Hardware Acceleration of Pulsar Search on FPGAs using OpenCL

Oliver Sinnen    Haomiao Wang
&
Prabu Thiagaraj (Manchester Uni)

Parallel and Reconfigurable Computing

Department of Electrical and Computer Engineering
University of Auckland

Computing for SKA, 2017
Strong-field Test of Gravity using Pulsars

Image credit: NASA/Tod Strohmayer (GSFC)/Dana Berry (Chandra X-Ray Observatory)
Outline

1. Overview and Task
2. FT Convolution Decomposition
3. High-level Techniques and Implementation
4. Evaluation
5. What’s Next
Outline

1. Overview and Task
2. FT Convolution Decomposition
3. High-level Techniques and Implementation
4. Evaluation
5. What’s Next

Oliver Sinnen, Haomiao Wang & Prabu Thiagaraj (Manchester Uni)
- Observed radiation is a pulse Binary pulsar (Doppler effect)
- Acceleration search: 1) Time-domain 2) Frequency-domain
Frequency-domain

- Using **matched filtering** technique in Fourier domain to recover the signal into single bin.

\[
A_{r_0} \propto \sum_{k=[r_0]-m/2}^{[r_0]+m/2} A_k A^*_{r_0-k},
\]

where frequency \( r_0 \) is unknown.

- Summation is computed at a range of frequencies \( r \).
Block Overview of Pulsar Search Engine
Fourier-domain Acceleration Search (FDAS)

FDAS module is applied to search for (binary) pulsars with constant frequency derivatives in frequency-domain.

Over 2,000 beams are formed at 4,096 channels/beam.

Beam signals are de-dispersed for 6,000 DMs.

PSS Engine \(_i\)
- Single Pulse Search Modules
- Time Domain Acceleration or Harmonic-sum Module

FDAS Module
- FT Convolution Module
- 85 FIR filters, maximum length is 421-tap

Pre-Processing
- RFIM
- DDTR
- PSBC
- CXFT
- BRDZ
- DRED

Post-processing
## Specification of Task

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B$</td>
<td># of beams</td>
<td>1000 ~ 2000</td>
</tr>
<tr>
<td>$DM$</td>
<td># of de-dispersion measure (DM) trails</td>
<td>6000</td>
</tr>
<tr>
<td>$T_{obs}$</td>
<td>Observation period</td>
<td>540s</td>
</tr>
<tr>
<td>$t_{limit}$</td>
<td>Time of executing one sample group</td>
<td>88ms</td>
</tr>
<tr>
<td>$N$</td>
<td># of complex samples per group</td>
<td>$2^{22}$</td>
</tr>
<tr>
<td>$M$</td>
<td># of templates/filter</td>
<td>85</td>
</tr>
<tr>
<td>$K$</td>
<td># of average template/filter length</td>
<td>&gt; 200</td>
</tr>
</tbody>
</table>
Outline

1. Overview and Task
2. FT Convolution Decomposition
3. High-level Techniques and Implementation
4. Evaluation
5. What’s Next
FT Convolution

- Complex floating-point operations
- Multiple long FIR filters
- Large input size
- Strict time limit

Number of acceleration devices (CapEx)

Energy consumption (OpEx)
Basic Element

- **Time-domain FIR Filter (TDFIR)**
  
  \[
  y_m[i] = \sum_{k=0}^{K-1} x_m[i - k] h_m[k], \text{ for } i = 0, 1, \ldots, N - 1
  \]

- **Frequency-domain FIR Filter (FDFIR)**
  
  \[
  \mathcal{F}\{f \ast h\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{h\}
  \]
Hardware Limitation

- **Naïve Time Domain**
  - DSP
    - block Single precision floating-point (SPF) multiplications
      \[(A + iB) \times (C + iD) = (A \times C - B \times D) + i(A \times D + B \times C)\]

- **Naïve Frequency Domain**
  - Off-chip (global) memory
  - Off-chip memory bandwidth
  - RAM block
    - On-chip (local) memory size
    - 4-Million elements = 32MBytes
Decomposition Algorithms

Overlap-add Algorithm
- Split the coefficient array
  $\rightarrow$ OLA-TD

Overlap-save Algorithm
- Split the input array
  $\rightarrow$ OLS-FD

Oliver Sinnen, Haomiao Wang & Prabu Thiagaraj (Manchester Uni)
Outline

1. Overview and Task
2. FT Convolution Decomposition
3. High-level Techniques and Implementation
4. Evaluation
5. What’s Next
High-level Techniques and Implementation

- Maxeler MaxCompiler using Java to develop FPGA (*HPCC2016*)
- Open Computing Language (**OpenCL**) for FPGAs (**Intel FPGA Cards**), GPUs, and CPUs (**FPT2016, best paper candidate**)

Diagram showing a diagram of memory and interconnects, including DDR Controller & PHY, Global Memory Interconnect, Local Memory Interconnect, PCIe, Kernel Pipeline, Block RAM, Memory (DDR3 and SSD).
Kernel Structures–OLA

Naïve-TD

OLA-TD
Kernel Structures–OLS

AOLS

Input

1st launch

Global Memory
1st launch: Bank1
2nd launch: Bank2

Data Fetch and Multiplication Kernel (NDRange)

FFT and Multiplication Kernel (Single)

Channels

FFT Data Fetch Kernel (NDRange)

Processed coefficients

2nd launch

Output

FFT/IFFT Kernel (Single)

1st launch FFT
2nd launch IFFT

Switch

Global Memory 1st launch: Bank2
2nd launch: Bank1

Bit-Reverse Kernel (NDRange)

Channels

TOLS

Input

Global Memory (Bank1)

FFT Data Fetch Kernel (NDRange)

Channels

Processed coefficients

Output

Global Memory (Bank2)

IFFT Bit-Reversed Kernel (NDRange)

Channels

Oliver Sinnen, Haomiao Wang & Prabu Thiagaraj (Manchester Uni)
FDAS on FPGA using OpenCL
Outline

1. Overview and Task
2. FT Convolution Decomposition
3. High-level Techniques and Implementation
4. Evaluation
5. What’s Next
### Platform

**Table: Details of FPGA and GPU Platforms**

<table>
<thead>
<tr>
<th>Device (Board)</th>
<th>Terasic DE5-Net</th>
<th>Sapphire Nitro R7 370</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>Intel Stratix V 5SGXA7</td>
<td>AMD Radeon R7 370</td>
</tr>
<tr>
<td>Technology</td>
<td>28nm</td>
<td>28nm</td>
</tr>
<tr>
<td>Compute resource</td>
<td>622,000 LEs</td>
<td>1024 Stream Processors</td>
</tr>
<tr>
<td></td>
<td>256 DSP blocks</td>
<td></td>
</tr>
<tr>
<td>On-chip memory size</td>
<td>50Mb</td>
<td>—</td>
</tr>
<tr>
<td>Global memory size</td>
<td>2 x 2GB DDR3</td>
<td>3GB GDDR5</td>
</tr>
<tr>
<td>Global memory frequency</td>
<td>800MHz</td>
<td>5,600MHz</td>
</tr>
<tr>
<td>Memory interface width</td>
<td>2 x 64-bit</td>
<td>256-bit</td>
</tr>
<tr>
<td>Max clock frequency</td>
<td>—</td>
<td>985MHz</td>
</tr>
<tr>
<td>Max power consumption</td>
<td>1.0</td>
<td>1.2</td>
</tr>
<tr>
<td>Max power consumption</td>
<td>—</td>
<td>150W</td>
</tr>
</tbody>
</table>
Latency – TDFIR vs FDFIR

4 TDFIR Kernels

- Naïve
  - TD-Naïve-64S
  - TD-Naïve-64N
- OLA
  - OLA-64S
  - OLA-64N

5 FDFIR Kernels

- Naïve
- FD-Naïve
- OLS
- AOLS
  - AOLS-1024
  - AOLS-2048
  - AOLS-4096
- TOLS
  - TOLS-1024
Latencies of a single FPGA (Intel Stratix V A7) in processing same input array using 9 different OpenCL kernels:
Multiple FIR Filters

- Even fastest kernel cannot meet time limit
  
  => Implement multiple FIR filters in parallel

Problem:
Bandwidth of off-chip memory is main problem

Solution:
- Do more processing!
- Calculate power of complex values (need input in next stage)

Problem:
Number of DSP blocks limits number of parallelisable filters

Solution:
- Downscale the FFT engine input size: 8 points→ 4 points
Multiple FIR Filters

- Even fastest kernel cannot meet time limit
  
  => Implement multiple FIR filters in parallel

Problem:
Bandwidth of off-chip memory is main problem

Solution:
- Do more processing!
- Calculate power of complex values (need input in next stage)

Problem:
Number of DSP blocks limits number of parallelisable filters

Solution:
- Downscale the FFT engine input size: 8 points→ 4 points
Multiple FIR Filters

- Even fastest kernel cannot meet time limit
  => Implement multiple FIR filters in parallel

Problem:
Bandwidth of off-chip memory is main problem

Solution:
- Do more processing!
- Calculate power of complex values (need input in next stage)

Problem:
Number of DSP blocks limits number of parallelisable filters

Solution:
- Downscale the FFT engine input size: 8 points→ 4 points
Multiple FIR Filters

1) Multiple FIR filters 2) Power of complex values
Becomes optimisation problem!
Multiple FIR Filters and FPGAs

- **Latency of 84 FIR Filters (ms)**
- **Kernel Performance (GFLOPS)**
- **Power Efficiency (GFLOPS/watt)**
- **Energy Dissipation (Joule)**

*Device*
- 1 FPGA
- 2 FPGAs
- 3 FPGAs

Oliver Sinnen, Haomiao Wang & Prabu Thiagaraj (Manchester Uni)
Latency–FPGA vs GPU

Latencies of a single GPU (AMD Radeon R7 370) and 3 FPGAs in processing 2 and 4 Million points:
Energy–FPGA vs GPU

Energy dissipation of single FPGA and GPU in executing the same task with different kernels:
Outline

1. Overview and Task
2. FT Convolution Decomposition
3. High-level Techniques and Implementation
4. Evaluation
5. What’s Next
Harmonic-summing

- **Input:** Filter-output-plane (FOP, $85 \times 2^{22}$ SPF points, \(~1.33\text{GBytes}\))
- **Processing flow:**
  - 8 harmonic planes are generated based on FOP and the stretched planes
  - One threshold for each row of each harmonic plane (overall: $85 \times 8 = 680$
  - Hundreds of candidates are recorded
- **Output:** Candidates
  - each candidate contains the indexes of filter, harmonic, bin, and amplitude (up to 64-bit)
Harmonic-summing

Problems:

- Input data size is too large (~1.33 GBytes)
  - On-chip memory size too small for all planes
- The cost of computation task is very cheap (SPF adds) and easy to parallelise
  - Off-chip memory bandwidth is issue

Challenge:

- Optimise data use (and reuse), computation not an issue
Conclusion

- FPGA-based implementation and optimisation of FT convolution (FIR filter), based on OLA and OLS algorithms
- High-level approaches to such tasks works well
  - Covered large design space
  - Easy porting and sharing with partners
- With multiple FPGAs, FPGA implementation has advantage over GPU in both performance (GFLOPS) and Energy efficiency