OpenCL for FPGAs/HPC
Case Study in 3D FFT

Ahmed Sanaullah  Martin Herbordt  Vipin Sachdeva

Boston University  Silicon Therapeutics
What gives FPGAs high performance?

► Deep pipelines
► Block RAMs
► Flexible on-chip communication/networks
► High utilization
What gives FPGAs high performance?

► Deep pipelines
► Block RAMs
► Flexible on-chip communication/networks
► High utilization

To sum it up…

Application Specific Architecture
What gives FPGAs high performance?

► Deep pipelines
► Block RAMs
► Flexible on-chip communication/networks
► High utilization

To sum it up…

Application Specific Architecture

But creating these designs in HDL is very complex
How do we solve the programmability problem?
IP Cores

- 3rd party solutions
- Highly optimized
- Ease of use
- Reduces implementation timeframes
IP Cores

- 3rd party solutions
- Highly optimized
- Ease of use
- Reduces implementation timeframes

But …

- Limited customizability
- Implementation specifics hidden to protect intellectual property
IP Cores

- 3rd party solutions
- Highly optimized
- Ease of use
- Reduces implementation timeframes

But …

- Limited customizability
- Implementation specifics hidden to protect intellectual property

Which means…

Pseudo Application Specific Architecture
How about OpenCL?

- Develop application in C99 and compile to hardware
- Primitives and pragmas
  - further customize hardware translations
  - e.g. loop unroll, compute unit replication, single/multiple work item

Doesn’t OpenCL generate a complete .aocx file?

- Do not have to complete compilation
- Can obtain generated HDL from kernel_system folder
- Isolate and integrate required modules into existing design
Case Study

3D FFT
3D FFT Compute Units

OpenCL Radix-2

1D Vector → Stage 1 → Stage 2 → Stage log(N) → 1D Vector

IP Core Radix-4/2

1D Vector → FFT IP Core 1 → 1D Vector → FFT IP Core 2 → 1D Vector → FFT IP Core 3 → 1D Vector → FFT IP Core 4 → 1D Vector → FFT IP Core N

Individual Complex Values
FPGA: Altera Arria 10-X115

- 427K ALMs
- 1518 DSP blocks
- 53Mb BRAMs

FFT Size: $64^3$

Throughput Constraint: 64

- Mix of ALMs and DSPs used for FFT IP cores
  - Insufficient DSP resources
  - DSPs preferred over ALMs
Resource and Performance Comparison

<table>
<thead>
<tr>
<th>Design</th>
<th>ALM</th>
<th>DSP</th>
<th>BRAM</th>
<th>Freq.(MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenCL</td>
<td>22K</td>
<td>1,280</td>
<td>33Kb</td>
<td>400</td>
</tr>
<tr>
<td>IP Core</td>
<td>181K</td>
<td>1,412</td>
<td>810Kb</td>
<td>208</td>
</tr>
</tbody>
</table>

- OpenCL FFT has:
  - ≈ 10x fewer ALMs usage
  - ≈ 25x less on-chip memory usage
  - ≈ 2x higher frequency
- OpenCL FFT can meet the required throughput using DSPs only
Conclusion

- OpenCL based designs can perform better than IP core based one
  - For $64^3$ FFT

- FFT IP cores are constrained to a specific computational flow
  - May not be optimal for all FFT sizes

- OpenCL enables more application specific designs
  - with less effort than HDL programming
Memory Architecture

- Ping-pong Primary Memory buffers
- Primary Memory Bank: $O(N^2)$ complexity (single read, single write)
- Secondary Memory Bank: $O(N)$ complexity (single read, parallel write)
- Transpose
  - Outputs of Compute Unit write to the same Secondary Memory Bank
  - Secondary Memory Banks write to Primary Memory Banks
- New writes to Secondary Memory Bank every N cycles
Can this design source and sink data stall-free?

\[
\text{Index}_{3D} = \text{Buffer}# \times N^2 + \text{Offset} \times N + \text{Loc}
\]

<table>
<thead>
<tr>
<th>IP Core</th>
<th>Loc</th>
<th>Offset</th>
<th>Buffer #</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFTx</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>FFTy</td>
<td>Y</td>
<td>Z</td>
<td>X</td>
</tr>
<tr>
<td>FFTz</td>
<td>Z</td>
<td>X</td>
<td>Y</td>
</tr>
</tbody>
</table>

- **Buffer#** varies for a given cycle
- **Loc** changes every cycle
- **Offset** changes every N cycles
- **Buffer#** → **Offset** for next FFT dimension
Can this design source and sink data stall-free?

\[ \text{Index}_3D = Buffer\# \times N^2 + Offset \times N + Loc \]

**IP Core**

<table>
<thead>
<tr>
<th></th>
<th>Loc</th>
<th>Offset</th>
<th>Buffer #</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFTx</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>FFTy</td>
<td>Y</td>
<td>Z</td>
<td>X</td>
</tr>
<tr>
<td>FFTz</td>
<td>Z</td>
<td>X</td>
<td>Y</td>
</tr>
</tbody>
</table>

**OpenCL Radix-2**

<table>
<thead>
<tr>
<th></th>
<th>Loc</th>
<th>Offset</th>
<th>Buffer #</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFTx</td>
<td>Y</td>
<td>Z</td>
<td>X</td>
</tr>
<tr>
<td>FFTy</td>
<td>Z</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>FFTz</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
</tbody>
</table>

- **Buffer#** varies for a given cycle
- **Loc** changes every cycle
- **Offset** changes every N cycles
- **Buffer#** → **Offset** for next FFT dimension
- Only difference is in initial data locations
- Hence, no stalls