Logisim Operon Circuits

Aman Chandra Kaushik

Aman Chandra Kaushik (M.Sc. Bioinformatics)
Department of Bioinformatics, University Institute of Engineering & Technology
Chhatrapati Shahu Ji Maharaj University, Kanpur-208024, Uttar Pradesh, India
Email: amanbioinfo@gmail.com
Phone: +91-8924003818

Abstract: Creation of Logism Operon circuits for bacterial cells that not only perform logic functions, but also remember the results, which are encoded in the cell's DNA and passed on for dozens of generations. This circuits is coded in JAVA Language. "Almost all of the previous work in synthetic biology that we're aware of has either focused on logic components and logical circuits or on memory modules that just encode memory. We think complex computation will involve combining both logic and memory, and that's why we built this particular framework for designing circuits Life Science.

In one circuit described in the paper, one DNA sequences have three genes called Repressor, Operator (terminators) are interposed between the promoter (DNA Polymerase) and the output Proteins (Beta glactosidase, permease, Transacetylase in this case). Each of these terminators inhibits the transcription and Translation of the output gene and can be flipped by a different Promoter enzyme (DNA Polymerase), making the terminator inactive.

Key words: Logical Gate, JAVA, Circuits, Operon, AND, OR, NOT, NOR, NAND, XOR, XNOR, Operon Logisim Circuits.

1. Introduction

Molecular Logic Gates Coding in Java Language and circuits are designed in Logisim

The underlying molecular automaton is a Boolean network of logic gates that are made of allosterically modulated deoxyribozymes, and input and output are given by oligonucleotides. The logic gates are based on molecular beacons introduced by S. Tyagi and F. Kramer (1996), which are oligonucleotide probes able to report the presence of specific DNA in homogenous solutions. Molecular beacons are hairpin-shaped molecules with a quencher (R) at the 3′ end and a fluorosphore (F) at the 5′ end. These molecules are non-fluorescent as the loop keeps the quencher close to the fluorophore. However, when the oligonucleotide probe sequence in the loop hybridizes to a target DNA molecule forming a rigid double helix, the quencher is separated from the fluorophore restoring fluorescence. Deoxyribozymes are nucleic acid catalysts made of deoxyribonucleic acid. Examples of deoxyribozymes are phosphodiesterases, which cleave other oligonucleotides with shorter products as output, and ligases, which combine two oligonucleotides into a larger product. The molecular gates are based on phosphodiesterase E6, which cleaves an input fluorogenic oligonucleotide upon binding and releases the cleft oligonucleotides as output. This produces an increase in fluorescence by fluorescein (F) as the quencher (R) is separated.

The YES gate is comprised of a deoxyribozyme E6 module and a hairpin module complementary to input oligonucleotide x. If the input oligonucleotide hybridizes to the hairpin module, the gate enters the active state, increasing fluorescence.

The NOT gate is constructed by extending the non-conserved loop of the E6 core with a hairpin structure that is complementary to input oligonucleotide x. If the input oligonucleotide hybridizes to the hairpin module, the gate enters the inactive state not increasing fluorescence.

The AND gate is obtained from the E6 core module by extending both ends with hairpin modules, one complementary to input oligonucleotide x and the other complementary to input oligonucleotide y.

The AND-NOT gate realizes the Boolean function (x, y) _→ xy and is derived from the E6 core module by extending one end with a hairpin module.

Using this design strategy, the researchers can create all two or more input logic gates and implement sequential logic systems for Life Science. It's really easy to swap things in and out," says Lu, who is also a member of MIT's Synthetic Biology Center. "If you start off with a standard parts library, you can use a one-step reaction to assemble any kind of function that you want simply input 1 or 0 and get output according to circuits."
.

Applications

1. Such operon logisim circuits could also be used to create a type of circuit known as a digital-to-analog converter. This kind of circuit takes digital inputs — for example, the presence or absence of single chemicals or molecular pathway— and converts them to an analog output, which can be a range of values, such as continuous levels of gene expression.

2. That type of operon circuit could offer better control over the production of cells or organs or whole body of the System generate biofuels, drugs or other useful compounds. Instead of creating circuits that are always on, or using promoters that need continuous inputs to control their output levels, scientists could transiently program the circuit to produce at a certain level. The cells and their progeny would always remember that level, without needing any more information it is user friendly software put 0 or 1 and get results.

3. Used as environmental sensors, such circuits could also provide very precise long-term memory. Java provide JVM (Java Virtual Machine), it provide portable environment for operon logistic circuits.

4. This platform could also allow scientists to more accurately control the fate of stem cells as well as germinal cells they develop into other cell types. Any bio molecular and chemical pathway type of task you perform within a time.

5. Logic and memory are essential functions of circuits that generate complex, state-dependent responses. Here we describe a strategy for efficiently assembling synthetic genetic circuits that use Operon System to implement Boolean logic functions with stable DNA-encoded memory of events. Application of this strategy allowed us to create all two-input Boolean logic functions in living Bacteria cells without requiring cascades comprising multiple logic gates like alignment of two sequences in term of Bioinformatics.

2. Material and Method

1- Software Requirements -

- Logisim
- JAVA

METHODOLOGY

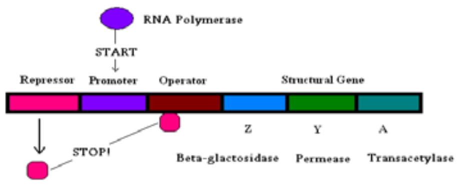


Fig1- Biological Operon Model.

3. Results & Discussion

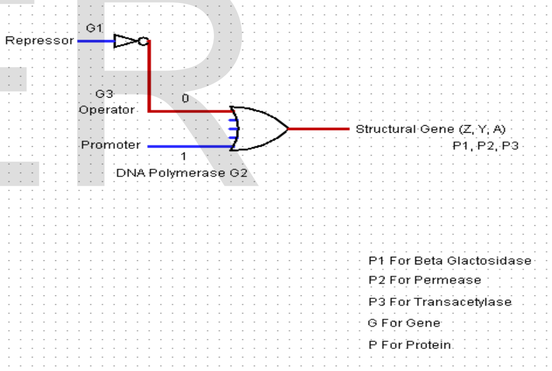


Fig2- Operon Logisim Circuits.

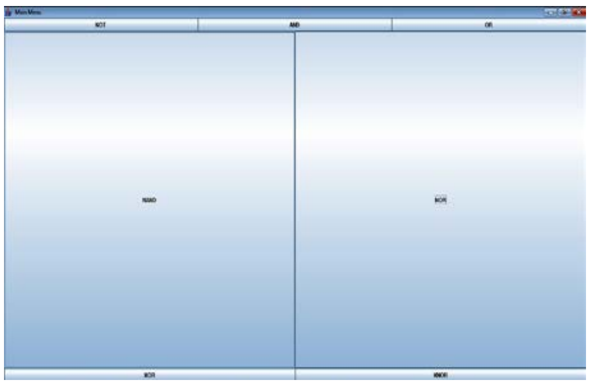
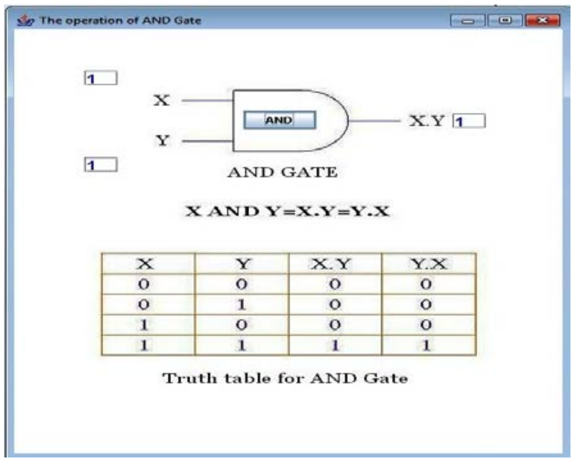


Fig3- Dashboard of Operon Logisim Circuits.



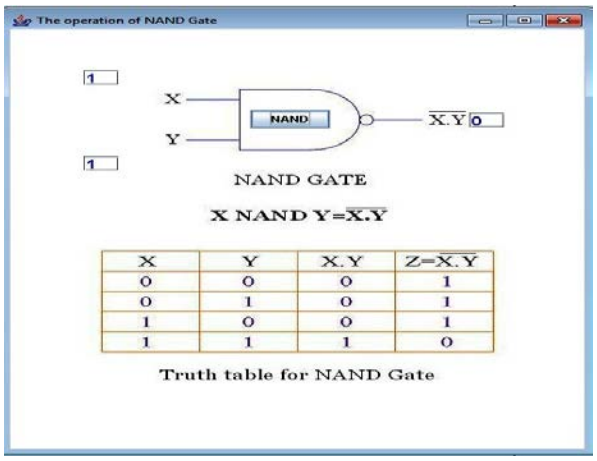
AND GATE

X AND Y=X.Y=Y.X

X	Y	X.Y	Y.X
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

Truth table for AND Gate

Fig4- AND GATE of Operon Logisim Circuits.



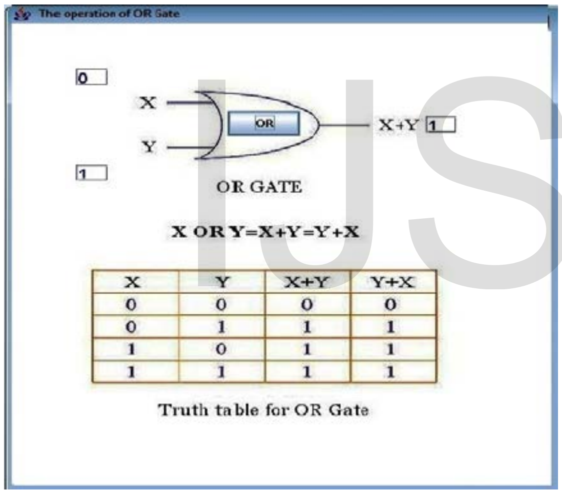
NAND GATE

X NAND Y=$\overline{X.Y}$

X	Y	X.Y	Z=$\overline{X.Y}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Truth table for NAND Gate

Fig7-NAND GATE of Operon Logisim Circuit.



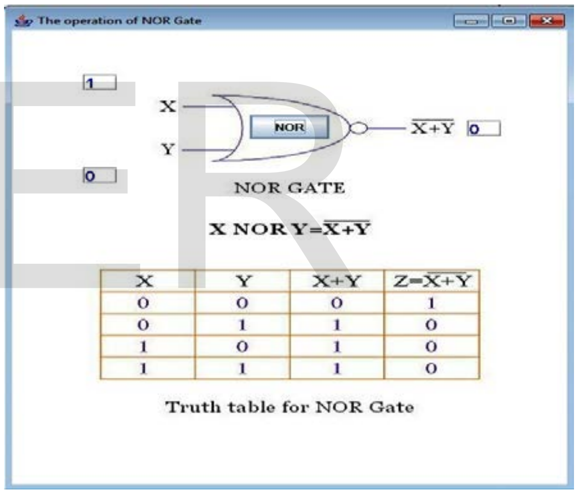
OR GATE

X OR Y=X+Y=Y+X

X	Y	X+Y	Y+X
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

Truth table for OR Gate

Fig5- OR GATE of Operon Logisim Circuits.



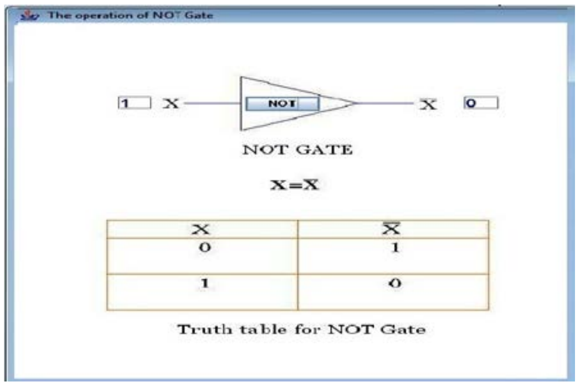
NOR GATE

X NOR Y=$\overline{X+Y}$

X	Y	X+Y	Z=$\overline{X+Y}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Truth table for NOR Gate

Fig8- NOR GATE of Operon Logisim Circuits.



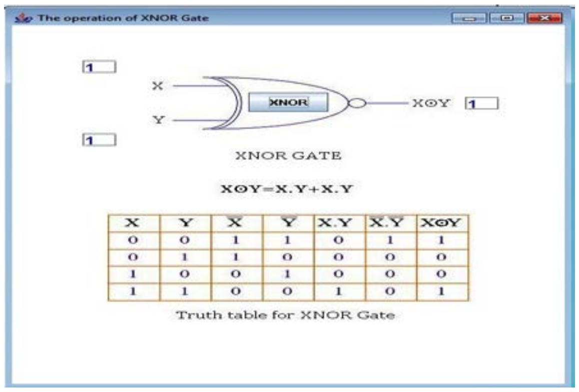
NOT GATE

X=$\overline{X}$

X	$\overline{X}$
0	1
1	0

Truth table for NOT Gate

Fig6- NOT GATE of Operon Logisim Circuits.



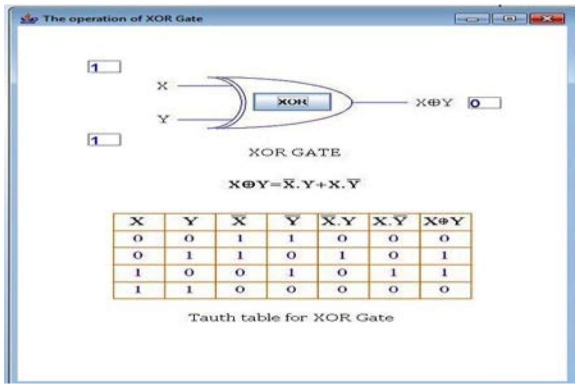
XNOR GATE

X⊙Y=X.Y+X.Y

X	Y	$\overline{X}$	$\overline{Y}$	X.Y	$\overline{X}.\overline{Y}$	X⊙Y
0	0	1	1	0	1	1
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	1	0	0	1	0	1

Truth table for XNOR Gate

Fig9- XNOR GATE of Operon Logisim Circuit.



$X \oplus Y = \overline{X}.Y + X.\overline{Y}$

X	Y	$\overline{X}$	$\overline{Y}$	$\overline{X}.Y$	$X.\overline{Y}$	$X \oplus Y$
0	0	1	1	0	0	0
0	1	1	0	1	0	1
1	0	0	1	0	1	1
1	1	0	0	0	0	0

Tauth table for XOR Gate

Fig10- XOR GATE of Operon Logisim Circuits.

4. References:

[1] Gheorghe Paun, *Computing with Bio-Molecules (Theory and Experiments)*, Springer-Verlag Singapore Pte. Ltd, May 1998.

[2] Leonard M. Adleman, *Computing with DNA*, Scientific American, August 1998.

[3] Masahito Yamamoto, Nobuo Matsuura, Toshikazu Shiba, Yumi Kawazoe and Azuma Ohuchi, "Solution of Shortest Path Problem by Concentration Control," DNA based computers 5, DIMACS series in Discrete Mathematics and Theoretical Computer Volume 54, 2000, 101-110.

[4] Gheorghe Paun, Rozenberg and Salomaa, *DNA computing, New computing paradigms*, Springer-Verlag, Berlin,1998.

[5] L.M Adleman, "Molecular computation of solutions to combina- torial problems," Science, 266, 1994, 1021-1024.

[6] Martyn Amos, Gheorghe Paun, Grzezorg, Rozenberg and Arto Salomaa, "Topics in the theory of DNA computing," Theoretical computer science, 287, 2000, 3-38.

[7] Yachun Liu, Jin Xu, Linqiang and Shiying Wang, "DNA solution to the Graph-Coloring Problem," J. Chem. Inf, Comput. Sci. 42, 2002, 524-528.

[8] Lloyd Smith, Hall and Mark, "Practical feature subset selection for machine learning," In Proc Australian Computer Science Con- ference, Perth, Australia, 1998, 181-191.

[9] L. Adleman, P. Rothemund, S. Roweis and E. Winfree, "On Apply- ing Molecular Computation To The Data Encryption Standard," 2nd DIMACS workshop on DNA based computers, Princeton, 1996, 28-48.

[10] Gupta, GauraV, Mehra, Nipun and ChakraVerty, *DNA Comput- ing*, The Indian Programmer, June 12, 2001.

[11] V. Shekhar, Ch.V.Ramana Murthy, P.V. Gopalacharyulu and G.P.Rajasekhar, *DNA solution to the shortest path problem*, ICADS-2002, Dec22-24, Dept of Mathematics, IIT Kharagpur.