SeGraM: A Universal Hardware Accelerator for Genomic Sequence-to-Graph and Sequence-to-Sequence Mapping

Presented by Nicolà Lohr

27.04.2023

Mentors: Joel Lindegger, Banu Cavlak
Outline

- Executive Summary
- Background
- Motivation
- Sequence-to-graphs
- Goal
- SeGraM
- Results
- Conclusion
- Discussion
Executive Summary

- **Motivation**: Lack of accelerator for genomic sequence-to-graph mapping
- **Problem**: Current software solutions for sequence-to-graph are slow
- **Goal**: Build software, hardware co-designed algorithm for solving sequence-to-graph mapping
- **SeGraM**:
  - A universal hardware accelerator for sequence-to-graph mapping
  - **MinSeed**: The first minimizer-based seeding accelerator for genome graphs
  - **BitAlign**: The first bit vector-based sequence-to-graph alignment accelerator
- **Results**:
  - Compared to the current sequence-to-graph software, it has a 3.9x speed up in mapping
  - **BitAlign outperforms** state-of-the-art mapping tools in sequence-to-graph and sequence-to-sequence
- **Conclusion**:
  - Introduces first universal genomic mapping acceleration framework for sequence-to-graph
  - SeGraM has a higher throughput and lower-power consumption
Outline

- Executive Summary
- Background
- Motivation
- Sequence-to-graphs
- Goal
- SeGraM
- Results
- Conclusion
- Discussion
Background

- Genome is the **blueprint for life**
- Variations in the genome lead to **variation in beings**
- Certain **diseases are detectable** on the genome
  - Genome is interesting for medicine
Sequence of AGCT
Sequencing machines give only a partial sequence
  That is the reason why we need mapping
Millions of partial measurements of the genome, called reads
Genome sequencing is the process of getting the partial reads
Outline

- Executive Summary
- Background
- Motivation
- Sequence-to-graphs
- Goal
- SeGraM
- Results
- Conclusion
- Discussion
The African pan-genome contains \(~10\%\) more DNA than the current human reference genome.
Motivation – Reference Bias

Human reference genome

Reads
Motivation – Reference Bias

Human reference genome

Reads mapped on reference
Motivation – Reference Bias

Human reference genome

Reads but with unfitting variations
Motivation – Reference Bias

Human reference genome

Reads but mapping doesn't work
Motivation – Reference Bias

We have multiple reference genomes
One may fit better than another
Simple solution, try all of them
Motivation – Reference Bias

• Better solution is to merge them into a graph
• Avoids redundant data and computation
• Captures unknown combinations
Outline

- Executive Summary
- Background
- Motivation
- Sequence-to-graphs
- Goal
- SeGraM
- Results
- Conclusion
- Discussion
Genome graph example

- Combine different reference sequence genomes to a graph
- One graph covers all references and combinations of it

**Sequence #1:** ACGTACGT

**Sequence #2:** ACGGACGT

**Sequence #3:** ACGTTACGT

**Sequence #4:** ACGACGT
Sequence to Graph

1. Genome Graph Construction
   * Linear reference genome
   * Known genetic variations
   * Genome graph

2. Indexing
   * Hash-table-based index (of graph nodes)

3. Seeding
   * Reads from sequenced genome
   * Candidate mapping locations (subgraphs)

4. Filtering/Chaining/Clustering
   * Remaining candidate mapping locations (subgraphs)

5. Alignment
   * Optimal alignment between read & subgraph

Pre-Processing Steps (Offline)

Seed-and-Extend Steps (Online)
Sequence to Sequence Alignment

*Single linear reference*

```
<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
</table>
```

*Query read*

```
<table>
<thead>
<tr>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
</tr>
<tr>
<td>G</td>
</tr>
<tr>
<td>T</td>
</tr>
</tbody>
</table>
```

Only look at direct neighbors
Sequence to Graph Alignment

Looks at neighbors and hops
Outline

- Executive Summary
- Background
- Motivation
- Sequence-to-graphs
- Goal
- SeGraM
- Results
- Conclusion
- Discussion
Goal

- Design high-performance, scalable, power- and area-efficient hardware accelerators
- Alleviate bottlenecks in both the seeding and alignment steps of sequence-to-graph mapping
- With support for both short and long reads
Outline

- Executive Summary
- Background
- Motivation
- Sequence-to-graphs
- Goal
- SeGraM
- Results
- Conclusion
- Discussion
SeGraM

- Provides efficient and general purpose acceleration for sequence-to-graph mapping
- A co-design between algorithm and hardware accelerator
- Consists out of two parts
  - MinSeed: The first minimizer-based seeding accelerator for genome graphs
  - BitAlign: The first bit vector-based sequence-to-graph alignment accelerator
SeGraM – Pre-Processing

- **Generate Genome Graph** using linear reference genome
  - vg toolkit
- Use variations for **alternative paths** in graph
- One graph per chromosome
- Sort the graph **topologically**
SeGraM – Hardware

Main Memory (graph-based reference & index)

Host CPU
MinSeed
SeGraM Accelerator
BitAlign
SeGraM – Hardware

Main Memory (graph-based reference & index)

Host CPU

Minimizer Scratchpad
Find Minimizers
Read Scratchpad

Seed Scratchpad
Filter Minimizers by Frequency
Find Candidate Seed Regions

SeGraM Accelerator

Input Scratchpad
Generate Bitvectors
Hop Queue
Bitvector Scratchpad
Perform Traceback
BitAlign

MinSeed
SeGraM – Hardware

Main Memory (graph-based reference & index)

- Minimizer Scratchpad
- Seed Scratchpad
- Input Scratchpad
- Bitvector Scratchpad
- Hop Queue
- Generate Bitvectors
- Perform Traceback

- Find Minimizers
- Filter Minimizers by Frequency
- Find Candidate Seed Regions

- Read Scratchpad

Host CPU

SeGraM Accelerator
SeGraM – Hardware

Main Memory (graph-based reference & index)

- Minimizer Scratchpad
  - Find Minimizers
  - Filter Minimizers by Frequency
  - Find Candidate Seed Regions
  - Minimizer Scratchpad

- Seed Scratchpad
  - Generate Bitvectors
  - Hop Queue
  - Bitvector Scratchpad
  - Perform Traceback

- Input Scratchpad

- BitAlign

Host CPU

- Read Scratchpad

SeGraM Accelerator
SeGraM – Algorithm – MinSeed

<table>
<thead>
<tr>
<th>Position</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence</td>
<td>A</td>
<td>G</td>
<td>T</td>
<td>A</td>
<td>G</td>
<td>C</td>
<td>A</td>
<td>...</td>
</tr>
<tr>
<td>$k$-mer$_1$</td>
<td>A</td>
<td>G</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$k$-mer$_2$</td>
<td>G</td>
<td>T</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$k$-mer$_3$</td>
<td>T</td>
<td>A</td>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$k$-mer$_4$</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>G</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$k$-mer$_5$</td>
<td></td>
<td></td>
<td></td>
<td>G</td>
<td>C</td>
<td>A</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Stores only selected seeds
  - Decreases the storage requirements
  - Lexicographically smallest $k$-mer
- Extracts the same seed consistently
Finding out which part to load from memory
Only arithmetic operations
SeGraM – Hardware – MinSeed

Diagram:
- Read Scratchpad (6 kB)
- Minimizer Finder
- Minimizer Scratchpad (40 kB)
- Minimizer Filter by Frequency (<?)
- Seed Scratchpad (4 kB)
- Candidate Seed Region Calculator (+/−/×)

Main Memory (High Bandwidth Memory)
- 2 Bytes frequency
- 8 Bytes seed

Candidate subgraph (OUTPUT)

Query read (INPUT)
Frequency threshold (INPUT)
Error rate, read length (INPUT)
SeGraM – Algorithm – BitAlign

- Dynamic Programming Table
  - With successors not only neighbors
  - Could access any previous values

**Algorithm 1 BitAlign Algorithm**

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>n ← length of linearized reference subgraph</td>
</tr>
<tr>
<td>2:</td>
<td>m ← length of query read</td>
</tr>
<tr>
<td>3:</td>
<td>PM ← genPatternBitmasks(query-read) ▶ pre-process the query read</td>
</tr>
<tr>
<td>4:</td>
<td>allR[n][d] ← 111...111 ▶ init R[d] bitvectors for all characters with 1s</td>
</tr>
<tr>
<td>5:</td>
<td>for i in (n-1):-1:0 do ▶ iterate over each subgraph node</td>
</tr>
<tr>
<td>6:</td>
<td>curChar ← subgraph-nodes[i].char</td>
</tr>
<tr>
<td>7:</td>
<td>curPM ← PM[curChar] ▶ retrieve the pattern bitmask</td>
</tr>
<tr>
<td>8:</td>
<td>R0 ← 111...111 ▶ status bitvector for exact match</td>
</tr>
<tr>
<td>9:</td>
<td>for j in subgraph-nodes[i].successors do ▶ exact match calculation</td>
</tr>
<tr>
<td>10:</td>
<td>R0 ← (R[i][j][d] &amp; curPM) &amp; R0</td>
</tr>
<tr>
<td>11:</td>
<td>allR[i][j][d] ← R0</td>
</tr>
<tr>
<td>12:</td>
<td>for d in 1:k do ▶ insertion</td>
</tr>
<tr>
<td>13:</td>
<td>I ← (allR[i][d-1]==1) ▶ status bitvector for d errors</td>
</tr>
<tr>
<td>14:</td>
<td>Rd ← I</td>
</tr>
<tr>
<td>15:</td>
<td>for j in subgraph-nodes[i].successors do ▶ deletion</td>
</tr>
<tr>
<td>16:</td>
<td>D ← allR[i][j][d-1]</td>
</tr>
<tr>
<td>17:</td>
<td>S ← allR[i][j][d-1] ▶ substitution</td>
</tr>
<tr>
<td>18:</td>
<td>M ← (allR[i][j][d] &amp; curPM) ▶ match</td>
</tr>
<tr>
<td>19:</td>
<td>Rd ← D &amp; S &amp; M &amp; Rd</td>
</tr>
<tr>
<td>20:</td>
<td>allR[i][d] ← Rd</td>
</tr>
<tr>
<td>21:</td>
<td>&lt;editDist, CIGARstr&gt; ← traceback(allR, subgraph, query-read)</td>
</tr>
</tbody>
</table>
SeGraM – Hardware – BitAlign
SeGraM – Hardware

- Stackable

* Each <MS + BA> accelerator communicates with the host independently of other <MS + BA> accelerators. There is no communication required between different <MS + BA> accelerators in a single SeGraM module.
SeGraM – Use Cases

- **End-to-End Mapping**
  - The whole SeGraM can be employed
  - Long and short reads
  - Sequence-to-Graph but also **Sequence-to-Sequence Mapping**

- **Alignment**
  - As standalone sequence-to-graph aligner
  - BitAlign can be coupled with **any seeding accelerator**

- **Seeding**
  - As standalone seeding accelerator
  - MinSeed can be coupled with **any alignment tool or accelerator**
Outline

- Executive Summary
- Background
- Motivation
- Sequence-to-graphs
- Goal
- SeGraM
- Results
- Conclusion
- Discussion
## Results

- Data from synthesis with 28nm process at 1GHz

<table>
<thead>
<tr>
<th>Component</th>
<th>Area (mm²)</th>
<th>Power (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MinSeed – Logic</td>
<td>0.017</td>
<td>10.8</td>
</tr>
<tr>
<td>Read Scratchpad (6 kB)</td>
<td>0.012</td>
<td>7.9</td>
</tr>
<tr>
<td>Minimizer Scratchpad (40 kB)</td>
<td>0.055</td>
<td>22.7</td>
</tr>
<tr>
<td>Seed Scratchpad (4 kB)</td>
<td>0.008</td>
<td>6.4</td>
</tr>
<tr>
<td>BitAlign – Edit Distance Calculation Logic with Hop Queue Registers (64 PEs)</td>
<td>0.393</td>
<td>378.0</td>
</tr>
<tr>
<td>BitAlign – Traceback Logic</td>
<td>0.020</td>
<td>2.7</td>
</tr>
<tr>
<td>Input Scratchpad (24 kB)</td>
<td>0.033</td>
<td>13.3</td>
</tr>
<tr>
<td>Bitvector Scratchpads (128 kB)</td>
<td>0.329</td>
<td>316.2</td>
</tr>
<tr>
<td><strong>Total – 1 SeGrA_M Accelerator</strong></td>
<td>0.867</td>
<td>758.0 (0.8 W)</td>
</tr>
<tr>
<td><strong>Total – 32 SeGrA_M Accelerators</strong></td>
<td>27.744</td>
<td>24256.0 (24.3 W)</td>
</tr>
<tr>
<td>HBM2E (4 stacks)</td>
<td>3.8</td>
<td></td>
</tr>
</tbody>
</table>

Intel Xeon E5-2630 v4 CPU at 2.20 GHz at 40nm

- 246.2
- 85000 (85 W)
Results

Throughput improves by 5.9x and 3.9x over GraphAligner and vg with long read
Results

Throughput improves by 106x and 742x over GraphAligner and vg with short reads.
Results – BitAlign

- BitAlign can be used for sequence-to-sequence alignment
  - Extra cost: additional hop queue registers
  - no sacrifice in performance relative to GenASM

- BitAlign is comparable with GenASM, GACT of Darwin and SillaX of GenAx)
  - For long reads (over GACT of Darwin and GenASM):
    - 4.8× and 1.2× throughput improvement
    - 1.5× and 2.6× higher area overhead
    - 2.7× and 7.5× higher power consumption
  - For short reads (over SillaX of GenAx and GenASM):
    - 2.4× and 1.3× throughput improvement
Results – Analysis

- **Analysis of MinSeed**
  - MinSeed is not on the critical path of the overall SeGraM
  - MinSeed and the baseline software tools implement the same optimization of discarding the seeds that have higher frequency than some threshold
  - Decreases seeds from 77M to 35M instead of the filtering approach which reduces it to 48k

- **Still, SeGraM is overall faster because of BitAlign**
Outline

- Executive Summary
- Background
- Motivation
- Sequence-to-graphs
- Goal
- SeGraM
- Results
- Conclusion
- Discussion
Conclusion

- **SeGrAM** is a **promising framework** for accelerating both graph-based and traditional linear-sequence-based genome sequence analysis
  - **MinSeed**: First minimizer-based seeding accelerator
  - **BitAlign**: First sequence-to-graph alignment accelerator
- **Accelerating graph-based genome analysis via efficient algorithm/hardware co-design has potential**
  - **Inspires future research**
- **SeGrAM** outperforms state-of-the-art software and hardware solutions
Outline

- Executive Summary
- Background
- Motivation
- Sequence-to-graphs
- Goal
- SeGraM
- Results
- Conclusion
- Discussion
Strengths

- Introduces an actual product
- SeGraM even outperforms sequence-to-sequence accelerators
- All processes well explained
Weaknesses

- Very minimalistic code (on github)
  - HDL code not available
- No filtering, although paper says it is a stage
- Lack of figures for some numbers
- Rushed over a few of the results
Discussion

- Currently no filtering in SeGram
- Why didn't they use SneakySnake filter?
  - SneakySnake is not compatible with graphs
Discussion

- Why didn't they use another filter?
- Currently there doesn't exist a filter for sequence-to-graph mapping
- Needs algorithm and hardware which can hold up to BitAlign
  - Otherwise bottleneck
- Huge difference between a few millions and a few thousands
  - Could have potential for speed up
Discussion

- Does it make sense to do the same with inexact string matching?
- The genome sequencing machines make mistakes
  - Partially false data
Discussion

ISMatch: A real-time hardware accelerator for inexact string matching of DNA sequences on FPGA

Alberto Marchisio a,*, Federico Teodonio b, Antonello Rizzi b, Muhammad Shafique c

a Technische Universität Wien (TU Wien), Vienna, Austria
b University of Rome “La Sapienza”, Rome, Italy
c eBrain Lab, Division of Engineering, New York University Abu Dhabi, United Arab Emirates

- Created an own inexact string matching algorithm for hardware
- Increased performance of up to 70x compared to software
- Not for sequence-to-graph
Questions?
SeGraM: A Universal Hardware Accelerator for Genomic Sequence-to-Graph and Sequence-to-Sequence Mapping

Presented by Nicolà Lohr

27.04.2023

Mentors: Joel Lindegger, Banu Cavlak
Figure 4: Overview of SeGraM.