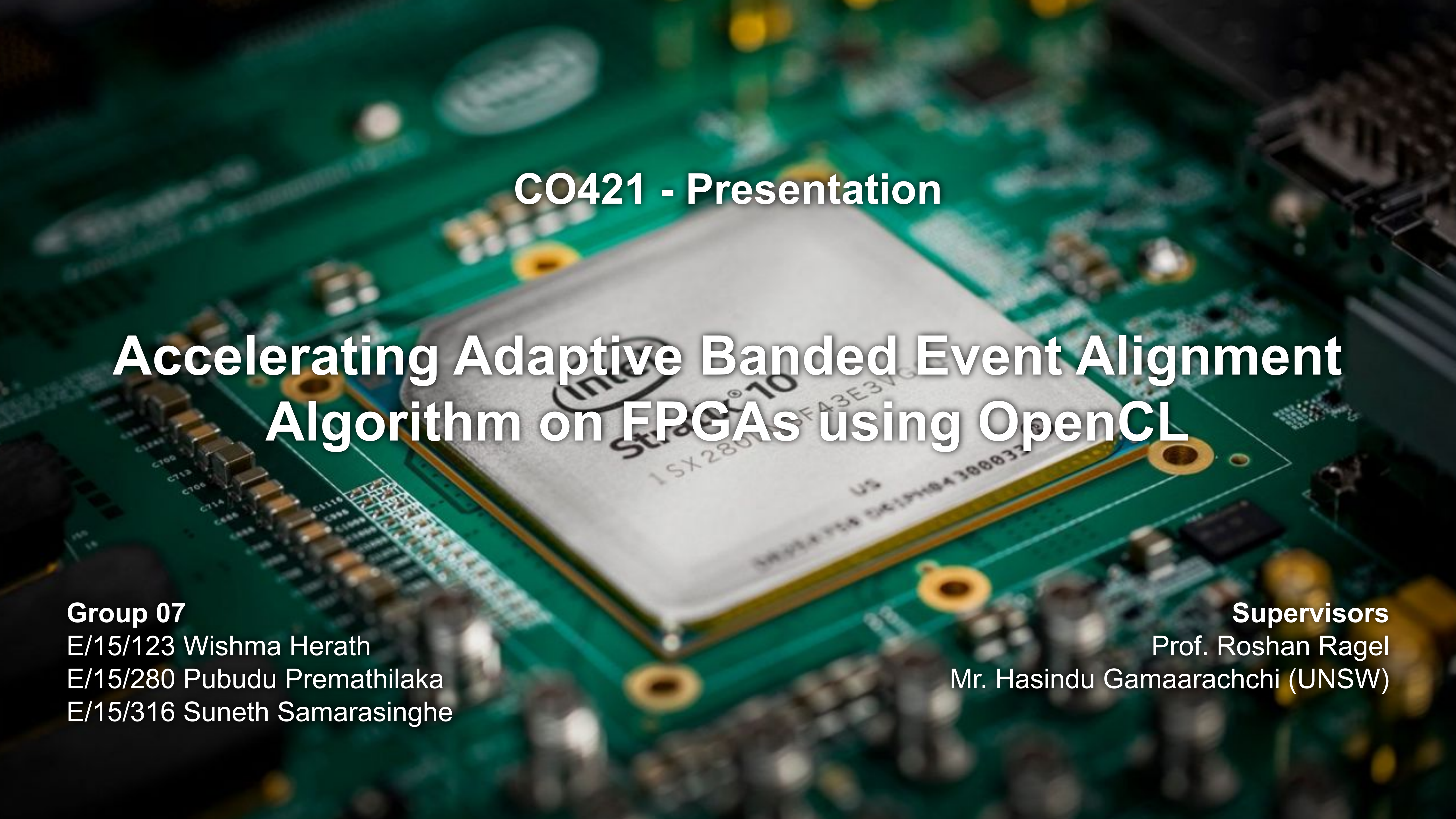


CO421 - Presentation

Accelerating Adaptive Banded Event Alignment Algorithm on FPGAs using OpenCL

Group 07
E/15/123 Wishma Herath
E/15/280 Pubudu Premathilaka
E/15/316 Suneth Samarasinghe

Supervisors
Prof. Roshan Ragel
Mr. Hasindu Gamaarachchi (UNSW)

Terminology

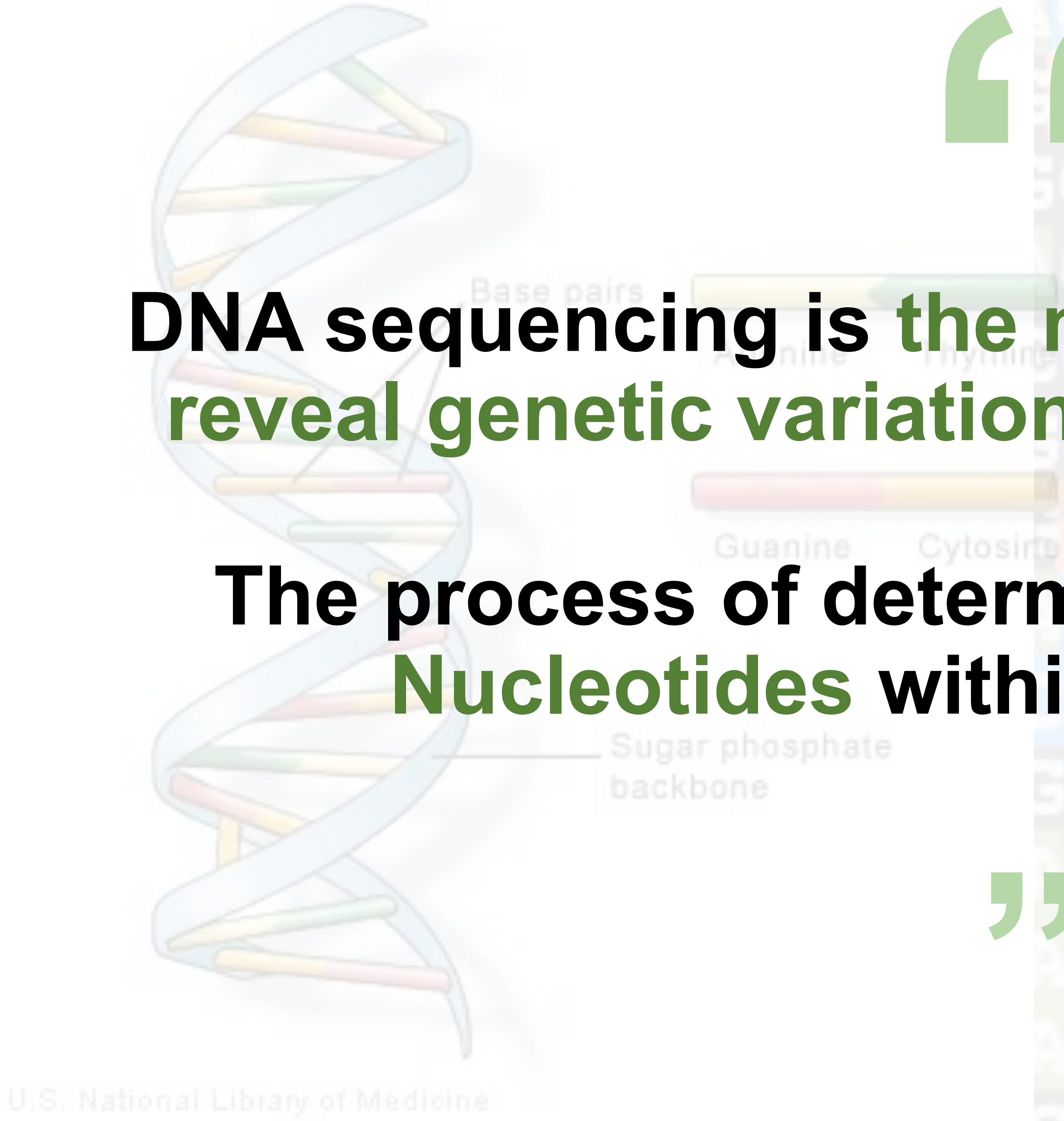
Genome : a long sequence composed of four types of nucleotide bases

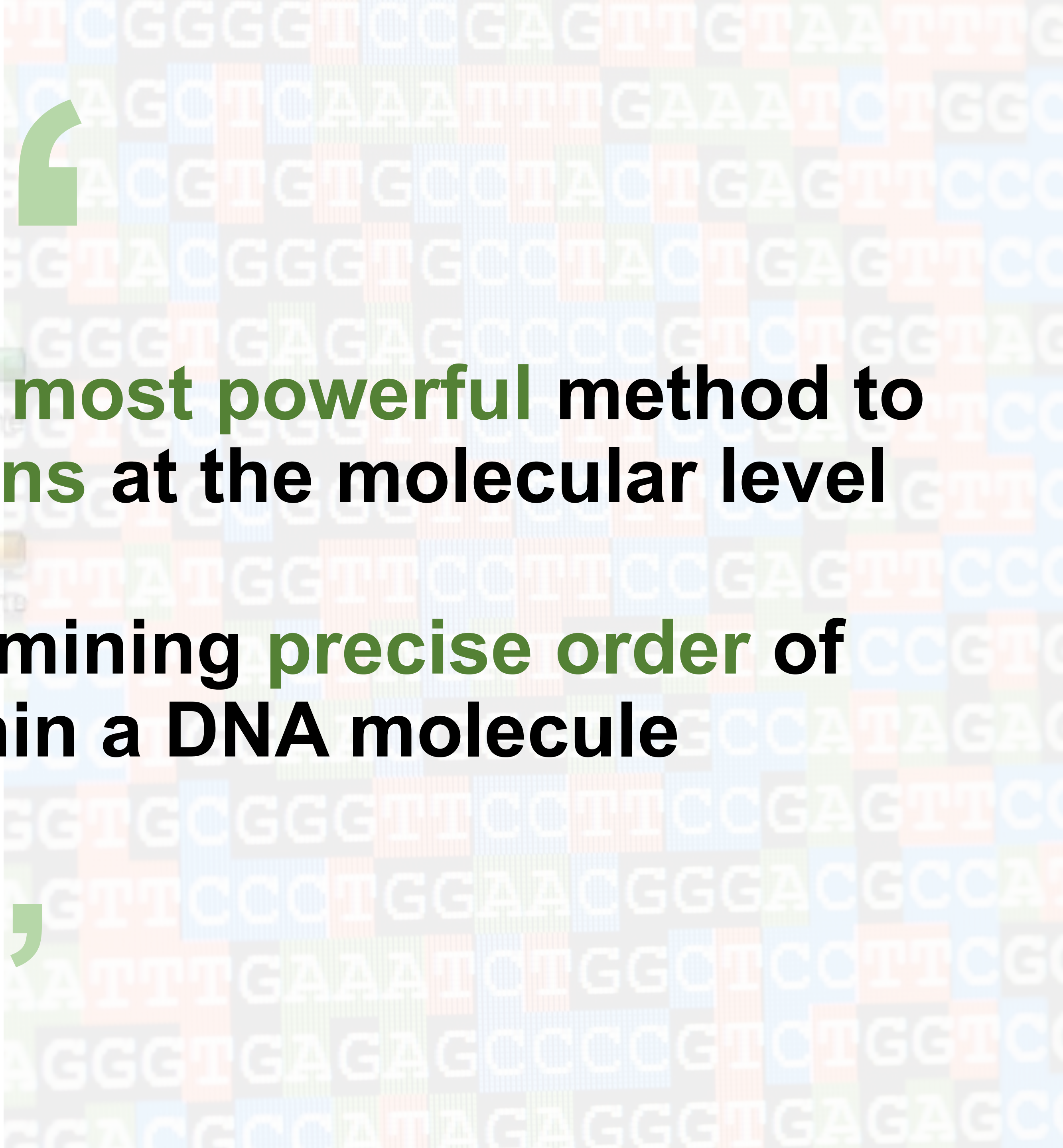
4 Bases : adenine (**A**), cytosine(**C**), guanine (**G**) and thymine (**T**)

Sequencing : the process of reading strings of contiguous bases

Reads : the resulting strings of bases from sequencing

Introduction





“

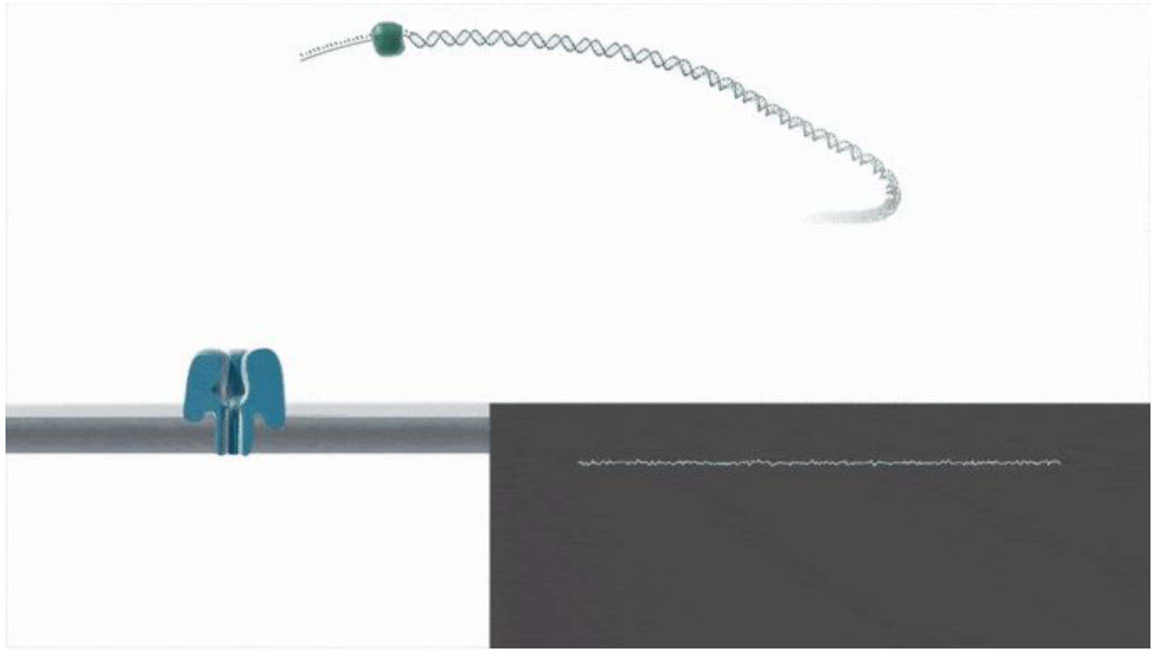
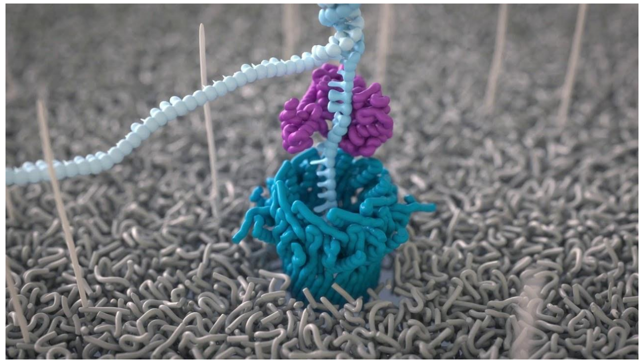
DNA sequencing is the most powerful method to reveal genetic variations at the molecular level

The process of determining precise order of Nucleotides within a DNA molecule

”

Oxford Nanopore Sequencing Technology (ONT)

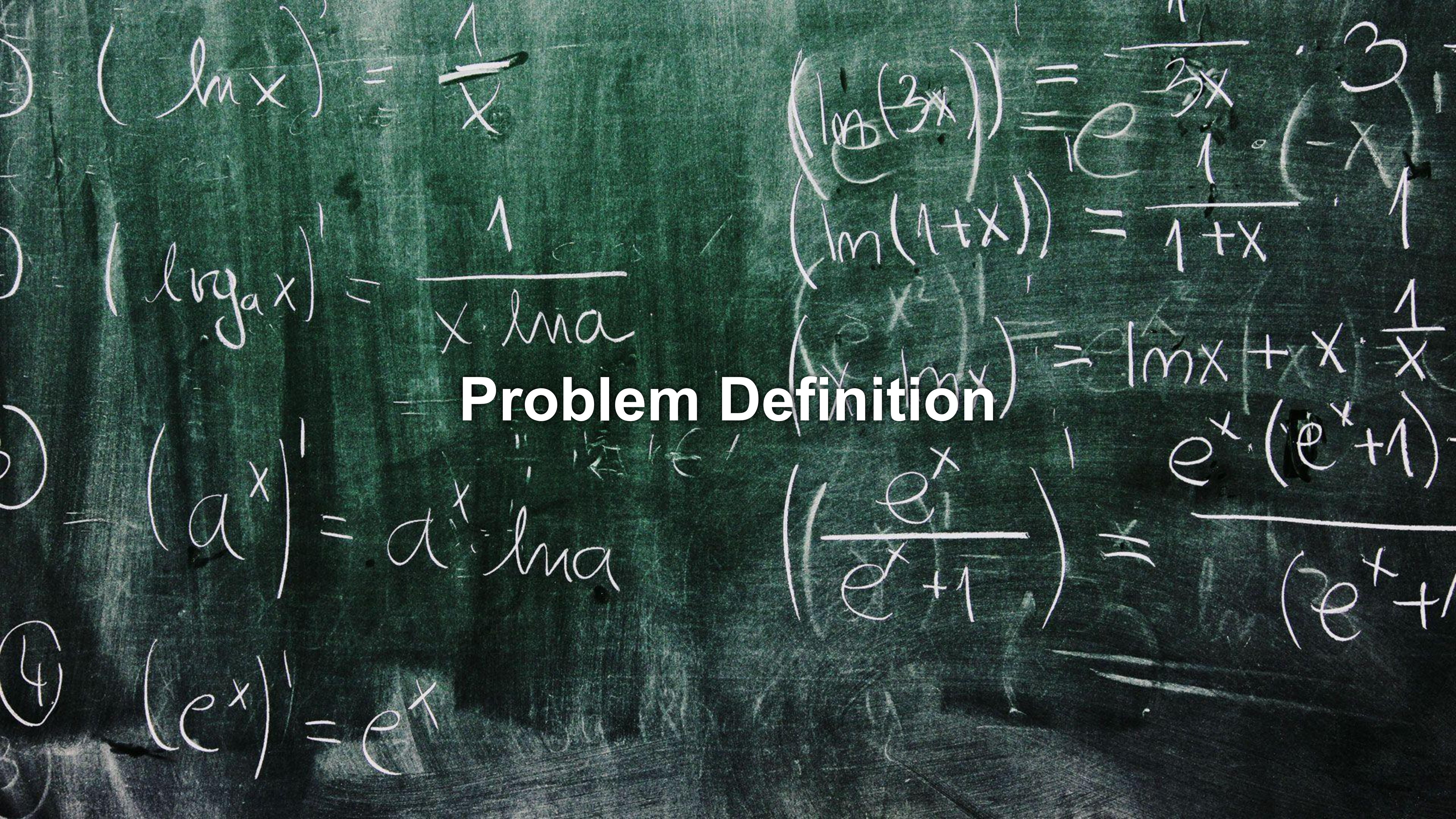
- An enzyme unwinds DNA feeding one strand through a nanometer size protein pore.
- Unique shapes of DNA bases cause disruption in electrical current

Oxford Nanopore Sequencing Technology (ONT)

- Base calling
- Sequence alignment
- Downstream analysis (Polishing)
  - Methylation calling:
    - Events detection
    - Signal space alignment
    - Hidden Markov model

$$\begin{tikzcd}
\arrow[r] & \text{Suzuki-Kasahara} \arrow[d] \\
\text{ABEA} & \text{Nanopolish} \arrow[l]
\end{tikzcd}$$



Problem Definition

ABEA is one of the most time consuming steps when analyzing raw nanopore data

~70%

of the total CPU time

Original ABEA

Nanopore long reads can be >1M bases long

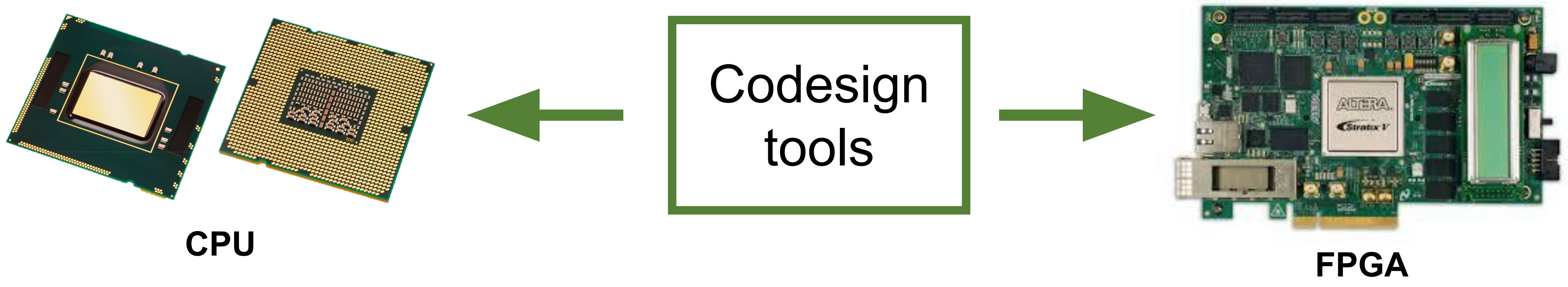
>10^{12} computations
Hundreds of GB of RAM
Takes ~ 48-64 hours

A fine-tuned version of ABEA is used in a recent work (called f5c by Hasindu Gamaarchchi et al.) - an optimized version of nanopolish to run on general purpose GPUs and CPUs.

According to the literature

FPGAs are more power efficient relative to GPUs and provides reasonable performance for HPC

Literature suggests to do a Hardware-software codesign to get the best performance



Related work (literature review)

Takeaways from FPGA based accelerations using OpenCL (1/5)

From SWIFOLD by Rucci et al., 2018

- Use of smaller data types for kernel(Eg: ALMs Usage: int(32 bit) 89%, short(16-bit) 52%)
- Larger pipelines
- In SWIFOLD, regardless of the sequence length and sequence similarity they have achieved higher average performance
- The exploitation of OpenCL memory hierarchy has offered a considerable benefit in performance.

Takeaways from FPGA based accelerations using OpenCL (2/5)

From SW Protein Search by Rucci et al., 2015

- Data level parallelism

From KNN implementation by Fahad et al., 2016

- FPGA offers better power/ energy efficiency compared to GPU implementation

Takeaways from FPGA based accelerations using OpenCL (3/5)

Design of FPGA-based computing systems with openCL by Waidyasooriya et al., 2017

- **Performance improvement techniques for NDRange kernels**
    - Specifying the Work-Group Size (Compiler level optimization)
    - Kernel Vectorization (SIMD)
    - Increasing the Number of Compute Units

Takeaways from FPGA based accelerations using OpenCL (4/5)

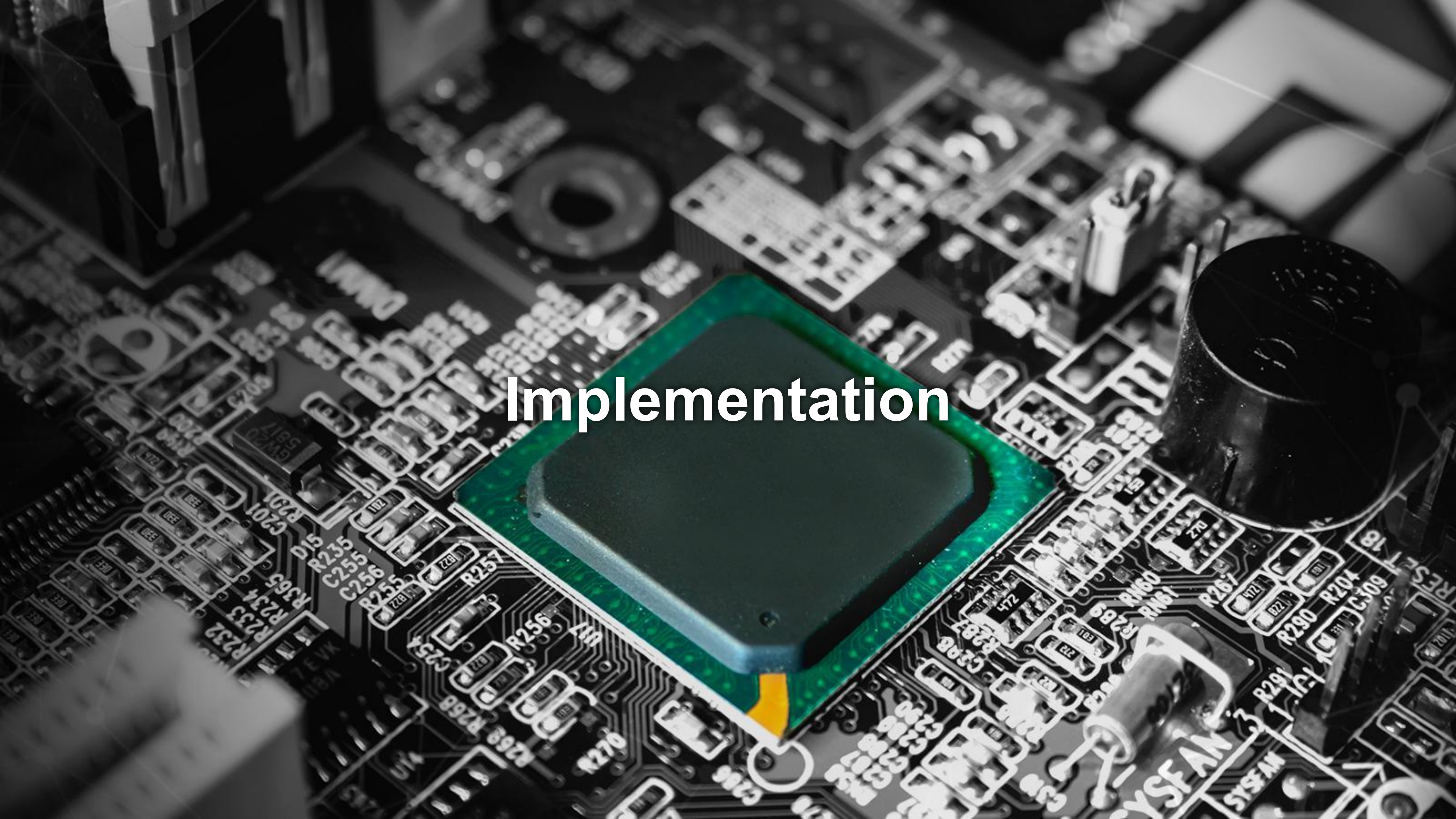
Design of FPGA-based computing systems with openCL by Waidyasooriya et al., 2017

- **Performance improvement techniques for Single Work Item kernels**
    - Avoiding Nested Loops
    - Reducing Initiation Interval Due to Read-Modify-Write Operations to Global Memory
    - Ignore Loop-Carried Dependencies

Takeaways from FPGA based accelerations using OpenCL (5/5)

Design of FPGA-based computing systems with openCL by Waidyasooriya et al., 2017

- **Common performance improvement techniques**
    - Use of Loop unrolling
    - Inserting the ‘*restrict*’ keyword in pointer arguments whenever possible



Implementation

Implementation Choices

- **OpenCL**
  - OpenCL allows a programmer to use various compute devices(FPGA,GPU,CPU)
  - HDL is a time consuming and more complex. Therefore, HLS tools are favourable
  - Most of the HLS tools does not support interface between FPGA and CPU. Designer still needs to use HDL to design interface circuit
  - OpenCL allows designers to describe whole computation: computation on the host, data transfer between the host and accelerators, and computation on accelerators
  - Board Support Package(BSP) facilitates to use the same OpenCL code to different FPGA boards
- **FPGA**
  - Efficient power consumption when working as an accelerator
  - Reconfigurable hardware provides more flexibility

Tools, devices and technologies

- Altera Stratix V GX (DE5-NET)
- Intel FPGA SDK for OpenCL
- Intel SDK for OpenCL
- Quartus Prime
- Visual Studio

Implementation (1/2)

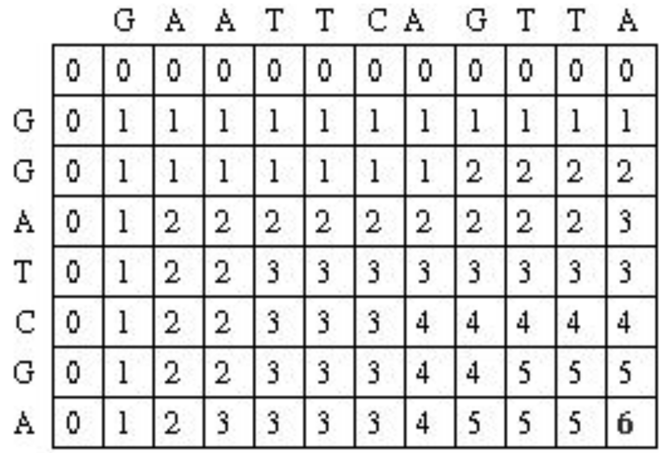
ABEA

Input:

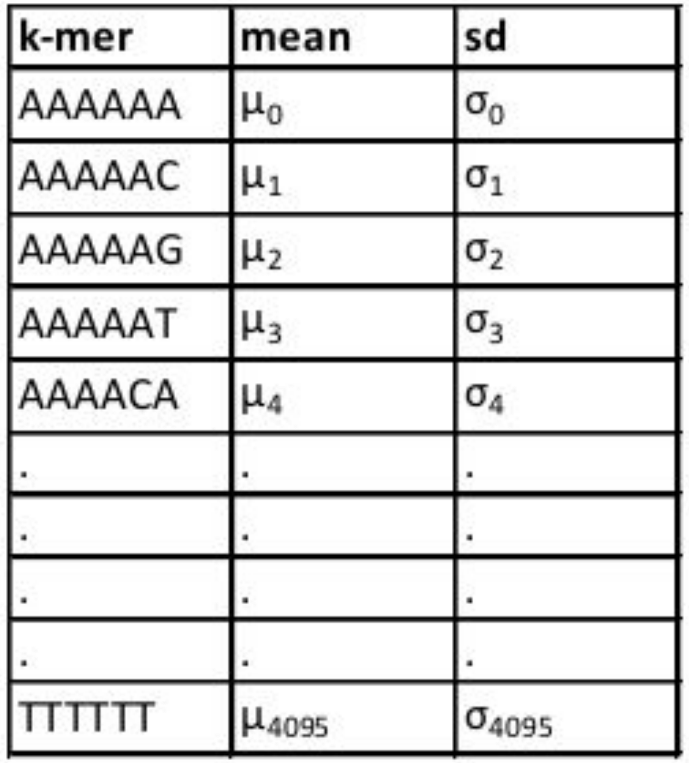
- ref [] : base-called read (1D char array)
- model : pore-model
- events [] : the output of the event detection step

Output:

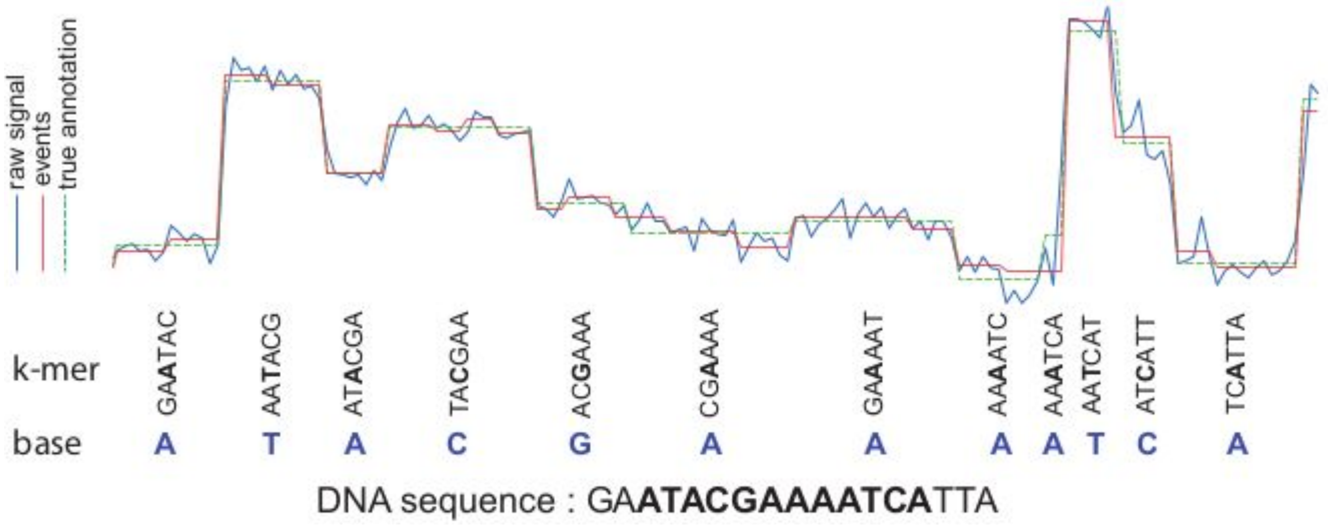
- alignment [] : list of {event index, k-mer index}



		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0	1	1	1	1	1	1	1	1	1	1	1
G	0	1	1	1	1	1	1	1	2	2	2	2
A	0	1	2	2	2	2	2	2	2	2	2	3
T	0	1	2	2	3	3	3	3	3	3	3	3
C	0	1	2	2	3	3	3	4	4	4	4	4
G	0	1	2	2	3	3	3	4	4	5	5	5
A	0	1	2	3	3	3	3	4	5	5	5	6



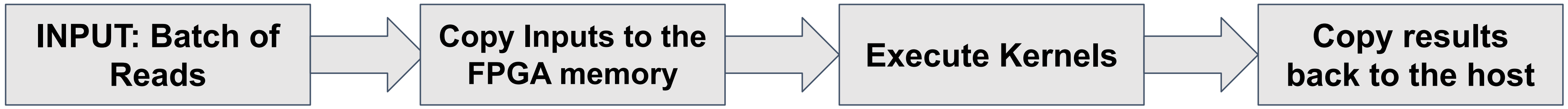
k-mer	mean	sd
AAAAAA	μ_0	σ_0
AAAAAC	μ_1	σ_1
AAAAAG	μ_2	σ_2
AAAAAT	μ_3	σ_3
AAAACA	μ_4	σ_4
.	.	.
.	.	.
.	.	.
.	.	.
TTTTTT	μ_{4095}	σ_{4095}



Implementation (2/2)

Phase 1 (Related to F5c)

- Flow of Execution

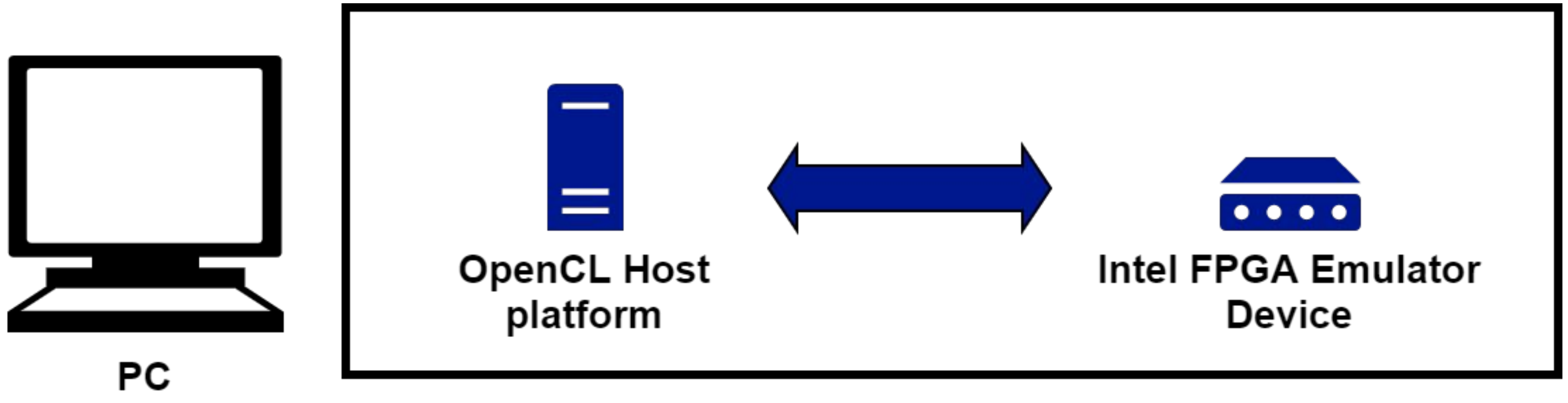


- Alignment function is divided into 3 kernels:
  1. **Pre-Kernel** : Initialising the first two bands of the dynamic programming table and pre-computing frequently accessed values by the next kernel.
  2. **Core-Kernel** : The filling of the dynamic programming table which is the compute intensive portion of the ABEA algorithm.
  3. **Post-Kernel** : Performs backtracking.

- Memory Pre-allocation

Evaluation

Experimental setup



Host specifications

PLATFORM NAME	Intel(R) FPGA SDK for OpenCL(TM)
PLATFORM VERSION	Version 18.0
CPU	Intel Core i5-4200H 2.80GHz x 2
RAM (GB)	12

Device specifications

DEVICE NAME	Emulated Device
DEVICE VERSION	OpenCL 1.0 Intel(R) FPGA SDK for OpenCL(TM), Version 18.0
DEVICE MAX CLOCK	1000 MHz
DEVICE GLOBAL MEM SIZE	12 GB
DEVICE MAX CU	1

Dataset

Publicly available reads that aligned to a 2kb region in the E. coli draft assembly

Sample	E. coli str. K-12 substr. MG1655
Instrument	MinION sequencing R9.4 chemistry
Basecaller	Albacore v2.0.1
Region	”tig00000001:200000-202000”
Note	Ligation-mediated PCR amplification performed

Dataset	**Number of reads**	**Number of bases**	**Mean read length (Bases)**	**Max read length (Bases)**
testset	143	819102	5727	12618

Execution time (1/2)

OpenCL implementation on Intel FPGA emulator

Kernel	**Execution time (s)**	**Percentage**
Pre-kernel	0.125	0.008%
Core-kernel	1501.740	99.922%
Post-kernel	1.045	0.070%

Explanation

The Core-kernel takes the longest time of around 99.9% of the total execution time since it is the most compute intensive step

Execution time (2/2)

Implementation	Total execution time (s)
OpenCL implementation on DE5net FPGA Emulator	1503.126
CPU implementation on Host PC	6.187

around 240x slowdown

Explanation

Intel FPGA emulator has limited resources such as 1 GHz maximum clock frequency and one compute unit(CU) compared to a physical FPGA hardware. Since the PC acts as both the Host and the Emulator Device, the resources are allocated for both host platform and the emulator device.

Comparison with CPU implementation

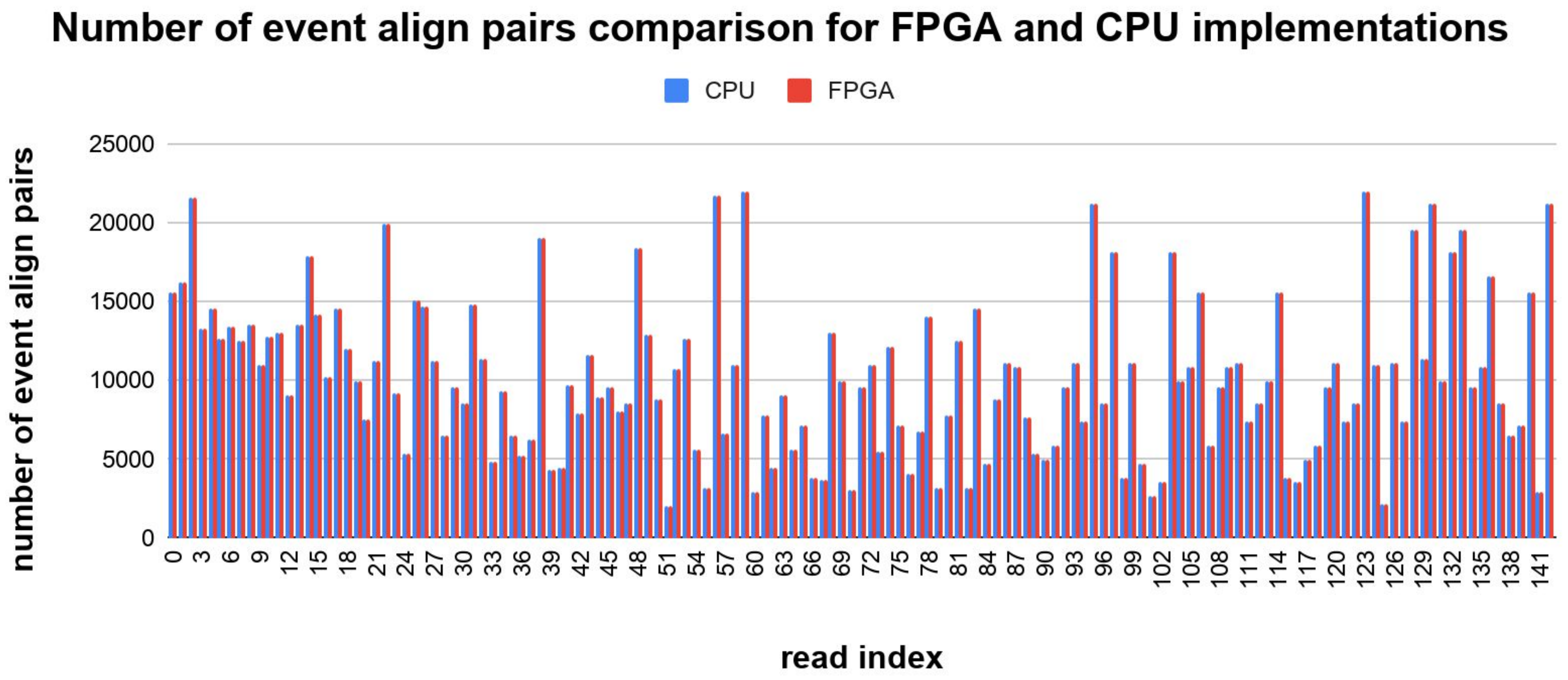
Output

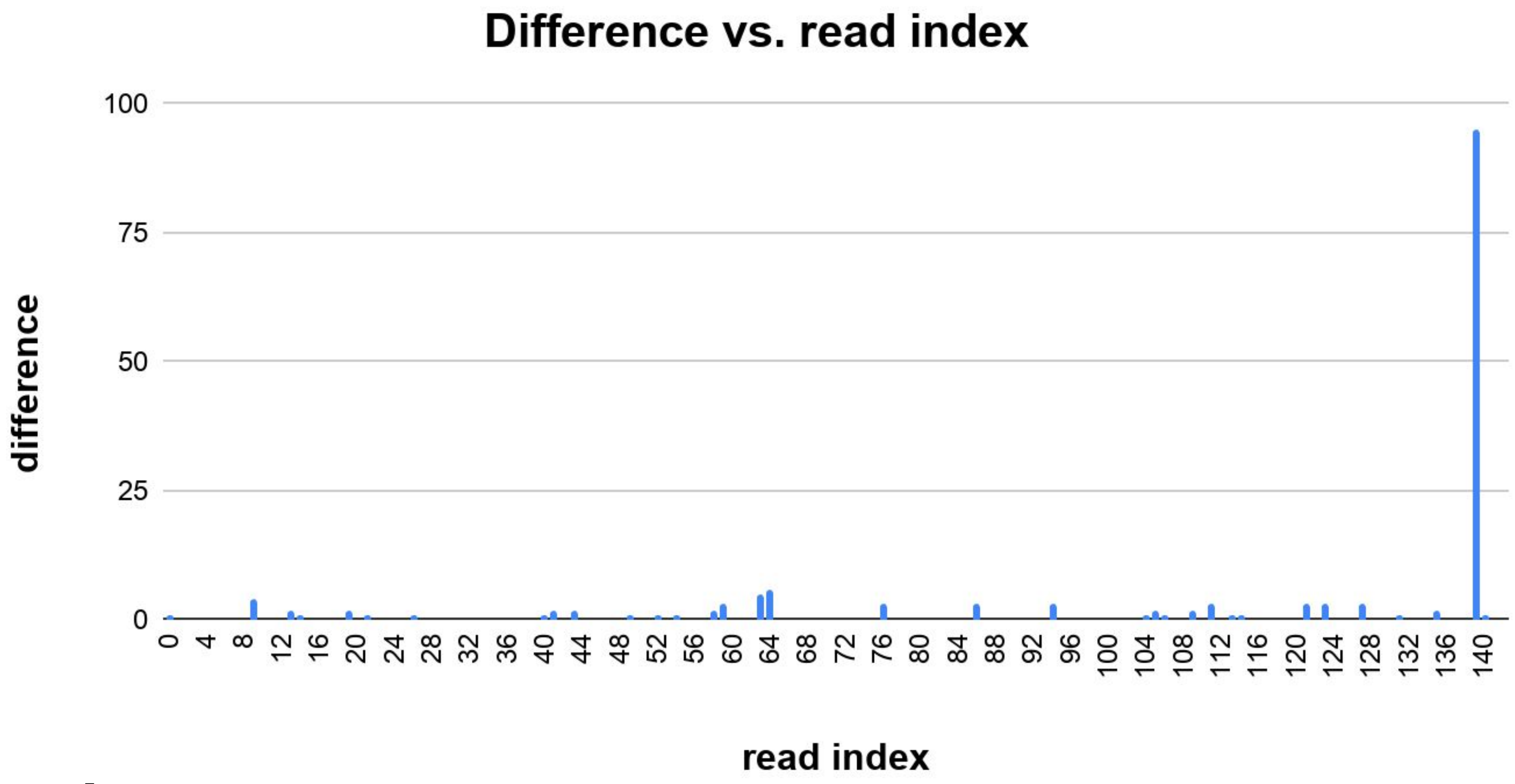
Per read:

N: number of event align pairs

reference position	read position
x1	y1
x2	y2
x3	y3
...	...

N


Number of event align pairs comparison for FPGA and CPU implementations
CPU
FPGA
number of event align pairs
read index
25000
20000
15000
10000
5000
0

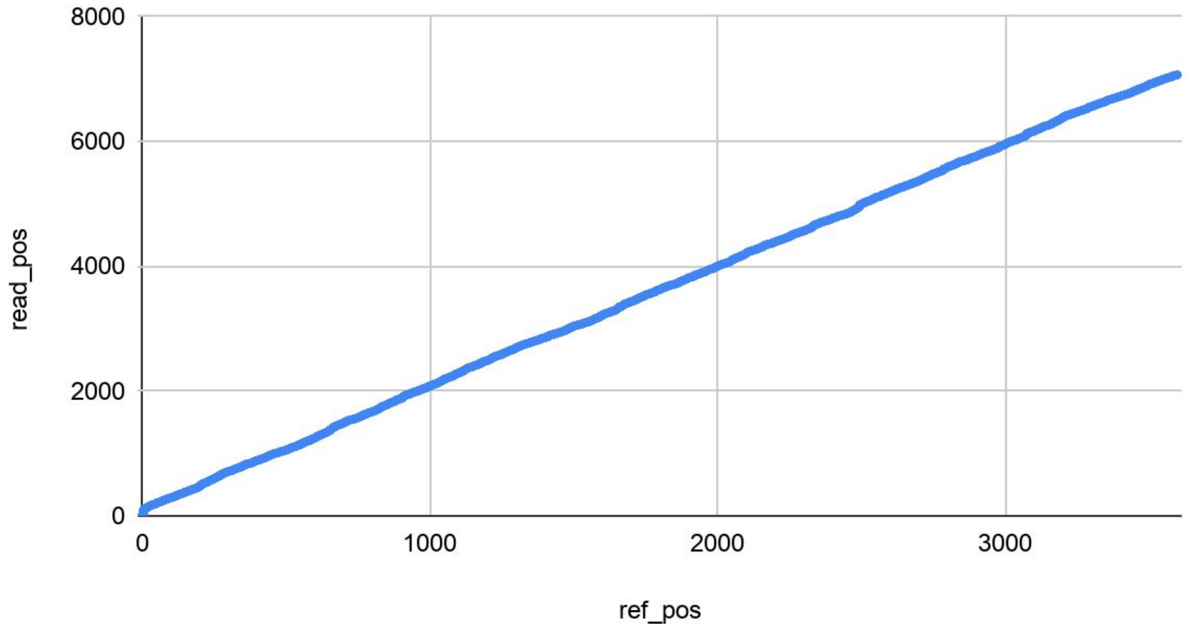


Explanation

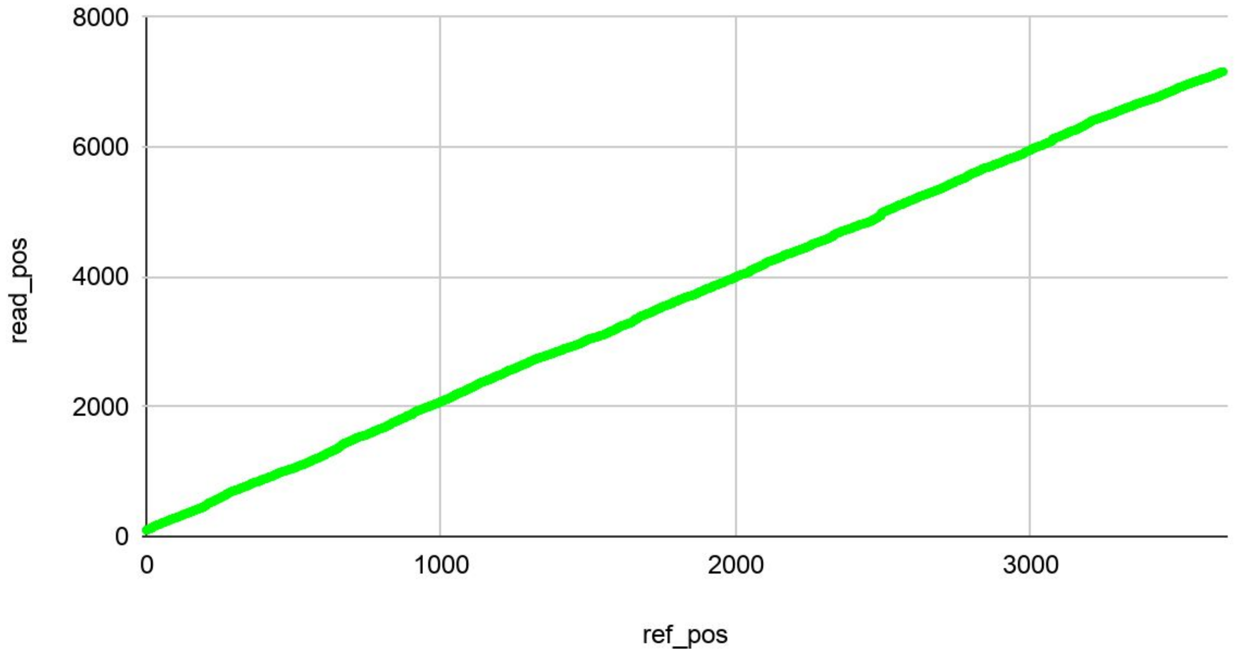
Almost all reads gives exact same no. of event align pairs for both OpenCL implementation and CPU implementation. Others with deviation of 1-6.
Only for 139th read has a deviation of 95. For now we suspect it is due to the floating point precision difference. In the next phase we hope to identify the reasons for this deviations.

139th read

read_pos with ref_pos for CPU implementation



read_pos with ref_pos for OpenCL implementation



Further justification

Source: Intel ® FPGA SDK for OpenCL ™ Standard Edition Programming Guide

- Emulator run uses floating point computation hardware of the CPU whereas the hardware run uses floating point cores implemented as FPGA cores.
- The emulation of channel behavior has limitations, In such cases, the Emulator might execute channel operations in an order different from that on the hardware.
  (especially for conditional channel operations where the kernel does not call the channel operation in every loop iteration)
- Difference in channel depths might lead to execution on the hardware hangs while kernel emulation works without any issue.

Conclusion and Future Directions

Conclusion

- ABEA algorithm is a key component in nanopore data analysis
- f5c has been fine-tuned to run on CPUs/GPUs
- Accelerations using FPGAs has a lower power requirement relative to GPUs
- The portable version of f5c can be superior to deploy in many hardware platforms

Future Directions

“

Include the OpenCL implementation into nanopolish package

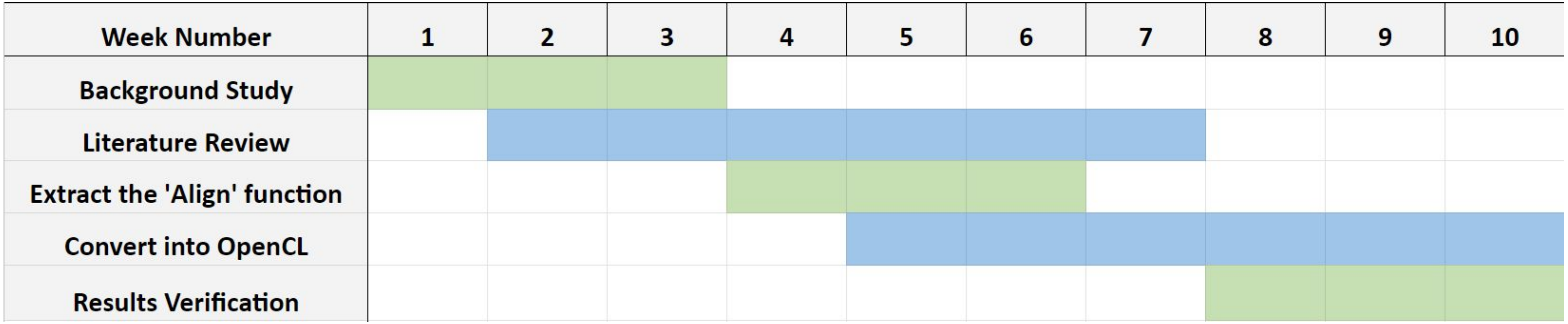
Adapt the hardware design into an embedded device to perform event alignment in real-time

”

Work Progress

Timeline

- Semester 7 progress and timeline



Week Number	1	2	3	4	5	6	7	8	9	10
Background Study	🟩	🟩	🟩							
Literature Review		🟦	🟦	🟦	🟦	🟦	🟦			
Extract the 'Align' function				🟩	🟩	🟩				
Convert into OpenCL					🟦	🟦	🟦	🟦	🟦	🟦
Results Verification								🟩	🟩	🟩

Plan for Semester 8

- Evaluate performance on Altera Stratix V FPGA and compare with f5c
- Improve performance on FPGA through optimization techniques identified from literature review
- Deploy on different hardware platforms such as CPU, GPU and evaluate performance

Q & A