


DP-HLS: A High-Level Synthesis Framework for Accelerating Dynamic Programming Algorithms in Bioinformatics

Anshu Gupta, Yingqi Cao, Jason Liang, Yatish Turakhia

University of California San Diego

IEEE International Symposium on High-Performance Computer Architecture (HPCA) 2026

Outline

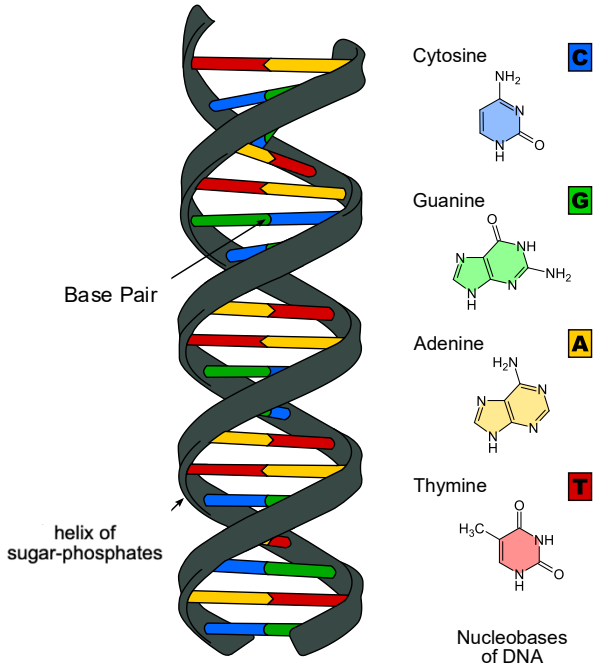
- Dynamic Programming (DP) in Bioinformatics
- DP Algorithmic Variations and Hardware Accelerators
- DP-HLS Framework
- Key Contributions and Results
- Conclusion and Future Applications

Outline

- **Dynamic Programming (DP) in Bioinformatics**
- DP Algorithmic Variations and Hardware Accelerators
- DP-HLS Framework
- Key Contributions and Results
- Conclusion and Future Applications

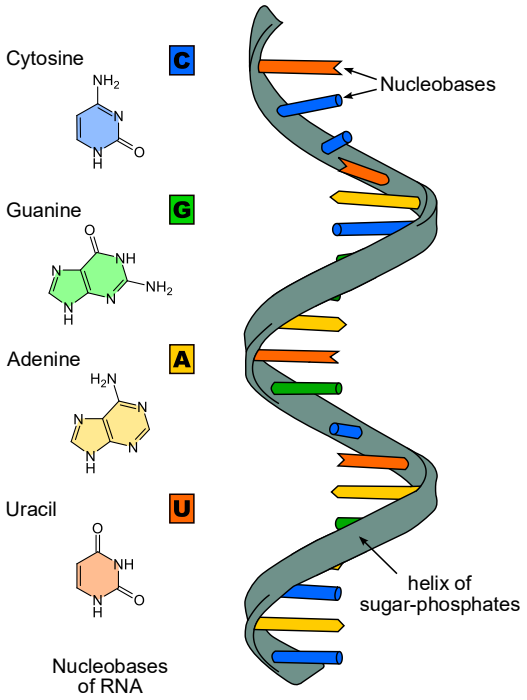
Biological Sequences: The Language of Life

DNA Sequences



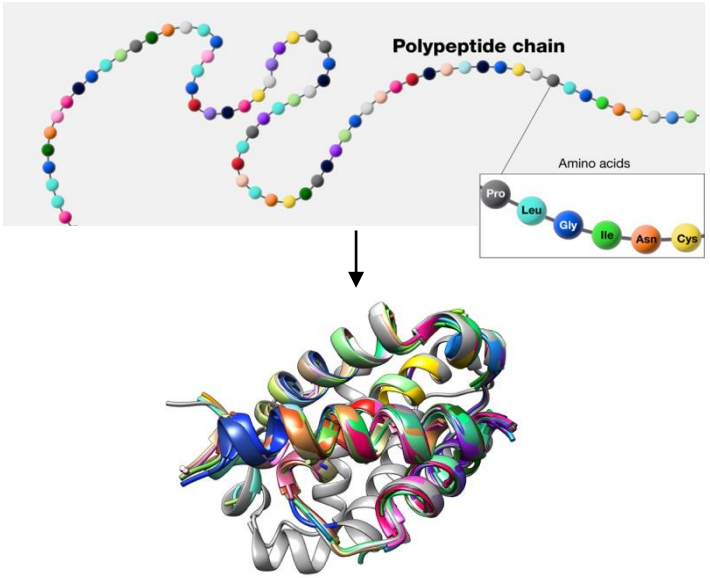
{ATCG}

RNA Sequences



{AUCG}

Protein Sequences



{ARNDCEQGHILKMFPSTWYV}

Image source: Wikipedia, National Human Genome Research Institute

Explosion of Biological Sequence Data

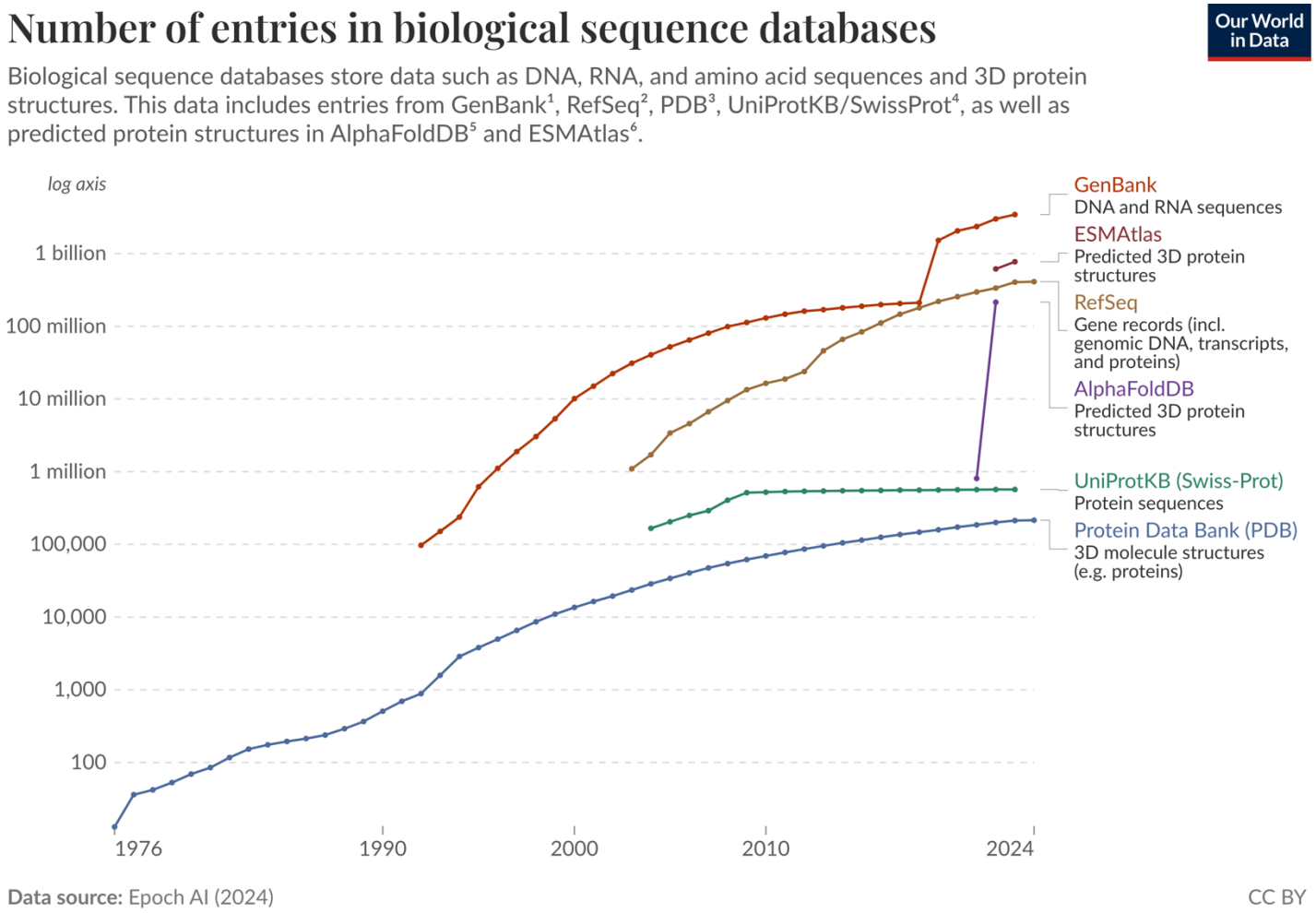
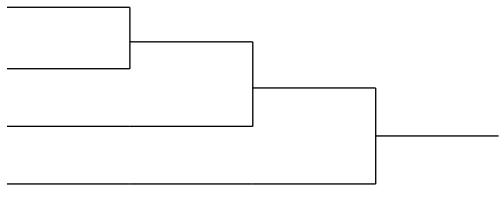


Image source: ourworldindata.org

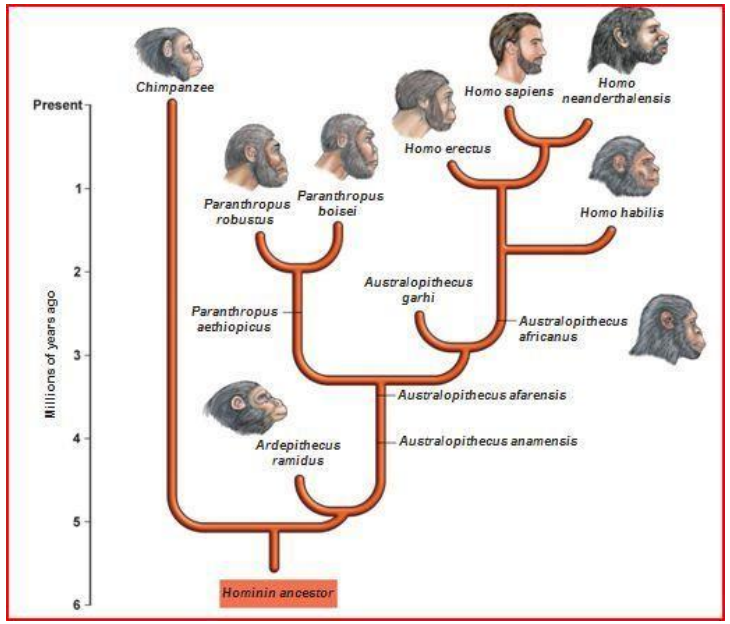
Biological Sequence Comparison

```
ATGCTACGATCCGTAT--ACGCACA---GGTTTCAGAC  S1
ATGGTATGATACGTACCTACGCACA---AGTATCAGAC  S2
ATCCTACGTTTCGTATT-ATGCATA---GGTCTTAGAT  S3
ATGATACAGCTCGTATATACGCACACATGGTTACAGGC  S4
```



Compare sequences to form trees

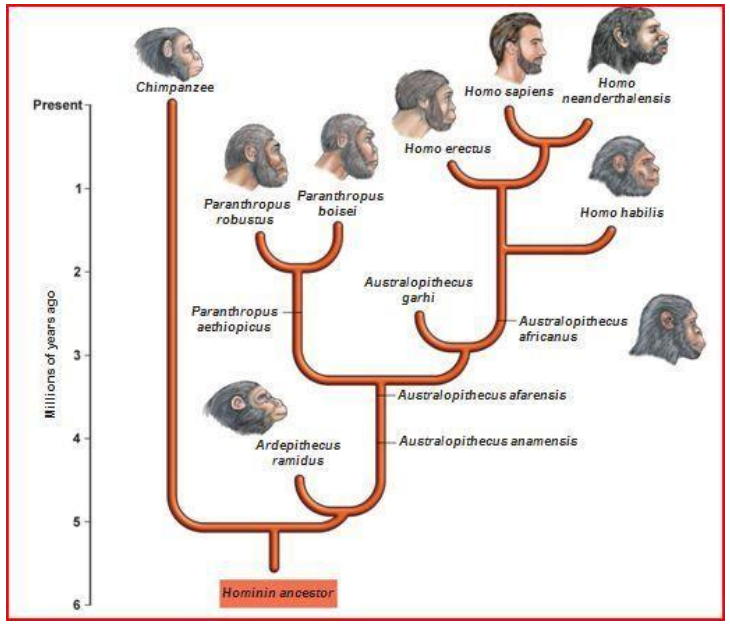
Biological Sequence Comparison



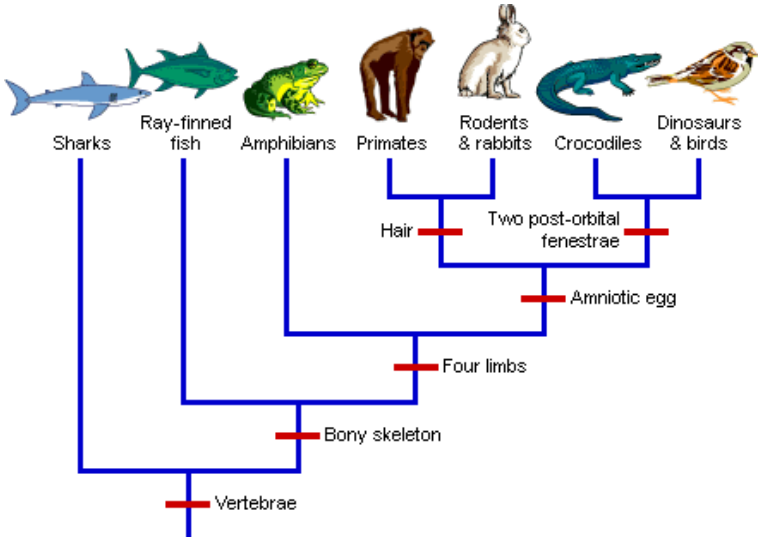
Human Ancestry

Image source: https://kaiserscience.wordpress.com/biology-the-living-environment/classification/classifying-groups-of-humans/

Biological Sequence Comparison



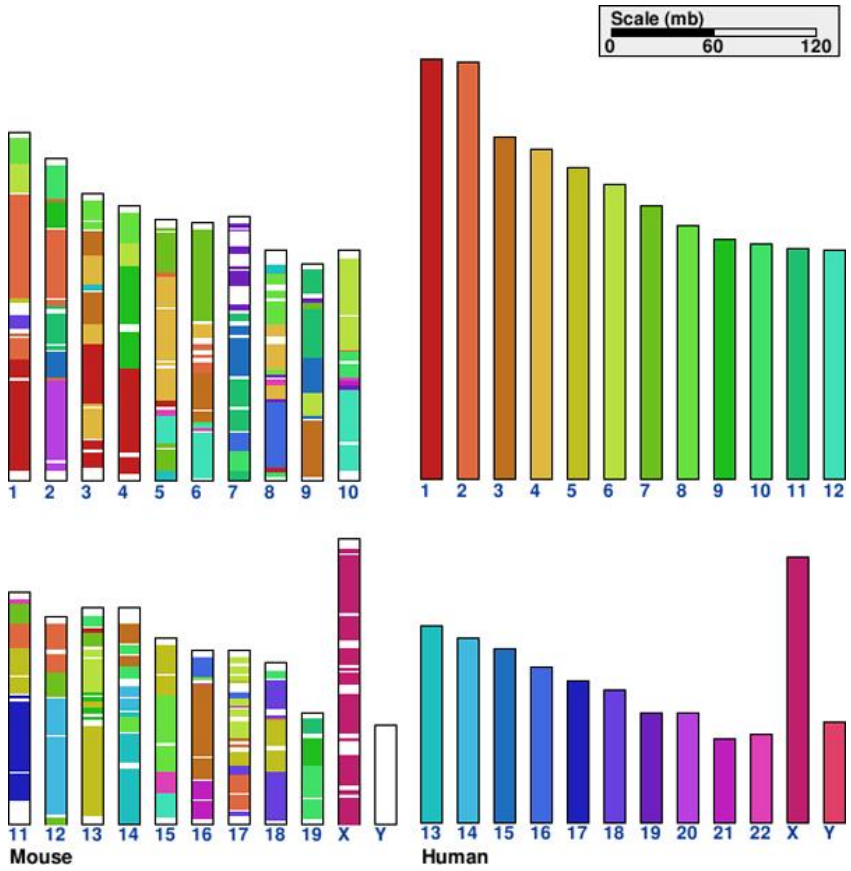
Human Ancestry



Evolutionary Biology

Image source: https://kaiserscience.wordpress.com/biology-the-living-environment/classification/classifying-groups-of-humans/, University of California Museum of Paleontology's Understanding Evolution

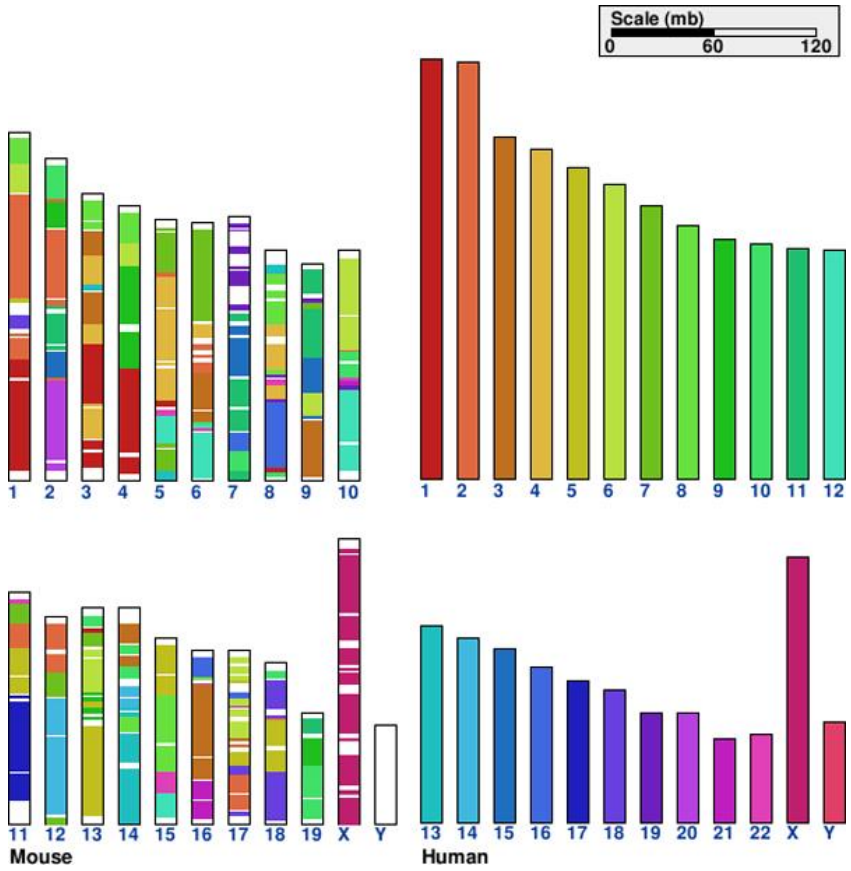
Biological Sequence Comparison



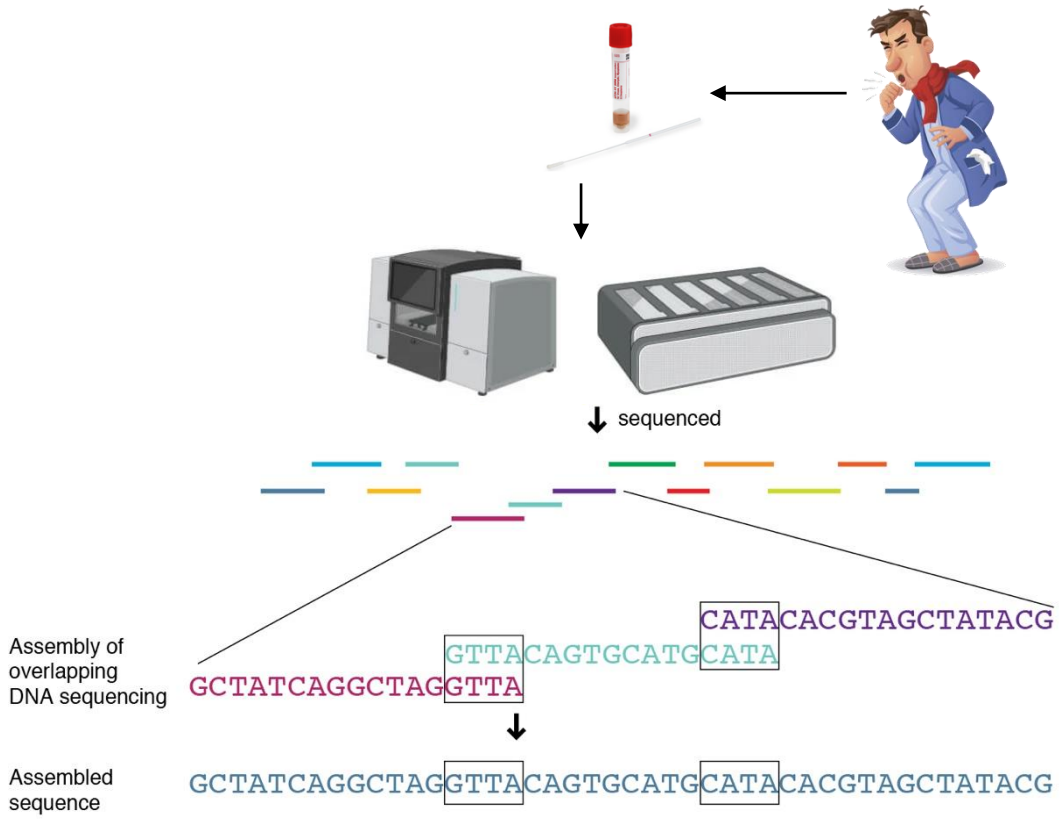
Homology Detection

Image source: Zhang et al, Springer Nature Link (2012)

Biological Sequence Comparison



Homology Detection



Genome Assembly

Image source: Zhang et al, Springer Nature Link (2012), avantonder.github.io

Signal Comparison in Bioinformatics

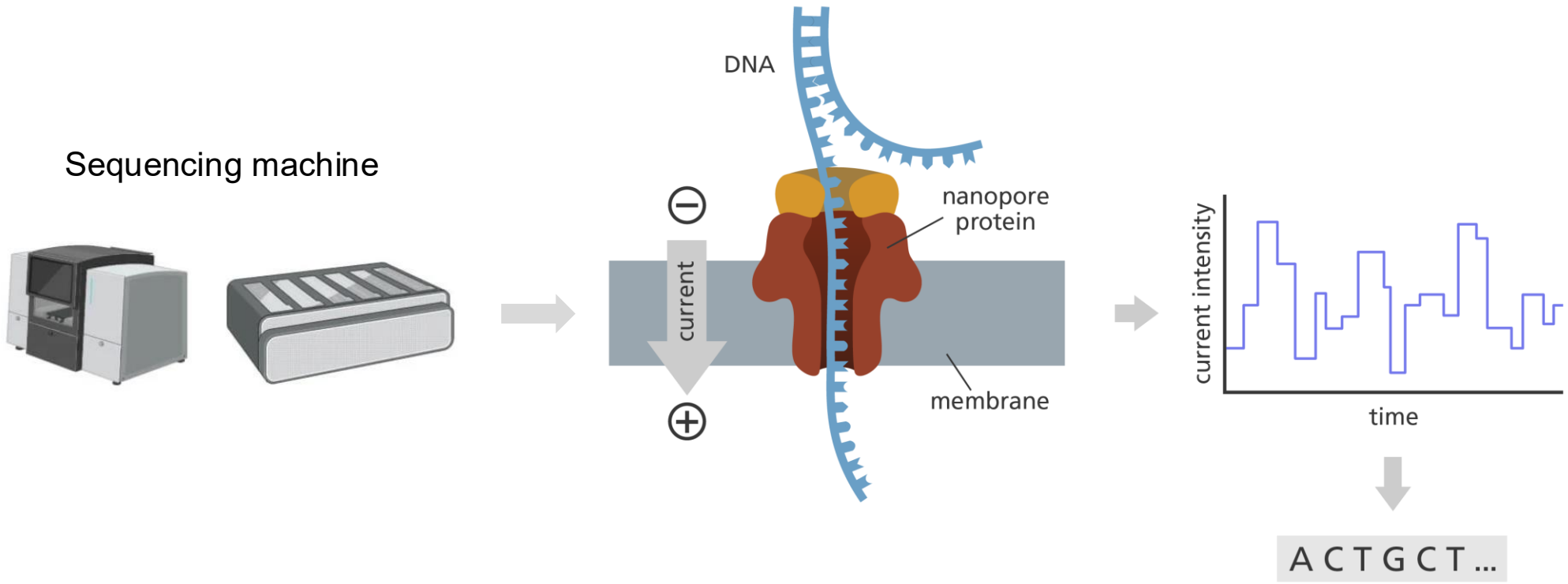


Image source: yourgenome.org

Introduction to Sequence Alignment

- Sequence comparison relies on sequence **alignment**

Input Sequences

```
GACTACGATCCGTATACGCACAGGTTCAGAC
and
GACTACAGCTCGTATACGCACACATGGTTCAGAC
```

Pairwise Sequence Alignment

```
GACTACGATCCGTATACGCACA---GGTTCAGAC
||||||    ||||||||||||   |||||||||
GACTACAGCTCGTATACGCACACATGGTTCAGAC
```

Image source: An eBook on Practical Bioinformatics

Introduction to Sequence Alignment

- Sequence comparison relies on sequence **alignment**

- Goal
  - Maximize **matches**
  - Minimize **mismatches or substitutions**
  - Insert **gaps (insertions/deletions)** to maximize alignment score

Input Sequences

GACTACGATCCGTATACGCACAGGTTCAGAC
and
GACTACAGCTCGTATACGCACACATGGTTCAGAC

Pairwise Sequence Alignment

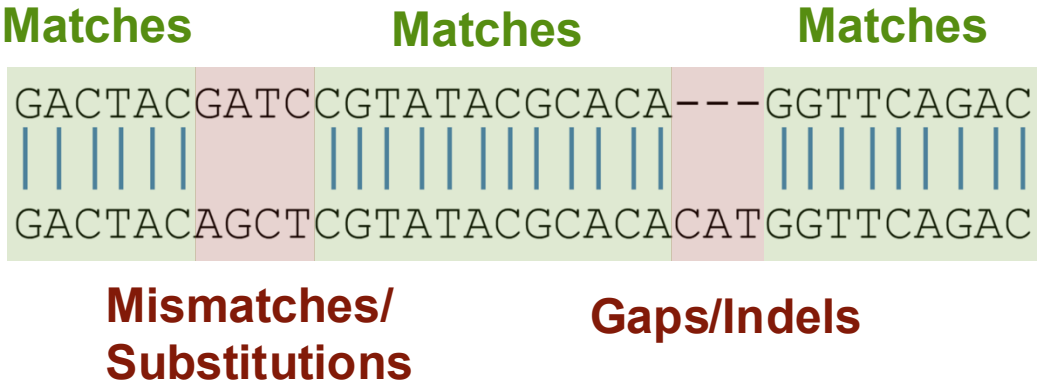


Image source: An eBook on Practical Bioinformatics

Introduction to Sequence Alignment

- Sequence comparison relies on sequence **alignment**
- Goal
  - Maximize **matches**
  - Minimize **mismatches or substitutions**
  - Insert **gaps (insertions/deletions)** to maximize alignment score
- We align **pair of sequences** or **multiple sequences**

Input Sequences

```
GACTACGATCCGTATACGCACAGGTTCAGAC
and
GACTACAGCTCGTATACGCACACATGGTTCAGAC
```

Pairwise Sequence Alignment

```
GACTACGATCCGTATACGCACA---GGTTCAGAC
||||||    ||||||||||||    |||||||||
GACTACAGCTCGTATACGCACACATGGTTCAGAC
```

Multiple Sequence Alignment

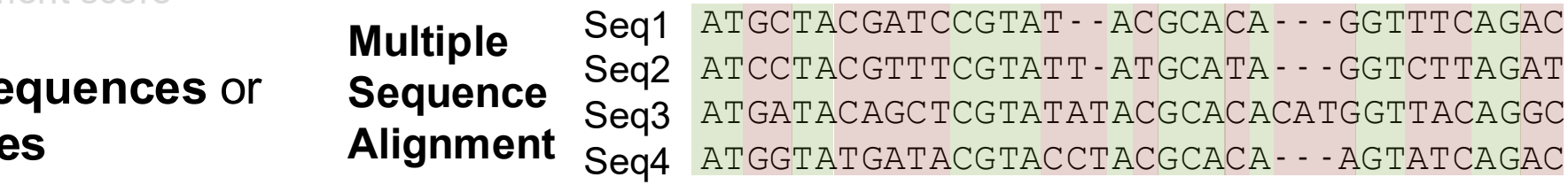
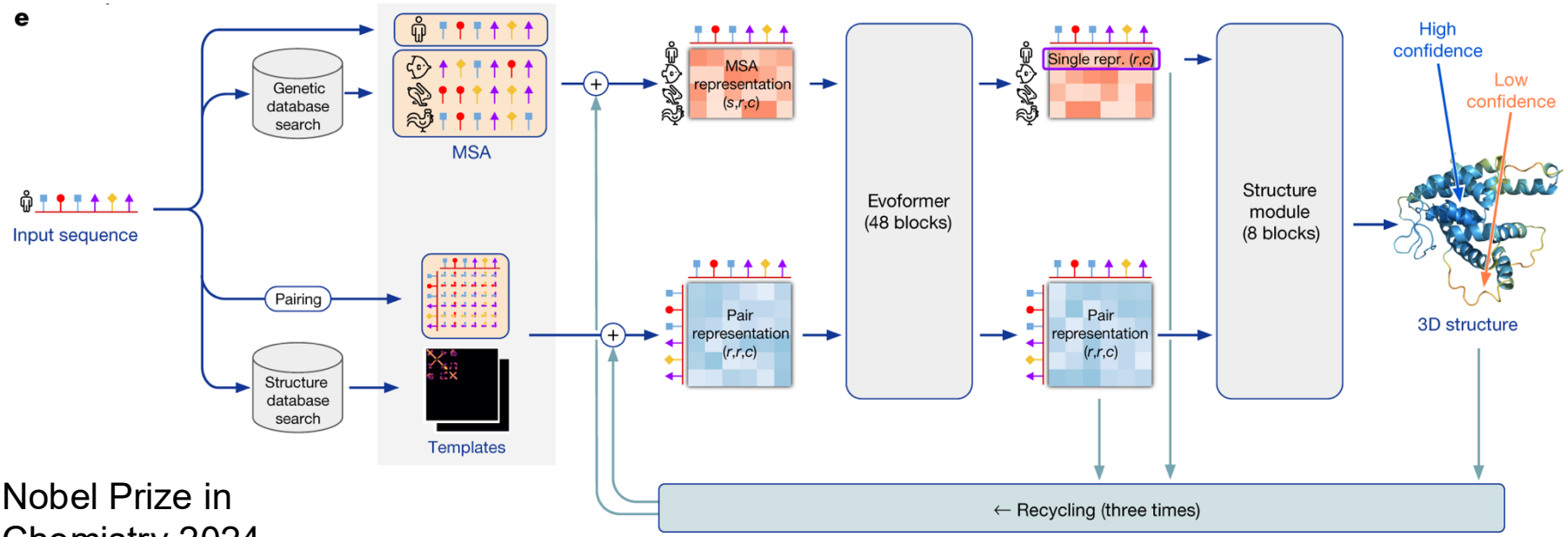


Image source: An eBook on Practical Bioinformatics

Sequence Alignment at Scale

- **AlphaFold** predicts 3-D protein structures from sequences



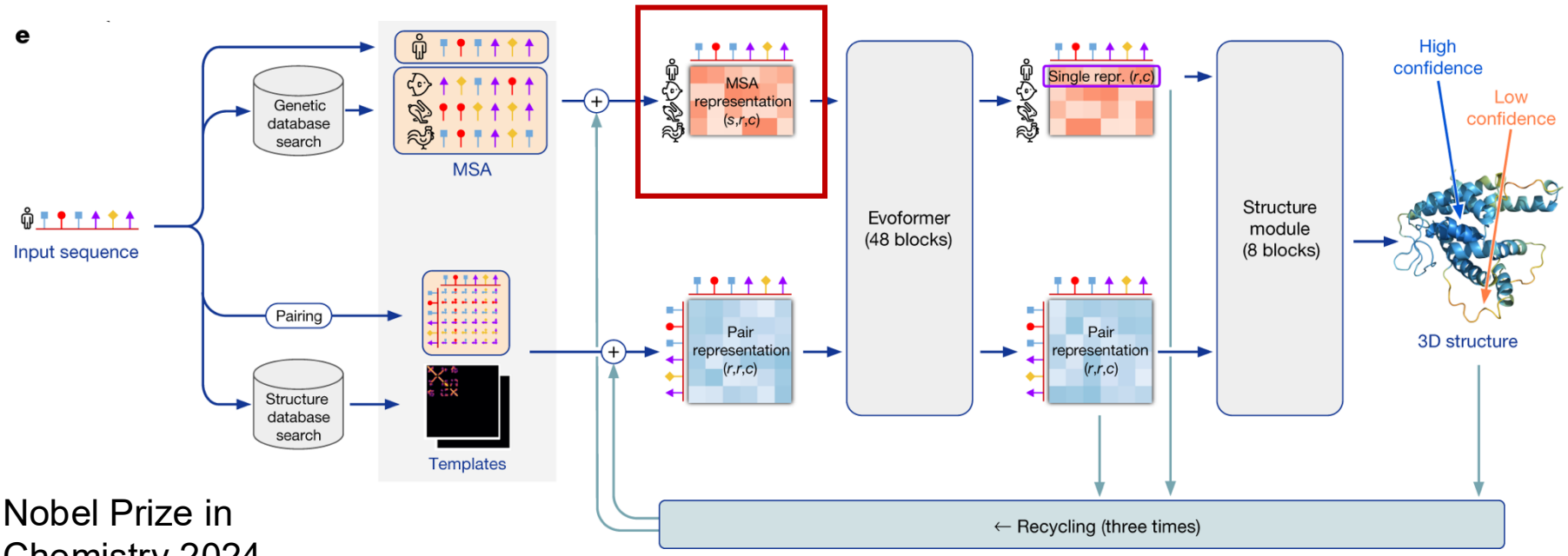
Nobel Prize in Chemistry 2024

AlphaFold2 Architecture

Image source: Jumper et al, Nature (2021)

Sequence Alignment at Scale

- **AlphaFold** predicts 3-D protein structures from sequences
- Performs **multiple sequence alignment** (MSA) at scale



Nobel Prize in Chemistry 2024

AlphaFold2 Architecture

Image source: Jumper et al, Nature (2021)

The Algorithm behind Sequence Alignment: 2D Dynamic Programming (DP)

Compare two input sequences

A C G

G A C

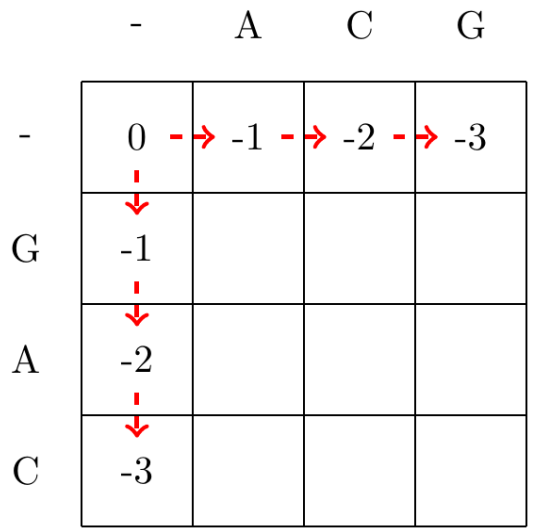
Image source: KTH's course CB2442, Bioinformatics

The Algorithm behind Sequence Alignment: 2D Dynamic Programming (DP)

Compare two input sequences

A C G

G A C



Initialization

Equations

$F(0,j) = -j*d$ where $j \in \{0 \dots N\}$

$F(i,0) = -i*d$ where $i \in \{0 \dots M\}$

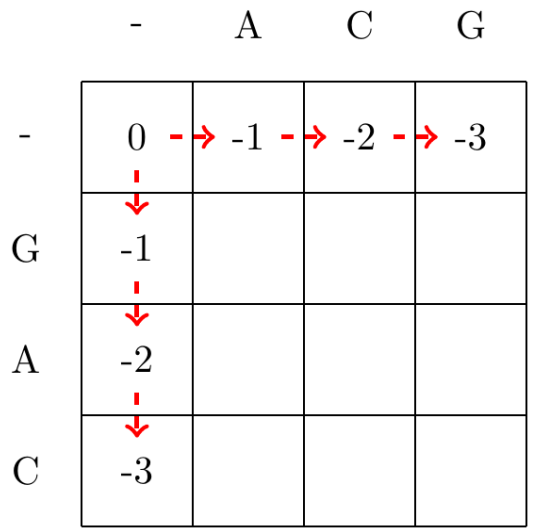
Image source: KTH's course CB2442, Bioinformatics

The Algorithm behind Sequence Alignment: 2D Dynamic Programming (DP)

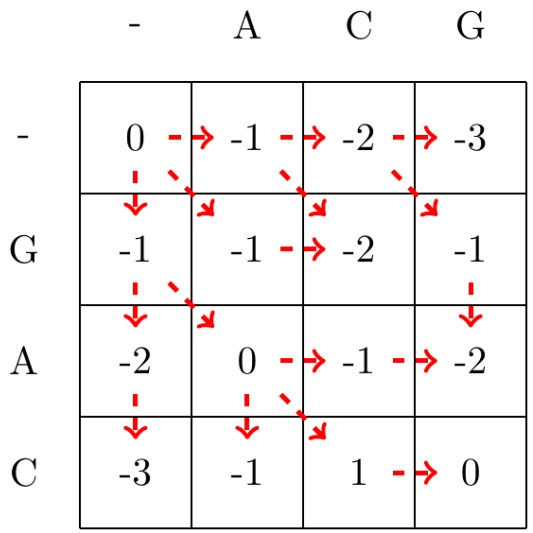
Compare two input sequences

A C G

G A C



Initialization



Scoring

Equations

$$F(0,j) = -j*d \text{ where } j \in \{0 \dots N\}$$
$$F(i,0) = -i*d \text{ where } i \in \{0 \dots M\}$$

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

where $i \in \{0 \dots N\}$ and $j \in \{0 \dots M\}$

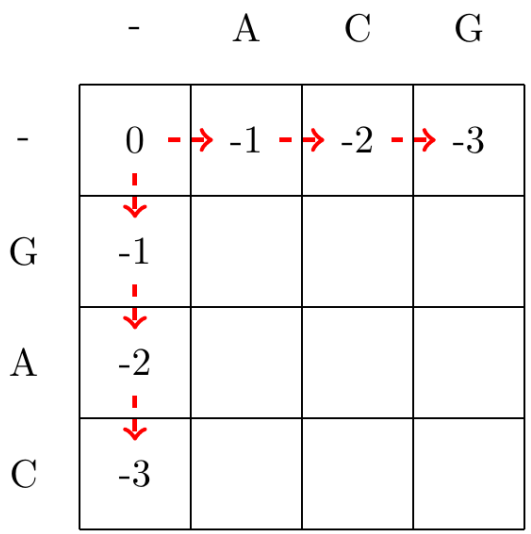
Image source: KTH's course CB2442, Bioinformatics

The Algorithm behind Sequence Alignment: 2D Dynamic Programming (DP)

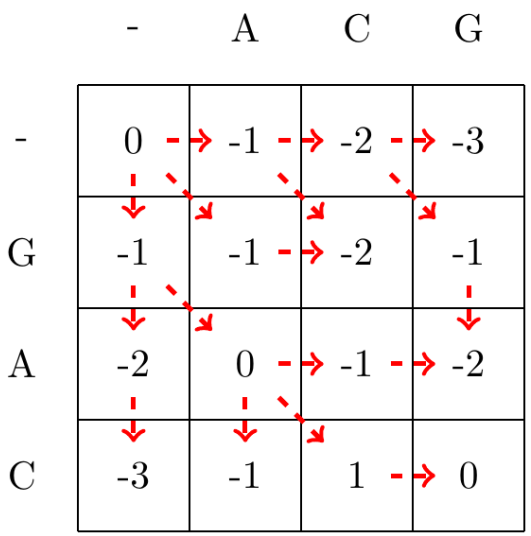
Compare two input sequences

A C G

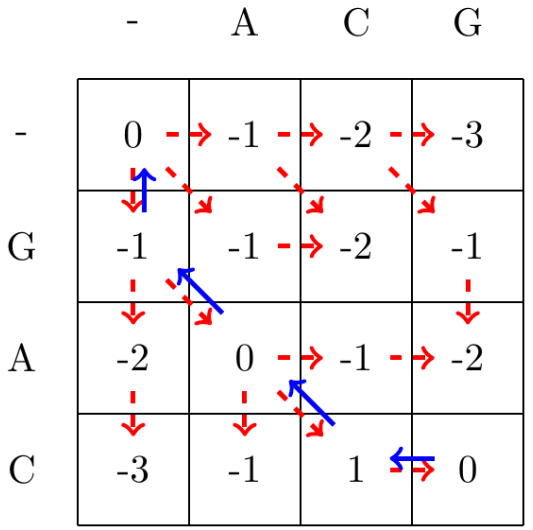
G A C



Initialization



Scoring



Traceback

Alignment Output

_ A C G

G A C _

Equations

$$F(0,j) = -j*d \text{ where } j \in \{0 \dots N\}$$
$$F(i,0) = -i*d \text{ where } i \in \{0 \dots M\}$$

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

where $i \in \{0 \dots N\}$ and $j \in \{0 \dots M\}$

Image source: KTH's course CB2442, Bioinformatics

Examples of DP Paradigm

Needleman-Wunsch Algorithm (Global Alignment)

```
M L A D F R - L Q V K N
| | | |   |   |   | | |
M L A D - R Q L N V K N
```

Initialization:

$$F(0,j) = -j*d \text{ where } j \in \{0 \dots N\}$$
$$F(i,0) = -i*d \text{ where } i \in \{0 \dots M\}$$

Scoring:

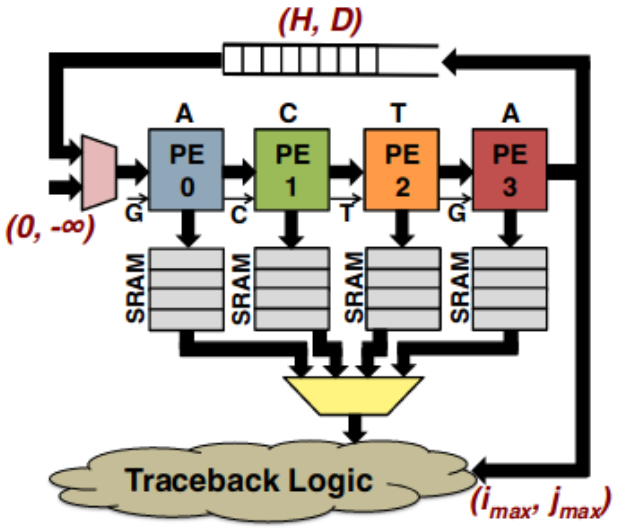
$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$
$$\text{where } i \in \{0 \dots N\} \text{ and } j \in \{0 \dots M\}$$

Traceback:

D		M_1	L_2	A_3	D_4	R_5	Q_6	L_7	N_8	V_9	K_{10}	N_{11}
	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11
M_1	-1	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9
L_2	-2	0	2	1	0	-1	-2	-3	-4	-5	-6	-7
A_3	-3	-1	1	3	2	1	0	-1	-2	-3	-4	-5
D_4	-4	-2	0	2	4	3	2	1	0	-1	-2	-3
F_5	-5	-3	-1	1	3	2	1	0	-1	-2	-3	-4
R_6	-6	-4	-2	0	2	4	3	2	1	0	-1	-2
L_7	-7	-5	-3	-1	1	3	2	4	3	2	1	0
Q_8	-8	-6	-4	-2	0	2	4	3	2	1	0	-1
V_9	-9	-7	-5	-3	-1	1	3	2	1	3	2	1
K_{10}	-10	-8	-6	-4	-2	0	2	1	0	2	4	3
N_{11}	-11	-9	-7	-5	-3	-1	1	0	2	1	3	5

Hardware Acceleration of DP in Bioinformatics

- Many hardware accelerators for dynamic programming algorithms exists
- Typically based on **1-D systolic array architecture**



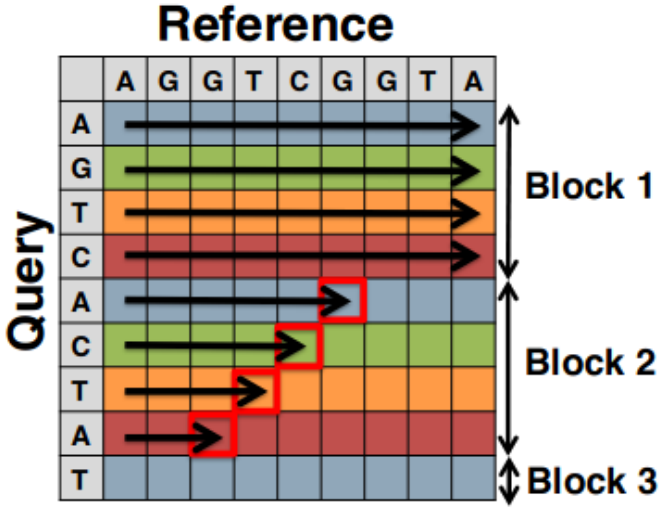


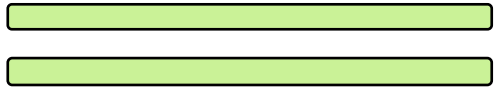
Image source: Turakhia et al, ASPLOS (2018)

Outline

- Dynamic Programming (DP) in Bioinformatics
- **DP Algorithmic Variations and Hardware Accelerators**
- DP-HLS Framework
- Key Contributions and Results
- Conclusion and Future Applications

One Paradigm, Many Applications

Global Alignment



Initialization:

$$F(0,j) = -j*d \text{ where } j \in \{0 \dots N\}$$
$$F(i,0) = -i*d \text{ where } i \in \{0 \dots M\}$$

Scoring:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$
$$\text{where } i \in \{0 \dots N\} \text{ and } j \in \{0 \dots M\}$$

Traceback:

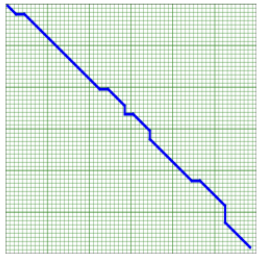
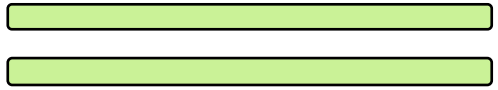


Image source: UCSD's course ECE 284

One Paradigm, Many Applications

Global Alignment



Initialization:

$$F(0,j) = \text{-j*d} \quad \text{where } j \in \{0 \dots N\}$$
$$F(i,0) = \text{-i*d} \quad \text{where } i \in \{0 \dots M\}$$

Modify initialization scores

Initialization:

$$F(0,j) = 0$$
$$F(i,0) = 0$$

Scoring:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$
$$\text{where } i \in \{0 \dots N\} \text{ and } j \in \{0 \dots M\}$$

Traceback:

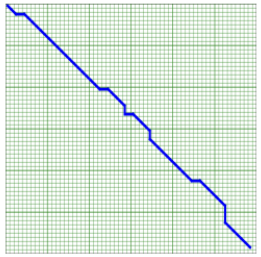
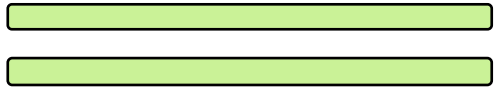


Image source: UCSD's course ECE 284

Differences are highlighted in

One Paradigm, Many Applications

Global Alignment



Initialization:

$F(0,j) = -j*d \text{ where } j \in \{0 \dots N\}$
$F(i,0) = -i*d \text{ where } i \in \{0 \dots M\}$

Scoring:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

where $i \in \{0 \dots N\}$ and $j \in \{0 \dots M\}$

Modify scoring function

Scoring:

$$F(i,j) = \max \begin{cases} 0 \\ F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

where $i \in \{0 \dots N\}$ and $j \in \{0 \dots M\}$

Traceback:

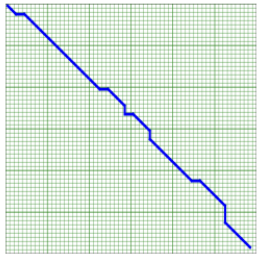
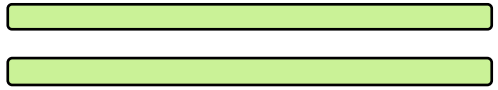


Image source: UCSD's course ECE 284

Differences are highlighted in

One Paradigm, Many Applications

Global Alignment



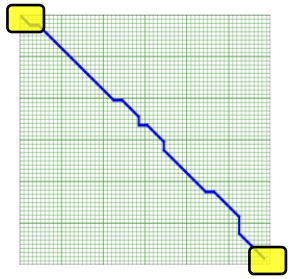
Initialization:

$$F(0,j) = -j*d \text{ where } j \in \{0 \dots N\}$$
$$F(i,0) = -i*d \text{ where } i \in \{0 \dots M\}$$

Scoring:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$
$$\text{where } i \in \{0 \dots N\} \text{ and } j \in \{0 \dots M\}$$

Traceback:



Modify traceback start and end points

Traceback:

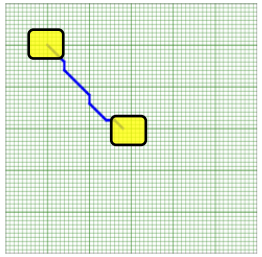
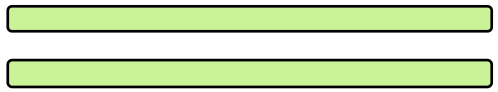


Image source: UCSD's course ECE 284

Differences are highlighted in

One Paradigm, Many Applications

Global Alignment



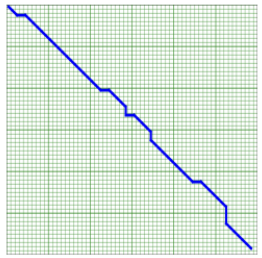
Initialization:

$$F(0,j) = -j*d \text{ where } j \in \{0 \dots N\}$$
$$F(i,0) = -i*d \text{ where } i \in \{0 \dots M\}$$

Scoring:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$
$$\text{where } i \in \{0 \dots N\} \text{ and } j \in \{0 \dots M\}$$

Traceback:



Local Alignment

Initialization:

$$F(0,j) = 0$$
$$F(i,0) = 0$$

Scoring:

$$F(i,j) = \max \begin{cases} 0 \\ F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$
$$\text{where } i \in \{0 \dots N\} \text{ and } j \in \{0 \dots M\}$$

Traceback:

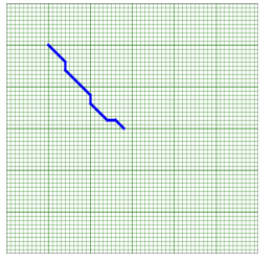
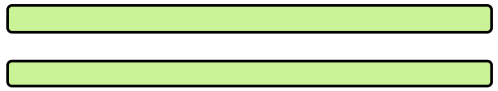


Image source: UCSD's course ECE 284

One Paradigm, Many Applications

Global Alignment



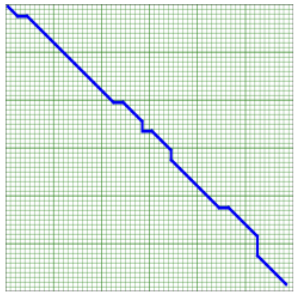
Initialization:

$F(0,j) = -j*d \text{ where } j \in \{0 \dots N\}$
$F(i,0) = -i*d \text{ where } i \in \{0 \dots M\}$

Scoring:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

Traceback:



Local Alignment

Initialization:

$F(0,j) = 0$
$F(i,0) = 0$

Scoring:

$$F(i,j) = \max \begin{cases} 0 \\ F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

Traceback:

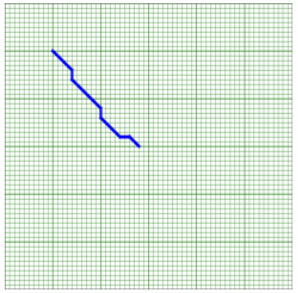
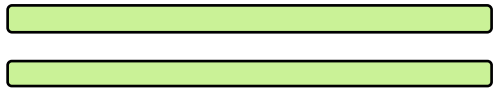


Image source: UCSD's course ECE 284

One Paradigm, Many Applications

Global Alignment



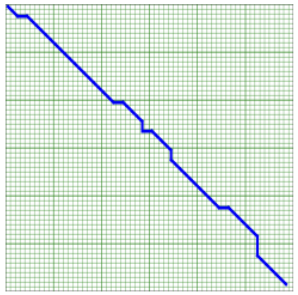
Initialization:

$F(0,j) = -j*d \text{ where } j \in \{0 \dots N\}$
$F(i,0) = -i*d \text{ where } i \in \{0 \dots M\}$

Scoring:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

Traceback:



Local Alignment

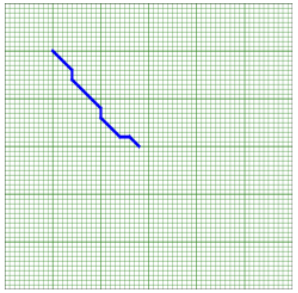
Initialization:

$F(0,j) = 0$
$F(i,0) = 0$

Scoring:

$$F(i,j) = \max \begin{cases} 0 \\ F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

Traceback:



Semi-Global Alignment

Initialization:

$F(0,j) = 0$
$F(i,0) = 0$

Scoring:

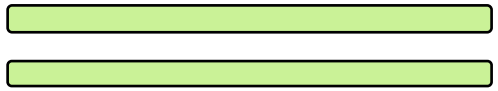
$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

Traceback:

Image source: UCSD's course ECE 284

One Paradigm, Many Applications

Global Alignment



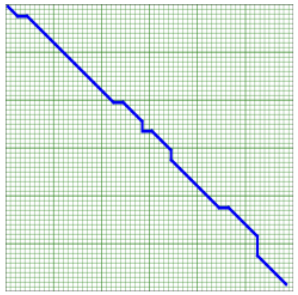
Initialization:

$F(0,j) = -j*d \text{ where } j \in \{0 \dots N\}$
$F(i,0) = -i*d \text{ where } i \in \{0 \dots M\}$

Scoring:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

Traceback:



Local Alignment

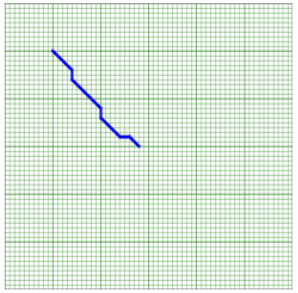
Initialization:

$F(0,j) = 0$
$F(i,0) = 0$

Scoring:

$$F(i,j) = \max \begin{cases} 0 \\ F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

Traceback:



Semi-Global Alignment

Initialization:

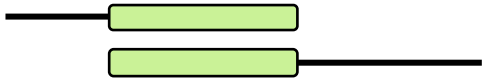
$F(0,j) = 0$
$F(i,0) = 0$

Scoring:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

Traceback:

Overlap Alignment



Initialization:

$F(0,j) = 0$
$F(i,0) = 0$

Scoring:

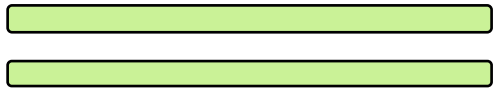
$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

Traceback:

Image source: UCSD's course ECE 284

One Paradigm, Many Applications

Global Alignment



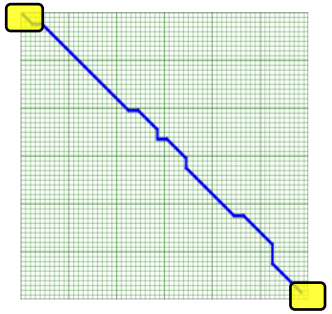
Initialization:

$F(0,j) = -j*d$ where $j \in \{0 \dots N\}$
$F(i,0) = -i*d$ where $i \in \{0 \dots M\}$

Scoring:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

Traceback:



Applications:
Whole genome alignment

Local Alignment

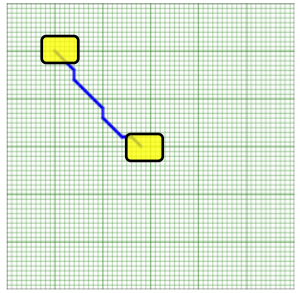
Initialization:

$F(0,j) = 0$
$F(i,0) = 0$

Scoring:

$$F(i,j) = \max \begin{cases} 0 \\ F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

Traceback:



Applications:
Homology detection

Semi-Global Alignment

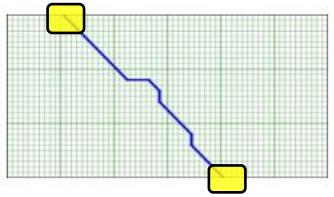
Initialization:

$F(0,j) = 0$
$F(i,0) = 0$

Scoring:

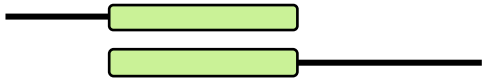
$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

Traceback:



Applications:
Short read alignment

Overlap Alignment



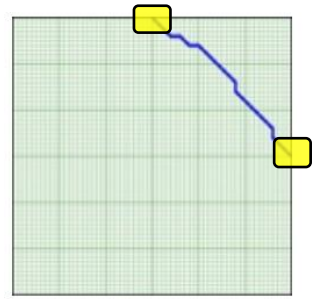
Initialization:

$F(0,j) = 0$
$F(i,0) = 0$

Scoring:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

Traceback:



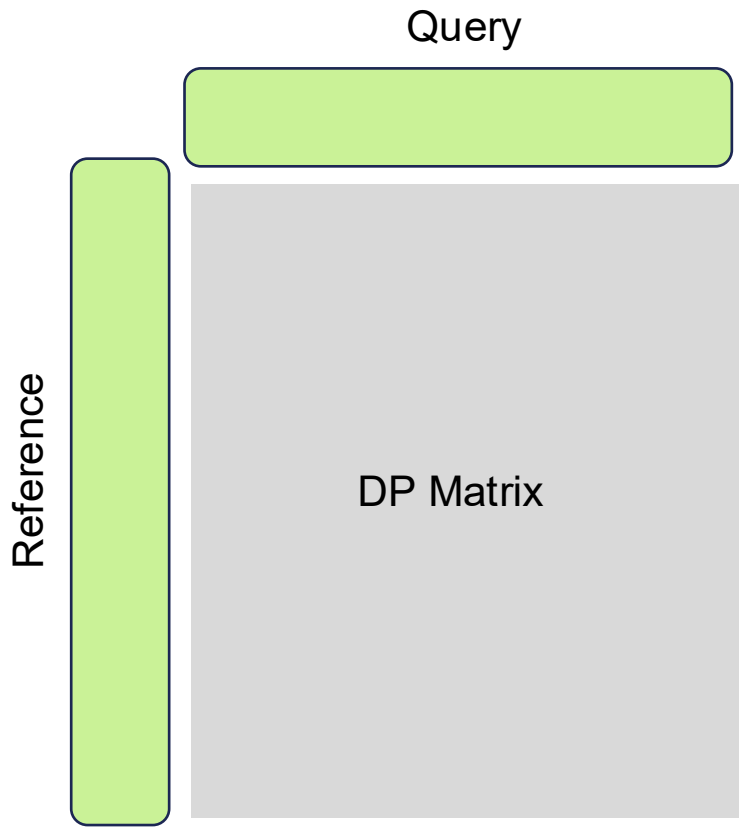
Applications:
Genome Assembly

Image source: UCSD's course ECE 284

Differences are highlighted in

One Paradigm, Many Applications

Variations in Sequence Alphabets

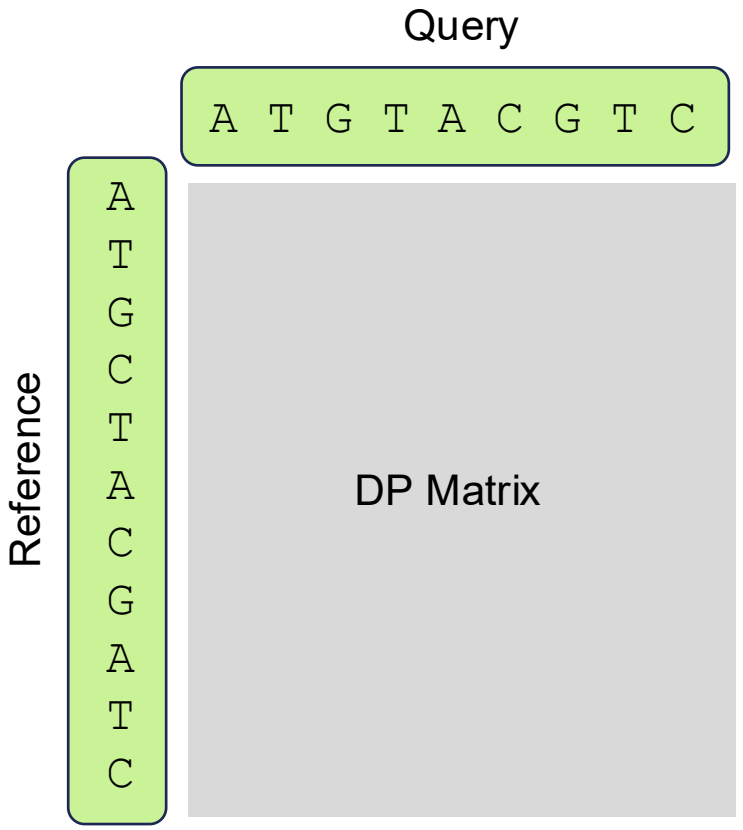


One Paradigm, Many Applications

Variations in Sequence Alphabets

1. DNA Sequence Alignment

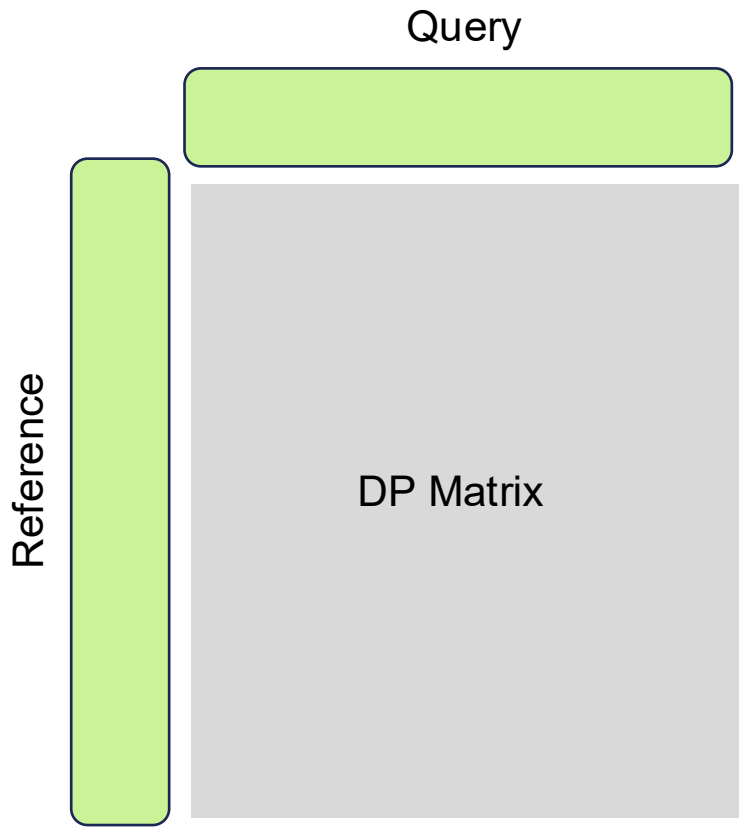
```
A T G C T A C G A T C
A T G - T A C G - T C
```



One Paradigm, Many Applications

Variations in Sequence Alphabets

1. DNA Sequence Alignment

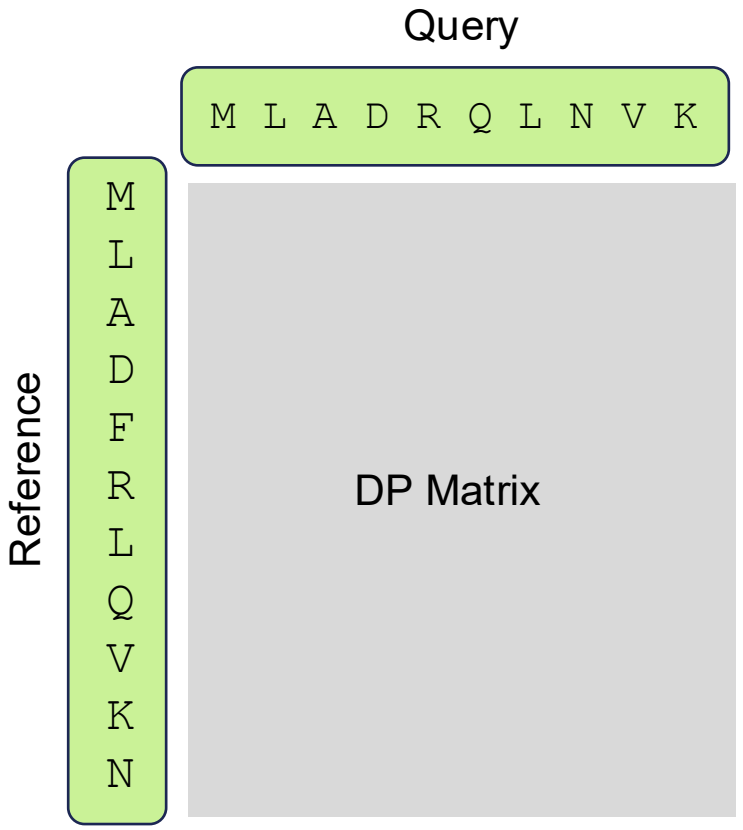


One Paradigm, Many Applications

Variations in Sequence Alphabets

1. DNA Sequence Alignment
2. **Protein Sequence Alignment**

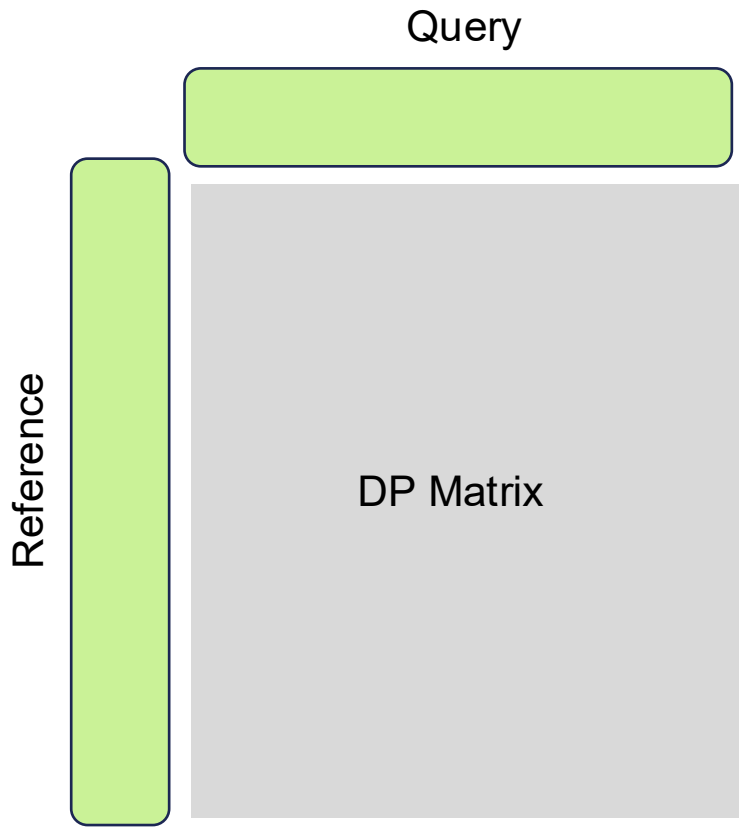
```
M L A D F R - L Q V K N
M L A D - R Q L N V K -
```



One Paradigm, Many Applications

Variations in Sequence Alphabets

1. DNA Sequence Alignment
2. Protein Sequence Alignment



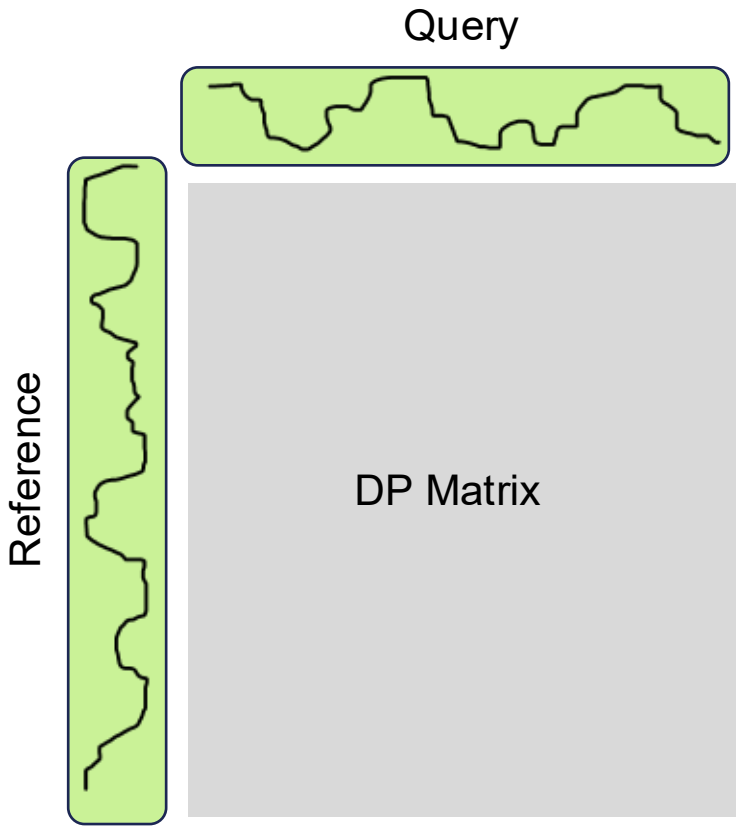
One Paradigm, Many Applications

Variations in Sequence Alphabets

1. DNA Sequence Alignment
2. Protein Sequence Alignment
3. **Raw Signal Alignment**

$(1.2 + 0.3i)$ $(1.5 + 0.4i)$ $(0.9 + 0.2i)$ $(1.8 + 0.5i)$

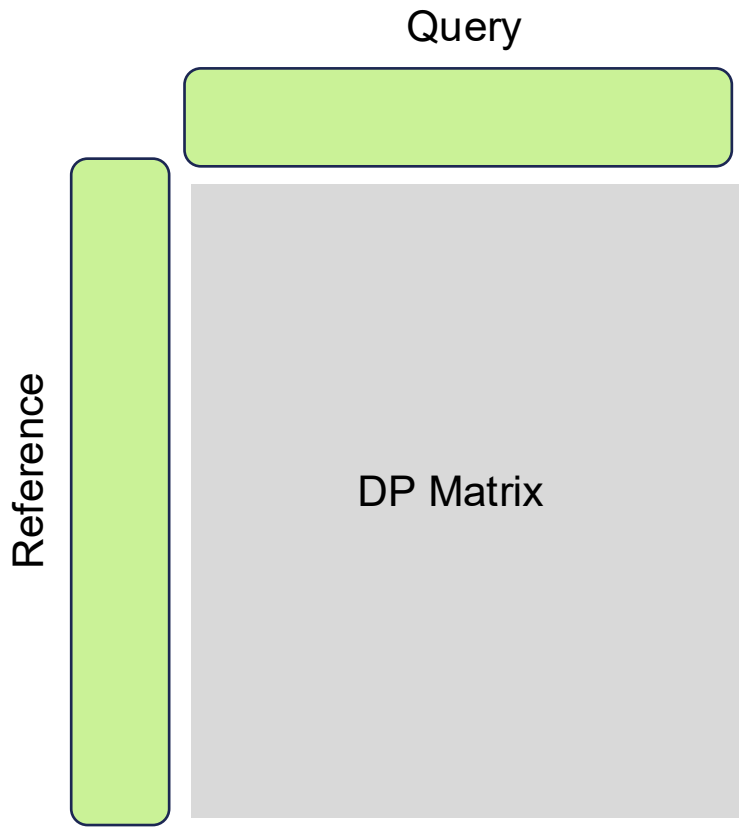
$(1.2 + 0.3i)$ $---$ $(0.95 + 0.25i)$ $(1.8 + 0.52i)$



One Paradigm, Many Applications

Variations in Sequence Alphabets

1. DNA Sequence Alignment
2. Protein Sequence Alignment
3. Raw Signal Alignment



One Paradigm, Many Applications

Variations in Sequence Alphabets

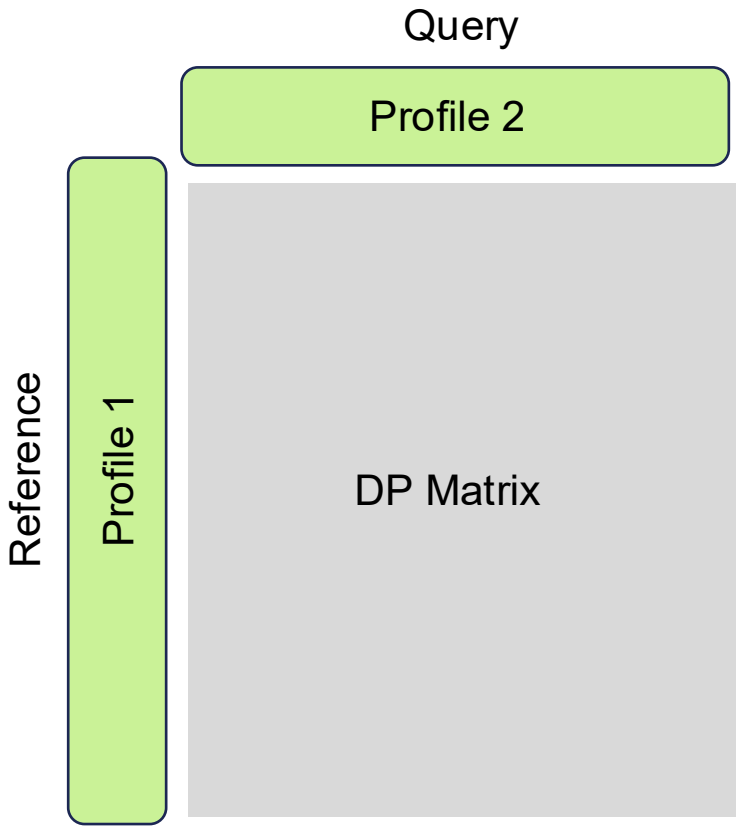
1. DNA Sequence Alignment
2. Protein Sequence Alignment
3. Raw Signal Alignment
4. **Profile Alignment**

Col:	1	2	3	4
A	0.8	0.1	0.0	0.7
C	0.1	0.6	0.1	0.1
G	0.1	0.2	0.8	0.1
T	0.0	0.1	0.1	0.1

Profile 1

Col:	1	2	3
A	0.7	0.2	0.6
C	0.2	0.5	0.1
G	0.1	0.2	0.2
T	0.0	0.1	0.1

Profile 2



One Paradigm, Many Applications

Variations in Scoring functions

Scoring:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

where $i \in \{0 \dots N\}$ and $j \in \{0 \dots M\}$

One Paradigm, Many Applications

Variations in Scoring functions

Scoring:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

where $i \in \{0 \dots N\}$ and $j \in \{0 \dots M\}$

$$I_x(i,j) = \max \begin{cases} I_x(i-1,j) + gap_{extend} \\ F(i-1,j) + gap_{open} \end{cases}$$

$$I_y(i,j) = \max \begin{cases} I_y(i,j-1) + gap_{extend} \\ F(i,j-1) + gap_{open} \end{cases}$$

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ Ix(i,j) \\ Iy(i,j) \end{cases}$$

Global Alignment with Affine Gap Penalty

One Paradigm, Many Applications

Variations in Scoring functions

Scoring:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

where $i \in \{0 \dots N\}$ and $j \in \{0 \dots M\}$

$$I_x(i,j) = \max \begin{cases} I_x(i-1,j) + gap_{extend} \\ F(i-1,j) + gap_{open} \end{cases}$$

$$I_y(i,j) = \max \begin{cases} I_y(i,j-1) + gap_{extend} \\ F(i,j-1) + gap_{open} \end{cases}$$

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ Ix(i,j) \\ Iy(i,j) \end{cases}$$

Global Alignment with Affine Gap Penalty

$$V_j(i,j) = Q(y_j) * \max \begin{cases} d * V_m(i,j-1) \\ e * V_j(i,j-1) \end{cases}$$

$$V_i(i,j) = Q(x_i) * \max \begin{cases} d * V_m(i-1,j) \\ e * V_i(i-1,j) \end{cases}$$

$$V_m(i,j) = P(x_i, y_j) * \max \begin{cases} (1-e) * V_j(i-1,j-1) \\ (1-e) * V_i(i-1,j-1) \\ (1-2d) * V_m(i-1,j-1) \end{cases}$$

Viterbi Algorithm

One Paradigm, Many Applications

Variations in Scoring functions

Scoring:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

where $i \in \{0 \dots N\}$ and $j \in \{0 \dots M\}$

$$I_x(i,j) = \max \begin{cases} I_x(i-1,j) + gap_{extend} \\ F(i-1,j) + gap_{open} \end{cases}$$

$$I_y(i,j) = \max \begin{cases} I_y(i,j-1) + gap_{extend} \\ F(i,j-1) + gap_{open} \end{cases}$$

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ Ix(i,j) \\ Iy(i,j) \end{cases}$$

Global Alignment with Affine Gap Penalty

$$V_j(i,j) = Q(y_j) * \max \begin{cases} d * V_m(i,j-1) \\ e * V_j(i,j-1) \end{cases}$$

$$V_i(i,j) = Q(x_i) * \max \begin{cases} d * V_m(i-1,j) \\ e * V_i(i-1,j) \end{cases}$$

$$V_m(i,j) = P(x_i, y_j) * \max \begin{cases} (1-e) * V_j(i-1,j-1) \\ (1-e) * V_i(i-1,j-1) \\ (1-2d) * V_m(i-1,j-1) \end{cases}$$

Viterbi Algorithm

$$S(i,j) = dist(Q_i, R_j) + \min \{S_{i-1,j}, S_{i-1,j-1}, S_{i,j-1}\}$$

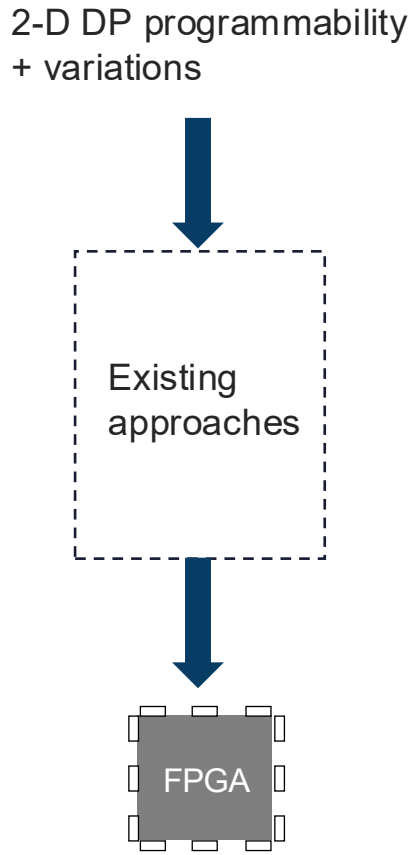
Dynamic Time Warping

List of 2-D DP Kernels for Diverse Applications

Sl No.	Input Alphabets	Kernels	State-of-the-art Tools	Applications	Modifications in DP-HLS
1	DNA	Global Linear Alignment	BLAST, EMBOSS Stretcher	Similarity Search	N/A
2	DNA	Global Affine Alignment	BLAST, EMBOSS Needle	Accurate Similarity Search	Scoring
3	DNA	Local Linear Alignment	BLAST, FASTA, BLAT	Homology Search	Scoring, Initialization and Traceback
4	DNA	Local Affine Alignment	BLAST, LASTZ	Whole Genome Alignment	Scoring, Initialization and Traceback
5	DNA	Global Two-piece Affine Alignment	Minimap2	Long Genome Alignment	Scoring
6	DNA	Overlap Alignment	CANU, Flye	Genome Assembly	Initialization and Traceback
7	DNA	Semi-global Alignment	BWA-MEM	Short Read Alignment	Initialization and Traceback

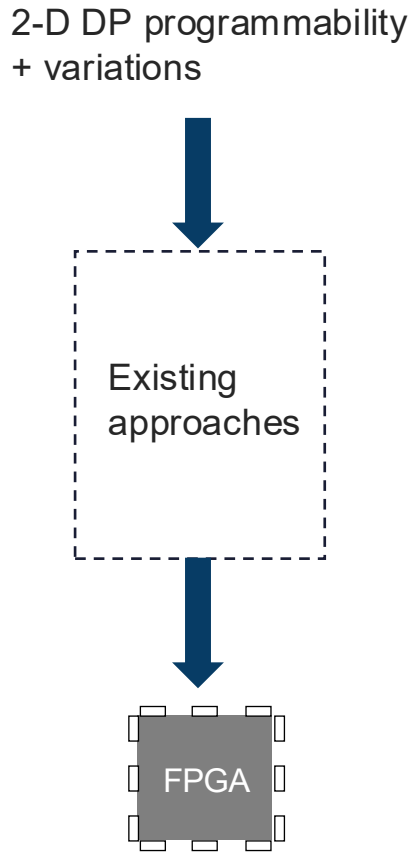
Existing Approaches

- **Custom RTL** based hardware acceleration
- **High-level Synthesis (HLS)** based approaches
- **Software programmable** accelerators



Existing Approaches

- **Custom RTL** based hardware acceleration
- **High-level Synthesis (HLS)** based approaches
- **Software programmable** accelerators



Hardware Acceleration in Bioinformatics

- List of some hardware accelerators for 2-D DP kernels targeting specific applications

X. Fei, Z. Dan, L. Lina, M. Xin, and Z. Chunlei, "**Fpgasw: accelerating large-scale smith–waterman sequence alignment application with backtracking on fpga linear systolic array**," Interdisciplinary Sciences: Computational Life Sciences, vol. 10, pp. 176–188, 2018.

Y. Turakhia, G. Bejerano, and W. J. Dally, "**Darwin: A genomics co-processor provides up to 15,000 x acceleration on long read assembly**," ACM SIGPLAN Notices, vol. 53, no. 2, pp. 199–213, 2018.

Y. Turakhia, S. D. Goenka, G. Bejerano, and W. J. Dally, "**Darwin-wga: A co-processor provides increased sensitivity in whole genome alignments with high speedup**," in 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE, 2019, pp. 359–372.

A. Haghi, S. Marco-Sola, L. Alvarez, D. Diamantopoulos, C. Hagleitner, and M. Moreto, "**An fpga accelerator of the wavefront algorithm for genomics pairwise alignment**," in 2021 31st International Conference on Field-Programmable Logic and Applications (FPL). IEEE, 2021, pp. 151–159.

D. Fujiki, S. Wu, N. Ozog, K. Goliya, D. Blaauw, S. Narayanasamy, and R. Das, "**Seedex: A genome sequencing accelerator for optimal alignments in subminimal space**," in 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2020, pp. 937–950.

T. Dunn, H. Sadasivan, J. Wadden, K. Goliya, K.-Y. Chen, R. Das, D. Blaauw, and S. Narayanasamy, "**SquiggleFilter: An Accelerator for Portable Virus Detection**," Sep. 2021, arXiv:2108.06610 [q-bio]. [Online].

Hardware Acceleration in Bioinformatics

- List of some hardware accelerators for 2-D DP kernels targeting specific applications

X. Fei, Z. Dan, L. Lina, M. Xin, and Z. Chunlei, "**Fpgasw: accelerating large-scale smith–waterman sequence alignment application with backtracking on fpga linear systolic array**," Interdisciplinary Sciences: Computational Life Sciences, vol. 10, pp. 176–188, 2018.

→ Accelerates Smith Waterman Kernel

Y. Turakhia, G. Bejerano, and W. J. Dally, "**Darwin: A genomics co-processor provides up to 15,000 x acceleration on long read assembly**," ACM SIGPLAN Notices, vol. 53, no. 2, pp. 199–213, 2018.

→ Accelerates Smith Waterman Kernel with tiling

Y. Turakhia, S. D. Goenka, G. Bejerano, and W. J. Dally, "**Darwin-wga: A co-processor provides increased sensitivity in whole genome alignments with high speedup**," in 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE, 2019, pp. 359–372.

→ Accelerates X-drop (adaptive banding) with tiling

Hardware Acceleration in Bioinformatics

- List of some hardware accelerators for 2-D DP kernels targeting specific applications

X. Fei, Z. Dan, L. Lina, M. Xin, and Z. Chunlei, "**Fpgasw: accelerating large-scale smith–waterman sequence alignment application with backtracking on fpga linear systolic array**," Interdisciplinary Sciences: Computational Life Sciences, vol. 10, pp. 176–188, 2018.

Y. Turakhia, G. Bejerano, and W. J. Dally, "**Darwin: A genomics co-processor provides up to 15,000 x acceleration on long read assembly**," ACM SIGPLAN Notices, vol. 53, no. 2, pp. 199–213, 2018.

Y. Turakhia, S. D. Goenka, G. Bejerano, and W. J. Dally, "**Darwin-wga: A co-processor provides increased sensitivity in whole genome alignments with high speedup**," in 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE, 2019, pp. 359–372.

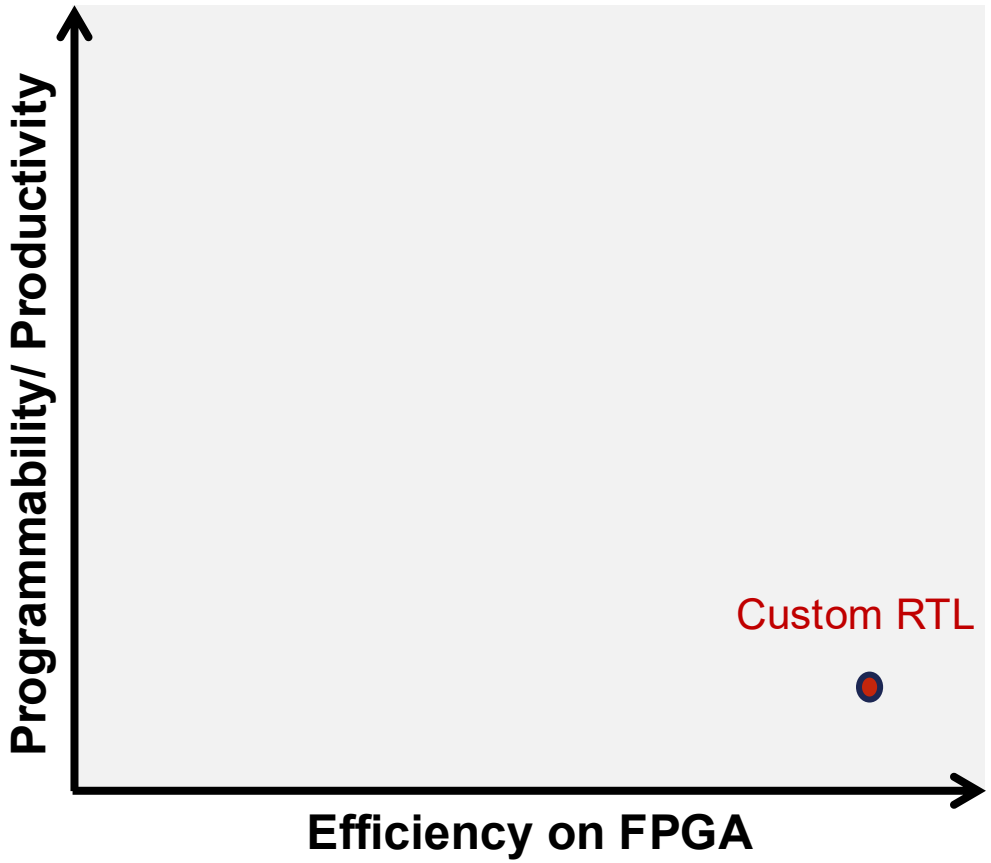
A. Haghi, S. Marco-Sola, L. Alvarez, D. Diamantopoulos, C. Hagleitner, and M. Moreto, "**An fpga accelerator of the wavefront algorithm for genomics pairwise alignment**," in 2021 31st International Conference on Field-Programmable Logic and Applications (FPL). IEEE, 2021, pp. 151–159.

D. Fujiki, S. Wu, N. Ozog, K. Goliya, D. Blaauw, S. Narayanasamy, and R. Das, "**Seedex: A genome sequencing accelerator for optimal alignments in subminimal space**," in 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2020, pp. 937–950.

T. Dunn, H. Sadasivan, J. Wadden, K. Goliya, K.-Y. Chen, R. Das, D. Blaauw, and S. Narayanasamy, "**SquiggleFilter: An Accelerator for Portable Virus Detection**," Sep. 2021, arXiv:2108.06610 [q-bio]. [Online].

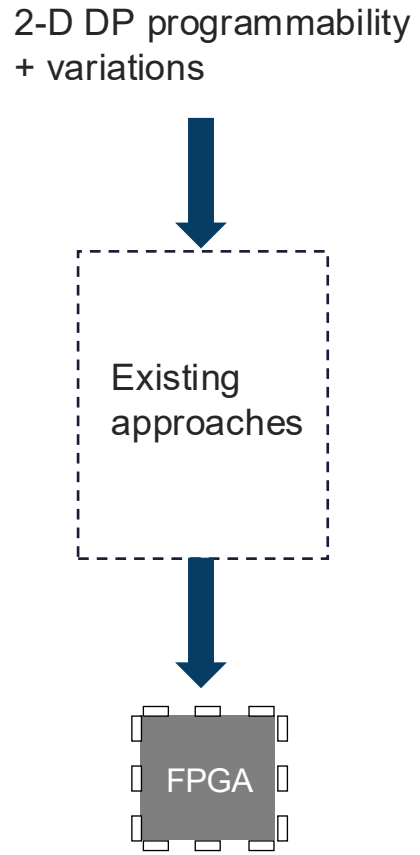
Limitations to Existing Hardware Approaches

- Typically **designed for one or a few fixed applications**
- Implementations rely on low-level hardware description languages (Ex: Verilog, VHDL)
- Any algorithmic change **require months of manual implementation effort**



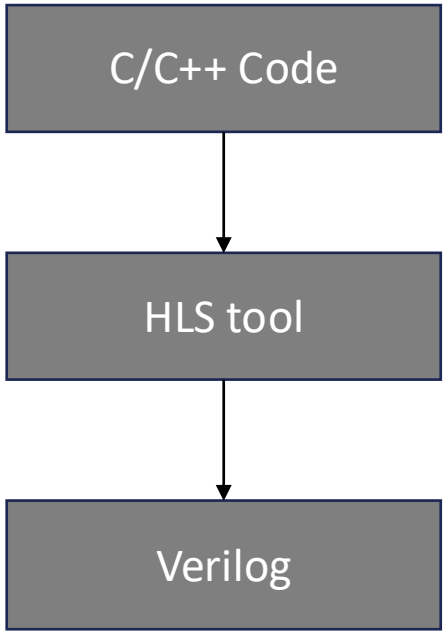
Existing Approaches

- **Custom RTL** based hardware acceleration
- **High-level Synthesis (HLS)** based approaches
- **Software programmable** accelerators



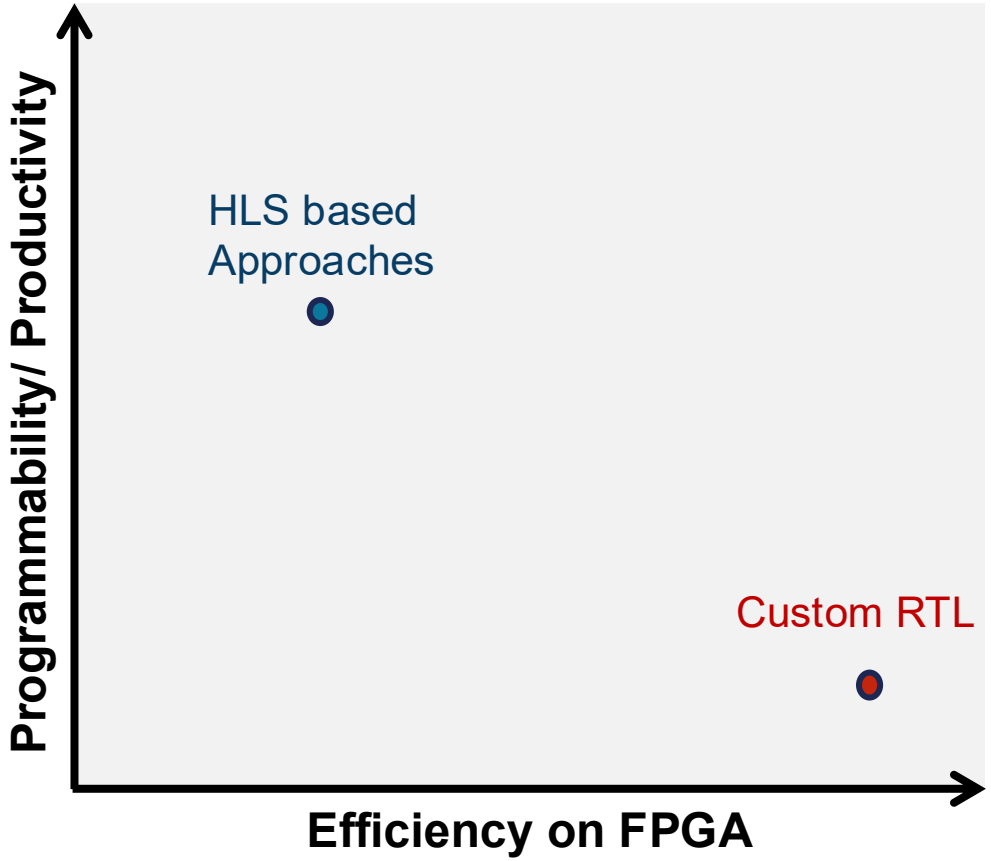
High-Level Synthesis (HLS)

- HLS enables **C/C++ to hardware**
- **Reduces hardware implementation effort** from months to weeks



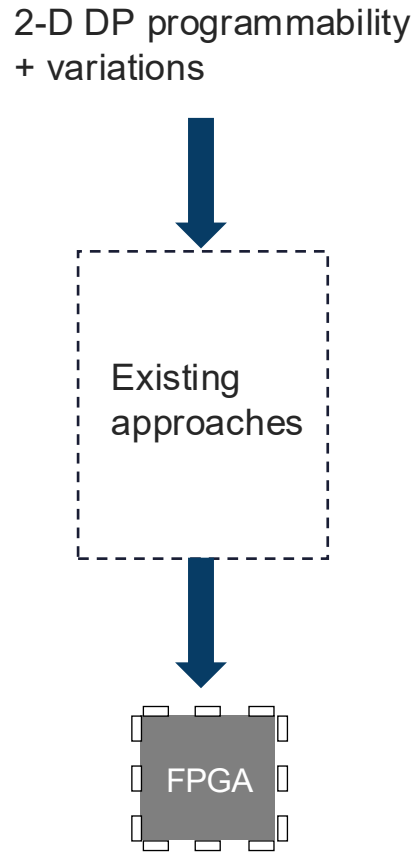
High-Level Synthesis (HLS)

- HLS enables C/C++ to hardware
- **Reduces hardware implementation effort** from months to weeks
- HLS **does not automatically guarantee efficiency**
- Achieving RTL-like performance still requires:
  - **Careful usage of pragmas**
  - **Hardware expertise**



Existing Approaches

- **Custom RTL** based hardware acceleration
- **High-level Synthesis (HLS)** based approaches
- **Software programmable** accelerators



Software-Programmable Accelerator

- Key idea
  - Fixed hardware architecture
  - Software-programmable instruction set

Software-Programmable Accelerator

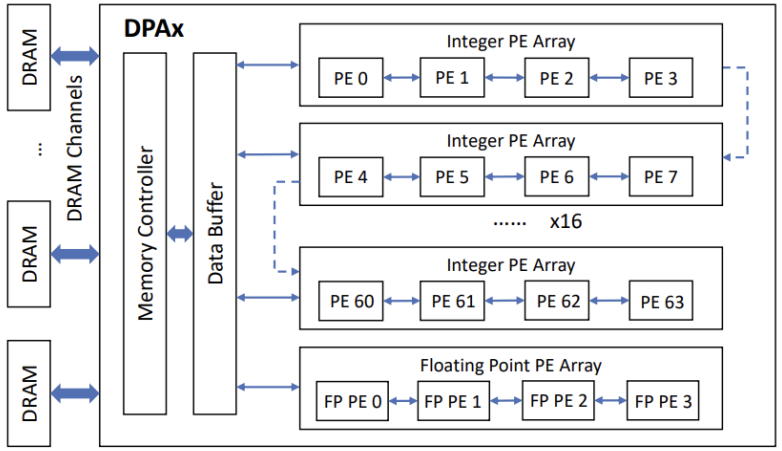
- Key idea
  - Fixed hardware architecture
  - Software programmable instruction set

- Example: **GenDP**
  - DP kernels maps on programmable PE array
  - **Provides flexibility** using **GenDP ISA**
  - Fixed hardware architecture with **flexible PE interconnects**

GenDP: A Framework of Dynamic Programming Acceleration for Genome Sequencing Analysis

Authors: Yufeng Gu, Arun Subramaniyan, Tim Dunn, Alireza Khadem, Kuan-Yu Chen, Somnath Paul, Md Vasimuddin, Sanchit Misra, David Blaauw, Satish Narayanasamy, Reetuparna Das Authors Info & Claims

ISCA '23: Proceedings of the 50th Annual International Symposium on Computer Architecture
Article No.: 25, Pages 1 - 15 • https://doi.org/10.1145/3579371.3589060



DPAx Architecture

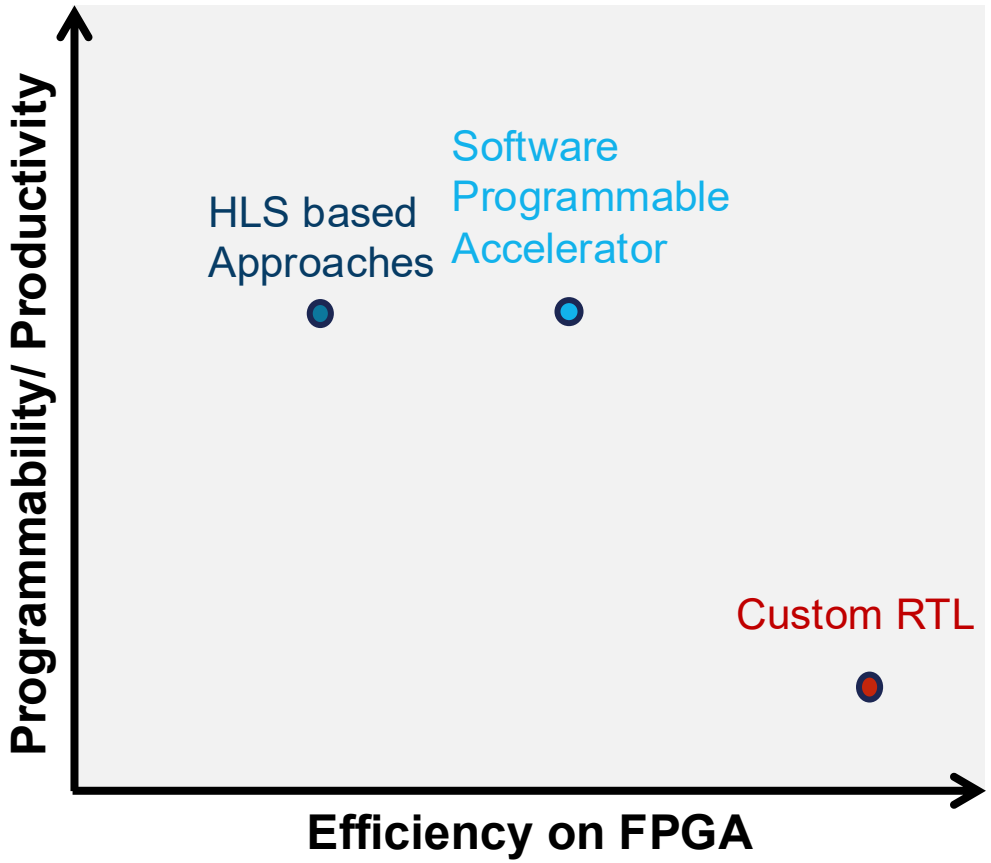
Type	Instruction	Assembly
Arithmetic	add	add rd rs1 rs2
	addi	addi rd rs1 #imm
Move	li	li [dest addr] #imm
	mv	mv [dest addr] [src addr]
Branch	beq/bne/bge/blt	branch rs1 rs2 offset
Other	set	set PC
	no-op	no operation
	halt	halt

GenDP Control ISA

Image source: Gu et al, ISCA (2023)

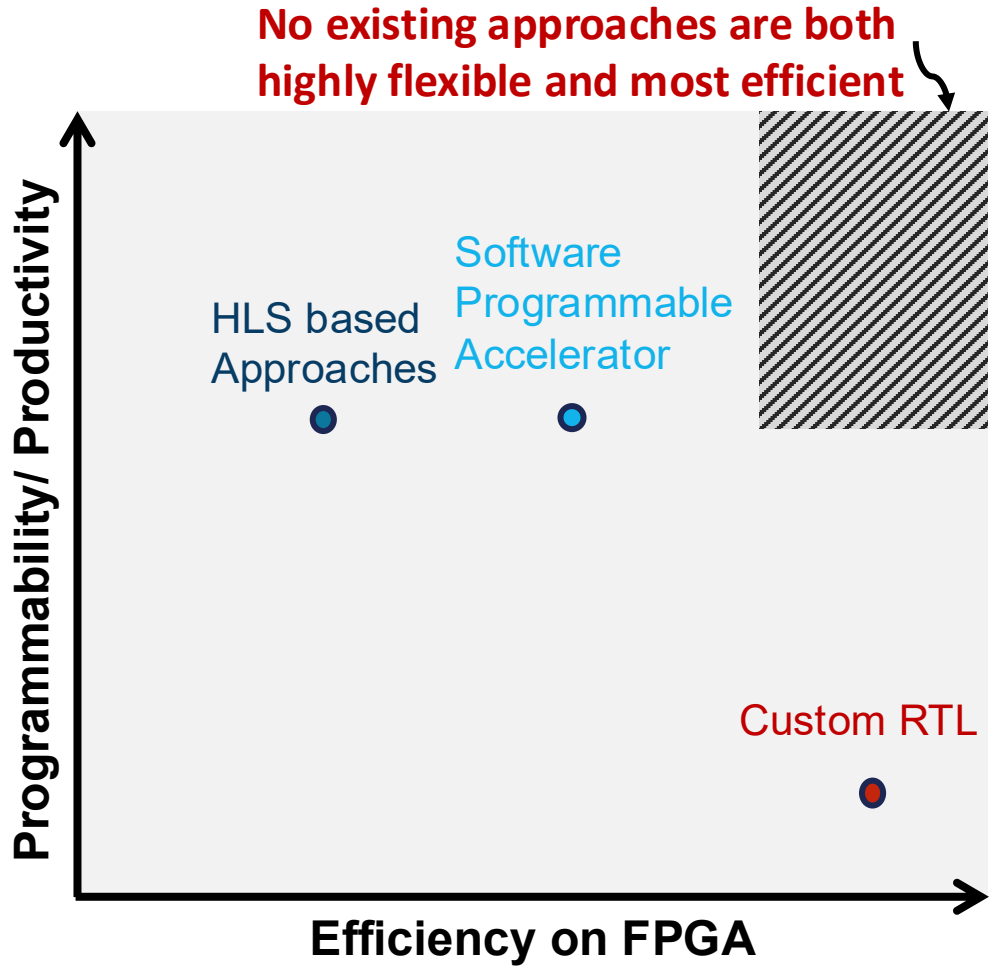
Software-Programmable Accelerator

- Key idea
  - Fixed hardware architecture
  - Software programmable instruction set
- Example: **GenDP**
  - DP kernels maps on programmable PE array
  - **Provides flexibility** using **GenDP ISA**
  - Fixed hardware architecture with **flexible PE interconnects**
  - **Overheads from instruction fetch and decode**
  - **Loses performance for programmability**



Existing Approaches

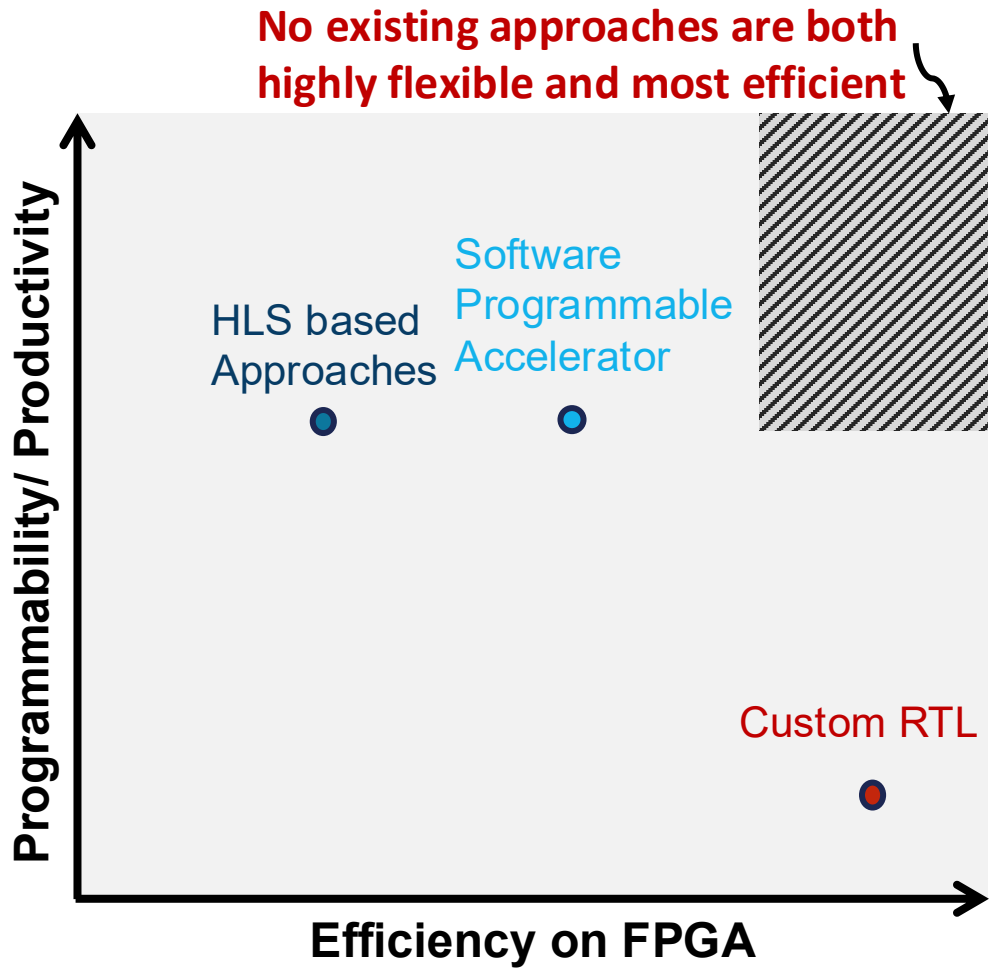
- **Custom RTL** based hardware acceleration
- **High-level Synthesis (HLS)** based approaches
- **Software programmable** accelerators



Existing Approaches

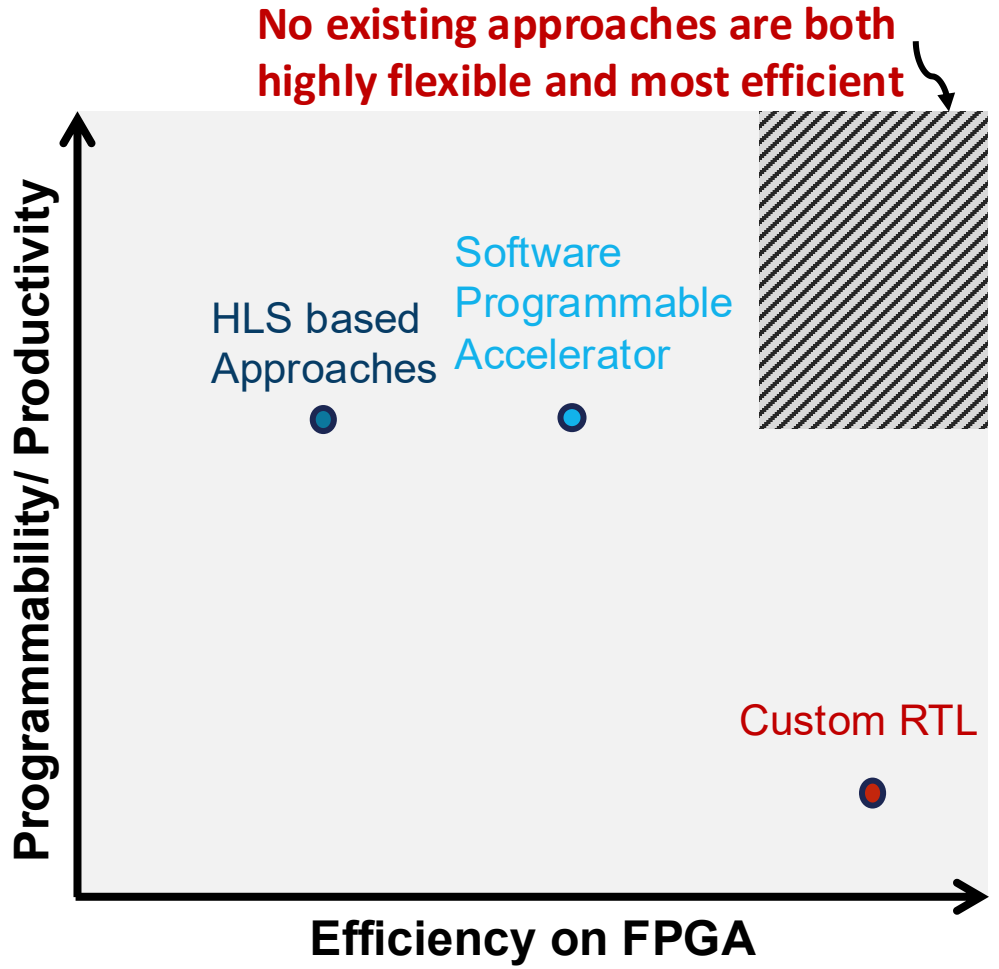
- **Custom RTL** based hardware acceleration
- **High-level Synthesis (HLS)** based approaches
- **Software programmable** accelerators

- Ideal accelerator should offer:
  - **Algorithmic flexibility**
  - **Near custom RTL efficiency**



Objective

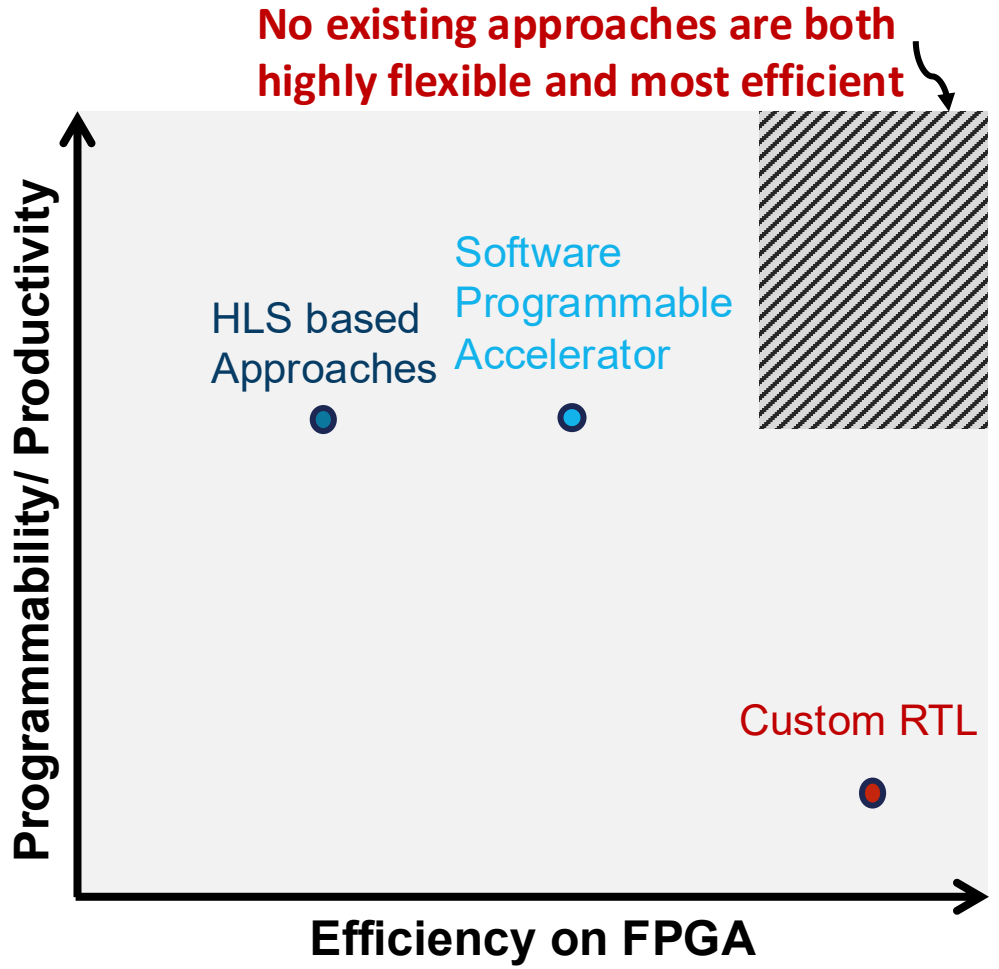
Develop a framework, which is:



Objective

Develop a framework, which is:

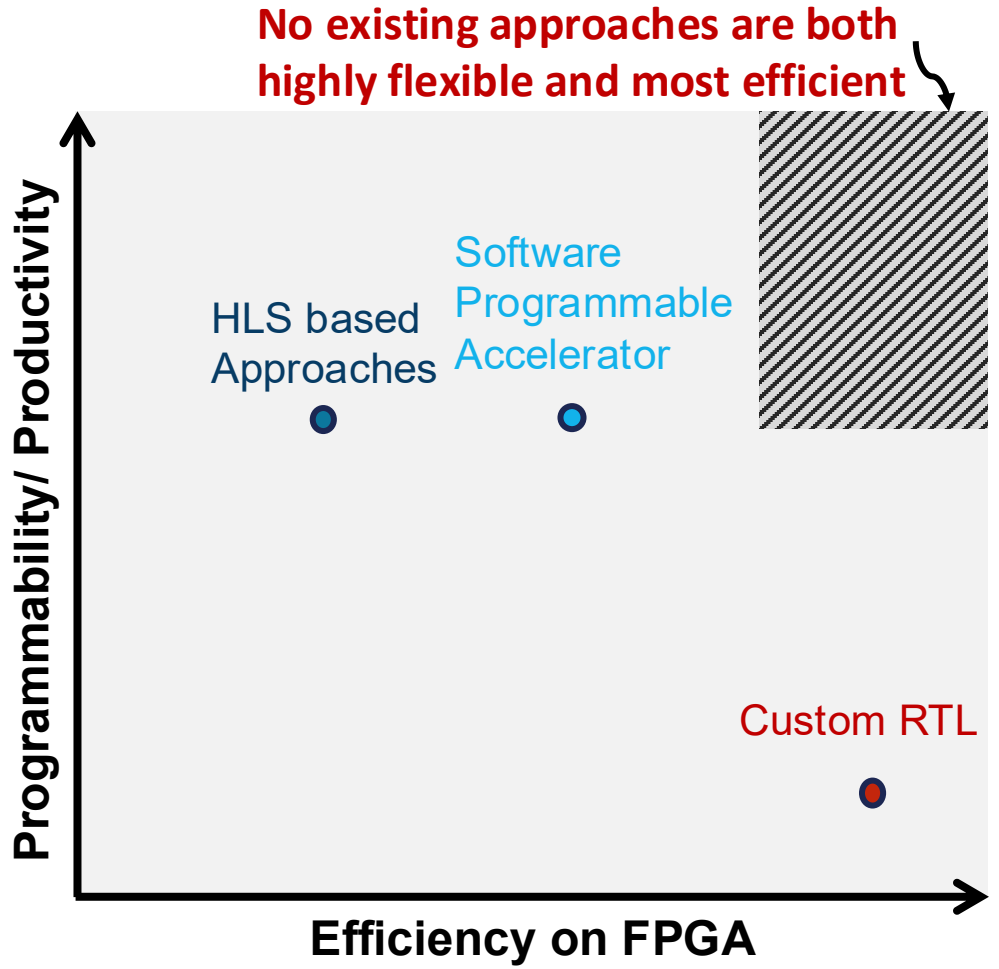
- **Highly programmable** to define any new 2-D DP kernel



Objective

Develop a framework, which is:

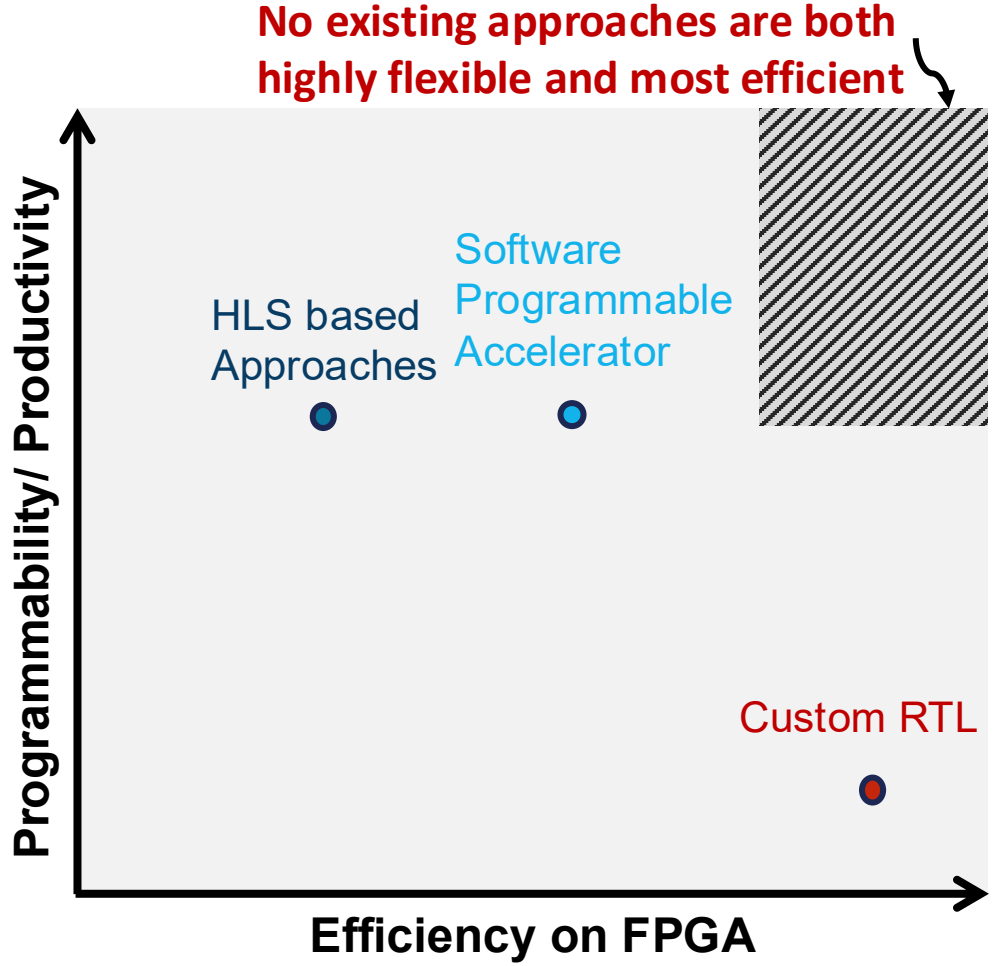
- **Highly programmable** to define any new 2-D DP kernel
- Create **efficient hardware designs** on FPGA



Objective

Develop a framework, which is:

- **Highly programmable** to define any new 2-D DP kernel
- Create **efficient hardware designs** on FPGA
- **No hardware expertise** needed



Outline

- Dynamic Programming (DP) in Bioinformatics
- DP Algorithmic Variations and Hardware Accelerators
- **DP-HLS Framework**
- Key Contributions and Results
- Conclusion and Future Applications

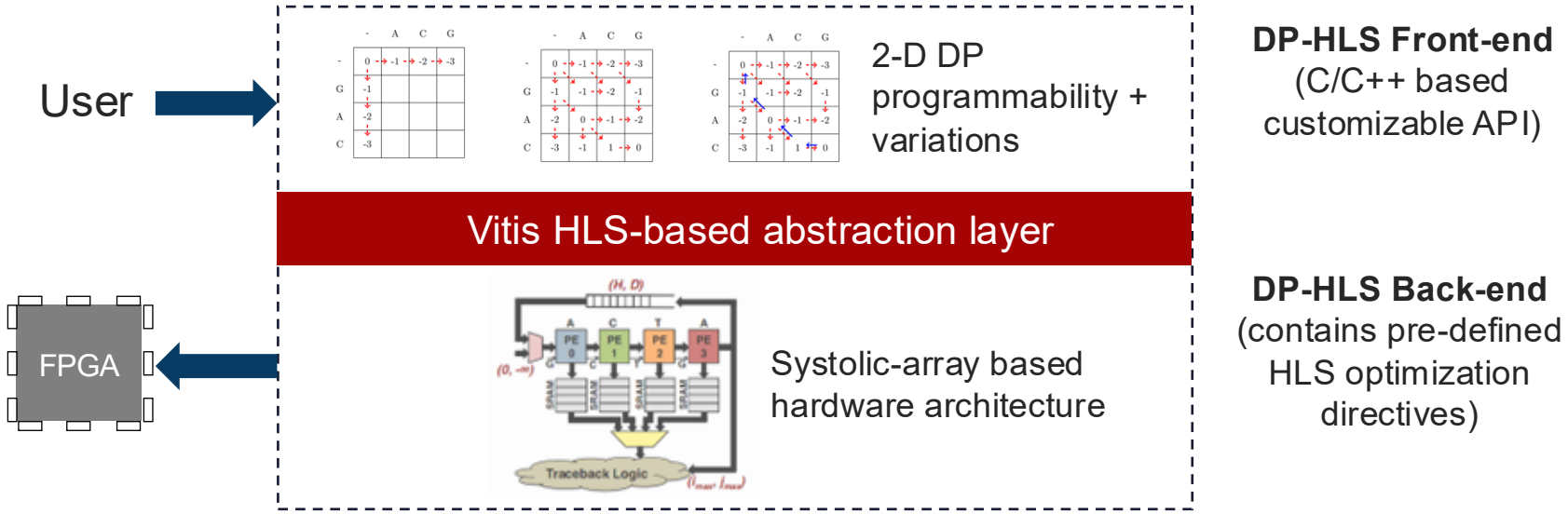
DP-HLS: A New Abstraction Layer

- **Key insight:** Broad class of 2-D DP algorithms maps to the **same hardware primitive (linear systolic arrays)**

DP-HLS: A New Abstraction Layer

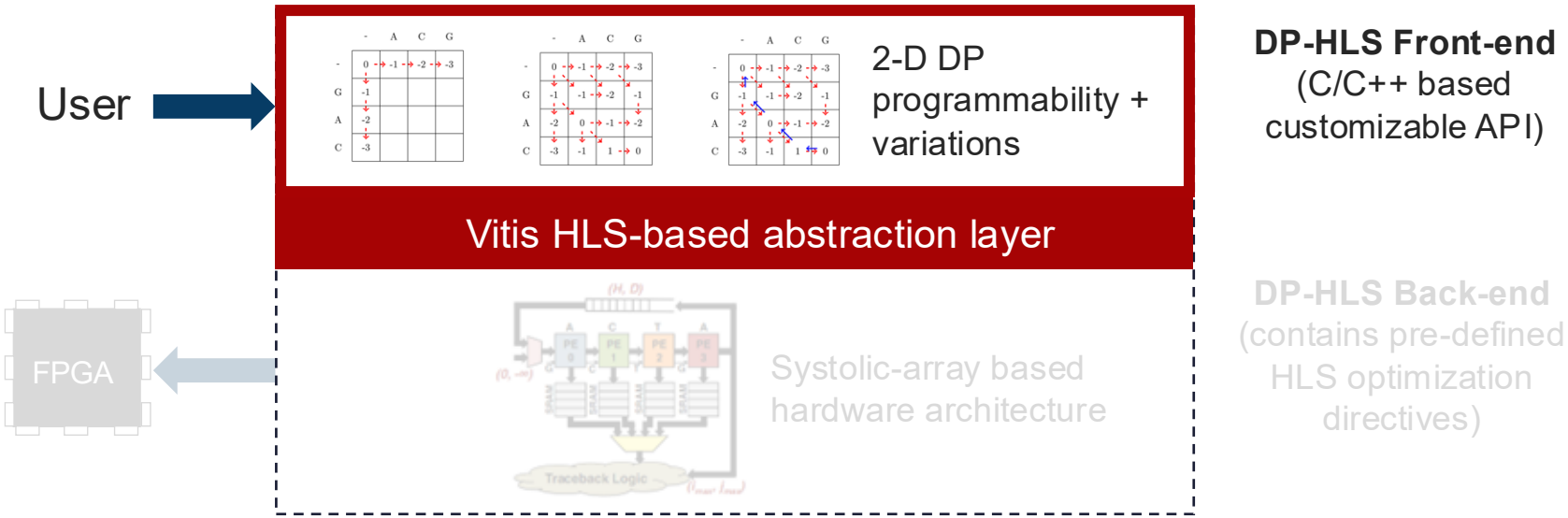
- **Key insight:** Broad class of 2-D DP algorithms maps to the **same hardware primitive (linear systolic arrays)**

- Introduces **a layer of abstraction in the HLS flow**



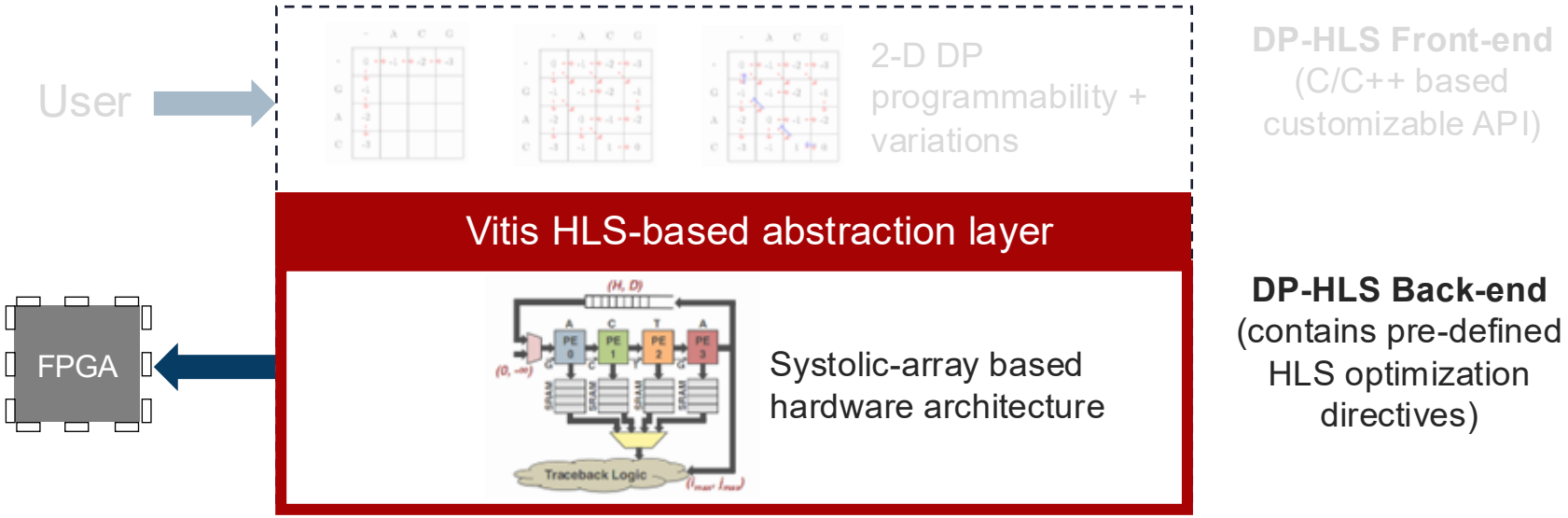
DP-HLS: Front-End Design

- **Customizable front-end** with 2-D DP algorithmic specifications → **provides flexibility**
- Users can **define new kernels** in C++ **without HLS or hardware design expertise**
- User can provide their own **custom function** along with parametric configuration



DP-HLS: Back-End

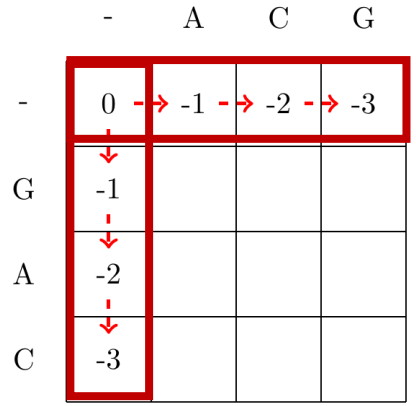
- **Unified back-end** with hardware-specific HLS directives → **provides efficiency**
- Generates an **efficient linear systolic array architecture**
- **Each hardware architecture is customized** to match the target algorithm



DP-HLS: Front-End Design

- **Initialize row and column scores of DP matrix**

DP Matrix:



Initialization:

$$F(0,j) = -j*d \text{ where } j \in \{0 \dots N\}$$
$$F(i,0) = -i*d \text{ where } i \in \{0 \dots M\}$$

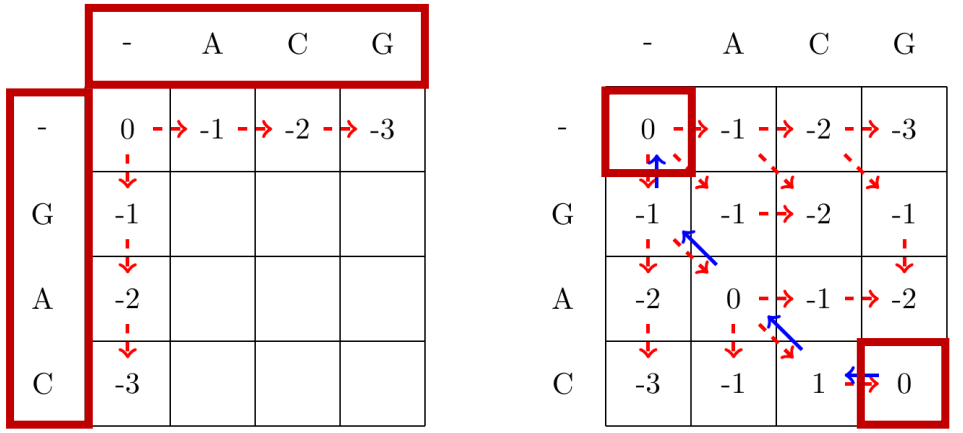
Lines of code:

```
type_t gap = scoring_params.linear_gap;
for (int i = 0; i < MAX_REFERENCE_LENGTH; i++){
    init_row_scr[i][0] = i*gap; }
for (int i = 0; i < MAX_QUERY_LENGTH; i++){
    init_col_scr[i][0] = i*gap; }
```

DP-HLS: Front-End Design

- Initialize row and column scores of DP matrix
- **Customize data types and parameters**
  - Sequence Alphabets
  - Scoring parameters
  - Traceback start, end points

DP Matrix:



Scoring:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

where $i \in \{0 \dots N\}$ and $j \in \{0 \dots M\}$

DP-HLS: Front-End Design

- Initialize row and column scores of DP matrix
- Customize data types and parameters
  - Sequence Alphabets
  - Scoring parameters
  - Traceback start, end points
- **Customize scoring function**

Scoring:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

where $i \in \{0 \dots N\}$ and $j \in \{0 \dots M\}$

Lines of code:

```
type_t linear_gap = params.linear_gap;
type_t ins = dp_mem_left[0] + linear_gap;
type_t del = dp_mem_up[0] + linear_gap;
type_t match = dp_mem_diag[0] + (lc_qry_val ==
    lc_ref_val) ? params.match : params.mismatch;
```

DP-HLS: Front-End Design

- Initialize row and column scores of DP matrix
- Customize data types and parameters
  - Sequence Alphabets
  - Scoring parameters
  - Traceback start, end points
- Customize scoring function
- **Specify traceback strategy**

Traceback:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(x_i, y_j) & \text{(move diagonal)} \\ F(i-1,j) + d & \text{(move up)} \\ F(i,j-1) + d & \text{(move left)} \end{cases}$$

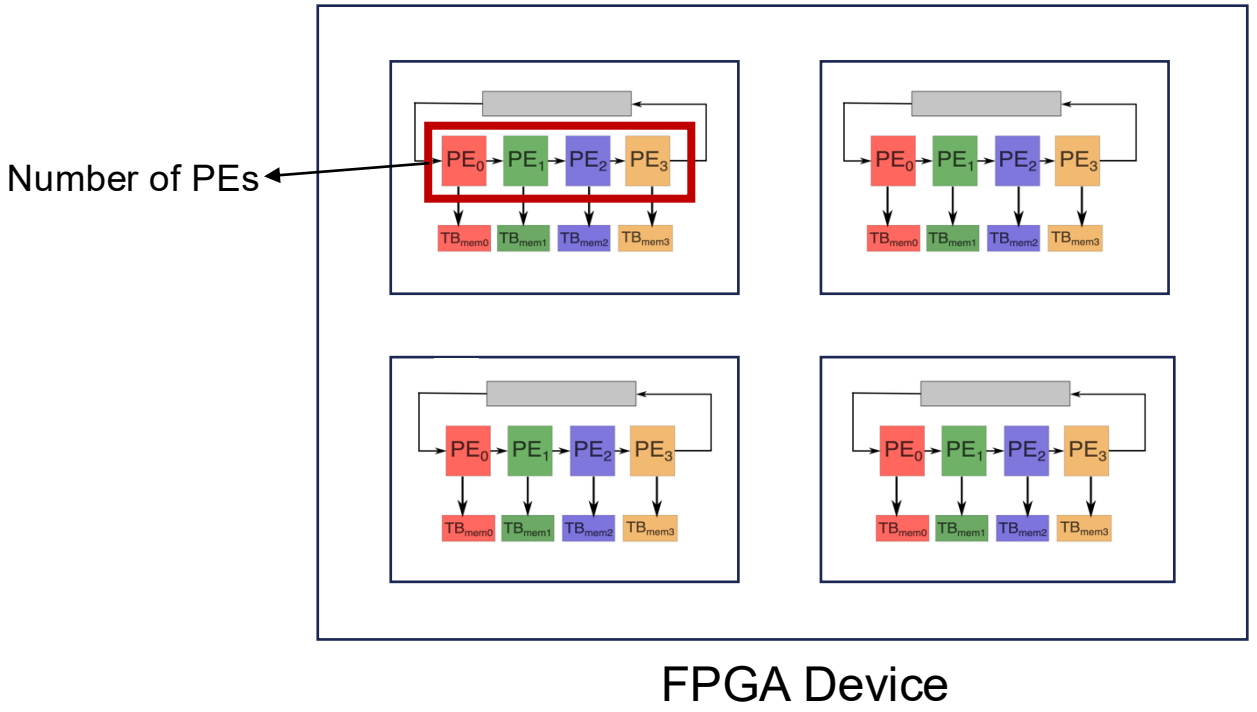
Start at F(N,M), terminate at F(0,0)

Lines of code:

```
if (tb_state == TB_STATE::MM){
    if (tb_ptr == TB_DIAG){tb_move = AL_MMI; }
    else if (tb_ptr == TB_UP){tb_move = AL_DEL; }
    else if (tb_ptr == TB_LEFT){tb_move = AL_INS;}
    else if (tb_ptr == TB_END) {tb_move = AL_END;}
    else {tb_move = AL_END;}
    tb_state = TB_STATE::MM;}
```

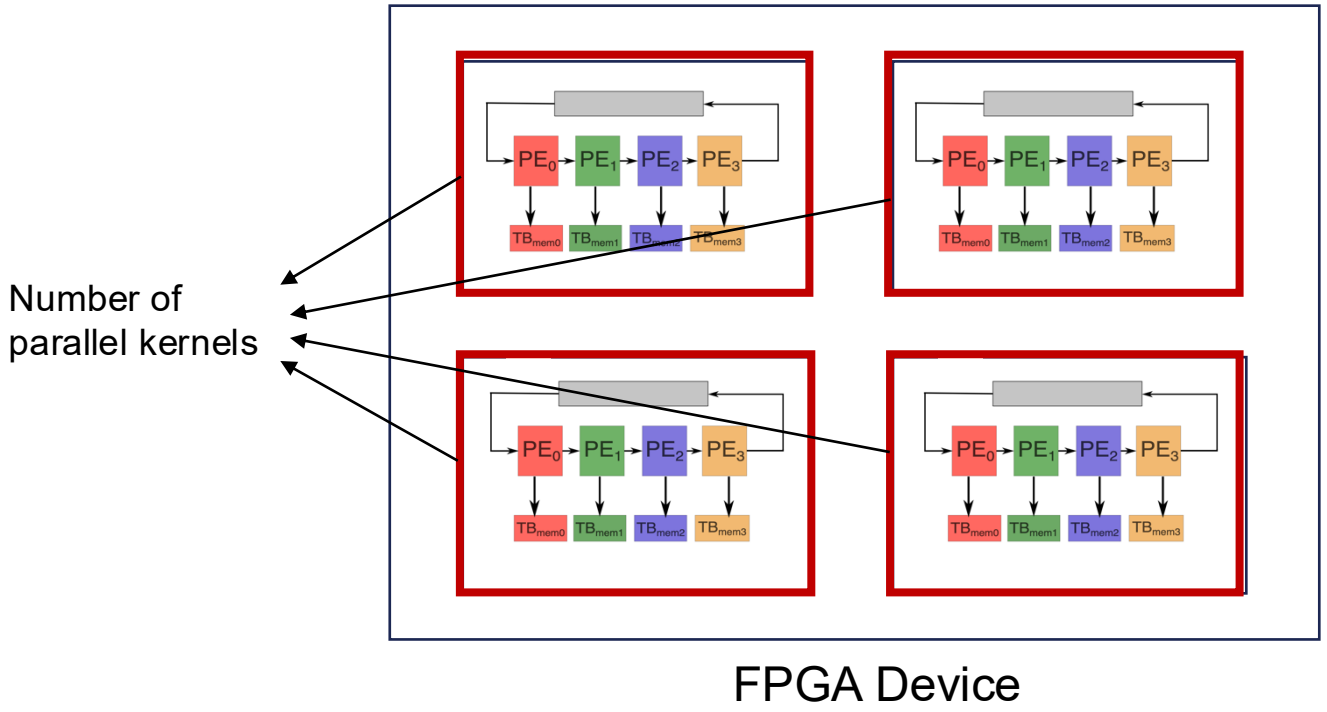
DP-HLS: Front-End Design

- Initialize row and column scores of DP matrix
- Customize data types and parameters
  - Sequence Alphabets
  - Scoring parameters
  - Traceback start, end points
- Customize scoring function
- Specify traceback strategy
- **Specify inner and outer loop parallelism**



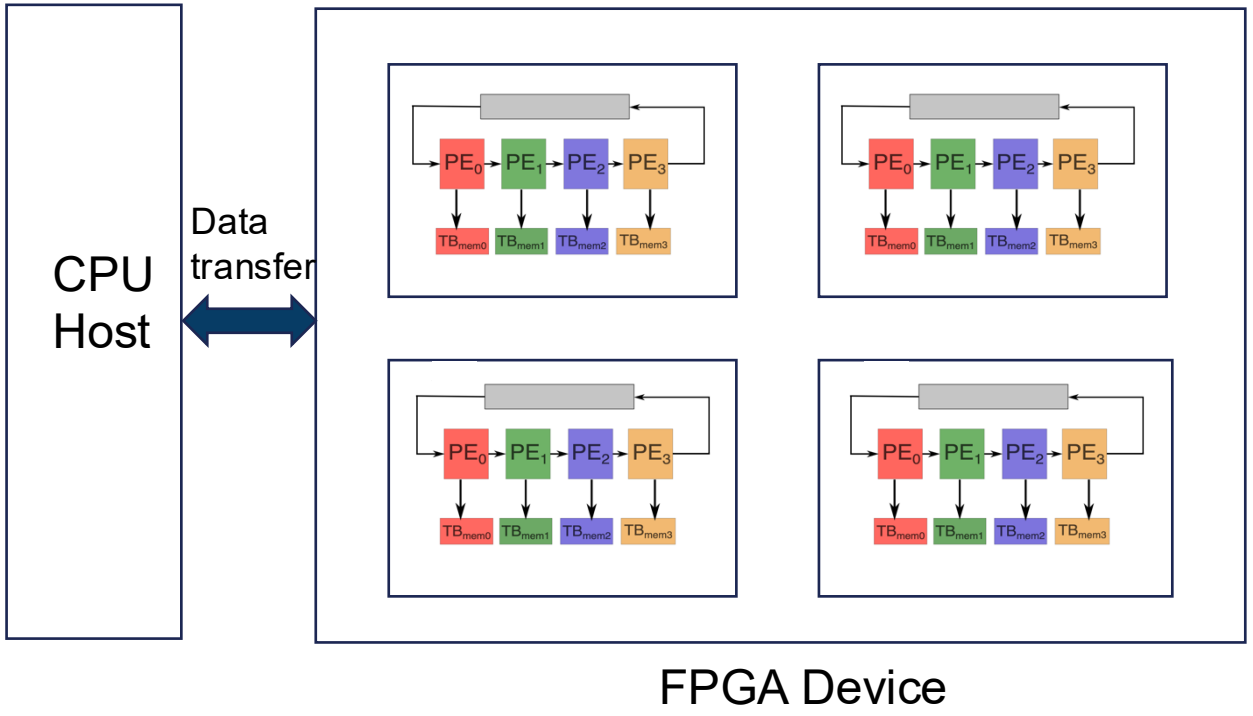
DP-HLS: Front-End Design

- Initialize row and column scores of DP matrix
- Customize data types and parameters
  - Sequence Alphabets
  - Scoring parameters
  - Traceback start, end points
- Customize scoring function
- Specify traceback strategy
- **Specify inner and outer loop parallelism**



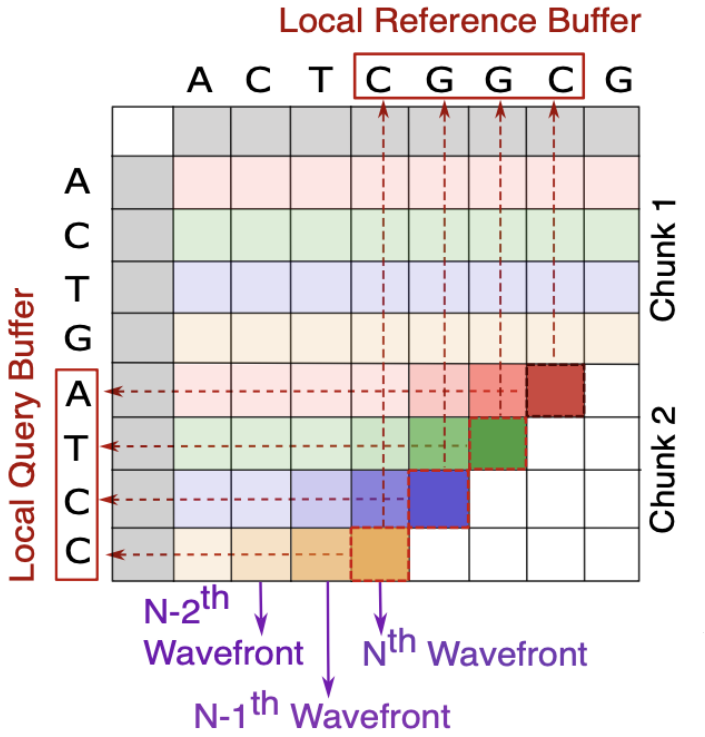
DP-HLS: Front-End Design

- Initialize row and column scores of DP matrix
- Customize data types and parameters
  - Sequence Alphabets
  - Scoring parameters
  - Traceback start, end points
- Customize scoring function
- Specify traceback strategy
- Specify inner and outer loop parallelism
- **Write host-side program**



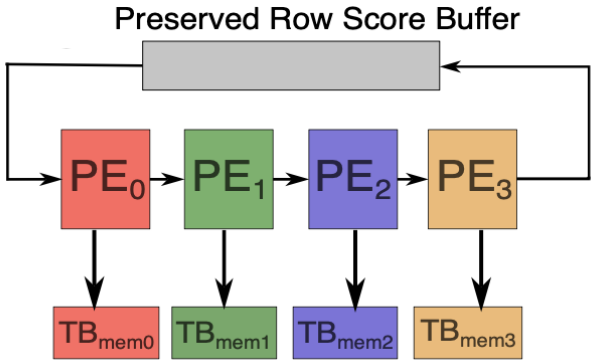
DP-HLS: Back-End - Key Optimizations

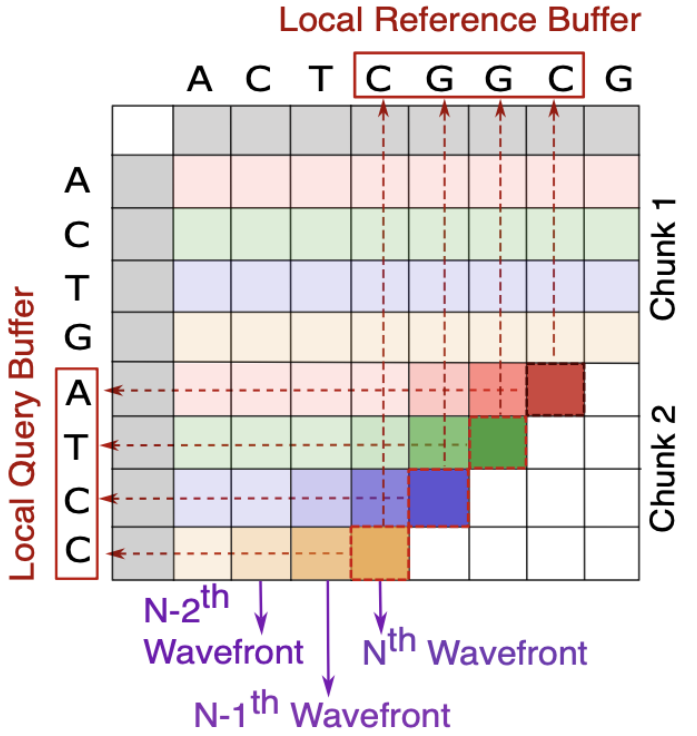
- Computation follows **wavefront parallelism**



DP-HLS: Back-End - Key Optimizations

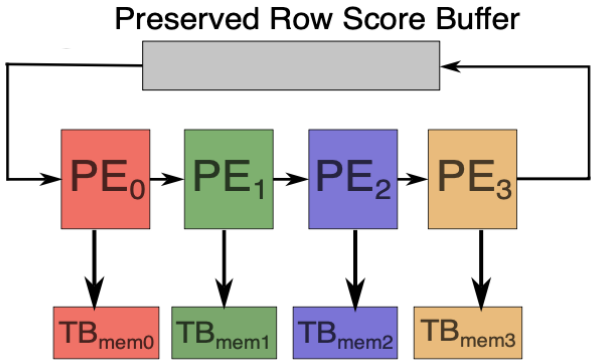
- Computation follows **wavefront parallelism**
- Generates **1-D systolic array architecture**

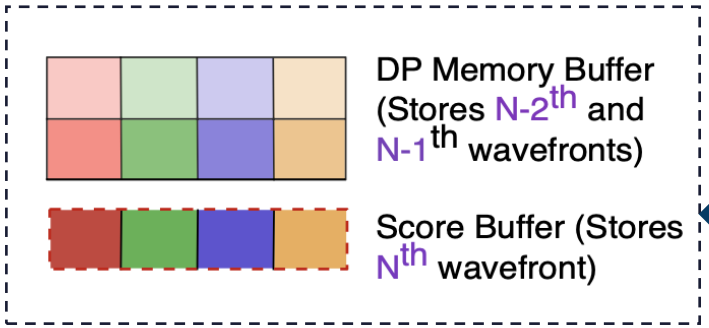


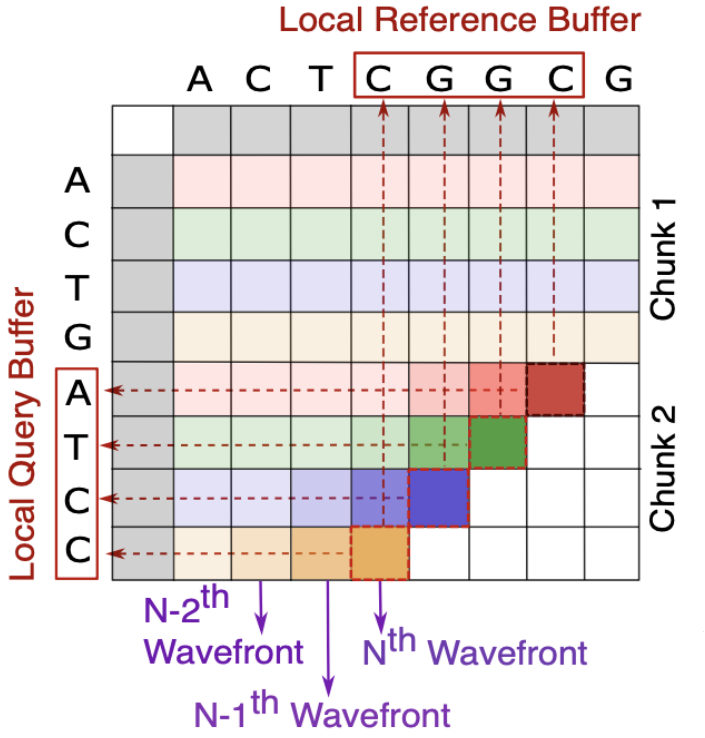


DP-HLS: Back-End - Key Optimizations

- Computation follows **wavefront parallelism**
- Generates **1-D systolic array architecture**
- Local buffers are partitioned for parallel access

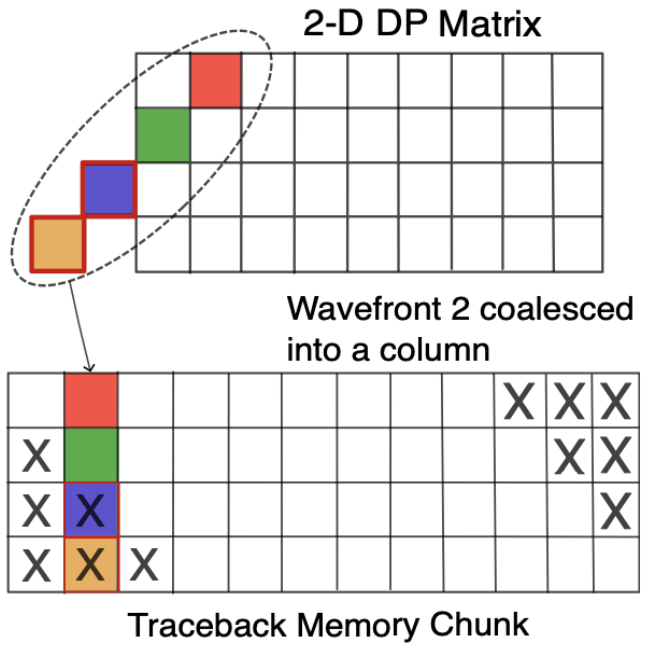






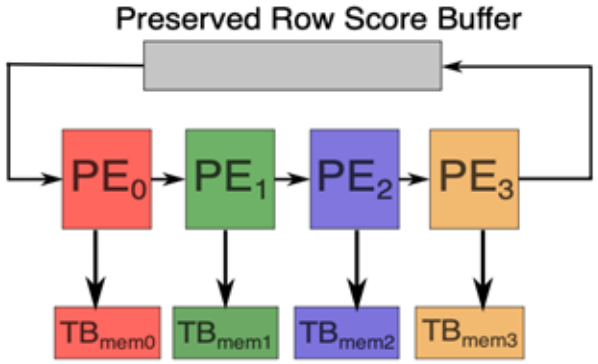
DP-HLS: Back-End - Key Optimizations

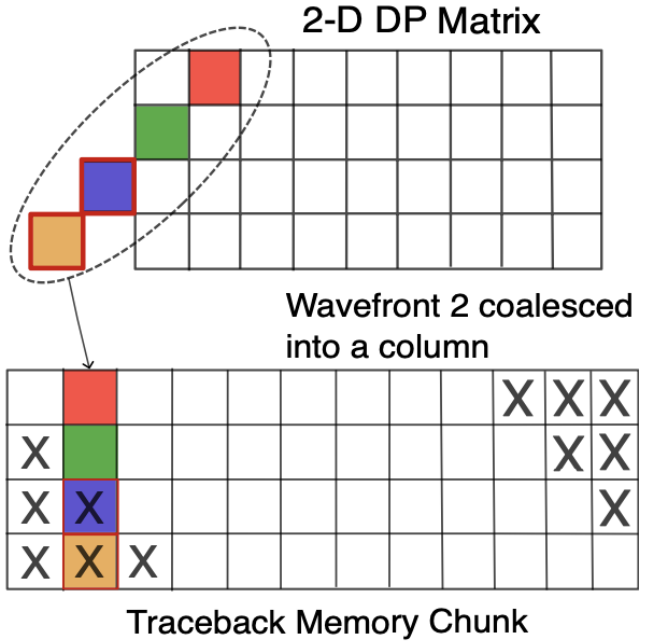
- Performs **memory address coalescing** to map wavefronts in Traceback matrix



DP-HLS: Back-End - Key Optimizations

- Performs **memory address coalescing** to map wavefronts in Traceback matrix
- Each PE accesses to **independent memory banks**





Outline

- Dynamic Programming (DP) in Bioinformatics
- DP Algorithmic Variations and Hardware Accelerators
- DP-HLS Framework
- **Key Contributions and Results**
- Conclusion and Future Applications

DP-HLS: Key Contributions & Results

- Showcases **versatility** to support **all possible types of 2D DP kernels on FPGA**

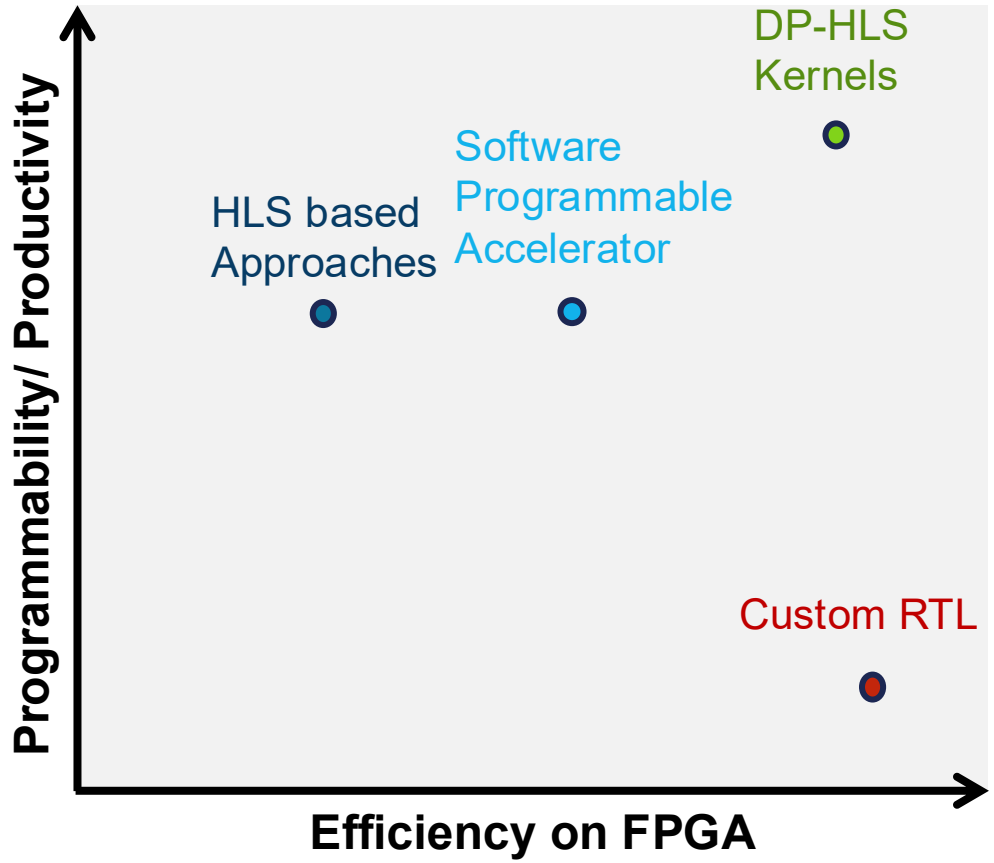
DP-HLS: Key Contributions & Results

- Showcases **versatility** to support **all possible types of 2D DP kernels on FPGA**

Sl No.	Input Alphabets	Kernels	State-of-the-art Tools	Applications	Modifications in DP-HLS
1	DNA	Global Linear Alignment	BLAST, EMBOSS Stretcher	Similarity Search	N/A
2	DNA	Global Affine Alignment	BLAST, EMBOSS Needle	Accurate Similarity Search	Scoring
3	DNA	Local Linear Alignment	BLAST, FASTA, BLAT	Homology Search	Scoring, Initialization and Traceback
4	DNA	Local Affine Alignment	BLAST, LASTZ	Whole Genome Alignment	Scoring, Initialization and Traceback
5	DNA	Global Two-piece Affine Alignment	Minimap2	Long Genome Alignment	Scoring
6	DNA	Overlap Alignment	CANU, Flye	Genome Assembly	Initialization and Traceback
7	DNA	Semi-global Alignment	BWA-MEM	Short Read Alignment	Initialization and Traceback
8	Seq. Profiles	Profile Alignment	CLUSTAL-W, MUSCLE	Multiple Sequence Alignment	Sequence Alphabet and Scoring
9	Complex Nos.	Dynamic Time Warping Algorithm	SquiggleKit	Basecalling	Sequence Alphabet and Scoring
10	DNA	Viterbi Algorithm	HMMER, Augustus	Remote Homology Search, Gene Prediction	Scoring (no Traceback)
11	DNA	Banded Global Linear Alignment	BLAST, Bowtie	Fast Similarity Search	Scoring and Initialization
12	DNA	Banded Local Affine Alignment	Minimap2	Long Read Assembly	Initialization and Scoring (no Traceback)
13	DNA	Banded Global Two-piece Affine Alignment	Minimap2	Long Read Assembly	Scoring, Initialization and Traceback
14	Integers	Semi-global DTW	SquiggleFilter, RawHash	Basecalling	Sequence Alphabet and Scoring
15	Amino Acids	Local Linear Alignment with protein sequences	EMBOSS Water, BLASTp, DIAMOND	Protein Sequence Alignment	Sequence Alphabet and Scoring

DP-HLS: Key Contributions & Results

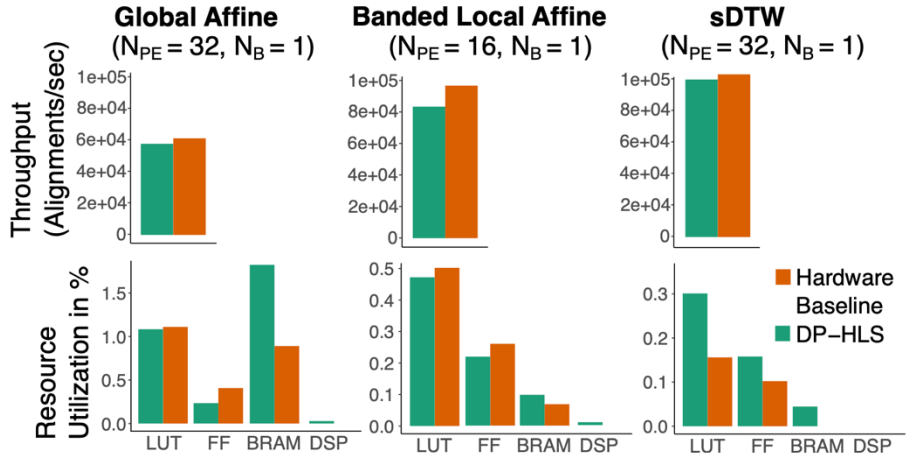
- Showcases **versatility** to support **all possible types of 2D DP kernels on FPGA**
- Gives **competitive performance to RTL designs**



DP-HLS: Key Contributions & Results

- Showcases **versatility** to support **all possible types of 2D DP kernels on FPGA**

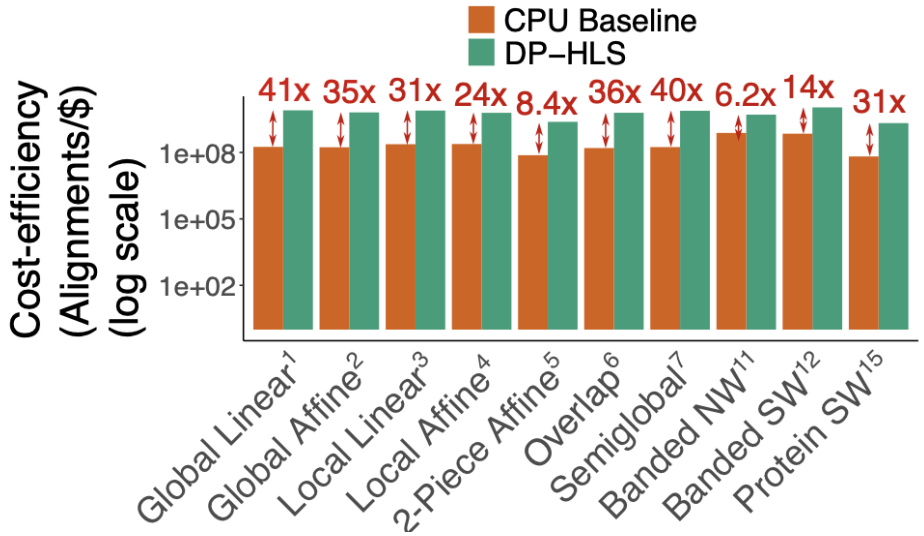
- Gives **competitive performance to RTL designs**



Methodology:
DP-HLS: AWS EC2 F1 instance
Global Affine: Darwin (Turakhia et al, ASPLOS 2018
Banded Local Affine: Darwin WGA (Turakhia et al, HPCA 2019)
sDTW: Squigglefilter (Dunn et al, MICRO 2021)

DP-HLS: Key Contributions & Results

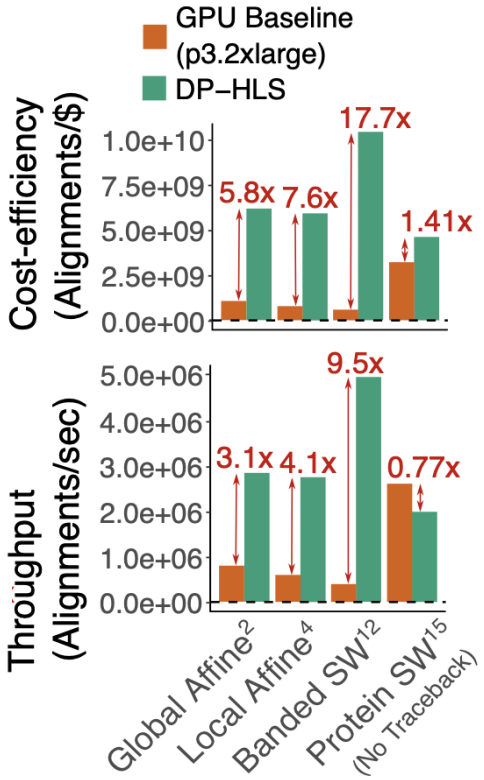
- Showcases **versatility** to support **all possible types of 2D DP kernels on FPGA**
- Gives **competitive performance to RTL designs**
- Achieves up to **41× higher cost-efficiency** than CPU baselines



Methodology:
DP-HLS: AWS EC2 F1 instance
CPU Baseline: SeqAn3, Minimap2. EMBOSS Water on 36-core AWS EC2 CPU instance (c4.8xlarge)

DP-HLS: Key Contributions & Results

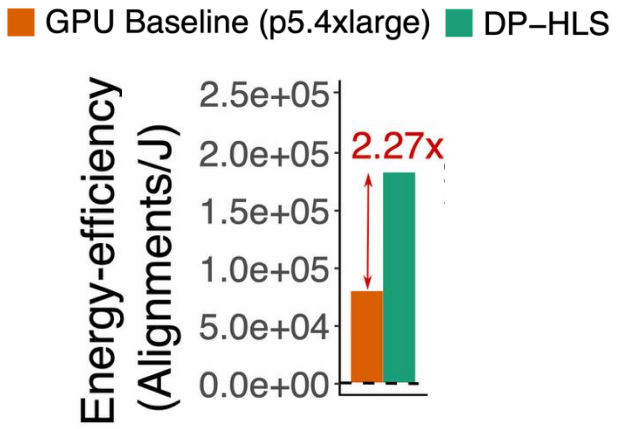
- Showcases **versatility** to support **all possible types of 2D DP kernels on FPGA**
- Gives **competitive performance to RTL designs**
- Achieves up to **41× higher cost-efficiency** than CPU baselines
- Gives up to **17.7× higher cost-efficiency** and **9.5× higher throughput** than GPU baselines



Methodology:
DP-HLS: AWS EC2 F1 instance
GPU Baseline: GASAL2, CUDASW++ on AWS EC2 p3.2xlarge with an NVIDIA Tesla V100 GPU

DP-HLS: Key Contributions & Results

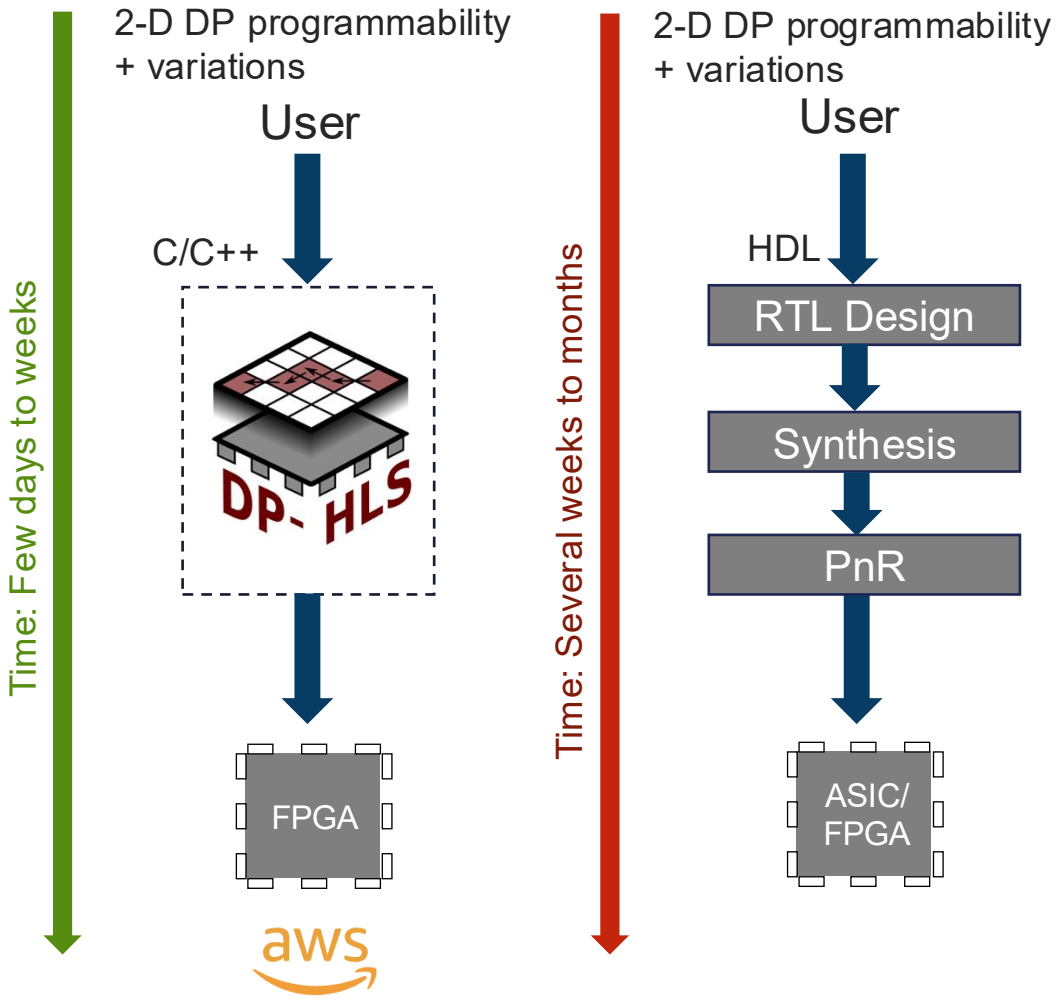
- Showcases **versatility** to support **all possible types of 2D DP kernels on FPGA**
- Gives **competitive performance to RTL designs**
- Achieves up to **41× higher cost-efficiency** than CPU baselines
- Gives up to **17.7× higher cost-efficiency** and **9.5× higher throughput** than GPU baselines
  - **2.27x more energy-efficient** than modern GPU baseline (NVIDIA Hopper GPU with DPX instructions)



Methodology:
DP-HLS: AWS EC2 F1 instance
GPU Baseline: CUDASW++ on AWS EC2 p5.4xlarge with an NVIDIA Hopper H100 GPU

DP-HLS: Key Contributions & Results

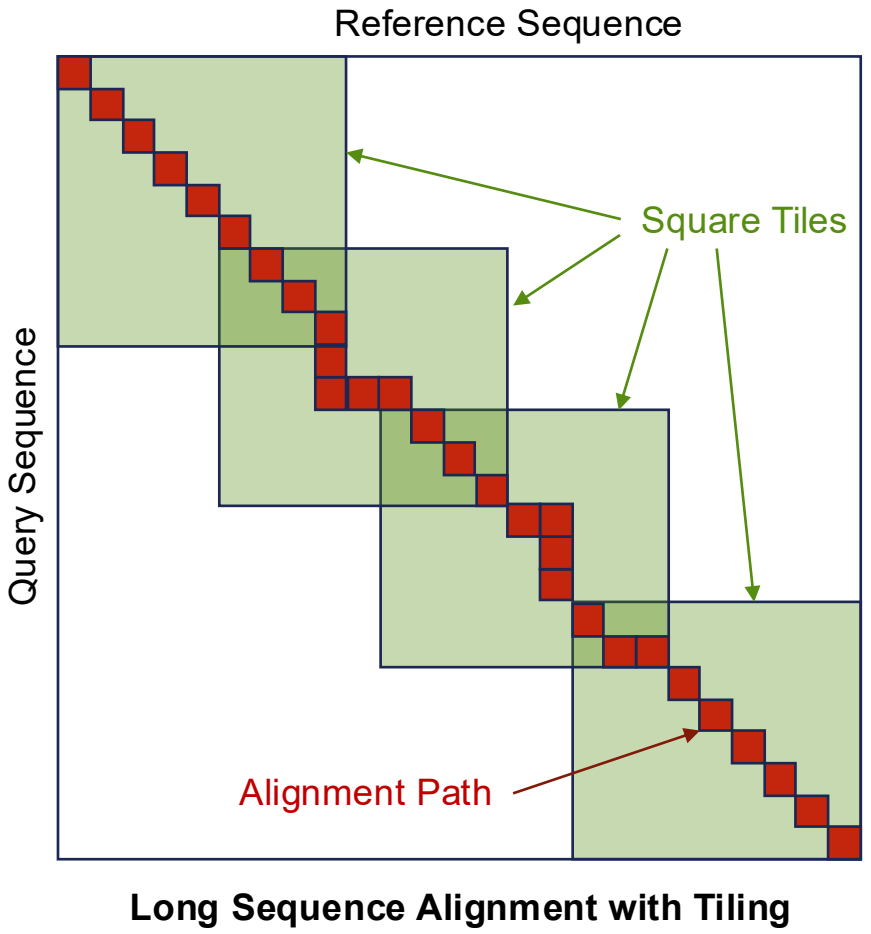
- Kernels are easily deployable on AWS FPGA instances → **Massive improvement in design productivity**



DP-HLS: Key Contributions & Results

- Kernels are easily deployable on AWS FPGA instances → **Massive improvement in design productivity**

- **Efficient for both short and long sequence alignments**



DP-HLS: Key Contributions & Results

- Kernels are easily deployable on AWS FPGA instances → **Massive improvement in design productivity**
- **Efficient for both short and long sequence alignments**
- Artifacts are **open-sourced, functional and reproducible**

https://github.com/TurakhiaLab/DP-HLS

HPCA Artifact Evaluation

Outline

- Dynamic Programming (DP) in Bioinformatics
- DP Algorithmic Variations and Hardware Accelerators
- DP-HLS Framework
- Key Contributions and Results
- **Conclusion and Future Applications**

Key Takeaways

- **DP-HLS** leverages the key insight that a broad **class of 2-D DP algorithm** can be mapped to **same hardware primitive**

Key Takeaways

- **DP-HLS** leverages the key insight that a broad **class of 2-D DP algorithm** can be mapped to **same hardware primitive**

- Introduces a **new layer of abstraction in the HLS flow**
  - Front-end provides programmability and flexibility
  - Back-end provides efficiency

Key Takeaways

- **DP-HLS** leverages the key insight that a broad **class of 2-D DP algorithm** can be mapped to **same hardware primitive**
- Introduces a **new layer of abstraction in the HLS flow**
- Provides a **software-like programming model**
  - Enabling rapid and flexible implementation of any new 2-D DP kernel
  - Without requiring hardware design expertise

Key Takeaways

- **DP-HLS** leverages the key insight that a broad **class of 2-D DP algorithm** can be mapped to **same hardware primitive**
- Introduces a **new layer of abstraction in the HLS flow**
- Provides a **software-like programming model**
- Leverages a **linear systolic array-based hardware abstraction**
  - Support a broad class of 2-D DP algorithms

Key Takeaways

- **DP-HLS** leverages the key insight that a broad **class of 2-D DP algorithm** can be mapped to **same hardware primitive**
- Introduces a **new layer of abstraction in the HLS flow**
- Provides a **software-like programming model**
- Leverages a **linear systolic array-based hardware abstraction**
- Achieves **competitive throughput and RTL-comparable performance** on AWS FPGAs

Future Applications

- DP-HLS insights can be **extended to applications beyond bioinformatics, sharing same hardware primitive**

Future Applications

- DP-HLS insights can be **extended to applications beyond bioinformatics, , sharing similar hardware primitive**

- **AI/ML workloads** map naturally to 2-D systolic arrays
  - Convolution
  - Matrix multiplication

- **2-D Systolic array can have variations**
  - Different data types
  - Different types of quantization
  - Different types of sparsity
  - Variations in non-linear functions

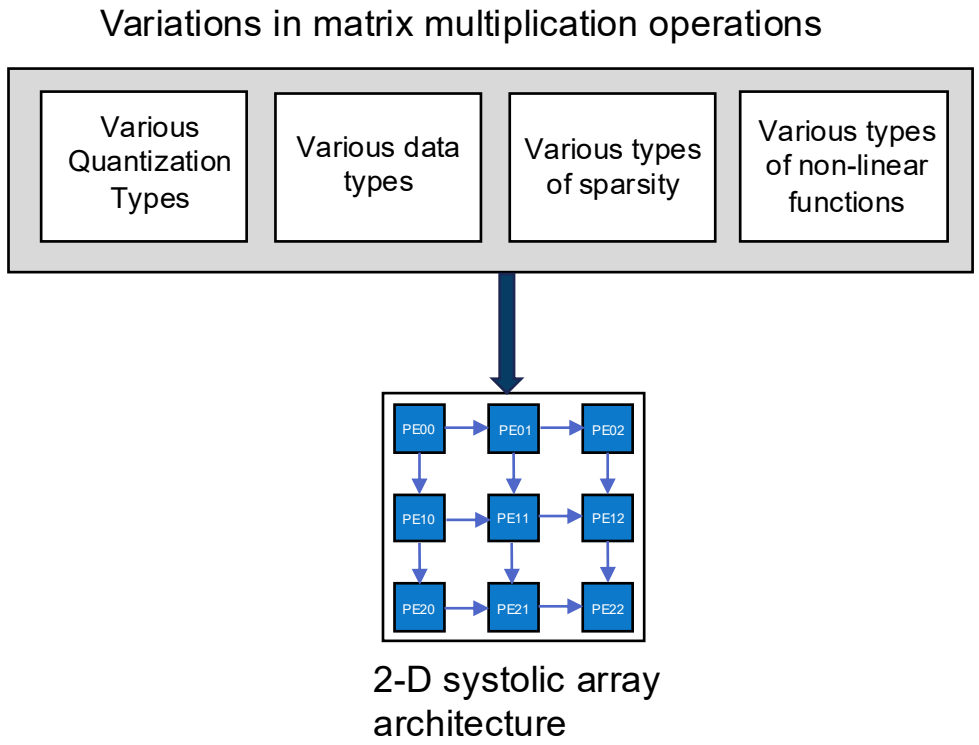
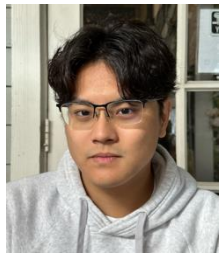


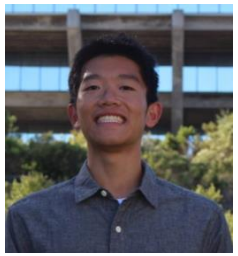
Image source: AI generated

Acknowledgements

Co-authors:


Anshu Gupta


Yingqi Cao


Jason Liang


Yatish Turakhia,
Assistant Professor

Collaborators:

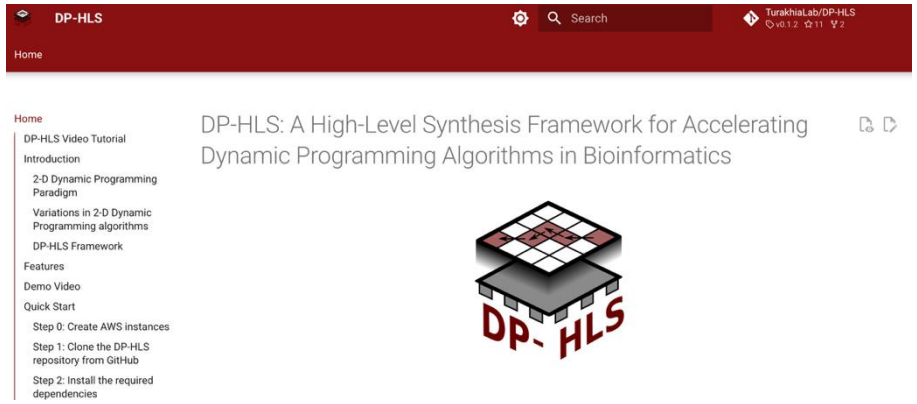
- AMD-omics group

AMD

Affiliations:

UC San Diego
JACOBS SCHOOL OF ENGINEERING

Turakhia Lab



DP-HLS Wiki: https://turakhia.ucsd.edu/DP-HLS/

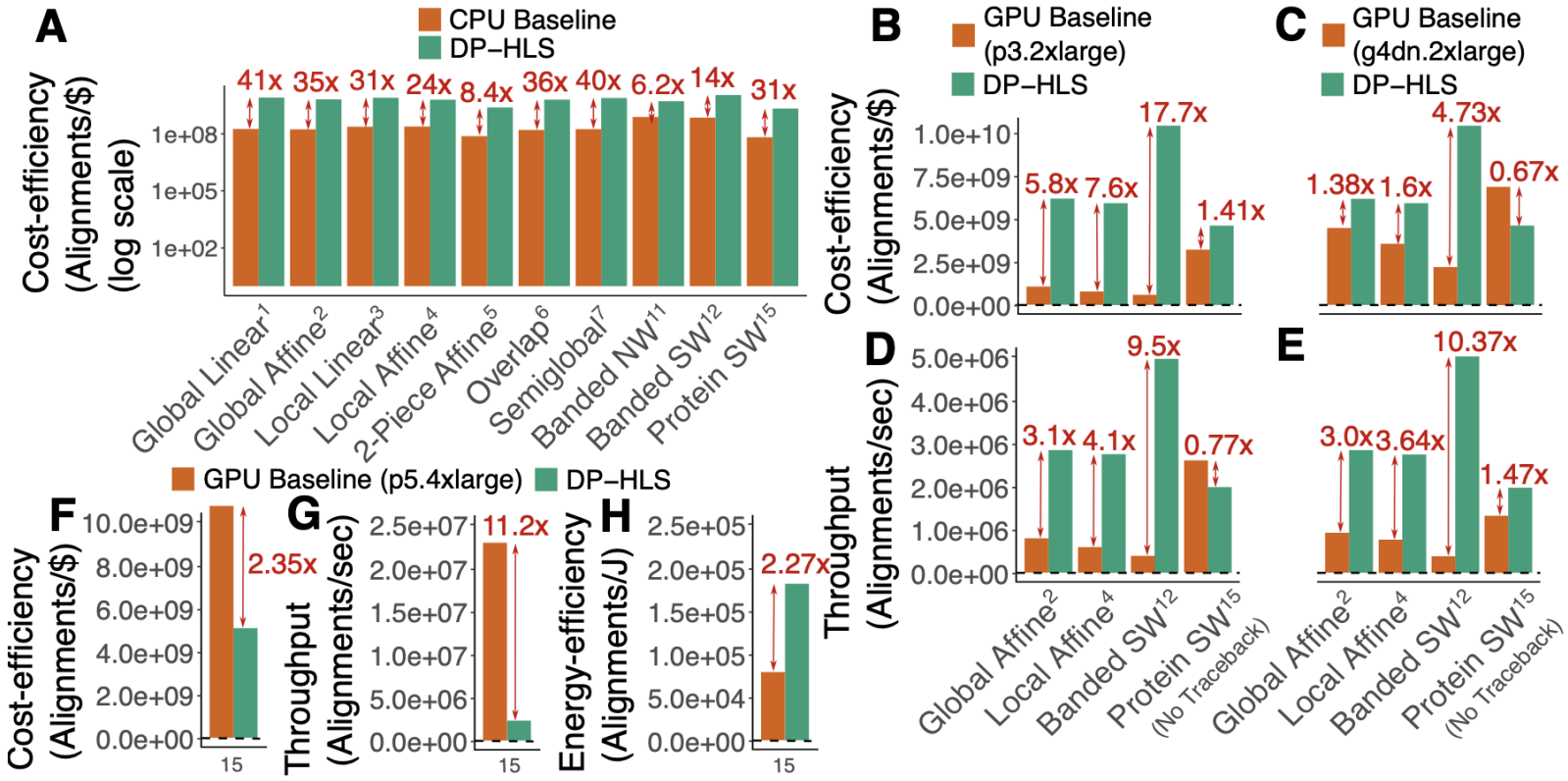
Deployable on

aws

Artifacts Available,
Functional, Reproducible

GitHub Repo

Extra Slides



Extra Slides

Kernel No.	Resource utilization for 32PE LUT	FF	BRAM	DSP	Optimal (N_{PE}, N_B, N_K)	Max Freq (MHz)	Alignments /sec	Total Power (watt)	Alignments /J
#1	0.72%	0.42%	1.78%	0.029%	64,16,4	250.0	3.51e6	17.6	2.0e5
#2	1.30%	0.517%	1.78%	0.029%	32,16,4	250.0	2.85e6	14	2.03e5
#3	0.95%	0.63%	1.67%	0.014%	32,16,5	250.0	3.43e6	16.4	2.09e5
#4	1.60%	0.75%	1.67%	0.014%	32,16,4	250.0	2.71e6	19.2	1.41e5
#5	2.03%	0.65%	2.67%	0.029%	32,8,5	150.0	1.06e6	10.73	9.87e4
#6	0.98%	0.66%	1.67%	0.014%	32,16,4	250.0	2.73e6	14.9	1.83e5
#7	1.17%	0.67%	0.83%	0.014%	32,16,4	250.0	3.34e6	17.54	1.90e5
#8	3.66%	2.56%	2.56%	28.11%	16,1,5	166.7	3.70e4	9.18	4.02e3
#9	1.62%	1.55%	1.88%	2.84%	64,4,3	200.0	2.31e5	15.47	1.49e4
#10	3.78%	1.69%	1.67%	0.014%	16,4,7	125.0	4.90e5	8.15	6.01e4
#11	1.02%	0.40%	0.94%	0.029%	64,8,7	166.7	2.25e6	11.6	1.93e5
#12	1.44%	0.70%	0.57%	0.014%	16,16,7	200.0	4.77e6	14.4	3.31e5
#13	2.25%	0.69%	1.83%	0.029%	16,8,7	125.0	1.24e6	7.6	1.63e5
#14	1.22%	0.76%	0.57%	0.014%	32,16,5	250.0	5.16e6	17.67	2.92e5
#15	1.47%	0.95%	2.56%	0.014%	32,8,5	200.0	9.33e5	11.96	7.80e4