

ASAP: Accelerated Short Read Alignment on Programmable Hardware

Subho Sankar Banerjee, Mohamed El Hady, Jong Bin Lim, Daniel Chen, Zbigniew T. Kalbarczyk, Deming Chen, Ravishankar K. Iyer
University of Illinois at Urbana-Champaign

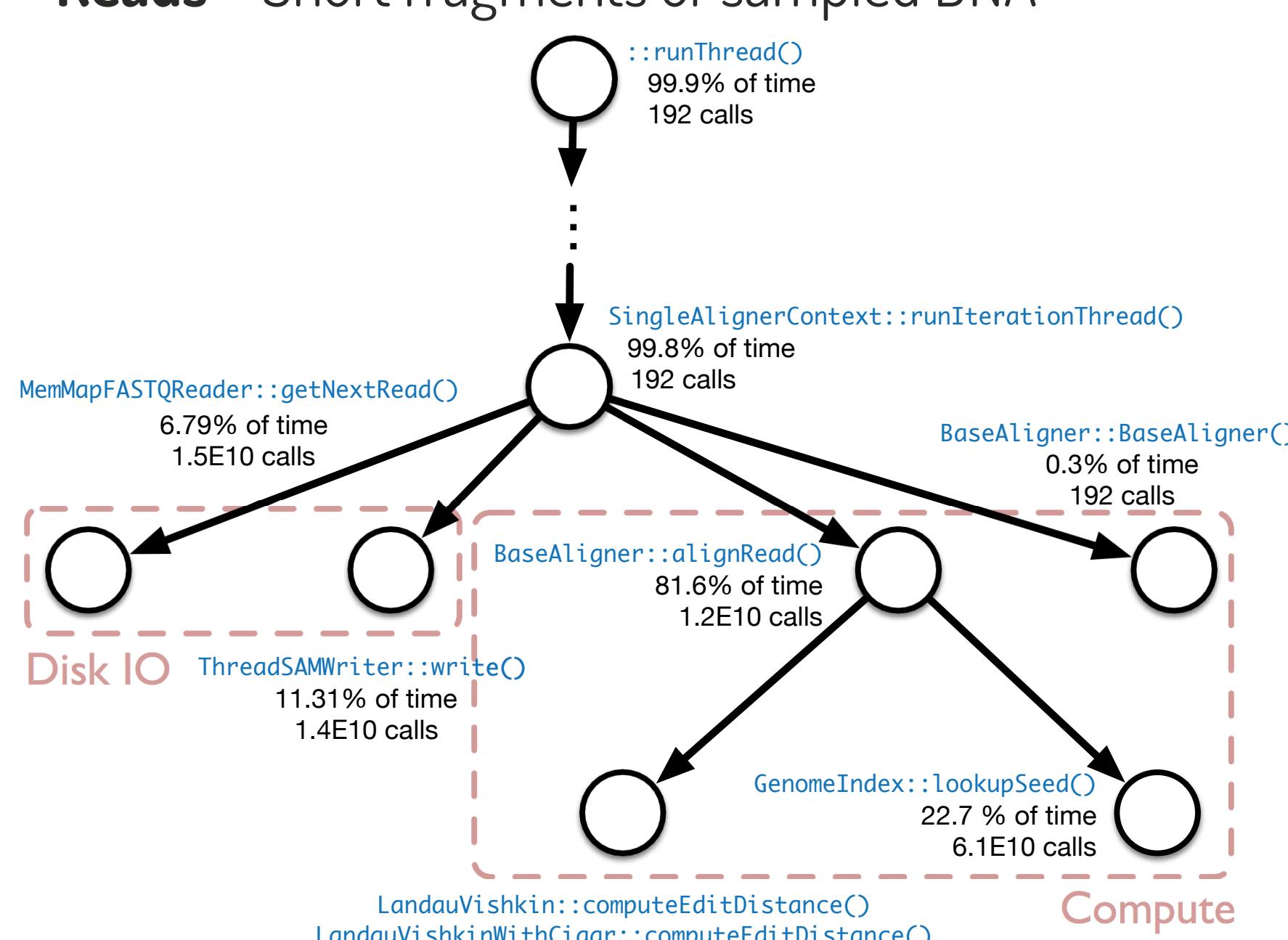


Abstract

- Rapidly generated short-nucleotide fragments and huge genomic-data easily overwhelm today's HPC infrastructure
- Levenshtein distance (edit-distance) is a crucial step in processing in short-read data for genomic analysis
- By uniquely utilizing intrinsic delay of circuit as a proxy for computation, ASAP achieved **200x faster** operation than equivalent C implementation on CPU.
- Integrating ASAP on Altera Stratix V FPGA with IBM POWER 8 via CAPI interface, our heterogeneous system achieved **2.2x faster** than an end-to-end alignment tool for 120-150 bp short-read sequences.

Motivation

- Short-read alignment** – Process of mapping the sequenced **reads** to their most likely point of origin in the **genome**
- Reads** – Short fragments of sampled DNA



- Levenshtein distance (LD) computation** – responsible for **50-70%** of short-read alignment runtime, which in turn

Key Ideas:

- In most resequencing experiments, **most nucleotides match the reference**.
- Use this observation along with circuit-delay based computation (RaceLogic, ISCA15), to compute LD.

Smith-Waterman & Needleman-Wunch Algorithm

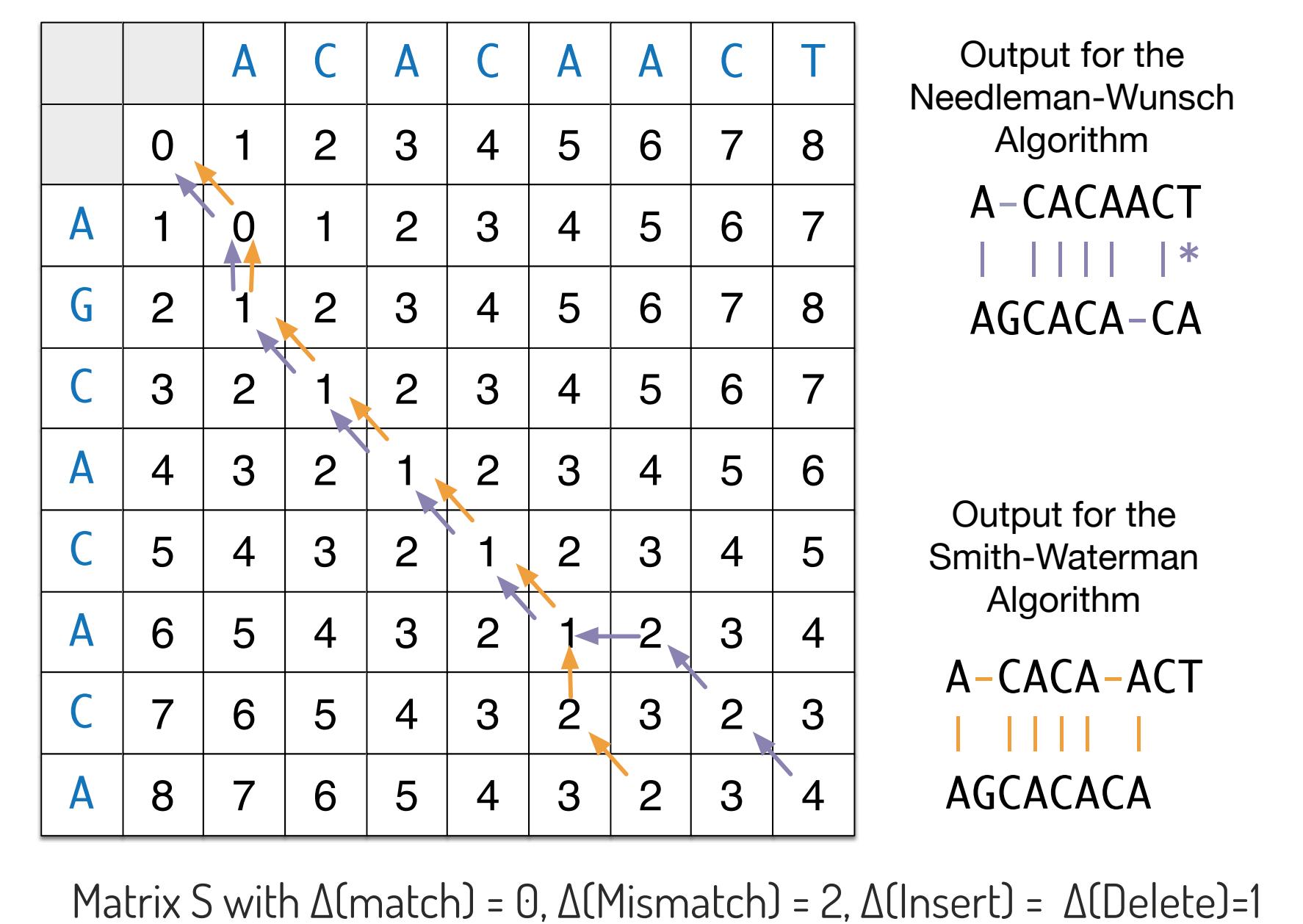
Dynamic Programming based algorithm

- Calculates the alignment score (Levenshtein distance) between **read(q)** and **reference genome(r)**

Key Ideas of Algorithm

- Matrix S of size $|q| \times |r|$
- Δ : gap penalty
- (i, j)th Element of S – Minimum edit distances between strings $Q[1 : i]$ and $R[1 : j]$

$$S(i, j) = \min \left\{ \begin{array}{l} S(i - 1, j) + \Delta(-, R_j), \\ S(i, j - 1) + \Delta(Q_i, -), \\ S(i - 1, j - 1) + \Delta(Q_i, R_j) \end{array} \right\}$$

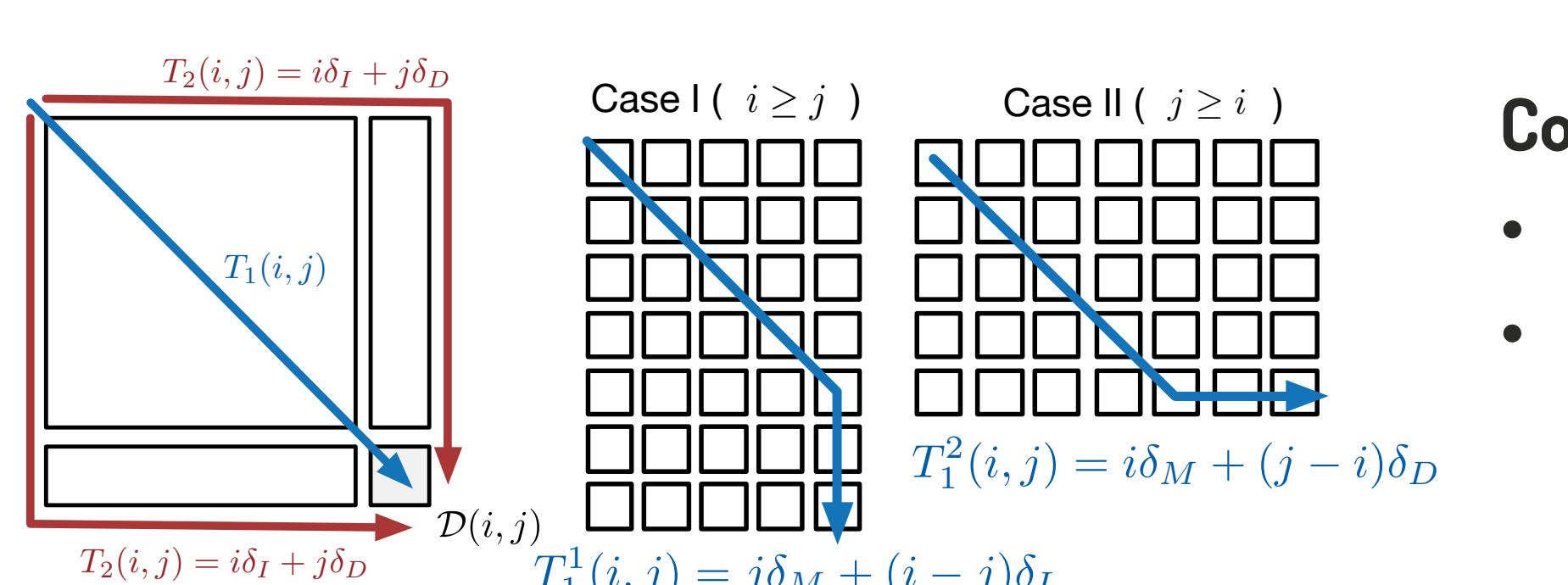


Matrix S with $\Delta(\text{match}) = 0$, $\Delta(\text{Mismatch}) = 2$, $\Delta(\text{Insert}) = \Delta(\text{Delete}) = 1$

FPGA Area Optimization

Motivation:

- Upper-left DE** requires **smaller N_{DE} -bit** counter & registers than **lower right DE** does.
- Values of T_1 , T_2 , and T_3 are calculated with $\delta(\text{Match}) \leq \delta(\text{Mismatch})$, $\delta(\text{Insert})$, $\delta(\text{Delete})$



$$T_1(i, j) = \begin{cases} j\delta_M + (i - j)\delta_I & \text{if } i \geq j \\ i\delta_M + (j - i)\delta_D & \text{otherwise} \end{cases}$$

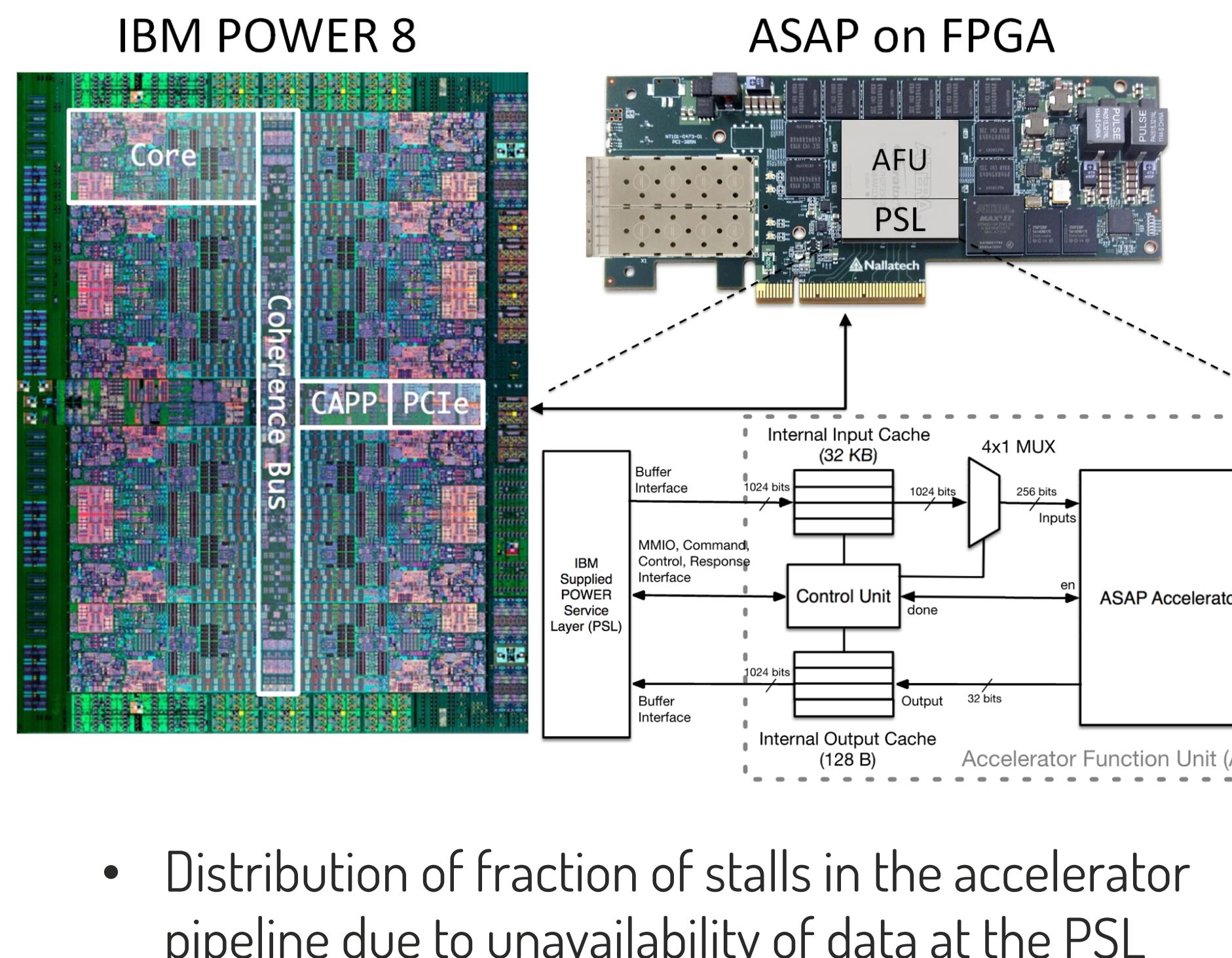
$$T_2(i, j) = i\delta_M + j\delta_D$$

$$N_{DE}(i, j) = \begin{cases} \lceil \log_2 |j(\delta_M - \delta_I - \delta_D)| \rceil & \text{if } i \geq j \\ \lceil \log_2 |i(\delta_M - \delta_I - \delta_D)| \rceil & \text{otherwise} \end{cases}$$

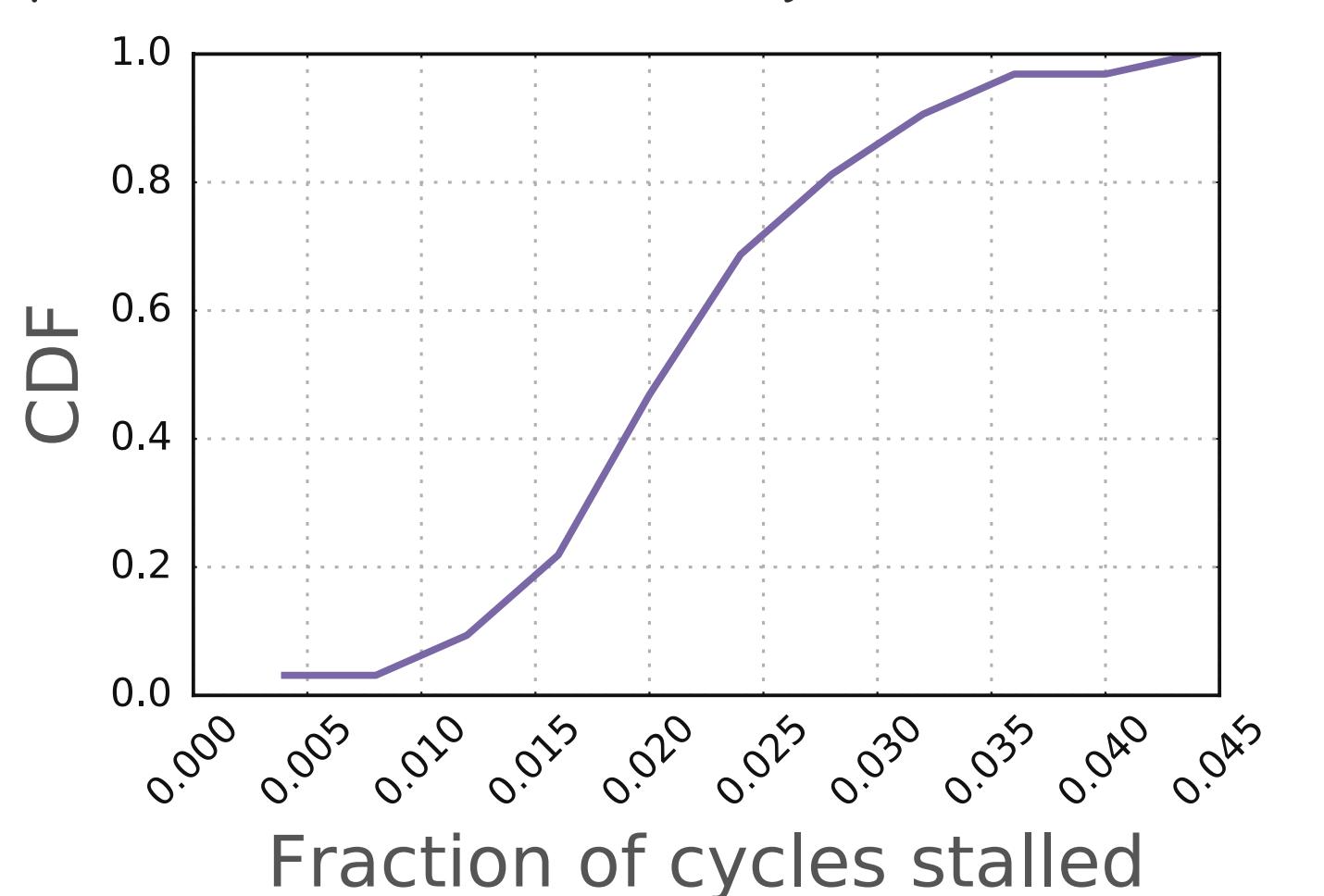
Comparison – 150(q) & 600(r) bp

- Area reduction by **4.9x**
- From **2,880,000** down to **587,000 FF's**

POWER 8 - FPGA Interface via CAPI

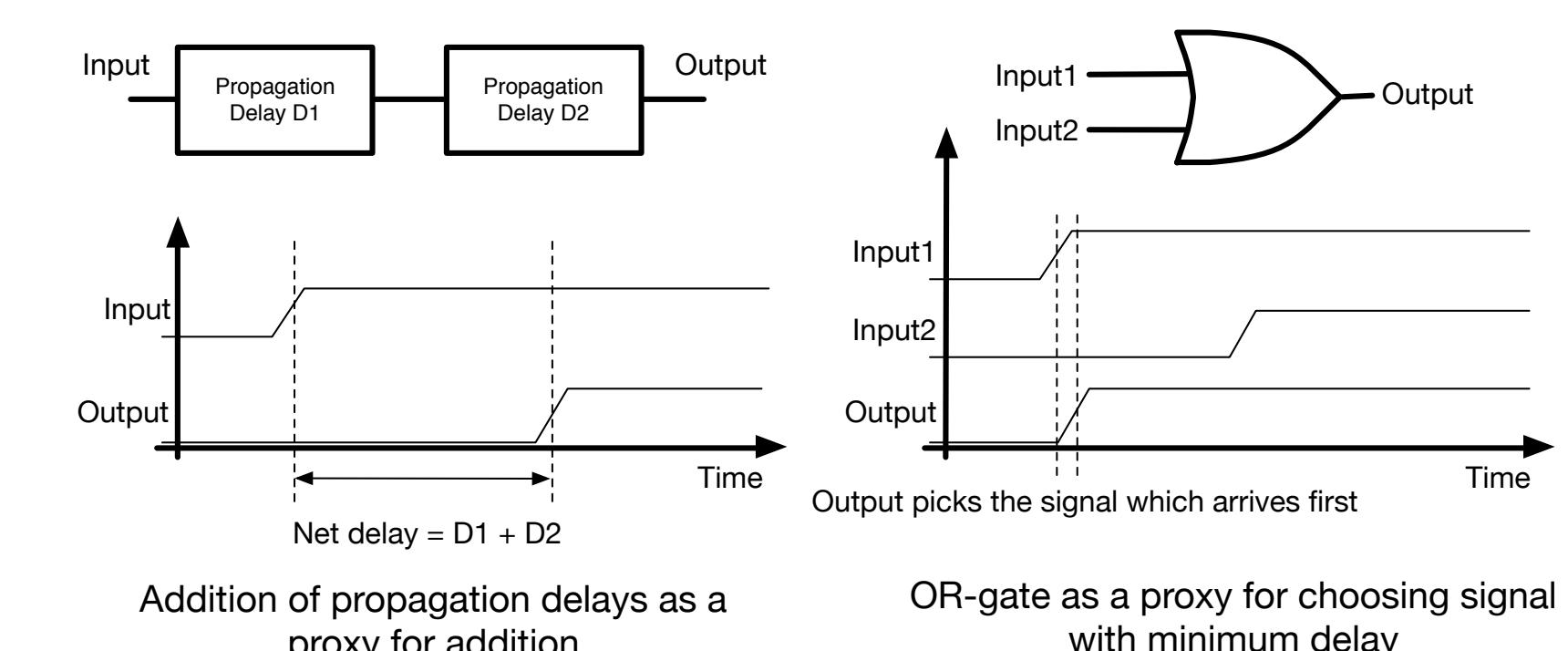


- Distribution of fraction of stalls in the accelerator pipeline due to unavailability of data at the PSL



ASAP: Using Delays to Compute Faster

Computing with Circuit Delays (RaceLogic, ISCA15)

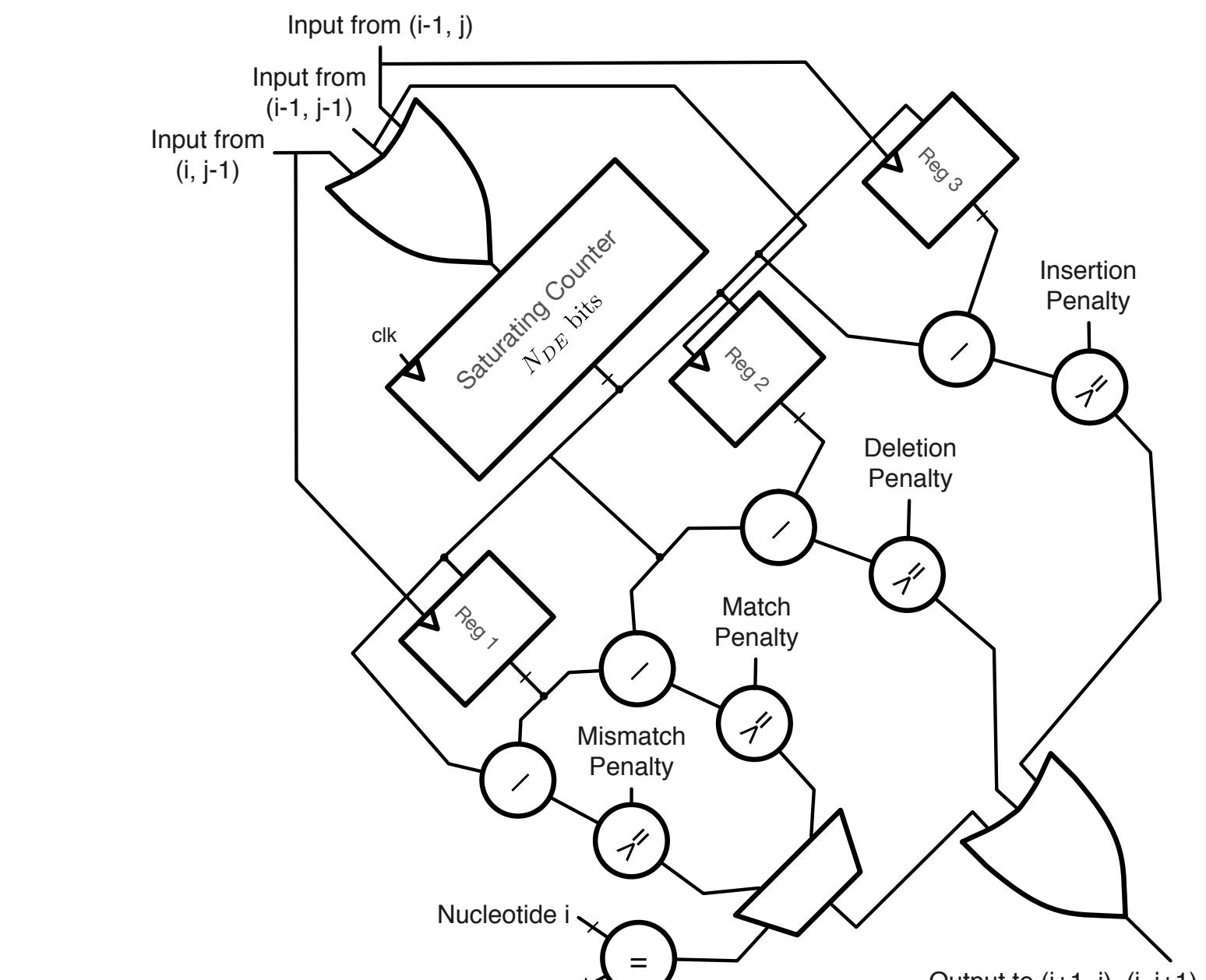


Novelty: Approximate LD using Circuit Delays

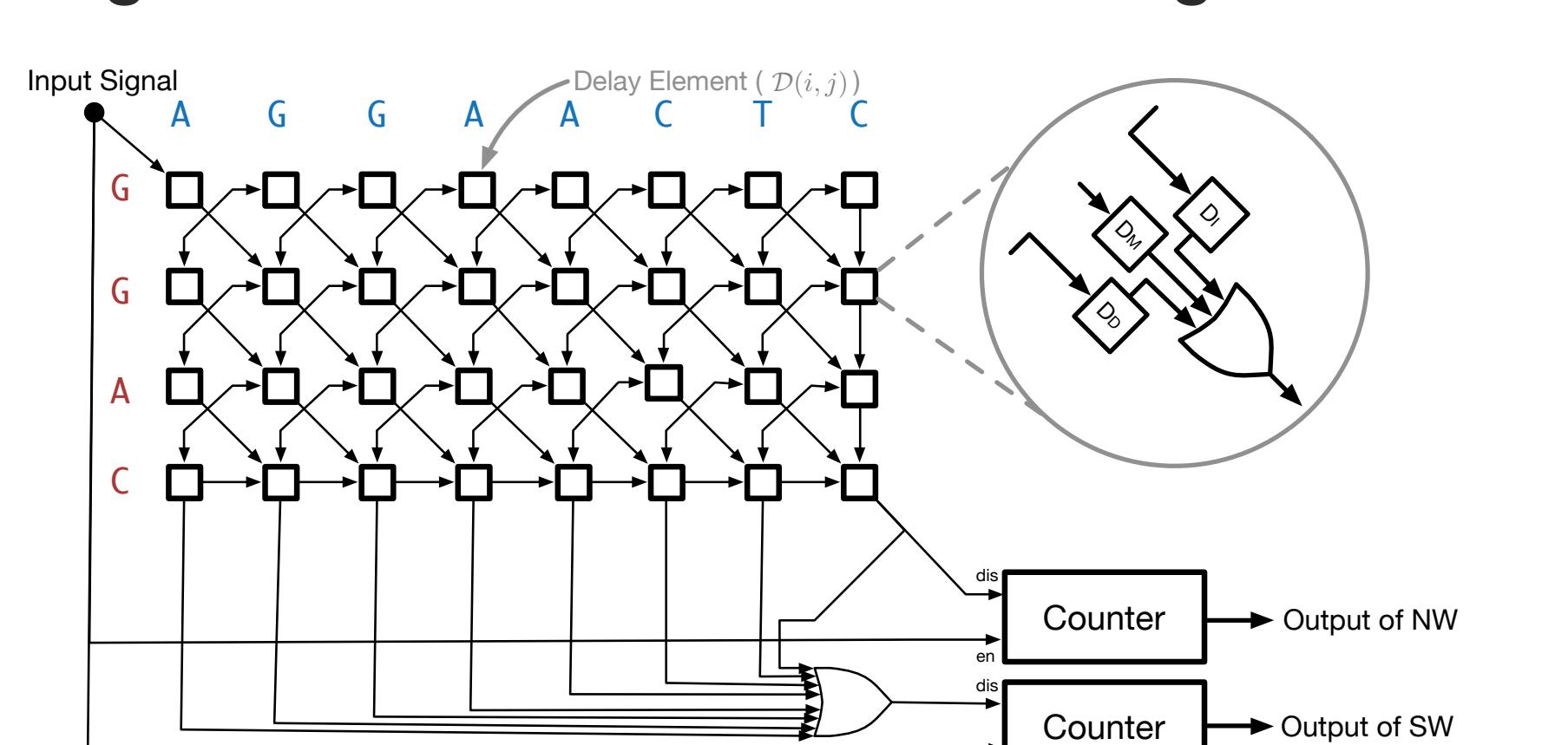
- Match happens most often \Rightarrow Set delay to 0
- Approximation maintains total-ordering \Rightarrow answers are still correct

Components of ASAP Delay Element (DE)

- 3-input signals connected to preceding DE
- 2-input signals to compare strings
- 3-input signals representing Δ

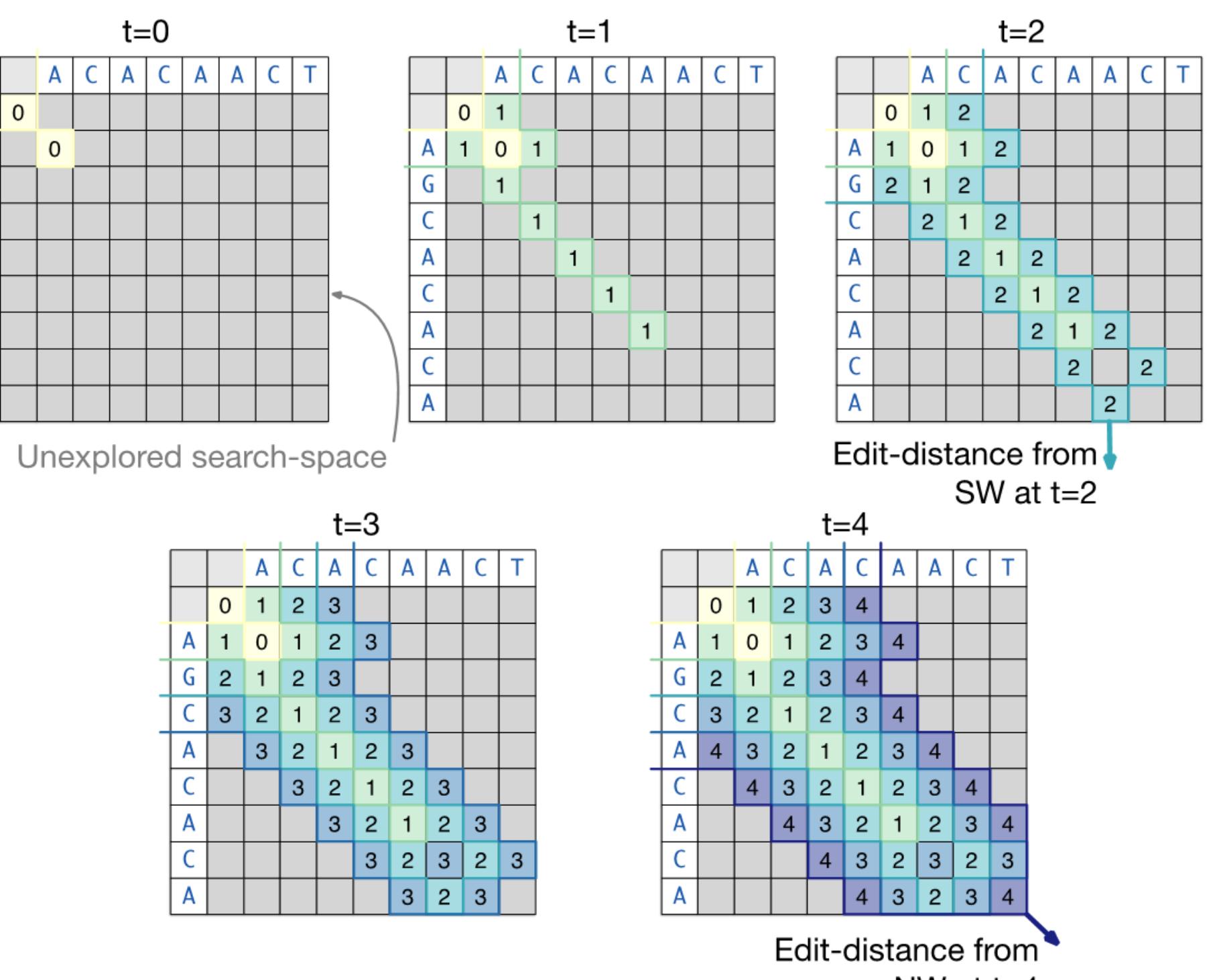


High-level Structure of the ASAP Design



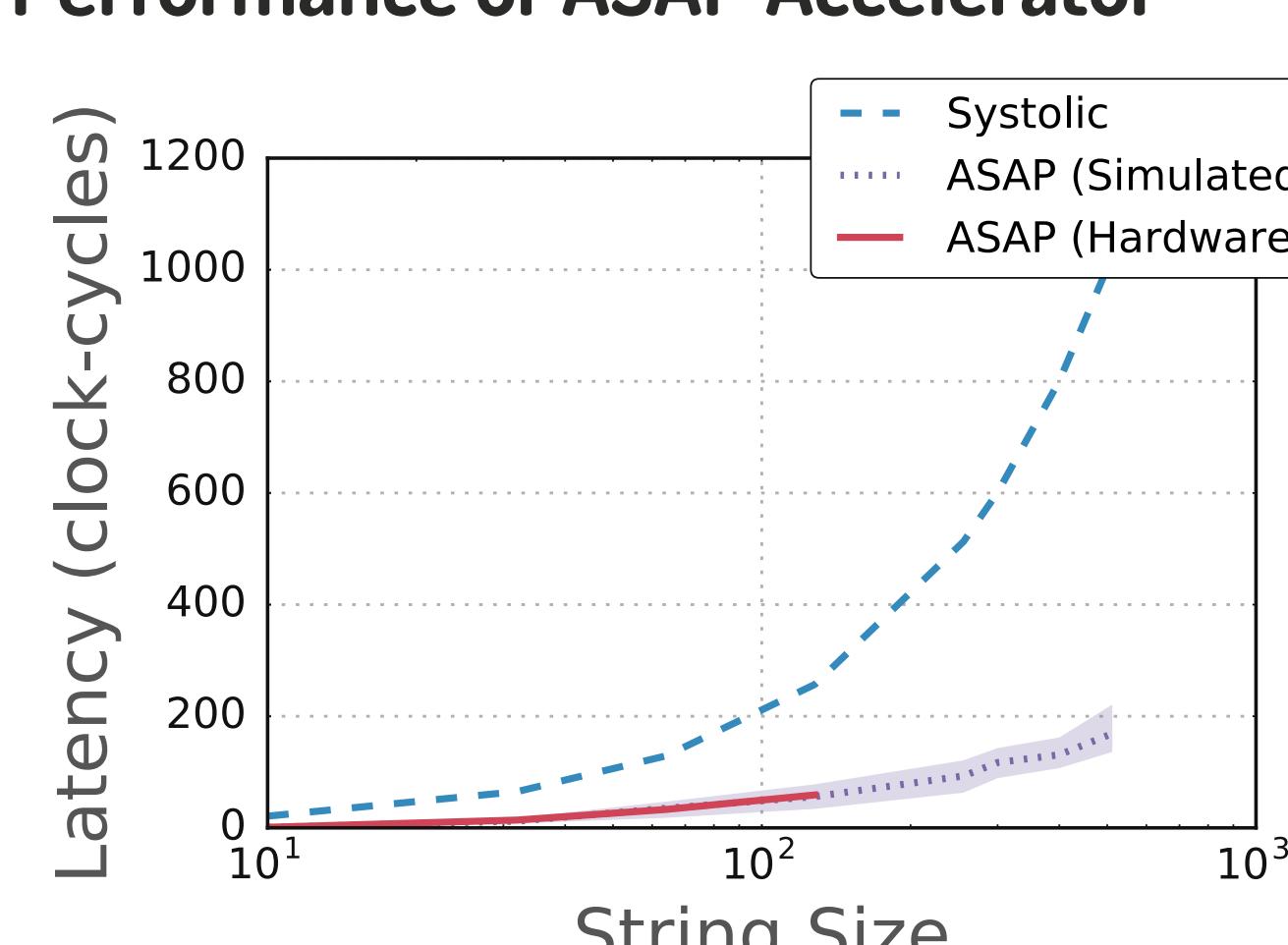
Computing LD Variants

- Smith-Waterman Algorithm:** 2 cycles
- Needleman-Wunsch Algorithm:** 4 cycles
- Landau-Vishkin Algorithm:** Reset circuit after max tolerable delay
- Values in Matrix – Clock cycle of when DE was triggered

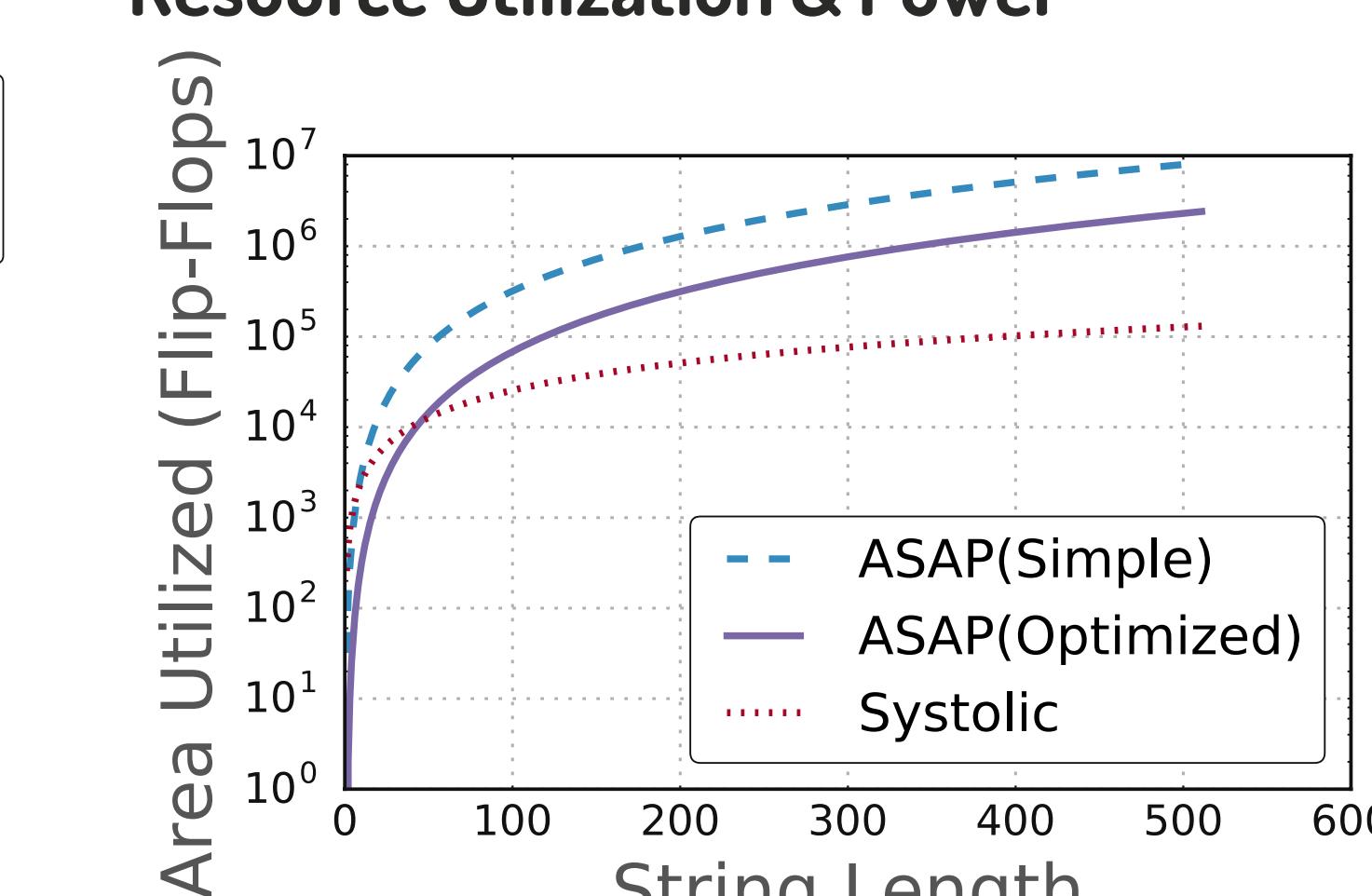


Experimental Results

Performance of ASAP Accelerator



Resource Utilization & Power

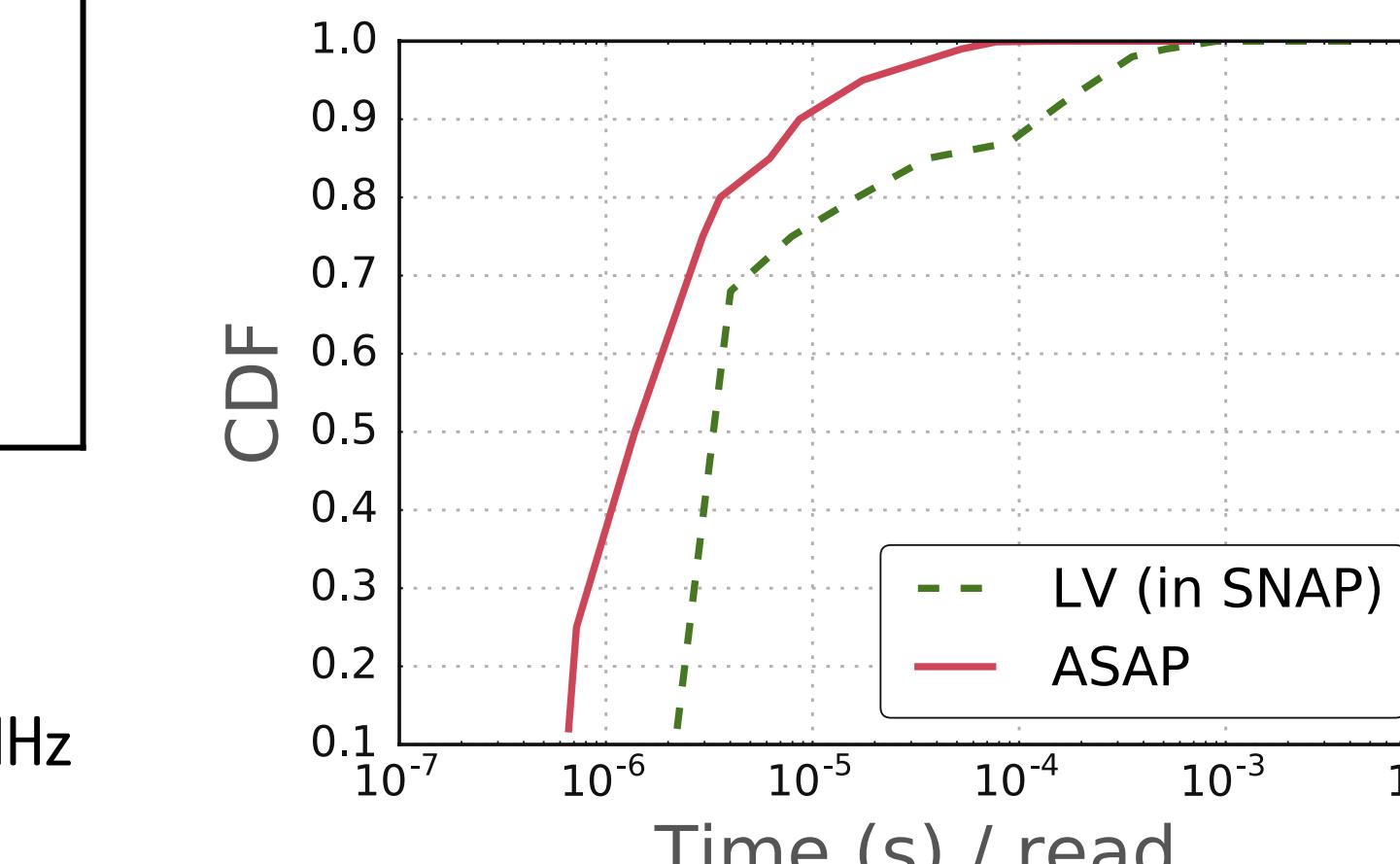


- FPGA FF utilization VS Length of strings

	Read Length		
	32	64	128
ALM Utilization (%)	26	38	76
Total Power (mW)	5613	9642	19912

End-to-End Comparison

- Default LV SNAP Aligner VS ASAP-Integrated SNAP Aligner \Rightarrow **2.2x Speedup**



Conclusion

- Intrinsic circuit delay was utilized to replace arithmetic addition & min-operation
- With this novel computation method, ASAP rapidly computes Levenshtein Distance for short-read alignment.
- FPGA-implemented ASAP is compatible with CAPI interface, allowing more efficient high-throughput genomic data computation
- ASAP can be further applied to any problems where a total ordering of LDs need to be computed

Acknowledgements

We would like to thank IBM and NSF for the support of this project. This material is based upon work supported in part by the National Science Foundation under Grant No. CNS 13-37732.