"Different" doesn't mean "Difficult": FPGA Programming Demystified

Sergei Storojev Staff Tools and Design Methodology Application Engineer



The Case for FPGAs in Financial Computing **Standard Initial Margin Model** 10x - 50xCalculation https://www.xilinx.com/products/boards-and-kits/alveo/applications/standard-initial-margin-model.html **Real-Time Risk Dashboard 89x** https://www.xilinx.com/products/boards-and-kits/alveo/applications/real-time-risk.html **High Performance Monte Carlo** 42x - 540x**Option Pricing Simulation** https://github.com/KitAway/FinancialModels AmazonF

© Copyright 2018 Xilinx

NOT STAC BENCHMARKS



Customizable Architectures – The FPGA Advantage

CPUs and GPUs

- > High speed, high power, low efficiency
- > Fixed instruction set and rigid memory hierarchy
- > Developer adapts the program to the architecture

FPGAs

- > High throughput, low power, high efficiency
- > 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



FPGA Accelerator Cards That Fit Your Performance Needs



> Alveo U200

- >> 18.6 Peak INT8 TOPs
- >> 77GB/s DDR Memory Bandwidth
- >> 31TB/s Internal SRAM Bandwidth
- >> 892,000 LUTs

> Alveo U250

- >> 33.3 Peak INT8 TOPs
- >> 77GB/s DDR Memory Bandwidth
- >> 38TB/s Internal SRAM Bandwidth
- >> 1,341,000 LUTs

> Deploy in the Cloud or On-Premises

E XILINX.

Sounds Great! But...

HW languages are low-level and very difficult

I know nothing of hardware programming concepts

How can I even get software applications to interact with FPGAs?



FPGA Solutions Have Come a Long Way

You can actually use C, C++ or OpenCL to program FPGAs

Familiar parallel programming concepts apply to FPGA development



Off-the-shelf platforms make SW acceleration with FPGAs very easy



Off-the-shelf Platforms Simplify FPGA Acceleration



- > No need to develop cards, HW interfaces or SW stacks
- > Acceleration platforms provides HW and SW APIs to ingrate your code
- > Focus on the application, not on the implementation details

Software Programmability: FPGA Development in C/C++



EXILINX.

Using C, C++ or OpenCL to Program FPGAs



- > Xilinx pioneered C to FPGA compilation technology (aka "HLS") in 2011
- > No need for low-level hardware description languages
- > Enables "Software Programmability" of FPGAs

High-Performance FPGA Applications: Think "Parallel"

> Data-level parallelism

>> Processing different blocks of a data set in parallel

> Task-level parallelism

- >> Executing different tasks in parallel
- >> Executing different tasks in a pipelined fashion

> Instruction-level parallelism

- >> Parallel instructions (superscalar)
- >> Pipelined instructions

> Bit-level parallelism

>> Custom word width



Task-Level Parallelism

```
for (int i=0; i<N; i++)
{
    #pragma HLS DATAFLOW
    func1();
    func2();
    func3();
}</pre>
```



- > Create custom dataflow pipelines
- > Multiple tasks executing simultaneously
- > Streaming programming paradigm



Instruction-Level Parallelism

```
for (int i=0; i<N; i++)
{
    acc += A[i] * B[i];
}</pre>
```





- > Create custom datapaths within each task
- > Execute hundreds of operations in parallel
- > Automatic or user guided compiler optimization

Learn More about Parallel Programming for FPGAs

- > Parallel Programming for FPGAs
- > Open-source book
- > Teaches HW and SW developers how to efficiently program FPGAs using high-level synthesis (HLS)
- > Freely available from: http://hlsbook.ucsd.edu



Follow the Guide!

> Expert system built-in the FPGA compiler

- > Analyzes results; provides actionable feedback on how to improve the design
- > Explicit messages, links to detailed explanation and solutions
- > Helps developers achieve better results

Name	Expects	Actual	Details		
ODR_BANK_CONNECTIONS #1	> 0	0	DDR bank 3 was used by 0 ports.		_
ODR_BANK_READ_TRANSFER_UTIL #1	> 5.000	0.432	DDR bank 0 read utilization was 0.432% on device xilinx_kcu1500_dynamic_5_0-	D.	
ODR_BANK_READ_TRANSFER_UTIL #2	> 5.000	0.169	DDR bank 1 read utilization was 0.169% on device xilinx_kcu1500_dynamic_5_0-	D.	
ODR_BANK_WRITE_TRANSFER_UTIL #1	> 5.000	0.432	DDR bank 1 write utilization was 0.432% on device xilinx_kcu1500_dynamic_5_0-	-0.	
• KERNEL_COUNT #1	> 1	1	Compute unit Filter2DKernel_1 is called 3 time(s).		
• KERNEL_COUNT #2	> 1	1	Compute unit Filter2DKernel_1 is called 3 time(s).		
• KERNEL_READ_TRANSFER_SIZE #1	> 0.512	0.128	Kernel average read size on port Filter2DKernel_1/m_axi_gmem0 was 0.128 KB.		
VERNEL_READ_TRANSFER_SIZE #2	> 0.512	0.016	Kernel average read size on port Filter2DKernel_1/m_axi_gmem1 was 0.016 KB.		
• KERNEL_WRITE_TRANSFER_SIZE #1	> 0.512	0.128	Kernel average write size on port Filter2DKernel_1/m_axi_gmem1 was 0.128 KB.		
• KERNEL_READ_TRANSFER_UTIL #1	> 5.000	0.432	Kernel read utilization on port Filter2DKernel_1/m_axi_gmem0 was 0.432%.		
• KERNEL_READ_TRANSFER_UTIL #2	> 5.000	0.169	Kernel read utilization on port Filter2DKernel_1/m_axi_gmem1 was 0.169%.		
• KERNEL_WRITE_TRANSFER_UTIL #1	> 5.000	0.432	Kernel write utilization on port Filter2DKernel_1/m_axi_gmem1 was 0.432%.		
• KERNEL_PORT_DATA_WIDTH #1	= 512	128	Port Filter2DKernel_1/m_axi_gmem0 has a data width of 128.		
VERNEL_PORT_DATA_WIDTH #2	= 512	128	Port Filter2DKernel_1/m_axi_gmem1 has a data width of 128.		
ODR_BANK_CONNECTIONS #2	> 0	0	DDR bank 1 was used by 0 ports.		
ODR_BANK_CONNECTIONS #3	> 0	0	DDR bank 2 was used by 0 ports.		
DDR_BANK_CONNECTIONS #4	> 0	2	DDR bank 0 was used by 2 ports.		
HOST_WRITE_TRANSFER_SIZE #1	> 4.096	5.058	Host write average size was 5.058 KB across 6 transfers.		
✓ CU_UTIL #1	> 10.000	196.714	Compute unit Filter2DKernel_1 was utilized 196.714% of the device time.		
✓ OVERUSED_CUS #1	< 16	1	Kernel Filter2DKernel required 1 compute unit call(s).		
✓ KERNEL_UTIL #1	= 100.000	600.000	Kernel Filter2DKernel: global size: 1, local size: 6.		
HOST_READ_TRANSFER_SIZE #1	> 4.096	9.216	Host read average size was 9.216 KB across 3 transfers.		
KERNEL_READ_TRANSFER_AMOUNT_MAX #1	< 2.000	1.267	Total kernel read of 0.038448 MB on xilinx_kcu1500_dynamic_5_0-0 was 126.690	0% of h	iost d
✓ UNUSED_CUS #1			Design Guidance Report - Mozilla Firefox (on xsjrdevl118)		- 0
KERNEL_READ_TRANSFER_AMOUNT_MIN #1	E Vivado Design × Design Guidance × +				
✓ HOST_MIGRATE_MEM #1	♠ (♦) ③ file:	///proj/xsjhd	staff1/herver/_SDAccel/bash/20182/gui_test2/prj1/src/ @ Q. Search	+ 0	0 =
✓ CU_UTIL #2					
✓ DEVICE_UTIL #1			Design Guidance Report		
		Convright C	nurinht 1088-2018 Villov, Inc. All Dichte Basarvard		
Decise Foodbadk	т т	ool Version v.	(lin64) Build 0		
Design Feedback	Date Mon May 7 18:26:20 2018				
	Host xsjrdevl118 running 64-bit Red Hat Enterprise Linux Workstation release 6.5 (Santiago)				
		command x	cccompile krnl_vadd.clplatform xilinx_kcu1500_dynamic_5_0xp param:compiler.useSdxRuleService=true -s		
	Table of Contents				
	2 VIOLATION DETAILS				
	1 REPORT SUMMARY				
	Violations found: 1				
	Rule Specs Violated: 1				
	2 VIOLATION DETAILS				

HTML Report

Visit www.xilinx.com/sdaccel to Get Started !



Learn and practice how to accelerate applications with FPGAs

© Copyright 2018 Xilinx



Summary - "Different" doesn't mean "Difficult"

> FPGA "Software Programmability" is now a reality

>> Develop FPGA-accelerated applications entirely in C/C++

> FPGAs excels at high-throughput streaming applications

>> Think "parallel", adopt "dataflow" programming model

> Off-the-shelf platforms make it easy to start and deploy at scale

>> No need to develop custom card, hardware interfaces or software stacks



Adaptable. Intelligent.



© Copyright 2018 Xilinx