DNA-Based Active Logic Design and Its Implications

[1] Christy M. Gearheart, [2] Eric C. Rouchka , [3] Benjamin Arazi
[1] Department of Computer Engineering and Computer Science, Speed School of Engineering, Duthie Center for Engineering, University of Louisville, Louisville, KY 40292 USA
[2] Department of Pediatrics, School of Medicine, University of Colorado, Aurora, CO 80045 USA
[3] Department of Electrical and Computer Engineering, Ben-Gurion University, POB 653 Beer-Sheva 84105 Israel
[2] Email: eric.rouchka@louisville.edu

ABSTRACT

DNA computing is an emerging field of research in which biochemistry and molecular biology elements are utilized to complete computational processes performed by traditional silicon-based technologies. Current DNA-based circuits are non-homogenous, consisting of discrete combinatorial gates with corresponding static processing abilities. This research focuses on further developing DNA-based methodologies to mimic digital data manipulation. Homogenous logic design principles are introduced, laying the foundations of dynamic processing whereby information can be parsed through a digital circuit comprised of DNA–based logic gates. The ultimate aim would be to facilitate shift-and-add calculation. More immediate applications concern data tamper-proofing, serving security needs whose progress was previously limited. A novel logic gate design based on chemical reactions is presented in which observance of double stranded sequences indicates a truth evaluation. Circuits are obfuscated by removing of physical sequence connections, allowing client-specific representative strands for input sequences, altering the input sequence strands over time, and varying the input sequence length.

Keywords: DNA-based logic gates, DNA-based shift register, dynamic computing, secure computing.

1. INTRODUCTION

Traditional silicon-based circuitry is susceptible to security attacks as a consequence of the static nature of its design. Metrics utilized to evaluate security are often based on the 'good feeling' of engineers rather than empirical evidence. Assessments often result in statements such as "it is practically impossible to access the memory from the outside" or "it is impossible to access the data bus that carries the key from storage to the processor if they are all on the same piece of silicon." This diminishes the entire subject of security to a mere art form rather than scientific proof [1]. The reality is, once a static circuit is obtained by an attacker, it is a matter of time before one can reverse engineer its configuration.

True tamper-proof security must satisfy three principal requirements: (1) resist static attacks that involve direct penetration of memory cells, (2) resist dynamic attacks that attempt to access information as it is moves from memory to the processing unit, and (3) resist dynamic attacks that attempt to access information during processing [1]. To circumvent such tampering, circuits must be dynamic by nature. We argue that DNA-based logic circuits, when the technology matures, may provide revolutionary solutions to tamper proofing. A DNA-based design enables circuitry to be based on biochemical and environmental stimuli. Discrete components, such as those observed in CMOS circuits, are non-existent. Tampering would thus have new meaning, possibly preventing it altogether based on accurate scientific observations.

With this vision in mind, biological methodologies have been developed to mimic existing silicon-based technologies in data manipulation as a first step. Within the digital world, data manipulation encompasses a number of essential processes, including data generation, storage, retrieval, and processing. In terms of complexity, data storage and retrieval are considered the least difficult. As such, the first step is to introduce a methodology by which information could reliably be stored and retrieved within a DNA sequence [2]. Translating between the computer science binary bits and the DNA quaternary characters is easily accomplished through a direct substitution of two binary base pairs encoding for a single DNA quaternary character. To ensure data is accurately retrieved, multiple sequence alignment is performed using multiple copies of the file to find the most probable contents. Consensus sequences from the multiple sequence alignment can serve a similar role as a parity bit by indicating if an error has occurred. However, while a parity bit can only indicate an error has occurred, multiple sequence alignment enables the location and type of error to be determined.

Once a methodology by which data can be accurately stored and retrieved is in place, the next step is to devise a methodology by which information could be generated. A novel schema for a micro fluidic chip was developed where solid phase synthesis enables random sequence generation and plasmid vectors in conjunction with restriction enzymes enable temporary storage. Chromatograms enable the random value to be output to the user. Once outside the micro fluidic chip, the random number could be stored and retrieved with the methodology presented above [3].

After construction of an information storage methodology with data validation and a random number generation circuitry, the next logical objective is to develop a DNA-based shift register. A shift register is a primary component of the computational processor that enables computation at a gate level followed by shifting of the information to the proceeding gate. Simply moving the data by itself has no computational meaning. A shift register requires the integration of both logic and shifting; thereby creating a complete processing unit that performs serial calculations on an input stream of information. Thus, its development is critical to the continued advancement of dynamic computing.

With each of these new theories introduced, we move closer to the practical applications afforded by dynamic computing. As a first step, DNA-based methodologies have been developed to mimic existing silicon-based technologies in data manipulation, specifically information storage, random number generation, and a shift register.

The rest of the paper treats the following. Active DNA-based logic gates as we perceive them to be constructed are introduced. A discussion of techniques utilized to obfuscate gate design, leading to tamper proofing, is then presented. The ability to create naturally occurring ternary logic gates using the proposed gate design is described. This is followed by a discussion of how the proposed gates could be joined to construct circuits. A biological approach to shifting is then presented, proceeded by a methodology by which DNA sequences that serve as gate inputs could be temporarily stored. Finally, circuit fabrication considerations are discussed.

2. DNA-BASED LOGIC GATES

Deoxyribonucleic acid (DNA) has a number of characteristics that enable one to mimic traditional logical operations. DNA prefers to be in double stranded form; single stranded DNA sequences naturally migrate towards complementary sequences to form double stranded complexes. Complementary sequences pair the bases adenine (A) with thymine (T) and cytosine (C) with guanine (G). DNA sequences pair in an anti parallel manner, with the 5' end of one sequence pairing with the corresponding 3' end of the complementary sequence. When complementary sequences are written in the 5' $\rightarrow$ 3' direction, the complementary sequence pairing is observed in the opposite order (read right to left), and is called the reverse complement [4].

2.1 Gate Inputs

Each DNA-based logic operation input is represented by a single stranded DNA sequence, with the requirement that the sequence representing a "true" evaluation is complementary to the sequence representing a "false" evaluation for a single gate. For example, one could assign ACCTAG to represent "true" and CTAGGT representing "false," as CTAGGT is the reverse complement of ACCTAG. This enables sequence assignment to be dynamic in nature. A user could arbitrarily assign a new set of representative sequences for each gate in a circuit. Consider the circuit comprised of three DNA-based logic gates in Figure 1. The first gate uses the sequences presented above, where ACCTAG represents "true" and CTAGGT represents "false." After evaluating the first gate, the user dynamically changes the representative input sequences, where TTTTTT now represents "true" and AAAAAA now represents "false." Finally, the user reuses the first set of sequences, but reverses the assignment such that CTAGGT now represents "true" and ACCTAG now represents "false."

2.2 Gate Computation

DNA's preference to be in double stranded form enables traditional logic operations to be performed. For each respective DNA-based gate design, a predetermined mixture is supplied containing a specific single stranded sequence to induce the appropriate chemical reaction, known as the base mixture. If the gate input sequence provided is complementary to the base sequence, then the corresponding double stranded DNA sequence will form. Thus, the presence or absence of a double stranded sequence is used to evaluate gate output; the presence of a double stranded sequence represents a "true" evaluation while its absence represents a "false" evaluation.

2.3 Detection of Sequences

Fluorescent labels can be used to detect the presence or absence of the double stranded sequence. In this process, fluorescent molecules are attached to the nucleotide sequence, and absorb and emit light at a particular wavelength. Thus, by attaching the fluorescent molecule to one of the strands of the double stranded sequence, the double stranded sequence can be detected as present by examining the sequence solution at the fluorescent probe's characteristic wavelength.

The presence of a fluorescently labeled double stranded sequence only works if the single stranded labeled sequences are removed. Deoxyribonucleic (DNA ase) is an enzyme that breaks down single stranded DNA sequences by degrading the sugar bonds connecting adjacent nucleic acids [4].

VOL. 3, NO. 5, May 2012 ISSN 2079-8407
Journal of Emerging Trends in Computing and Information Sciences
©2009-2012 CIS Journal. All rights reserved.

http://www.cisjournal.org

The final step of the DNA-based logic gates is to insert the gates' observed output as input into the next logic gate in the circuit. When a double stranded sequence is observed, the single stranded DNA sequence representing "true" for the next logic gate will be reinserted as input, while the single stranded DNA sequence representing "false" for the next logic gate will be reinserted as input in the absence of a double stranded molecule.

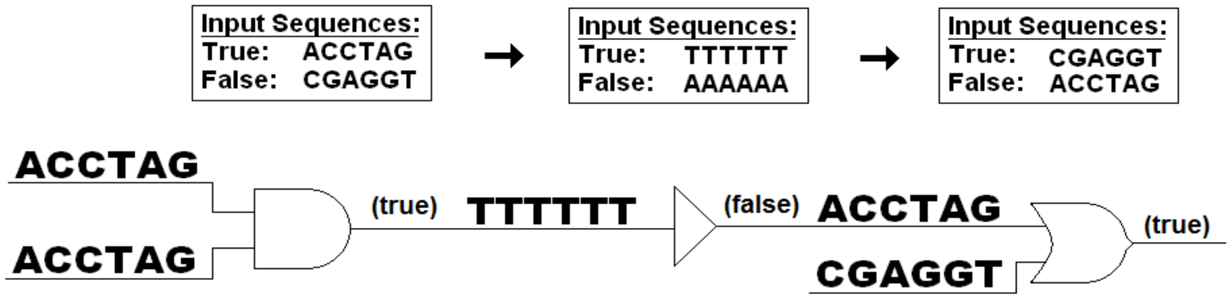

Fig 1: Complementary gate input sequences can be assigned dynamically on a gate to gate basis.

Because the representative sequences can be dynamically assigned, a new set of complementary sequences can be substituted between evaluation of the previous gate and the insertion of the representative sequence in the next proceeding gate.

Each DNA-based logic gate design is based on the preceding set of procedures. Individual gate logic is achieved through the introduction of a specific complementary sequence in the base mixture provided to each gate. Specific gate construction for traditional DNA-based Boolean logic gates for NOT, OR, XOR, and NAND are discussed in the proceeding sections. All other Boolean logic gates can be easily derived from these four pillar gates.

2.4 NOT Gate

The NOT gate, often referred to as an inverter, is one of the simplest DNA-based logic gates. Only one input is supplied to the gate, and the output is the corresponding complementary sequence. Because the output should evaluate "true" only in the presence of a "false" input, the base mixture provided to the gate contains the representative "true" sequence. DNAase is supplied to destroy any single stranded sequences. If a double stranded sequence is observed, then the result is "true"; otherwise, the result is "false."

Consider the example presented in Figure 2, where the sequence TTTTTT represents a "true" input and AAAAAA represents a "false" input. The base mixture would thus contain the sequence TTTTTT. If the input sequence is "false," then AAAAAA will bind with the provided TTTTTT sequence to form a double stranded sequence (Figure 2a). DNAase will have no effect on the sequences, and the double stranded sequence will be observed, representing a "true" evaluation. Conversely, if

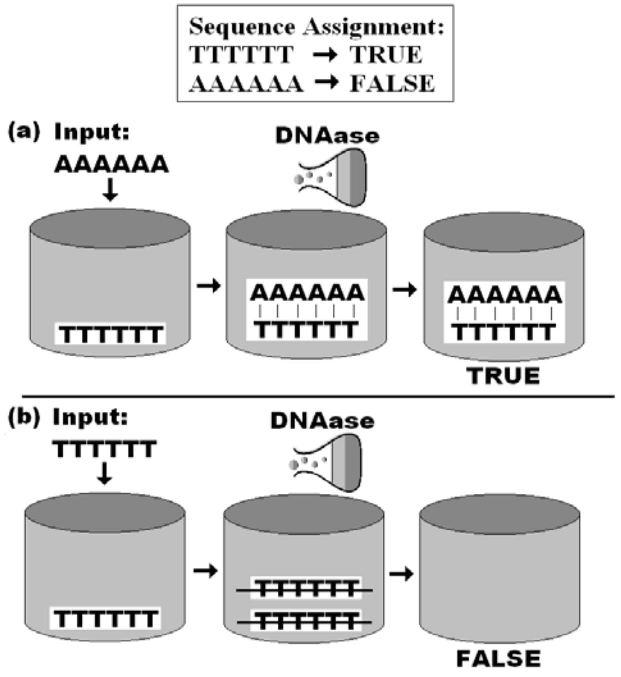

Fig 2: DNA-Based Implementation of the NOT Gate.

(a) A false input results in a double stranded sequence representing a truth evaluation while (b) a true input does not the input sequence is "true," then TTTTTT will not bind with the base TTTTTT sequence (Figure 2b). DNAase will destroy both sequences; no double stranded sequences will be observed, representing a "false" evaluation.

2.5 XOR Gate

The XOR gate evaluates "true" only if exactly one of the input sequences evaluates "true." With binary inputs, XOR is defined as evaluating "true" if input values are opposite. In DNA-based logic gates, the XOR gate is the most simplistic design in that no base sequence needs to be supplied to the gate. Opposite input sequences are complementary and will bind together to form a double stranded sequence (Figure 3a). If the inputs are not opposite, then the sequences will not be able to bind to one another and DNAase will destroy both input sequences (Figure 3b and 3c). If a double stranded sequence is observed, then the result is "true;" otherwise, the result is "false."

2.6 OR Gate

The OR gate evaluates "true" if at least one of the gate inputs are "true." Introducing the "false" sequence as the base mixture will require at least one of the inputs be "true" to form a double stranded sequence. DNAase will destroy any single stranded sequence in the mixture. If a double stranded sequence is observed, then the result is "true;" otherwise, the result is "false."

Consider the example in Figure 4 where the sequence TTTTTT represents a "true" input and the sequence AAAAAA represents a "false" input. If both of the input sequences are "true" TTTTTT sequences, then one of the sequences will combine with the base "false" AAAAAA sequence to produce a double stranded sequence (Figure 4a). DNAase will destroy the remaining input sequence and the double stranded sequence will result in a "true" evaluation.

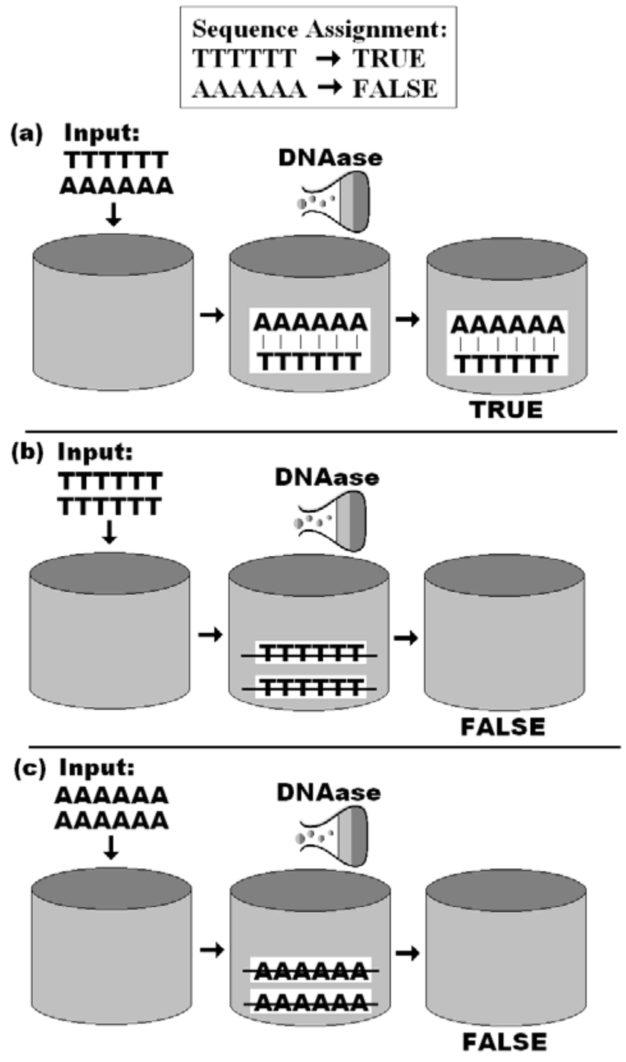

Fig 3: DNA-Based Implementation of the XOR Gate.

(a) One true and one false input results in a double stranded sequence representing a truth evaluation. Conversely, (b) two true inputs and (c) two false inputs do not create a double stranded sequence.

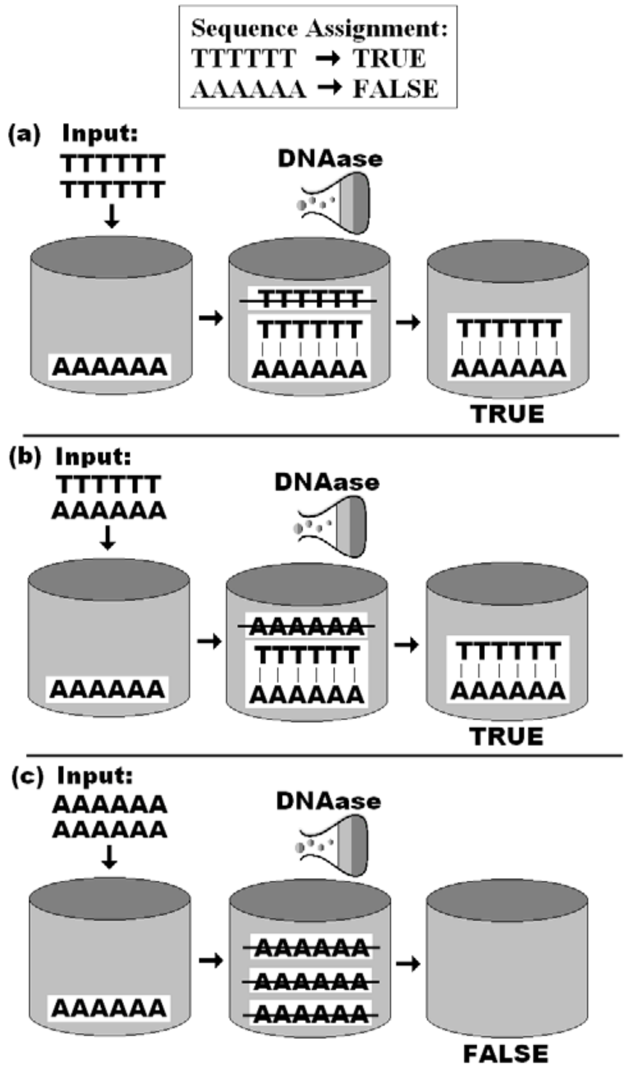

Fig 4: DNA-Based Implementation of the OR Gate.

Both (a) two true inputs and (b) one true input and one false input result in the formation of a double stranded sequence representing a truth evaluation. (c) However, two false inputs do not create a double stranded sequence.

If one input sequence is "false" and the other is "true," then the "true" TTTTTT input sequence will combine with the "false" AAAAAA sequence to produce a double stranded sequence (Figure 4b). DNAase will destroy the remaining "false" sequence and the gate will still result in a "true" evaluation.

If both input sequences are "false" AAAAAA sequences, neither will be able to combine with the base "false" sequence (Figure 4c). DNAase will destroy all sequences in the mixture, resulting in a "false" evaluation of the gate.

2.7 NAND Gate

The NAND gate evaluates "true" if inputs are not both "true," in other words, if there is at least one false input. The DNA-based NAND logic gate is similar to the OR gate presented previously, except the base sequence contains the sequence representing "true" rather than "false." Thus, at least one of the inputs must be "false" in order to form a double stranded sequence. DNAase will destroy any single stranded sequence in the mixture. If a double stranded sequence is observed, the result is "true"; otherwise, it evaluates to "false."

Continuing with the example above, if both of the input sequences are "false" AAAAAA sequences, then one will combine with the reference "true" TTTTTT sequence to produce a double stranded molecule (Figure 5a). DNAase will destroy the remaining input sequence and the double stranded sequence will result in a "true" evaluation.

If one input sequence is "false" and the other input sequence is "true," the "false" AAAAAA input sequence.

Both (a) two false inputs and (b) one true input and one false input result in the formation of a double stranded sequence representing a truth evaluation. (c) However, two true inputs do not create a double stranded sequence will combine with the "true" TTTTTT sequence to produce the necessary double stranded sequence (Figure 5b). DNAase will then destroy the remaining "false" sequence and the gate will still result in a "true" evaluation.

Finally, if both of the input sequences are "true" TTTTTT sequences, then neither of the sequences will be able to combine with the base "true" sequence (Figure 5c). DNAase will destroy all sequences in the mixture, resulting in a "false" evaluation of the gate.

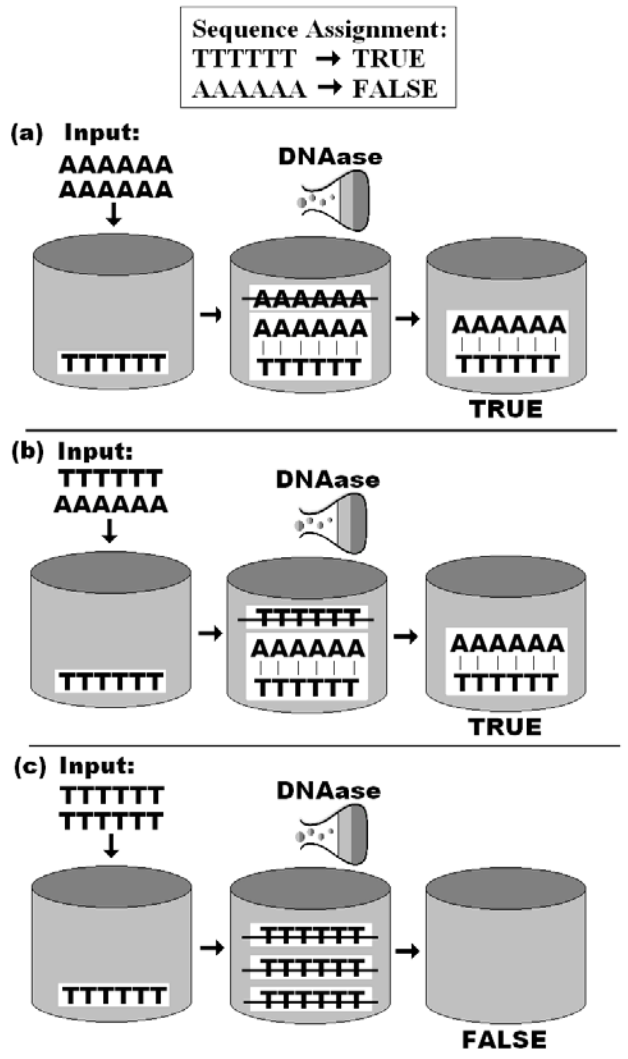

Fig 5: DNA-Based Implementation of the NAND Gate.

2.8 AND, NOR, and XNOR Gates

NOT, XOR, OR, and NAND represent four of the seven most common Boolean logic gates. From these four DNA-based logic gates, one can easily devise a DNA-based representation for all other Boolean logic gates. Consider the three remaining digital logic gates of the seven most common – AND, NOR, and XNOR. The AND gate, which evaluates "true" only when both inputs are "true," is created by applying the NOT gate to the output of the NAND gate. The NOR gate, which evaluates "true" when both inputs are "false," is created by applying the NOT gate to the result of the OR gate. Finally, the XNOR gate, which evaluates "true" when both inputs are the same, is created by applying the NOT gate to one of the inputs, then applying the XOR gate to the result and the other input. Like the preceding gate designs, the presence of a double stranded sequence indicates a "true" evaluation of the gate, while the absence of a double stranded sequence indicates a "false" evaluation of the gate.

3. OBFUSCATING THE LOGIC GATES

Currently, there are a multitude of publications that attempt to address DNA-based logic gates [5-18]. Most require a static sequence as gate input, meaning the circuit only works with a set input sequence that cannot be changed. Obviously, a set sequence can be easily discovered. Thus, current circuits can be reverse engineered by examining the distinct sequences used to represent specific logic gates. Additionally, each logic gate has its own design, so the combination of two different gates, such as the AND and OR, requires significant user interaction. Conversely, the approach presented is consistent across all gate design and allows the user to select any set of complementary sequences to be inputs, not just a static sequence set. Furthermore, the length of the sequences can be easily modified. Six was chosen to achieve a low probability of $1:4^6$ (1:4096) the sequence would randomly align. It is equally attainable to create input sequences of various length, yielding a probability of $1:4^n$, where n is the sequence length.

Finally, gate design is based solely on biochemical and molecular approaches, meaning gates are obfuscated by removing the physical sequence connections present in current DNA-based designs. By moving to biological and environmental circuits, the structure is hidden from the attacker. Because the proposed gates are a function of the chemical reactions among input sequences and base mixtures, meaning the physical blueprint of the circuit cannot simply be observed.

4. NON-BOOLEAN DNA-BASED LOGIC GATES

The DNA-based design to logic gates proposed enables one to break out of the Boolean logic mentality and move towards ternary logic. By design, digital circuits are limited to the Boolean inputs of zero and one. DNA, however, is comprised of four nucleotides, enabling four possible input values, not two. With four inputs, output values are no longer exclusively restricted to "true" or "false." Rather, one can now consider three possible output values: (1) inputs are identical, (2) inputs are complementary, or (3) inputs are different. An output of "identical" implies the two inputs are the same nucleotide base. An output of "complementary" implies the first input base will pair when in the presence of the second. Finally, an output of "different" implies the two inputs are neither the same nor complementary. Table 1 outlines the logical output value for each pair of DNA-based inputs.

Table 1: Logical output value for pairs of nucleotide inputs

INPUT 1	INPUT 2	OUTPUT
A	A	Identical
A	C	Different
A	G	Different
A	T	Complementary
C	A	Different
C	C	Identical
C	G	Complementary
C	T	Different
G	A	Different
G	C	Complementary
G	G	Identical
G	T	Different
T	A	Complementary
T	C	Different
T	G	Different
T	T	Identical

Advancing to a ternary logic output enables more complex logical operations which cannot be easily achieved with current digital truth-functional propositional logic. Consider the basic task of comparing two numbers with binary logic. With traditional binary logic, comparison of two numbers is a two-stage process requiring one to first determine if the first number is greater than the second, and depending on the answer, then determine if first value is equal to the second. Conversely, ternary logic enables a single comparison to indicate one of three outputs – "less than," "equal to," or "greater than."

With ternary systems having the benefit of an additional output stage over their binary counterparts, why did ternary systems fail to thrive? The twentieth century has multiple attempts to design and fabricate digital tri-state logic gates, including the Setun system developed at Moscow State University [19] and the ternac system developed at the State University of New York at Buffalo [20]. While some were successful, solutions were often cost-prohibitive and unreliable when compared to their binary counterparts. DNA-based logic gates are the first proposed solution to naturally produce a ternary logical system.

The benefits of DNA-based logic gates are not limited to the reduction in the number of the gates based on the representation of an additional output state; it also enables circuits to be compressed. The proposed DNA-based logic gate output evaluation is based solely on the presence or absence of the double stranded molecule. Thus, a myriad of input sequences can be condensed into a single gate mixture. For example, a series of OR gates can be integrated into a single DNA-based OR gate. The presence of a single "true" sequence in the mixture will result in the formation of the double stranded molecule regardless of the magnitude of inputs present. Perhaps the benefits of DNA-based logic gate design lies not in mimicking the Boolean logic of their digital counterparts, but in devising a new set of logical operations enabled by the ternary logic structure combined with the DNA-based design.

5. FROM LOGIC GATES TO CIRCUITS

Each DNA-based logic gate is capable of having any set of complementary sequences as valid inputs. Therefore, in order to construct logic circuits, a communication mechanism must be implemented by which a given gate can correctly format the single stranded sequence representing its output as a valid input sequence to the proceeding gate.

The proposed molecular logic gate design evaluation of output is based on the presence or absence of the double stranded sequence. This evaluation determines which single stranded sequence should be supplied to represent the corresponding output state. Rather than constructing this single stranded sequence based on the current gate's input sequence set, the single stranded sequence should be constructed from the proceeding gate's input sequence set. Thus, regardless of what the input sequence set was for the preceding gate, the preceding gate's output is the valid sequence input for the next gate.

Consider the circuit presented in Figure 6, wherein the outputs from an AND gate and an OR gate are combined in an XOR gate. Input sequences for the AND gate are represented by ACCTAG and CTAGGT while inputs for the OR gate are represented by the sequences TTGCAT and ATGCAA, each representing "true" and "false" for their respective gates.

Regardless of whether the outputs from the AND and OR gates are true or not, these sequence combinations cannot be combined in any meaningful manner in the XOR gate, which accepts the sequences CGAACT and AGTTCG for "true" and "false" inputs.

By implementing a "look ahead" mechanism, the outputs from the AND and OR gates can be constructed to be valid inputs for the XOR gate. Thus, the double stranded "true" output from the AND gate is replaced with the sequence CGAACT representing "true" in the proceeding XOR gate. Likewise, the absence of the double stranded sequence output from the OR gate is replaced with the sequence AGTTCG representing "false" in the proceeding XOR gate. These new corresponding sequences represent a valid input sequence combination for input into the XOR gate.

It is worth noting that implementing such "look ahead" mechanism does not require static sequences for gate input. While a gate is idle, the gate could continuously generate a single stranded random nucleotide sequence representing the valid "true" input. When the preceding gate accesses the random sequence to translate its corresponding output, the random sequence is temporarily locked from further changes. Locking the random sequence ensures all inputs are valid because all have been generated from the same random sequence. Once all inputs to the gate have been generated, the lock is removed and random sequences are again continuously generated. A communications mechanism implemented in such a fashion maintains the dynamic nature of gate inputs without requiring continued interaction from the user.

6. FROM CIRCUITS TO A SHIFT REGISTER

A shift register is a primary component of the computational processor that enables information computation at a gate level and then shifts the information to the proceeding gate [21]. Simply moving the data by itself has no computational meaning. A shift register requires the integration of both logic and shifting, thereby creating a complete processing unit that performs serial calculations on an information input stream. It is the integration of logic and shifting that enables information processing in a shift register.

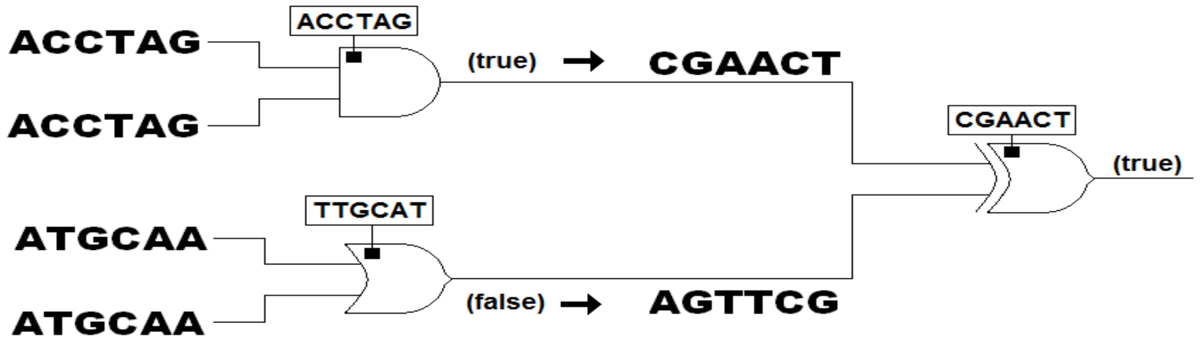

Fig 6: DNA-Based Circuit.

VOL. 3, NO. 5, May 2012 ISSN 2079-8407
Journal of Emerging Trends in Computing and Information Sciences
©2009-2012 CIS Journal. All rights reserved.

http://www.cisjournal.org

The single stranded sequence representing “true” for the given gate is stored locally within the gate.

6.1 Biological Approach to Shifting

The biological process of alternative splicing naturally lends itself to isolating a given segment of information from a data stream for a DNA-based shift register. Alternative splicing is a molecular biology process utilized to produce multiple protein isoforms from a single gene through various sequentially-ordered subset permutations of the set of possible exons [4]. A DNA sequence is subdivided into exons, encoding regions of nucleic acid sequences expressed in translation for protein formation, and introns, non-coding regions of nucleic acid sequences independent of protein formation. Prior to protein formation, intronic regions are discarded while select exonic regions are recombined in sequential order. The protein isoform being created determines which, if any, exonic regions will be discarded. Figure 7 shows three of the possible fourteen splices that can be created from the four exonic DNA: (1) combining the first, third, and last exons, (2) combining all four exonic regions, (3) combining the first, second, and last exons.

It is important to note that any subset of exons is a valid splice permutation only if sequential order is maintained. Permutations not maintaining sequential ordering are not valid splices. For example, combining the second, first, and last exons is invalid because the second exon precedes the first.

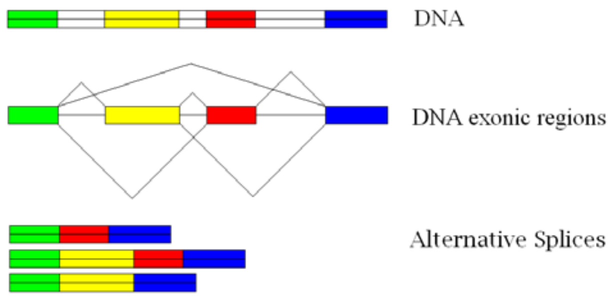

Fig 7: Alternative splicing.

Alternative splicing enables specific exonic regions of a DNA to be selected from the entire sequence. Intronic regions, indicated by colored blocks, are spliced from the sequence. The remaining exonic regions are sequentially concatenated to form valid alternative splices.

Alternative splicing assists DNA computing by enabling a given segment of information to be isolated from a DNA sequence while still maintaining sequential ordering. A shift register must be able to first isolate the segment of information to be processed. By encoding individual elements within the exonic regions of a sequence, one could use alternative splicing to extract the regions desired. Because sequential ordering is maintained, one is assured data segments are read successively, similar to their digital counterparts. Thus, when exonic regions are spliced, they can be inserted into the corresponding logic gate registers for processing.

Alternative splicing enables an assortment of naturally occurring security measures to aid in concealing various input sequences represented in the data stream. First, the input sequences are intermittently spliced with intronic, or meaningless, segments of DNA. Consider three logic inputs represented by the sequences CTAGGT, CTAGGT, and ACCTAG respectively. When hidden within the exonic regions of the DNA sequence shown in Figure 8, it becomes seemingly impossible to decipher the valid input sequences from the stream of nucleotides, especially if one does not know the contents of the input sequences.

ATCCGACTAGGTGATCCTCATCTAGGTCATAAAATATAGACCTAGTGAATT

atccga**CTAGGT**gatcctcat**CTAGGT**cataaaatatag**ACCTAG**tgaatt

Fig 8: Hidden exons and introns.

When hidden within the exonic regions of the DNA sequence, it becomes seemingly impossible to decipher the valid input sequences from the stream of nucleotides. Only when the exonic sequences (bolded uppercase) are distinguished from intronic regions (lowercase) are the values evident.

In addition to concealing input sequences, alternative splicing enables one to selectively choose which inputs to provide to a given gate. For example, if the data stream in Figure 9 represents two inputs for a DNA-based AND gate, one has three valid pairs of inputs from which to select: (1) the first and second exons, (2) the first and third exons, and (3) the second and third exons. Even if an intruder were to determine which regions were exonic, and thus which sequences represent the logic gate inputs, he or she would be left with only a probabilistic guess as to which exonic regions would be selected.

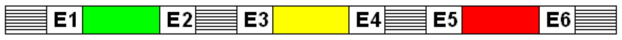

Fig 9: Exon/Intron representation.

Colored inputs are spliced based on the selection of the bounding restriction enzymes added, while meaningless segments of DNA (lined blocks) further obfuscate the input sequences.

6.2 Implementing Alternative Splicing

While alternative splicing occurs naturally and seems ideal in theory to implement the shifting aspect, it is

VOL. 3, NO. 5, May 2012 ISSN 2079-8407
Journal of Emerging Trends in Computing and Information Sciences
©2009-2012 CIS Journal. All rights reserved.

http://www.cisjournal.org

impractical to synthetically coerce splicing to occur at designated locations. To date, the mechanisms by which DNA select exonic regions from a sequence are not fully understood. Inserting foreign DNA sequences into exonic regions could yield unpredictable results. There is no guarantee the intended input sequence will not be spliced out as an intronic region from the data stream.

However, one can mimic the functionality of alternative splicing through the use of restriction enzymes. A restriction enzyme is a small protein sequence that aligns with a specific complementary DNA sequence and cleaves the sequence at such location [4]. In mimicking alternative splicing, input sequences are inserted between predetermined restriction enzyme sequences. Utilizing restriction enzymes in place of intronic regions enables the location of the input sequence to be chemically located and spliced from the data stream, just as it would have been with alternative splicing. Furthermore, intronic segments can be inserted between bounding restriction enzyme sites in order to further obfuscate input sequences within the data stream.

Similar to its counterpart of alternative splicing, restriction enzymes enable a number of permutations to be constructed based on the ordered selection. For example, adding restriction enzymes three, four, five and six in sequential order will splice the second and third DNA sequences for input into the logic gates from the data stream shown in Figure 9.

6.3 Temporary Storage of DNA Sequences

A DNA sequence that is not immediately consumed quickly becomes deteriorated by environmental factors, making the sequence unusable. Therefore, if a sequence is to be stored for later use, a temporary storage mechanism must be provided to preserve the sequence. One method of temporary storage involves inserting the sequence into plasmid vectors. Plasmid vectors are small, circular DNA molecules found in bacteria that enable inserted DNA gene sequences to be transported between various organisms [4]. In order to encompass the sequence, plasmid vectors are spliced open with restriction enzymes, the random oligonucleotide sequence is inserted, and then the vector is reconstructed to its original circular molecule (Figure 10).

In order to retrieve the random sequence from the vector, the process of insertion is reversed. The restriction enzyme cleaves the vector open, the next n bases are sequentially read from the vector, where n is the length of the input sequence, and finally the vector is reconstructed to its original circular molecule. It is important to note that retrieval of the input sequence requires the plasmid vector with inserted sequence, the restriction enzyme used to initially insert the random sequence, and the length of the input sequence.

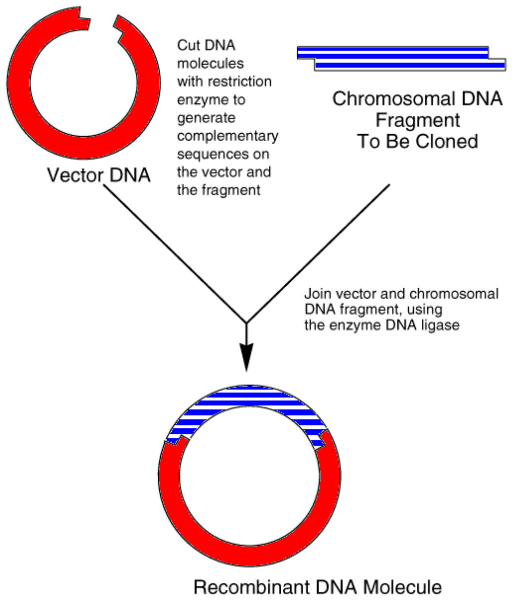
Cut DNA molecules with restriction enzyme to generate complementary sequences on the vector and the fragment
Vector DNA
Chromosomal DNA Fragment To Be Cloned
Join vector and chromosomal DNA fragment, using the enzyme DNA ligase
Recombinant DNA Molecule

Fig 10: Illustration of the insertion of chromosomal DNA into a plasmid vector cut by a restriction enzyme.

Image adapted from [22].

7. CIRCUIT FABRICATION

It is imperative to assess the practicality of fabricating the proposed DNA-based shift register using current and envisioned future technologies. To begin, elements selected for use in the prototype schema are tools and techniques currently employed in biochemical and molecular laboratories on a microscopic level. Requiring such enables one to theoretically be able to construct the circuit here and now, provided funding and resource availability. While the end goal is mass production, invention of prototypes demonstrating successful integration are often cost prohibitive initially.

Fabrication requires the presence of the micro fluidic inputs to the circuit, including the single stranded DNA sequences, fluorescently labeled molecules, DNAase, restriction enzymes and plasmid vectors. Liquids do not dry up; rather, they are consumed by the circuit just as electricity is consumed by their silicon counterparts. This is not considered a limitation of DNA-based circuitry. Regardless of the venue, there is no perpetual circuit in existence. Just as the silicon chip must be replenished with electricity to remain functional, the DNA chip must be replenished with nucleotide solutions, plasmid vectors, and restriction enzymes.

DNA-based circuit design is inherently scalable on an almost endless spectrum. This is enabled by integrating input sequence construction with the evaluation of the preceding logic gate. Essentially such a design eliminates fan-out limitations on circuit size found in digital counterparts. It is envisioned that single stranded inputs are created dynamically just prior to individual gate evaluation, reducing degradation of input sequences, while other micro fluidic resources required are continually pulled as needed from an on-board renewable well.

8. CONCLUSIONS

Traditional silicon-based circuitry is susceptible to security attack as a consequence of the static nature of its design. True tamper-proof security requires circuits be dynamic by nature. We argue that DNA-based logic circuits, when the technology matures, may provide revolutionary solutions to tamper proofing. As DNA paradigms can be developed to represent their digital equivalents, we move closer to the practical applications afforded by dynamic computing.

As a first step, DNA-based methodologies have been developed to mimic existing silicon-based shift registers, demonstrating how information can be parsed through a circuit comprised of DNA–based logic gates. A novel logic gate design based on chemical reactions is presented in which observance of double stranded sequences indicate a truth evaluation. Circuits are obfuscated by removing physical sequence connections, allowing client-specific representative strands for input sequences, altering the input sequence strands over time, and varying the input sequence length. Shifting along the input stream to parse individual inputs is accomplished through simulated alternative splicing of DNA sequences.

ACKNOWLEDGMENTS

C. M. Gearheart would like to thank Hank and Becky Conn for their continued support.

This work was supported in part by NIH-NCRR Grant P20RR16481S1 and NIH-NIEHS Grant P30ES014443. Its contents are solely the responsibility of the authors and do not represent the official views of NCRR, NIEHS, or NIH.

REFERENCES

[1] B. Arazi, "Comprehensive security in constrained environments," presented at the 4th Cyber Security & Information Intelligence Research Workshop (CSIIRW-08), Oak Ridge National Laboratory, 2008.

[2] C. M. Bogard, E. C. Rouchka, and B. Arazi, "DNA Media Storage," Progress in Natural Science, vol. 18, pp. 603-609, 2008.

[3] C. M. Gearheart, B. Arazi, and E. Rouchka, "DNA-Based Random Number Generation in Security Circuitry," Bio Systems, vol. 100, pp. 208-214, 2010.

[4] W. S. Klug and M. R. Cummings, Genetics: A Molecular Approach, First ed. Upper Saddle River, NJ: Pearson Education, Inc, 2003.

[5] M. Amos and P. E. Dunne, "DNA Simulation of Boolean Circuits," presented at the Genetic Programming 1998, San Francisco, CA, 1998.

[6] D. Beaver, "A Universal Molecular Computer," in DNA Based Computers: Proceedings of a DIMACS Workshop vol. 27, R. J. Lipton and E. B. Baum, Eds., ed: Amer Mathematical Society, 1995, pp. 29-36.

[7] D. Boneh, C. Dunworth, R. J. Lipton, and J. Sgall, "On the Computational Power of DNA," Discrete Applied Mathematics, vol. 71, pp. 79-94, 1996.

[8] W.-L. Chang, M. Guo, and M. S.-H. Ho, "Fast Parallel Molecular Algorithms for DNA-Based Computation: Factoring Integers," IEEE Transactions on NanoBioScience, vol. 4, pp. 149-163, 2005.

[9] W.-L. Chang, M. Ho, and M. Guo, "Molecular Solutions for the Subset-Sum Problem on DNA-Based Supercomputing," Biosystems, vol. 73, pp. 117-130, 2004.

[10] A. Goel and M. Ibrahimi, "Renewable, Time-Responsive DNA Logic Gates for Scalable Digital Circuits," in DNA Computing and Molecular Programming. vol. 5877, ed: Springer Berlin / Heidelberg, 2009, pp. 67-77.

[11] F. Guarnieri, M. Fliss, and C. Bancroft, "Making DNA Add," Science, vol. 273, pp. 220-223, 1996.

[12] I. Kawamata, F. Tanaka, and M. Hagiya, "Automatic Design of DNA Logic Gates Based on Kinetic Simulation," in DNA Computing and Molecular Programming. vol. 5877, ed: Springer Berlin / Heidelberg, 2009, pp. 88-96.

[13] M. Ogihara and A. Ray, "Simulating Boolean Circuits on a DNA Computer," presented at the 1st Annual International Conference on Computational Molecular Biology, Santa Fe, NM, 1997.

[14] M. Ogihara and A. Ray, "Molecular Computation: DNA Computing on a Chip," Nature, vol. 403, pp. 143-144, 2000.

[15] L. Qian and E. Winfree, "A Simple DNA Gate Motif for Synthesizing Large-Scale Circuits," in DNA

Computing. vol. 5347, ed: Springer Berlin / Heidelberg, 2009, pp. 70-89.

[16] Y. Sakai, Y. Mawatari, K. Yamasaki, and K.-i. Shohda, "Construction of AND Gate for RTRACS with the Capacity of Extension to NAND Gate," in DNA Computing and Molecular Programming. vol. 5877, ed: Springer Berlin / Heidelberg, 2009, pp. 137-143.

[17] G. Seelig, D. Soloveichik, D. Y. Zhang, and E. Winfree, "Enzyme-Free Nucleic Acid Logic Circuits," Science, vol. 314, pp. 1585-1588, Dec 8 2006.

[18] R. Weiss and S. Basu, "The Device Physics of Cellular Logic Gates," presented at the 8th International Symposium on High-Performance Computer Architecture: The First Workshop on Non-Silicon Computing, Cambridge, MA, 2002.

[19] G. Trogemann and W. Ernst, Computing in Russia: The History of Computer Devices and Information Technology reveals. Braunschweig/Wiesbaden: GWV-Vieweg, 2001.

[20] G. Frieder, "Ternary computers part I: motivation for ternary computers," presented at the 5th Annual Workshop on Microprogramming, Urbana, IL, 1972.

[21] J. G. Brookshear, Computer Science: An Overview, Ninth ed.: Addison Wesley, 2006.

[22] U.S. Department of Energy Genome Research Projects, "PRIMER: Genomics and Its Impact on Science and Society: The Human Genome Project and Beyond," ed: Oak Ridge National Laboratory, 2008.