A 4-mm^2 180-nm-CMOS 15-Giga-Cell-Updates-per-Second DNA Sequence Alignment Engine Based on Asynchronous Race Conditions

Advait Madhavan
Electrical and
Computer Engineering
UC Banta Barbara
advait@ece.ucsb.edu

Timothy Sherwood
Computer Science
UC Banta Barbara
sherwood@cs.ucsb.edu

Dmitri Strukov
Electrical and
Computer Engineering
UC Banta Barbara
strukov@ece.ucsb.edu

***Abstract*—We have fabricated and successfully tested, for the first time, a prototype chip of a Race Logic computing paradigm, which makes positive use of race conditions for accelerating a broad class of optimization problems, such as ones solved by dynamic programming algorithms. In Race Logic, information is encoded in signal propagation delay, rather than conventional logic levels, and the result of the computation is observed from relative timing differences between injected signals, i.e. the outcome of races. The 2×2 mm^2 chip, fabricated in standard 180-nm CMOS technology, is designed to perform real-world DNA sequence alignment. Measurement results on typical benchmark data show 15 GCUPS sustained throughput at 70 mW power consumption, with only ~ 15 mW spent for actual computation. These numbers compare very favorably with the state-of-the-art implementations.**

I. Introduction

As we enter into a leakage limited regime, faced with transistor utilization issues, the proliferation of hardware accelerators seems inevitable [2], [9]. In order to extract maximum performance and energy efficiency, researchers have begun questioning fundamental assumptions of traditional designs such as precision and binary encoding. Guided by error tolerant applications and human perceptual imperfections, approximate circuits are coming to the fore, in which the digital domain is often abandoned to improve energy efficiency [4], [12], [13]. In this paper, we present a prototype of an architecture, that seems to lie somewhere between traditional digital and analog realms. Though all wires are driven to a ”1” or ”0” like in digital systems, information is encoded in the delay of the signal, which allows a continuum of values to be encoded onto a single wire. Computation is then performed by observing the relative timing differences between injected signals in a reconfigurable circuit. Such a temporal information representation makes some arithmetic operations, such as additions and comparisons, trivial to implement, allowing for significant performance and energy gains.

The race formulation is especially well suited to solve dynamic-programming-like problems. Though a whole host of such problems exist [6], we chose the well studied problem of DNA sequence alignment [1] such that a fair comparison can be made. With the field of personalized medicine setting the goal of a complete genome sequencing at a $1000, sequence alignment has recently seen considerable research activity with a number of general purpose [3], GPU [5], as well as ASIC [11] implementations. In contrast to these methods, we present for the first time a prototype of a fully custom 4 mm^2 DNA sequence alignment engine fabricated in 0.18 μm process, that makes positive use of asynchronous race conditions.

II. Background

The core idea behind race logic is to encode information in a timing delay, which regardless of its implementation, simplifies certain computation primitives [7]. For example, in the case of addition, simply stacking two delay elements one after another causes the resultant delay to be a sum of the two. Similarly, in the case of comparisons, the smallest (or largest) magnitude between multiple data values can be easily computed by arranging the corresponding delay values in parallel and selecting the first arriving (or last arriving) signal. Figure 1a shows the hardware implementations of simple addition and comparison operations with simple CMOS primitives, under the basic assumption that the event that is being delayed is a rising edge. It can be seen that selecting the first (or last) arriving rising edge can be implemented with OR (or AND) gates respectively. The simple ADD, MIN and MAX primitives are not to be underestimated, as when chained effectively, the can prove to be very powerful operations.

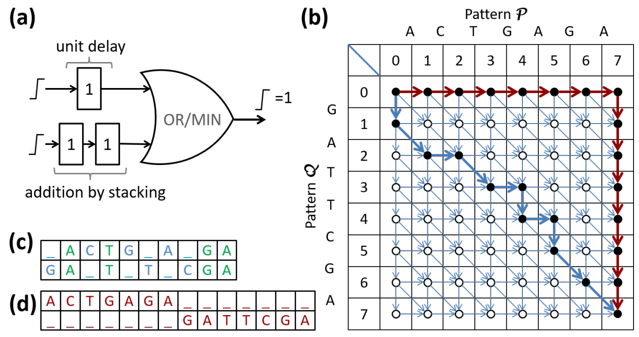

Fig. 1. (a) Race Logic basics showing unit delays, addition, and comparison techniques. (b) Edit graphs with two highlighted paths representative of (c) nominal case and (d) worst case alignments. Best case alignment would involve only the major diagonal and is not representative of such a computation.

To understand how these simple primitives can be chained effectively, we look to examples of shortest paths on directed acyclic graphs (DAGs), e.g. as shown in Figure 1b, which are known to be good representations of dynamic programming algorithms. Starting from the root nodes, the shortest path to the next node is computed by performing addition and comparison operations at that node. Conventional software methods implement this algorithm by stepping through nodes and following graph dependencies one computation step at a time, while GPU and FPGA based techniques aid in speedup by concurrently compute the scores of independent nodes.

ASIC implementations use either systolic arrays or heuristic SRAM-based processing elements with clever encoding schemes for density and performance.

Contrary to such implementations, the delay encoding explicitly constructs the graphical dependency chain in hardware by replacing edges with delay elements and nodes with OR gates respectively. A counter is used to calculate the total time taken to traverse the graph which tells us the length of the optimal path.

To highlight the full potential of Race Logic, we chose the well studied problem of DNA sequence alignment, in which the objective is to measure the similarity between two given DNA sequences (also called patterns or strings) based on a given metric, known as the edit distance. String similarity is a typical bottleneck operation, whether it is in reference-assisted or de-novo DNA sequencing and is performed billions of times in the sequencing of a whole human genome [1]. Not only does his problem map very effectively to a shortest path on a graph, which can be implemented in a reconfigurable way for different sequences, the resultant graph has a regular structure with the edges weights having a dynamic range of about an order of magnitude. This edit graph is a two-dimensional representation of all the possible alignments between the two input sequences as shown in Fig. 1b. Any specific alignment is just a path in this graph where every edge corresponds to an edit operation (vertical arrows = insertions, horizontal arrows = deletions and diagonal arrows = matches or mismatches). To select the alignment with the maximum number of matches, a scoring function (or matrix) is introduced, which penalizes mismatches with a higher score and a match with a lower one. Determining the "best" alignment is therefore a matter of finding the shortest path in the graph. Typical score matrices can be written as as [1, 2, 1] [1, 4, 3] etc., where the scores represent [match, mismatch, in-dels], respectively.

III. Race Logic Implementation

In order to be representative of the state of the art, we chose a problem size of 50 which lies within the nominal range of NGS sequence data. Using two 50-symbol long strings, a reconfigurable edit graph structure, similar to the one shown in Fig. 1b, is implemented in which weights of the match and mismatch conditions are based on the real score matrices from Ref. [1]. Though the structure of the graph looks the same for all query DNA sequence input, the magnitudes of the particular edge weights (i.e. values of delays) are governed only the by the specific input pair of sequences. For each new pair of sequences, time spent in navigating the resultant graph is measured using a clock, and is representative of the quality of alignment/similarity between the respective sequences. The choice of delay element is also an important one as previous studies have shown that synchronous delay elements are very area expensive and incur cubic energy scaling with problem size[7]. Current starved inverter based asynchronous delay elements have been proposed as a compact, fast and energy efficient alternative [8]. The main result of this paper is that we show that Race Logic with 10,000 delay elements can be designed in a robust and variation-tolerant manner.

A. Race Logic Array and Bias Networks

The 2,500 cell array consists of repetitions of a fundamental unit cell whose tileable structure is shown in Fig. 2b. Each cell implements 4 delay elements, XOR based matching circuitry and a symmetric resettable OR gate at the input. The timing on the array reset circuitry is adjusted externally to ensure that all delay elements have been reset. Each delay element is constructed out of a current starved inverter with its output split by the current control transistor for better matching [10], followed by a regular minimum size inverter to sharpen the edge (Fig. 2c). Since only rising edges go through the delay element, the current starved inverter only uses NMOS bias and cascode control nodes. Each cell receives 2-bit *symbol input* from the nucleotides being compared, to determine which delays to choose based on match and mismatch conditions, and *rising edge input* from its top, left and diagonal neighbours. The symbol input goes into XOR-based matching circuitry which chooses a diagonal delay element through a pass-gate MUX to make sure that both paths have similar delay characteristics. Dummy pass-gates were also added in off diagonal paths to ensure similar delay across all delay paths.

In the design of this first prototype, functional correctness and accuracy in the face of process variations and switching/coupling noise was a major concern. Intra-die process variations by virtue of large array size (~ 1 mm a side) as well as switching noise from high speed switching activity could both negatively affect correctness of the array with injected noise being data dependant and hence more important to get rid of. To address these issues, we proposed heavy decoupling of bias nodes with MIM-caps to keep silicon area down to a minimum, as well as partitioning of the array into regions of 10×10 with their own local bias networks that allow for decoupling of switching activity from one part of the array to another as shown in Figure 3c. The local biases networks (Fig. 3b) are generated by a set of global bias networks that distribute timing information across the entire array. To ensure that the generated currents are variable, yet tolerant to process and supply variations, the global bias network uses an Op-Amp to pin a fixed voltage across a precision potentiometer.

B. I/O system, Clock and Control Logic

The I/O system consists of a binary coded SPI interface for sequence inputs with the following encoding for DNA nucleotides: $A =$ "00", $G =$ "01", $C =$ "10" and $D =$ "11". The system can load up to four 50-symbol long sequences at once and run them in either patterned or burst mode. The patterned mode allows for checking functional correctness of each pattern against a reference, while the burst mode repeats one sequence pair alignment at maximum allowable throughput of the system until the user initiates stoppage. The idea here is to allow enough time for correct measurement of power through the external system.

The function of the clock generator block (Fig. 2a), is to generate an output clock to count the time period of the critical path race in unit delay steps. Since the exact unit delay is not known post fabrication, we directly vary the supply voltage of a 11-stage ring oscillator, to generate large dynamic range (700 ps to 15 ns), controllable, and calibratable clock. A differential amplifier based level shifter was used to scale the clock signal, which is then buffered out to the rest of the circuit, as well as 12 bit pre-scaled counter that divides it down for off-chip measurement.

The function of the control logic block is to take all the elements discussed in this section and create a simple state machine based interface that can be used externally for performance and power measurement. The state machine of

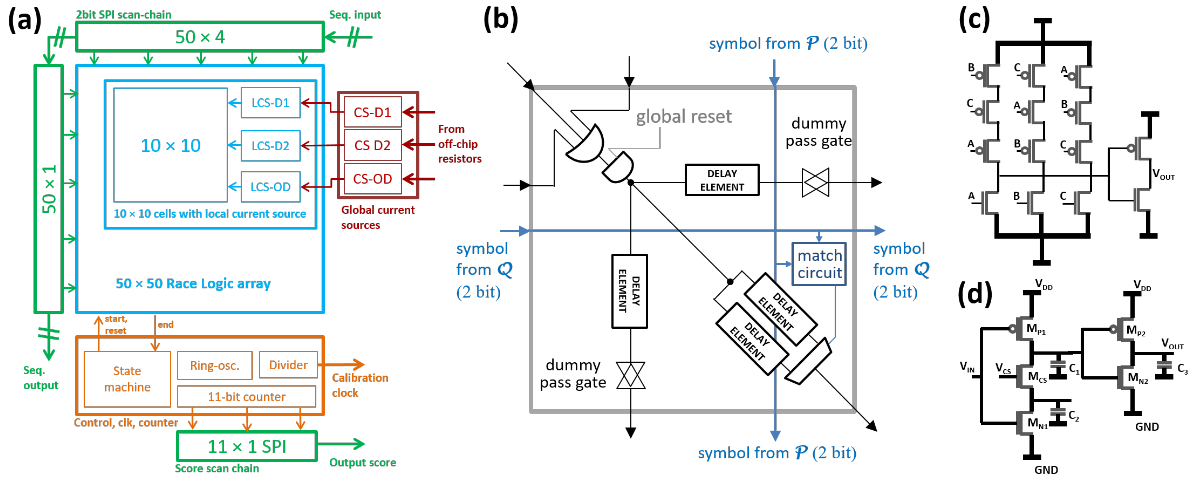

Fig. 2. (a) Organization of the implemented chip. CS D1, D2 and OD represent current control for diagonal match, diagonal mismatch, and off diagonal delays, respectively, while LCS represents the local current sources. (b) Unit cell of the Race Logic array and gate level implementation of its (c) symmetric OR gate, and (d) delay element. The implemented delay element also has a cascode control node (not shown on panel d).

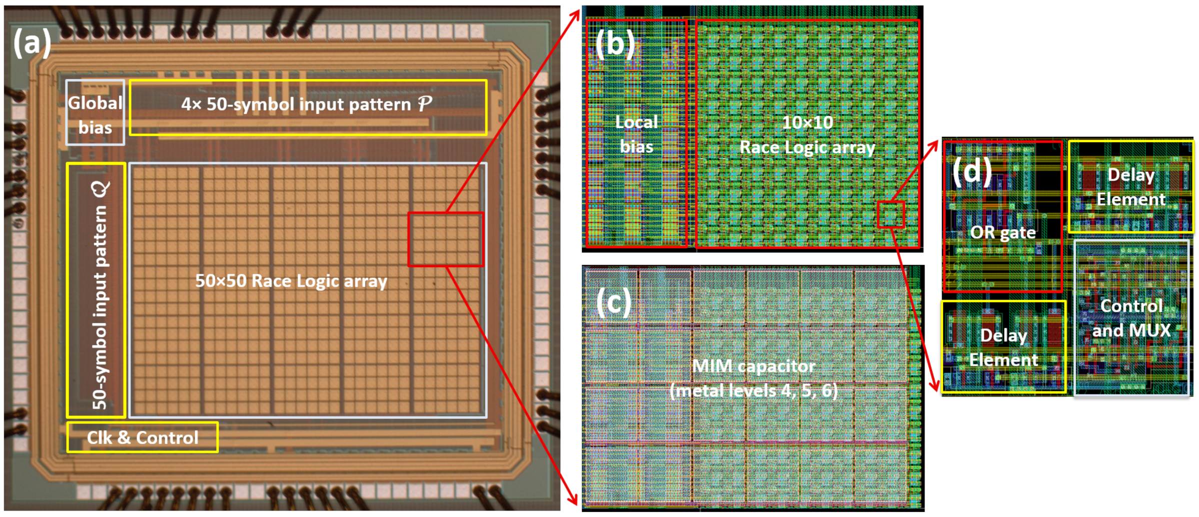

Fig. 3. (a) Micrograph of the Race Logic chip with its major functional units highlighted. The Race Logic array is an explicit implementation of the edit graph, and is reconfigured every computation based on the input patterns. (b) and (c) show a 10 × 10 race array with MIM-caps and local bias network. The local bias networks receive their control input from the global bias network as shown in Fig. 2a. Panels (d) and (e) show the circuit and layout of the unit cell which is tiled to construct the whole array, while panels (f) and (g) show symmetric OR gate design and delay element.

such a control logic block is responsible for interfacing with the array, and counter blocks, starting the race computation (array and counter), detecting the end of the race resetting the array and counter.

IV. Results and Discussion

Characterization of the delay elements revealed a minimum unit delay of approx 2 ns, with about an order of magnitude of control over the dynamic range through the precision potentiometer. By using specific input sequences, diagonal and off diagonal paths can be specifically isolated allowing reliable construction of score matrices from NCBI blast benchmark (such as, e.g. [1, 2, 1] [1, 4, 3] [1, 6, 4] [1, 10, 6]).

To test the functionality of the designed circuit, we ran similarity measures on real DNA sequence data from human genome by simulating the process of its shotgun sequencing. In particular, a section from chromosome 1 was partitioned into 50-symbol long sequences taken at random places, with a coverage of 15. These “shotgunned strands” are compared against each other, both in simulation as well as our implementation. The results for score matrix [1, 4, 3] for a set of 100 sequence samples is shown in Figure 4. The measured score closely tracks the expected one with about 2.9% error due to process, mismatch and noise based variations, while the average and maximum sequencing throughput are $\sim 10^{10}$ / $\sim 2.5 \times 10^{10}$ cell-updates-per-second (CUPS), respectively. For the representative threshold value of 90 [8], the throughput is close to $\sim 1.5 \times 10^{10}$ CUPS.

In order to understand the power distribution of the array,

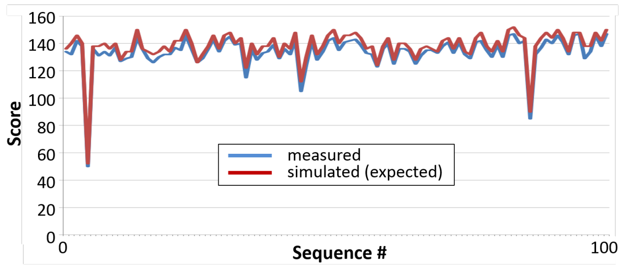

Fig. 4. Expected versus measured alignment scores for the case when multiple sequences are compared against a reference sequence.

we separated the power domains such that the array bias and switching power could be measured independently of each other. Using specific input we isolated the best case and worst case array paths and compared energy numbers for various score matrices (Table I). For very large delay values, bias currents are very small and hence idle power is in the low mWs range. When the match condition is isolated for maximum speed, 25 local biases are simultaneously activated causing an increase in bias power. Switching power is, however, minimally affected as only the main diagonal is switching. For the case of [1, inf, 1] score matrix, the bias power is almost doubled as expected, but both best and worst case power remains the same. Careful analysis reveals that though the best case alignment takes half the time as the worst case one, only the upper left triangle of the array switches, whereas in the worst case the whole array switches. Hence both cases have similar power. Looking at more representative score matrices (bottom two rows of the table) we see that at maximum throughput the power of the array is about 70 mW.

Score matrix	Bias power (mW)	Array power (mW)		Comments
		best	worst	
[inf, inf, inf]	1.53	4.08	3.90	idle power
[1, inf , inf]	48.96	5.58	N/A	switching at unit delay along central diagonal path whole array biased
[1, inf, 1]	82.8	26.1	26.46	diagonal & off diagonal paths switching at unit delay whole array biased
[1, 4, 3]	52.38	14.94	11.52	all paths switching with their own delay values whole array biased
[1, 6, 4]	51.12	13.32	9.9	

TABLE I. MEASURED POWER NUMBERS FOR BEST AND WORST CASE ALIGNMENTS FOR VARIOUS SCORE MATRICES.

The performance compares very favorably with those of state-of-the-art implementations [5], [11], [3] (Table II). (The table only shows results for comparable alignment algorithms, and, e.g., does not include recent work with higher reported throughput due to much simpler, dynamic-range-of-2 score matrix.) Clearly, the throughput could be easily increased for DNA sequence alignment problem by processing multiple strings in parallel in different arrays, and will roughly scale linearly with chip area for Race Logic implementation. A crude estimates show that the Race Logic implementation based on the same process and comparable chip area could have at least two orders of magnitude higher throughput at similar power consumption as compared to GPU implementations [5]. Moreover, threshold-based strategies, discussed in Refs. [8], [1], can be used to further improve Race Logic throughput and power consumption. It should be noted, that though this work focus on accelerating a particular task, we expect that Race Logic would be useful for variety of other applications which heavily rely on dynamic programming algorithms, such as protein alignment, image seam carving, dynamic time warping, stereo correspondence [6].

Architecture	Process (nm)	Chip area (mm^2)	Through put (GCUPS)	Ref.	Comments
GPU	28	300/600	119/185	[5]	Single/dual GPU GeForce GTX 680/690
ASIP	90	n/a	9	[11]	Application specific 32-core processor
SIMD	65	143	3	[3]	SSE2 of 2.0 GHz Xeon Core 2 Duo
Race Logic	180	4	15	-	This work

TABLE II. COMPARISON OF THIS WORK WITH RECENT ASIC, GPU AND SIMD IMPLEMENTATIONS.

V. CONCLUSION

This paper presents a functional prototype of a DNA sequence alignment engine based on radically different computing paradigm - asynchronous Race Logic to perform high throughout and low energy computation. A 4 mm^2 prototype chip, fabricated in 180 nm CMOS process, allows for 15×10^9 CUPS throughput at 70 mW power when running practical DNA sequence alignment tasks, which compares very favorably with state-of-the-art results.

ACKNOWLEDGMENT

The authors would like to thank Melika Payvand for her help in chip layout and tapeout.

REFERENCES

[1] R. Ekblom and J. B. Wolf. A field guide to whole-genome sequencing, assembly and annotation. *Evolutionary applications*, 7(9):1026–1042, 2014.

[2] H. Esmaeilzadeh, E. Blem, et al. Dark silicon and the end of multicore scaling. In *Proc. ISCA'11*, pages 365–376, 2011.

[3] M. Farrar. Striped smith–waterman speeds database searches six times over other simd implementations. *Bioinformatics*, 23(2):156–161, 2007.

[4] Y. Huang, N. Guo, M. Seok, Y. Tsividis, and S. Sethumadhavan. Evaluation of an analog accelerator for linear algebra. In *Proc. ISCA'16*, pages 570–582, 2016.

[5] Y. Liu, A. Wirawan, and B. Schmidt. Cudasw++ 3.0: accelerating smith-waterman protein database search by coupling cpu and gpu simd instructions. *BMC bioinformatics*, 14(1):1, 2013.

[6] A. Madhavan. *Abusing hardware race conditions to perform useful computation*. PhD thesis, UC Santa Barbara, 2016.

[7] A. Madhavan, T. Sherwood, and D. Strukov. Race logic: A hardware acceleration for dynamic programming algorithms. In *Proc. ISCA'14*, pages 517–528. IEEE, 2014.

[8] A. Madhavan, T. Sherwood, and D. Strukov. Energy efficient computation with asynchronous races. In *Proc. DAC'16*, page 108, 2016.

[9] I. Magaki, M. Khazraee, L. V. Gutierrez, and M. B. Taylor. Asic clouds: Specializing the datacenter. In *Proc. ISCA'16*, pages 178–190, 2016.

[10] P. Mroszczyk and P. Dudek. Tunable CMOS delay gate with reduced impact of fabrication mismatch on timing parameters. In *Proc. NEWCAS'13*, pages 1–4, 2013.

[11] N. Neves, N. Sebastião, D. Matos, P. Tomás, P. Flores, and N. Roma. Multicore simd asip for next-generation sequencing and alignment biochip platforms. *IEEE Trans. VLSI*, 23(7):1287–1300, 2015.

[12] R. St Amant, A. Yazdanbakhsh, J. Park, B. Thwaites, H. Esmaeilzadeh, A. Hassibi, L. Ceze, and D. Burger. General-purpose code acceleration with limited-precision analog computation. *ACM SIGARCH Computer Architecture News*, 42(3):505–516, 2014.

[13] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan. Approximate computing and the quest for computing efficiency. In *Proc. DAC'15*, page 120, 2015.