**Seminar Report on**

**CELLO: A Programming Language For Living Cell**

**By Christopher Voigt**, Associate Professor of Bio Logical Department at Massachusetts Institute of Technology.

**Abstract:**

Professor starts with introduction about the programming language for the cell that combines the hardware descriptive language framework and biomedicine to give cellular logic known as CELLO programming. Many permutation and combination have been applied to the complier to give the most accurate results in least amount of time. It gives the complete synthesis of Verilog code written. The basics of cell development are explained, for example how the whole circuit is build with NOR gate and how it is used to give the best area and power optimization. The Verilog code can be used for input to output sequence. The DNA sequence, which is automated at the output compatible with all major labs, can be given for analysis. The most of it in on cloud, hence can be accessed through Internet anywhere anytime.

**Introduction:**

Professor started with discussion on cellular decision-making and how to choose best cell selection in description. The synthetic circuit application was followed later showing that 3-circuit plant and 9 circuits plant, which could describe the behavior in detail. Hence to derive the result as the one with more complex circuit delivers more accurate behavior. This then poses the challenge to make the design of the most accurate and most efficient design that can deliver the required behavior.

The challenge faced is the 3 – 4 gene circuits are still multi-year projects by the specialist. This is further divided in three categories. The first being Design, in which number of well characterized regulatory parts, toxicity and genetic environment are crucial design parts. The second is construction of Encoded DNA fragments consisting of many parts and final is debugging which relied on highly-expressed fluorescent proteins.

The most accurate and simple way to address this is Genetic Programming. Genetic Programming is fast and service driven motive to help automate the large part of this process.

**Methodology:**

Genetic Programming:

Software

Cello (Verilog)

Genetic Circuit

(Gate Level)

Convert to DNA Sequence

Fig 1 – Flow Diagram of programming

The diagram explains the cycle to generate the DNA sequence. The software written in cello, which is based on Verilog, gives the genetic circuit on gate level. This then generates the DNA sequence, which represents the biological elements.

Verilog:

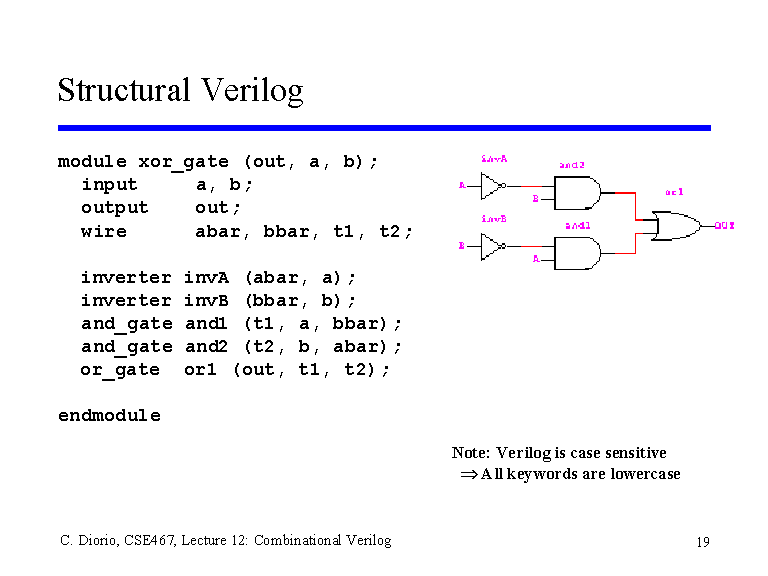
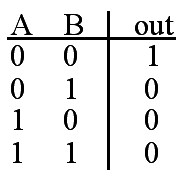


Fig 2 – Structural Verilog programming

Hardware description languages such as Verilog differ from software [programming languages](http://en.wikipedia.org/wiki/Programming_language) because they include ways of describing the propagation time and signal strengths (sensitivity). The designers of Verilog wanted a language with syntax similar to the [C programming language](http://en.wikipedia.org/wiki/C_(programming_language)), which was already widely used in engineering software development. Like C, Verilog is [case-sensitive](http://en.wikipedia.org/wiki/Case-sensitive) and has a basic [preprocessor](http://en.wikipedia.org/wiki/Preprocessor).

NOR Gate:

Boolean logic is completed by the universal NOR gate. The truth table of the gate explains the operation of the gate. This gate can be used to realize any logic. This gate requires minimum area in design of the same.



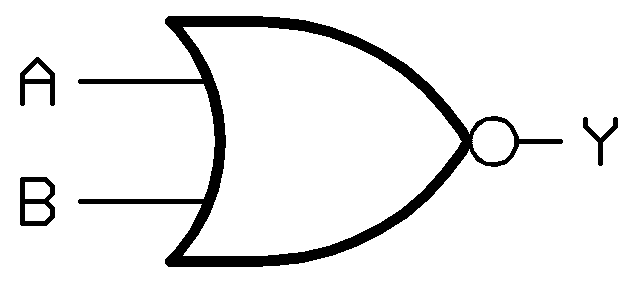


Fig 3 – Truth Table and Logic Diagram

DNA Sequence:

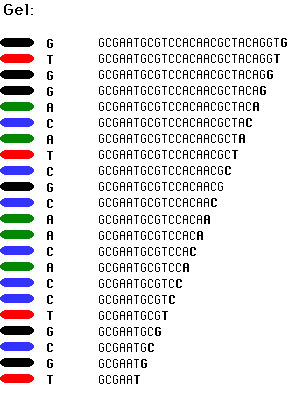
DNA sequencing is the process of determining the precise order of [nucleotides](http://en.wikipedia.org/wiki/Nucleotides) within a [DNA](http://en.wikipedia.org/wiki/DNA) molecule. It includes any method or technology that is used to determine the order of the four bases—[adenine](http://en.wikipedia.org/wiki/Adenine), [guanine](http://en.wikipedia.org/wiki/Guanine), [cytosine](http://en.wikipedia.org/wiki/Cytosine), and [thymine](http://en.wikipedia.org/wiki/Thymine)—in a strand of DNA. The advent of rapid DNA sequencing methods has greatly accelerated biological and medical research and discovery.

Fig 4 – Example of DNA sequencing

**Key Results:**

Gates based on Repressor and Promoter:

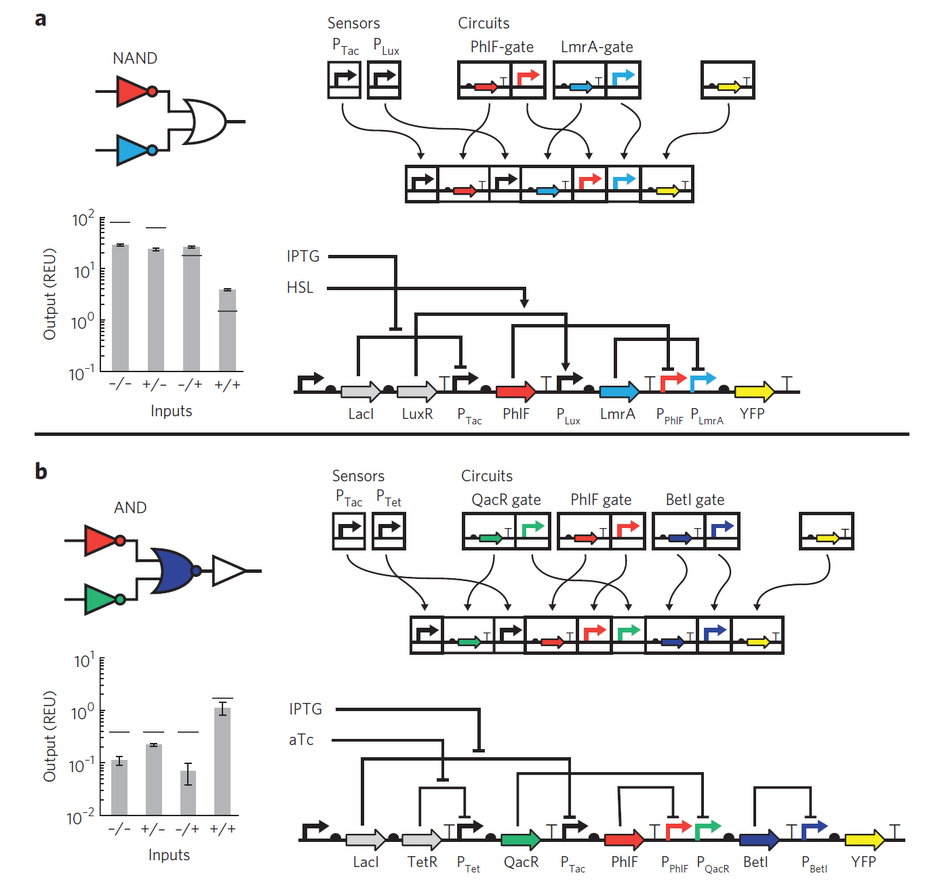
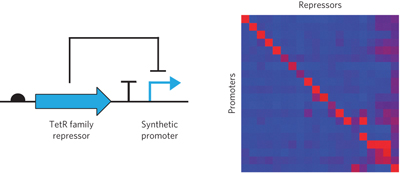


Fig 6 – Non Interfering Gates example

Fig 5 – NAND and AND on repressor and Promoter

* Non-Interfering gate – Only one repressor interacts with promoters.
* Connecting gates – Form a good pair or bad pair using the response in the above diagram so that there is no crosstalk.
* Transcriptional Insulation – requires strong stops of RNAP flow.
* Insulator of Promoter context – measured in context that is changed when building a circuit.
* Gate selection and optimization – Better complier design.

**Critique:**

By implementing the Cello, the hardware descriptive language design of complex biological elements is simple and effective. One more advantage to it is that it can be synthesized on cloud and all the work is stored against your online account. This makes it accessible and can be worked or modified at any given time. The more libraries that will be developed will make it more advanced. The DNA sequence that is generated in an automated process will speed up the development even further.