GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis

Damla Senol Cali (damlasenolcali@gmail.com), Gurpreet S. Kalsi, Zulal Bingol, Can Firtina, Lavanya Subramanian, Jeremie S. Kim, Rachata Ausavarungnirun, Mohammed Alser, Juan Gomez-Luna, Amirali Boroumand, Anant Nori, Allison Scibisz, Sreenivas Subramoney, Can Alkan, Saugata Ghose, and Onur Mutlu

Carnegie Mellon · ETH zürich · intel · Bilkent University · University of Illinois Urbana-Champaign · SAFARI

Genome Sequencing

- ❑ **Genome sequencing:** Enables us to determine the order of the DNA sequence in an organism's genome
  - Plays a pivotal role in:
    - Personalized medicine
    - Outbreak tracing
    - Understanding of evolution

- ❑ Modern genome sequencing machines extract smaller randomized fragments of the original DNA sequence, known as **reads**
  - *Short reads:* a few hundred base pairs, error rate of ~0.1%
  - *Long reads:* thousands to millions of base pairs, error rate of 10–15%

Genome Sequence Analysis (GSA)

- ❑ **Read mapping:** *First key step* in genome sequence analysis (GSA)
  - Aligns reads to one or more possible locations within the reference genome, and
  - Finds the matches and differences between the read and the reference genome segment at that location
- ❑ Multiple steps of read mapping require ***approximate string matching***
  - Approximate string matching (ASM) enables read mapping to account for sequencing errors and genetic variations in the reads
- ❑ Bottlenecked by the computational power and memory bandwidth limitations of existing systems

Approximate String Matching (ASM)

- ❑ Sequenced genome may not exactly map to the reference genome due to genetic variations and sequencing errors

```
Reference: AAAATGTTTAGTGCTAC TG
     Read: AAA TGTTTACTGCTACTTG
           deletion  substitution  insertion
```

- ❑ **Approximate string matching (ASM):**
  - Detect the differences and similarities between two sequences
  - In genomics, ASM is required to:
    - Find the *minimum edit distance* (i.e., total number of edits)
    - Find the *optimal alignment* with a *traceback* step
      - Sequence of matches, substitutions, insertions and deletions, along with their positions
      - **3M-1D-6M-1S-6M-1I-2M** for the above example
  - Usually implemented as a dynamic programming (DP) based algorithm

Bitap Algorithm

- ❑ Bitap[1,2] performs ASM with fast and simple bitwise operations
  - Amenable to efficient hardware acceleration
  - Computes the **minimum edit distance** between a **text** (e.g., reference genome) and a **pattern** (e.g., read) with a maximum of ***k*** errors
- ❑ **Step 1: Pre-processing (per pattern)**
  - Generate a pattern bitmask (PM) for each character in the alphabet (A, C, G, T)
  - Each PM indicates if character exists at each position of the pattern
- ❑ **Step 2: Searching (Edit Distance Calculation)**
  - Compare all characters of the text with the pattern by using:
    - Pattern bitmasks
    - Status bitvectors that hold the partial matches
    - Bitwise operations

[1] R. A. Baeza-Yates and G. H. Gonnet. "A New Approach to Text Searching." *CACM*, 1992.
[2] S. Wu and U. Manber. "Fast Text Searching: Allowing Errors." *CACM*, 1992.

Limitations of Bitap

Algorithm

1) **Data Dependency Between Iterations:**
   - Two-level data dependency forces the consecutive iterations to take place sequentially
2) **No Support for Traceback:**
   - Bitap does not include any support for optimal alignment identification
3) **No Support for Long Reads:**
   - Each bitvector has a length equal to the length of the pattern
   - Bitwise operations are performed on these bitvectors

Hardware

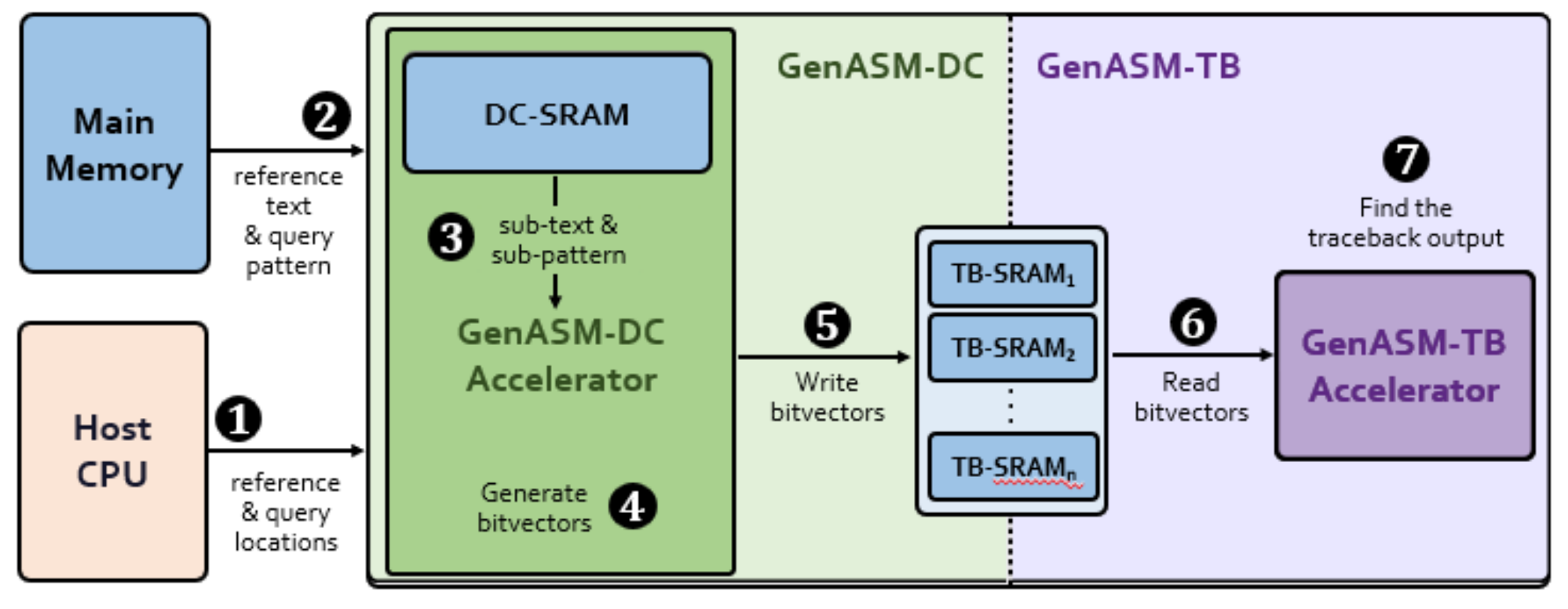
4) **Limited Compute Parallelism:**
   - Text-level parallelism
   - Limited by the number of compute units in existing systems
5) **Limited Memory Bandwidth:**
   - High memory bandwidth required to read and write the computed bitvectors to memory

GenASM: ASM Framework for GSA

Our Goal:
Accelerate approximate string matching
by designing a fast and flexible framework,
which can accelerate *multiple steps* of genome sequence analysis

- ❑ **GenASM:** *First* ASM acceleration framework for GSA
  - Based upon the *Bitap* algorithm
    - Uses fast and simple bitwise operations to perform ASM
- ❑ We overcome the five limitations that hinder Bitap's use in genome sequence analysis:
  - Modified and extended ASM algorithm
    - Highly-parallel Bitap with long read support
    - Novel bitvector-based algorithm to perform *traceback*
  - Specialized, low-power and area-efficient hardware for both modified Bitap and novel traceback algorithms

GenASM Hardware Design



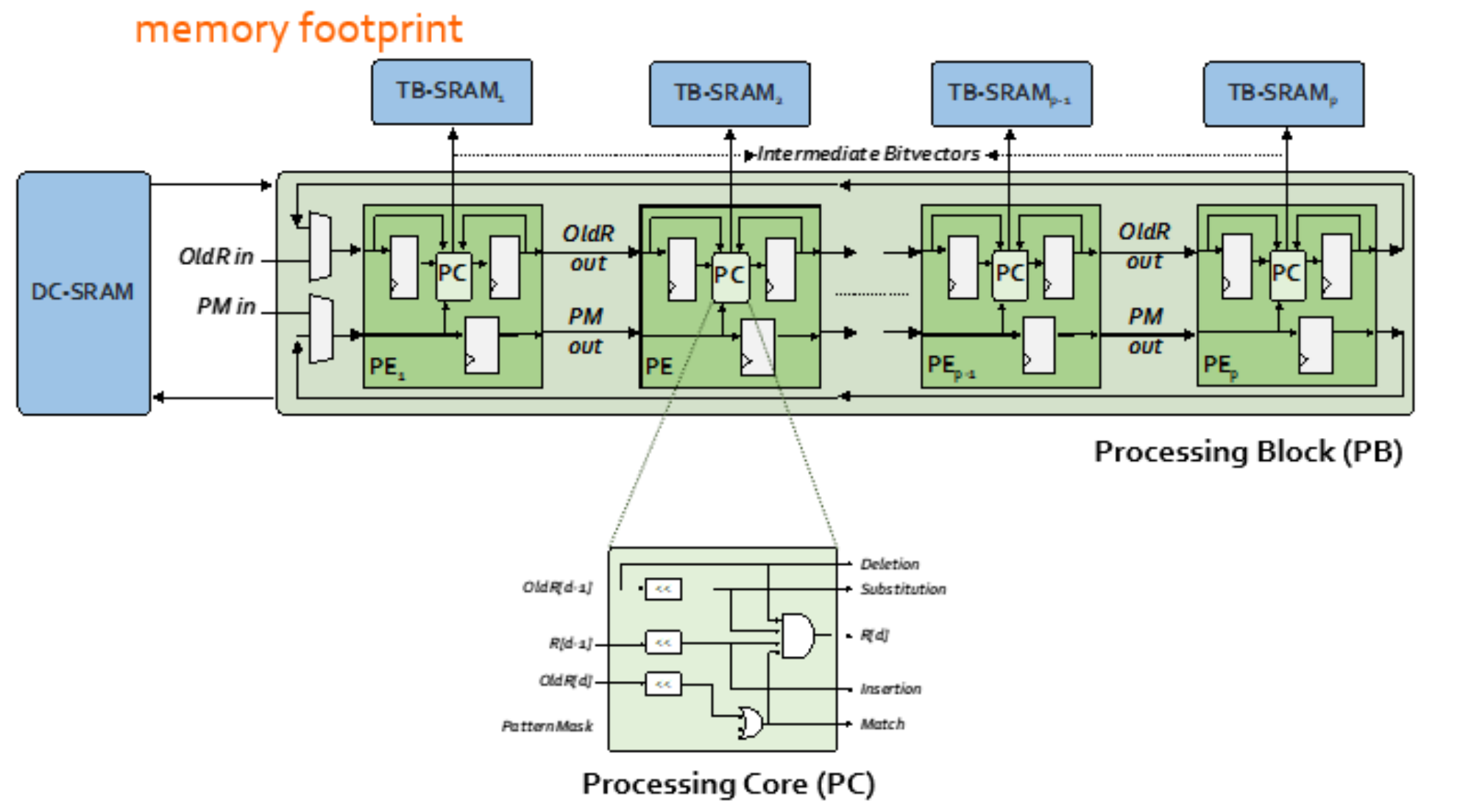
GenASM-DC: generates bitvectors and performs edit Distance Calculation

GenASM-TB: performs TraceBack and assembles the optimal alignment

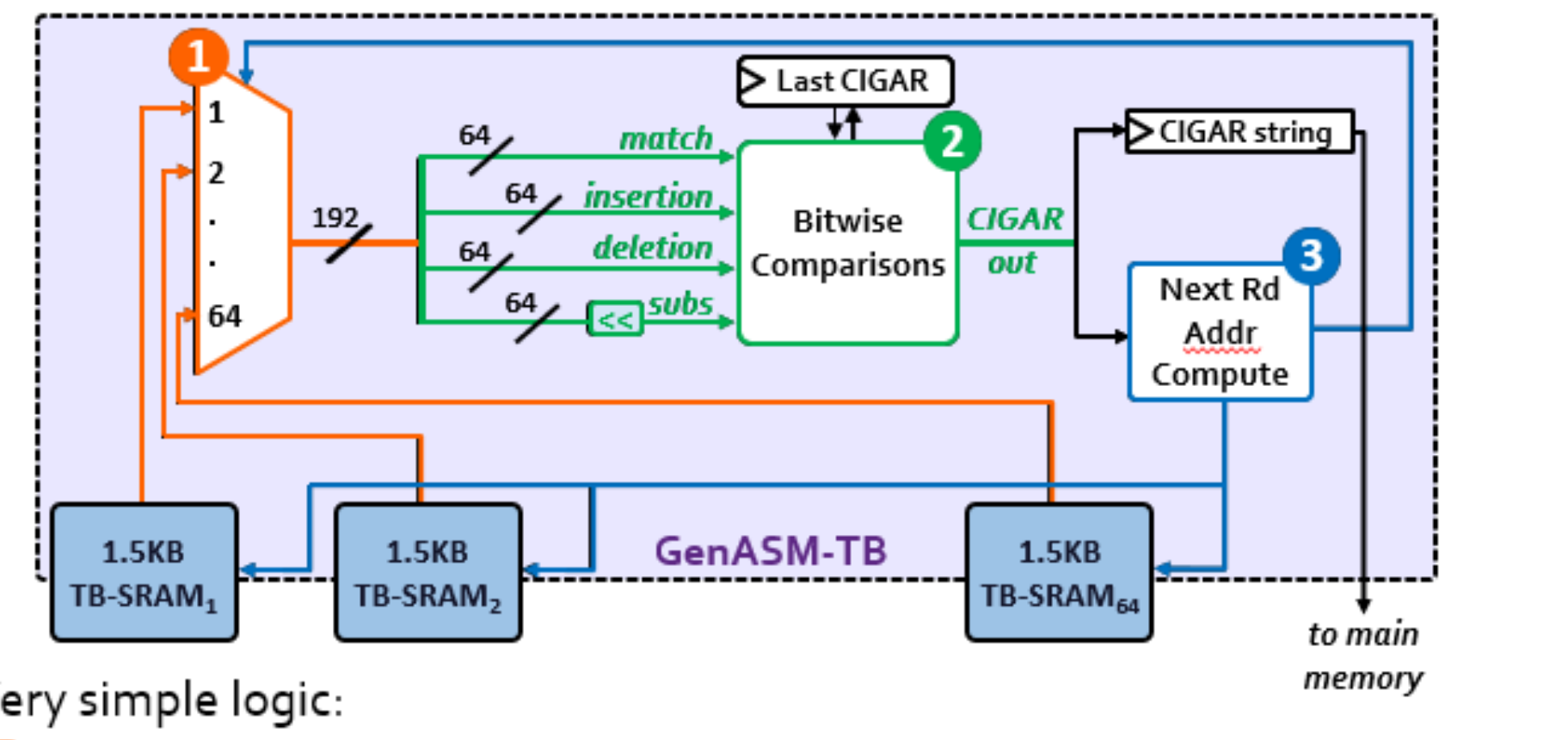
Our *specialized compute units* and *on-chip SRAMs* help us to:
→ Match the rate of computation with memory capacity and bandwidth
→ Achieve high performance and power efficiency
→ Scale linearly in performance with the number of parallel compute units that we add to the system

GenASM-DC: Hardware Design

- ❑ **Linear cyclic systolic array** based accelerator
  - Designed to maximize parallelism and minimize memory bandwidth and memory footprint



GenASM-TB: Hardware Design



- ❑ Very simple logic:
  1. Reads the bitvectors from one of the TB-SRAMs using the computed address
  2. Performs the required bitwise comparisons to find the traceback output for the current position
  3. Computes the next TB-SRAM address to read the new set of bitvectors

Use Cases of GenASM

(1) **Read Alignment Step of Read Mapping**
- Find the optimal alignment of how reads map to candidate reference regions

(2) **Pre-Alignment Filtering for Short Reads**
- Quickly identify and filter out the unlikely candidate reference regions for each read

(3) **Edit Distance Calculation**
- Measure the similarity or distance between two sequences

- ❑ We also discuss other possible use cases of GenASM in our paper:
  - Read-to-read overlap finding, hash-table based indexing, whole genome alignment, generic text search

Evaluation Methodology

- ❑ We evaluate GenASM using:
  - Synthesized SystemVerilog models of the GenASM-DC and GenASM-TB accelerator datapaths
  - Detailed simulation-based performance modeling
- ❑ 16GB HMC-like 3D-stacked DRAM architecture
  - 32 vaults
  - 256GB/s of internal bandwidth, clock frequency of 1.25GHz
  - In order to achieve high parallelism and low power-consumption
  - Within each vault, the logic layer contains a GenASM-DC accelerator, its associated DC-SRAM, a GenASM-TB accelerator, and TB-SRAMs.

	SW Baselines	HW Baselines
Read Alignment	Minimap2[1] BWA-MEM[2]	GACT (Darwin)[3] SillaX (GenAx)[4]
Pre-Alignment Filtering	–	Shouji[5]
Edit Distance Calculation	Edlib[6]	ASAP[7]

[1] H. Li. "Minimap2: Pairwise Alignment for Nucleotide Sequences." In *Bioinformatics*, 2018.
[2] H. Li. "Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM." In *arXiv*, 2013.
[3] Y. Turakhia et al. "Darwin: A genomics co-processor provides up to 15,000 x acceleration on long read assembly." In *ASPLOS*, 2018.
[4] D. Fujiki et al. "GenAx: A genome sequencing accelerator." In *ISCA*, 2018.
[5] M. Alser. "Shouji: A fast and efficient pre-alignment filter for sequence alignment." In *Bioinformatics*, 2019.
[6] M. Šošić et al. "Edlib: A C/C++ library for fast, exact sequence alignment using edit distance." In *Bioinformatics*, 2017.
[7] S.S. Banerjee et al. "ASAP: Accelerated short-read alignment on programmable hardware." In *TC*, 2018.

- ❑ **For Use Case 1: Read Alignment**, we compare GenASM with:
  - Minimap2 and BWA-MEM (state-of-the-art SW)
    - Running on Intel® Xeon® Gold 6126 CPU (12-core) operating @2.60GHz with 64GB DDR4 memory
    - Using two simulated datasets:
      - Long ONT and PacBio reads: 10Kbp reads, 10-15% error rate
      - Short Illumina reads: 100-250bp reads, 5% error rate
  - GACT of Darwin and SillaX of GenAx (state-of-the-art HW)
    - Open-source RTL for GACT
    - Data reported by the original work for SillaX
    - GACT is best for long reads, SillaX is best for short reads
- ❑ **For Use Case 2: Pre-Alignment Filtering,** we compare GenASM with:
  - Shouji (state-of-the-art HW – FPGA-based filter)
    - Using two datasets provided as test cases:
      - 100bp reference-read pairs with an edit distance threshold of 5
      - 250bp reference-read pairs with an edit distance threshold of 15
- ❑ **For Use Case 3: Edit Distance Calculation,** we compare GenASM with:
  - Edlib (state-of-the-art SW)
    - Using two 100Kbp and 1Mbp sequences with similarity ranging between 60%-99%
  - ASAP (state-of-the-art HW – FPGA-based accelerator)
    - Using data reported by the original work

Additional Details in the Paper

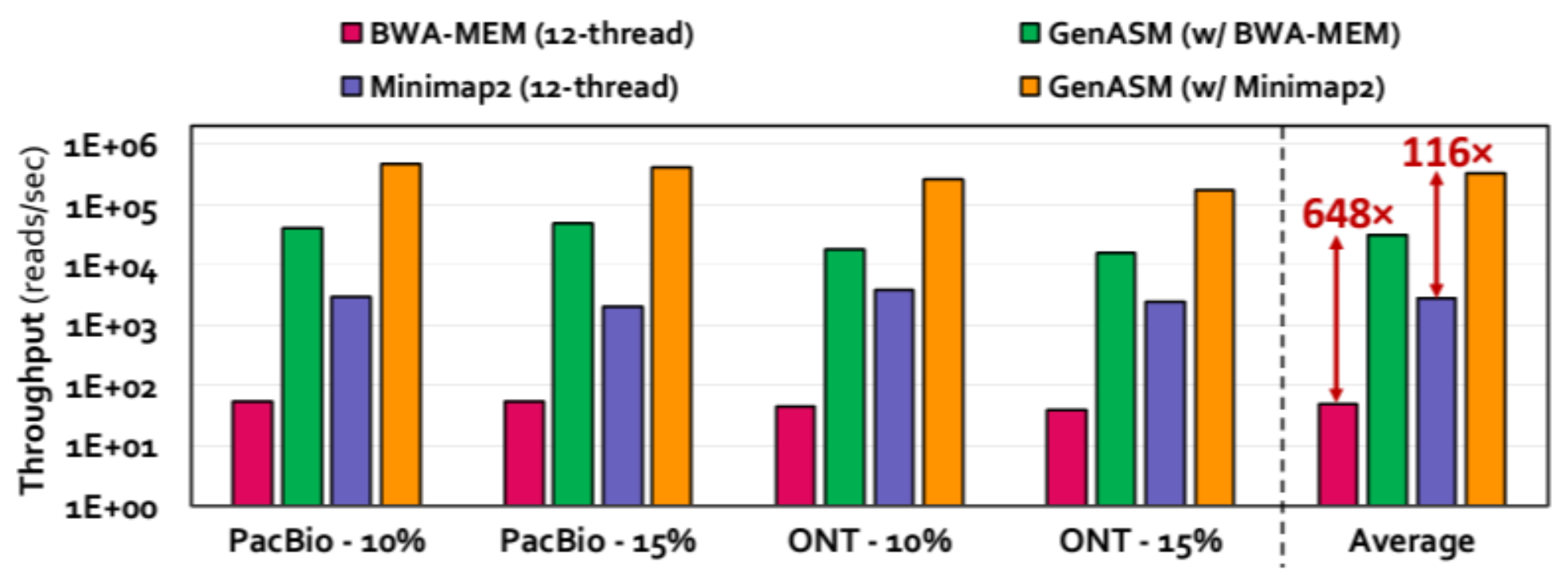
- ❑ Details of the **GenASM-DC and GenASM-TB algorithms**
- ❑ **Big-O analysis** of the algorithms
- ❑ Detailed explanation of **evaluated use cases**
- ❑ **Evaluation methodology details** (datasets, baselines, performance model)
- ❑ **Additional results** for the three evaluated use cases
- ❑ **Sources of improvements in GenASM** (algorithm-level, hardware-level, technology-level)
- ❑ Discussion of **four other potential use cases** of GenASM

Related Links

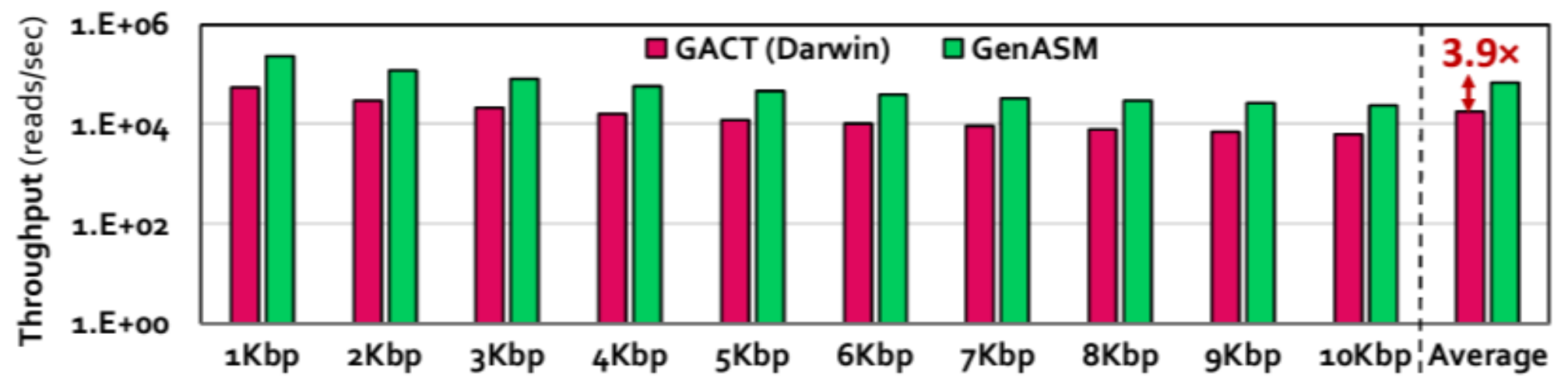
Paper (PDF) · **MICRO'20 Talk** · **Source Code**

Results – Use Case 1

Long Reads

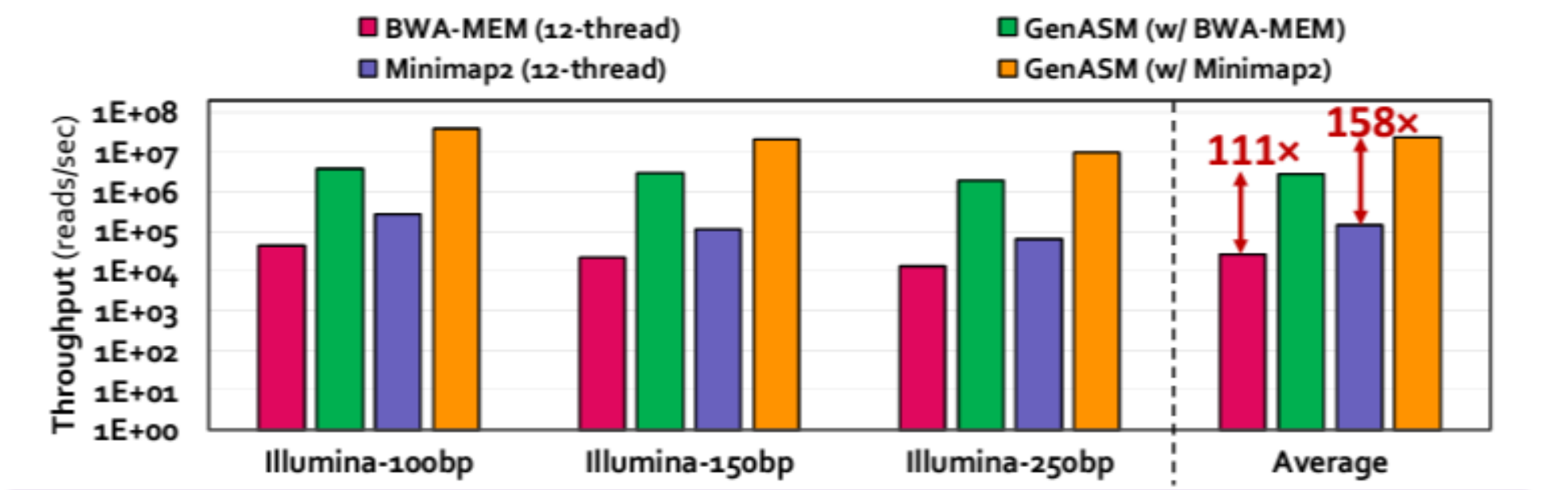


SW GenASM achieves 648× and 116× speedup over 12-thread runs of BWA-MEM and Minimap2, while reducing power consumption by 34× and 37×



HW GenASM provides 3.9× better throughput, 6.6× the throughput per unit area, and 10.5× the throughput per unit power, compared to GACT of Darwin

Short Reads



SW GenASM achieves 111× and 158× speedup over 12-thread runs of BWA-MEM and Minimap2, while reducing power consumption by 33× and 31×

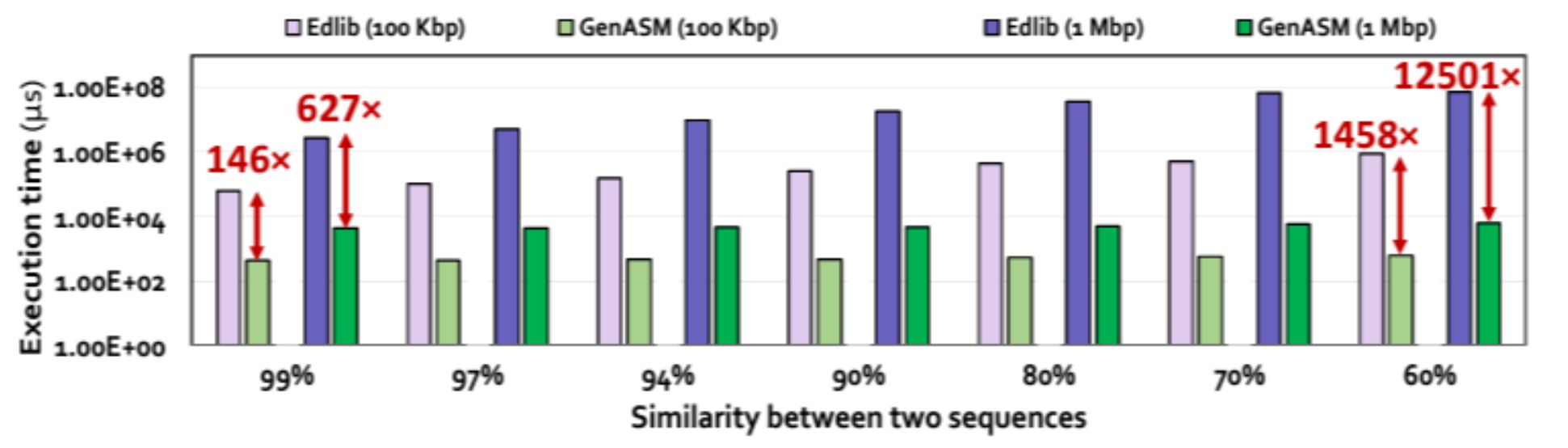
HW GenASM provides 1.9× better throughput and uses 63% less logic area and 82% less logic power, compared to SillaX of GenAx

Results – Use Case 2

- ❑ Compared to Shouji:
  - 3.7× speedup
  - 1.7× less power consumption
  - False accept rate of 0.02% for GenASM vs. 4% for Shouji
  - False reject rate of 0% for both GenASM and Shouji

HW GenASM is more efficient in terms of both speed and power consumption, while significantly improving the accuracy of pre-alignment filtering

Results – Use Case 3

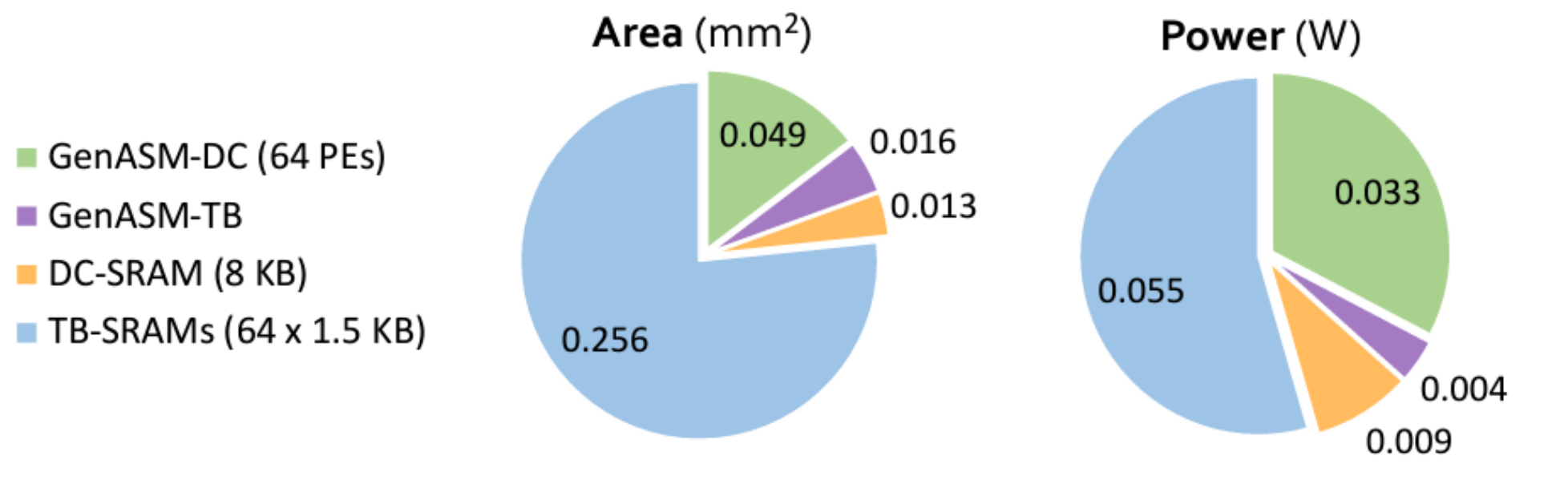


SW GenASM provides 146 – 1458× and 627 – 12501× speedup, while reducing power consumption by 548× and 582× for 100Kbp and 1Mbp sequences, respectively, compared to Edlib

HW GenASM provides 9.3 – 400× speedup over ASAP, while consuming 67× less power

Results – Area & Power

- ❑ Based on our synthesis of GenASM-DC and GenASM-TB accelerator datapaths using the Synopsys Design Compiler with a 28nm LP process:
  - Both GenASM-DC and GenASM-TB operate @ 1GHz



	Area	Power
Total (1 vault):	0.334 mm²	0.101 W
Total (32 vaults):	10.69 mm²	3.23 W
% of a Xeon CPU core:	1%	1%

GenASM is significantly more efficient for all the three use cases (in terms of throughput and throughput per unit power) than state-of-the-art software and hardware baselines

GenASM has low area and power overheads