A Top-down Approach to Genetic Circuit Synthesis and Optimized Technology Mapping

Baig, Hasan; Madsen, Jan

Published in:
Proceedings of 9th International Workshop on Bio-Design Automation

Publication date:
2017

Document Version
Peer reviewed version

A Top-down Approach to Genetic Circuit Synthesis and Optimized Technology Mapping

Hasan Baig and Jan Madsen
Department of Applied Mathematics and Computer Science
Technical University of Denmark
{haba, jama}@dtu.dk

1. INTRODUCTION

Genetic logic circuits are becoming popular as an emerging field of technology. They are composed of genetic parts of DNA and work inside a living cell to perform a dedicated boolean function triggered by the presence or absence of certain proteins or other species.

In this work, we introduce a top-down approach to synthesize genetic logic circuit. This approach is based on translating high-level description of genetic circuit (in the form of boolean function) to its low-level representation in the form of SBOL [1] notation. This approach is implemented in the *Gene*tic *Tech*nology mapping tool, *GeneTech*. It takes the Boolean expression of a genetic circuit as input, and then first optimize it. It then synthesizes the optimized Boolean expression into NOR-NOT form in order to construct the circuit using the real NOR/NOT gates available in the genetic gates library [2]. In the end, *GeneTech* performs technology mapping to generate all the feasible circuits, with different genetic gates, to achieve the desired logical behavior.

There are some existing tools which supports technology mapping of genetic circuits including Cello [2] and iBioSim [3]. *GeneTech* differs from these tools by generating all feasible genetic circuits from a Boolean expression. This work is originally inspired from the processes of optimization and technology mapping of electronic circuits in the *electronic design automation* (EDA) industry. In EDA, the combinatorial circuit optimization is always required to implement the circuit with the minimum number of logic gates [4]. This area-efficient implementation of digital circuits not only helps reducing the size of electronic devices but also avoid wasting power and redundant resources.

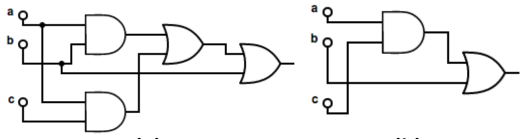


Figure 1. Digital circuit of the expression *ab+b+ac*.
(a) Original circuit. (b) Optimized circuit having two gates.

In order to get the insight of logic optimization, consider the digital circuit for the Boolean expression, *ab + b + ac*, shown in Figure 1(a). In this figure, the circuit consists of four logic gates. After running the optimization algorithm, the number of gates in the circuit reduces down to two while preserving the original functionality, as illustrated in Figure 1(b).

This optimization of digital electronic circuits seems simple and straight forward. However, the optimization and technology mapping of genetic circuits is not similar to electronic circuits. This is because the input and output quantities of electronic circuits are the same i.e. voltage, and therefore the electronic gates can easily be cascaded together. On the contrary, the input and output quantities of genetic gates are different, and therefore the signal matching has to be considered while mapping genetic gates on the circuit. This makes it very challenging to integrate genetic logic gates to construct complex genetic circuits. Similar to the above process of optimizing digital logic in electronic circuits, we want to avoid having redundant logic in genetic circuits as well.

2. METHODOLOGY

Two different ways to represent the same boolean logic or digital circuit are the *minterm* and *maxterm* canonical forms. Minterms are also called the *products* because the variables (or literals) in the Boolean expressions are represented as the *logical AND*. Maxterms are referred to as *sums* because the variables (or literals) are represented as the *logical OR*. Therefore, the same Boolean function can either be expressed as the *sum of products/minterms* (SOP) or the *product of sums/maxterms* (POS), as shown in equation (1). In this example, the left-hand side represents the SOP form and the right-hand side represents its equivalent POS form.

$$\mathrm{ab} + \mathrm{b} + \mathrm{ac} = (\mathrm{a} + \mathrm{b} + \mathrm{c})(\mathrm{a} + \mathrm{b} + \overline{\mathrm{c}})(\overline{\mathrm{a}} + \mathrm{b} + \mathrm{c}) \qquad (1)$$

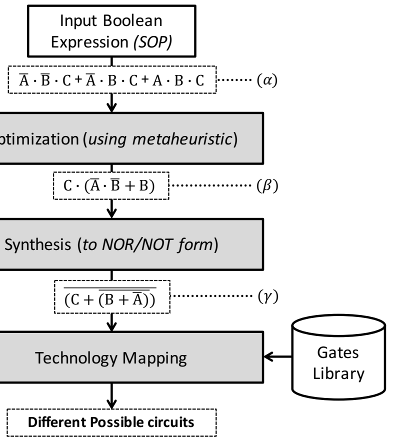
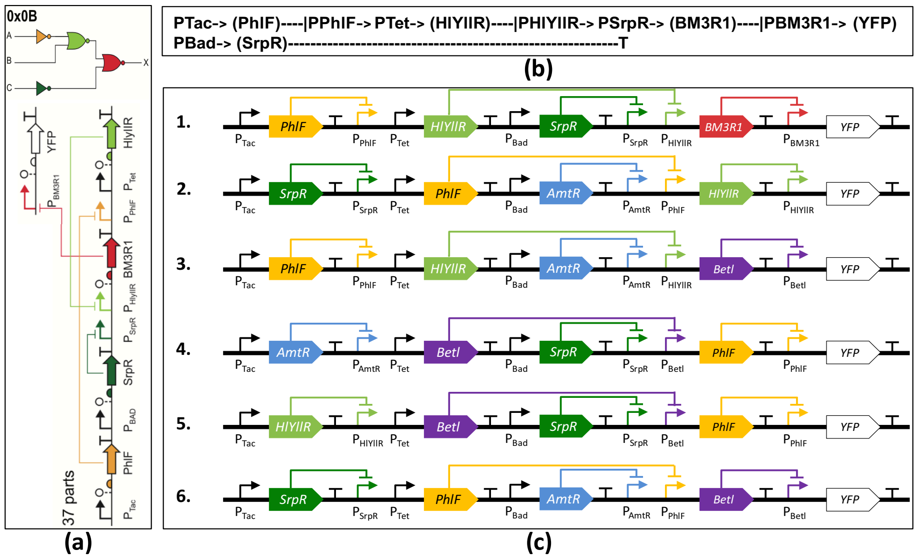

Input Boolean Expression (SOP)
$\overline{A}\cdot\overline{B}\cdot C + \overline{A}\cdot B\cdot C + A\cdot B\cdot C$
(α)
Optimization (using metaheuristic)
$C\cdot(\overline{A}\cdot\overline{B} + B)$
(β)
Synthesis (to NOR/NOT form)
$\overline{(C + \overline{(B + \overline{A})})}$
(γ)
Technology Mapping
Gates Library
Different Possible circuits

Figure 2. The technology mapping flow of *GeneTech*.

The flow of genetic technology mapping in *GeneTech* is shown in Figure 2. It takes the raw Boolean expression in the SOP form and then first optimize it using the *simulated annealing* (SA) [5] optimization algorithm. The goal of optimization at this step is to reduce the number of variables (or literals) in the expression while keeping the output logic the same. Reducing the number of literals in the Boolean expression results in the reduction of logic components required to obtain the desired logic.

To construct real genetic circuits, *GeneTech* uses the gates library from [2], which consists of genetic gates in the form of NOR and NOT functions. Therefore, to map the genetic gates on the Boolean expression, it is necessary to bring it into NOR/NOT form. Hence, when the Boolean expression is optimized, it then goes to a process of synthesis, as shown in Figure 2. Once the Boolean expression is available in NOR/NOT form, a mapping



```
PTac-> (PhlF)----|PPhlF-> PTet-> (HlYIIR)----|PHlYIIR-> PSrpR-> (BM3R1)----|PBM3R1-> (YFP)
PBad-> (SrpR)---------------------------------------------------------T
```

Figure 3. Experimental results of *GeneTech* for 0x0B [2]. (a) Circuit schematic and SBOL representation of 0x0B shown in [2]. (b) Circuit representation generated by *GeneTech*. (c) The SBOL notations of all possible circuits generated by *GeneTech* to achieve the same logic of the circuit 0x0B.

algorithm of *GeneTech* checks for the logic components in the gates library and find all feasible genetic circuits.

We have extracted the genetic gates by analyzing the SBOL notations of all the circuits shown in [2] and organized them in separate lists of genetic NOT and NOR gates. The algorithm is based on a deterministic depth-first search approach and maps the genetic gates on the deepest most elements in the expression. For example, in the expression (γ) shown in Figure 2, all NOT gates from the library which are compatible with $\bar{A}$ are selected first. Then the mapping algorithm checks for any available NOR gate with one of the input as B and the other input matching to the outputs of any of the NOT gates selected previously for $\bar{A}$. If any such NOR gate is found, the algorithm then search for another NOR gate with the inputs compatible to the output of first NOR gate and the output of any of the NOT gates available for $\bar{C}$. In this way, all the compatible components are used to achieve the same boolean functionality with different possible genetic circuits.

While constructing genetic circuits, *GeneTech*, avoid using those genetic gates which generate the same output protein. This is to make sure that the signals of the gates do not interfere with each other.

3. EXPERIMENTATION AND RESULTS

We performed experiments on the genetic circuits shown in [2]. Due to space limitation, the results of one circuit, 0x0B, are shown in Figure 3. Figure 3 (a) shows the schematic and SBOL representation of the circuit 0x0B obtained directly from [2].

The input expression of the circuit 0x0B is obtained from the truth table given in [2], which is shown as expression (α) in Figure 2. After optimization, it is reduced to the expression (β) shown in Figure 2. Afterwards the synthesis is performed to bring this expression into NOR/NOT form shown as (γ) in Figure 2. Figure 3(b) shows the multi-line text string format (similar to SBOL notation) which is used by *GeneTech* to represent the structure of a generated circuit. In Figure 3(b), promoters are shown with the symbol "->", the proteins are represented by round braces "( )", and the repression is indicated by the symbol "----|" or "T". Figure 3(b) indicates that the P_{Tac} promoter generates a protein PhlF which in turn represses the output promoter P_{PhlF}. The promoter P_{PhlF} together with the promoter P_{Tet} generate the protein HlYIlR, which represses the output promoter P_{HlYIIR}. In the second line, promoter P_{Bad} generates the protein SrpR which supresses its corresponding output promoter P_{SrpR}. This promoter P_{SrpR} together with the promoter P_{HlYIIR} generate the protein BM3R1, which represses the activity of the output promoter P_{BM3R1}. The promoter P_{BM3R1} is used to produce the output indicator, the *yellow fluorescent protein* (YFP). The SBOL notation of Figure 3(b) is shown as notation 1 in Figure 3(c). Figure 3(c) shows the SBOL representations of the circuits generated by *GeneTech* tool. This figure shows that the *GeneTech* tool, beside suggesting the solution given in [2] (Figure 3(c)-1), it also finds all other possible circuits to achieve the same logic function using other genetic components available in the gates library [2].

4. SUMMARY

In Cello [2], circuits are constructed by selecting the appropriate genetic components, based on matching their threshold levels, through non-deterministic search using simulated annealing algorithm. Therefore, for every compilation of the same code in Cello, the generated circuit may contain the same or different genetic components. On the contrary, *GeneTech* gives the number of possible solutions to achieve the same logic. With the correct set of parameters, the threshold levels of these circuits can then be obtained using D-VASim [6] and then can be verified in the laboratory. More design constraints can be added in *GeneTech* to make sure that the circuits generated by this tool would work in the laboratory. Furthermore, *GeneTech*, at its current state, supports technology mapping of genetic gates based on repression. In future, it will be upgraded to support genetic gates based on other technologies.

5. REFERENCES

[1] B. Bartley, *et al.*, "Synthetic Biology Open Language (SBOL) Version 2.0.0.," *J. Integr. Bioinform.*, 2015.

[2] A. A K. Nielsen, *et al.*, "Genetic circuit design automation", *Science*, vol. 352, no. 6281, pp. 7341, 2016.

[3] N. Roehner and C. J. Myers, "Directed Acyclic Graph-Based Technology Mapping of Genetic Circuit Models," *ACS Synth. Biol*, 2014.

[4] Giovanni De Micheli, "Synthesis and Optimization of Digital Circuits," McGraw-Hill Series in Elec. and Comp. Engg., 1994.

[5] S. Kirkpatrick, *et al.*, "Optimization by Simulated Annealing", *Science (80)*, vol. 220, no. 4598, pp. 671–680, 1983.

[6] H. Baig and J. Madsen, "Simulation Approach for Timing Analysis of Genetic Logic Circuits," *ACS Synth. Biol.*, Feb. 2017.