Simulation Approach for Timing Analysis of Genetic Logic Circuits

Baig, Hasan; Madsen, Jan

Published in:
A C S Synthetic Biology

Link to article, DOI:
10.1021/acssynbio.6b00296

Publication date:
2017

Document Version
Peer reviewed version

Link back to DTU Orbit

Citation (APA):
Baig, H., & Madsen, J. (2017). Simulation Approach for Timing Analysis of Genetic Logic Circuits. A C S Synthetic Biology, 6(7), 1169-1179. DOI: 10.1021/acssynbio.6b00296

ACS Synthetic Biology

Research Article

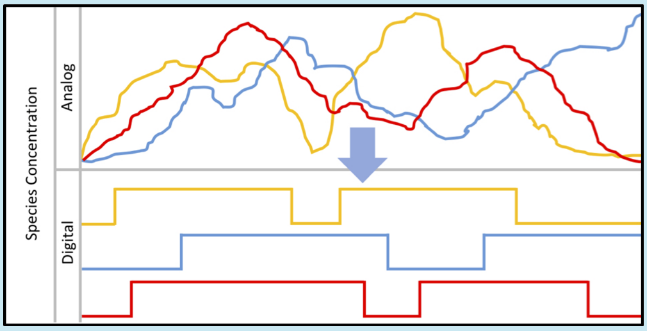
pubs.acs.org/synthbio

Simulation Approach for Timing Analysis of Genetic Logic Circuits

Hasan Baig* and Jan Madsen*

Department of Applied Mathematics and Computer Science, Technical University of Denmark, 2800 Kongens Lyngby, Denmark

S *Supporting Information*



ABSTRACT: Constructing genetic logic circuits is an application of synthetic biology in which parts of the DNA of a living cell are engineered to perform a dedicated Boolean function triggered by an appropriate concentration of certain proteins or by different genetic components. These logic circuits work in a manner similar to electronic logic circuits, but they are much more stochastic and hence much harder to characterize. In this article, we introduce an approach to analyze the threshold value and timing of genetic logic circuits. We show how this approach can be used to analyze the timing behavior of single and cascaded genetic logic circuits. We further analyze the timing sensitivity of circuits by varying the degradation rates and concentrations. Our approach can be used not only to characterize the timing behavior but also to analyze the timing constraints of cascaded genetic logic circuits, a capability that we believe will be important for design automation in synthetic biology.

KEYWORDS: *timing analysis, stochastic simulation, SBML, genetic logic circuits, synthetic biology, virtual instrumentation, threshold value analysis*

One of the ultimate goals of synthetic biology[1] is the engineering of dedicated cell behavior by introducing new sequences of DNA (encoding genetic circuits) into a cell's DNA. A genetic circuit represents a gene regulatory network that is triggered by a combination of external input signals, such as chemicals, proteins, light, and temperature, to emit signals for controlling the expression of other genes or metabolic pathways accordingly.

The ability to engineer living cells has created completely new ways of manufacturing food, drugs, biofuel, and materials.[4−9] It is possible to produce not only new materials, such as reprogramming bacteria to produce spider silk,[2] but also active components for therapeutics, such as the "living pill",[3] a bacteria reprogrammed to automatically sense the presence of two disease-causing molecules in the body and respond by triggering the production of two other molecules that treat the disease.

The later is an example of how the design and implementation of logic functions, in this case a simple AND gate, can be used to design synthetic genetic circuits for many different applications including cancer destruction, biosensors, cell therapy, regenerative medicine,[4−9] and so forth. The Boolean behavior of genetic logic circuits is similar to that of electronic logic circuits. In digital electronics, circuits are made up of digital logic gates, which themselves are composed of transistors.[10,11] However, genetic logic circuits are composed of biological components of DNA,[12] including promoters, ribosome binding sites (RBSs), terminators, and gene coding regions. The genetic circuits are usually meant to produce output protein based on the presence of input proteins. For example, Figure 1 shows the diagram of a genetic AND gate circuit.[13]

Genetic circuits are assembled either from a standard library of well-defined genetic gates[19] or from the available library of basic biological components, for instance, BioBricks.[14] The behavior of these circuits is usually first validated through *in silico* (in computer) analysis either by solving reaction kinetics using ordinary differential equations or by performing stochastic simulations, with the aim of reducing the number of required *in vitro* (in laboratory test tubes) experiments.

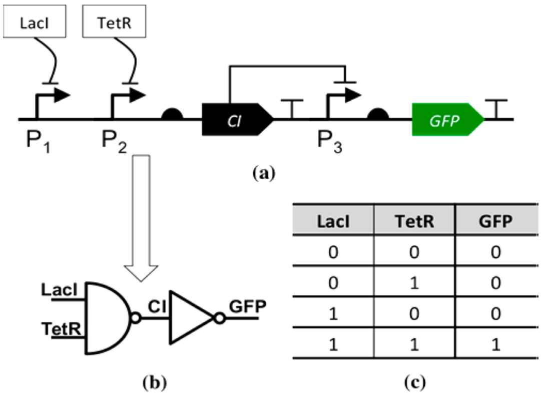
Analogous to microelectronics, where timing analysis is a crucial requirement for ensuring the correct operation of a logic circuit, the timing analysis of genetic logic circuits may become an essential design characteristic as well. The transistors, used in the composition of digital logic gates, have well-defined threshold voltage values,[10] which categorize logic levels 0 and 1. Hence, the timing characteristics, like propagation delay, hold time, setup time, etc., are all well characterized.

However, this is not the case in genetic logic gates, where each gate is composed of different proteins and promoters, resulting in different threshold concentration values. Furthermore, digital logic gates have the same physical quantity, i.e., voltage, as their input and output. On the contrary, genetic logic gates use different biological components including proteins, RNA, inducers, etc., to control the regulation of the

Special Issue: IWBDA 2016

Received: October 14, 2016
Published: January 19, 2017

DOI: 10.1021/acssynbio.6b00296



LacI	TetR	GFP
0	0	0
0	1	0
1	0	0
1	1	1

Figure 1. Genetic AND gate circuit.[13] (a) P_1 and P_2 are promoters, which are the regions of DNA that initiate the process of transcription (or production) of a particular gene. In this example, when two proteins, LacI and TetR, are present in significant amounts within the cell, they inhibit promoters P_1 and P_2 to produce the output gene CI. When the concentration of CI falls below a certain level, promoter P_3 is activated and produces an output protein, i.e., green fluorescent protein (GFP). (b) Schematic diagram equivalent to a two-input electronic AND gate. (c) Truth table.

corresponding output biological components. Additionally, signals in electronic circuits propagate in separate wires that do not directly interfere with each other. However, in genetic circuits, signals are molecules, drifting in the same volume of the cell, and hence easily merge with the concentration of other compounds, resulting in crosstalk with the neighboring circuit components. These facts make the timing analysis of genetic circuits very challenging.

Challenges due to crosstalk have also been encountered in microelectronics; however, most of these have been solved through enhanced fabrication processes or the development of advanced electronic design automation (EDA) tools. Similarly, advances in genetic design automation (GDA) tools may help to address these challenges, resulting in a reduction of the design complexity of genetic logic circuits.

In this article, we introduce a method to perform timing and threshold value analyses on genetic logic circuits and implement the algorithm on D-VASim.[15] We demonstrate that it is possible to perform timing analysis of a genetic circuit and that it can be used to achieve the desired circuit behavior. We perform timing analysis on some of the genetic circuit models proposed in refs 13 and 19 and investigate the sensitivity of circuit timings in relation to varying different circuit parameters. In particular, we investigate the timing sensitivity due to the degradation rate and the concentration of input proteins.

RESULTS AND DISCUSSION

D-VASim is an interactive virtual laboratory environment for the simulation and analysis of genetic logic circuits.[15] It takes a genetic logic circuit model, developed in the Systems Biology Markup Language (SBML),[16] and lets the user analyze the model in an intuitive manner. For each logic circuit model, D-VASim generates a separate virtual laboratory environment for deterministic (by solving ODEs) and stochastic (using Gillespie's stochastic simulation algorithm[20,21]) simulations. This virtual environment serves as a standalone virtual instrument for the specific logic circuit model, which allows users to interact with the model, observe its behavior, and make direct changes in the concentration of input proteins, all during runtime. This runtime interaction gives the user a feeling of being in the lab performing live experiments without being afraid of over reactions occurring or mishandling lab equipment.

Timing Analysis. Threshold value and timing analysis can be used to verify the Boolean function of a genetic logic circuit by extracting the observed logic behavior from the simulation's results. This functionality is useful in two ways: first, it allows the user to verify more complex genetic logic circuits, built by cascading several genetic logic gates; second, it helps the user to extract the Boolean logic of the biomodel even when the user does not have any prior knowledge about the expected behavior of the model. For Boolean logic analysis, the genetic logic circuit model can be considered a black box. Applying all possible input combinations and observing the output can result in the combinatorial behavior of this black box. For instance, if a circuit contains two inputs, then there are four possible input combinations: 00, 01, 10, and 11. The key challenge in determining the correct Boolean logic function from the analog simulation data is to categorize the input concentration levels into logic 0 and 1. As mentioned earlier, this is similar to digital electronic circuits in which a certain threshold value of input voltage differentiates logic levels 0 and 1.[10] Digital electronic circuits are also analog in nature, but a logical abstraction has been employed to reduce the complexity of circuits. Similar abstraction has to be employed to categorize the genetic concentration levels into logic 0 and 1. To categorize these concentration levels into logic 0 and 1, the threshold value for the concentration of input proteins, which significantly affects the concentration of the output protein of a genetic logic circuit, must be identified.

As different proteins in a genetic circuit may have different threshold concentration values, the proposed approach calculates a single threshold value of the input proteins that trigger the output, instead of estimating the threshold values of each input protein separately. For instance, in the genetic AND gate (Figure 1), D-VASim estimates the threshold value of LacI and TetR, which together trigger the production of GFP (green fluorescent protein), rather than evaluating the separate threshold values for each of them. It may be possible that the threshold value of LacI is, for example, 13 molecules, and that of TetR is, say, 9 molecules. In this case, D-VASim tells that 13 molecules is the threshold value of an entire circuit, which triggers the circuit output when the concentrations of input proteins reach this level. Consider another example of an OR gate in which input-1 triggers the output if the molecular count is greater than 5 and input-2 triggers the output if the molecular count is greater than 10. Setting the upper input threshold to 10 would give the correct answer, i.e., the gate remains off, if the input molecular counts are (4,7). Now, if the input molecular counts are (7,4), then input-1 may trigger the output but it may not be considered logic 1 until the output concentration increases above 10 molecules. It is observed, through simulations, that the triggered output for such scenarios is highly unstable (frequently oscillating between logic 0 and 1), and this region should be considered a transition region. Therefore, instead of estimating the threshold values of each input protein separately, our algorithm estimates the global upper and lower threshold values for all inputs. Furthermore, D-VASim considers the entire circuit as a black box and obtains the input threshold value that is required to trigger the final output. Therefore, the threshold value and the number of intermediate circuit components do not matter; D-VASim ensures that the estimated input threshold value is sufficient to trigger the intermediate circuit components all the way from input to final output. However, the separate threshold values of intermediate circuit components can also be analyzed in D-VASim.

In order to understand the algorithm for estimating the threshold value, consider the simulation traces of the genetic AND gate using iBioSim[18] shown in Figure 2. In Figure 2b, it is seen that keeping the input concentrations to 10 molecules causes the average output concentration to stay below the level of the input concentration. Upon increasing the input concentrations further to 13 molecules, the average output concentration goes above the level of the input concentrations, as depicted in Figure 2c. We also performed the same analyses with different concentrations on different logic circuits. These analyses show a relation between the input and output proteins of a genetic circuit. On the basis of these analyses, we define an input−output relation of a genetic circuit in terms of its threshold value as follows:

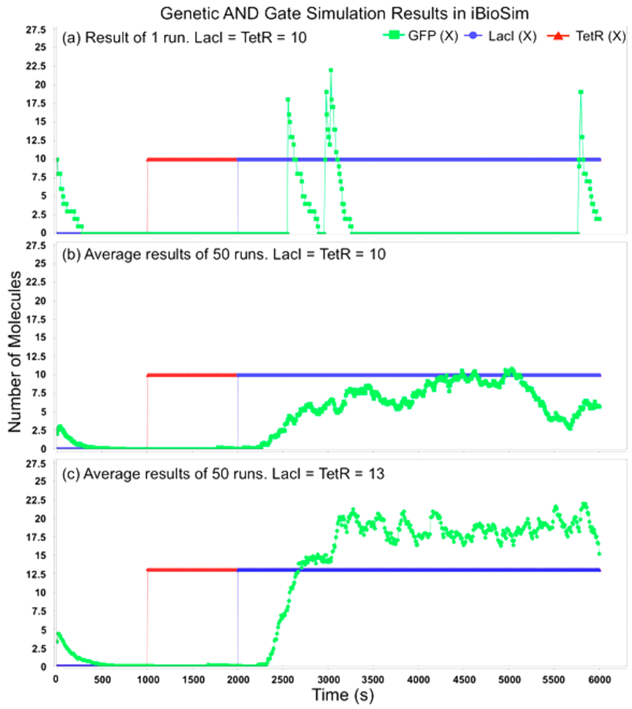


Figure 2. Preliminary analysis of a threshold value for the genetic AND gate using iBioSim.[18] It shows the results from running the stochastic simulation of the genetic AND gate 1 time (a) and 50 times (b) and (c). The unit of species' concentration used in the circuit models of ref 13 is the "number of molecules". Panel (a) shows that both of the inputs are triggered to 10 molecules, TetR after 1000 time units and LacI after 2000 time units, and that the output is highly stochastic, which makes it difficult to determine the input threshold value. A smooth output curve is obtained by plotting the average of 50 runs, as shown in (b) and (c). Panel (b) shows the lower threshold value of inputs LacI = TetR = 10, and panel (c) shows the upper threshold value of inputs LacI = TetR = 13.

Definition I. Threshold value: The minimum concentration of input protein(s) that causes the average concentration of output protein to cross the concentration of input protein(s).

In the example shown in Figure 2, the upper threshold value of input is 13 molecules; that is, the input concentration above 13 molecules is considered logic 1, and that below 10 molecules is considered logic 0. There is a transition region between these two levels (not shown in Figure 2), where the average output concentration is not clearly distinguishable with the input concentration level. Hence, when the concentration levels of both inputs are 10 or fewer molecules, i.e., logic 0, the average output concentration remains low (logic 0), else it goes high (logic 1) when the concentration of both inputs reaches 13 or more molecules (logic 1). This relation of input and output concentration is justified because, according to this definition, one do not need to care about how many circuit levels are cascaded between input and output. It simply identifies the input concentration required to trigger the final output. The same definition is applicable to determine the threshold values of intermediate circuit components separately.

Another important factor for automatically obtaining the correct Boolean expression from the simulation data is the propagation delay. Figure 3 shows a zoomed-in version of Figure 2c, which shows that the effect of changes in the input concentration is reflected in the output concentration after a time delay of approximately 700 s. That is, the output protein takes about 700 time units to cross the level of the input concentration when the inputs are triggered to their threshold value. Thus, we define the propagation delay of a genetic circuit as follows:

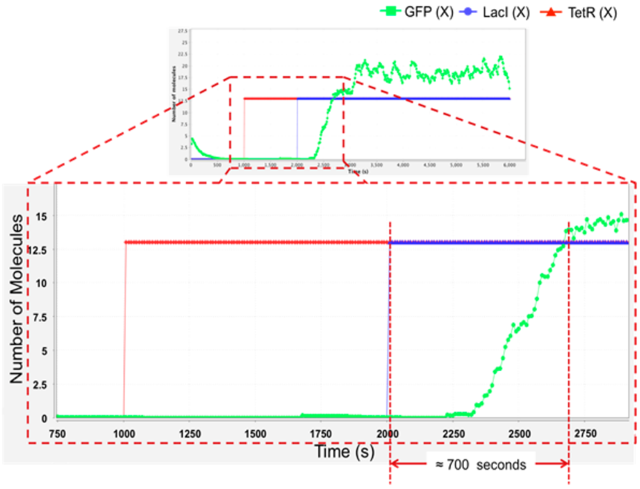


Figure 3. Zoomed-in image of Figure 2c indicating the preliminary propagation delay analysis using iBioSim.[18] In this figure, the propagation delay is approximately 700 s.

Definition II. Propagation delay: The time from when the input concentration reaches its threshold value until the corresponding output concentration crosses the same threshold value.

Figure 3 shows that the output goes "high" after approximately 700 s from the time when both inputs have reached the significant concentration level (13 molecules). During these 700 s, the output remains low and hence does not produce the expected logic output. It also means that, during simulation (or even during experimentation in the laboratory), the user should not change the inputs before this time delay has elapsed. In order to identify the threshold levels of a circuit in the laboratory, the biologist could perform this analysis by adding the input concentration periodically to see if it significantly affects the concentration of the output. To identify the input concentration, which significantly affects the output, different input combinations must be tried with different concentration levels, which is a very tedious and time-consuming task to do in the laboratory. Furthermore, as mentioned above, it must be ensured that each input combination is applied after a certain time delay. In this work, we have developed an algorithm to automate this process and tested it on D-VASim.[15] Since the behavior of a genetic circuit is well described by stochastic simulations, we therefore apply the proposed method to the stochastic behavior of a genetic circuit obtained from Gillespie's stochastic simulation algorithm.[20,21]

The algorithm for the threshold value and propagation delay analysis is shown as pseudocode in Chart 1. The algorithm is initialized by some user-defined parameters as indicated in Chart 1. C_{in} specifies the value of the input protein(s) concentration, from which the tool should start its threshold analysis. *Inc* is the value with which the input concentration is increased for each iteration, in order to observe if the resulting concentration level of the input affects the concentration of the output. The C_{inE} value specifies the input concentration at which the algorithm should stop the analysis of the threshold value. The algorithm also requires an initial assumption of the input−output propagation delay value, T_D. It was already mentioned that the input−output propagation delay value is critical for extracting the correct logic behavior of a circuit model. Thus, it is necessary to wait until this time value has elapsed before applying the next combination of inputs. Since the time delay value is unknown for the automatic analysis, the tool begins the analysis with an assumed value and later estimates the approximate one. Assuming a higher value increases the estimation time but gives a better estimation of the threshold value.

Chart 1. Threshold Value Analysis Algorithm

```
Threshold value and propagation delay analysis
begin
      INITIALIZE ($C_{in}$, $Inc$, $C_{inE}$, $T_D$, $S_T$, $i$, $O_S$, $V_T$, $OC_{DUTh}$, $OC_{DLTh}$)
      /* $C_{in}$ = Initial input concentration where the analysis should start from
        $Inc$ = Increment value: The value which is added to the previous
        concentration level until the concentration level reaches to $C_{inE}$
        $C_{inE}$ = End concentration of input where the analysis should stop at
        $T_D$ = Assumed time delay
        $S_T$ = Settling Time
        $O_S$ = Name of output specie
        $i$ = Number of iterations to verify the consistency of results
        $V_T$ = Amount of time to verify a model for each iteration $i$
        $OC_{DUTh}$ and $OC_{DLTh}$ = user defined percentage acceptance of output
        consistency for upper and lower threshold values respectively */
  for all possible input combinations do
   if ($C_{inC}$ == 0) then      /* $C_{inC}$ = current input concentration level*/
      Determine initial output concentration ($C_{Oinit}$)
   else
     while1 ($C_{inC} \leq C_{inE}$)do
       while2 ($T_{C1} \leq T_D$) do        /* $T_{C1}$ = current time 1 */
         Execute simulation
           if ($C_{O_S} > C_{inC}$)** then    /* $C_{O_S}$ = output concentration of selected specie*/
             PT = $C_{inC}$              /* PT = Possible Threshold Value */
             /*Verification process*/
             for number of iterations $i$ do
               while3 ($T_{C2} \leq V_T$) do       /* $T_{C2}$ = current time 2 */
                 Execute simulation
                   if ($T_{C2} \geq S_T$) then
                     Trigger the input to the value of PT
                   Store the output concentration data in array
               end
               Take the running average of all output i arrays
             end
             Estimate time delay ($T_E$) and consistency ($OC_E$)
             Terminate loop2
       end
       if ($C_{O_S} > C_{inC}$)** then
         if ($OC_E > OC_{DUTh}$) then
           Consider lower threshold value = 0 if not found already
           Return the results and terminate all loops
         else if ($OC_E < OC_{DLTh}$)
           Save lower threshold level and resume analysis
         else
           Resume analysis
       $C_{inC}$ = $C_{inC}$ + $Inc$
       $T_{C1}$ = $T_{C2}$ =0
     end
   end
  end
end
```

**Valid when C_{Oinit} is low. For high C_{Oinit} it will become ($C_{O_S} < C_{inC}$)

For a simulation, if every node of a genetic circuit model is not initialized to a stable value, then some of the genetic circuits' outputs are initially unstable and exhibit unexpected behavior for a certain amount of time. For example, in the simulation traces (see Figure 2) of the genetic AND circuit shown in Figure 1, the initial values of LacI and TetR are zero, but when the simulation starts, the output, CI, of the first circuit's component (i.e., NAND gate, see Figure 1) is also zero, which enables the inverter and produces GFP until the input value 0 propagates through the NAND gate. In order to perform correct timing analysis, it is therefore required to initialize all of the circuit nodes to a stable value. If the values are not initialized, it is important that the algorithm should wait for the circuit's output to become stable first. The parameter S_T (settling time) helps the user to specify a rough value for the initial time during which the circuit's output is expected to become stable. When the algorithm performs automatic analysis, it waits for the value defined for S_T to allow the circuit's output to become stable first and then triggers the input combinations to determine the appropriate threshold and propagation delay values of a circuit. For small genetic circuits, containing a single gate only (for example, NOT, NAND, and NOR) and having a low degradation rate (k_d = ∼0.0015), it is observed through simulations that these circuits usually take at least 1000 time units to become stable. This implies that, for these circuits and k_d, the S_T parameter should not be less than 1000 time units. If a value less than this is chosen, then the algorithm will not be able to produce the correct estimation.

The algorithm further verifies the obtained threshold value by iterating the model for a predefined number of iterations, *i*. During this iterative verification process, the algorithm obtains the average propagation delay by running the model for the length of time defined by V_T for each iteration *i*. It also identifies the extent to which the average output for the estimated threshold value is consistent. In order to understand this procedure, lets assume the parameter values shown in Table 1. The unit for concentration here is the "number of molecules".

Table 1. Sample Values of Parameters Required for Threshold Value and Timing Analysis

parameter name	value
C_{in}	0
Inc	2.75
C_{inE}	15
T_D	800
S_T	200
i	10
V_T	1000
OC_{DUTh}	90
OC_{DLTh}	30

DOI: 10.1021/acssynbio.6b00296
ACS Synth. Biol. XXXX, XXX, XXX−XXX

Now consider the sample time scale plots shown in Figure 4. To find the threshold value of the input concentration that significantly affects the output concentration, a specific input combination should be applied. This means that all possible combinations should be checked one by one until the specific combination of inputs that triggers the output concentration is found. For logic circuits like AND, NAND, OR, NOR, and NOT, the output transition can be observed by triggering both of the inputs to the same concentration level at the same time. The algorithm, therefore, triggers both input combinations from 00 to 11 first, instead of following the traditional pattern of 00 → 01 → 10 → 11. Because of this, the algorithm estimates the threshold value of some circuits, for example, AND gate, relatively faster.

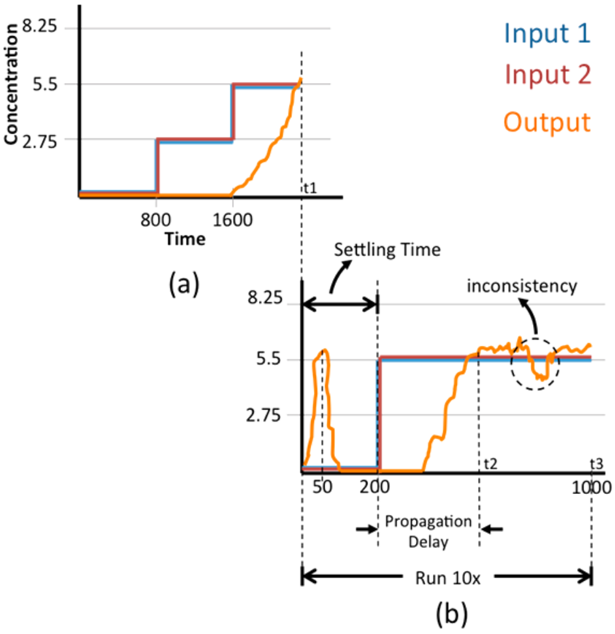


Figure 4. Sample time scale plots of the genetic AND gate. This figure shows how the automatic threshold value and timing analysis takes place by the proposed algorithm. (a) First loop to detect the threshold value. (b) Separate loop to verify the estimated threshold value repeatedly for a predefined number of iterations, *i* (10 in this case). If more than 90% of the average output data, between instants *t*2 and *t*3, remains above the input level, then the input concentration level is considered to be the upper threshold level. Similarly, if less than 30% of the average output data remains above the input level between instants *t*2 and *t*3, then the input concentration level is considered to be the lower threshold level. The propagation delay is measured from the instant when the input is triggered from its lower threshold level to its expected upper threshold level to the instant when the average output crosses the same input level.

Figure 4a shows the case of input logic combination “11”, i.e., when both inputs are triggered high. According to the settings shown in Table 1, the algorithm runs the model first by keeping the input concentration at zero until the assumed time delay of 800 time units has elapsed. In order to determine if the output concentration crosses the level of the input concentration as defined in *Definition I*, or in other words, to determine whether the output concentration goes above the input concentration level or falls below it, we need to know the initial concentration of output protein at input logic level combination “00”. Therefore, during the first 800 time units (T_D), the average of the initial output concentration is obtained by keeping the concentration of both inputs at zero, i.e., logic 0. On the basis of this average initial output concentration, the estimation of the output concentration crossing the input concentration level is performed.

Once the assumed time delay has elapsed, the input concentration level is incremented to the next level, indicated by line 32 in Chart 1. The example case shown in Figure 4a portrays the scenario of an AND gate where the initial average output of a circuit (with both input concentrations at zero) is zero. The algorithm also works for the case where the average initial output concentration is high, for instance, a NOT gate, by iteratively increasing the input concentration and checking if the output concentration falls below the concentration level of the input. Note that this still satisfies *Definition I*.

Point *t1* shown in Figure 4a implies that the algorithm halts the current loop execution when the value of the output protein crosses the input concentration level. This anticipates the possible threshold value (5.5 molecules in this example) as it makes the output concentration cross the input concentration level. To verify this threshold value, the algorithm executes a separate loop to run the simulation of the circuit model for the defined number of iterations, 10 times in this example, as shown in Figure 4b. This process is executed in lines 13−21 in Chart 1.

In order to measure the correct propagation delay, it is necessary to trigger the input protein, particularly from zero to the expected threshold level, only when the model’s initial output is settled. As mentioned before, the outputs of some circuits are unexpectedly high, which gradually settles to zero. This scenario is depicted in Figure 4b. Therefore, the initial concentrations of input proteins must not be triggered to their expected threshold level until the output becomes stable. As mentioned above, the settling time, S_T, lets the user provide a period of time by which the initial output is expected to become stable. This is the time at (or after) which the algorithm triggers the inputs to their expected threshold levels to determine the time it takes to trigger the output concentration. If a low value is assumed for S_T, then the algorithm may produce an incorrect propagation delay. For example, in Figure 4b, if a value of 50 was chosen as the settling time, then the inputs would be triggered at 50 time units. At this instant, when the inputs are triggered to their expected threshold level, the concentration of the output is already above the threshold level and thus the algorithm would estimate the propagation delay to zero. Therefore, depending on the complexity of a circuit and the degradation rate (k_d), this value should be chosen carefully.

The simulation output data from all 10 iterations were averaged to obtain the average estimated propagation delay and the inconsistency present in the output plot for the estimated threshold values. The inconsistency, illustrated in Figure 4b, is calculated by determining the size of the average output data, which is less than the the input concentration level immediately after the output crosses the input level for the first time, i.e., the inconsistency is estimated between points *t2* and *t3* as shown in Figure 4b. In other words, for examining the upper threshold level, the idea is to determine how consistently the average output data remains above the input concentration level between points *t2* and *t3*. The algorithm accepts the estimated threshold value based on the user-defined parameter, % acceptance of consistency, shown as OC_{DUTh} (for upper threshold level) and OC_{DLTh} (for lower threshold level) in Chart 1.

The results are accepted if the estimated consistency is greater (for the upper threshold) and less (for the lower threshold) than the user-defined values, OC_{DUTh} and OC_{DLTh}, respectively.

This is shown in the lines 26−31 in Chart 1. The results are otherwise discarded, and the algorithm resumes the analysis from point *t1*, shown in Figure 4a. The percentage output consistency is calculated according to eq 1.

$$\% \text{ output consistency} = \frac{O_{t2-t3} - D}{O_{t2-t3}} \times 100 \qquad (1)$$

where O_{t2-t3} is the size of the average output data between instants *t2* and *t3* (Figure 4b) and *D* is the deviation, which defines the number of times the output data is found to deviate from the expected (greater or less than) threshold value.

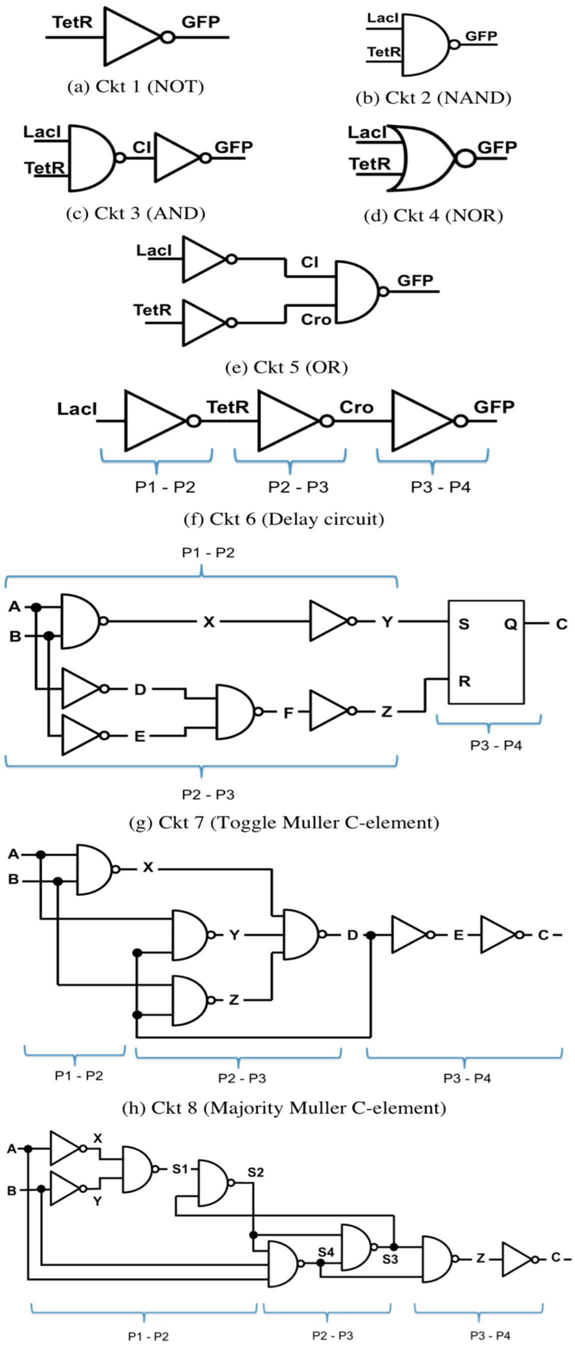
The quantity *D* in eq 1 is considered to be different in two different cases; i.e., when the initial input concentration is found low, then *D* in eq 1 indicates the number of times the output data is found to be “less” than the threshold value, as in the case shown in Figure 4. Else, if the initial input concentration is high, then *D* signifies the number of times the output data is found to be “greater” than the threshold value. For the sample parameters (Table 1) used for the sample plots shown in Figure 4, the algorithm estimates the input concentration as the upper threshold level if the output consistency is 90% or above. Likewise, the input level is assessed as the lower threshold level if the estimated output consistency is less than 30%.

Simulation Results. In this research, timing analysis is performed on the nine genetic logic circuit models (Figure 5) proposed in ref 13. The genetic implementation and the description of these circuits can be found in ref 13. These circuits are considered fairly complex in the context of genetic circuits because each gate is composed of several genetic components. Their kinetic interactions are described by a number of mathematical equations in the SBML model. We ran the SBML models of these genetic logic circuits on D-VASim and performed their threshold value and propagation delay analysis.

In microelectronic devices, the behavior of a circuit depends on many different parameters. For example, in MOS transistors, the drain current depends on the width and length of the gate, oxide capacitance, gate-to-source voltage, and so forth. Similarly, the behavior of a genetic circuit also depends on different parameters, including degradation rate, forward repression binding rate, forward activation binding rate, and so on. These parameters of a genetic circuit model are described in the SBML file. We carried out simulations on these nine genetic circuit models by observing the effects of varying the degradation rate (k_d) on the propagation delay and the threshold value of a circuit.

The degradation rate is the rate at which a chemical compound (e.g., a protein) is decomposed into intermediate products, i.e., a produced protein will be effective only for a certain period of time determined by the degradation rate. A zero degradation rate means that the protein does not degrade and hence will be effective forever. This is an often-used assumption, which clearly is not realistic, which is why understanding the impact of the degradation rate on the timing analysis is an important investigation.

The threshold value and propagation delay of each circuit is obtained for five different values of k_d (0.0015, 0.0055, 0.0095, 0.0135, 0.0215). These set of values are chosen based on a degradation rate value used in ref 13. It has been experimentally observed that the variation in degradation rate (k_d) greatly effects the settling time, S_T, of an output. Hence, for each circuit, we chose different parameter values (shown in Table 1), except the number of iterations, *i*, and % acceptance of consistency for the upper and lower threshold values (OC_{DUth} and OC_{DLth}), which were set to 5, 70, and 30, respectively, for



(a) Ckt 1 (NOT)

(b) Ckt 2 (NAND)

(c) Ckt 3 (AND)

(d) Ckt 4 (NOR)

(e) Ckt 5 (OR)

(f) Ckt 6 (Delay circuit)

(g) Ckt 7 (Toggle Muller C-element)

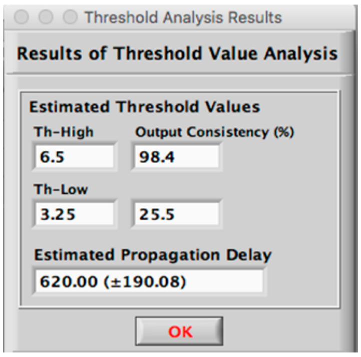
(h) Ckt 8 (Majority Muller C-element)

(i) Ckt 9 (Speed Independent Muller C-element)

Figure 5. Set of genetic logic circuit models by Myers[13] that is used for the experimentation in this work. More complex circuits, (f)−(i), are further categorized into three intermediate levels: P1−P2, P2−P3, and P3−P4. The timing analyses are performed on these three levels separately, which are mentioned in Table 2. The SR latch shown in Ckt 7 is asynchronous and does not require a clock input.

all circuits. We purposely used OC_{DUth} = 70% to demonstrate that it is affected by threshold values.

Figure 6 shows how D-VASim reports the outcomes of a threshold value and propagation delay analysis once the algorithm finishes execution. This figure shows the threshold value and timing analysis results obtained for Ckt 8 (Figure 5h) when the degradation rate (k_d) was set to 0.0135. It indicates that the estimated upper and lower threshold values are 6.5 and 3.25 molecules with 98.4 and 25.5% consistency, respectively. It also calculates the approximate input−output propagation delay value to 620 time units with a standard deviation of ±190.08, in this case based on five iterations. When the results are obtained, the user may interact with the model during run time, apply all possible input combinations in a significant amount, and match the propagation delay with the one estimated by D-VASim. Figure 7 shows the simulation traces of the same Ckt 8 with k_d = 0.0135. Due to space limitation, only screen shots of the analysis results and graphical simulation of Ckt 8 (for k_d = 0.0135) are included in this article. However, the complete experimental data of all circuits is available in the Supporting Information.



Threshold Analysis Results

Results of Threshold Value Analysis

Estimated Threshold Values

Th-High	Output Consistency (%)
6.5	98.4
Th-Low	
3.25	25.5

Estimated Propagation Delay

620.00 (±190.08)

OK

Figure 6. Results of threshold value and propagation delay analysis of Ckt 8 generated by D-VASim for k_d = 0.0135. The estimated upper and lower threshold values are 6.5 and 3.25 molecules with 98.4 and 25.5% consistency, respectively. The approximate input−output propagation delay value is 620 time units with a standard deviation of ±190.08.

Figure 8 shows the graphical plots of timing analysis for all circuits. The values of propagation delays are plotted along the y axis on the left-hand side. The threshold values and percentage output consistency for each value of k_d are plotted along the y axis on the right-hand side. The x axis contains the degradation rate values. The general impression of these

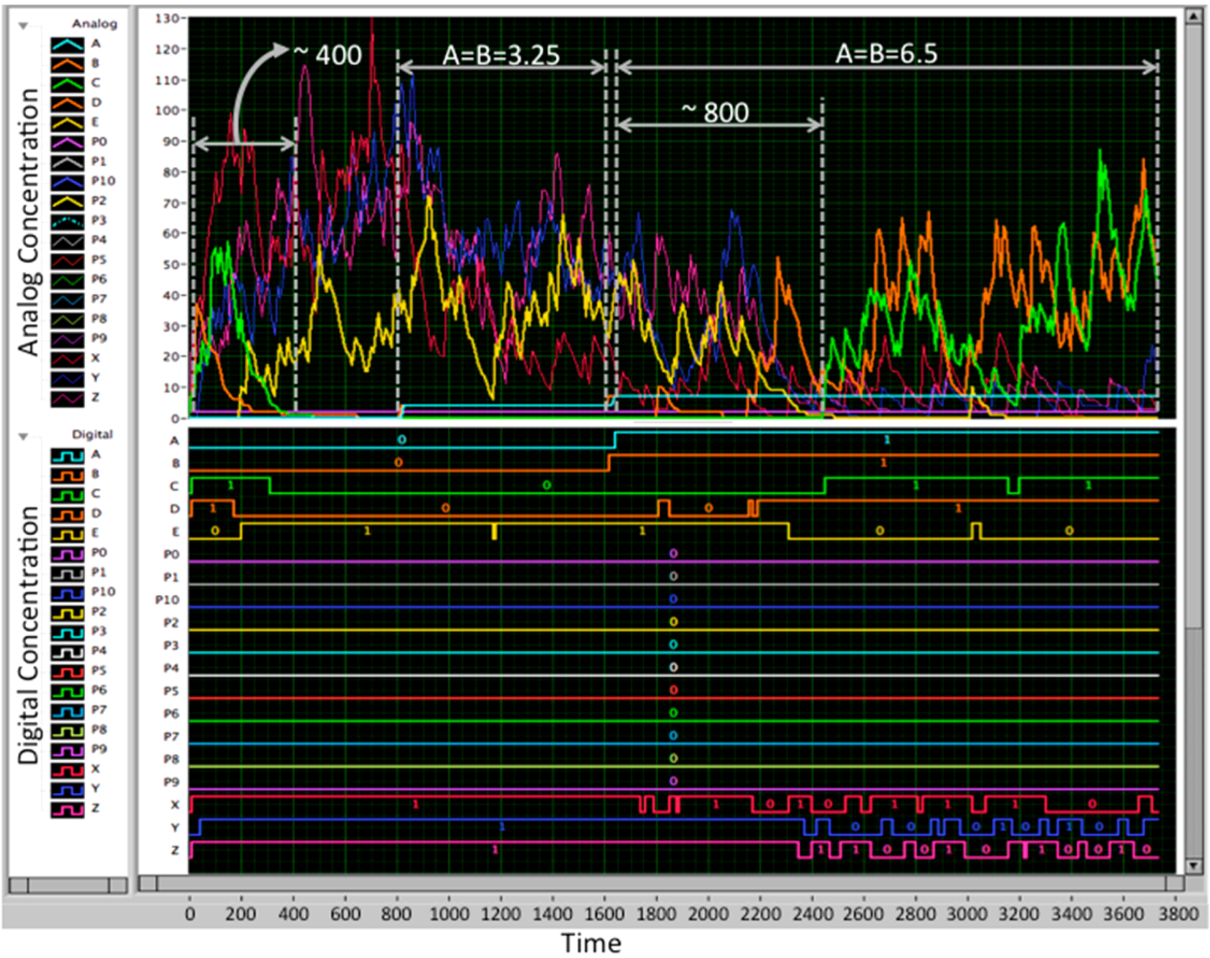


Figure 7. Analog simulation traces of Ckt 8 with its corresponding digital waveforms for k_d = 0.0135. Inputs are A and B; output is C. It can be observed that the initial concentration of output protein, C (shown as green plots), is high above the threshold value and it takes approximately 400 time units to settle. Furthermore, when the input concentrations are triggered to their lower threshold level, i.e., 3.25 molecules, the output concentration remains zero. When the input concentration levels are triggered sharply to their estimated threshold value, i.e., 6.5 molecules, the output of a circuit is high after approximately 800 time units.

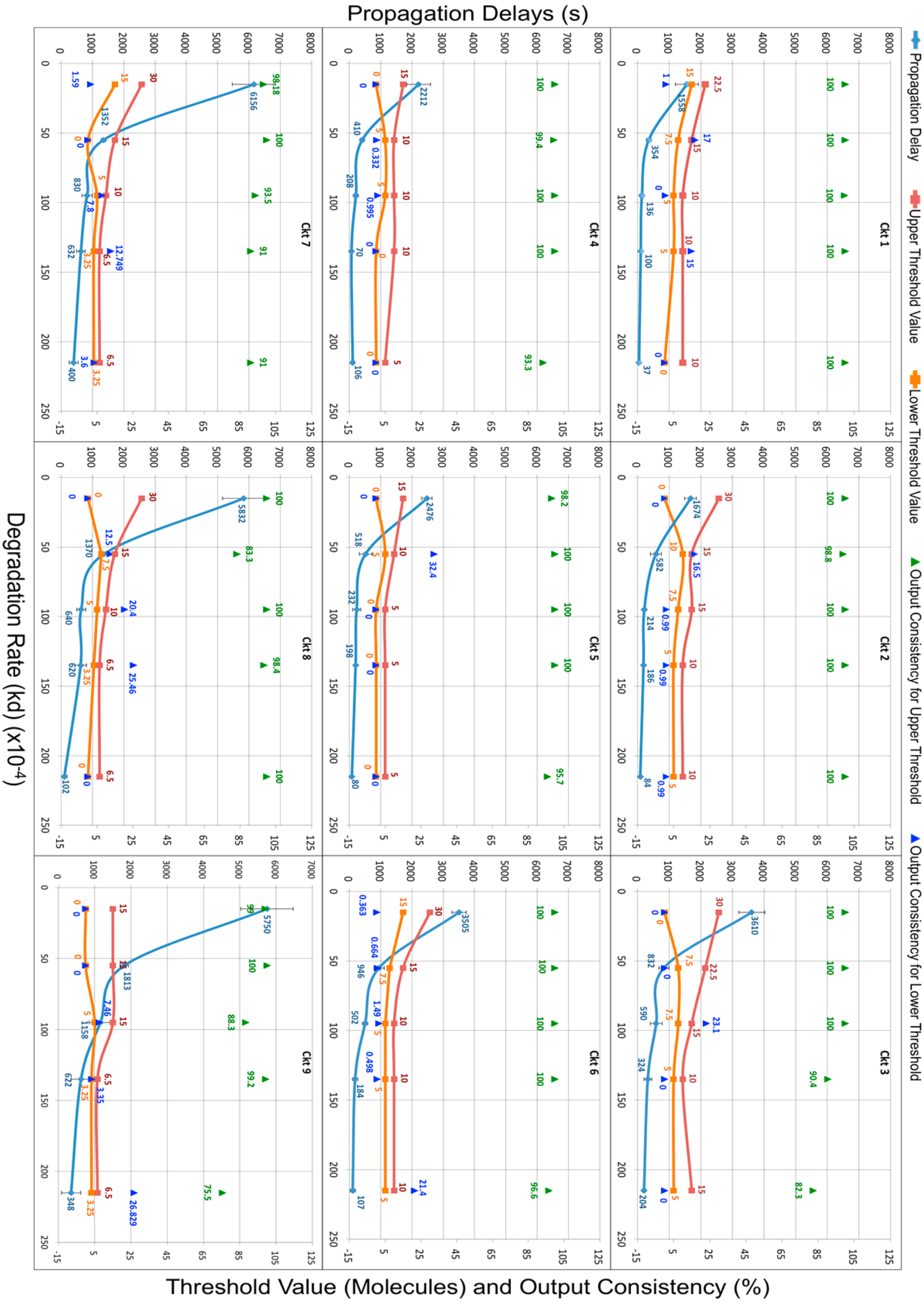
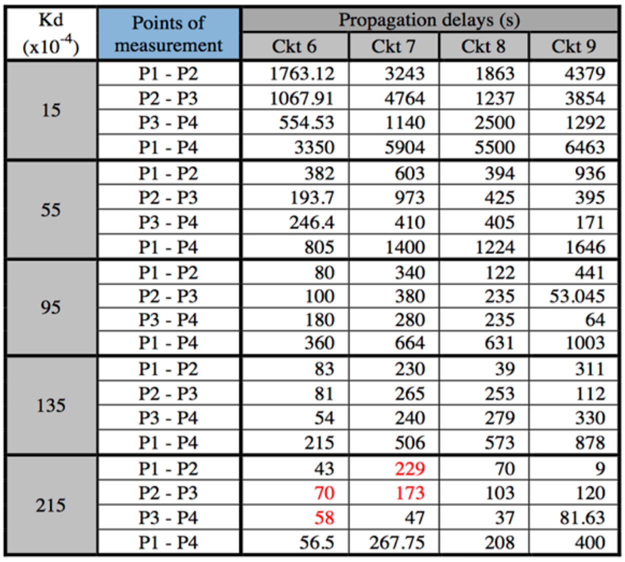


Figure 8. Effects of varying the degradation rate (k_d) on the propagation delay of genetic logic circuits.

experiments is that the propagation delay of genetic circuit decreases with the increase in degradation rate (k_d). This was expected because when the degradation rate is high the protein degrades faster and thus contributes in a reduction in the propagation delay. However, the propagation delay does not seem to have an inverse relation with the degradation rate. The propagation delay for all circuits dropped considerably with the first decrement of 40 × 10^{-4} in k_d, and it then decreases slowly for the next higher values of k_d. The standard deviation in the propagation delay, calculated for the specified number of iterations (i.e., five in these experiments), is also included for each circuit in the plots shown in Figure 8. It can be noticed that the propagation delay of a circuit is more variable for low degradation rates. The variation in the propagation delay decreases with the increase in degradation rate; however, a high degradation rate makes the cascaded circuit's output unstable. This is because the genetic components decay quickly when the degradation rate is high, thus causing the circuit's logic to switch faster even when a small input concentration is applied. This also reduces the transition region between the upper and lower threshold levels, as shown in the Ckt 8 data in Figure 8. However, it can also be noticed for Ckt 8 that a transition region is small for k_d = 0.0135 (i.e., 3.25) as compared to k_d = 0.0215 (i.e., 6.5). This is because, at k_d = 0.0215, Ckt 8 becomes unstable and produces high output glitches even when the input concentration levels were kept to zero (see simulation traces in the Supporting Information). This is the reason that the lower threshold value of Ckt 8 at k_d = 0.0215 is estimated to be zero.

We further analyzed the intermediate delays of larger genetic circuit models by splitting them into three points of measurement, as shown in Figure 5. The propagation delays for each of these points are mentioned in Table 2. The propagation delay, indicated by a point of measurement P1−P4 in Table 2, is the entire circuit's propagation delay. The reader should not confuse these estimations with those depicted in Figure 8. The results mentioned in Figure 8 are estimated by D-VASim using the proposed algorithm, and the results mentioned in Table 2 are those that are obtained by a user through a runtime stochastic simulation.

Table 2. Intermediate Propagation Delays of Genetic Logic Circuits



Kd (x10^{-4})	Points of measurement	Propagation delays (s)			
		Ckt 6	Ckt 7	Ckt 8	Ckt 9
15	P1 - P2	1763.12	3243	1863	4379
	P2 - P3	1067.91	4764	1237	3854
	P3 - P4	554.53	1140	2500	1292
	P1 - P4	3350	5904	5500	6463
55	P1 - P2	382	603	394	936
	P2 - P3	193.7	973	425	395
	P3 - P4	246.4	410	405	171
	P1 - P4	805	1400	1224	1646
95	P1 - P2	80	340	122	441
	P2 - P3	100	380	235	53.045
	P3 - P4	180	280	235	64
	P1 - P4	360	664	631	1003
135	P1 - P2	83	230	39	311
	P2 - P3	81	265	253	112
	P3 - P4	54	240	279	330
	P1 - P4	215	506	573	878
215	P1 - P2	43	229	70	9
	P2 - P3	70	173	103	120
	P3 - P4	58	47	37	81.63
	P1 - P4	56.5	267.75	208	400

The argument that a circuit's output becomes unstable for larger values of k_d can also be supported by observing the intermediate delays of Ckt 6 for k_d = 0.0215 in Table 2. As shown in Figure 5, Ckt 6 is composed of three inverters connected back-to-back in series. When input protein LacI is triggered to its threshold value, it suppresses the production of TetR. When the concentration of TetR drops below its threshold level, it produces Cro, which in turn suppresses the production of output protein GFP. However, the intermediate propagation delays of Ckt 6 for k_d = 0.0215 shows that when the input protein, LacI, is triggered to the estimated threshold value, the overall output of a circuit, GFP, is produced in 56.5 time units. However, one of the intermediate outputs, Cro, is produced in a significant amount after ∼70 time units, which is greater than the propagation delay of the entire circuit. This invalidates the desired circuit's behavior and makes the output unstable, which indicates that the circuit does not behave as designed.

For Ckt 7, the results of intermediate propagation delays indicate that an intermediate output Y is always triggered first before another intermediate output Z. This is obvious because the number of components in path P1−P2 is fewer than the components in path P2−P3. However, this timing analysis also implies that both S and R inputs of the SR-latch remain high for some interval of time. This, however, seems to have no effect on the output, C, during runtime simulations. It may be because this time interval is less than the propagation delay of SR-latch (i.e., P3−P4). If both S and R remain high for a time greater than a delay of P3−P4, the output is likely to become unpredictable. For a large degradation rate (k_d = 0.0215), the intermediate output Z (P2−P3) is produced first, before intermediate output Y (P1−P2), which invalidates the intended circuit's behavior because the number of components in path P1−P2 are higher than in P2−P3. Similarly, the outputs of other cascaded circuits are also unstable for higher values of k_d.

Aside from these analyses, we observed some other interesting facts from varying the upper threshold value on the propagation delay of a circuit. It is observed that smaller concentrations of the input protein have a weak impact on the output protein, which is analogous to the behavior of microelectronic devices. For instance, in MOS transistors, weak applied V_{GS} (gate-to-source voltage) results in a weak drain current, I_D.[17] This effect can be observed in the graphical plot of Ckt 4 in Figure 8. In this plot, for k_d = 0.0215, the threshold value of a circuit is reduced to 5 molecules as compared to its previous data point, which is 10 molecules at k_d = 0.0135. Due to the increment in k_d, the propagation delay at this point is supposed to decrease if the input threshold value remains the same. However, it slightly increases because the input threshold is reduced to 5 molecules. We have observed this effect on other circuits as well during run time simulations. For instance, we increased the applied input concentration to 60 molecules for Ckt 7 at k_d = 0.0015 and observed that the propagation delay is decreased from 6156 to 5570 time units (see Supporting Information). This inverse relation between propagation delay and threshold value holds to a certain extent, and then a further increment or decrement in the applied input concentration does not affect the propagation delay.

Moreover, for higher values of k_d, we observed that the output consistency of the upper threshold level is increased by reducing the threshold value. This is shown in the plots of

DOI: 10.1021/acssynbio.6b00296

Ckt 9 in Figure 8. The output consistency of the upper threshold level, at $k_d = 0.0135$ was reduced to 49% (not shown in Figure 8) when the threshold value was set to 30 molecules. We then analyzed the output consistency of Ckt 9 at $k_d = 0.0215$ by keeping the threshold value to the same level, i.e., 30 molecules (not shown in Figure 8), and we noticed that the output consistency decreased to 2%. We then decreased the threshold value to 6.5 molecules and observed that the output consistency increased to 75.5%, as shown in Figure 8.

The plots for lower threshold values of each circuit are also shown in Figure 8. It has been observed that the lower threshold value of all circuits approaches zero as k_d increases. The values of the upper and lower threshold levels also depend on the parameter *Inc* (see Table 1), which specifies the input concentration to be added to the previous input concentration level during each iteration, *i*. For example, in the case of Ckt 2 and Ckt3, the value of *Inc* was set to 30 at $k_d = 0.0015$. The algorithm thus triggers the input concentration from 0 to 30 directly during the analysis. Because of this, the average output was found to be 100% consistent for the upper threshold level, which results in estimations of the upper and lower threshold levels of 30 and 0 molecules, respectively. If a lower value of *Inc* was chosen, then the results would be different but more precise.

We further explored the possibility of analyzing the SBML models of real genetic circuits. We picked the genetic AND gate circuit (composed of inverters and NOR gates) from ref 19. The genetic circuits presented in ref 19 were first developed with a tool named Cello, which generates the Synthetic Biology Open Language (SBOL) file.[22] Unlike SBML, the SBOL representation does not describe the behavior of a biological model. We, therefore, first used the SBOL–SBML converter[23] to generate the behavioral model of the above-mentioned real genetic AND circuit. This SBOL–SBML converter is available as a plug-in in iBioSim,[18] which uses the default parameters while defining the reaction kinetics during the conversion process. Since the actual parameters, like degradation rate, forward repression binding rate, etc., are not disclosed in ref 19, we performed timing analysis of the real genetic AND gate circuit using the default iBioSim parameters. However, the parameter values can always be changed, and new parameters can also be added to observe more realistic results. Furthermore, the SBOL file generated by Cello does not include the input sensor block of a circuit (which includes the input inducers); thus, these inducers are also not included during the SBOL–SBML conversion process. Hence, we added the input inducers manually in the SBML model using iBioSim, as shown in Figure 9. The components inside the yellow box are manually added, and rest of the model is a result of SBOL-SBML conversion process.

Figure 10 shows timing analysis results of the SBML model of the genetic AND gate circuit.[19] All of these analyses for different degradation rates were obtained within 30 min, and the simulations with all possible input combinations were performed within 10 min. This is obviously faster compared to testing the model in a lab, where the models were first placed in the logic-0 state for 3 h and then switched to other possible states, one by one, each for another 5 h.[19]

Figure 10 indicates that the results of the genetic AND gate[19] are similar to those obtained for the other nine genetic circuit models.[13] In general, it is observed that the propagation delay, threshold value, and degradation rate are all interlinked. The output of a circuit is stable for small values of k_d, but it increases the propagation delay. The variation in the propagation delay is also greater for small values of k_d. On the other hand, the output of a circuit becomes unstable for large values of k_d but decreases the propagation delay. Large values of k_d also contribute to a reduction in the threshold value to a certain point. This is because the circuit becomes faster for large k_d; therefore, a small input concentration is sufficient to trigger the output protein. The degradation rate cannot be increased beyond a certain point because it makes the output oscillate highly. This implies that the threshold value of a circuit cannot be decreased beyond a certain point. This corresponds to scaling trends for the MOSFET device, where the gate width cannot be reduced beyond a certain number of nanometers.

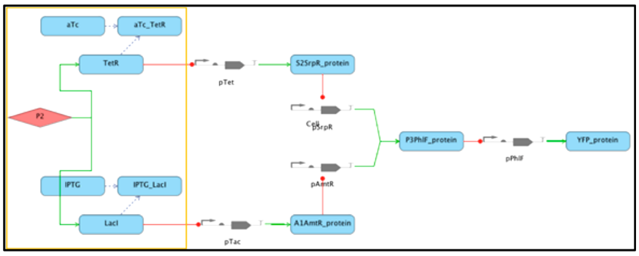


Figure 9. SBML design of the genetic AND gate circuit obtained from ref 19. When both of the input inducers, aTc and IPTG, are present, they form a complex with their corresponding regulators. These regulators then gradually stop inhibiting their respective promoters, which eventually leads to the production of the output protein, YFP.

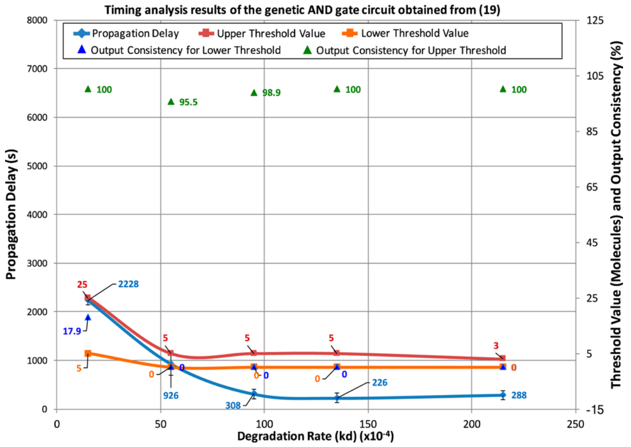


Figure 10. Effects of varying the degradation rate (k_d) on the propagation delay and threshold value of the genetic AND gate circuit.[19] The propagation delay of the circuit decreases with the increase in k_d. Additionally, low input concentrations are required to trigger the output of a circuit at higher values of k_d. Note that these results may differ when the actual parameter values are used.

Summary. In this work, we proposed a method to perform timing analysis of genetic logic circuits, which is then implemented and tested in D-VASim. We primarily performed the threshold value and timing analysis on entire circuits instead of on each individual circuit component. However, D-VASim is also able to analyze the threshold value and timing analysis of individual circuit components. In this work, D-VASim estimates the overall threshold value of an entire circuit, which gives user a minimum value of the input species required to trigger the output of a genetic circuit. We further explored the effects of a circuits' timing upon varying certain parameters. This may assist genetic circuit designers in finding an appropriate set of parameters to achieve the desired timings of a genetic circuit. D-VASim could actually help reduce the time-consuming *in vitro* experiments (laboratory experiments) needed to analyze and design genetic circuits with the desired behavior and timings. We anticipate that the ability to analyze the timings of a genetic circuit may open up a new research area, helping biologists and scientists design and characterize the timing properties of genetic circuits. Depending on the complexity of a genetic circuit and the user-defined settings for these analyses, D-VASim may take up to an hour to estimate the threshold value and propagation delays. This estimation time is still reasonable as compared to the number of days of laboratory experimentation that are required for a single combination of inputs and for a specific set of parameters. In future work, we will perform threshold value analysis for each input protein of a genetic circuit separately. We will also include an option in D-VASim to perform these analyses faster by running the algorithm on GPUs (graphical processor units). Moreover, we will continue analyzing the timings of all other circuits[19] with a more detailed set of parameters.

■ METHODS

All of the experiments were performed using stochastic simulations on D-VASim v1.2. The latest version of D-VASim (v1.2) is available to download from http://bda.compute.dtu.dk/downloads. The user manual and video demonstration of D-VASim (including threshold value and timing analysis) can be accessed at http://bda.compute.dtu.dk/user-manuals/.

■ ASSOCIATED CONTENT

Ⓢ Supporting Information

The Supporting Information is available free of charge on the ACS Publications website at DOI: 10.1021/acssynbio.6b00296.

> Complete simulation data and the screen-captured images. The data for each circuit is enclosed in its respective subfolder. Readers are encouraged to go through the "Read Me" file for a detailed explanation of each file and folder (ZIP)

■ AUTHOR INFORMATION

Corresponding Authors

*E-mail: haba@dtu.dk (H.B.).
*E-mail: jama@dtu.dk (J.M.).

ORCID

Hasan Baig: 0000-0002-8765-2604

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

We would like to thank Prof. Chris Myers (University of Utah) for his continuous support and help regarding iBioSim and for providing us with the genetic circuit models in SBML format. We would also like to thank Arash Khoshparvar and Nicholas Roehner from CIDAR Lab (Boston University) for helping us fix issues in the SBOL and SBML files. Further, we thank Alec Nielsen (MIT) for fruitful discussions on his Cello lab experiments.

■ REFERENCES

(1) Lucks, J., and Arkin, A. (2011) The hunt for the biological transistor. *IEEE Spectrum 48* (3), 38−43.

(2) Eisenstein, M. (2016) Living factories of the future. *Nature 531*, 401−403.

(3) Schukur, L., Geering, B., Charpin-El Hamri, G., and Fussenegger, M. (2015) Implantable synthetic cytokine converter cells with AND-gate logic treat experimental psoriasis. *Sci. Transl. Med. 7* (318), 318ra201.

(4) Cases, I., and de Lorenzo, V. (2005) Genetically modified organisms for the environment: Stories of success and failure and what we have learned from them. *Int. Microbiol. 8*, 213−222.

(5) Anderson, J. C., Clarke, E. J., Arkin, A. P., and Voigt, C. A. (2006) Environmentally controlled invasion of cancer cells by engineering bacteria. *J. Mol. Biol. 355*, 619−627.

(6) Ro, D.-K., et al. (2006) Production of the antimalarial drug percursor artemisinic acid in engineered yeast. *Nature 440*, 940−943.

(7) Atsumi, S., and Liao, J. C. (2008) Metabolic engineering for advanced biofuels production from Escherichia coli. *Curr. Opin. Biotechnol. 19* (5), 414−419.

(8) Wang, B., Barahona, M., and Buck, M. (2013) A modular cell-based biosensor using engineered genetic logic circuits to detect and integrate multiple environmental signals. *Biosens. Bioelectron. 40* (1), 368−376.

(9) Ruder, W. C., Lu, T., and Collins, J. J. (2011) Synthetic biology moving into the clinic. *Science 333* (6047), 1248−1252.

(10) Balch, M. (2003) *Complete Digital Design: A comprehensive guide to digital electronic and computer system architecture*, McGraw-Hill Press, New York.

(11) Maini, A. K. (2007) *Digital Electronics: Principles, Devices and Applications*, John Wiley & Sons, Ltd.

(12) Andrianantoandro, E., Basu, S., Karig, D. K., and Weiss, R. (2006) Synthetic biology: new engineering rules for an emerging discipline. *Mol. Syst. Biol. 2* (1), 2006.0028.

(13) Myers, C. J. (2009) *Engineering Genetic Circuits*, Chapman & Hall/CRC Press.

(14) Knight, T. (2003) Idempotent Vector Design for Standard Assembly of Biobricks, *MIT Artificial Intelligence Laboratory*. http://hdl.handle.net/1721.1/21168.

(15) Baig, H., and Madsen, J. (2017) D-VASim − An Interactive Virtual Laboratory Environment for the Simulation and Analysis of Genetic Circuits. *Bioinformatics 32* (20), 297−299.

(16) Hucka, M., et al. (2010) *The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version 1 Core.*

(17) Visvesvara Rao, B., et al. (2007) *Electronic Devices and Circuits*, 2nd ed., Pearson Education.

(18) Myers, C. J., Barker, N., Jones, K., Kuwahara, H., Madsen, C., and Nguyen, N.-P. D. (2009) iBioSim: A tool for the analysis and design of genetic circuits. *Bioinformatics 25* (21), 2848−2849.

(19) Nielsen, A. A., Der, B. S., Shin, J., Vaidyanathan, P., Paralanov, V., Strychalski, E. A., Ross, D., Densmore, D., and Voigt, C. A. (2016) Genetic circuit design automation. *Science 352* (6281), aac7341.

(20) Gillespie, D. T. (1977) Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem. 81* (25), 2340−2361.

(21) Gillespie, D. T. (1976) A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comput. Phys. 22* (4), 403−434.

(22) Bartley, B. (2015) Synthetic Biology Open Language (SBOL) version 2.0.0. *J. Integrative Bioinformat. 12* (2), 272.

(23) Roehner, N., Zhang, Z., Nguyen, T., and Myers, C. J. (2015) Generating Systems Biology Markup Language Models from the Synthetic Biology Open Language. *ACS Synth. Biol. 4* (8), 873−879.