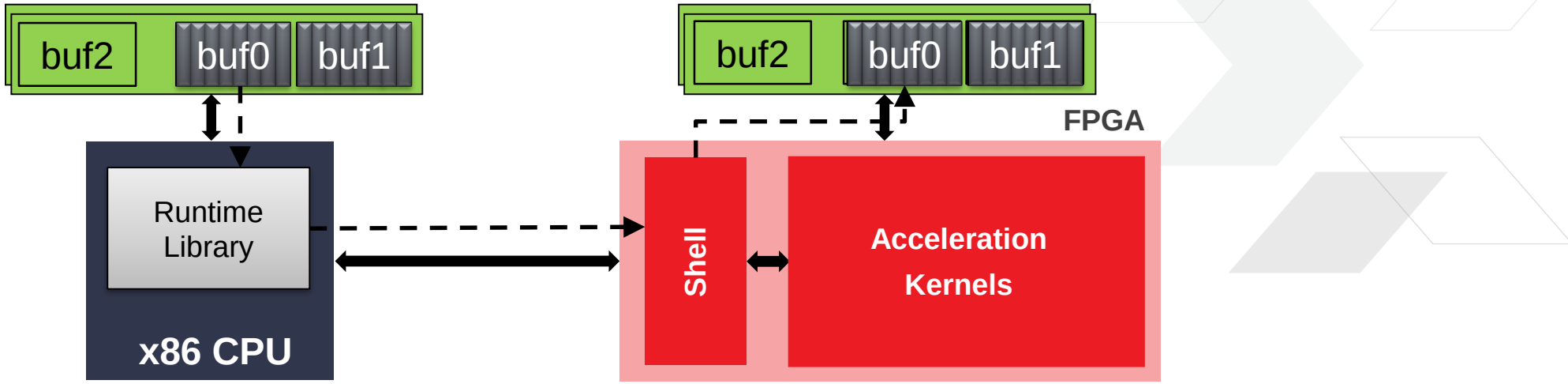
Allocating Buffers and Migrating Data to the Device

- **Host allocates input and output buffers in the device**
  - Buffers are used to transfer data from the CPU to the FPGA and back



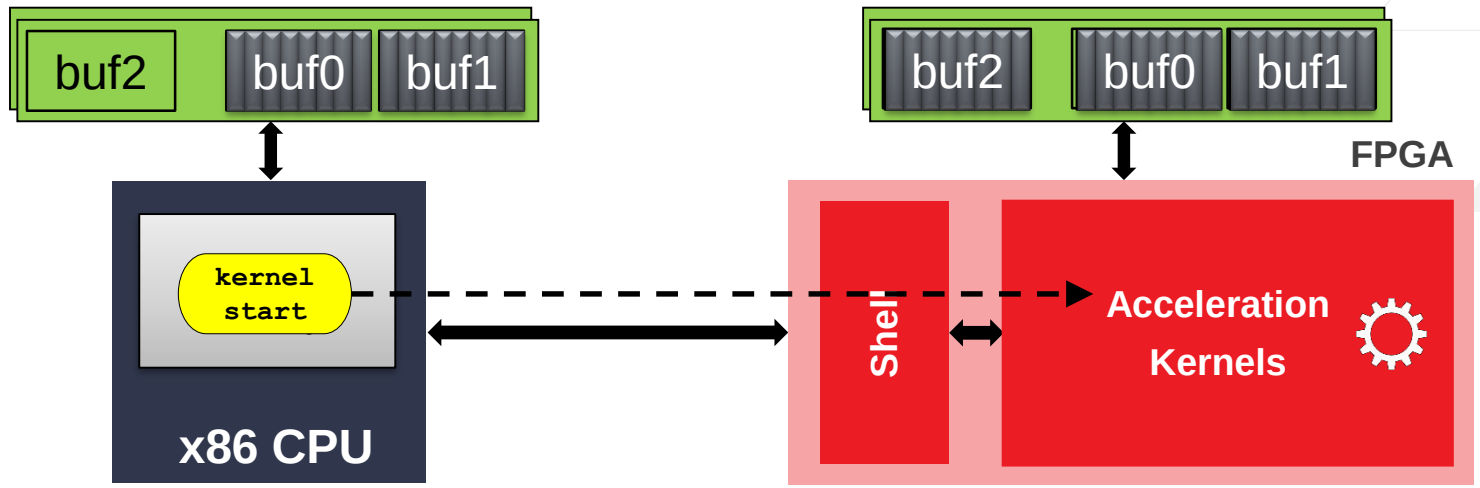
Allocating Buffers and Migrating Data to the Device

- **Host allocates input and output buffers in the device**
  - Buffers are used to transfer data from the CPU to the FPGA and back
- **Host migrates data to be processed by the accelerator to the buffer FPGA**



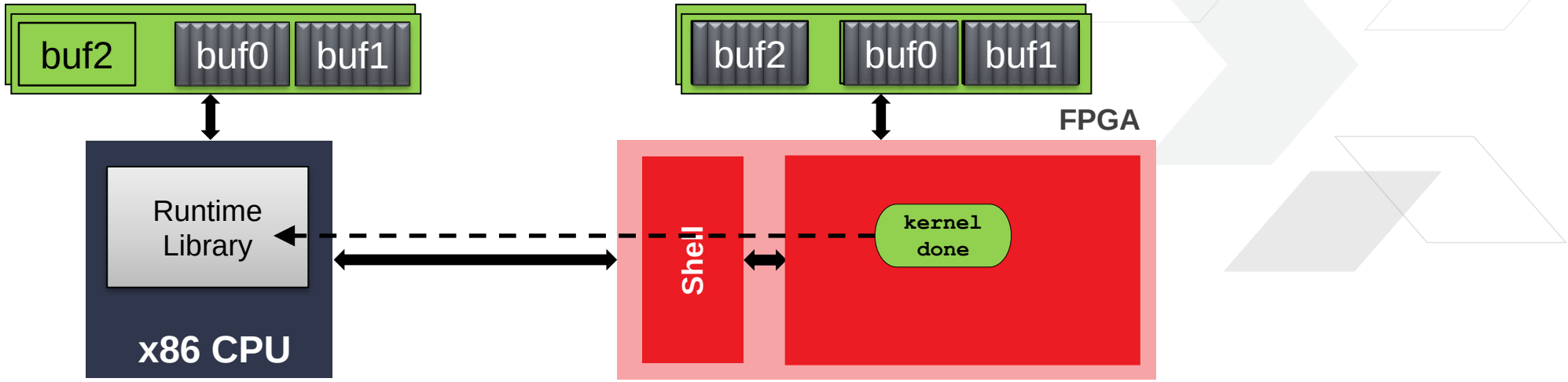
Running the Accelerators

- Host schedules execution of the desired kernel
- The Runtime is responsible for starting the kernel at the right moment
- Kernel reads data from the input buffer, processes it and writes results in the output buffer previously allocated



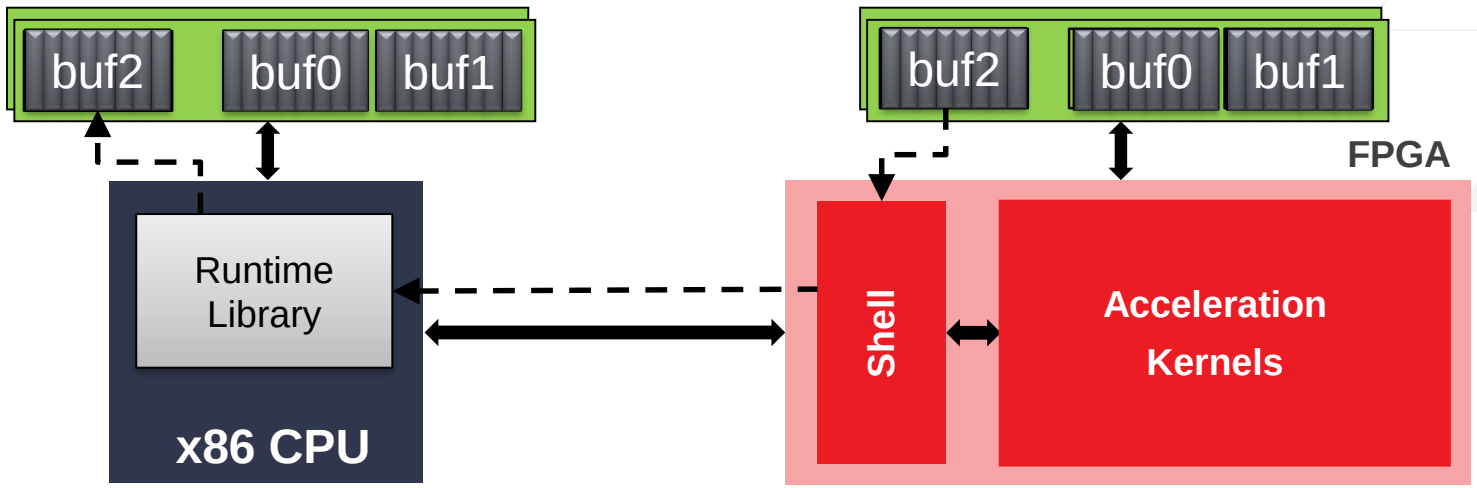
Running the Accelerators

- Host schedules execution of the desired kernel
- The Runtime is responsible for starting the kernel at the right moment
- Kernel reads data from the input buffer, processes it and writes results in the output buffer previously allocated
- After the kernel finishes processing the data, it notifies the host



Migrating Data Back to the Host

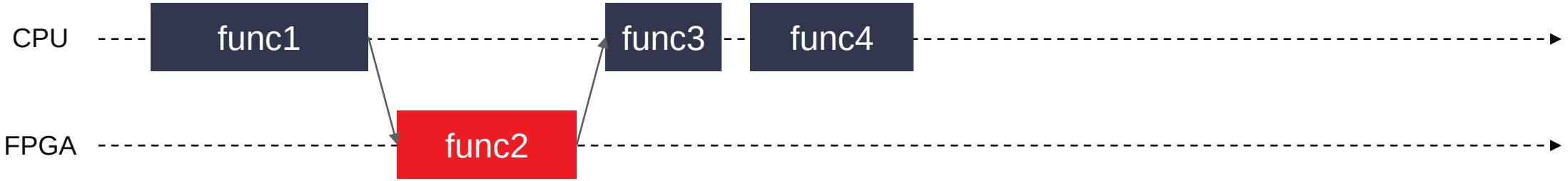
- **The host retrieves the results by scheduling a copy of the desired buffer back to host memory**



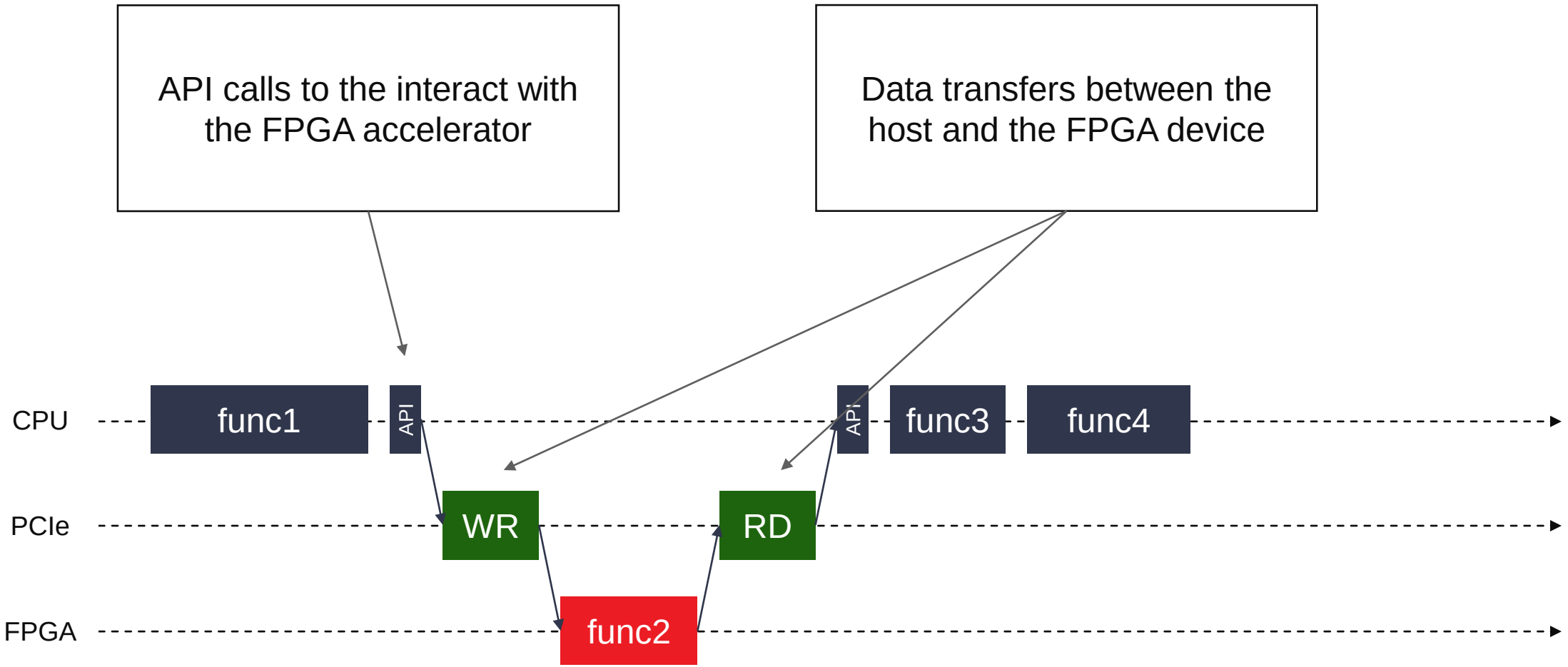
FPGA Acceleration : A Simplified View

Without acceleration

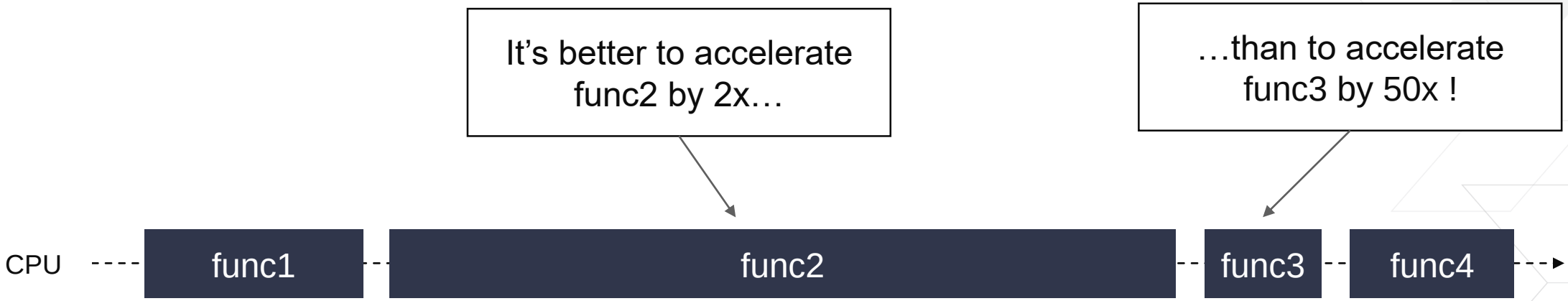
With FPGA acceleration



FPGA Acceleration : A More Accurate View

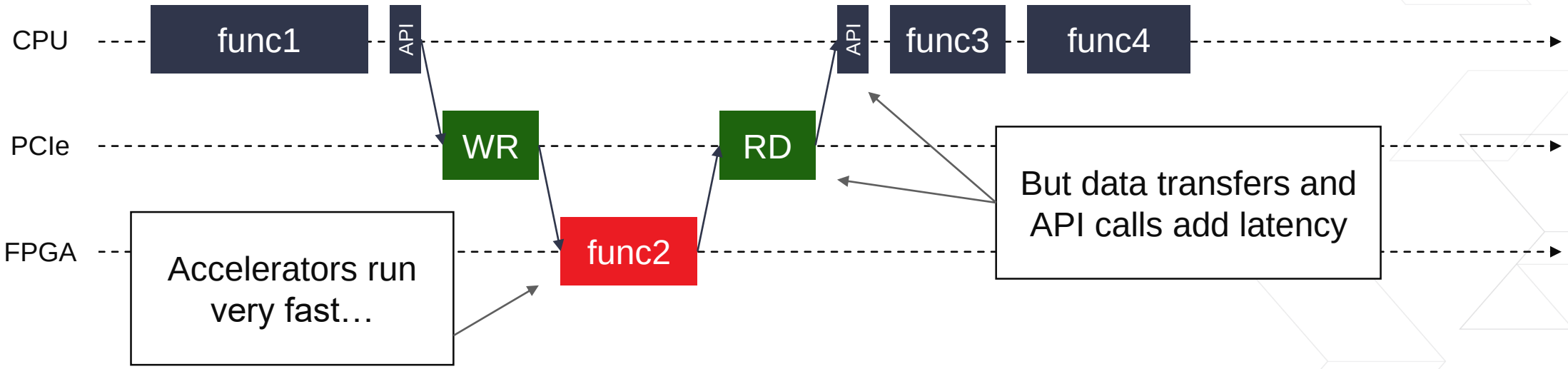


Rule #1 – Remember Amdahl's Law



- **Consider overall performance, not just individual functions**
- **When working "top-down", identify performance bottlenecks in the application**
  - Use profiling tools, analyze the "roof line" of a Flame Graph
- **Target accelerators that will impact end-to-end performance of the application**

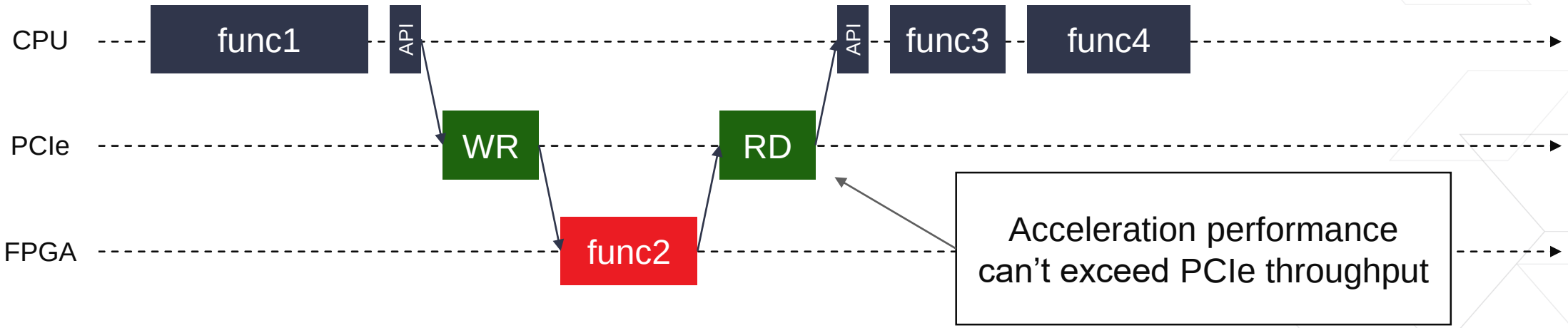
Rule #2 – Target Large, Compute-Bound Tasks



- **Look for functions with where {compute time} is much greater than {data transfer time}**
  - Good: Monte-Carlo – a few inputs, a lot of computations
  - Bad: Vector-addition – 2x more inputs than computations

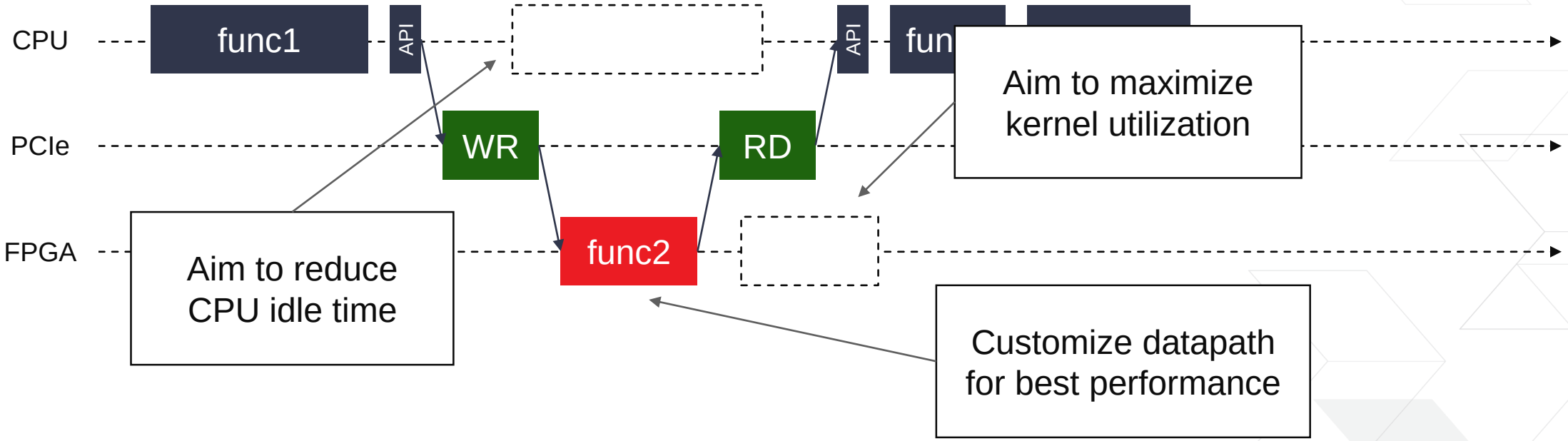
- **Prefer functions that perform a lot of processing per invocation to small functions which get called many times**
  - Minimizes API calls and event management overhead

Rule #3 - Know Your Ceiling



- **Compute-bound problems (on the CPU) are good for FPGA acceleration**
- **But maximum throughput will be limited by PCIe**
- **Maximum acceleration potential : {PCIe throughput} / {SW throughput}**

Rule #4 – Think Throughput, not only Latency



- **Target applications with inherent SW and HW parallelism**
  - Task-level, Data-level, Instruction-level, Bit-level parallelism

- **Adapt programming model to exploit parallelism**
  - Threads, asynchronous programming, dataflow model

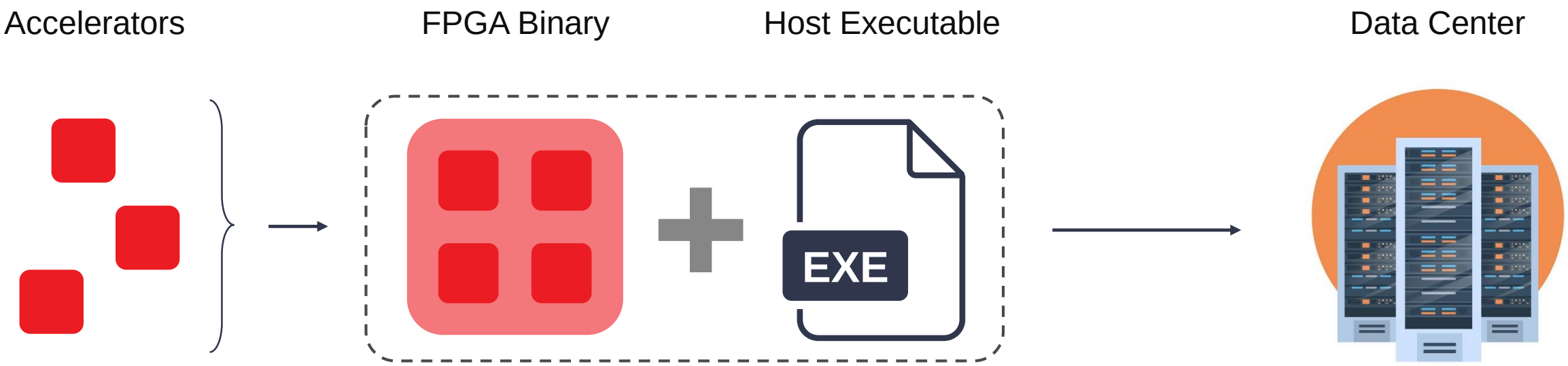
FPGA-Based Acceleration

- **When to USE**
  - Algorithm allows for parallelization
  - Many similar tasks

- **When May Not be beneficial**
  - Small problem size
  - Cost of Host to Device transfers outweighs benefit

- **When NOT beneficial**
  - Little to no parallelism
    - Algorithm is highly sequential over multiple data
    - Tasks are highly dependent

Developers vs End Users



Developers

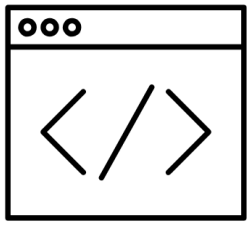
- Require development tools and skills
- Can use 3rd party libraries of accelerated functions
- Develop for own use or for external customers

End Users

- No need for development tools and skills
- Run Apps on FPGA-equipped servers
- Cloud or on-premise

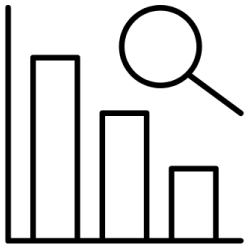
The SDAccel Development Environment

- **Fully integrated Eclipse-based IDE**
- **Develop host applications in C/C++**
- **Develop accelerators in RTL, C/C++ and OpenCL**
- **Debug, profiling and performance analysis tools**
- **Supports both GUI and command-line users**

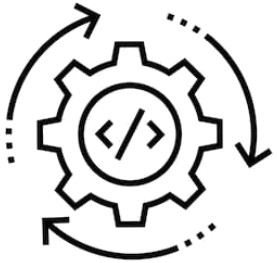


Develop

Debug

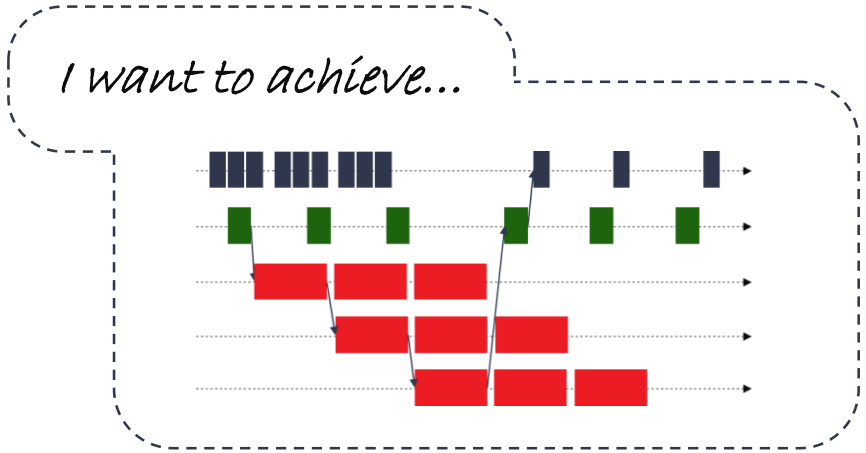


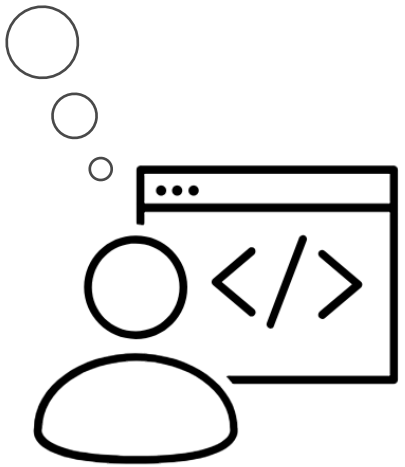
Profile

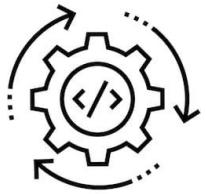


Optimize

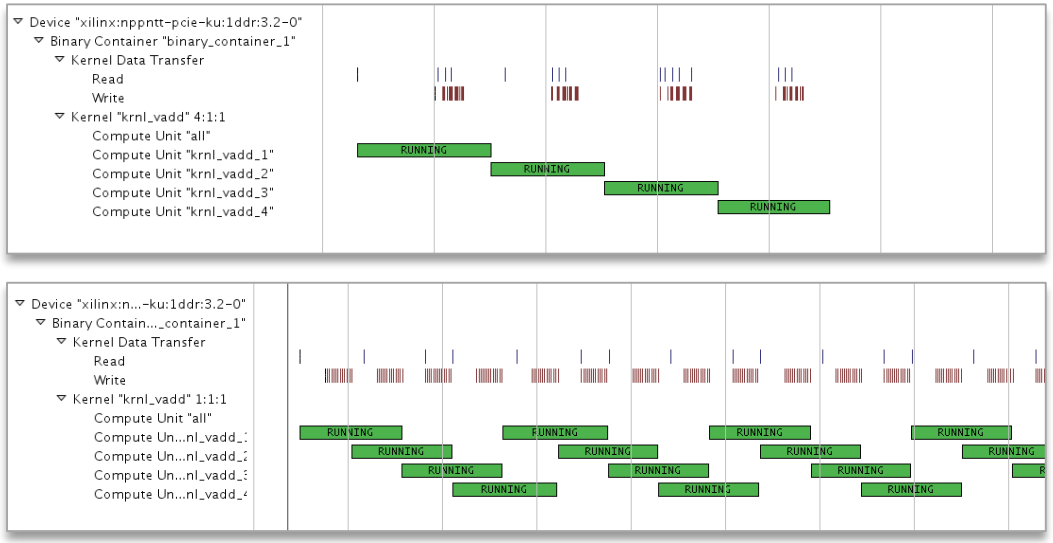
SDAccel Supports and Guides the Optimization Process







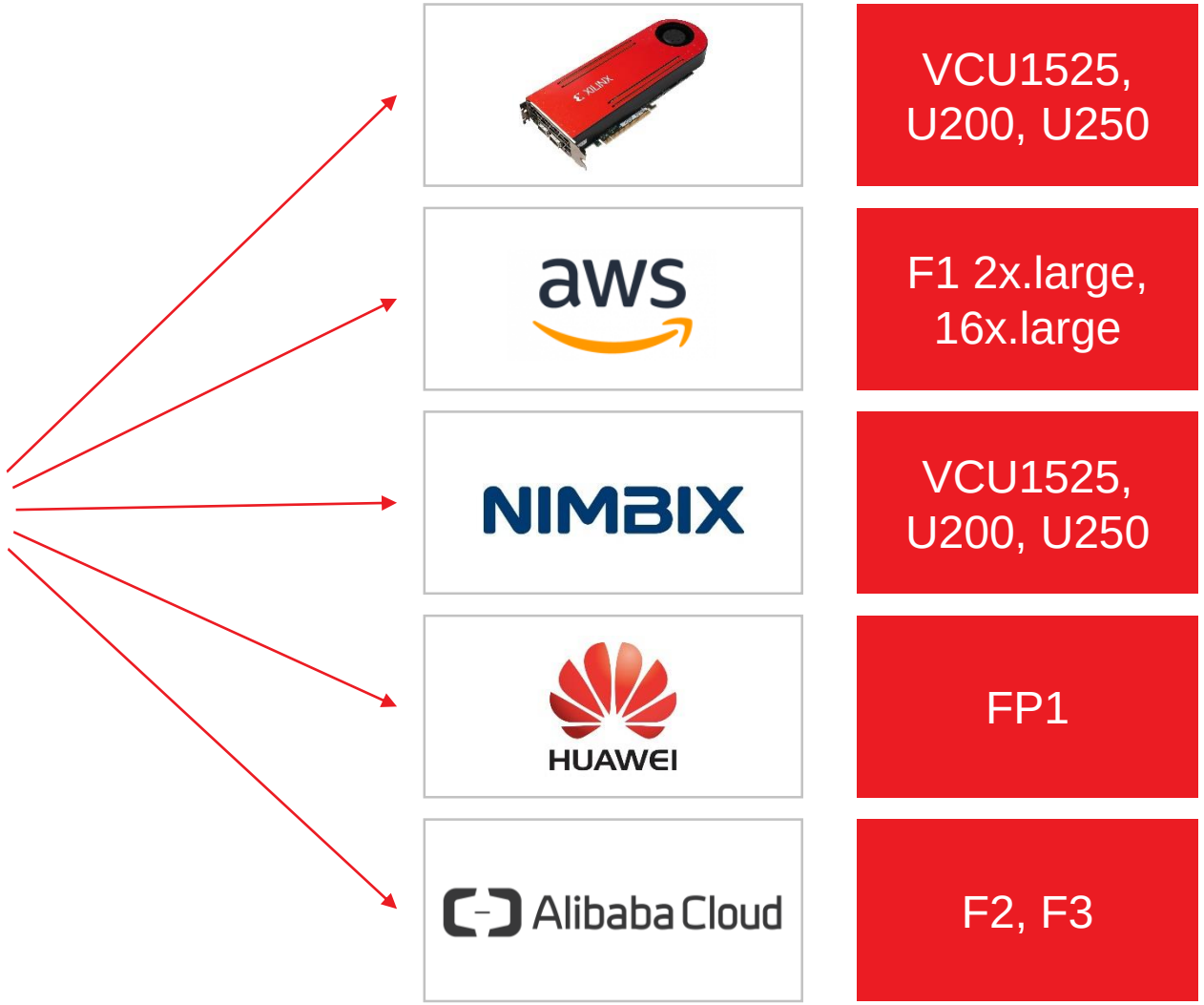
SDAccel Application Timeline View



- **Start with the end in mind → conceptualize desired results**
- **Use visualization and guidance tools → confirm and converge**

SDAccel Makes FPGA Applications Portable

> **Develop once**



> **Build for different target platforms**

> **Deploy on-premise or in the cloud**

Getting Started is Easy

- SDAccel Tutorials for U200 and VCU1525
- www.github.com/Xilinx/SDAccel-Tutorials

- On-Demand Developer Labs for AWS F1
- https://github.com/Xilinx/SDAccel-AWS-F1-Developer-Labs

- Free trial of the Nimbix FPGA Developer Program
- https://www.nimbix.net/fpga-developer-program/

Learn and practice how to accelerate applications with FPGAs