12-2010

An FPGA Based Implementation of the Exact Stochastic Simulation Algorithm

Phani Bharadwaj Vanguri
pvanguri@utk.edu

To the Graduate Council:

I am submitting herewith a thesis written by Phani Bharadwaj Vanguri entitled "An FPGA Based Implementation of the Exact Stochastic Simulation Algorithm." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Engineering.

Dr. Gregory Peterson, Major Professor

We have read this thesis and recommend its acceptance:

Dr. Donald W Bouldin, Dr. Robert Hinde

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:
I am submitting herewith a thesis written by Phani Bharadwaj Vanguri entitled "An FPGA Based Implementation of the Exact Stochastic Simulation Algorithm." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Engineering.

Gregory Peterson, Major Professor

We have read this thesis and
recommend its acceptance:

Donald W. Bouldin

Robert Hinde

Acceptance for the Council:

Carolyn R. Hodges
Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

An FPGA Based Implementation of the Exact Stochastic Simulation Algorithm

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Phani Bharadwaj Vanguri
December 2010

Dedication

This thesis is dedication to my family for all of their support, guidance, and encouragement; and to my dearest friends.

Acknowledgments

I would like to express my gratitude to all of those who have contributed their expertise and directed me towards the completion of my Master of Science degree. First and foremost, I must thank my advisor, Dr. Gregory Peterson, for spending the time and effort to both get me started and guide me down the path towards finishing my studies. I would not be where I am without his knowledge and understanding. Secondly, I would like to express my sincere gratitude to Dr. Donald W. Bouldin and Dr. Robert Hinde for reviewing my work and serving on my committee.

I also express my thanks to all my lab mates and friends, especially David, for assisting me in my work in various ways.

Finally, I owe my heartfelt regards for my family who have constantly been my force of inspiration, determination and encouragement. I extend my special thanks to all my friends for their care and support which has helped me through this arduous task.

Abstract

Mathematical and statistical modeling of biological systems is a desired goal for many years. Many biochemical models are often evaluated using a deterministic approach, which uses differential equations to describe the chemical interactions. However, such an approach is inaccurate for small species populations as it neglects the discrete representation of population values, presents the possibility of negative populations, and does not represent the stochastic nature of biochemical systems. The Stochastic Simulation Algorithm (SSA) developed by Gillespie is able to properly account for these inherent noise fluctuations. Due to the stochastic nature of the Monte Carlo simulations, large numbers of simulations must be run in order to get accurate statistics for the species populations and reactions. However, the algorithm tends to be computationally heavy and leads to long simulation runtimes for large systems. Therefore, this thesis explores implementing the SSA on a Field Programmable Gate Array (FPGA) to improve performance. Employing the Field programmable Gate Arrays exploits the parallelism present in the SSA, providing speedup over the software implementations that execute sequentially. In contrast to prior work that requires re-construction and re-synthesis of the design to simulate a new biochemical system, this work explores the use of reconfigurable hardware in implementing a generic biochemical simulator.

Contents

List of Figures

Chapter 1

Introduction

For several decades, biochemical systems analysis had been studied extensively. Understanding of how different molecular species interact and communicate is critical to predict the behavior of a biochemical system. By learning these interactions, chemists can explore methods that can control or inhibit the course of a reaction. Modeling of complex biochemical systems is one way to obtain this knowledge. Modeling of complex biochemical systems while producing reliable data is a time consuming task and efforts are made to develop such tools more efficient.

Biochemical systems are often formulated using a deterministic approach. In the deterministic approach a system of chemical reactions and species are represented by ordinary differential equations. Each of the species has a differential equation that represents the rate of change of population. Chemists and biologists are able to model numerous biochemical systems using the deterministic approach; however, this approach becomes ineffective for systems with small populations of chemical species as it neglects the discrete representation of population values, presents the possibility of negative populations, and does not represent the stochastic nature of biochemical systems.

In order to over come these limitations a stochastic approach is used. Unlike the many equations that are used in the deterministic approach, a single master equation is used in the stochastic approach. The stochastic approach involves the formulation of a chemical master equation that defines the probability of finding specific species populations at any given point in time. Although the stochastic approach simulates all the models solved by the

deterministic approach and more, this comes with a cost of mathematical and computational complexity. For a complex biochemical system, solving a stochastic approach's chemical master equation becomes impractical due to the large set of differential equations. This leads to the need for computer based methods.

The stochastic simulation algorithm (SSA), developed by Daniel Gillespie, produces statistics mathematically equivalent to the CME [1-3]. In the stochastic simulation algorithm, in order to obtain statistically accurate results, the algorithm is executed several times to form a complete picture. In this case, even the most efficient implementation of the SSA will be very time consuming.

In order to address these issues, this work presents a hardware-accelerated version of the stochastic simulation algorithm on Field Programmable Gate Arrays (FPGA). The overall performance of the stochastic simulation algorithm has been improved by implementing a large number of parallel processing units and by distributing the workload over those units. Previous work with hardware simulators yielded impressive performances; however, these implementations are focused on specific biochemical models and require redesign for different biochemical models.

For this work the main focus is to design a general-purpose hardware accelerated stochastic simulator that works without the need to resynthesize the design. The second chapter of this work gives a more in-depth look into the stochastic simulation algorithm and its advantages. Chapter 3 will describe some of the previous works related to hardware accelerated stochastic simulators. It also includes descriptions of the other methods being used to implement the stochastic simulators. A brief explanation of the disadvantages of these methods is also addressed. Chapter 4 focuses on the hardware architecture used to implement the Block RAM based hardware accelerated stochastic simulator. The algorithm has been implemented in VHDL and the Xilinx Virtex5 family [19] of FPGAs targeted to implement the hardware design. The approach and the design of the algorithm have been discussed in detail. Results are given showing the necessary resources and maximum clock speed after synthesis. Finally, the last chapter presents some plausible avenues for future work, in addition to the conclusions for this work.

Chapter 2

Background

2.1 Biochemical System modeling

One motivation of biochemical system modeling is to learn how the interactions occur during the reactions, so that a way to enhance or interrupt them can be formulated. Once a method is created to modify or control the process, a scientist may be able to control the rest of the reactions any way they want. This means the scientist could create medical breakthroughs for different diseases and viruses. Mathematical modeling for cell mitosis in frog eggs [2] and modeling of phage T7 virus in Escherichia coli [3] are examples for this.

2.2 Chemical Master Equation

Understanding the purpose of modeling is necessary before modeling a biochemical system. The chemical master equation of a chemically reacting system reflects several aspects of the system. This equation is computationally complex and hence extremely difficult to solve. A brief overview of the derivation of the chemical master equation is given below. A detailed explanation of the Chemical Master Equation can be found in [1] [4].

2.2.1 Definitions

Before going ahead with the brief derivation of the chemical master equation a few definitions that are used in the derivation are explained.

Let a set of N species molecules be defined as S, where $S = S_1, S_2, S_3, \ldots, S_N$. S_i gives the species with index i.

Let the species population be denoted by X. X_i denotes the population of species with index i. At any given time t, the population of a species is given by $X_i(t)$, where i is the species index and t is the time.

Let a set of reactions in a network be denoted by R. network with M reactions is given by $R = R_1, R_2, R_3 \ldots .., R_M$, where R_j denotes the reaction with index j.

Let l and k denote the set of reaction coefficients and stochastic rate constants respectively.

Let a set of reactants and products by defined by r and p, respectively r_j and p_j denote the reactants and products in the reaction with index j.

2.2.2 Derivation

Chemical equations of a biochemical system generally come in the form shown below.

$$l_1 r_1 + ... + l_\rho r_\rho \overset{k}{\underset{k_{-1}}{\rightleftarrows}} l'_1 p'_1 + ... + l'_\gamma p'_\gamma \tag{2.1}$$

The purpose of the chemical master equation is to describe the time evolution of all species, $X(t)$, given an initial state $X(t_0)$. Each molecular species in the set of $X(t_i)$ is a random variable that describes the species population. Let $\alpha_j(X)$ denote the probability function that describes the likelihood of a reaction occurring in the next step, called the propensity function. Below is the equation for the propensity calculations.

$$\alpha_j = k_j * \prod_{i \in r_j} nCr(i, l_i) \tag{2.2}$$

Here nCr is the number of combinations of n molecules taken r at a time.

The probability that $X(t)$ will equal to some final population vector x due to this propensity is given by.

$$P(x, t \mid X(0), t_0) \tag{2.3}$$

This is the essence of the chemical master equation. Below is the CME [4] [5].

$$\frac{\partial}{\partial t}P(x,t \mid X(0),t_0) = \sum_{j=0}^{M-1}[\alpha_j(x - v_j,t \mid X(0),t_0) - \alpha_j(x)P(x,t \mid X(0),t_0)] \tag{2.4}$$

Due to its recursive nature and dependency on the previous reaction step taken to achieve the final population vector, this equation is difficult to solve analytically or numerically.

2.3 Different Approaches of Modeling

The analysis of complex biochemical networks is conducted in two popular conceptual frameworks for modeling. The deterministic approach and the stochastic approach are the two popular frameworks for modeling intracellular dynamics. The deterministic modelling is based on the construction of a set of rate equations to describe the reactions in the biochemical pathways of interest. These rate equations are in fact non-linear ordinary differential equations (ODEs) with concentrations of chemical species as variables. Deterministic simulation produces concentrations by solving the ODEs. The stochastic modelling involves the formation of a set of chemical master equations (CMEs) with probabilities as variables. Stochastic simulation, however, produces counts of molecules of some chemical species as realizations of random variables drawn from the probability distribution described by the CMEs.

2.3.1 Ordinary Differential Equations Modeling

Models of chemical species reacting within a spatially homogeneous environment are initially devised using a deterministic approach involving ordinary differential equations. The system in this case is assumed to be spatially homogeneous, meaning that the positions of individual molecules are irrelevant to the outcome of the system. Differential equations are used for each species to express the time rate of change of population as a function of all species populations. Usage of software packages that include differential equation solvers can be used to model a complex biochemical system; however, the results may not necessarily be accurate. This approach ignores the inherent stochastic nature of chemi-

cally reacting systems thereby being inaccurate for models of small populations. Also, the ordinary differential equations do not allow the representation of discrete values for the species populations. Since species with a small population can have a significant impact on the trajectory of the system, these limitations can be devastating to modeling chemical systems.

2.3.2 Gillespie's Stochastic Simulation

Gillespie's stochastic simulation algorithm was developed as a way to accurately simulate chemically reacting systems. Unlike the deterministic approach, there are not many equations; instead, there is only one master difference equation for a probability function. The master equation gives the probability of finding species population at any given point in time. Gillespie' stochastic simulation algorithm treats species populations as discrete values and properly handles the randomness and noise inherent in many chemically reacting systems. Hence this approach handles all the systems that are modeled by the deterministic approach and more.

In Gillespie's approach the initial set of species population and reactions are taken. The probability of a reaction is calculated, and applied to update the species population. This process is continued until the specified end time is reached. Since the approach turns out to be very computationally expensive for a large complex system, researchers have formulated a number of methods to help with this. Gillespie formulated two methods to perform exact stochastic simulations, the First Reaction Method [1][5] and the Direct Method [1][5]. Gibson and Bruck improved upon Gillespie's First Reaction Method in 2000 to develop the Next Reaction Method [6]. The Optimized Direct Method developed in 2004 by Cao, Li, and Petzold further improved the performance of exact stochastic simulations [7]. The Sorting Direct Method, recently developed by McCollum, further optimized stochastic simulations [8]. Each of these methods share the same steps; however, a few different approaches are used in the propensity calculations to reduce the computational complexity. Below are the steps needed to execute the stochastic simulation.

1. Initialization – An input model is read by the simulator and data structures are initialized.

2. Propensity Calculation – Where necessary, the propensity of each reaction is calculated based on the reaction rate constant and the current species populations.

3. Putative Time Estimation – Using the propensities and exponentially distributed random numbers, the time at which the next reaction will occur is determined.

4. Reaction Selection – The next reaction to execute is selected.

5. Reaction Execution – The species populations and system time are updated according to the execution of the selected reaction.

6. Termination – The program ends if the desired end time of the simulation has been reached. Otherwise, the process returns to the Propensity Calculation step and continues executing.

For this work parallel implementation of the direct method is explored. The parallel implementation of the First Reaction Method was not used because of the large amount of random numbers needed. The other methods are not easily parallelizable and are hence not used.

Due to the extensive use of the Direct Method throughout this thesis, a description of the Direct Method is given in the rest of this chapter. Further explanation of the other methods can be found in the cited papers previously mentioned.

Explanation of Terms

A few crucial terms used in formulating the algorithm and may require explanation. The stochastic reaction rate constant, c, is defined as the average probability that a molecular combination from a given reaction will collide and react in the next infinitesimal time interval. The stochastic reaction rate constant is directly related to the deterministic reaction rate constant, k.

The propensity, P, is associated with the probability that a reaction will occur. It is based upon the stochastic reaction rate constant and the number of distinct molecular combinations of reaction. The number of distinct molecular combinations of a reaction depends on the type of reaction and the number of molecules, X, of each reactant of a given

Table 2.1: Propensity Equations.

Reaction Equation	**Propensity Equation**
$A \rightarrow^{k1} B$	$a_0 = k_1 * X_A$
$A + B \rightarrow^{k2} C$	$a_1 = k_2 * X_A * X_B$
$2A \rightarrow^{k3} B$	$a_1 = k_3 * X_A * (X_A - 1)/2$

reaction. Table 2.1 shows the equations for some common reaction types along with the equations to calculate their propensities.

The putative time, τ, refers to the amount of time it will take before a reaction occurs. Other terms specific to the algorithm will be clarified as needed.

2.4 Gillespie's Direct Method

Gillespie formulated the Direct Method of the SSA [9].

Step 1: Initialization: The initialization step of the algorithm creates and loads the data into memory to be used later. The input data contains a vector set of size M containing the species population and a vector of chemical reactions. Each reaction in vector R is defined by a stochastic role constant and stoichiometry representing the reactants and product coefficients.

Step 2: Propensity Calculations: After the initialization the propensity function is be calculated for each reaction. The probability that the given reaction occurs in a given period of time, dt, is given by the following equation,

$$P_i(dt) = \alpha_i(X_i(t))dt + o(dt) \tag{2.5}$$

Where α_i is a propensity function that describes the likelihood of a reaction occurring for species $X_i(t)$. The propensity function is the product of the reaction rate constant and the number of ways the reaction can occur depending on the populations of reactants. Using the dimerization example from McCollum et. al. [8], an example reaction could be in the form of $S_4 + S_4 \rightarrow S_5$ and the propensity would be calculated by

$$\alpha_j(X(t)) = k_j \frac{X_4(t)(X_4(t) - 1)}{2} \tag{2.6}$$

Step 3: Reaction Time Generation: The occurrence of the time of the next reaction can be calculated by dividing an exponential distributed random number by the sum of the propensities from the previous step, as shown in the equation below.

$$\tau_{next} = \frac{-\ln(UNIFORMRAND)}{\sum_{i=1}^{N} \alpha_i(X(t))} \tag{2.7}$$

Step 4: Reaction Selection: The next reaction that will occur is selected in this step. The sum of the propensities is multiplied by a uniformly distributed random number. This value is then incrementally subtracted by the propensity of each reaction until the value equals or is less than 0. The index of the reaction whose propensity causes the value to reach this bound is the index of the next reaction.

Step 5: Reaction Execution: The reaction is executed by updating the species population. To do this, each reactant's coefficient is subtracted and each product's coefficient is added to the population. This gives the updated population values that represent $X(t)$. This is given in the equation below.

$$X(t) = X(t-1) - \sum_{i \in SelectedReactants} r_i + \sum_{i \in SelectedProducts} p_i \tag{2.8}$$

Step 6: Termination: At this step, if the current calculated time has not reached the specified end time, steps 2 through 5 are repeated. If the current calculated time is the same or larger than the upper bound tend, the program will terminate.

2.5 Field Programmable Gate Arrays

FPGA stands for Field Programmable Gate Array, which are reusable logic devices. An FPGA is a combination of a number of logic cells. Each logic cell is a combination of a look-up table, a D flip-flop, and a 2-to-1 multiplexer. The look-up table is similar to a small RAM and typically has 4 inputs. Arrays of logic cells contain logic gates, along with memory blocks, to form the underlying flexible fabric for FPGA integrated circuits. Testing a design is simplified on an FPGA, since it can be electronically programmed, erased, and then reprogrammed in a short amount of time. This is also the basis for using FPGAs in

Algorithm 2.1: Pseudocode for the Direct Method [17]

```
CurrentTime = 0.0                                          1.Initialization
X[1....M] = Initial Species Populations
R[1.....N] = Reactions
TotalPropensity = 0.0
for I = 1....N do                                          2. Propensity Calculation
    Prop[I] = CalcPropensity(S,R[I])
    TotalPropensity += Prop[I]
end for
T = -ln(rand())/TotalPropensity                            3. Reaction Time Generation
Selector = TotalPropensity * rand()                        4. Reaction Selection
for I = 1....N do
    Selector - Prop[I]
    if  Selector ≤ 0 then
    SelRxn = I
    endif
end for
X = X - R[SelRxn].reactants + R[SelRxn].products           5.Reaction Execution
CurrentTime +=T
if  CurrentTime < EndTime then                             6. Termination
    GOTO Propensity Calculation
endif
```

reconfigurable computing. A Hardware Description Language (HDL) or a schematic is used to define the desired functionality of an FPGA. Codes written in a Hardware Description Language (HDL), such as VHDL and Verilog, are synthesized and mapped to the devices, facilitating the mapping to the FPGA. FPGA device density currently exceeds three million logic gates with operational clock rate of 200 Megahertz in Xilinx Virtex 5 [19]. Today, the FPGA device densities are in millions of logic gates with synthesized logic capable of running at rates up to and exceeding 500MHz. Apart from an increase in logic density, the inclusion of high performance embedded arithmetic units, large amounts of memory, and high speed processor cores have facilitated the growth of FPGA integrated circuits beyond a simple prototyping devices or for "glue logic".

Chapter 3

Related Work

This chapter will provide an overview of the work done by others towards a hardware accelerated stochastic simulator. Most of the work done in the past focused on simulating specific biochemical models. Hence, a change of the biochemical system would require a re-synthesis and implementation. The first work that is considered is the work of Yoshimi, Osana, Fukushima, and Amano. The work of Salwinski and Eisenberg is considered next. Finally the work done by Keane, Bradley, and Ebeling is examined.

3.1 Yoshimi, Osana, Fukushima and Amano's FPGA Simulation

Yoshimi, Osana, Fukushima, and Amano's FPGA simulation approach takes advantage of the fine grain parallelism inherent in biological systems [10]. They developed a reconfigurable platform (ReCSiP) on a Xilinx Virtex II FPGA interfaced to a host CPU via a PCI bus. They modeled the Lotka system outline in Gillespie's original paper on exact stochastic simulation [1, 9], to show the performance of their simulator. The module designed to simulate the Lotka system consisted of two simulator modules, each containing two reactor modules, and a module to handle output control. In order to speed up the putative time calculation, look up tables (LUT) of logarithmic values are used. All reactor modules have a portion dedicated to basic tasks that are common to all simulations executing Gillespie's SSA. The other main portion of each reactor contains the specifics of the model being simu-

lated and must be replaced with each new model. The output is transferred to SRAM after being buffered into a FIFO. The design used is written in Verilog and needs re-synthesis if a different biochemical model is used. They developed a 37 stage pipelined reactor to allow 37 simulation processes to be executed in parallel. The design takes 37 clock cycles to output updated species values, for a speedup of 105 over a software implementation. The speedup was not based on the actual simulation run time; they choose to compare the throughput of the hardware with the software. They failed to specify the algorithm used in their implementation; in addition they failed to include details of their design. The design doesn't require redesign on the users part and hence was considered general purpose.

3.2 Keane, Bradley and Ebeling's FPGA Approximation

John Keane, Christopher Bradley, and Carl Ebeling developed an algorithm that approximates Gillespie's SSA. It reveals a fine-grained parallel structure that is well suited to a hardware implementation [11]. Initial works of this team shows work on implementing Gibson and Bruck's Next Reaction Method [6]; however, they realized that the complexity involved in the algorithm makes it unsuitable for the FPGA. In order to use the parallelism capabilities of the FPGA they devised a strategy to modify Gillespie's Direct Method. Each reaction is handled to be simulated simultaneously and is permitted at discrete instants of time. A Bernoulli random process was used to replace a Poisson process, and the propensity of the reaction was associated with the probability of an event at any given discrete time step. The equation below represents each reaction's propensity.

$$P[X_0 < K_0] \cdot P[X_1 < S_1] \cdot P[X_2 < S_2] = kS_1S_2. \triangle t \tag{3.1}$$

The need to determine the next reaction and summing the propensities was eliminated since reaction propensity is now based on the probability of a reaction to occur during a given time step. The approach allowed only one dependent reaction to occur in each time step. In the event of collision the hardware paused and waited for the software to resolve the issue. Although the example design was limited to second order systems, they indicated their approach would generalize to higher order systems. Since each new design needs to be

described, synthesized, and routed, a compiler that reads a SBML model to translate to a Verilog file is developed. The approach compared several models of varying sizes simulated using their FPGA approach with software running the Next Reaction Method. A speedup of 23 was achieved simulating a system containing 4 species and 32 reactions. Speed up was defined based on the average number of reaction events computed per second. An estimate was used to determine the event rate, in addition to two sources of overhead, data logging communications and recalculations at the time of collisions. They chose to ignore the I/O communication overhead despite revealing that it accounts for nearly 70% of the simulation time. In addition the design is a statistically equivalent representation of Gillespie's SSA.

3.3 Salwinski and Eisenberg's FPGA Approximation

Lukasz Salwinski and David Eisenberg examined the use of an FPGA to exploit the highly parallel nature of biochemical networks [12][13]. This approach takes advantage of parallelism to alleviate the high computational complexity of stochastic simulations. There are drawbacks to this approach; it was formulated to simulate specific models and also the approximations used in their hardware implementation are not true to Gillespie's original SSA. After simulating a system containing a single elementary bimolecular reaction and a system containing a simple equilibrium reaction, they tested the scalability of their approach. They proposed simulation rates at least an order of magnitude greater than the software counterpart. Their work served as a proof-of-principle that reprogrammable FPGAs have the potential to efficiently simulate the stochastic behavior of biological systems.

Chapter 4

Implementation

This chapter discusses the Block RAM based FPGA implementation of the stochastic simulation algorithm. First, background information of SSE is presented and then the implementation is examined. Finally, the results of this implementation are given and compared.

4.1 Partitioning of problem

Several questions arose, when the implementation of the hardware accelerated general-purpose simulator was considered. Deciding on the most effective stochastic simulation algorithm was the first step. The Next Reaction Method was avoided due to its difficulty to implement in hardware. The First Reaction Method requires a large number of random number generators and hence is not an effective algorithm to be implemented in hardware. So Gillespie's original Direct Method looks like the obvious choice for the implementation.

Once the algorithm is selected, the next step was to divide the tasks depending upon whether they are needed to be implemented in hardware or software. The initial data transfer of the species population, stoichiometry, and reaction constants are handled by the software. The FPGA handled the calculation and summation of propensities, random number generation, selection of next reaction, and the species update. This is a suitable choice as the propensity calculation and random number generation are tasks that require floating point arithmetic and can be highly parallelized.

4.2 Implementation Resources

This section describes the software design tools used for this work such as VHDL design, synthesis and place, and route. This section also describes the hardware resources.

4.2.1 VHDL Designs

ModelSim SE 6.5 was used for debugging the VHDL implementations of the block RAM based SSA. ModelSim is compatible with Xilinx hardware libraries, such as COREGEN models [20]. This was used for the compilation and the pre synthesis simulations.

4.2.2 Synthesis

Once the design was verified and pre synthesized through the ModelSim, the VHDL source code was synthesized using the Xilinx Synthesis tool (XST 11.4). In this process RTL level descriptions were converted to gate level descriptions.

4.2.3 Place and Route

After the synthesis process, the modules go through the Place And Route (PAR 11.4) process. The Xilinx PAR tool is employed for all the implementations. The PAR tool reports the maximum allowable frequency for each generator.

4.2.4 Xilinx University Program - XUPV5-LX110T Board

Xilinx University Program - XUPV5-LX110T Board is used for the implementation. The XUPV505-LX110T is a feature-rich general purpose evaluation and development platform with on-board memory and industry standard connectivity interfaces. It features the Virtex-5 XC5VLX110T device and provides up to 11.6Mbits of flexible embedded block RAM.

4.3 Hardware Implementation

We now present a Hardware realization of the block RAM based SSA design. Hardware design is a wide-open field with many different ways of implementing architectures, which

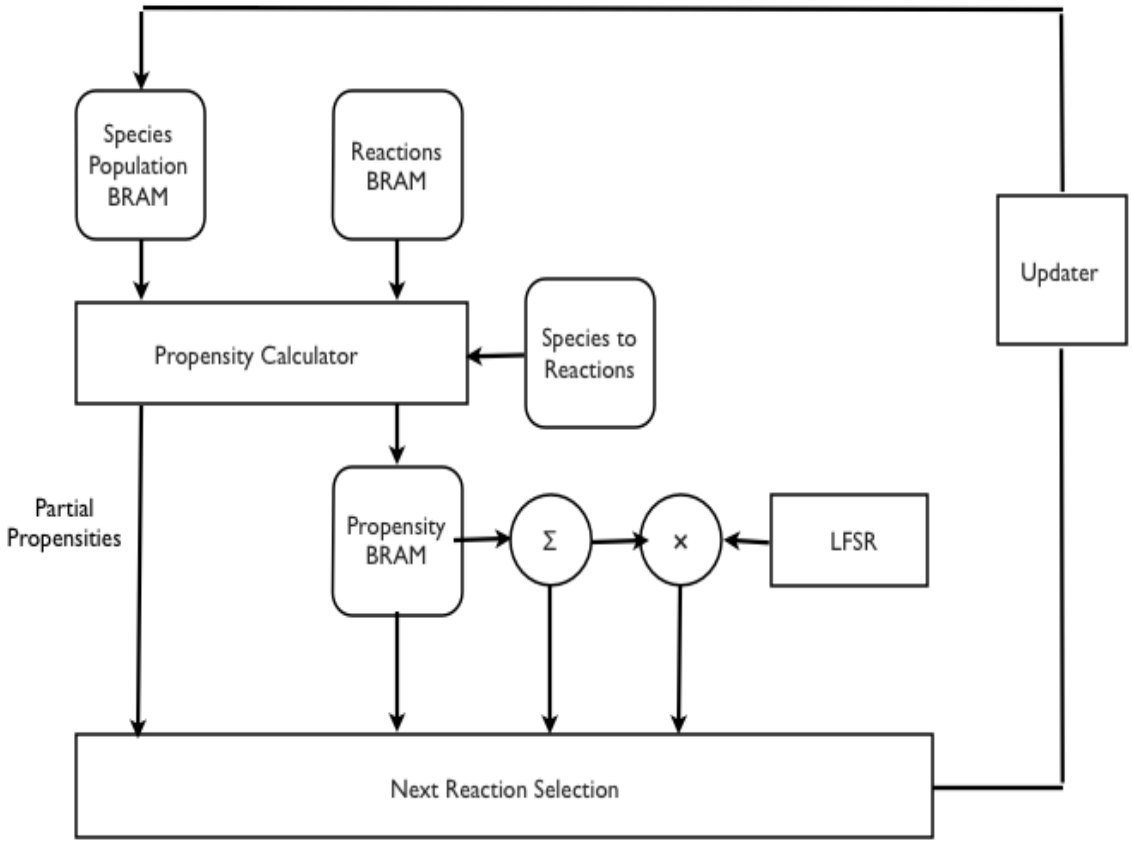

Figure 4-1: Block diagram of the hardware design

can often have a large impact on performance. The architecture is created in VHDL, targeted for the Xilinx Virtex-5 series of FPGAs, and consists of the following components:

- Block RAM Memory
- Propensity Calculator
- Species to Reaction Lookup
- Random Number Generator
- Next Reaction Selector and Species Updater

Each of these components are discussed next.

Table 4.1: Propensity Equations.

Reactants	**Propensity Equation**
A	KA
$2A$	$K(A)(A-1)/2$
$A+B$	$K(A)(B)$
$2A+B$	$K(A)(A-1)(B)/2$
$A+2B$	$K(A)(B)(B-1)/2$
$2A+2B$	$K(A)(A-1)(B)(B-1)/4$

4.3.1 Block RAM Memory

The Xilinx Virtex 5 FPGA (XUPV5-LX110T) [19] provides up to 11.6Mbits of flexible embedded block RAM that can efficiently store and buffer data without off-chip memory. Each memory block stores up to 36Kbits of data and can be configured as either two independent 18Kb Block RAMs, or a 36Kb Block RAM. The BRAM available on the FPGA was formatted to allow systems with a maximum of 32 species and 64 reaction equations to be simulated. In addition to that the BRAM was formatted to hold the resulting propensity associated with each reaction equation. A total of 5 block RAMs are used in the design. They are :

- Species Population
- Reaction Constant
- STOIC (the coefficients and the species index)
- Propensity
- Species to Reaction Lookup

4.3.2 Propensity Calculator

The propensity calculator component was designed such that it can calculate the propensity of any equation with a maximum of two reactants and a maximum reaction coefficient of two. The propensity calculator collects data from the STOIC, checks BRAM for the reaction type and then calculates the propensity depending on one of the 6 ways shown in Table 4.1.

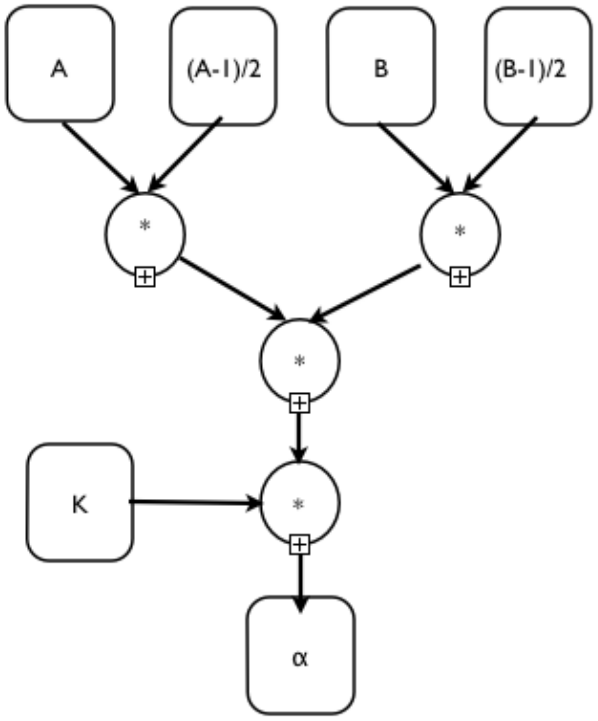

Figure 4-2: Block Diagram of Propensity Calculator

4.3.3 Species to Reaction Lookup

A species to reaction lookup table is created in order to calculate only those reactions that are affected by the next reaction selection. A lookup containing the species to reactions is pre compiled and loaded into the BRAM. This BRAM is 64 bits wide and 32 bits long. Each species is represented using 64 bit data. If the species is present as a reactant in a reaction the corresponding bit is 1 or else it is left as 0. Once we get the next reaction and update the species count, the reactions that are affected by these species are obtained using this lookup.

A maximum of 4 species can be changed in any reaction selection. The corresponding reactions dependent on each of these 4 species are taken and ORed to obtain the set of reactions that are affected. Now the propensity calculation of only these reactions is calculated instead of 64 reactions. Figure 4.3 gives a example of how the species to reaction lookup is maintained.

Considering the species to reaction lookup example provided in Figure 4.3 as example,

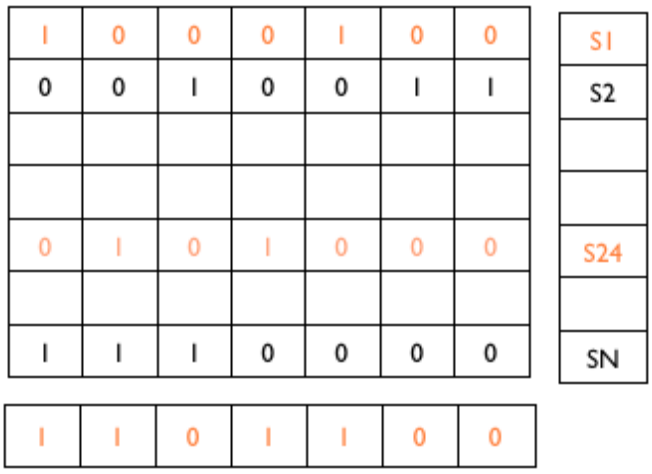

Figure 4-3: Species to reaction lookup example

consider that species S1 and S24 populations are changed due to the reaction selection. Now from Figure 4.3 we can see that species S1 affects only reactions 1 and 5. Similarly species S24 affects only reactions 2 and 4. Now the lookup for the species S1 and S24 are taken and ORed, that gives us that reactions 1,2,4, and 5 are affected by the reaction selection. Hence, only the propensities for reactions 1,2,4, and 5 are calculated instead of all.

4.3.4 Random Number Generator

The intellectual property (IP) block used to generate each random number was developed by McCollum [l5]. It uses a Linear Feedback Shift Register. This module will be replaced by the random number generator IP developed by Lee [14] which uses Hardware Accelerated Scalable Parallel Random Number Generation (HSPRNG).

4.3.5 Next Reaction Selector and Species Updater

Once the propensity is calculated, the next reaction selector reads the propensity sequentially in order to determine the next reaction. Once the reaction is determined the species updater updates the species population according to the reactants and products of the reaction selected. This stage the affected species populations were updated to reflect the any change pertinent in species populations.

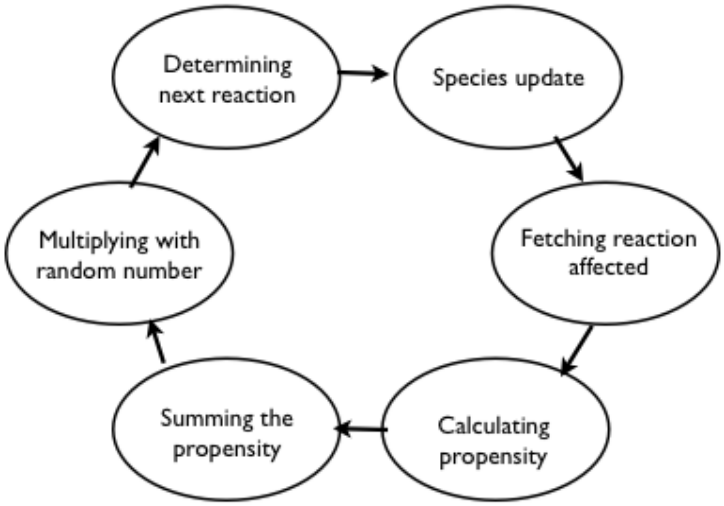

Figure 4-4: Design flow of the implementation

4.4 Algorithm Description

The following is a detailed description of the overall process of the block RAM based stochastic simulation algorithm.

1. Firstly, the Block RAM memory is populated with the species population, reaction constant, species to reactions table, and the stoichiometry reaction equations.

2. Next, the propensities are calculated. This is typically the most expensive step due to the number of multiplications needed for each reaction. To do these calculations for all the simulations, the rate constant is first loaded followed by the species population and the reaction rate constant.

3. Once all the propensities have been calculated and summed together, the sum is then stored back to memory to be used in later calculations. It takes two clock cycles to read from the BRAM, therefore the propensity calculator is pipelined such that the sequential data input gives an output every cycle. Once the propensity is calculated, it must then be summed with the other propensities.

4. Next the reaction selection step is executed; the random numbers generated by the IP developed by McCollum [15] is multiplied with the sum of the propensities.

5. Once the propensity sum and the random number are multiplied the next reaction is selected by subtracting the propensities taken from the BRAM. The reactant and product species that are affected by the reaction selected are updated in the species population block RAM.

6. This process goes for only the first reaction. From this point on only the reactions that are affected by the species that have been updated are considered, as the propensity of the remaining reactions remains unaltered.

7. The reactions that are affected by the species are taken from the species to reaction block RAM. Once the reactions are found, propensities of only the selected reactions are calculated. As propensity calculation is a computationally complex and time taking step, it saves a lot of time to only compute the necessary updates.

8. The sum of the propensities is calculated simultaneously along with the propensity calculation. Once the sum of the propensities is calculated, the next reaction step is executed and the process repeats until the desired simulation end time is reached.

4.5 Results

For the performance, hardware usage and maximum clock frequency are described separately. The maximum clock frequency after the place and route and the performance for the implementations are described.

4.5.1 Synthesis Results

Since FPGA hardware resources are limited, the hardware usage is an interesting area for the results. Even though performance is considerably faster, it is meaningless unless it fits into the FPGA. The design was simulated using Xilinx ISE 11.4 and synthesized using Xilinx Synthesis Technology (XST), targeted for the Xilinx Virtex-5 XUPV5-LX110T Board. A measurement of device resource utilization in terms of slices is given in Figure 4.5. The Figure 4.5 shows the results for design that supports 32 species and 64 reactions.

```
Selected Device : 5vlx110tff1136-3


Slice Logic Utilization:
 Number of Slice Registers:            5082  out of  69120     7%
 Number of Slice LUTs:                 5294  out of  69120     7%
    Number used as Logic:              5094  out of  69120     7%
    Number used as Memory:              200  out of  17920     1%
       Number used as SRL:              200

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:   5537
   Number with an unused Flip Flop:     455  out of   5537     8%
   Number with an unused LUT:           243  out of   5537     4%
   Number of fully used LUT-FF pairs:  4839  out of   5537    87%
   Number of unique control sets:        22

IO Utilization:
 Number of IOs:                         172
 Number of bonded IOBs:                 172  out of    640    26%

Specific Feature Utilization:
 Number of Block RAM/FIFO:                4  out of    148     2%
    Number using Block RAM only:          4
 Number of BUFG/BUFGCTRLs:                1  out of     32     3%
 Number of DSP48Es:                       6  out of     64     9%
```

Figure 4-5: Resource utilization for a single core of 32 species 64 reactions

As the 32 species 64 reactions block occupies only 7% of the slices and 2% of block RAMs, multiple copies of this block can be placed on the board to improve the performance. It is conceivable that we could implement multiple copies of the simulator block the block but the limiting factor would be the number of DSP48Es that are available. As we can see from Figure 4.5 basic block of the simulator occupies around 9% of the DSPs, so a maximum of 10 simulation blocks can be implemented.

4.5.2 Maximum Clock Frequency

Xilinx reports the maximum allowable clock cycle for a module after the place and route process. Table 4.2 shows the maximum allowable clock rates for the design. The limiting

Table 4.2: Maximum Allowable Clock Rates.

Minimum Period	6.116 ns
Maximum Frequency	163.5 MHz

factor of the clock is the propensity calculator block. As the propensity calculator is the most complex block with four multipliers and one divider, it is the critical path of the simulation. The propensity block is highly optimized to reduce this critical path time period.

4.5.3 Performance results

In order to get the performance results of the design the total number of clock cycles required to process a single reaction is calculated. The total number of clock cycles to process a single reaction is given by the sum of the clock cycles taken for propensity calculation, summing the total propensity, multiplication of propensity sum by the random number generator, reaction selection, and species update. The total number of clock cycles taken for one reaction is the sum of the following:

- Propensity Calculation (varies depending on the model) : 0 to N
- Summing the total propensity : $\log_2$N cycles
- Multiplying propensity with random number : 8 cycles
- Next reaction selection: log2N cycles
- Species to reaction lookup data read : 6 cycles
- Max 4 species need to be updated for a reaction : 12 cycles
- Reading species population from BRAM : 2 cycles
- Updating the updated species populations : 1 cycle
- Other latency and data operations : 14 cycles

Individual propensity calculation depends mostly on the number of reactions affected by the species that have been updated. This may vary from 0 to N, where N is the number

Table 4.3: Number of clocks required for single reaction.

Reaction Count	Max number of clock cycles	Min number of clock cycles
8	54	48
16	64	48
32	82	50
64	116	52

of reactions. Taking the worst-case scenario into consideration, it takes N clock cycles for this step. Next, in order to calculate the total propensity from the individual reaction propensities, a summing tree structure that takes $\log_2(\mathrm{N})$ is used. Once the total propensity is calculated it takes eight clock cycles to multiply the total propensity by the generated random number. The reaction selection stage consumes $\log_2(\mathrm{N})$ clock cycles. Finally reading the species from the species population memory requires two clock cycles. Each update to the species population requires one clock cycles. A maximum of four species can be affected in a reaction (two reactants and two products). Initial reading of the reactions and the corresponding species populations from the memory before starting with the propensity calculations and other latency clocks add up to another 20 clock cycles. Therefore, the total number of clock cycles required to process a single reaction is given by the following equations.

$$Maximum\ Clock\ Cycles = N + \log_2 N + 8 + \log_2 N + 8 + 4 + 20 = 2\log_2 N + 40 + N \quad (4.1)$$

$$Minimum\ Clock\ Cycles = \log_2 N + 8 + \log_2 N + 8 + 4 + 20 = 2\log_2 N + 40 \quad (4.2)$$

Table 4.3 shows the estimated number of clocks required to process a single reaction for 8, 16, 32, and 64 reactions. The estimated number of reactions executed per second for each design is given in Table 4.4.

4.5.4 Performance Comparisons

In order to test this hardware approach to stochastic simulation the Direct Method stochastic simulation algorithm was utilized. The algorithm chosen for comparison was the Acceler-

Table 4.4: Hardware Execution Rates.

Reaction Count	Min Execution Rate	Max Execution Rate
8	3 MR/s	3.5 MR/s
16	2.5 MR/s	3.4 MR/s
32	2.0 MR/s	3.25 MR/s
64	1.4 MR/s	3.15 MR/s

Table 4.5: Execution rate comparision and Speedup on System - 1

Model	AESS Rate	Hardware Rate	Speedup
HSR	1.6 MR/s	2.7 MR/s	1.7
Diffusion	1.3 MR/s	2.95 MR/s	2.3

ated Exact Stochastic Simulation Algorithm Direct Method. The software implementation was developed by Jenkins [16]. The software algorithm was compiled and executed on two different systems. The first one is a Core 2 Duo 2.26 GHz system with 3 MB cache and 5 GB RAM. The second one is a Quad Core 2.45 GHz system with 3 MB cache and 8 GB RAM. The performance of each, when given identical chemically reacting systems, was compared to the hardware version. In the following tables, the hardware version is labeled “Hardware Direct". Speedup values were calculated by dividing the execution rate of the software method by the execution rate of the hardware method. Two biochemical systems were chosen to use as models to calculate the resulting speedup of the hardware implementation.

The chemical systems considered for the comparison are the Heat Stress Response model [18] and Diffusion model [18]. Both these models are implemented on hardware as well as the software counterparts. The execution rates for each implementation along with the speedup is provided in Tables 4.5 and 4.6. It is clear from Figures 4.6 and 4.7 that simulating chemically reacting systems on an FPGA provides a speedup over methods executed in software. Furthermore, these results illustrate that employing FPGAs in stochastic simulation is an effective way to accelerate the simulation of useful biological systems.

Table 4.6: Execution rate comparision and Speedup on System - 2

Model	AESS Rate	Hardware Rate	Speedup
HSR	1.8 MR/s	2.7 MR/s	1.5
Diffusion	2 MR/s	2.95 MR/s	1.47

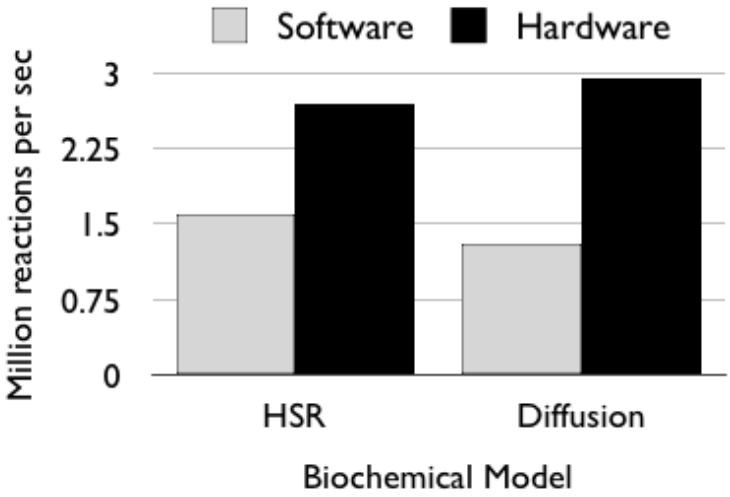

Figure 4-6: Speedup using system 1

4.5.5 Multiple Stochastic Simulation Cores on an FPGA

Looking again at Figure 4.5, we notice that the 32 species 64-reaction model consumes only 7% of the chip's slices. It is conceivable that we could implement multiple copies of our stochastic simulator core on the chip and could simulate multiple independent runs simultaneously.

To test this theory, a design is implemented that used six of the 32 species and 64 reactions. This new multicore design uses 55% of the available resources on XUPV5-LX110T Board. Prototyping this design produces identical simulation rate results to the single-core system for each independent simulation run, 1.75 million reactions per second, but since we are able to perform 6 independent simulations in parallel, the multi-core design is able to generate a total of 8.4 million reactions per second.

4.6 Conclusion

Prior work in hardware implementation of stochastic simulation has provided insights into how we may replace software simulation with hardware simulation, but have not sufficiently addressed the issue of re-synthesis for each model. The approach shown in this chapter provides a framework for allowing the chip to be reconfigured to simulate multiple models

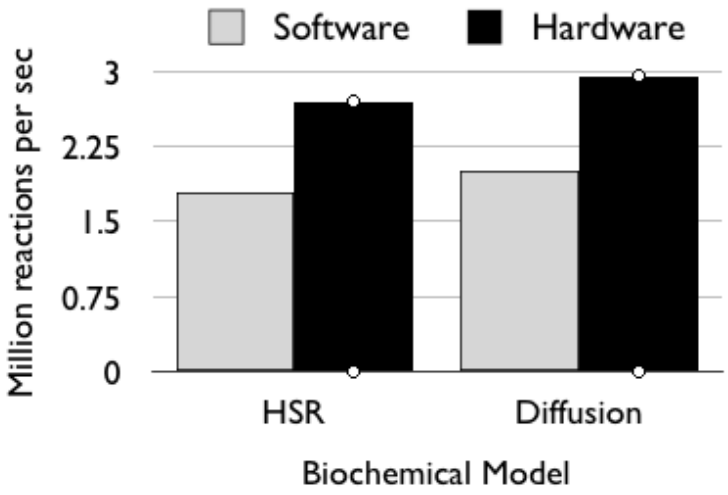

Figure 4-7: Speedup using system 2

without re-synthesis. A direct comparison of a hardware implementation of the system shows a range of 1.47X to 2.3X performance improvement over a software simulator for a single simulation. By implementing multiple cores on an FPGA, it may be possible to achieve larger performance improvements.

Chapter 5

Conclusions

5.1 Thesis Summary

This thesis presents a fast, flexible, generic hardware implementation of Gillespie's Stochastic Simulation Algorithm. By using the Block RAM based memory architecture, most of the data storage has shifted to BRAM, which offers increased flexibility in implementation and good control in the balance between speed and area. The hardware implementation of the block RAM based SSA has shown a speedup over the stochastic simulation algorithm implemented in software. Although the hardware design outperforms its software counterpart, work still remains to develop an optimized hardware design. The topics listed below for future work could direct new hardware designs towards an optimized solution. This could allow biological researchers to accurately model biochemical systems in order to develop the gene therapy and drugs of tomorrow.

The use of FPGAs to accelerate the simulation of biological models appears to be a plausible option. This work, along with related work has shown that FPGAs can play an important part towards speeding up the simulation times of biological models. However, a general-purpose hardware design is essential in order for biologists to consider using an FPGA for stochastic simulations. This was precisely the goal of the thesis.

5.2 Future Work

Much of the possible future work involves further optimizing the design in hardware. For example, implementation of some other stochastic simulation algorithm may provide a more optimum and better approach. Future algorithms may introduce enhancements that are readily adaptable to hardware. Even though this work accelerates the stochastic simulation, also with some changes such as implementing more parallel propensity calculators, better reaction selection modules, and better memory management can further enhance the performance of the design. Also, GPU and MPI implementations may provide a substantial speed up for the generic implementation of the stochastic simulations. Finally, the hardware acceleration framework may motivate enhancements of the SSA to facilitate faster execution.

Bibliography

1. D. T. Gillespie, "General Method for Numerically Simulating Stochastic Time Evolution of Coupled Chemical Reactions," *Journal of Computational Physics*, vol. 22, pp. 403-434, 1976.

2. G. Marlovits, C. J. Tyson, B. Novak and J. J. Tyson, "Modeling M-phase control in Xenopus oocyte extracts: the surveillance mechanism for unreplicated DNA", *Biophysical Chemistry*, vol. 72, pp. 179-184, 1998.

3. D. Endy, D. Kong, and J. Yin, "Intracellular kinetics of a growing virus: A genetically structured simulation for bacteriophage T7", *Biotechnology and Bioengineering*, vol. 55, pp. 375-389, 1997.

4. D. T. Gillespie, "A rigorous derivation of the chemical master equation", *Physica A: Statistical and Theoretical Physics*, vol. 188, no. 1-3, pp. 404-425, September 1992.

5. D. T. Gillespie, "Approximate Accelerated Stochastic Simulation of Chemically Reacting Systems", *Journal of Physical Chemistry*, vol. 115, pp. 1716-1733, 2001.

6. M. Gibson and J. Bruck, "Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels," *Journal of Physical Chemistry A*, vol. 104, pp. 1876-1889, 2000.

7. Y. Cao, H. Li, and L. R. Petzold, "Efficient Formulation of the Stochastic Simulation Algorithm for Chemically Reacting Systems," *Journal of Chemical Physics,* vol. 121, pp. 4059-4067, Sep 2004.

8. J. McCollum, C. Cox, G. Peterson, M. Simpson and N. Samatova, "The Sorting Direct Method: An Efficient Stochastic Simulation Algorithm for Modeling Biochemical Systems," *The Journal Computational Biology and Chemistry*, 2005.v

9. D. T. Gillespie, "Exact Stochastic Simulation of Coupled Chemical Reactions," *Journal of Physical Chemistry*, vol. 81, pp. 2340-2361, 1977.

10. M. Yoshimi, Y. Osana, T. Fukushima, H. Amano, "Stochastic Simulation for Biochemical Reactions on FPGA," *Field-Programmable Logic and Applications, Proceedings Lecture Notes in Computer Science*, vol. 3203, pp. 105-114, 2004.

11. J. F. Keane, C. Bradley, and C. Ebeling, “A Compiled Accelerator for Biological Cell Signaling Simulations,” *FPGA* 2004, pp. 233-241, 2004.

12. L. Salwinski and D. Eisenberg, “In silico simulation of biological network dynamics,” *Nature Biotechnology*, vol. 22, pp. 1017-1019, August 2004.

13. L. Lok, “The need for speed in stochastic simulation,” *Nature Biotechnology*, vol. 22, pp. 964-965, 2004.

14. Lee, Junkyu, "Hardware Accelerated Scalable Parallel Random Number Generation. " Master’s Thesis, Department of Electrical Engineering and Computer Science, University of Tennessee, 2007.http://trace.tennessee.edu/utkgradthes/163

15. J. McCollum, J. Lancaster, D. W. Bouldin and G. D. Peterson, “Hardware Acceleration of Pseudo-Random Number Generation for Simulation Applications,” *Proceedings, Southeastern Symposium on System Theory*, 2003.

16. Jenkins, David Dewayne, "Accelerating the Stochastic Simulation Algorithm Using Emerging Architectures. " Master’s Thesis, Department of Electrical Engineering and Computer Science, University of Tennessee, 2009.

17. J. M. McCollum, "Accelerating Exact Stochastic Simulation", MS thesis, Department of Electrical Engineering and Computer Science, University of Tennessee. 2004.

18. J. M. McCollum, "Accelerating Exact Stochastic Simulation of Biochemical Systems", Ph.D. thesis, Department of Electrical Engineering and Computer Science, University of Tennessee. 2006.

19. Xilinx Inc, "ML505/ML506/ML507 Evaluation Platform User Guide", Oct 2009 [Online], http://xilinx.com/products/boards/ml505/docs.htm.

20. Xilinx Inc, "ISE In-depth Tutorial", Sep 2010 [Oniline], http://xilinx.com/support/documentation/dt_ise12-3.htm.

Vita

Phani Bharadwaj Vanguri was born on 8 August 1985 in Bhimavaram, India. He spent his childhood in Bhimavaram. After graduating from Aditya Junior College in 2002, he attended National Institute of Technology in Kurukshetra. He recieved his Bachelor of Technology in Electronics and communication Engineering in May of 2006. In Jan 2007 Phani was hired by Cognizant Technology Solutions,Hyderabad as a software Programmer in JAVA. In Jan 2008, Phani started his Masters degree at University of Tennessee,Knoxville. During his Masters he interned with Siemens Medical Solutions. He recieved his Master of Science in Computer Engineering in December of 2010.