Specializing FGPU for Persistent Deep Learning

Rui Ma, Alex Hsu, Tian Tan (The University of Texas at Austin)
Eriko Nurvitadhi, David Sheffield, Aravind Dasu, Rob Pelt,
Martin Langhammer, Jaewoong Sim (Intel)
Derek Chiou (Microsoft / The University of Texas at Austin)
Time-to-Solution

• Time-to-Solution is an important performance metric
  • Includes everything to get all (one to many) needed results
    • E.g., design, implementation, validation, manufacturing, deployment, compilation, and running times
  • Time-to-Solution includes different components depending on approach
    • E.g., software does not include processor development
    • E.g., ASIC includes silicon design and implementation
  • Only if many runs are performed, development time is amortized

• Much of the published work focuses only on kernel run time

• Amdahl's Law is applicable to the total solution
FPGAs High Perf, Slow Development

• Modern FPGAs can achieve industry leading performance [1]
  • Requires high specialization

• Highly-specialized solutions often require long development time
  • Time-to-Solution may be longer than a fast-to-develop even though slower-when-run solution

• Fast dev, reasonable perf solutions used until specialized solution is available
  • May make optimal performance solution unnecessary

[1] Chung, et al. Serving DNNs in Real Rime at Datacenter Scale with Project Brainwave
Solution: Specialized Overlays

<table>
<thead>
<tr>
<th>General purpose?</th>
<th>No</th>
<th>No</th>
<th>Yes</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>Max</td>
<td>High / Max</td>
<td>Low / Medium</td>
<td>Good</td>
</tr>
<tr>
<td>Hardware expertise?</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Development time</td>
<td>Weeks - Month</td>
<td>Days - Weeks</td>
<td>Hours - Days</td>
<td>Hours – Days</td>
</tr>
<tr>
<td>Compile time</td>
<td>Hours - Days</td>
<td>Hours - Days</td>
<td>Seconds</td>
<td>Seconds</td>
</tr>
</tbody>
</table>

Outline

• Time-to-Solution

• PDL-FGPU Architecture and Case Study Workload

• Results

• On-Going Work and Conclusion
Approach

• Start with FGPU [2]
  • Open-source soft GPU programmed with OpenCL-based toolchain

• Specialize FGPU for Persistent RNNs to improve performance

• Target Intel Stratix 10 GX 2800
  • 933,120 ALMs
  • 5,760 DSPs (9.2 FP32 TFLOPS)
  • 11,721 M20Ks (117.2 TB/s BW)
  • 1 GHz

Architecture
Architecture

Specialized Macro: Dot
dot acc, vec, shr_ptr, shr_off

Specialized Scalar: Act
sigmoid dest, src
tanh dest, src
relu dest, src
Persistent RNN Algorithm
Persistent RNN Data Placement
Outline

• Time-to-Solution
• PDL-FGPU Architecture and Case Study Workload
• Results
• On-Going Work and Conclusion
## Case Study Workloads

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Precision</th>
<th>Matrix Size</th>
<th>Vector Size</th>
<th>Iters.</th>
<th>Batch</th>
<th>Lines of Code</th>
<th>Engr. Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN (skip input)</td>
<td>FP32</td>
<td>1024x1024</td>
<td>1024</td>
<td>256</td>
<td>1</td>
<td>82</td>
<td>Few hrs</td>
</tr>
<tr>
<td>RNN (skip input)</td>
<td>INT8</td>
<td>2048x2048</td>
<td>2048</td>
<td>256</td>
<td>1</td>
<td>75</td>
<td>Few hrs</td>
</tr>
<tr>
<td>RNN (skip input)</td>
<td>INT4</td>
<td>4096x4096</td>
<td>4096</td>
<td>256</td>
<td>1</td>
<td>81</td>
<td>Few hrs</td>
</tr>
<tr>
<td>RNN (linear input)</td>
<td>FP32</td>
<td>1024x1024</td>
<td>1024</td>
<td>256</td>
<td>1</td>
<td>93</td>
<td>Few hrs</td>
</tr>
<tr>
<td>LSTM</td>
<td>FP32</td>
<td>512x512</td>
<td>512</td>
<td>256</td>
<td>1</td>
<td>157</td>
<td>&lt; 1 day</td>
</tr>
<tr>
<td>GRU</td>
<td>FP32</td>
<td>512x512</td>
<td>512</td>
<td>256</td>
<td>1</td>
<td>139</td>
<td>&lt; 1 day</td>
</tr>
</tbody>
</table>
PDL-FGPU vs FGPU: Cycles

• One to three orders of magnitude performance improvement over baseline
  • 55-727x speedup in single precision and low-precision
• Major reasons for difference (85x total on skip input RNN FP32)
  • Vector dot product engine (36x)
  • Keeping weights on-chip (1.7x)
  • Better memory scheduling (1.3x)
  • Improved inter-thread communication (1.05x)
PDL-FGPU vs FGPU: Cycles—Non-PDL

- Generality maintained at close to the same performance
- Cycle reduction mostly due to memory controller scheduling
  - 6% fewer cycles on average
- Execution time increase due to reduced clock frequency
  - 15% slowdown on average
PDL-FGPU vs FGPU: ALM Utilization

- FP32 mode ~1.5x ALM consumption
  - Efficiently leveraged DSPs and on-chip RAM
- Low precision mode has higher ALM consumption
  - Low precision dot product functional units mapped into ALMs (at submission time)
  - Improved by packing into DSPs (in newer versions)

Note: Full FP32 configuration supports all single precision function units: fadd, fmul, fdiv, etc. Each unit can be disabled to save area/improve frequency but requires Quartus compilation.
PDL-FGPU vs V100: Execution Time

• 3-7x slower than Nvidia V100
  • For measured problems and sizes

• Performance gap factors
  • 5-6x slower frequency
    • ~280 MHz vs ~1500 MHz
  • Fewer floating-point units
    • More DSPs available on S10 than used

Note: cuDNN only supported FP32 kernels at submission time.
PDL-FGPU vs V100: Throughput Utilization

• PDL-FGPU is 2-3x higher in throughput utilization than Nvidia due to higher specialization

• Throughput utilization can be further improved by increasing FPGA resource utilization
Outline

• Time-to-Solution
• PDL-FGPU Architecture and Case Study Workload
• Results

• On-Going Work and Conclusion
On-Going Work

• Continue to optimize
  • Increase number of CUs
  • Increase frequency
  • Improve code generation

• Compare with other OpenCL, HLS, and overlay solutions

• Target other domains

• Improve usability
Conclusions

• Time-to-Solution is an important (but often overlooked) metric

• Using different implementations at different times can improve overall Time-to-Solution
  • Programmability speeds up development
    • Programmable solutions allows quick iteration for functional correctness
    • Domain-specific programmable solutions can minimize runtime
  • Highly-specialized solution maximizes performance once available

• Domain-specific programmable solutions provide higher performance
  • 55-727x speedup on persistent RNNs over baseline
  • Within a factor of 3-7x of Nvidia V100 on persistent RNNs at FP32
Thank you!
Backup Slides
Persistent RNN

• Recurrent neural networks are a class of deep learning networks that have layer(s) that feedback themselves

• Useful for sequential tasks such as speech recognition, text processing, and translation

• In persistent RNN, weights are kept in registers and activations are kept in shared memory
  • Leverages the large capacity and high bandwidth of SRAMs on modern FPGA
PDL-FGPU Architecture: Modifications

- Dot product vector instruction
  - Fused shared memory load, dot, and reduction operation
- Activation instructions
  - Reduces instruction pressure
- Synchronization instructions
  - Better inter-thread cooperation
- Conditional memory load/store instructions
  - if reg==0 then ld/st
  - Avoids control flow divergence

- Memory controller improvements
- High bandwidth register file with 1024-bit single-cycle registers
  - 128 bytes / cycle
- High bandwidth shared memory
  - 128 bytes / cycle
PDL-FGPU Configuration

• Hardware
  • 8 Compute Units per PDL-FGPU (16 in progress)
  • 8 Processing Elements per Compute Unit
  • 1024-bit wide operation (32 DSPs) per Processing Element

• Execution
  • 4096 threads in 64-wide SIMD
  • 16x1024-bit & 32x32-bit registers per thread
# Hardware Comparison Table

<table>
<thead>
<tr>
<th></th>
<th>Nvidia V100</th>
<th>S10-280</th>
<th>S10-210</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP32 throughput</td>
<td>15 TFLOPS</td>
<td>9.2 TFLOPS</td>
<td>6.3 TFLOPS</td>
</tr>
<tr>
<td>SRAM size</td>
<td>38 MB</td>
<td>30 MB</td>
<td>30 MB</td>
</tr>
<tr>
<td>SRAM bandwidth</td>
<td>145 TB/s</td>
<td>140 + 110 TB/s</td>
<td>65 + 80 TB/s</td>
</tr>
<tr>
<td>DRAM bandwidth</td>
<td>1 TB/s (HBM2*4)</td>
<td>64 GB/s (DDR4*4)</td>
<td>0.5 TB/s (HBM2*2)</td>
</tr>
<tr>
<td>Frequency</td>
<td>1.4 GHz / 1.67 GHz</td>
<td>1 GHz</td>
<td>1 GHz</td>
</tr>
<tr>
<td>I/O</td>
<td>300 GB/s (NVLink)</td>
<td>240 GB/s</td>
<td>240 GB/s</td>
</tr>
<tr>
<td>Power</td>
<td>345W</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
## PDL-FGPU vs FGPU: Resource Utilization

<table>
<thead>
<tr>
<th>Config</th>
<th>ALM</th>
<th>RAM</th>
<th>DSP</th>
<th>Min Freq (MHz)</th>
<th>Max Freq (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FGPU</td>
<td>PDL</td>
<td>FGPU</td>
<td>PDL</td>
<td>FGPU</td>
</tr>
<tr>
<td>FP32*</td>
<td>329226</td>
<td>494619</td>
<td>1318</td>
<td>5790</td>
<td>768</td>
</tr>
<tr>
<td>INT8</td>
<td>239714</td>
<td>726823</td>
<td>742</td>
<td>4766</td>
<td>128</td>
</tr>
<tr>
<td>INT4</td>
<td>239714</td>
<td>589425</td>
<td>742</td>
<td>4766</td>
<td>128</td>
</tr>
</tbody>
</table>

Note: The full FP32 configuration supports all single precision function units: fadd, fmul, fdiv, etc. The design allows any unit to be selectively disabled to save area/improve frequency but requires another full Quartus compilation.
## PDL-FGPU vs FGPU: Resource Util Breakdown

<table>
<thead>
<tr>
<th>FGPU baseline for LSTM / GRU</th>
<th>Global</th>
<th>Per CU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>global memory controller</td>
<td>workgroup dispatcher</td>
</tr>
<tr>
<td>ALM</td>
<td>39253</td>
<td>930.3</td>
</tr>
<tr>
<td>RAM</td>
<td>53</td>
<td>8</td>
</tr>
<tr>
<td>DSP</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PDL-FGPU for LSTM / GRU</th>
<th>Global</th>
<th>Per CU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>global memory controller</td>
<td>workgroup dispatcher</td>
</tr>
<tr>
<td>ALM</td>
<td>47510</td>
<td>885</td>
</tr>
<tr>
<td>RAM</td>
<td>61</td>
<td>8</td>
</tr>
<tr>
<td>DSP</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Feature-wise Speedup: FP32 RNN (Skip Input)

- Domain-specific macro unit (e.g. dot unit) provides the most performance improvement
Even More Backup Slides
FPGU vs PDL-FGPU: ALMs

- Most configurations ~1.5x ALM consumption
  - Efficiently leverage DSPs and on-chip RAM
- Low precision mode has higher ALM consumption
  - Currently low precision dot function units are mapped into ALMs and could be improved by packing them into DSPs
  - Fixed in new versions
FPGU vs PDL-FGPU: M20ks

- ~5x M20ks consumption
  - Vector register file
  - Shared memory
  - Other microarchitectural changes to better leverage on chip RAM
FPGU vs PDL-FGPU: DSPs

- FP32 configuration ~4.6x DSPs consumption
  - Dot product unit
  - Activation function unit
## Configurable FP32 Function Units

<table>
<thead>
<tr>
<th>Function unit</th>
<th>Description</th>
<th>Function unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FADD</td>
<td>Addition</td>
<td>FFMA</td>
<td>Multiplication and Accumulation</td>
</tr>
<tr>
<td>FMUL</td>
<td>Multiplication</td>
<td>SIGMOID</td>
<td>Sigmoid function</td>
</tr>
<tr>
<td>FDIV</td>
<td>Division</td>
<td>TANH</td>
<td>Tanh function</td>
</tr>
<tr>
<td>FSQRT</td>
<td>Square Root</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRSQRT</td>
<td>Inverse square root</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UITOFP</td>
<td>Cast unsigned INT to FP32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSLT</td>
<td>Comparison, less than</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Performance Evaluation Assumptions

• Exclude
  • host-side compute or data transfers (roughly the same between FPGA/GPU)
  • initialization effects
    • FGPU/PDL-FGPU: ~500 cycles of CU initialization per kernel
    • GPU: one-time JIT compilation of the application

• Nvidia’s terminology is used
  • Skip input RNN assumes the biased input weight activation multiply is precomputed, and thus only 1 GEMV is computed per input per iteration
  • Linear input RNN means both the input and hidden computation are computed
how much time does this take? Point is to say that they are roughly the same
Derek Chiou, 8/31/2019
FGPU vs PDL-FGPU: Dynamic Instruction Count

- 30-1342x less instructions than base line
  - Domain-specific instructions reduce instruction pressure
FPGA vs GPU Capabilities

• Flexible precision
  • Densely packed computational resources (Intel)
    • 5760 DSPs on Stratix 10 yield 7 TFLOPS, or 28 TOPS of INT8 arithmetic at 600 MHz
    • 15 TFLOPS on V100, 130 TOPS of INT8 on V100 tensor core

• On-chip memory bandwidth
  • 70 TB/s from M20Ks on Stratix 10 (excluding MLABs)
  • 140 TB/s from register files and shared memories on V100
PDL-FGPU Architecture: Chip
PDL-FGPU Architecture: Compute Unit
PDL-FGPU Architecture: Processing Element
PDL-FGPU Estimated Resource Usage

- **DSPs**
  - 4,480 DSPs (35 per processing element)
  - 78% utilization on Stratix 10 (280)

- **M20Ks**
  - ~10,000 M20Ks (~8000 in the vector regfiles and ~500 in the shared memory)
  - ~85% utilization on Stratix 10 (280)

- **ALMs**
  - ~700,000 ALMs
  - ~75% utilization on Stratix 10 (280)
PDL-FGPU Estimated Performance

- **INT8**
  - \(16 \text{ CUs} \times 8 \text{ vector } \text{ops} \times 128 \frac{\text{MACCs}}{\text{vector op}} \times 2 \frac{\text{ops}}{\text{MACC}} \times 500 \text{ MHz} \times 50\% \frac{\text{instrs}}{\text{kernel}}\)
  - = 8 INT8 TOPs

- **FP32**
  - \(16 \text{ CUs} \times 8 \text{ vector } \text{ops} \times 32 \frac{\text{FFMAs}}{\text{vector op}} \times 2 \frac{\text{ops}}{\text{FFMA}} \times 500 \text{ MHz} \times 50\% \frac{\text{instrs}}{\text{kernel}}\)
  - = 2 FP32 TOPs
Deep Learning Dataflow: RNN
Deep Learning Dataflow: LSTM
Deep Learning Dataflow: GRU