FPGA of Acceleration of Stochastic simulation

Tian Gao(tg293)

School of Electrical and Computer Engineering, Cornell University

ABSTRACT

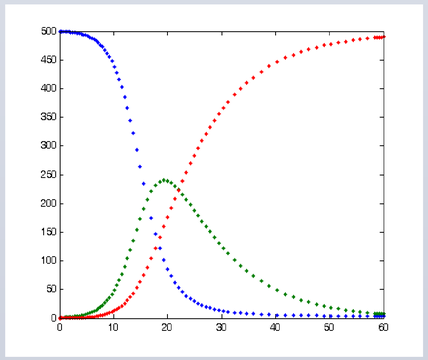
- Simulation of stochastic model like SIR model is very useful. However, the software simulation of this kind of model becomes so slow with the individual number increasing, so this project provides a much faster hardware solution for it.
- A simulation is implemented on MATLAB to test and verify the mathematical algorithm
- The final project is implemented on DE-115 board using Verilog, which gives a much faster performance than on PC
- To make the Verilog code more flexible and variable, a MATLAB program is designed to automatically generate the Verilog code

BACKGROUND

- In SIR model, each individual has to be one of the three stages: S(Susceptible), I(Infectious) or R(Recovered)
- Once the individual is recovered, it can't be infected again

- We can abstract the real world local relationship network to a simplified one, where each individual is connected to all the closely related ones, which means they can infect others through connections
- Initially, one or several individuals are set to be infectious and the transition rates(possibility for infecting and recovering) are fixed

- If we observe the number of individuals of each state, assuming all the individuals are randomly connected, we should expect a theoretical result as below
- The green one, which indicates the number of infected individuals, is the data that we observe in the actual simulation



Design and Implement

- Instead of solving the problem in continuous domain, we use discrete domain which is compatible with hardware system. Moreover, we use a uniform time step for simulation, which lead to a fixed infecting and recovering probability for every step.
- The most important component for this project is a high performance random number generator(RNG). In this case, we use a 64-bit XOR feedback shift register as the RNG, which is easy to implement in FPGA.
- Five parameters are set to describe the SIR model

Probability of Infecting | **Probability of Recovering**

Individual Number | **Average Connections** | **Infected Number**

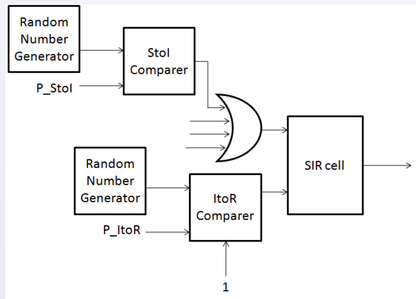
- Since we need a easily changeable code for implementing, a MATLAB program is designed with which users can change just the parameters and get auto-generated Verilog code.

MATLAB
Generate a random network
Set initial infected situation
Generate random seeds for RNG
Write Verilog code
based on given parameters

→

FPGA
Implement hardware network
Compute with a uniform clock
Result is determined by RNG
Add the infected individuals up
Display on VGA screen

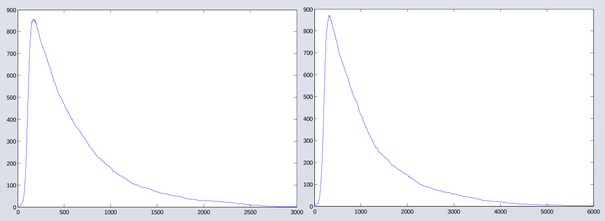
- The hardware realization for one individual is as follows



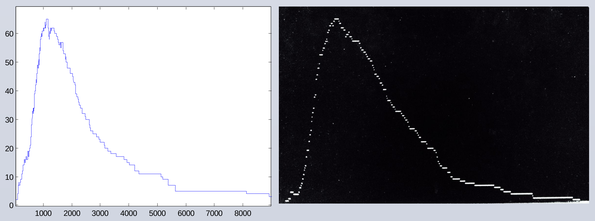
- If SIR cell is in S state, any connected infected individual could infect it with specified random number
- If SIR cell is in I state, it could recover with specified random number
- P_Stol and P_ItoR are fixed based on parameters, if RNG gives out a number less than fixed P, the comparer gives out 1 which enables the next stage.

RESULTS and VERIFICATION

- First of all, we test the algorithm in MATLAB simulation
- MATLAB simulation gives out a similar result as theoretical one which has a sharp rising and a slow decaying.
- If we double the possibility for both infecting and recovering, the shapes of the figures stay similar but the time steps decrease a half, which shows the validation of the algorithm

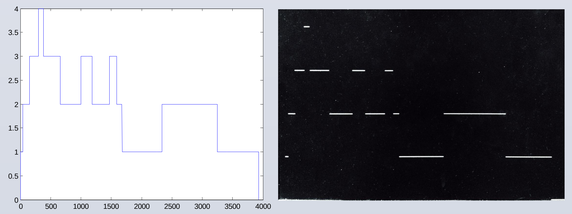


- After implementing on the FPGA, we get a result from FPGA, compared to MATLAB version with the same parameters (P_Infecting = P_Recovering = 0.001, Individual Number = 100, Average Connection = 10, Initial Infected Individuals = 2) as below



- However, even the figures seems to be similar, we can't assert that the implementation on FPGA is correct except that we can get exactly same results from MATLAB and FPGA
- For verification, there are three steps:
- First, with the same RNG, FPGA and MATLAB should give out exactly same results, which proves the Verilog code authentic
- Second, test the RNG that we used, especially compare it with the MATLAB rand() function which we assume is good enough
- Last, repeat the simulation on FPGA, with the same seeds for RNG, the result should not change

- To examine the FPGA version, we build a same RNG on MATLAB and give them the same seeds as FPGA. Also, setting a smaller group of individual makes it easier to identify them
- The exactly same results below validate the FPGA implementation



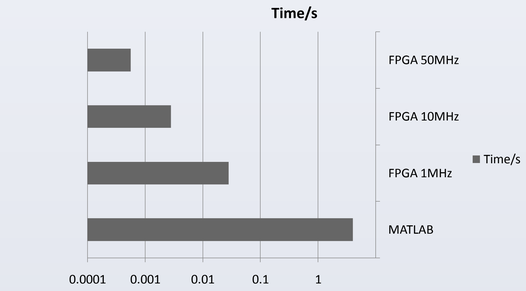
- To test the RNG, two methods are introduced: Chi-Square Test and Serial-Correlation Test
- Compared to rand() function in MATLAB, the RNG in this project not only pass the tests more, but shows more better results

	Chi-Square			S-C lag = 1			S-C lag = 5		
	pass	fail	good	pass	fail	good	pass	fail	good
RNG	46	4	30	46	4	27	49	1	27
Rand()	44	6	20	41	9	25	45	5	23

- At last, we add the infected numbers of first 2000 steps together and display it on board, the number stays the same during repeats

EVALUATION

- The most important factors of the implementation is speed and area. The major purpose to transplant the simulation from software to hardware is to accelerate it and how many individuals could be implemented is determined by area.
- For MATLAB simulation, with the parameters mentioned above, the average computing time is 4s with 7000 time steps.
- For FPGA simulation, with the same parameters and 50MHz clock, the computing time is 0.56ms
- As we can see, FPGA version shows a much better performance even with a relatively slow clock frequency



- On Altera DE2-115 board with 115000 logic elements, the maximum individual number that can be implemented is about 120.
- In general, the hardware solution transfers the complexity in time to area, which may provide a good way to simulate a large network in real time.

CONCLUSION

- We designed a hardware solution to accelerate the simulation for SIR model and implemented it on Altera DE2-115 board
- MATLAB simulation was introduced to validate the mathematical algorithm for the simulation
- Verilog code was automatically generated from a MATLAB program which made it easier to change the parameters
- We used a 64-bit XOR feedback shift register as random number generator in FPGA and prove it to be good enough for simulation
- Verilog code was verified by comparing to the MATLAB version with the same RNG
- The final implementation was stable during repeats
- The hardware solution gave a 5000 times better speed with 120 individuals than software

REFERENCES

[1]A. Hoogland, J. Spaa, B. Selman and A. Compagner, A special-purpose processor for the Monte Carlo simulation of ising spin systems, Journal of Computational Physics, Volume 51, Issue 2, August 1983, Pages 250-260

[2]http://www.cse.wustl.edu/~jain/cse567-08/ftp/k_27trg.pdf

[3]http://en.wikipedia.org/wiki/Compartmental_models_in_epidemiology