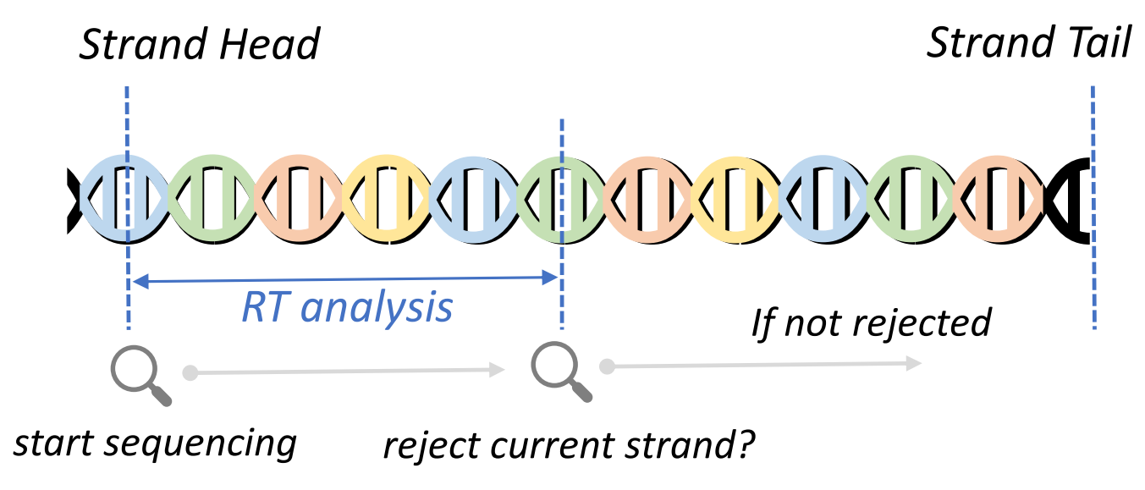
Efficient real-time selective genome sequencing on resource-constrained devices

Po Jui Shih[1], Hassaan Saadat[2], Sri Parameswaran[3], and Hasindu Gamaarachchi[1,4]

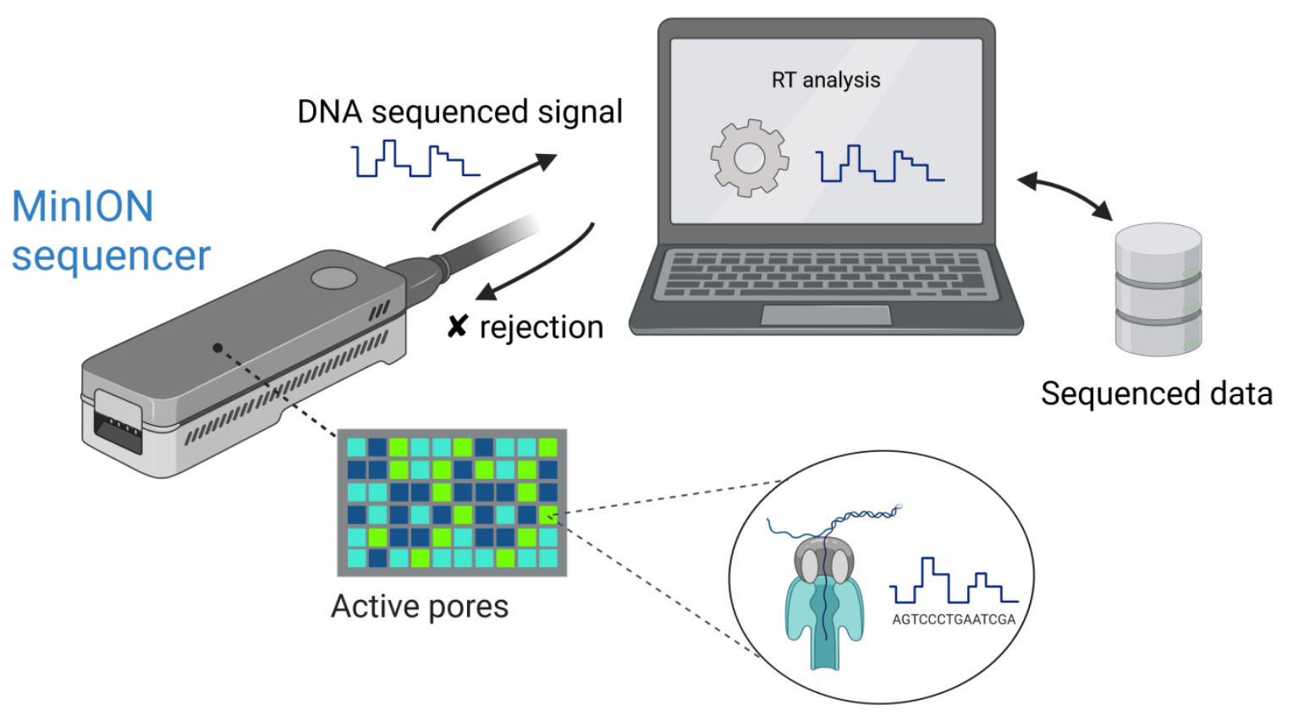
[1] School of Computer Science and Engineering, UNSW, Sydney [2] School of Electrical Engineering and Telecommunications, UNSW, Sydney [3] School of Electrical and Information Engineering, University of Sydney, Sydney [4] Kinghorn Centre for Clinical Genomics, Garvan Institute of Medical Research, Sydney

Introduction

Nanopore sequencers provide **portable** long-read sequencing and the ability to **access, analyse, and filter reads in real-time** → **Read Until**



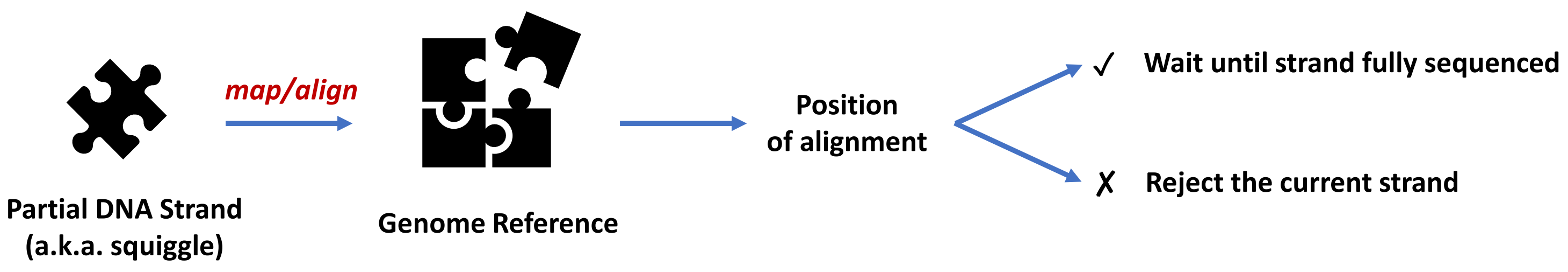
Reduces sequencing time by *rejecting strands not of interest!*



Steps:

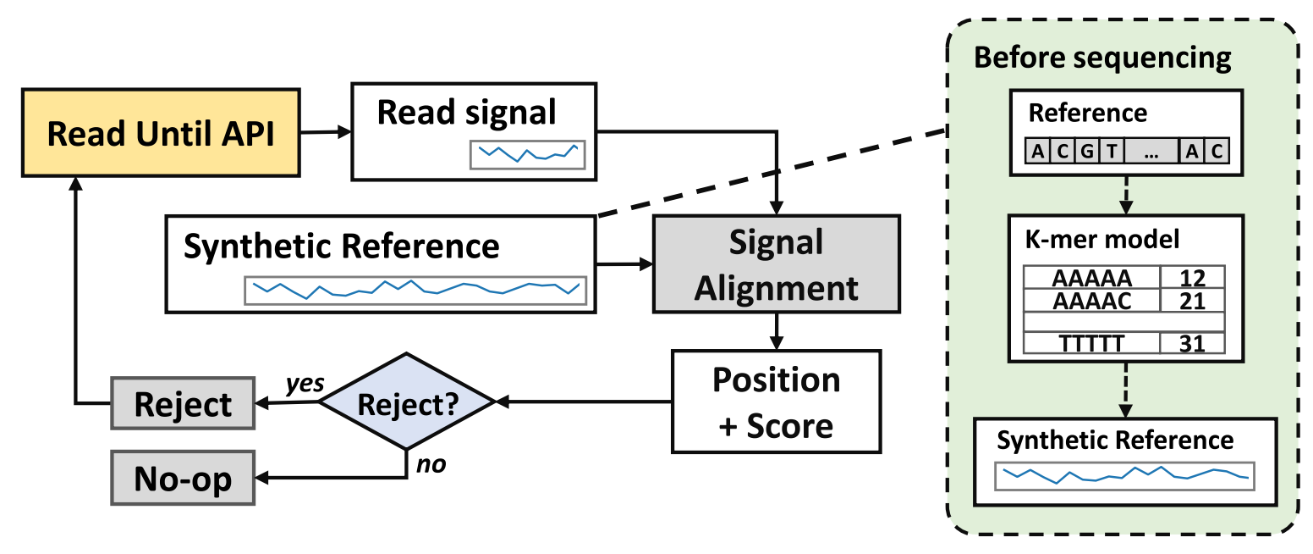
1. *Channel starts sequencing a DNA strand*
2. *Host machine performs* ***real-time analysis*** *as we sequence*
3. *If analysis results is to "skip", reject the strand*
4. *If analysis results is to "continue", let the pore finish sequencing*

Typical analysis flow


Partial DNA Strand (a.k.a. squiggle)
map/align
Genome Reference
Position of alignment
✓ Wait until strand fully sequenced
✗ Reject the current strand

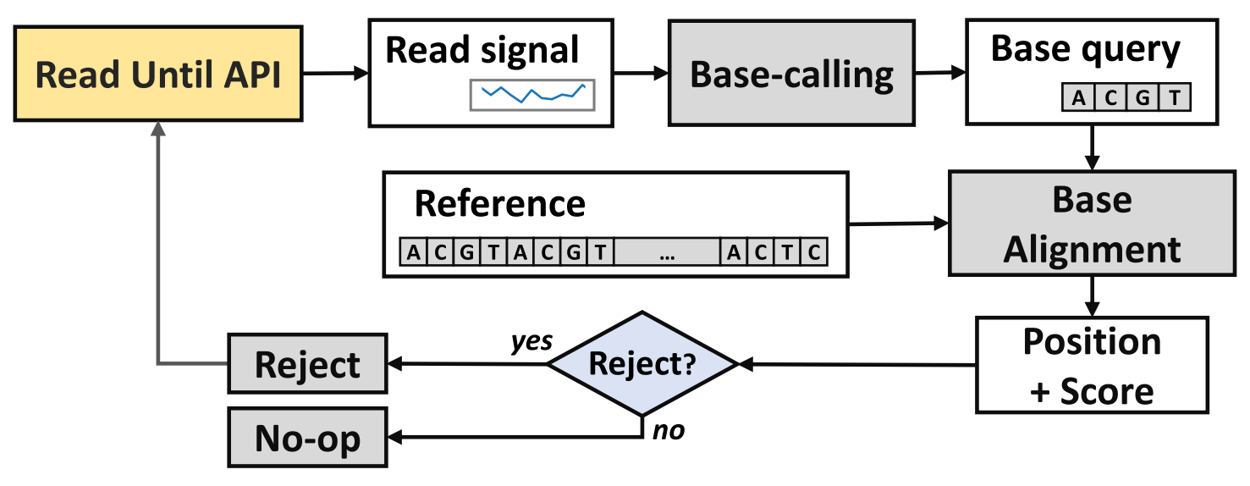
Problem Existing analysis pipelines are compute-intensive → *costly compute hardware requirements!*

Signal-alignment Read Until



→ Doesn't keep up with the sequencing rate

Base-alignment Read Until



→ Requires high-performance computing systems & high-end GPUs

High energy consumption
Not portable at all
Expensive and inaccessible

Our goal Real-time analysis on an SoC for Read Until on Nanopore Sequencers!

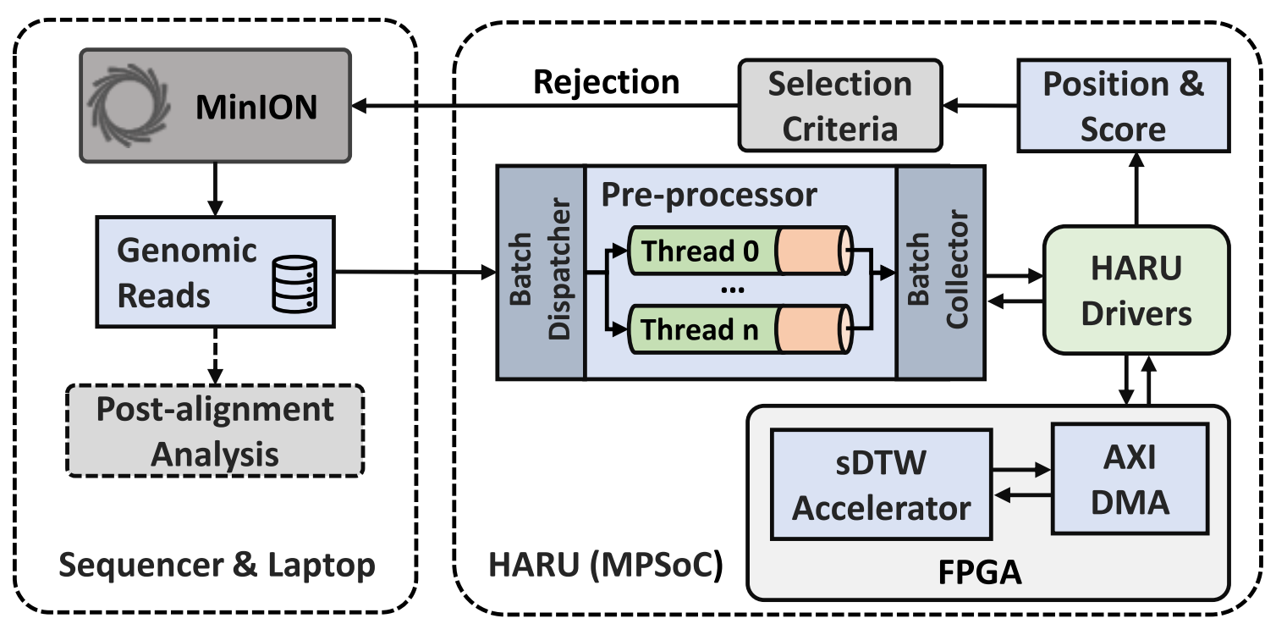
→ Low energy
→ Ultra portable and accessible
→ Scalable & adaptive to future sequencing advancements
→ Fully working from end-to-end
→ HIGH THROUGHPUT!

Hardware
Accelerated
Read
Until

System Design and Methodologies

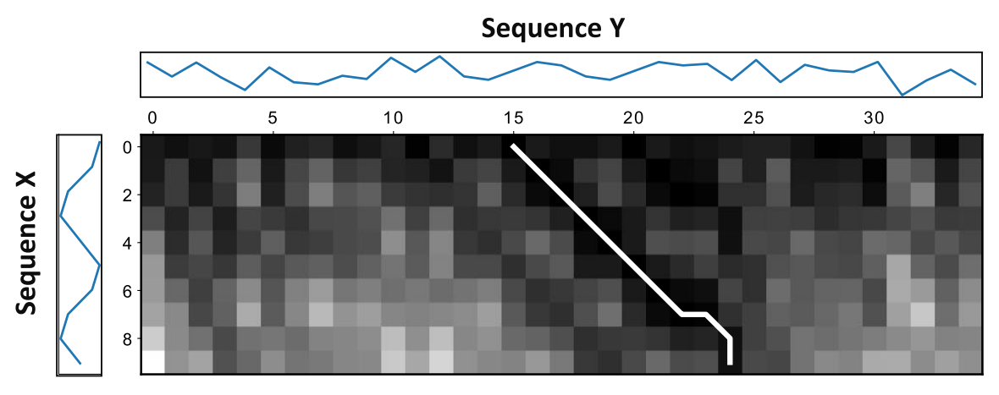
HARU Overview

HARU is a **signal-alignment** hardware-software co-design pipeline for Read Until. It features an efficient **subsequence dynamic time warping (sDTW) hardware accelerator** running in the FPGA of an AMD Zynq MPSoC (an SoC with an ARM processor and FPGA).



Subsequence DTW

Finds the optimal (warped) alignment of a subsequence to reference
→ sequences are represented in raw signal values (as opposed to bases)
→ computationally intensive! Naively quadratic for both space and time complexity
→ takes up 98% of processing time for signal-alignment Read Until!

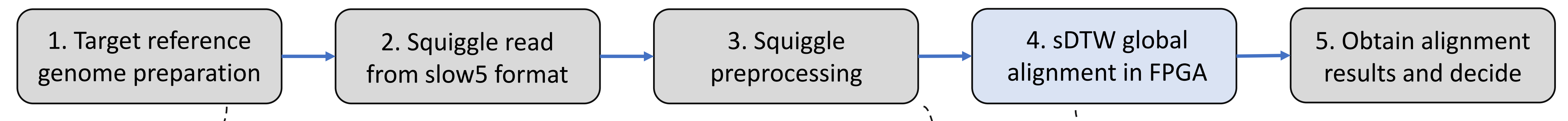


Base-alignment (e.g. SmithWatermann) execution is similar to signal-alignment but requires basecalling (expensive). With enough optimization and acceleration effort, signal-alignment is more direct!

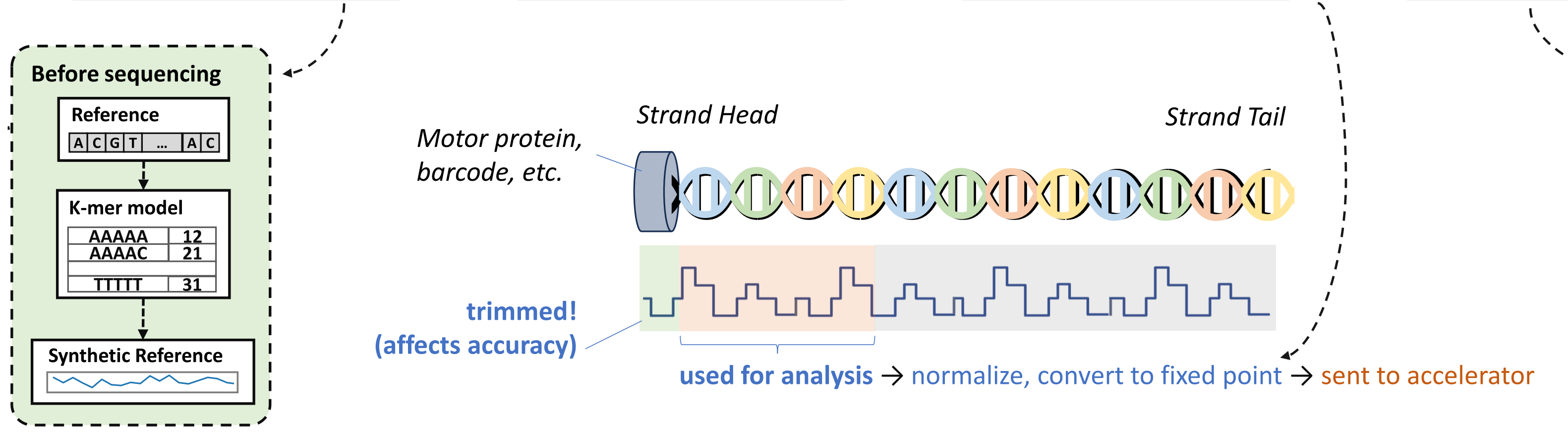
Algorithm 1: Subsequence DTW

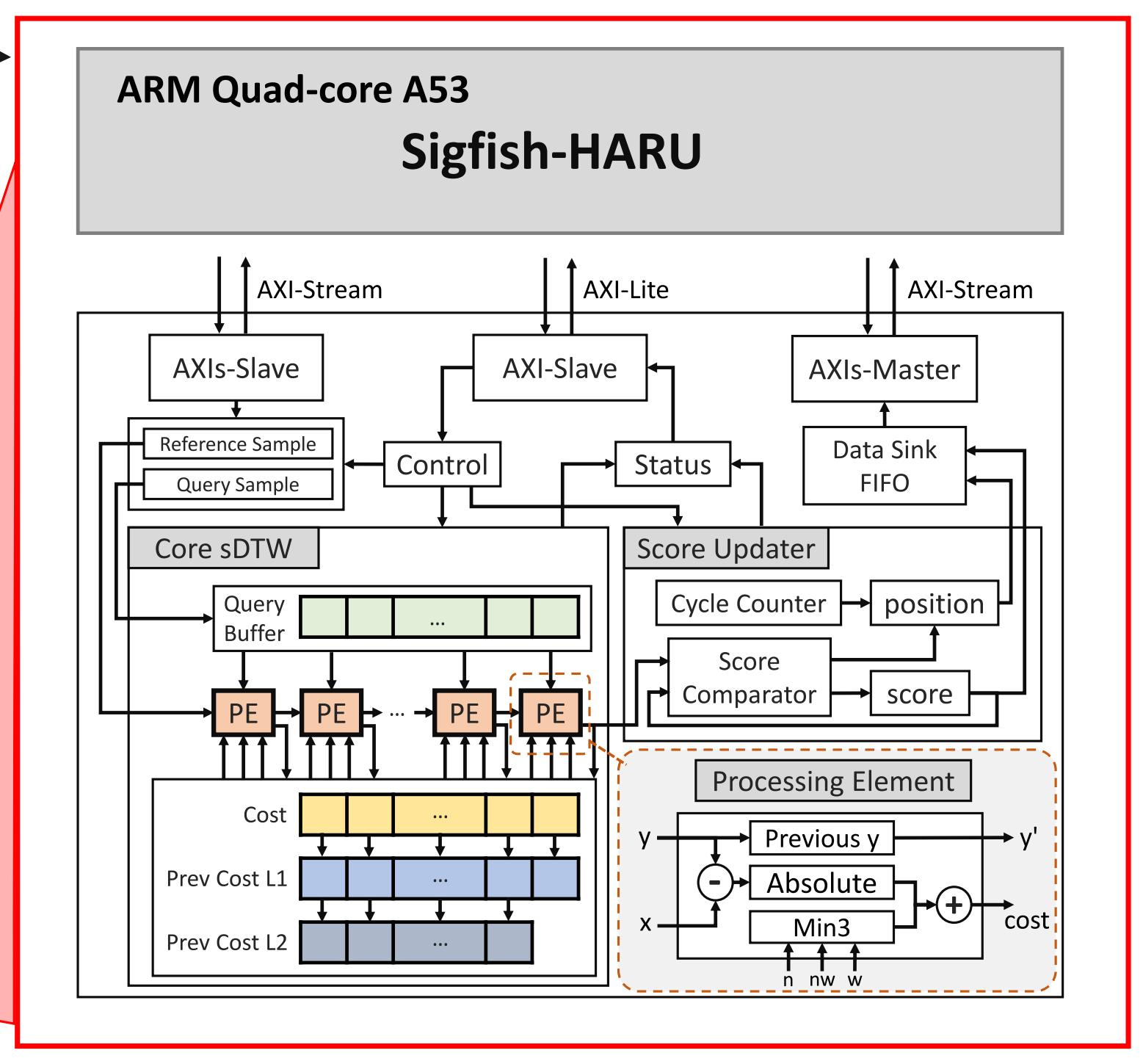
```
Input  : $X[1:M], Y[1:N], M, N$
Output: position, score
C: cost matrix of size $M * N$;
score ← ∞;
position ← −1;
for j in range 1 to N do
    $C[1, j] \leftarrow abs(X[1] - Y[j])$;
end
for i in range 2 to M do
    $C[i, 1] \leftarrow abs(X[i] - Y[1]) + C[i-1, 1]$;
end
for i in range 2 to M do                         data dependency!
    for j in range 2 to N do
        $d \leftarrow min(C[i-1, j], C[i, j-1], C[i-1, j-1])$;
        $C[i, j] \leftarrow abs(X[i] - Y[j]) + d$;
    end
end
min_score ← C[M, 1]for j in range 2 to N do
    if C[M, j] < min_score then
        position ← j;
        score ← C[M, j];
    end
end
```

HARU Co-Design Execution Breakdown



1. Target reference genome preparation → 2. Squiggle read from slow5 format → 3. Squiggle preprocessing → 4. sDTW global alignment in FPGA → 5. Obtain alignment results and decide


Before sequencing
Reference
A C G T ... A C
K-mer model
AAAAA 12
AAAAC 21
TTTTT 31
Synthetic Reference
Motor protein, barcode, etc.
Strand Head
Strand Tail
trimmed! (affects accuracy)
used for analysis → normalize, convert to fixed point → sent to accelerator



HARU's sDTW accelerator

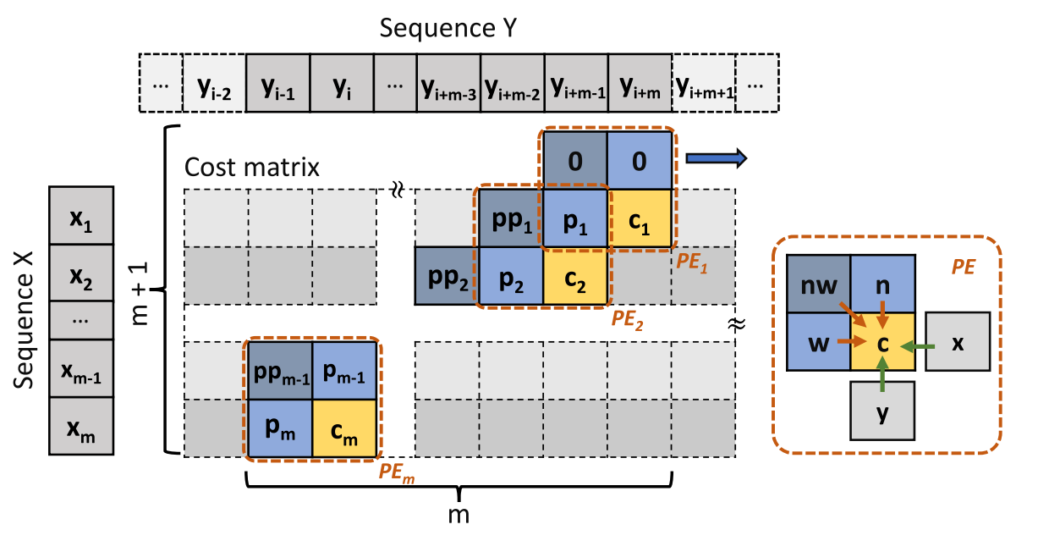
Optimizations

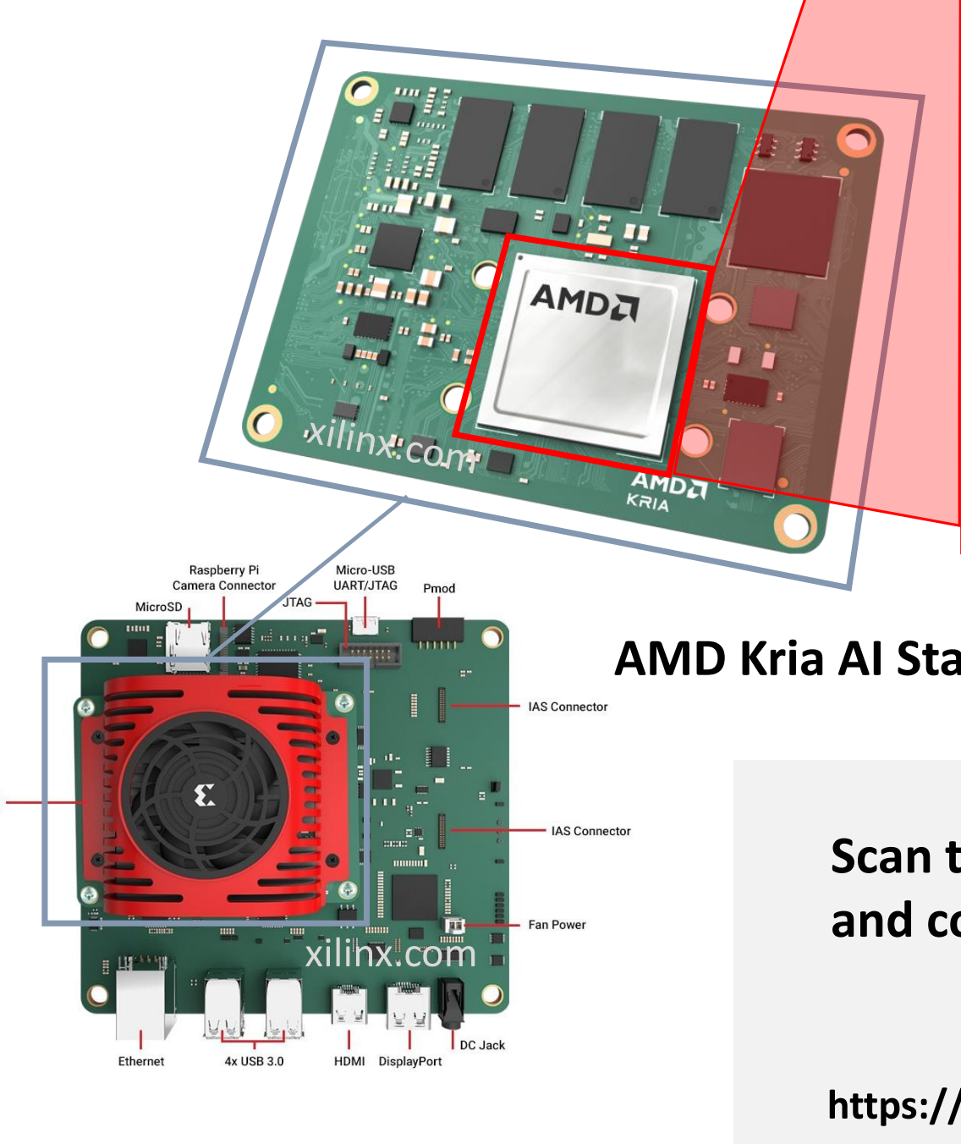
→ Column cells computed in parallel (via pipelining)
→ fixed-point data & tuned scaling factor (reduce computation cost)
→ data reusing, no backtracing (keeping only necessary data of the DP memoization)

Becomes a chain of Processing Elements (PEs) sliding through the reference (linear)

Algorithm 2: Memory-efficient subsequence DTW

```
Input  : $X[1:M], Y[1:N], M, N$
Output: position, score
C: array of size $M + 1$ initialised to ∞;
score ← ∞;
position ← −1;
for j in range 1 to N do
    n ← 0;
    nw ← C[1];
    w ← C[2];
    for i in range 1 to M do
        $C[i] := abs(x[i] - y[j]) + min(n, nw, w)$;
        n := C[i];
        nw := w;
        w := C[i + 2];
    end
    if C[M] < score then
        position ← j;
        score ← C[M];
    end
end
```





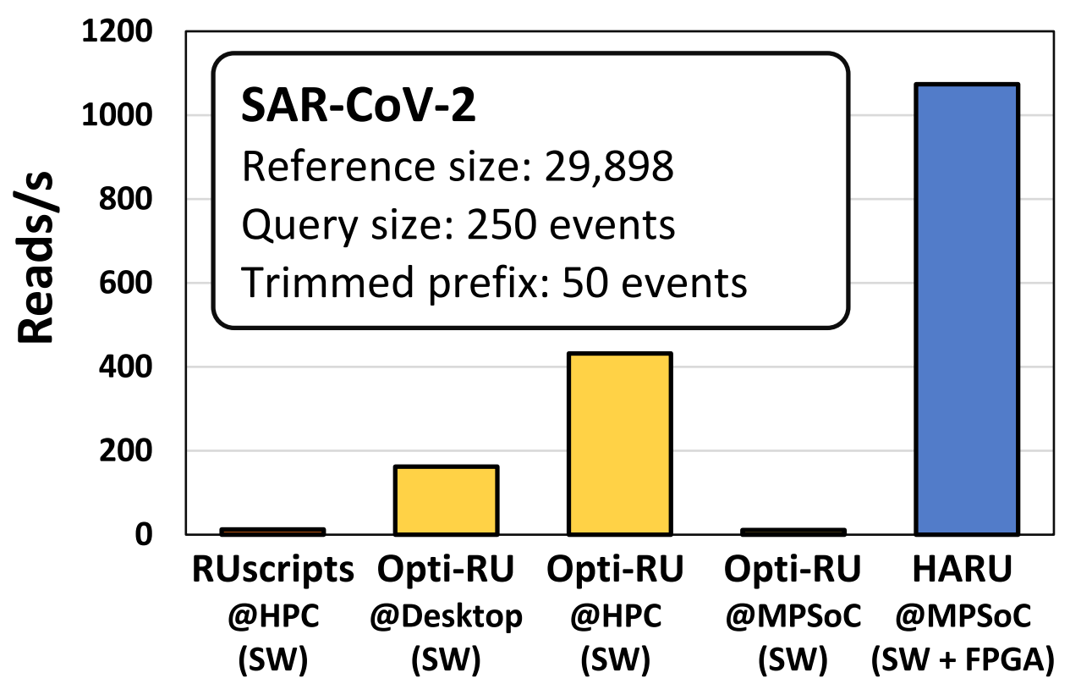
AMD Kria AI Starter Kit (249 USD)

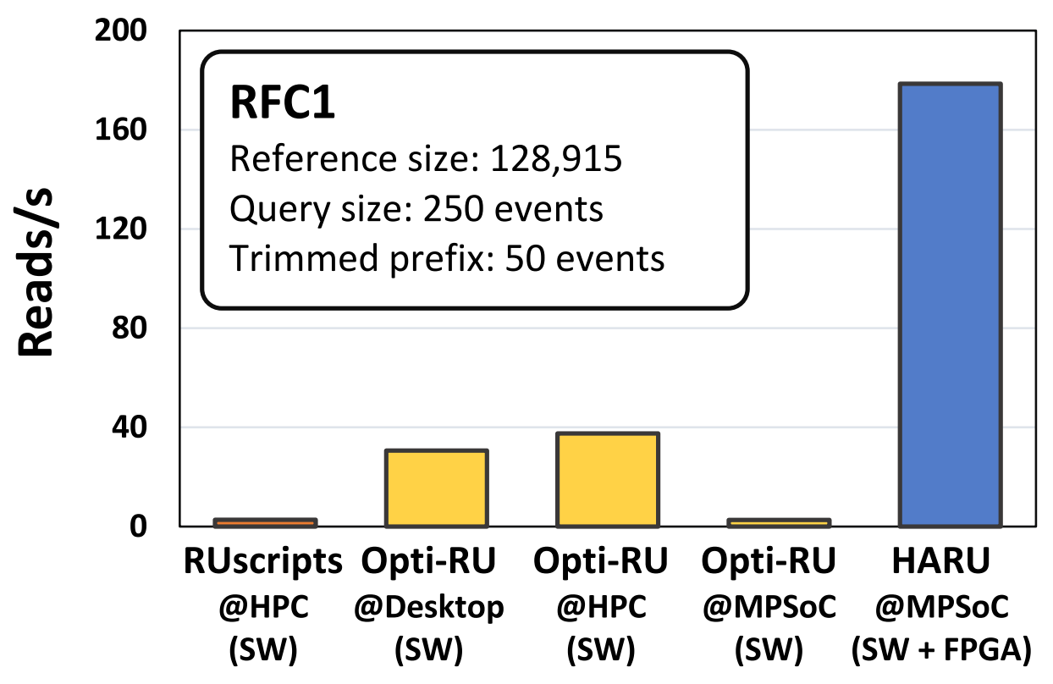
Scan to access paper and codebase

https://doi.org/10.1093/gigascience/giad046 HARU Sigfish-HARU

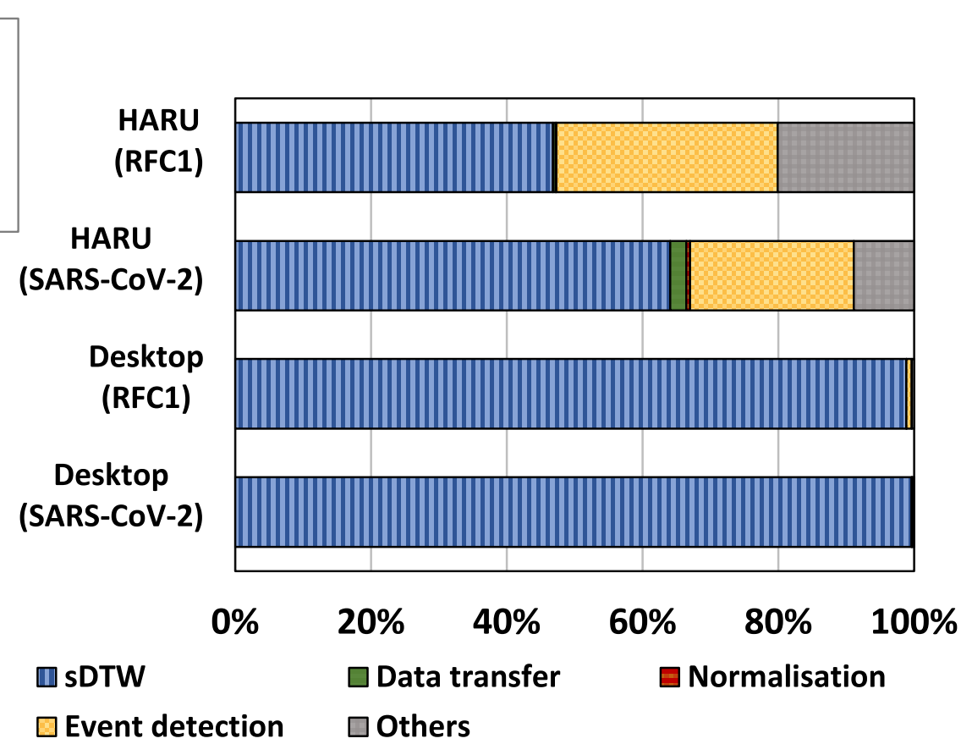
Results and Outcomes

Processing capability
(HARU vs sDTW implementations)





Desktop - Intel Core i9-10850K (10 cores, 32GB RAM)
HPC – Intel Xeon Gold 6154 (36 cores, 377GB RAM)
MPSoC – ARM Cortex A53 (4 cores, 4GB RAM)

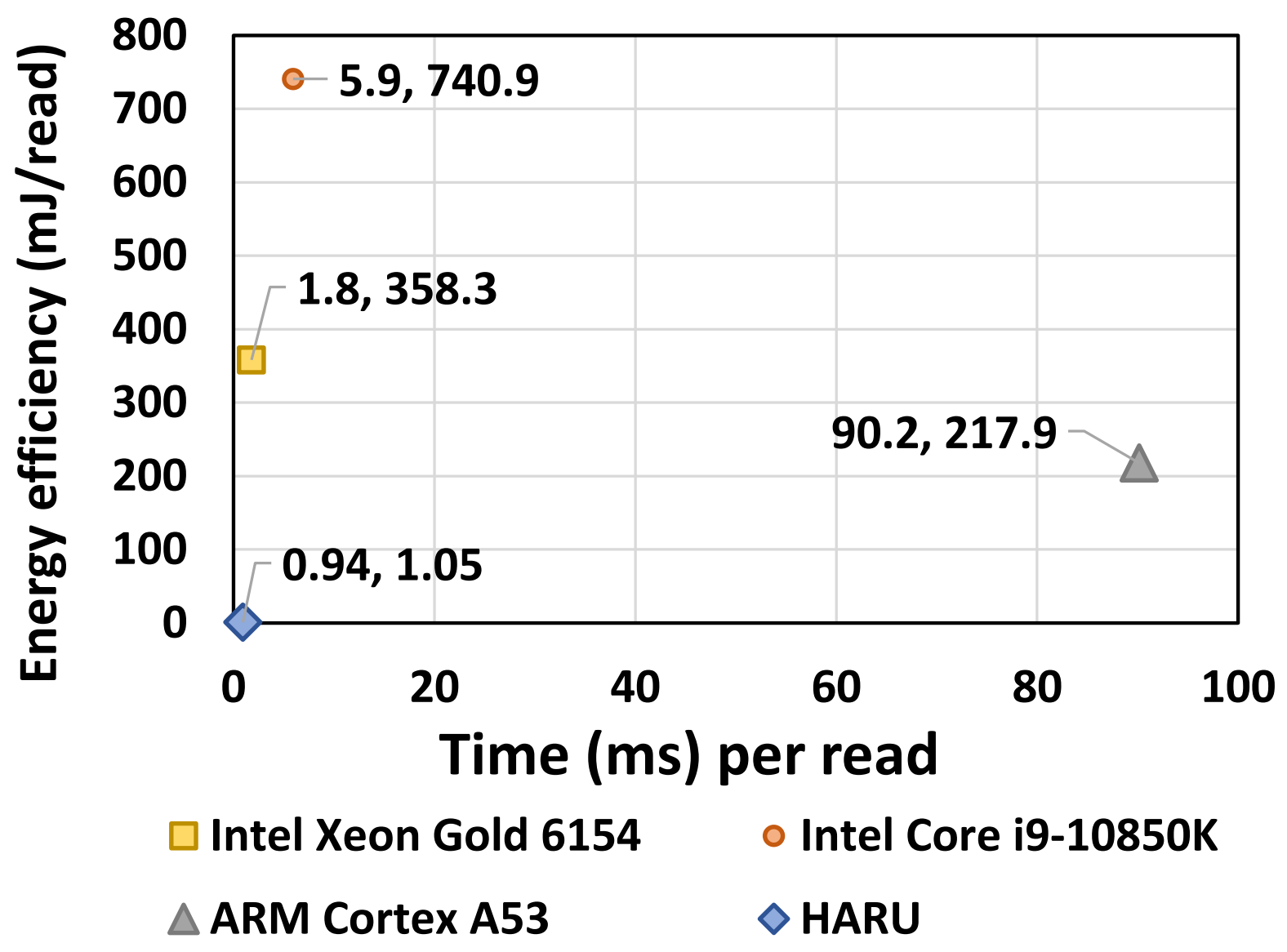


Speedups (HARU vs):

- RUScripts [1] @HPC→ 85.8 x
- Optimized RUScripts @Desktop → 6.6 x
- Optimized RUScripts @HPC → 2.5 x

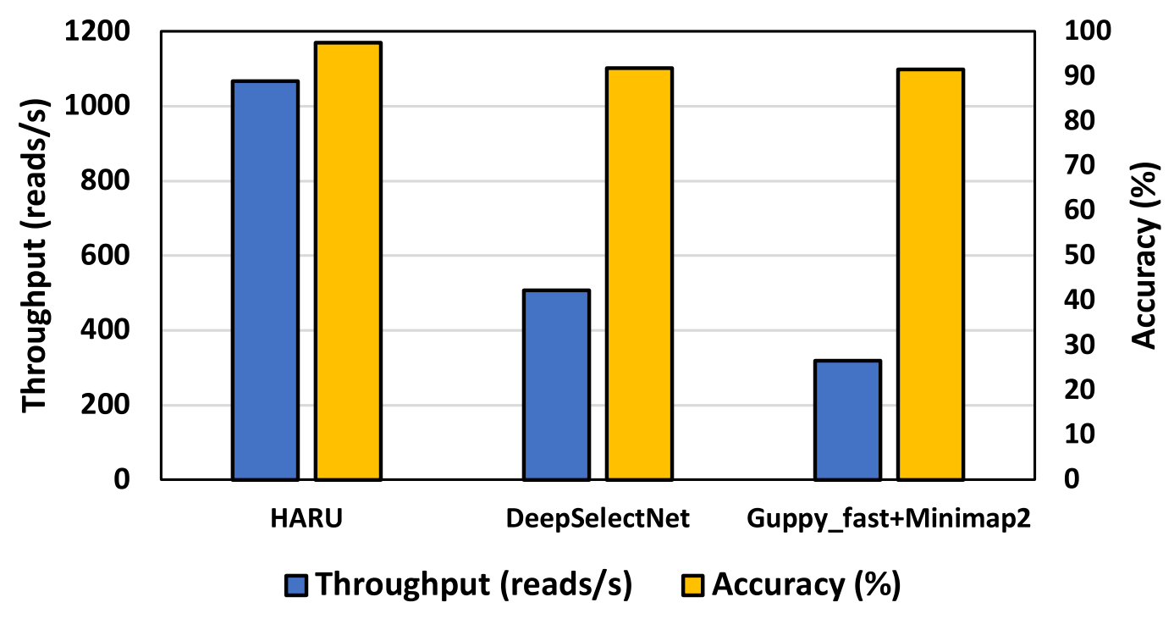
Smaller, cheaper, but higher throughput!

Energy-delay product
(HARU vs sDTW implementations)



Significant improvement in energy efficiency!

Processing capability & Accuracy
(HARU vs other SOTA implementations)



DeepSelectNet [2] running on HPC + Telsa V100 GPU
Guppy_fast + Minimap2 [3] running on Nvidia Jetson Xavier

The reduction of processing requirements enables in-field selective sequencing! + Fully working system!

Buy a KV260 board → Flash prebuilt binaries → RUN!

[1] Loose, M., Malla, S., & Stout, M. (2016). Real-time selective sequencing using nanopore technology. *Nature methods*, *13*(9), 751-754.
[2] Senanayake, A., Gamaarachchi, H., Herath, D., & Ragel, R. (2023). DeepSelectNet: deep neural network based selective sequencing for oxford nanopore sequencing. *BMC bioinformatics*, *24*(1), 31.
[3] Li, H. (2018). Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, *34*(18), 3094-3100.

Contact
eshih.pj@gmail.com