Parallel Binary Sorting and Shifting with DNA

Tonglin Chen
chen5202@umn.edu
University of Minnesota
Minneapolis, MN

Marc Riedel
mriedel@umn.edu
University of Minnesota
Minneapolis, MN

1 INTRODUCTION

This abstract presents a novel scheme for sorting and shifting binary values stored in DNA. The computation is effected via the mechanism of toe-mediated strand displacement [5], [1]. Recent research by the DNA-computing community has demonstrated how data can be stored in DNA by "nicking" it using gene-editing techniques [2]. Nicking can expose toeholds for strand displacement. Wang et al. have proposed a scheme called SIMDNA, for "Single Instruction Multiple Data DNA," that implements parallel DNA strand displacement operations on binary bits [3]. In their scheme, binary bits are encoded in segments of DNA called "cells". Instructions consist of strand displacement operations on the encoded bits, changing the location of exposed toeholds. Building on the work of Wang et al., we propose a scheme for performing parallel sorting and shifting of binary bits.

2 DESIGN OF THE SYSTEM

The coding scheme that we use for binary values is shown in Figure 1. For each cell, there are five regions. Region 1 is always exposed as a toehold. Regions 2 through 5 are covered. When storing a bit value of 0, there is a nick between regions 2 and 3; when storing a bit value of 1, there is a nick between regions 4 and 5.

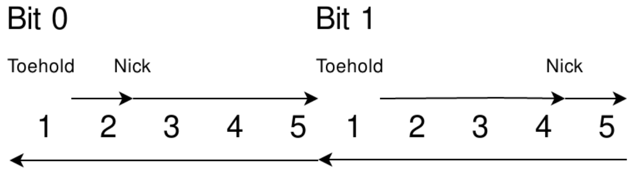

Figure 1: The coding scheme for bit values of 0 and 1.

Parallel Binary Bubble Sorting

We describe a scheme for sorting an arbitrary sequence of binary values. With the classical bubble sorting algorithm, a sequence of values is sorted by comparing and swapping adjacent values, from the beginning to the end of the sequence. With n values, the process is repeated n times, for a total run time of $O(n^2)$ steps. Sorting arbitrary values can be parallelized, but alternate passes are needed, sorting first odd and then even values, in order to avoid swapping conflicts.

With binary values, the bubble sorting algorithm is considerably simpler: it can be parallelized without alternate passes. To see why, note that there are four possible pairs of bits: $(0, 0)$, $(0, 1)$, $(1, 0)$ and $(1, 1)$. Of these, only the pair $(1, 0)$ needs a swap. In any sequence of three bits, there can be at most one swap. Performing the compare and swap operations on all pairs of bits in sequence of n bits in parallel, the algorithm requires $O(n)$ parallel steps.

The basic idea for implementing the instructions for sorting with DNA strand displacement is first to identify all the $(1, 0)$ pairs and expose their corresponding toeholds. Then rewrite the data in those positions. Figure 2 shows the sequence of instructions for each step (a total of 12 instructions).

Instructions 1 and 2 identify the combination of $(1, 0)$. The toehold between a bit value of 1 and a bit value of 0 is replaced by a strand with a label of S1. Instruction 3 seals off the region exposed during Instruction 1 and 2. In instruction 4, the strand with label S1 is detached, exposing region 5, in the case of a bit value of 1, or region 2, in the case of a bit value of 1. In instruction 5, in the case of a bit value of 0, region 2 is temporarily covered by a strand with label S2. In instruction 6, a bit value of 1 is replaced by a strand with label S3 via the toehold at region 5. The strand is then detached and the bit value of 0 is written to the location of a bit value of 1 in instructions 7 and 8. In instruction 9, the temporary cover for a bit value of 0 is lifted. Then, in instructions 10 through 12, a bit value of 1 is written to the location of a bit value of 0 using the same scheme as instructions 6 through 8.

Left Shift Register

The scheme for sorting can readily be adapted to implement shifting of binary values. The basic idea for a left-shift register is the following:

- Find all the pairs $(0, 1)$ and $(1, 0)$
- Cover the toeholds for the pairs $(0, 0)$ and $(1, 1)$.
- Identify the pairs $(1, 0)$ and flip the bit values of 1.
- Identify the pairs $(0, 1)$ and flip the bit values of 0.
- Uncover all the toeholds for the pairs $(0, 0)$ and $(1, 1)$.

Our implementation the left shift register consists of 14 instructions per shift operation. The details are omitted due to space constraints.

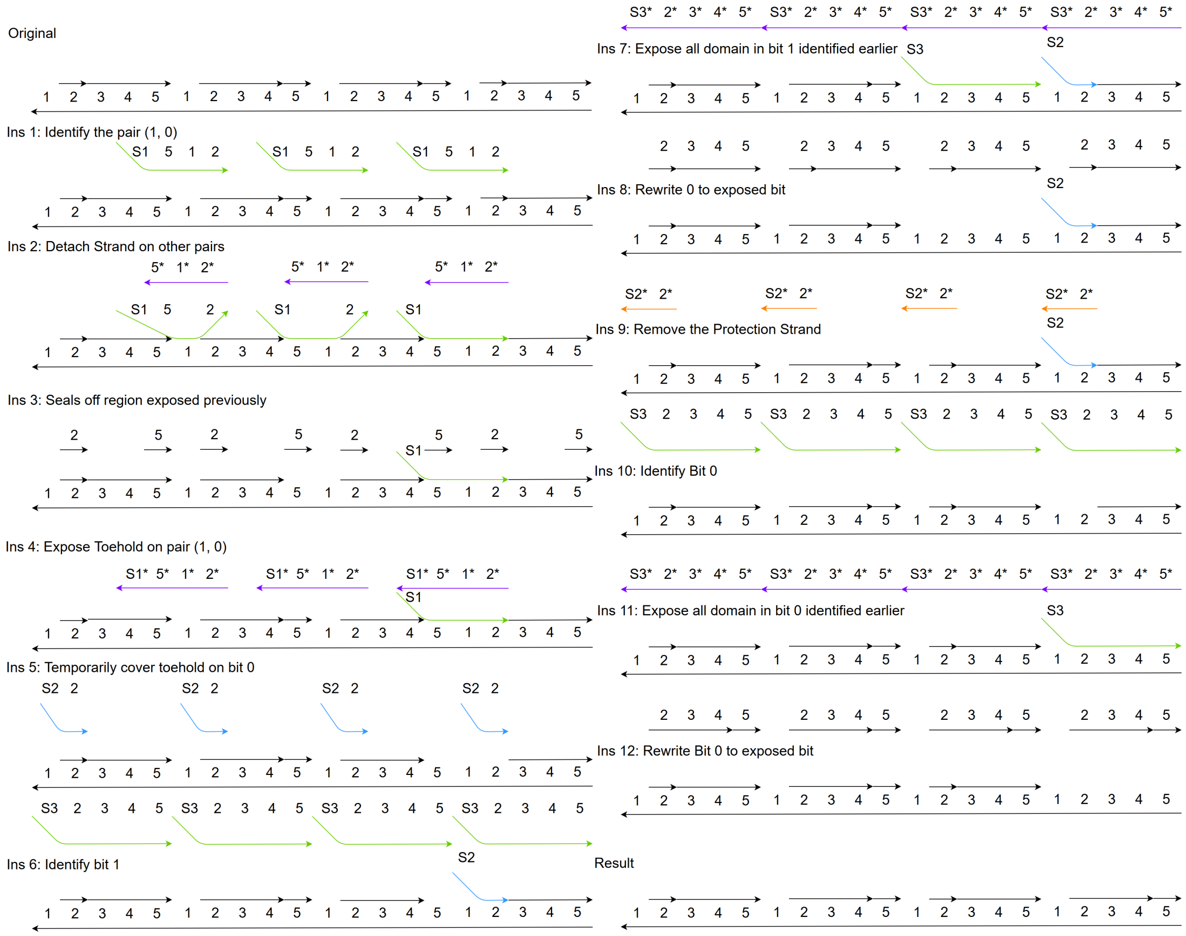
Original
Ins 1: Identify the pair (1, 0)
Ins 2: Detach Strand on other pairs
Ins 3: Seals off region exposed previously
Ins 4: Expose Toehold on pair (1, 0)
Ins 5: Temporarily cover toehold on bit 0
Ins 6: Identify bit 1
Ins 7: Expose all domain in bit 1 identified earlier
Ins 8: Rewrite 0 to exposed bit
Ins 9: Remove the Protection Strand
Ins 10: Identify Bit 0
Ins 11: Expose all domain in bit 0 identified earlier
Ins 12: Rewrite Bit 0 to exposed bit
Result
S1
S2
S3
S1* 5* 1* 2*
S3* 2* 3* 4* 5*
S2* 2*
1 2 3 4 5

Figure 2: Instructions for one parallel step of binary bubble sorting. The original data encodes 0110. After one parallel step, the data encodes 0101.

3 DISCUSSION

We proposed a scheme for pairwise parallel execution of operations on binary values via strand displacement, and discussed two possible applications: sorting and shifting. We discussed the parallel binary bubble sorting algorithm in detail as it is a natural fit for the paradigm, since it only requires pairwise exchanges. We have only validated the scheme through simulation. A practical and experimental concern is the amount of “leakage” per operation: what fraction of strand displacement operations will not execute correctly. Each parallel step consists of about 10 separate instructions; leakage will compound over the course of these 10 operations. Using leakless strand displacement [4] as part of the design is a possible strategy to mitigate experimental error. In future research, we will explore alternate encoding schemes to minimize the number of instructions per step.

4 ACKNOWLEDGEMENT

We thank David Soloveichik, Cameron Chalk, and Boya Wang for helpful discussions. This work was funded by DARPA Grant #W911NF-18-2-0032.

REFERENCES

[1] Soloveichik, D., Seelig, G., and Winfree, E. Dna as a universal substrate for chemical kinetics. *Proceedings of the National Academy of Sciences 107*, 12 (2010), 5393–5398.

[2] Tabatabaei, S. K., Wang, B., Athreya, N. B. M., Enghiad, B., Hernandez, A. G., Leburton, J.-P., Soloveichik, D., Zhao, H., and Milenkovic, O. Dna punch cards: Encoding data on native dna sequences via topological modifications. *Under Submission.*

[3] Wang, B., Chalk, C., and Soloveichik, D. Simdna: Single instruction, multiple data computation with dna strand displacement cascades. *Under Submission.*

[4] Wang, B., Thachuk, C., Ellington, A. D., Winfree, E., and Soloveichik, D. Effective design principles for leakless strand displacement systems. *Proceedings of the National Academy of Sciences 115*, 52 (2018), E12182–E12191.

[5] Yurke, B. A dna-fuelled molecular machine made of dna. *Nature 406*, 6796: 605 (2000).