FPGAs in the cloud?

Julien Simon, Principal Evangelist, AI/ML
@julsimon

Velocity Conference, NYC, 04/10/2017
Agenda

• The case for non-CPU architectures
• What is an FPGA?
• Using FPGAs on AWS
• Demo: running an FPGA image on AWS
• FPGAs and Deep Learning
• Resources
The case for non-CPU architectures
Powering AWS instances: Intel Xeon E7 v4

- **7.1 billion** transistors
  - 456 mm\(^2\) (0.7 square inch)
- **General-purpose architecture**
  - SISD with SIMD extension (AVX instruction set)
- **Best single-core performance**
- **Low parallelism**
  - 24 cores, 48 hyperthreads
  - Multi-threaded applications are hard to build
  - OS and libraries need to be thread-friendly
- **Thermal envelope**: 168W

[https://ark.intel.com/products/96900/Intel-Xeon-Processor-E7-8894-v4-60M-Cache-2_40-GHz](https://ark.intel.com/products/96900/Intel-Xeon-Processor-E7-8894-v4-60M-Cache-2_40-GHz)
Case study: Clemenson University

1.1 million vCPUs for Natural Language Processing

Optimized cost thanks to Spot Instances

Moore’s winter is (probably) coming

- "I guess I see Moore’s Law dying here in the next decade or so, but that’s not surprising", Gordon Moore, 2015
- Technology limits: a Skylake transistor is around 100 atoms across
- New workloads require higher parallelism to achieve good performance
  - Genomics
  - Financial computing
  - Image and video processing
  - Deep Learning
- The age of the GPU has come

http://www.economist.com/technology-quarterly/2016-03-12/after-moores-law
https://spectrum.ieee.org/computing/hardware/gordon-moore-the-man-whose-name-means-progress
State of the art GPU: Nvidia V100

- **21.1 billion** transistors
  - 815 mm$^2$ (1.36 square inch)
- Architecture optimized for floating point
  - SIMT (Single Instruction, Multiple Threads)
- Massive parallelism
  - 5120 CUDA cores, 640 Tensor cores
  - CUDA programming model
  - Large, high-bandwidth off-chip memory (DRAM)
- Thermal envelope: **250W**

https://devblogs.nvidia.com/parallelforall/inside-volta/
GPUs are not optimal for some applications

- Power consumption and efficiency (TOPS/Watt)
- Strict latency requirements
- Other requirements
  - Custom data types, irregular parallelism, divergence

- Building your own ASIC may solve this, but:
  - It’s a huge, costly and risky effort
  - ASICs can’t be reconfigured

- Time for an FPGA renaissance?
What’s an FPGA?
The FPGA

- First commercial product by Xilinx in 1985
- Field Programmable Gate Array
- Not a CPU (although you could build one with it)
- «Lego» hardware: logic cells, lookup tables, DSP, I/O
- Small amount of very fast on-chip memory
- Build custom logic to accelerate your SW application
FGPA architecture

Sources:
https://www.embedded-vision.com/industry-analysis/technical-articles/fpgas-deep-learning-based-vision-processing
http://www.bober-optosensorik.de/fpga-entwicklung.html
Developing FPGA applications

- **Languages**
  - VHDL, Verilog
  - OpenCL (C++)

- **Software tools**
  - Design
  - Simulation
  - Synthesis
  - Routing

- **Hardware tools**
  - Evaluation boards
  - Prototypes

Expensive and hard to scale
Using FPGAs on AWS
Amazon EC2 F1 Instances

- Up to 8 Xilinx UltraScale Plus VU9P FPGAs
- Each FPGA includes
  - Local 64 GB DDR4 ECC protected memory
  - Dedicated PCIe x16 connections
  - Up to 400Gbps bidirectional ring connection for high-speed streaming
  - Approximately 2.5 million logic elements, and approximately 6,800 DSP engines
The FPGA Developer Amazon Machine Image (AMI)

• Xilinx SDx 2017.1
  – Free license for F1 FPGA development
  – Supports VHDL, Verilog, OpenCL

• AWS FPGA SDK
  – Amazon FPGA Image (AFI) Management Tools
  – Linux drivers
  – Command line

• AWS FPGA HDK
  – Design files and scripts required to build an AFI
  – Shell: platform logic to handle external peripherals, PCIe, DRAM, and interrupts

• Run simulation, design, etc. on a C4 to save money!

https://aws.amazon.com/marketplace/pp/B06VVYBLZZ
https://github.com/aws/aws-fpga
FPGA Acceleration Using F1 instances

- Amazon Machine Image (AMI)
- Amazon FPGA Image (AFI)
- AWS Marketplace
- CPU
- PCIe
- DDR-4 Attached Memory
- DDR Controllers
- FPGA Link
Case study: Edico Genome

Highly Efficient

- Algorithms Implemented in Hardware
- Gate-Level Circuit Design
- No Instruction Set Overhead

Massively Parallel

- Massively Parallel Circuits
- Multiple Compute Engines
- Rapid FPGA Reconfigurability

Speeds Analysis of Whole Human Genomes from Hours to Minutes
Unprecedented Low Cost for Compute and Compressed Storage

http://www.edicogenome.com/
https://aws.amazon.com/marketplace/pp/B075JR57J1
Case study: NGCodec

- Provider of **UHD video compression** technology
- Up to 50x faster vs. software H.265
- **Higher quality video** than x265 ‘veryslow’ preset
  - Same bit rate
  - 60+ frames per second
- **Lower latency** between live stream and end viewing
- **Optimized cost**

https://ngcodec.com/markets-cloud-transcoding/
https://aws.amazon.com/marketplace/pp/B074W1FPKR
Demo: OpenCL on F1 instance
Building the OpenCL application

git clone https://github.com/aws/aws-fpga.git
cd aws-fpga
source sdk_setup.sh
source hdk_setup.sh
source sdaccel_setup.sh
source $XILINX_SDX/settings64.sh
cd $SDACCEL_DIR/examples/xilinux/getting_started/host/helloworld_ocl/
make clean
make check TARGETS=sw_emu DEVICES=$AWS_PLATFORM all
make check TARGETS=hw_emu DEVICES=$AWS_PLATFORM all
make check TARGETS=hw DEVICES=$AWS_PLATFORM all
Creating Vivado project and starting FPGA synthesis

INFO: [XOCC 60-586] Created xclbin/vector_addition.hw.xilinx_aws-vu9p-f1_4drr-xpr-2pr_4_0.xclbin
Total elapsed time: 2h 31m 7s

$(SDACCEL_DIR)/tools/create_sdaccel_afi.sh -xclbin=xclbin/vector_addition.hw.xilinx_aws-vu9p-f1_4drr-xpr-2pr_4_0.xclbin -o=vector_addition.hw.xilinx_aws-vu9p-f1_4drr-xpr-2pr_4_0 -s3_bucket=jsimon-fpga -s3_logs_key=logs -s3_dcp_key=dcp

... Generated manifest file '17_10_02-163912_manifest.txt'
upload: ./17_10_02-163912_Developer_SDAccel_Kernel.tar to s3://jsimon-fpga/dcp/17_10_02-163912_Developer_SDAccel_Kernel.tar17_10_02-163912_agfi_id.txt
Building the AFI

```bash
aws ec2 describe-fpga-images --fpga-image-id afi-056fb17d8cedf37
{
  "FpgaImages": [{
    "UpdateTime": "2017-10-02T16:39:17.000Z",
    "Name": "xclbin/vector_addition.hw.xilinx_aws-vu9p-f1_4ddr-xpr-2pr_4_0.xclbin",
    "FpgaImageGlobalId": "agfi-03a8031774fc4773f",
    "Public": false,
    "State": { "Code": "Pending" },
    "OwnerId": "6XXXXXXXXXXX",
    "FpgaImageId": "afi-056fb17d8cedf37",
    "CreateTime": "2017-10-02T16:39:17.000Z",
    "Description": "xclbin/vector_addition.hw.xilinx_aws-vu9p-f1_4ddr-xpr-2pr_4_0.xclbin"
  }
}
```
Loading the AFI and running the OpenCL application

```
aws ec2 describe-fpga-images --fpga-image-id afi-056fb17dd8cedf37
{
   "FpgaImages": [{
   "UpdateTime": "2017-10-02T16:39:17.000Z",
   "Name": "xclbin/vector_addition.hw.xilinx_aws-vu9p-f1_4ddr-xpr-2pr_4_0.xclbin",
   "FpgaImageGlobalId": "agfi-03a8031774fc4773f",
   "Public": false,
   "State": { "Code": "ready"},
   "OwnerId": "6XXXXXXXXXXX",
   "FpgaImageId": "afi-056fb17dd8cedf37",
   "CreateTime": "2017-10-02T16:39:17.000Z",
   "Description": "xclbin/vector_addition.hw.xilinx_aws-vu9p-f1_4ddr-xpr-2pr_4_0.xclbin"
   }
}
```

```
sudo fpga-load-local-image -S 0 -I agfi-03a8031774fc4773f
sudo fpga-describe-local-image -S 0

sudo sh
source /opt/Xilinx/SDx/2017.1.rte/setup.sh
./helloworld

sudo fpga-clear-local-image -S 0
```
FPGAs and Deep Learning
A chink in the GPU armor?

- GPUs are great for training, but what about inference?

- Throughput and latency: pick one?
  - Using batches increases latency
  - Using single samples degrades throughput

- Power and memory requirements
  - Floating-point operations are power-hungry
  - Floating-point weights need more DRAM, which is power-hungry too

- Neural networks can be implemented on FPGA
Using custom logic to Multiply and Accumulate

Source: « FPGA Implementations of Neural Networks », Springer, 2006

Smaller weights $\Rightarrow$ less gates, less data to load into the FPGA
Optimizing Deep Learning models for FPGAs

• **Quantization**: using integer weights
  – 8/4/2-bit integers instead of 32-bit floats
  – Reduces power consumption
  – Simplifies the logic needed to implement the model
  – Reduces memory usage

• **Pruning**: removing useless connections
  – Increases computation speed
  – Reduces memory usage

• **Compression**: encoding weights
  – Reduces model size

On-chip SRAM becomes a viable option

→ More power-efficient than DRAM

→ Faster than off-chip DRAM
Published results

[Han, 2016] Optimizing CNNs on CPU and GPU
• AlexNet 35x smaller, VGG-16 49x smaller
• 3x to 4x speedup, 3x to 7x more energy-efficient
• No loss of accuracy

[Han, 2017] Optimizing LSTM on Xilinx FPGA
• FPGA vs CPU: 43x faster, 40x more energy-efficient
• FPGA vs GPU: 3x faster, 11.5x more energy-efficient

[Nurvitadhi, 2017] Optimizing CNNs on Intel FPGA
• FPGA vs GPU: 60% faster, 2.3x more energy-efficient
• <1% loss of accuracy
Nvidia Hardware for Deep Learning

• Open architecture for **DL inference accelerators** on IoT devices
  – Convolution Core – optimized high-performance convolution engine
  – Single Data Processor – single-point lookup engine for activation functions
  – Planar Data Processor – planar averaging engine for pooling
  – Channel Data Processor – multi-channel averaging engine for normalization functions
  – Dedicated Memory and Data Reshape Engines – memory-to-memory transformation acceleration for tensor reshape and copy operations.

• Verilog model + test suite

• F1 instances are supported

http://nvdl.org/
https://github.com/nvdl
Conclusion

• CPU, GPU, FPGA: the battle rages on

• As always, pick the right tool for the job
  – Application requirements: performance, power, cost, etc.
  – Time to market
  – Skills
  – The AWS marketplace: the solution may be just a few clicks away!

• AWS offers you many options, please explore them and give us feedback
Resources

https://aws.amazon.com/ec2-instance-types/f1
https://aws.amazon.com/ec2-instance-types/f1/partners/
https://github.com/aws/aws-fpga

[Han, 2016] « Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding » https://arxiv.org/abs/1510.00149


Thank you!

http://aws.amazon.com/evangelists/julien-simon
@julsimon