Accelerating the Pulsar Search Pipeline with FPGAs, Programmed in OpenCL

Oliver Sinnen, Tyrone Sherwin, and Haomiao Wang

&

Prabu Thiagaraj (Manchester Uni/Raman Research Institute, Bangalore)

Parallel and Reconfigurable Computing

Department of Electrical and Computer Engineering
University of Auckland

Computing for SKA, 2018
Pulsars — strong field physics

Image credit: NASA/Tod Strohmayer (GSFC)/Dana Berry (Chandra X-Ray Observatory)
FDAS: Fourier Domain Acceleration Search
Median Filter
Conclusions & Future

Pulsar Search Pipeline

Oliver Sinnen, Tyrone Sherwin, and Haomiao Wang & Pral
Pulsar Search on FPGA using OpenCL
Abstract

- Designing pipeline stages of Pulsar Search Pipeline
  - FDAS – Fourier Domain Acceleration Search
    - FT convolution
    - Harmonic summing
  - RFIM – RFI Mitigation
    - Median filtering
- On FGPA : Arria 10 (Nallatech 385)
- Using OpenCL as high level approach
  - Comparison to CPU and GPU!
Outline

1. FDAS: Fourier Domain Acceleration Search
   - FT Convolution
   - Harmonic-summing
   - Putting it together

2. Median Filter
   - Design Space Exploration
   - Evaluation

3. Conclusions & Future
Outline

1. **FDAS: Fourier Domain Acceleration Search**
   - FT Convolution
   - Harmonic-summing
   - Putting it together

2. **Median Filter**
   - Design Space Exploration
   - Evaluation

3. **Conclusions & Future**
- 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.

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

where frequency \( r_0 \) is unknown.

- Summation is computed at a range of frequencies \( r \).
FDAS: Fourier Domain Acceleration Search

Median Filter

Conclusions & Future

Fourier-domain Acceleration Search (FDAS)

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

**FT Convolution module + Harmonic-summing module**

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

Beami signals are de-dispersed for 6,000 DMs

PSS Engine

---

*Oliver Sinnen, Tyrone Sherwin, and Haomiao Wang & Pral
Pulsar Search on FPGA using OpenCL*
### Specification of Task

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{beam}$</td>
<td># of beams</td>
<td>1000 ~ 2000</td>
</tr>
<tr>
<td>$N_{DM}$</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>$N_{channel}$</td>
<td># of complex samples per group</td>
<td>$2^{21}$</td>
</tr>
<tr>
<td>$N_{template}$</td>
<td># of templates/filter</td>
<td>85</td>
</tr>
<tr>
<td>$N_{tap}$</td>
<td># of average template/filter length</td>
<td>422</td>
</tr>
<tr>
<td>$N_{hm}$</td>
<td># of harmonic planes</td>
<td>8</td>
</tr>
<tr>
<td>$N_{candidate}$</td>
<td># of candidates per harmonic plane</td>
<td>200</td>
</tr>
</tbody>
</table>
FT Convolution-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)**
  \[ \mathcal{F} \{ f \ast h \} = \mathcal{F} \{ f \} \cdot \mathcal{F} \{ h \} \]
FT Convolution-Decomposition Algorithms

Overlap-add Algorithm
- Split the coefficient array
  \[ \rightarrow \text{OLA-TD} \]

Overlap-save Algorithm
- Split the input array
  \[ \rightarrow \text{OLS-FD} \]

(a) OLA

(b) OLS
## Evaluation Platforms

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

<table>
<thead>
<tr>
<th>Device</th>
<th>Terasic DE5-Net (S5)</th>
<th>Nallatech 385A (A10)</th>
<th>Sapphire Nitro R7 370 (R7)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>Intel Stratix V</td>
<td>Intel Arria 10</td>
<td>AMD Radeon</td>
</tr>
<tr>
<td></td>
<td>5SGXA7</td>
<td>GX1150</td>
<td>R7 370</td>
</tr>
<tr>
<td>Technology</td>
<td>28nm</td>
<td>20nm</td>
<td>28nm</td>
</tr>
<tr>
<td>Compute resource</td>
<td>622,000 LEs</td>
<td>1,506,000 LEs</td>
<td>1,024 Processors</td>
</tr>
<tr>
<td></td>
<td>256 DSP blocks</td>
<td>1,518 DSP blocks</td>
<td>(16 Compute Units)</td>
</tr>
<tr>
<td>On-chip memory size</td>
<td>50Mb</td>
<td>53Mb</td>
<td>—</td>
</tr>
<tr>
<td>Off-chip memory size</td>
<td>2 x 2GB DDR3</td>
<td>2 x 4GB DDR3</td>
<td>4GB GDDR5</td>
</tr>
<tr>
<td>Memory interface width</td>
<td>2 x 64-bit</td>
<td>2 x 72-bit</td>
<td>256-bit</td>
</tr>
<tr>
<td>Max clock frequency</td>
<td>600MHz</td>
<td>1.5GHz</td>
<td>985MHz</td>
</tr>
<tr>
<td>Max power consumption</td>
<td>—</td>
<td>75W</td>
<td>150W</td>
</tr>
</tbody>
</table>
Latency—TDFIR vs FDFIR

Latencies of a single S5 FPGA (Intel Stratix V A7) in processing same input array using 7 different OpenCL kernels:

- TD–Naïve–64
- OLA–64
- FD–Naïve
- AOLS–1024
- AOLS–2048
- AOLS–4096
- TOLS–1024

Oliver Sinnen, Tyrone Sherwin, and Haomiao Wang & Pral

Pulsar Search on FPGA using OpenCL
Multiple FIR Filters

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

TOLS-1024 8points
AOLS-1024 8points
2 x AOLS-1024 8points
AOLS-1024-P 8points
2 x AOLS-1024-P 8points

AOLS-1024-P 4points
3 x AOLS-1024-P 4points
AOLS-2048-P 4points
3 x AOLS-2048-P 4points

Unused DSP blocks FFT Engine Element-wise multiplications Power
FT Convolution Latency–FPGA vs GPU

Latencies of a single GPU (AMD Radeon R7 370) and 3 FPGAs (S5 and A10) in processing 2 Million points:

![Graph showing latency comparison between GPU and FPGAs for 3x AOLS-2048-P (S5x3) and 4x AOLS-2048-P (A10x3) and GPU-FD (R7).]

Oliver Sinnen, Tyrone Sherwin, and Haomiao Wang & Pral
Pulsar Search on FPGA using OpenCL
Harmonic-summing-Problem

optimizing irregular accesses to off-chip memory
Harmonic-summing-Processing Methods

Processing a single harmonic plane at a time
- **SingleHP method**
  - SingleHP-\((S/M, V/R, N_{\text{paral}})\)
  - parallelize the implementation by a factor of \(N_{\text{paral}}\)

Processing multiple harmonic planes in parallel
- **MultipleHP method**
  - Naïve MultipleHP
  - MultipleHP-H-\(\left(N_{\text{MultipleHP-H-preld}}\right)\)
    - Preloading data with high touching frequency
  - MultipleHP-N-\(\left(N_{\text{MultipleHP-N-col}}\right)\)
    - loading necessary data to calculate a block of points
  - MultipleHP-R-\(\left(N_{\text{MultipleHP-R-col}, N_{\text{points/wi}}}\right)\)
    - based on MultipleHP-N and the FOP is reordered to achieve stream mode
Harmonic-summing Latency

Latency (ms)

OpenCL kernels

SingleHP, Naive MultipleHP, MultipleHP-H, MultipleHP-N, MultipleHP-R

S5, A10, A10x3
An FOP preparing module is added to connect the FT convolution and harmonic summing modules.
Pipeline Computing

latency for a single input array: \( t_{FDAS} = t_{FT} + t_{FOP} + t_{HM} \)

latency for a new input array in a pipeline: \( \max(t_{FT}, t_{FOP}, t_{HM}) \)

\[
\begin{align*}
\text{Single} & & \text{Multiple} \\
& & \text{Single} & & \text{Multiple} \\
\begin{array}{c|c|c}
\text{Max}(t_{FT}, t_{FOP}, t_{HM}) \geq t_{FDAS}/2 \\
\hline
\text{Max}(t_{FT}, t_{FOP}, t_{HM}) < t_{FDAS}/2 \\
\hline
\end{array}
\end{align*}
\]
Multiple Devices

- Same configuration file
  - processing a single input array at a time
    - if $t_{HM} \geq \max(t_{FT}, t_{FOP})$ and $t_{HM} \geq t_{FDAS}/2$
    - $\max(t_{FT}, t_{FOP}, t_{HM}/N_{devices})$
  - processing multiple input arrays in parallel
    - $\max(t_{FT}, t_{FOP}, t_{HM})/N_{devices}$

  \[
  \frac{\max(t_{FT}, t_{FOP}, t_{HM})}{N_{devices}} \leq \max(t_{FT}, t_{FOP}, \frac{t_{HM}}{N_{devices}})
  \]

- Multiple configuration files
  - transfer data between the host and different devices frequently
  - affected by the PCIe bus transfer rate
A Case Study

AOLS-2048 x2 + discard & transpose + $\frac{1}{3}$ MultipleHP-N

2 x FIRs + discard & transpose + HM (1/3)

Ideal triple buffering

Real result

discard discard transpose
Outline

1. FDAS: Fourier Domain Acceleration Search
   - FT Convolution
   - Harmonic-summing
   - Putting it together

2. Median Filter
   - Design Space Exploration
   - Evaluation

3. Conclusions & Future
Median Filtering

Form of stencil computation:

- Recomputing an array element based on neighbour values
- In median filter, target value is replaced by median of all values in a window around target value
- $\Rightarrow$ sorting all values, tacking middle value

Example:

- 2D array, window (stencil) is 3 x 3

\[
\begin{array}{ccc}
3 & 7 & 2 \\
1 & 2 & 4 \\
6 & 4 & 0 \\
\end{array}
\]

\[012234467\]
Requirements - SKA1 Mid

Large window size.
- 63 frequency channels, 5MHz, \( n \).
- 1023 time steps, 65ms, \( m \).

Array Size.
- Frequency coverage 4096 channels, 300MHz.
- Time coverage potentially infinite, steps of 64 us.

Throughput requirements.
- 8bit input data.
- 4096 channels every 64 us.
- 64 Million values per second.
Algorithmic Approach

- Choice of algorithmic approach dominates performance
- Using best of various state-of-the-art methods
- Histograms
  - 256 bins for 8bit data
  - Each bin counts that value in current window.
- Sliding
  - Adjacent windows overlap in data
  - Reuse previous window, replacing oldest row with next row
OpenCL

- Using Altera SDK for OpenCL (AOCL)
- Allows for rapid development
  - Multiple techniques quickly set up for evaluation
- Parameter sweeping with macro expansions
  - Varying techniques
  - Changing size
- AOCL specific optimisations:
  - $P$ parallel calculations, orthogonal to sliding.
    - Uses $P$ histograms with overlapping data.
    - Shared loading.
    - LTL method; improve loading performance at the cost of memory.
Developed to determine theoretical performance.

Limitations placed on loads per cycle and calculations per cycle.

$n = 63$ requires $P > 64$.

\[ T = \left( \frac{V}{c} \right) \times f \quad (1) \]

\[ \frac{V}{c} = \frac{l}{L} \times P \quad (2) \]

\[ L = (n + P - 1) \times 2 \quad (3) \]

- $T$ is throughput, $f$ is frequency, $L$ is loads per row, $l$ is loads per cycle
Pipeline balancing
Able to achieve required throughput
$l_{l-psthru}(P = 128)$ reaching 99.3M value/s.
  - Theoretical performance of 117.7M, limited by calculation rate.
  - 84.4% → Room for improvement.
  - $l_{l-psthru}(P = 128, b = 7)$ reduces bit depth for higher performance
CPU & GPU versions not able to perform as well – despite OpenCL adjustments

CPU appears not to operate in parallel

GPU quickly drops off

- Appears similar in shape to roofline models
- Suggests possible resource constraints

FPGA starts slower, much better scaling with $P$
Outline

1. FDAS: Fourier Domain Acceleration Search
   - FT Convolution
   - Harmonic-summing
   - Putting it together

2. Median Filter
   - Design Space Exploration
   - Evaluation

3. Conclusions & Future
Investigated the optimization of FT convolution and harmonic-summing and median filtering modules

Pipelining modules on single device (no host-FPGA communication)

Using multiple FPGAs

High-level approach is employed to implement the optimized methods

- Covered large design space
- Easy porting and sharing with partners
What’s Next

- Designing FPGA implementation for FLDO – Candidate Folding and Optimisation

Long term:
- Creating OpenCL library (target FPGA) for all (computationally intensive) Pulsar Search Pipeline stages