绝佳的并行处理 - FPGA 加速的根本基石

朱 勇
赛灵思大中华区业务拓展总监
2019 年 3 月 19 日
FPGA 加速：大幅提升应用的性能

Without acceleration

With FPGA acceleration

FPGA handles compute-intensive, deeply pipelined, massively parallel operations. CPU handles the rest.
定制化的架构 – FPGA 的优势

**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
FPGA – 绝佳的并行处理器件

> 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
定制化的架构 - 加速的关键

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

> Custom datatypes, parallel and pipelined
> User-defined bitwidths

> Custom memory architectures
> Double-buffers, FIFOs, Shift-registers
Speech Recognition

Database Analytics

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

Genomic Analysis
Edico Genome. "DRAGEN Genome Pipeline." Last accessed April 6, 2017. [Link]
FPGA加速应用的架构

- x86 CPU
  - Host Application
  - Acceleration API
  - Runtime Library
  - Drivers

- FPGA
  - Accelerated Functions
  - AXI Interfaces
  - Global Memory
  - DMA Engine

- PCIe

- User Application Code
- Acceleration Platform
When the application starts, the FPGA device only contains the “Shell”

The Shell will be managing the communications with the host
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
> 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
Host allocates input and output buffers in the device
   
   Buffers are used to transfer data from the CPU to the FPGA and back
分配缓冲区并将数据迁移到器件

> 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
> 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
> 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
The host retrieves the results by scheduling a copy of the desired buffer back to host memory.
FPGA 加速: 简单理解

Without acceleration

CPU → func1 → func2 → func3 → func4

With FPGA acceleration

CPU → func1 → func3 → func4

FPGA → func2
FPGA 加速：更准确的观点

API calls to the interact with the FPGA accelerator

Data transfers between the host and the FPGA device

API calls to the interact with the FPGA accelerator

Data transfers between the host and the FPGA device

CPU

func1

API

func2

WR

PCIe

RD

FPGA

func3

func4

API calls to the interact with the FPGA accelerator

Data transfers between the host and the FPGA device
规则 1 – 牢记阿姆达尔定律（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
规则 2 – 针对大型、计算绑定的任务

> **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

Accelerators run very fast…

But data transfers and API calls add latency
> Compute-bound problems (on the CPU) are good for FPGA acceleration

> But maximum throughput will be limited by PCIe

> Maximum acceleration potential : \( \frac{\text{PCIe throughput}}{\text{SW throughput}} \)
规则 4 – 考虑吞吐量，而不仅仅是时延

目标应用具有固有的 SW 和 HW 并行性

- 任务级、数据级、指令级、位级并行性

- 调整编程模型以利用并行性
  - 线程、异步编程、数据流模型

- 目标是减少 CPU 空闲时间

- 目标是最大化内核利用率

- 定制数据通路以获得最佳性能
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
开发者 vs 终端用户

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
SDAccel 开发环境

> 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
SDAccel Application Timeline View

I want to achieve...

- Start with the end in mind → conceptualize desired results
- Use visualization and guidance tools → confirm and converge
Xilinx 使 FPGA 可以便携应用

- Develop once
- Build for different target platforms
- Deploy on-premise or in the cloud
Learn and practice how to accelerate applications with FPGAs.
Adaptable.
Intelligent.