Reconfigurable Computing

Customisation

"The only man who behaves sensibly is my tailor; he takes my measurements anew every time he sees me, while all the rest go on with their old measurements and expect me to fit them"

– George Bernard Shaw

Philip Leong (philip.leong@sydney.edu.au)
School of Electrical and Information Engineering

http://phwl.org/talks

THE UNIVERSITY OF SYDNEY

Permission to use figures have been gained where possible. Please contact me if you believe anything within infringes on copyright.

Overview

- The more you know about a problem, the better you can customise a solution
- It is hard to find general prinicples to guide opportunities for customisation. We will use 2 examples
  - Distributed arithmetic (where we rearrange equations for doing a dot product)
  - DNA sequence matching (where we fit a problem requiring 4 slices into 3)

Distributed Arithmetic

THE UNIVERSITY OF SYDNEY

Idea: Lookup Tables

- Calculate complex function f(x)
- If number of bits of x is small, just make a ROM-based lookup table
  - F[x] where we precompute for all possible x

Distributed Arithmetic – Reorganising Equations

› Consider a dot product $\langle \mathbf{w}, \mathbf{x} \rangle = \sum_{k=0}^{K-1} w[k]x[k]$

where the **w** are fixed coefficients and x[k] are 2's complement fractions

› Recall if x[k] = {$b_{N-1}[k]$, …, $b_0[k]$}

$$x[k] = 2^{-F}(-b_{N-1}[k]2^{N-1} + \sum_{n=0}^{N-2} b_n[k]2^n),$$

then

$$\langle \mathbf{w}, \mathbf{x} \rangle = \sum_{k=0}^{K-1} w[k](-b_{N-1}[k]2^{N-1-F} + \sum_{n=0}^{N-2} b_n[k]2^{n-F})$$

rearranging

$$\langle \mathbf{w}, \mathbf{x} \rangle = \sum_{n=0}^{N-2} \sum_{k=0}^{K-1} w[k]b_n[k]2^{n-F} - \sum_{k=0}^{K-1} w[k]b_{N-1}[k]2^{N-1-F}$$

Distributed Arithmetic – Lookup Table

$$\langle \mathbf{w}, \mathbf{x} \rangle = \sum_{n=0}^{N-2} \sum_{k=0}^{K-1} w[k] b_n[k] 2^{n-F} - \sum_{k=0}^{K-1} w[k] b_{N-1}[k] 2^{N-1-F} \quad (1)$$

Let's assume F=N-1 (this means our implementation will only shift in 1 direction)

Also remember $b_n[k]$ is binary and w[k] are constants so let's store

$$m[a_n] = \sum_{k=0}^{K-1} w[k] b_n[k]$$

where $a_n = \{b_n[K-1], \ldots, b_n[0]\}$

Then (1) becomes

$$\begin{aligned} \langle \mathbf{w}, \mathbf{x} \rangle &= \sum_{n=0}^{N-2} m[a_n] 2^{n-N+1} - m[a_{N-1}] 2^0 \\ &= m[a_0] 2^{1-N} + \ldots + m[a_{N-2}] 2^{-1} - m[a_{N-1}] 2^0 \end{aligned}$$

Computing Distributed Arithmetic

› <w,x> can be computed using the recurrence

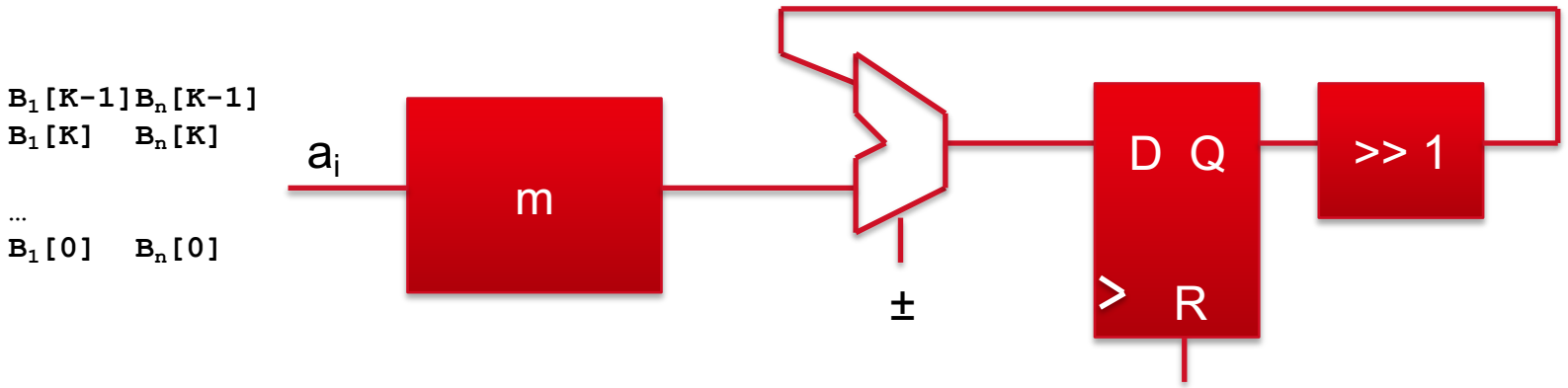
$$
\begin{aligned}
s_0 &= m[a_0] \\
s_i &= s_{i-1}2^{-1} + m[a_i]\ (i = 1, 2, \ldots, N-2) \\
s_i &= s_{i-1}2^{-1} - m[a_i]\ (i = N-1)
\end{aligned}
$$

› Can you draw a datapath which can implement this?

Computing Distributed Arithmetic

› <w,x> can be computed using the recurrence

$$
\begin{aligned}
s_0 &= m[a_0] \\
s_i &= s_{i-1}2^{-1} + m[a_i] \; (i = 1, 2, \ldots, N-2) \\
s_i &= s_{i-1}2^{-1} - m[a_i] \; (i = N-1)
\end{aligned}
$$



Distributed Arithmetic Example

Suppose $\mathbf{w} = [w[1], w[0]] = [0.375, -0.5]$ and $\mathbf{x} = [-0.75, 0.125]$, $N = 6$ and $K = 2$ is the length of the vectors. In binary, $\mathbf{x} = [1.0100, 0.0010]$. Thus

$$\begin{aligned} \langle \mathbf{w}, \mathbf{x} \rangle &= w[1] \times x[1] + w[0] \times x[0] && (26) \\ &= 0.375 \times -0.75 + -0.5 \times 0.125 && (27) \\ &= 0.375 \times (-2^0 + 2^{-2}) + -0.5 \times (2^{-3}) && (28) \end{aligned}$$

Distributed arithmetic rearranges this computation to be

$$\begin{aligned} \langle \mathbf{w}, \mathbf{x} \rangle &= 2^{-3} \times -0.5 + \\ &= 2^{-2} \times 0.375 - \\ &= 2^0 \times 0.375. \end{aligned}$$

a_1	a_0	$m[a_1, a_0]$	Numeric Value
0	0	0	0
0	1	$w[0]$	-0.5
1	0	$w[1]$	0.375
1	1	$w[0] + w[1]$	-0.125

a	$m[a]$	$\pm$	s_i
00	0	+	$s_0 = 0$
00	0	+	$s_1 = 0$
01	-0.5	+	$s_2 = -0.5$
10	0.375	+	$s_3 = 0.125$
00	0	+	$s_4 = 0.0625$
10	0.375	-	$s_5 = -0.34375$

Summary

- Dot products can be computed in time independent of K if it is not too big
- Precomputation often a useful customisation trick
  - Always think about whether lookup tables can be applied if the number of input bits is small
  - c.f. CORDIC which also uses a small lookup table and rearrangement of computation

THE UNIVERSITY OF SYDNEY

References

- Stanley A. White, “Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review”, IEEE ASSP Magazine, July 1989

DNA Sequence Matching

Overview

- Background
- Smith Waterman Algorithm
- Improved cell
- Results
- Conclusion

Background

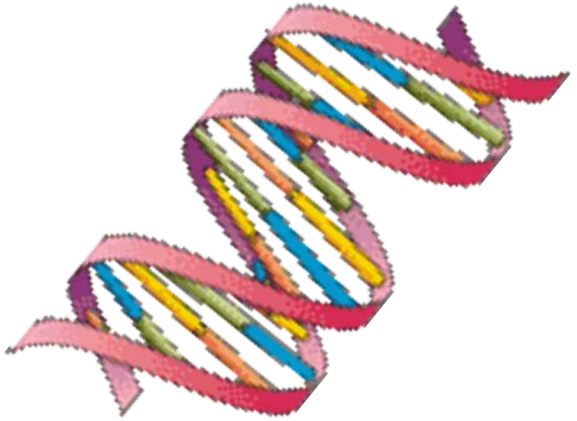
- DNA chain – long, unbranched polymer
- Basic unit of DNA – Nucleotides

- Nucleotides –

  4 different bases –

  adenine (A), cytosine (C),

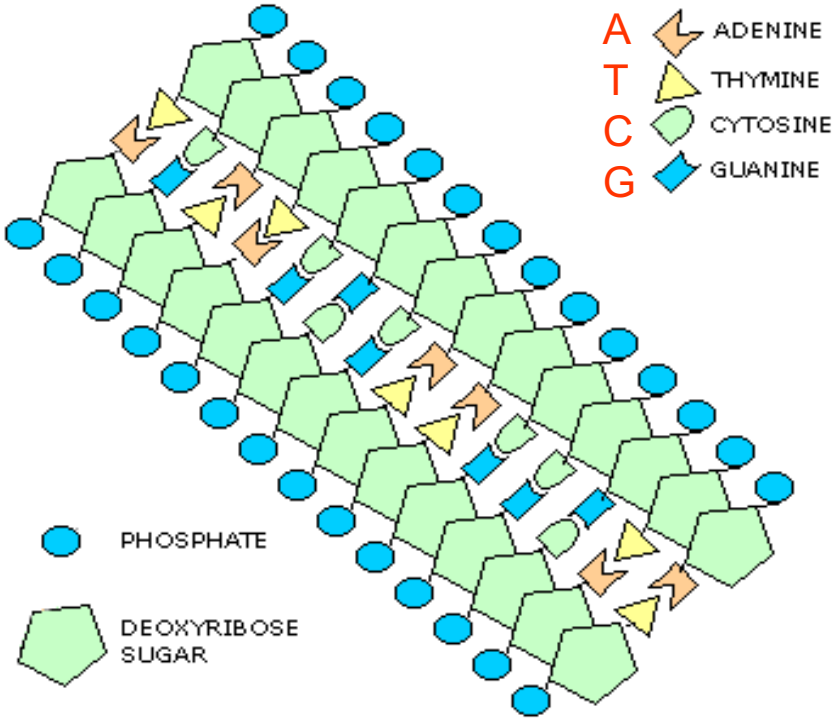
  guanine (G) and thymine (T)
- Long sequence of A, C, G or T nucleotides



DNA Sequence

› The sequence - one of the strands of DNA running in a particular direction

GCCGCACGNTCTCATCGGTCCCGGCCGCCATGGCG
GCAGGAATTCGATTCTATAGCANCCAGCAGACCACCT
ACGGCATATGGCAGCACAGGAAATGCTCTCTGCCAGC
ATCTCCCANACCCGCATCCTGCAGACCTGCAGCGTG
CCTCATGCCAACATGGTCAACGGNGCCAACTCACTGC
AAGGAGCTCTGGCTCCACGCCTCTATAAGTTCCCTGA
GCATGGTTTGGGTGGGGGCTCTTGTGCTTTGACCCA
CAGCTTCCCGCCGCTGCCCCAGGCCCTGCTCACGGA
CGAACCCACTCTAGGTGACATCAAGCAGGAGCTGCG
CAGGAAAAGTCGGCCCCTGGAAGAGCCGCCCGATAT
GGACTCACCTCAGATCCGTGAACTGGAGAAGTTTGCT
AATGACTTCAAACTGAGGAGGATCAAA



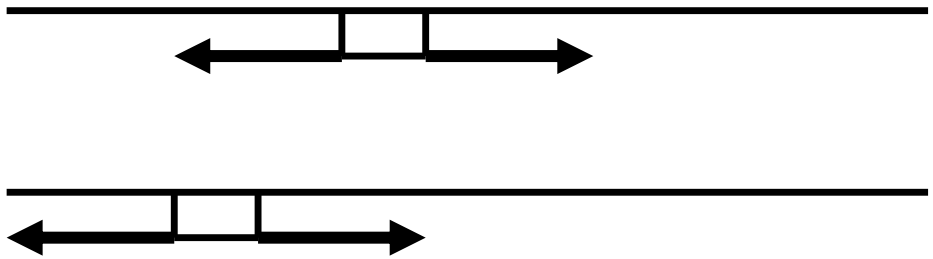
THE UNIVERSITY OF SYDNEY

Applications of Sequence Alignment

- Find edit distance between two strings (Similarity between two sequences)
- Identify the protein encoded by the unknown DNA sequence
- Decode the relation between disease and inheritance
- And others…

Sequence Alignment Algorithms

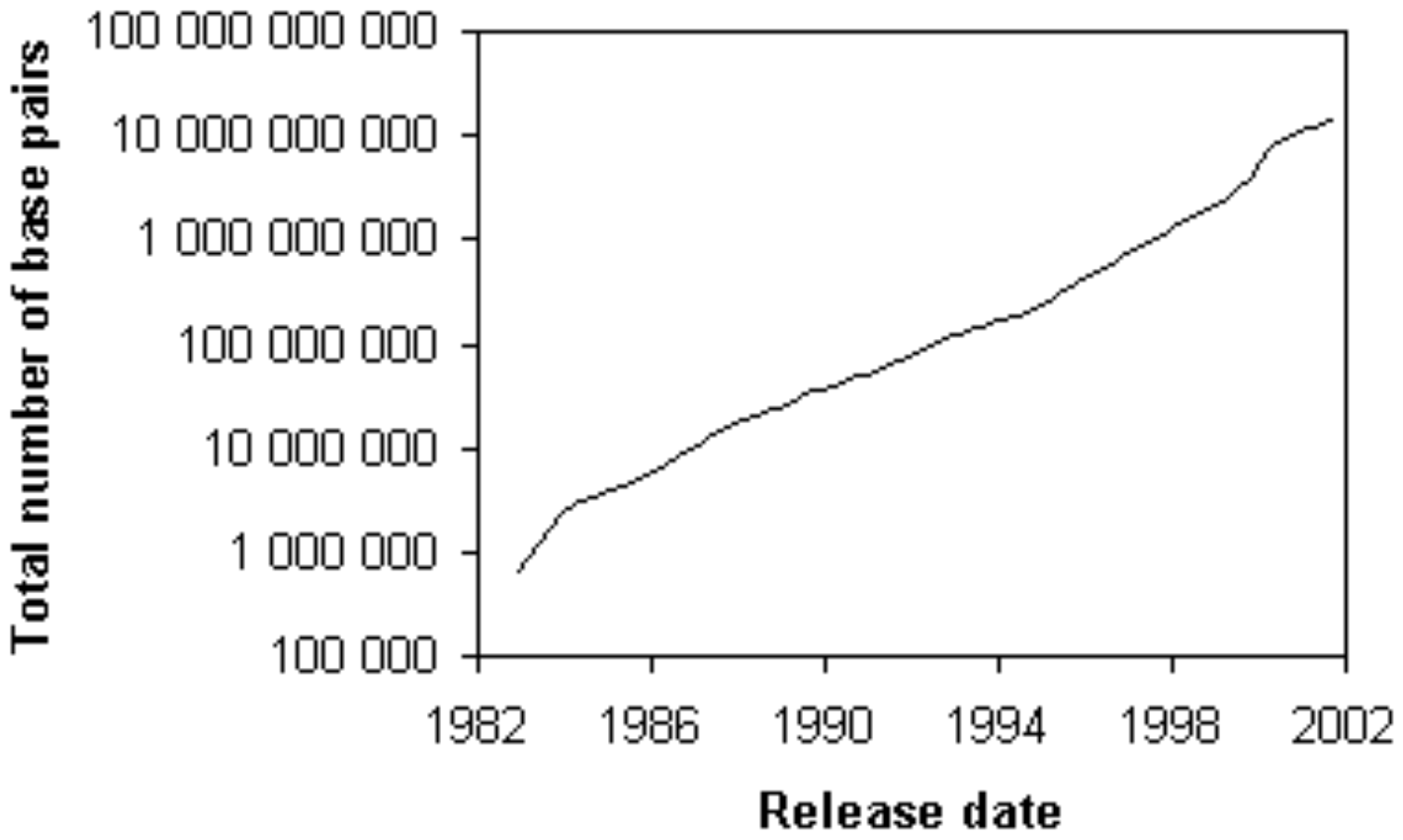
- Most popular: BLAST, FASTA
  - Heuristic method



  - Fast, moderate precision
- Smith-Watermann Algorithm
  - Higher time complexity
  - Approx ½ errors/query of BLAST/FASTA

Motivation

Public databases of DNA sequence of species



The graph shows the growth of GenBank

Previous Work (Smith Waterman Cells)

- Splash 2 system, Hoang - 1993
  - 33 LUT/FF pairs per cell (XCV4010)
- Xilinx, Guccione and Keller - 2002
  - Optimized Smith Waterman core uses runtime reconfiguration to store one of the strings and to implement the insertion, substitution and deletion penalties
  - 6 LUT/FF pairs per cell (XCV1000-6)
- HokieGene, Virginia Tech – 2002
  - Used Xilinx's core on an Osiris board (XCV6000-4)

		T0 A	C	G
	0	1	2	3
S0 A	1	0	1	2
T	2	1	2	3
C	3	2	1	2

$$d = \min \begin{cases} a & \text{if } S_i = T_j \\ a + 2 & \text{if } S_i \neq T_j \\ b + 1 & \\ c + 1 & \end{cases}$$

Alignment Score of "ACG" and "ATC"

Example

		A	C	T	A	T	C	G	T	A
	0	1	2	3	4	5	6	7	8	9
A	1	**0**	1	2	3	4	5	6	7	8
G	2	1	**2**	3	4	5	6	5	6	7
T	3	2	3	**2**	3	4	5	6	5	6
A	4	3	4	3	**2**	**3**	4	5	6	5
C	5	4	3	4	3	4	**3**	4	5	6
G	6	5	4	5	4	5	4	**3**	4	5
T	7	6	5	4	5	4	5	4	**3**	4
G	8	7	6	5	6	5	6	5	4	**5**

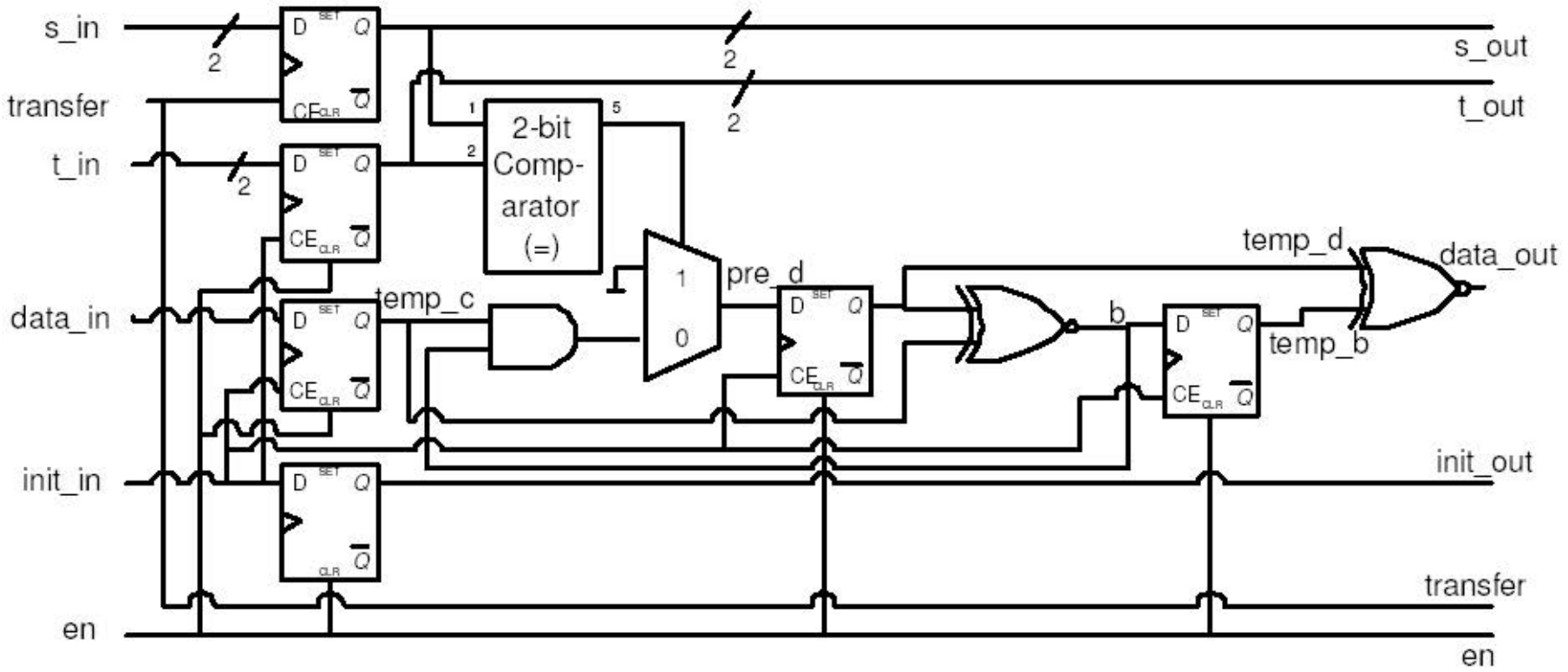
Aligned sequences

Simplified Equation (Lipton and Lopresti)

$$d = \begin{cases} a & if\ ((b\ or\ c)\ =\ a-1)\ or\ (S_i = T_j) \\ a+2 & if\ ((b\ and\ c)\ =\ a+1)\ and\ (S_i \neq T_j) \end{cases}$$

- d can only take two values: a or a+2
- Change in d encoded in 1-bit (accumulated with a counter at end of systolic array)

Systolic Cell

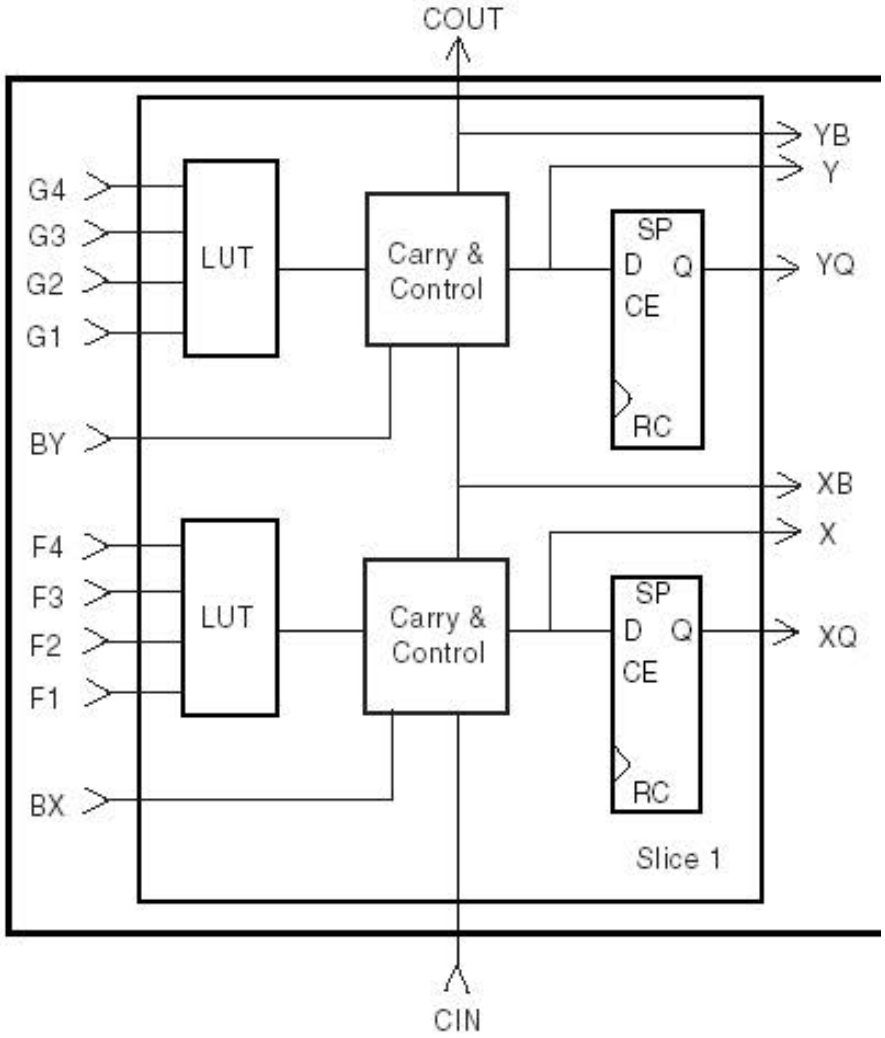


4 LUTs and 8 FFs = 4 slices

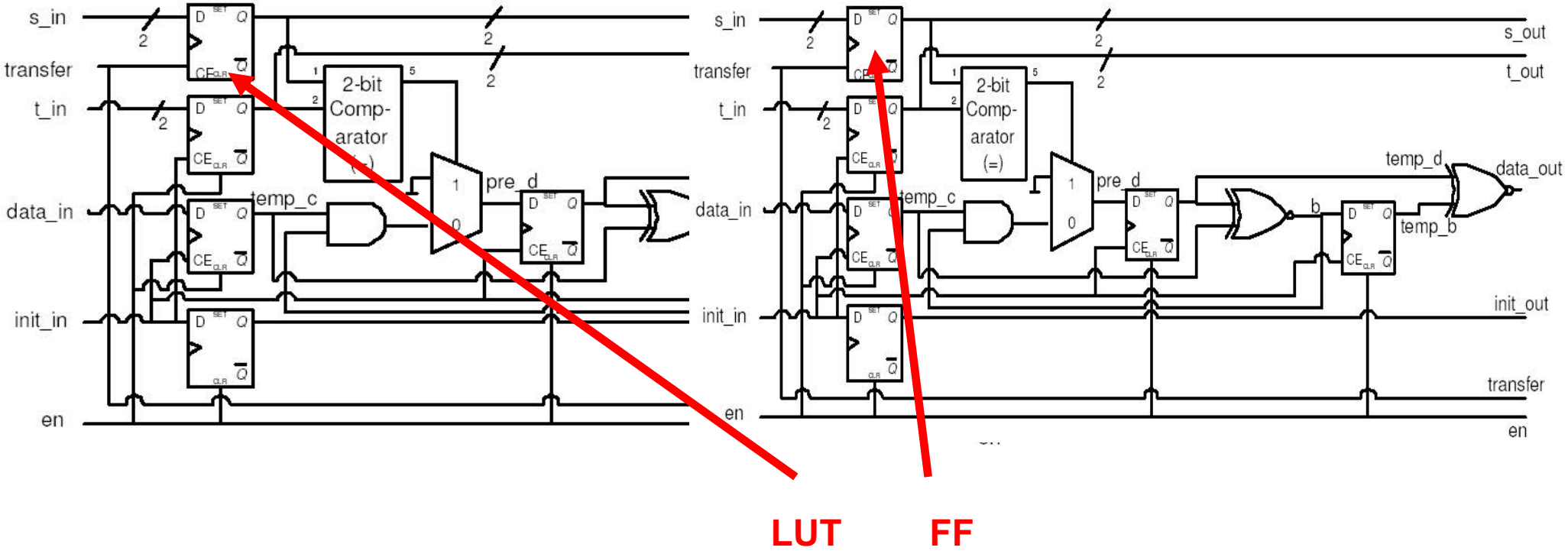
Guccione's cell uses 3 slices by writing string in bitstream

An idea

- Cell uses 4 LUTs, 8 FFs
- LUTs can serve as FFs using the distributed RAM feature
- Want to use 2 LUTs to implement 2 FFs
- Problem: there is no input which connects directly to the FF (can only do it via the LUT)



Solution



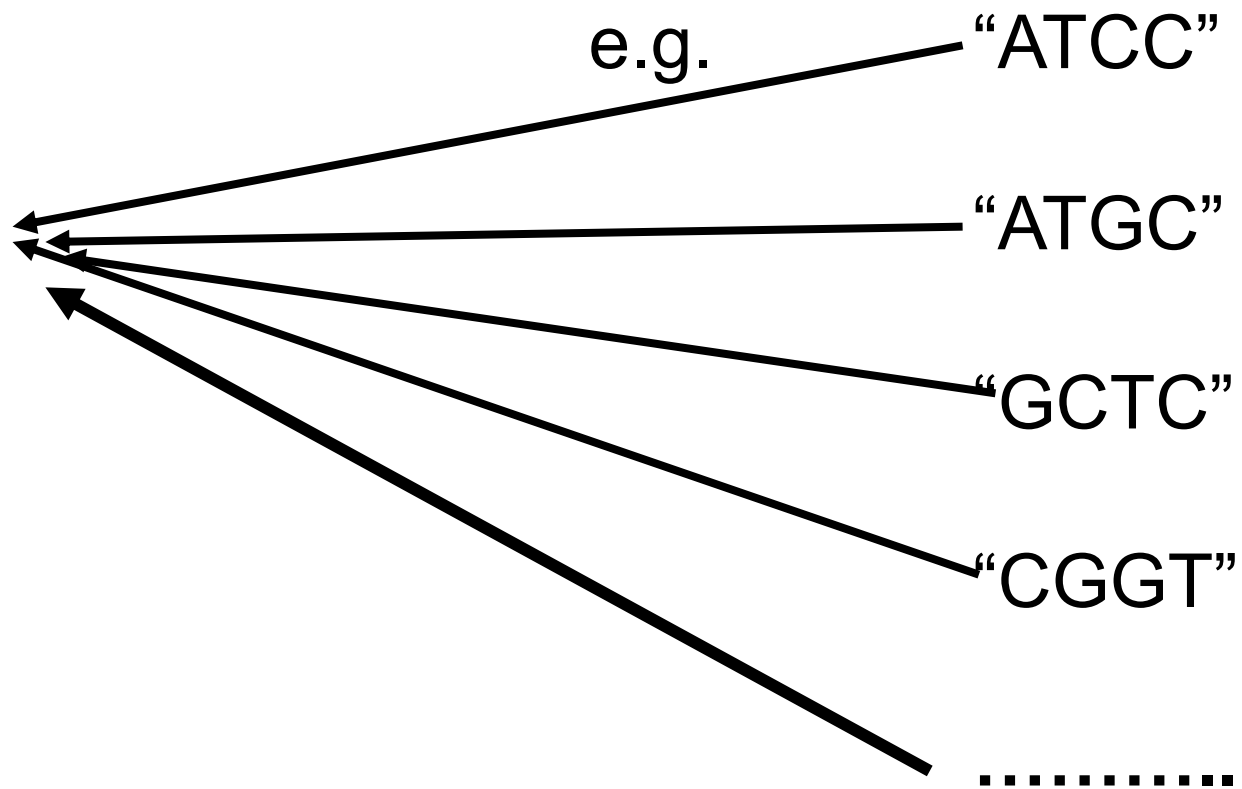
Merge two cells so that FFs are connected as in the slice (8 LUTs, 4 RAMs, 12 FFs) = 6 slices (2 cells)

The data input

Changing Sequence (T)

Sequence (S)

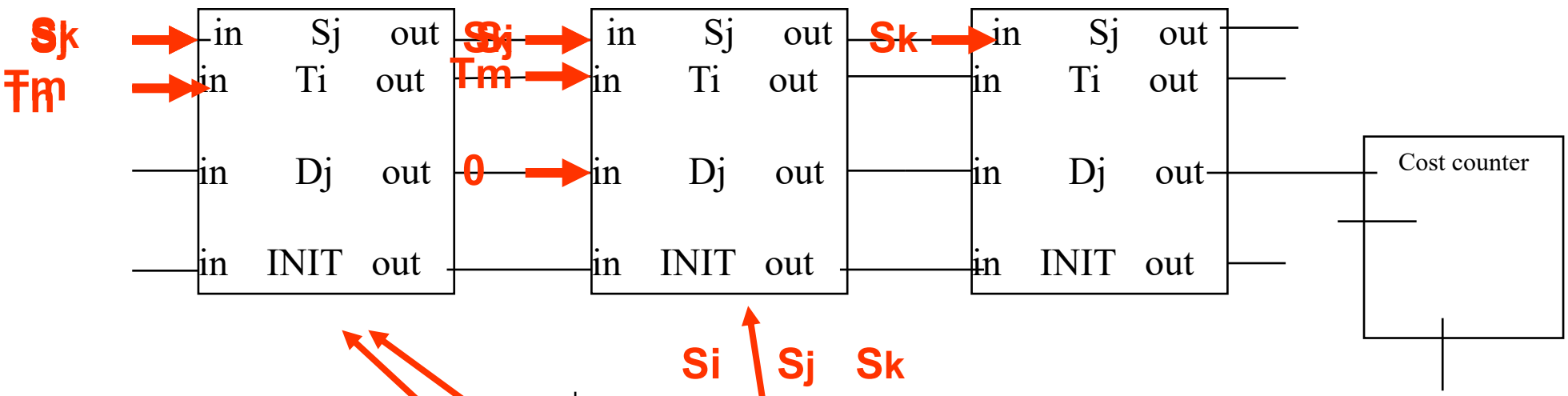
e.g. “ACGT”

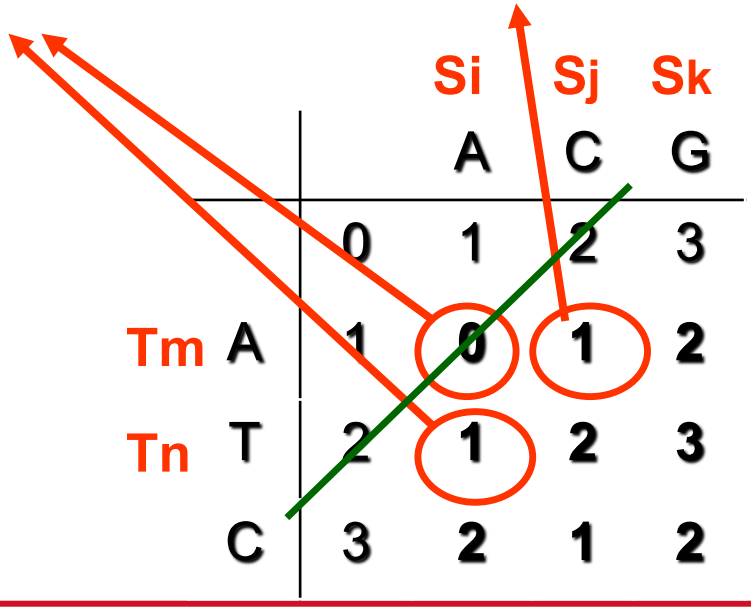


Implementation

- Input the S sequence
- Stream the changing sequences T through the array
- Read back the “alignment score”

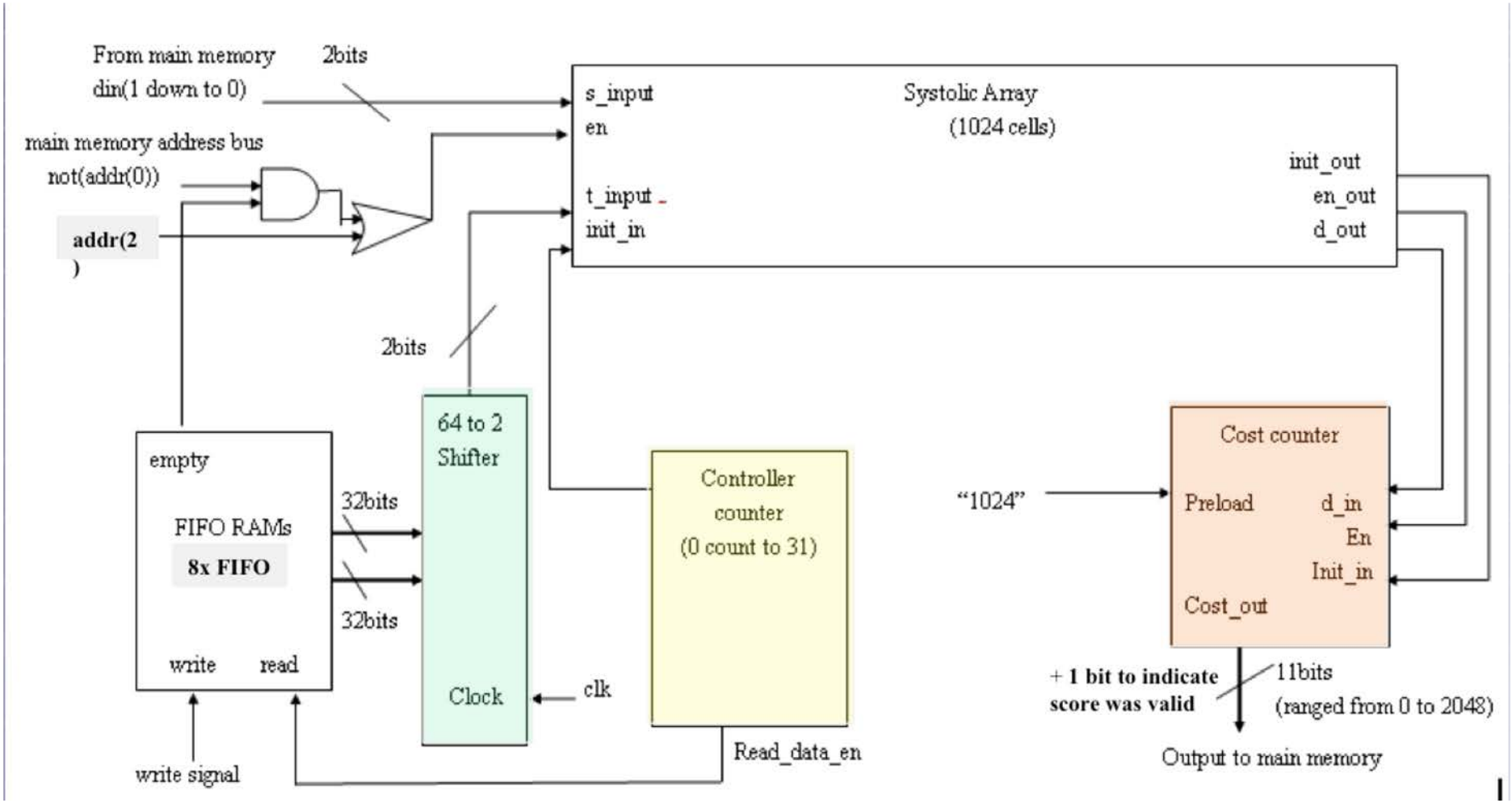
Implementation





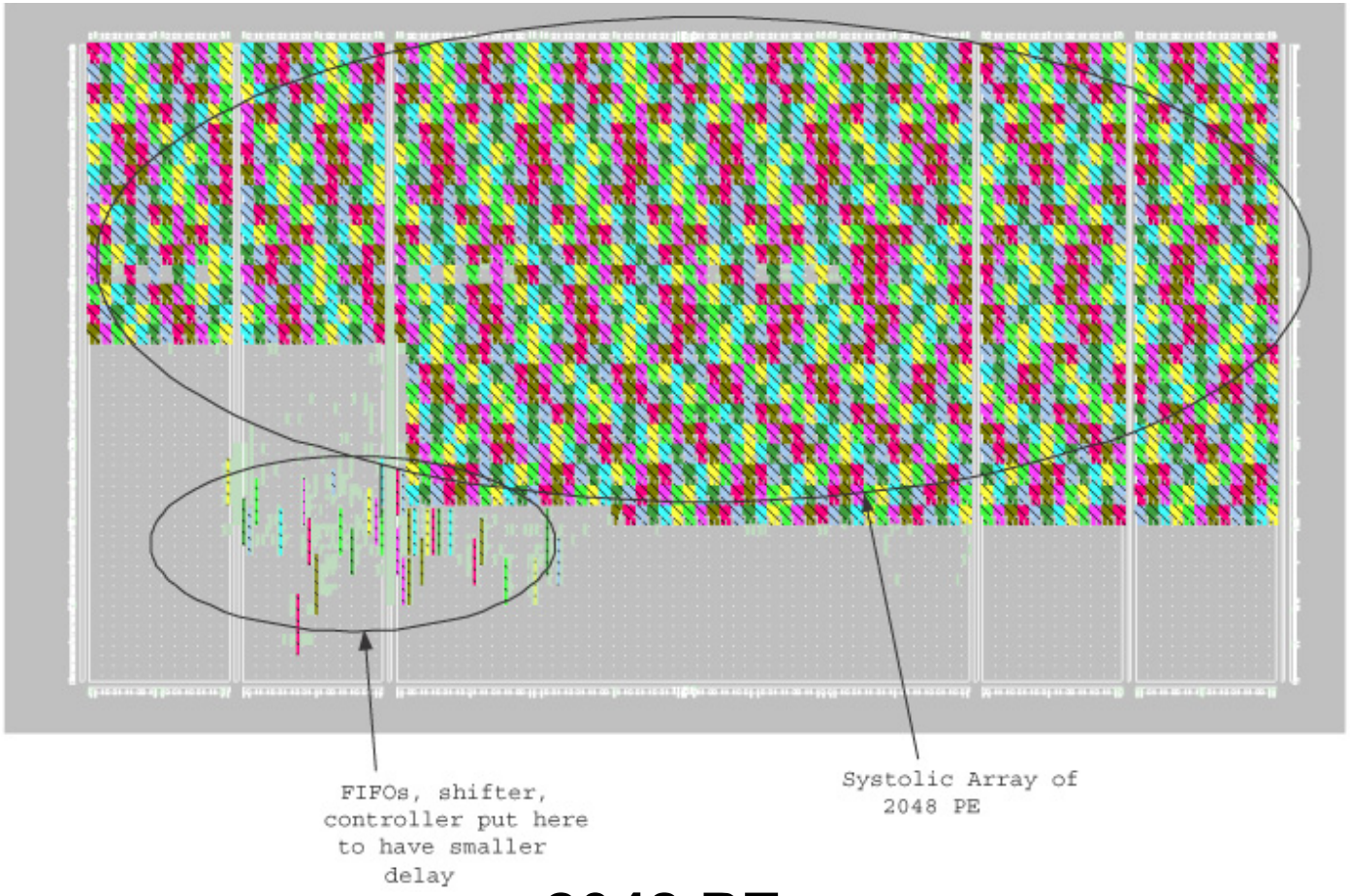
and so on…

Host interface

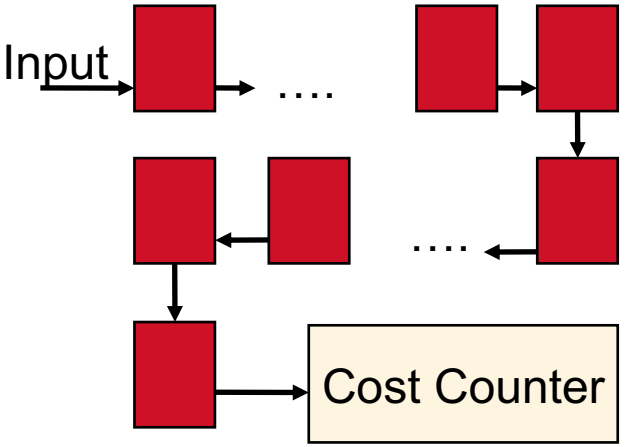


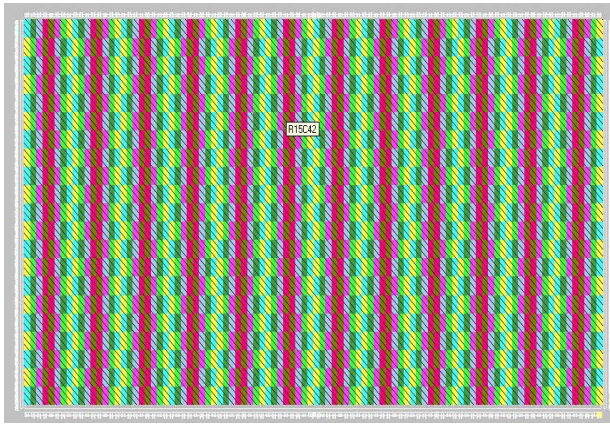
Compare 2 sequences

Floorplan



2048 PEs





4032 PEs

Results

- Implemented on a Xilinx XCV1000E-6
- 4032 cells implemented
- Operates at 202 MHz
- Performance of 814 billion connection updates per second

- Correctness verified on Pilchard system

Performance

System	Device	Performance (billion CUPS)
Splash 2	16× XC4010	2.7
Paracel	ASIC	1.9
Celera (software)	800 DEC Alphas	0.3
JBits	XCV1000-6	757
This work	XCV1000-6	742

Conclusion

- A new Smith-Waterman systolic cell was presented
- Proposed approach has the same area and speed of previous work
- Does not require runtime reconfiguration
  - Changing S is much faster
  - Can be adapted to other FPGA and ASIC families

References

› C.W. Yu, K.H. Kwong, K.H. Lee, and P.H.W. Leong. A Smith-Waterman systolic cell. In Proc. International Conference on Field Programmable Logic and Applications (FPL), pages 375–384, 2003