VLSI Implementation of DNN using Integral Stochastic Computing
Project Report (Final)

Reference Paper: https://arxiv.org/pdf/1509.08972.pdf

Title: VLSI Implementation of Deep Neural Network Using Integral Stochastic Computing
Authors: Arash Ardakani, Student Member, IEEE, François Leduc-Primeau, Naoya Onizawa, Member, IEEE, Takahiro Hanyu, Senior Member, IEEE and Warren J. Gross, Senior Member, IEEE

Team Members :

1) Krutika Bapat
2) Divya Krishnani
3) Kushashwa Ravi Shrimali
4) Aakash Kumar Shende
5) Balram Rathore
6) Tameshwar kumar Gaur
7) Dinesh Kumar Singh
8) Bhanudev Som
9) Preeti
10) Bhupendra Kumar
11) Saurabh Kumar Singh

(Underlined: ECE Students, Non-underlined: CSE Students)

Objectives realizations:

1) We need to create stochastic bit streams - similar to random variables. E[X] = x, like for ⅛, it will be 01000000 [probability of 1 to occur is ⅛]

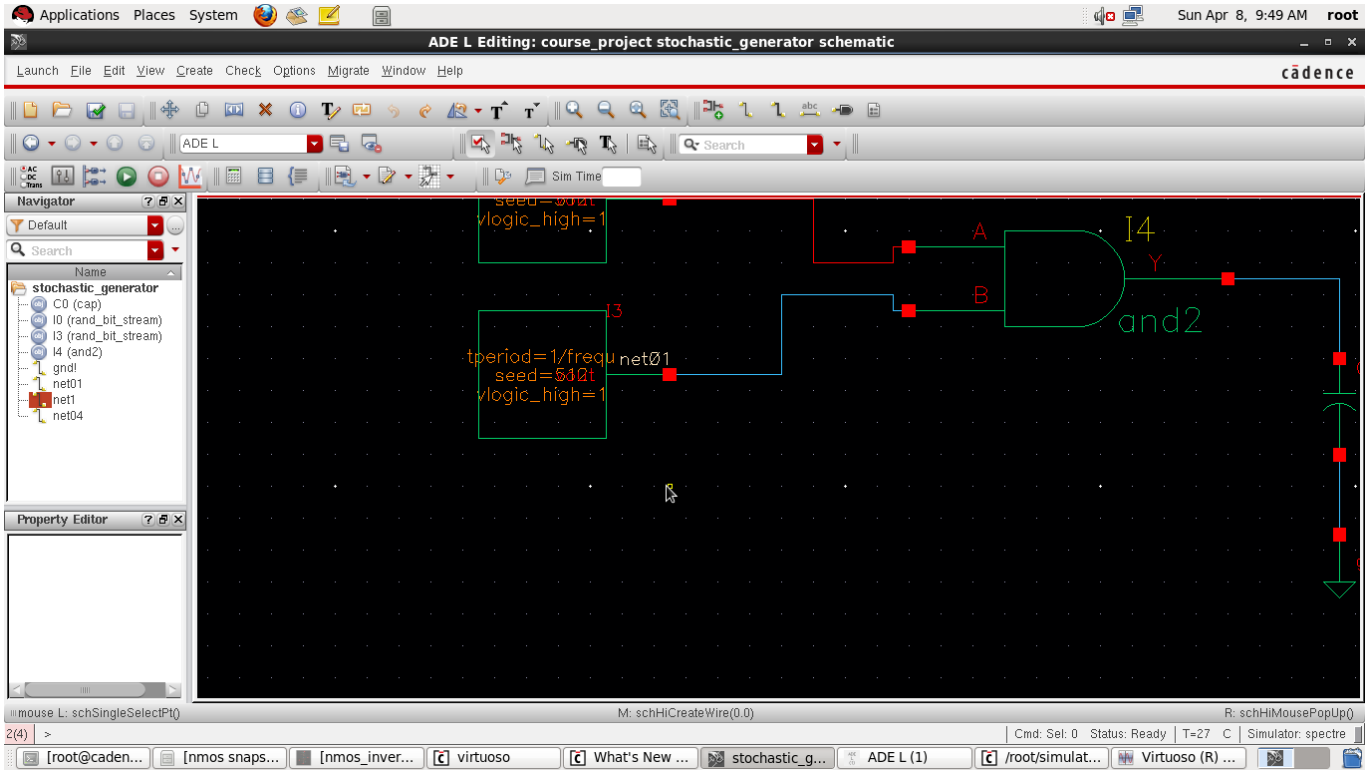
2) On creating stochastic bit stream, one way we thought to experiment using rand_bit_stream in ahdlib in Cadence, shown in this report. The two streams have been bit-wise ANDed and the output has been verified.

3) Stochastic Bit Streams Reference Paper Citation [24] in our reference paper, shows use of *LFSR and comparator* to generate stochastic bit streams.

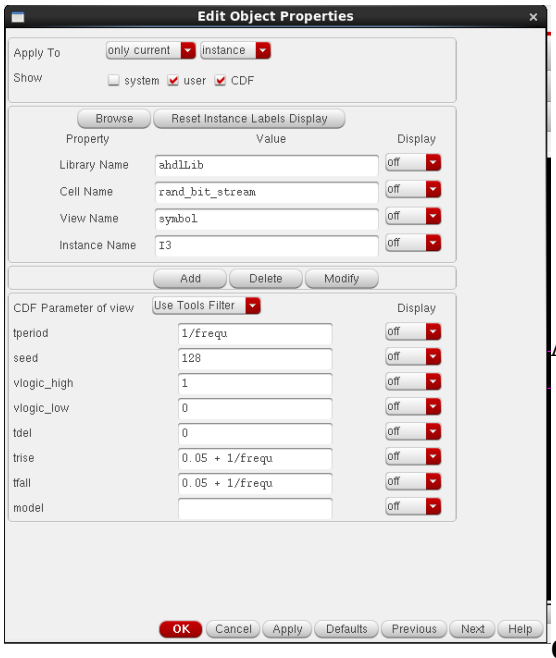
4) Target is to **create a neuron having the same inputs but operations for stochastic bit streams, and instead of binary inputs - we have stochastic streams using B2S and B2IS.**

5) On discussion with Japa sir, we realized the architecture of Neuron was not at all difficult except the basis being LFSR, Comparator, and inputs. Once done, then it just consists of bit-wise AND and Tree Adder and activation function (any - here, NStanh).

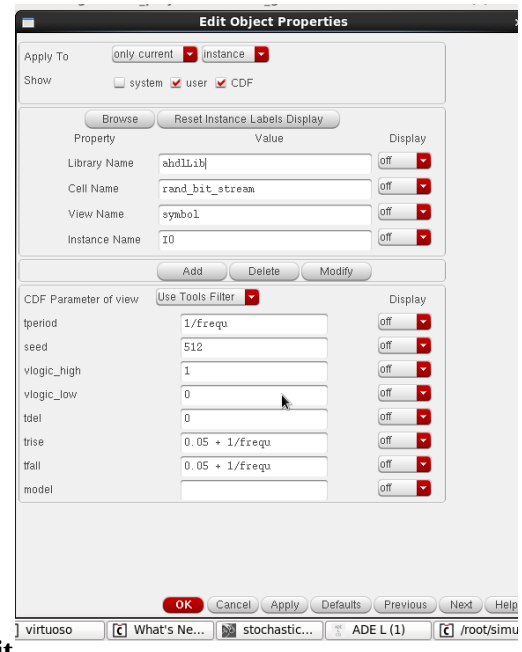
Task highlights on point 2 : rand_bit_stream --- name of the cell used from ahdlib (cadence), to generate random bits for I/P of AND gate. gpdk180 nm CMOS technology used.



rand_bit_stream block and parameters



A



B

Output Of Random Bit Stream Block

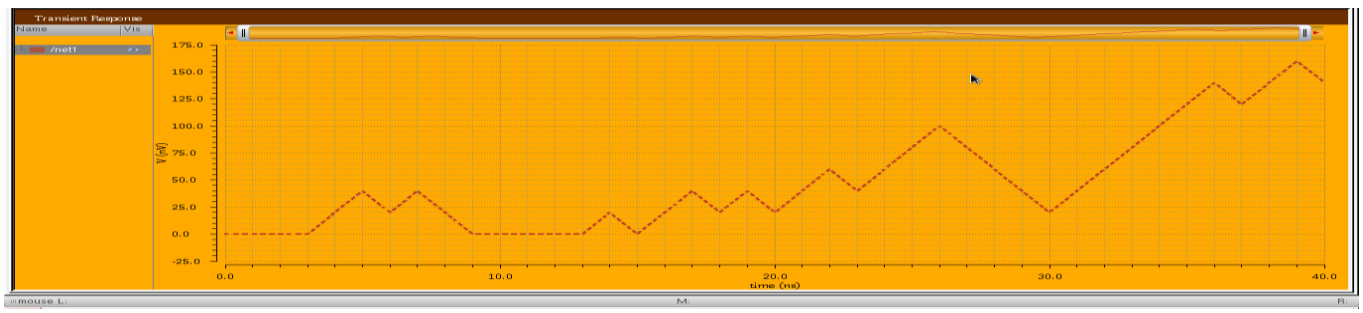


Fig. Denotes sample random bit stream generated

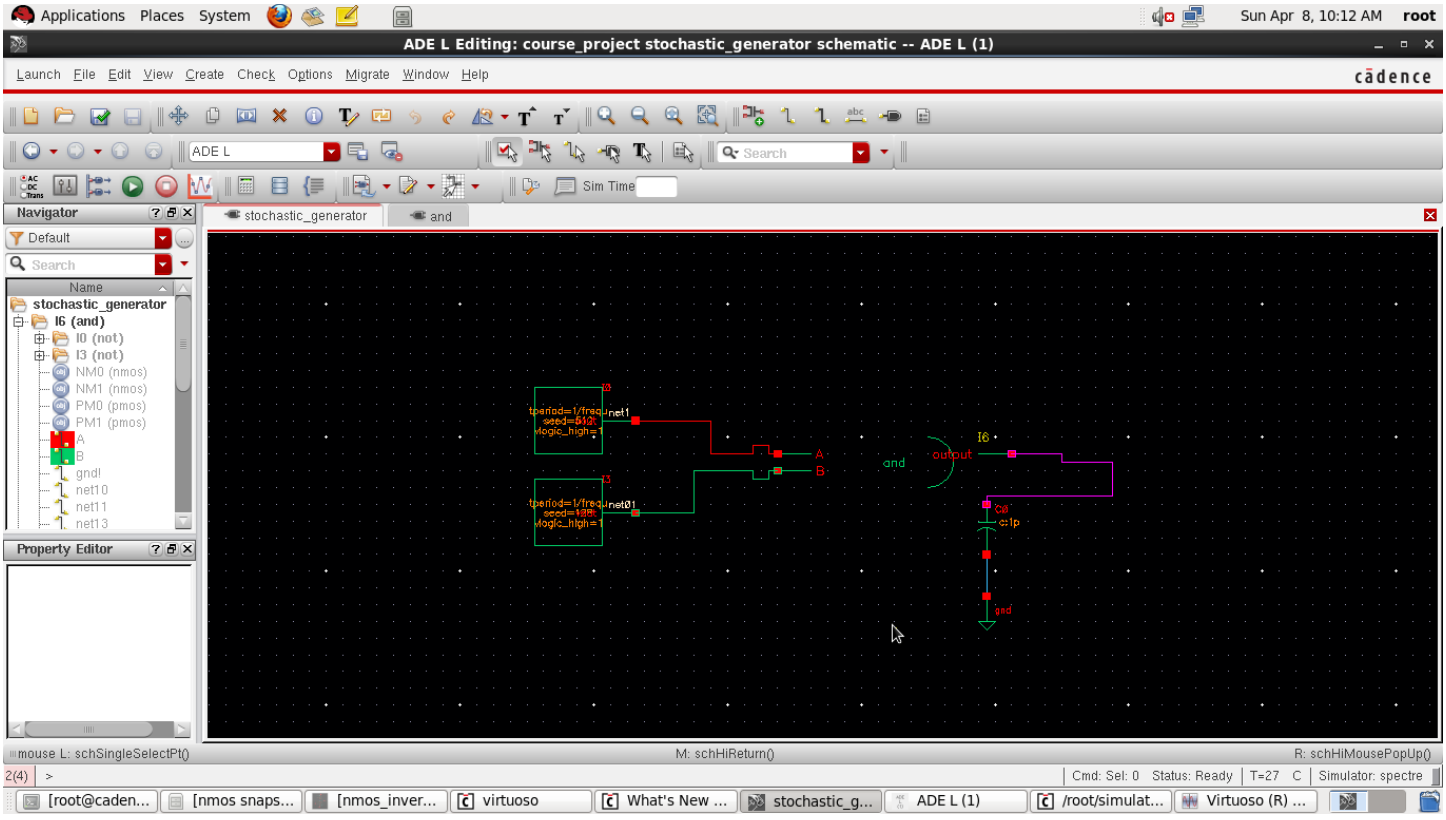


Fig. Block Diagram of Random Bit Stream generators

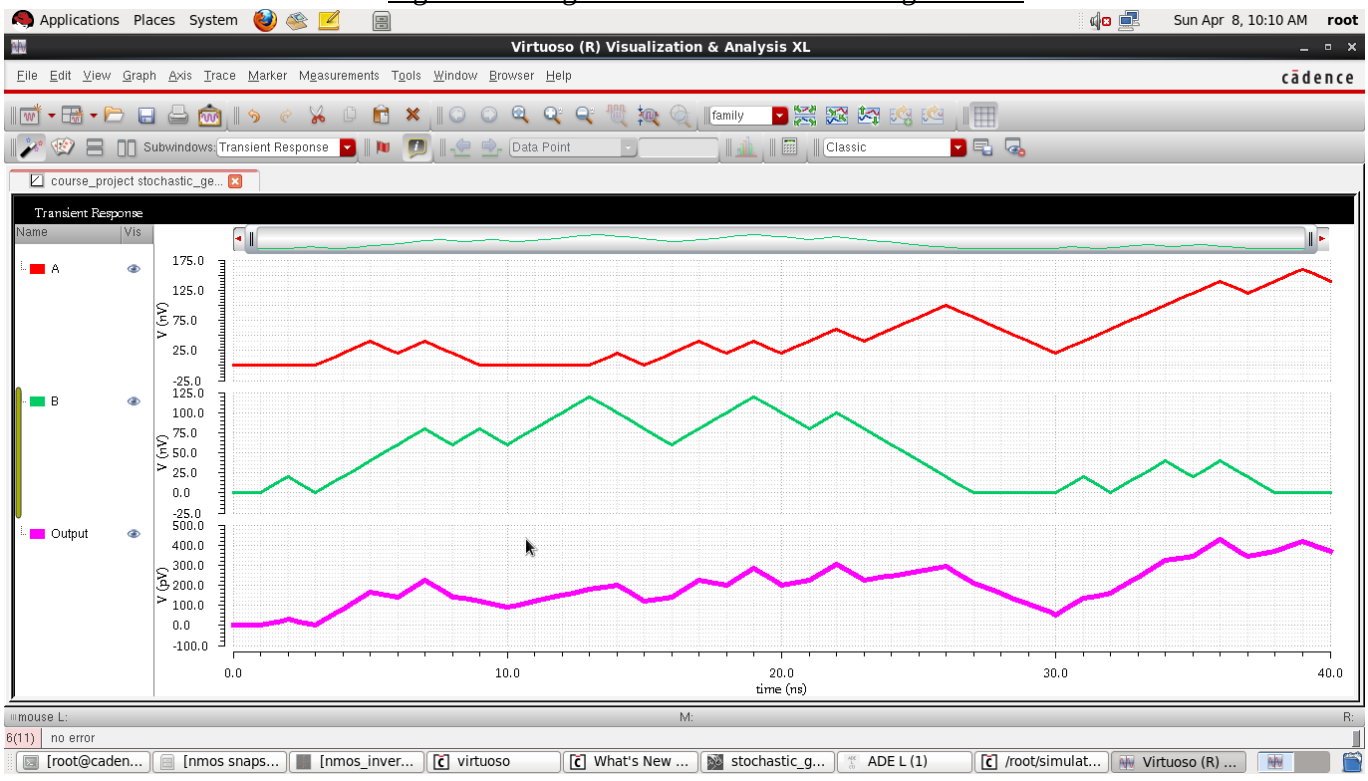


Fig. Denotes output of A AND B (A.B)

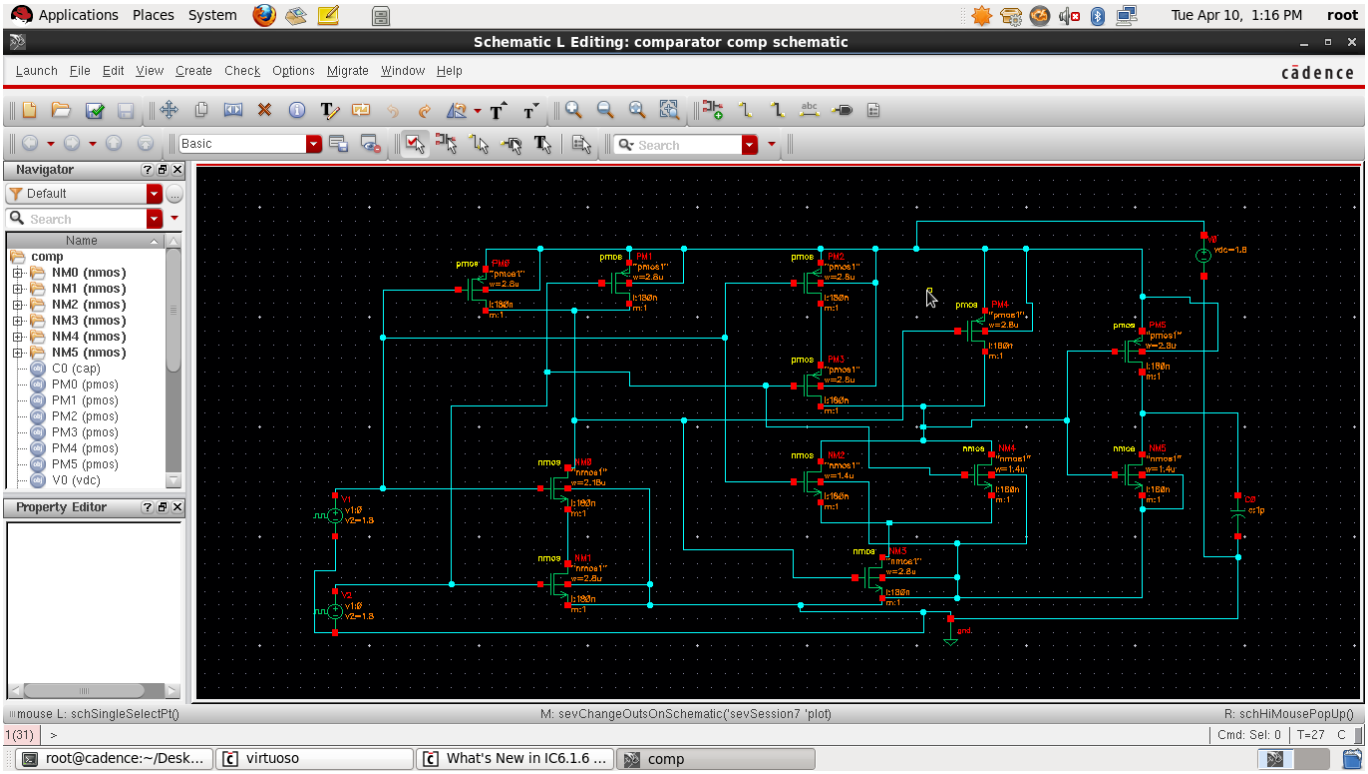
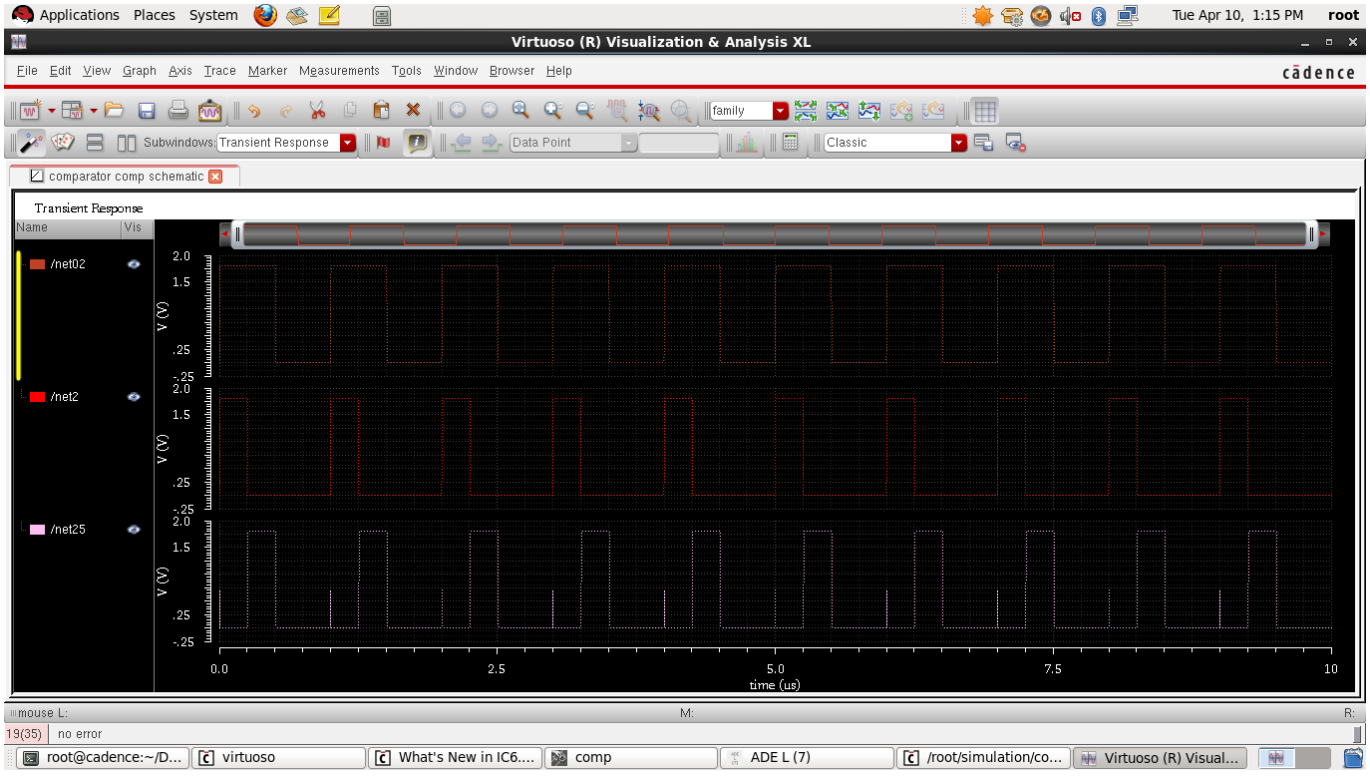
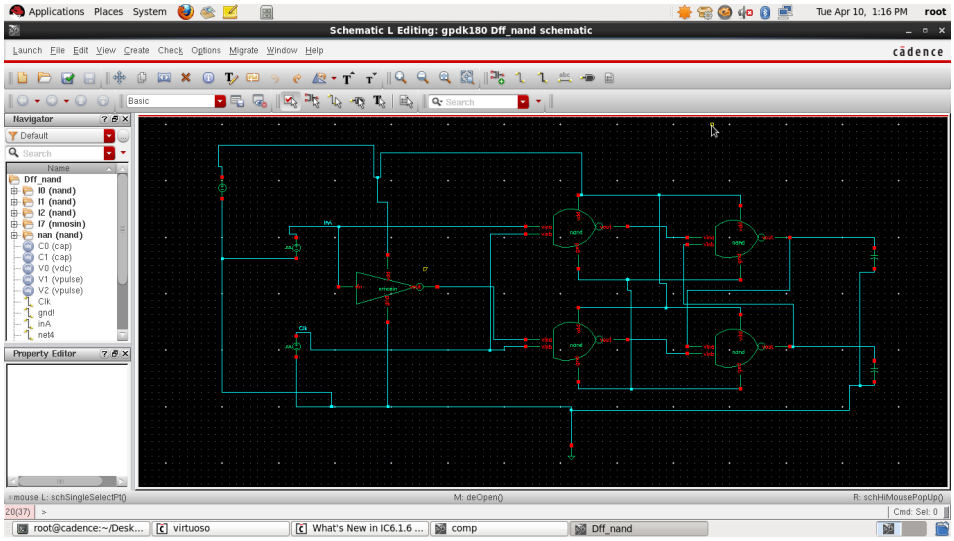


Fig. Comparator Circuit (Designed using CMOS Technology)





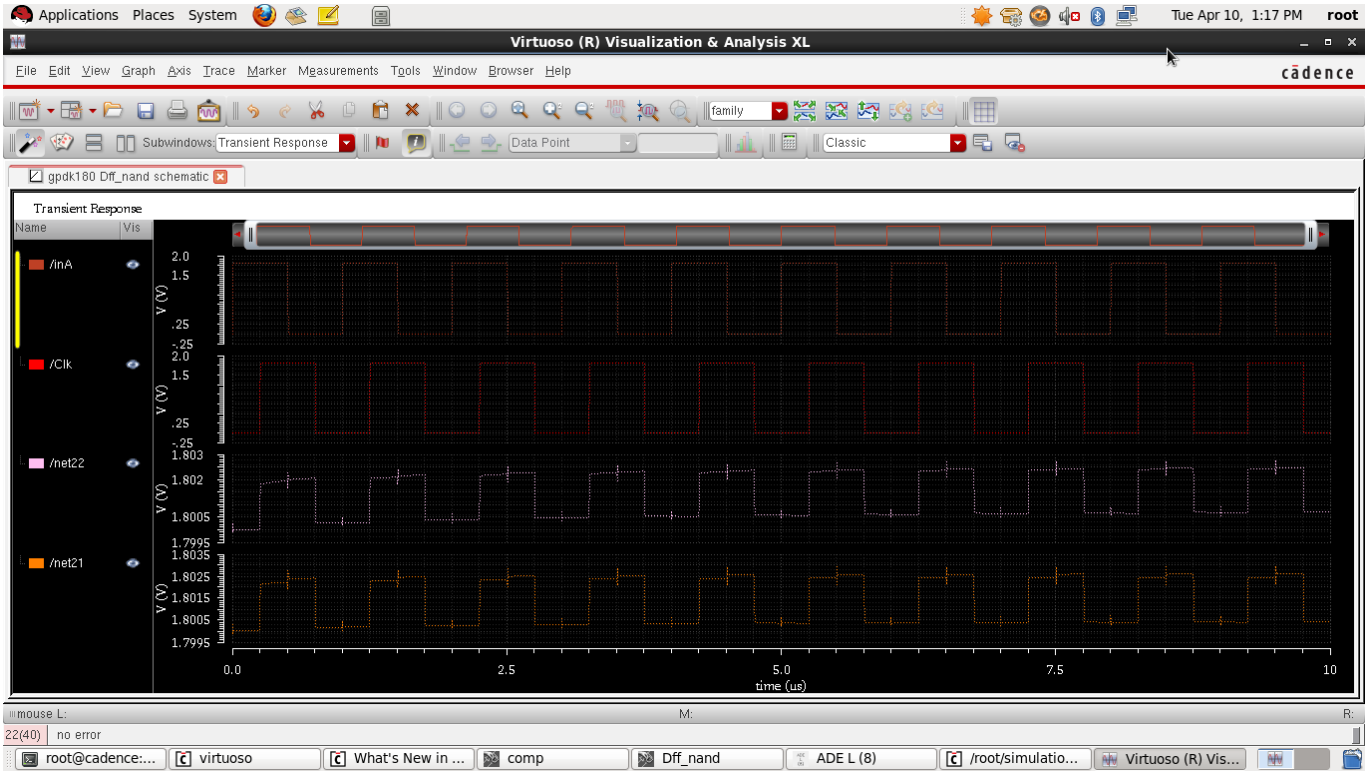


Fig. Comparator Output Graphical Image

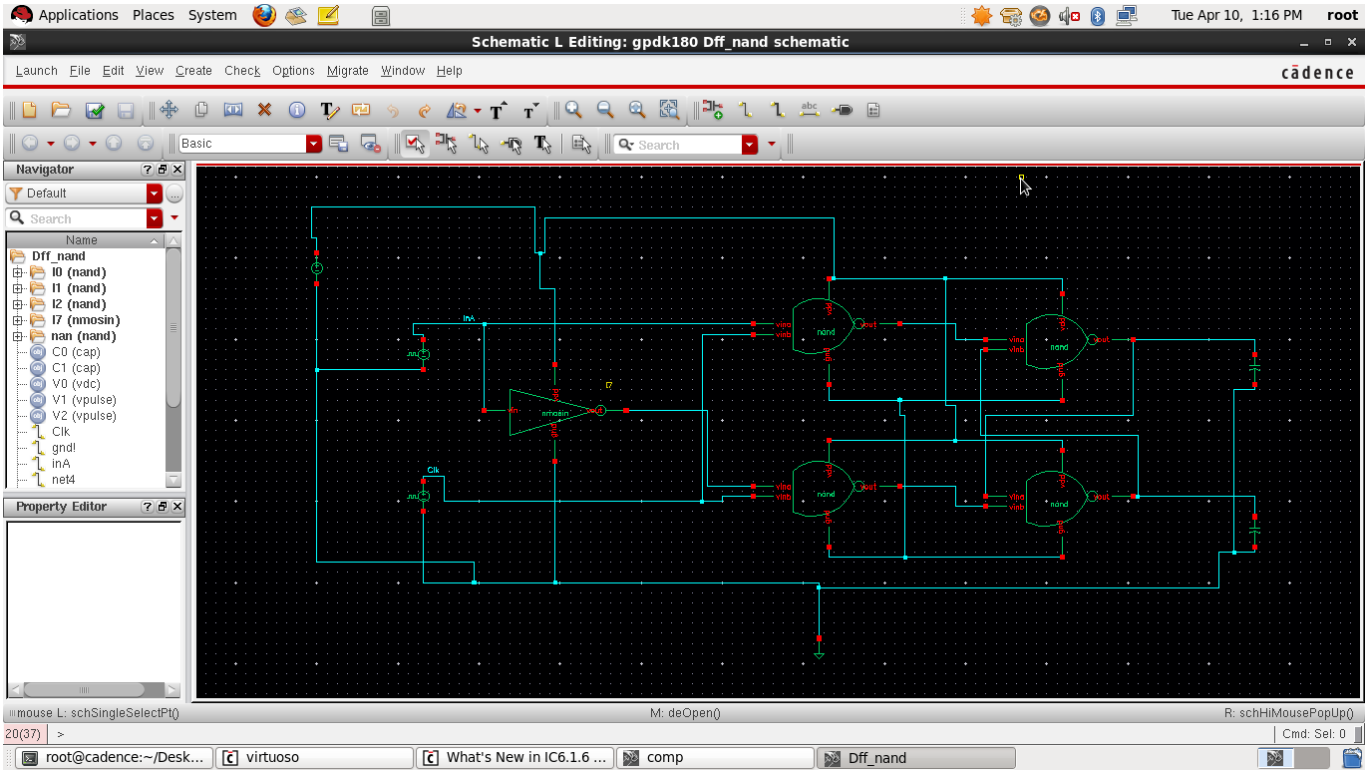


Fig. D Flip Flop Circuit using CMOS Technology (Cadence), 180nm

Note: There were some failed attempts and we have worked on correcting them.

Correct output:

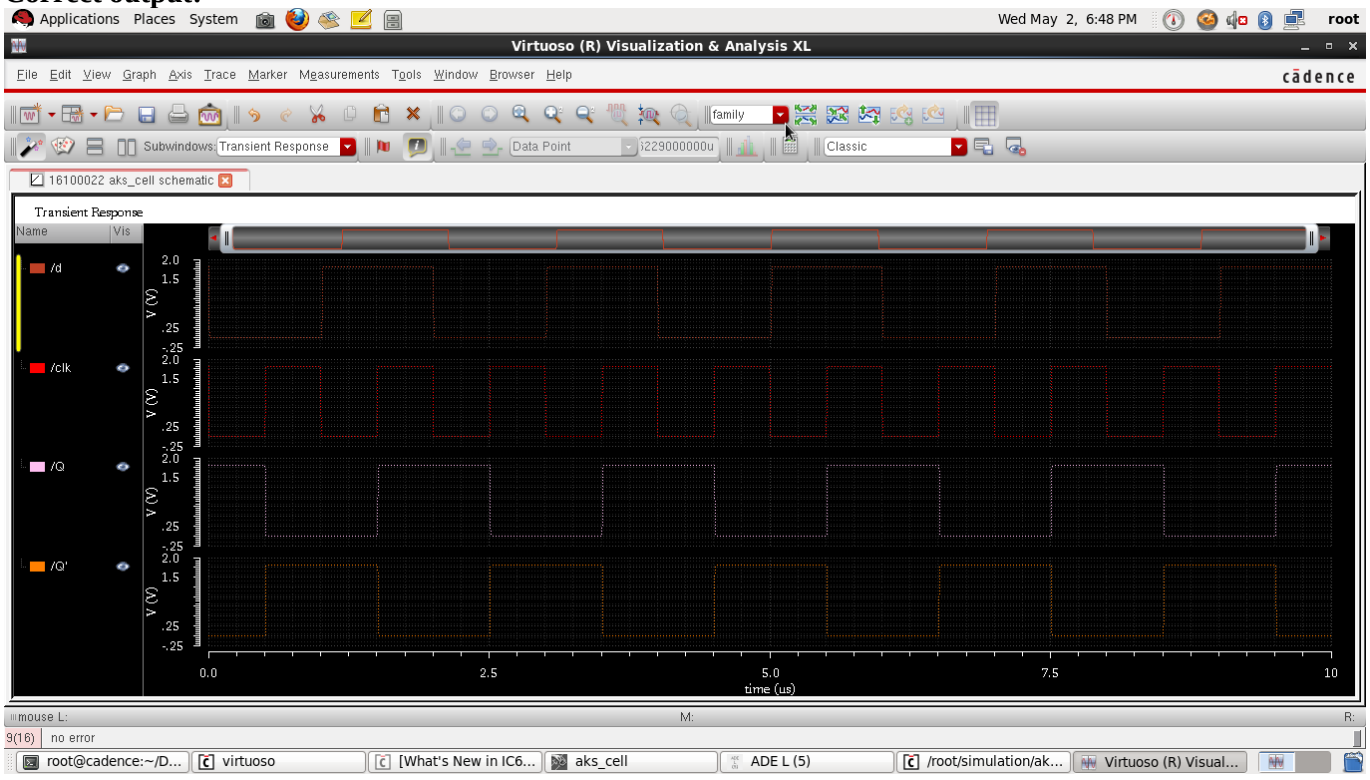


Fig. D Flip Flop Graph Output

Outputs using ASIC Design: (Verilog codes simulated)

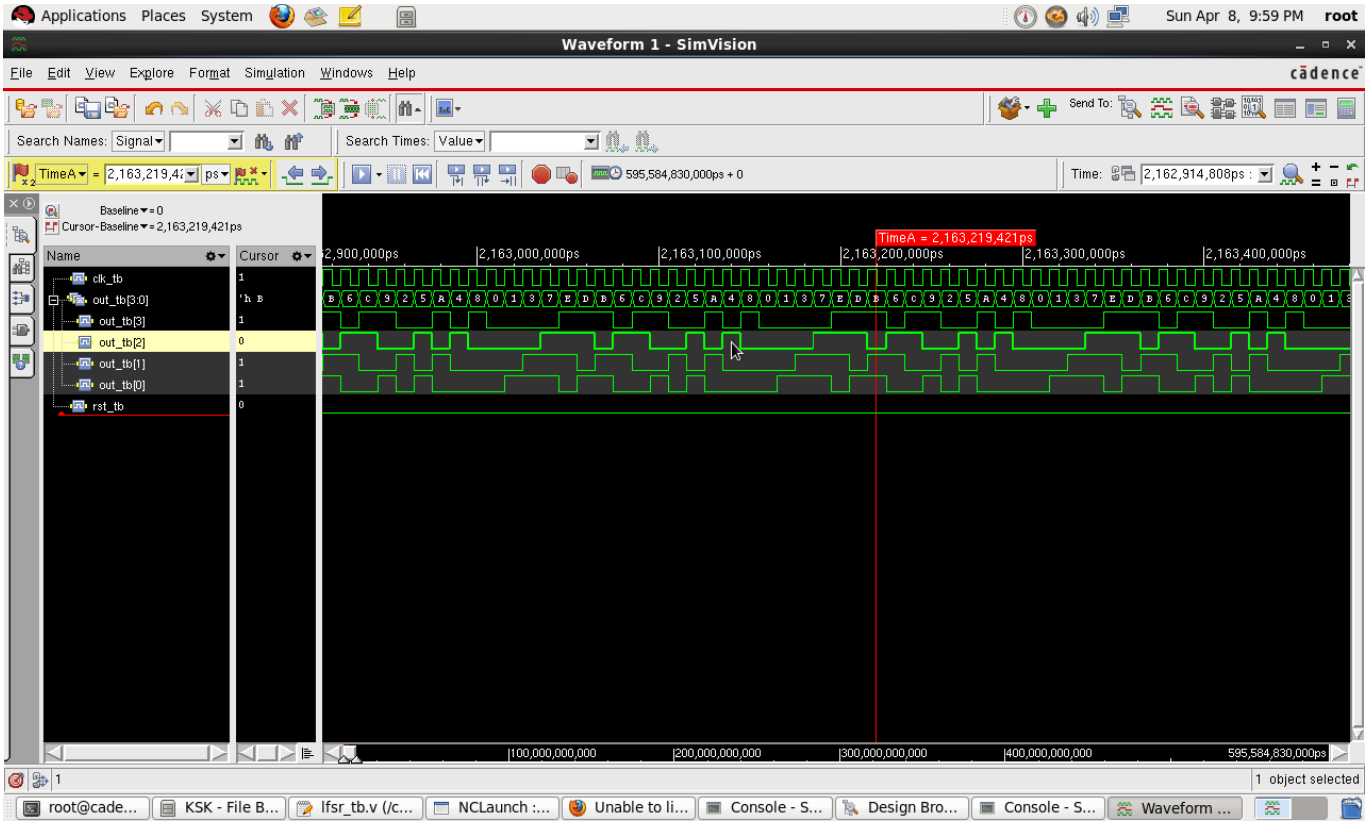


Fig. LFSR Output using ASIC Design

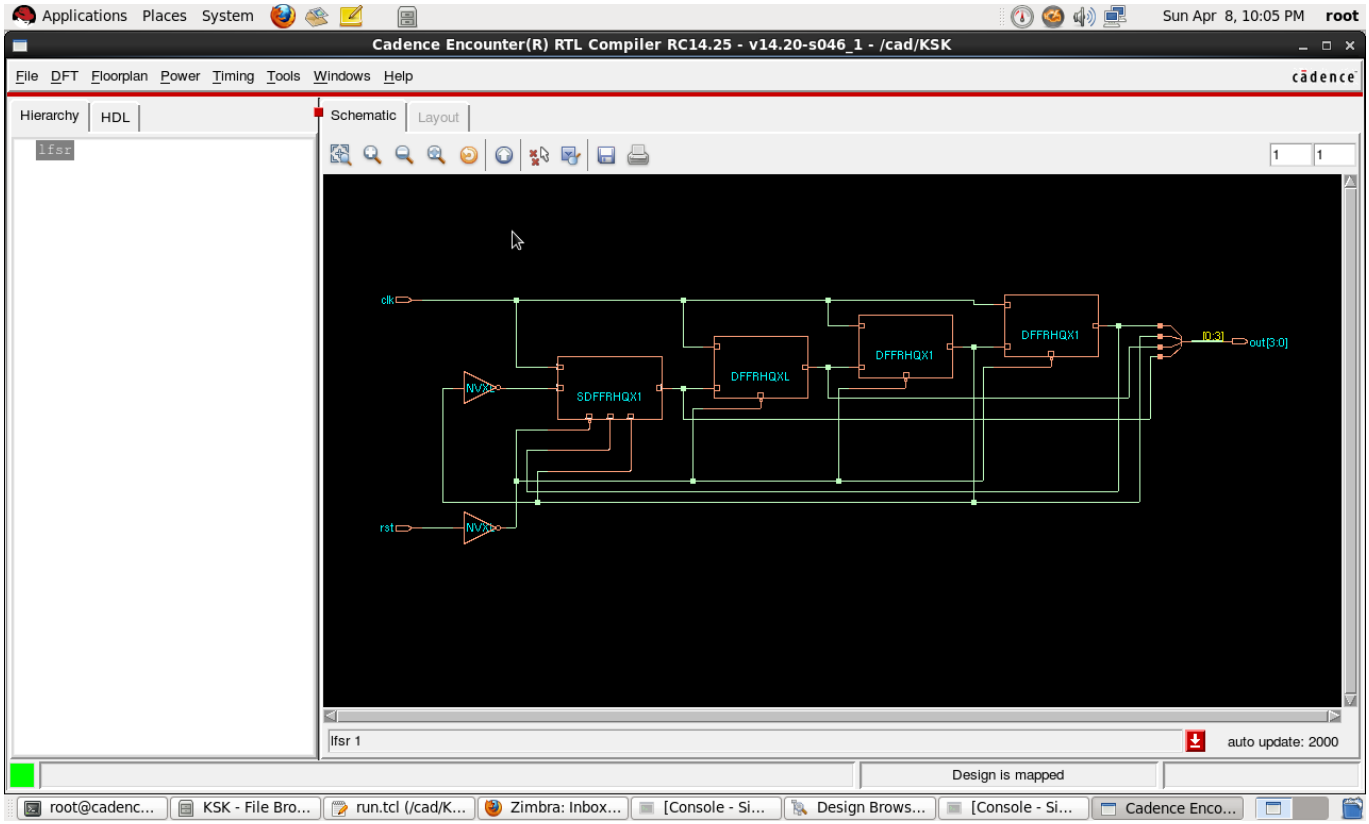


Fig. LFSR Circuit Generated using ASIC Design

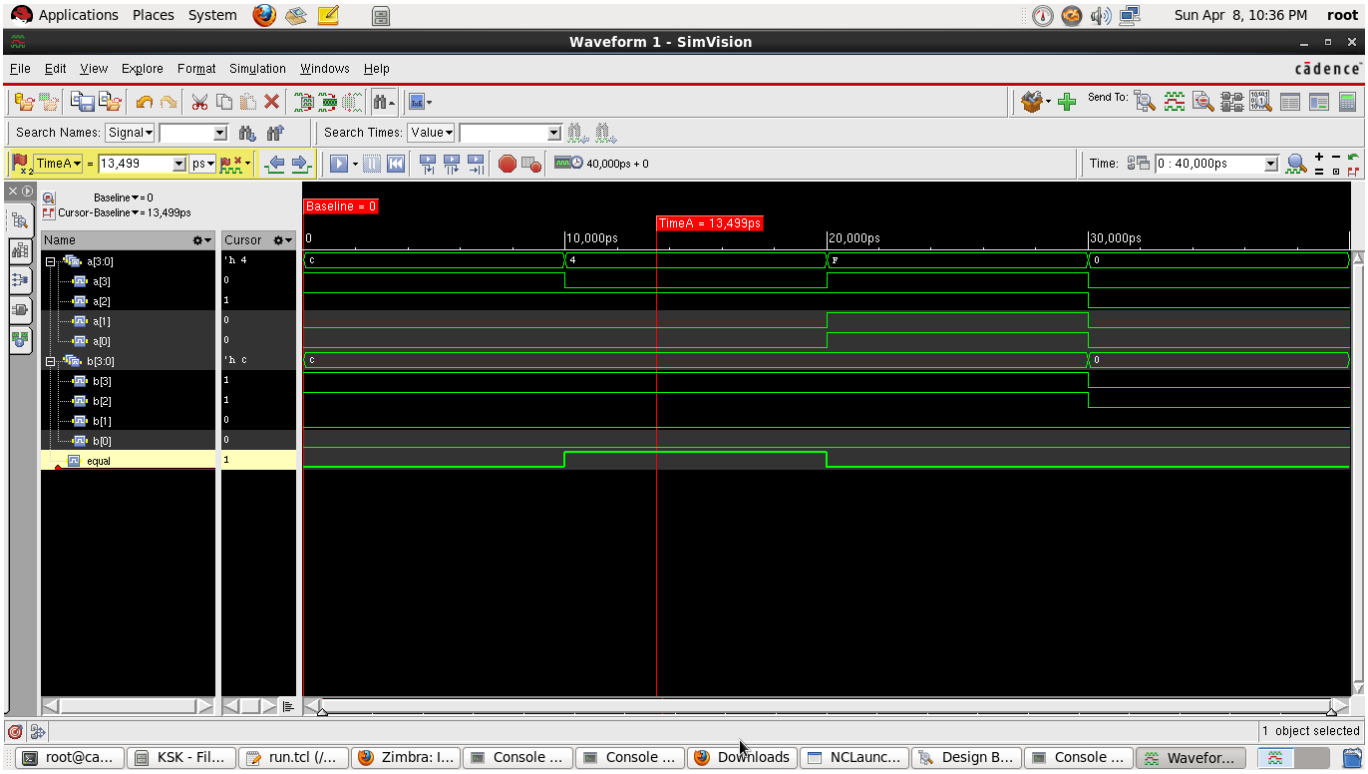


Fig. Comparator Output

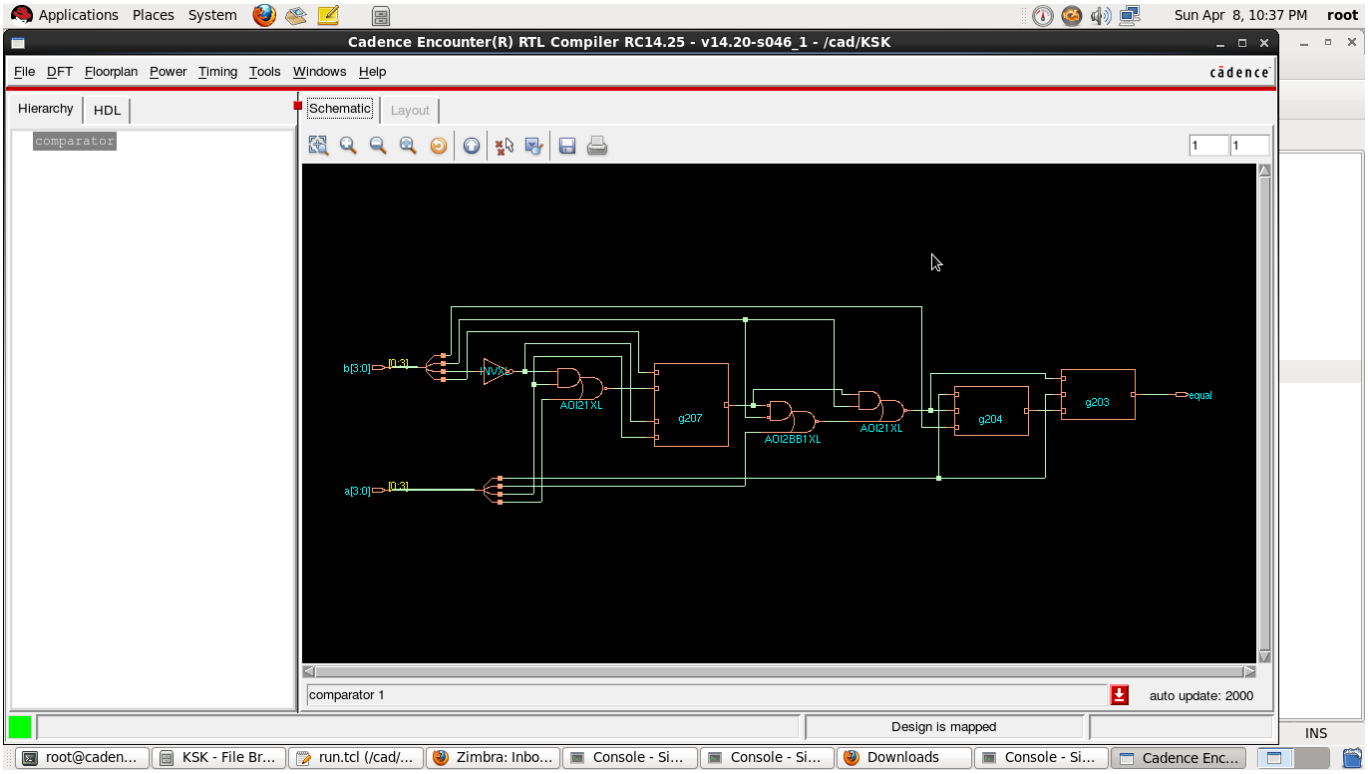
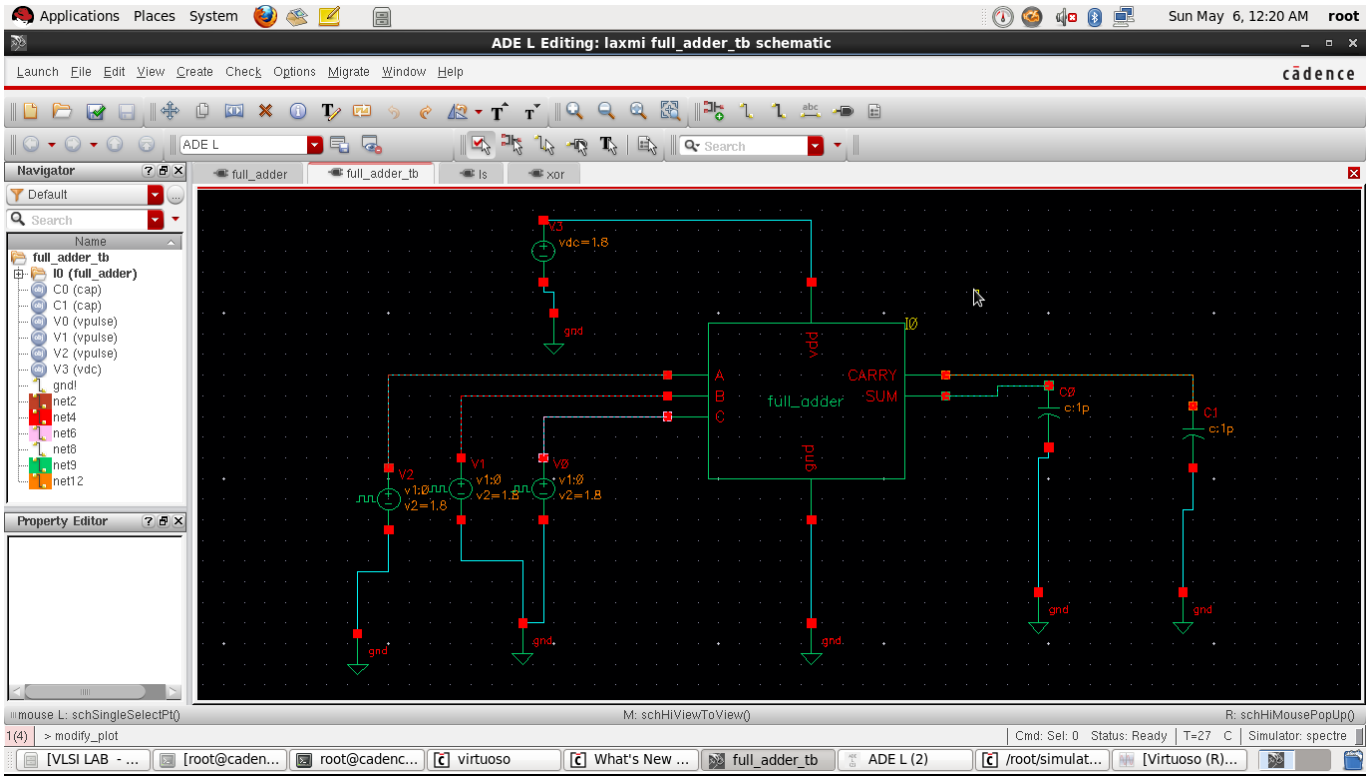
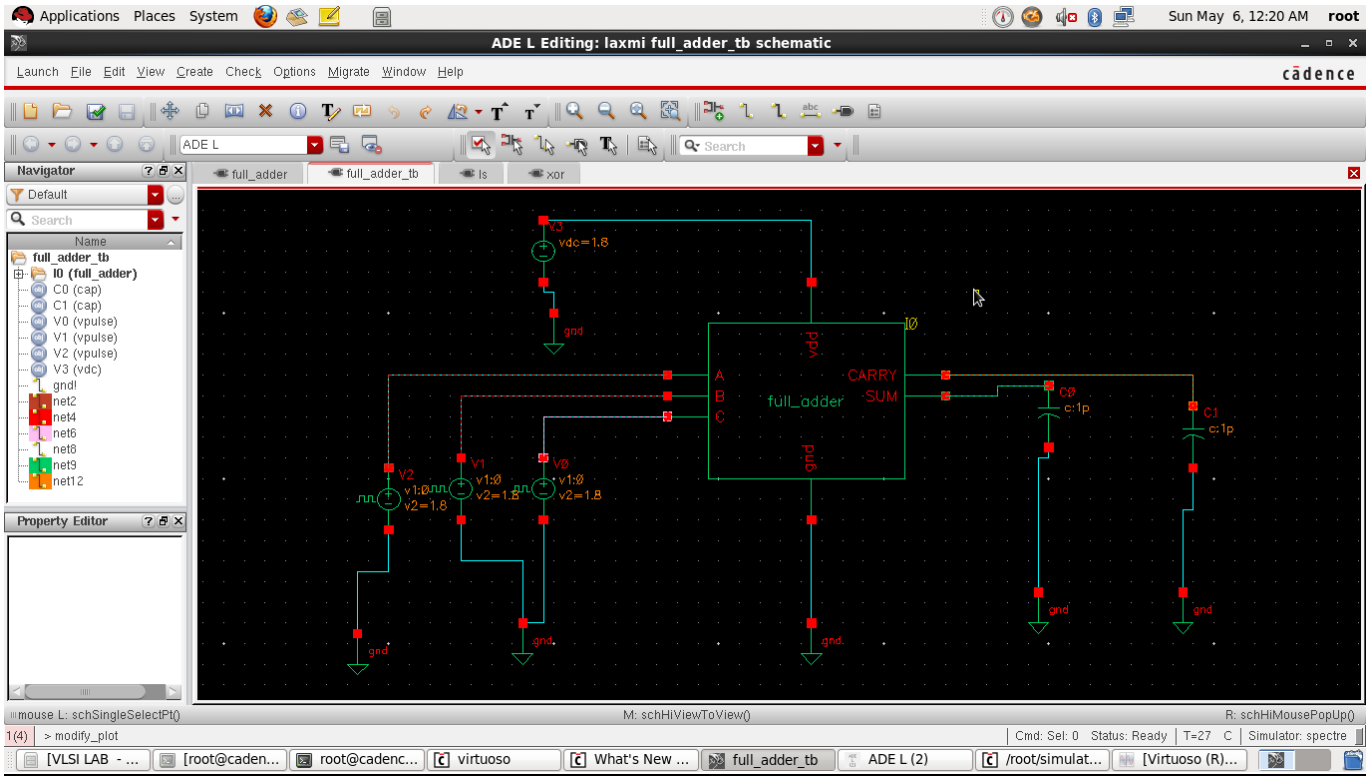


Fig. Comparator Circuit using ASIC Design

Full Adder Circuit



Full Adder Simulation Results:



Binary to Stochastic Generation

B2S (binary to stochastic) is the combinational circuit which is used for the converting binary number into the stochastic random stream. It has two-part, LFSR consist of several flip-flops and comparators which helps in generating sequence stochastic random bits.

Working of the LFSR is to generate unique bit at every output "Q" of the Flip-flop as shown in fig.2 which will be shifted at a particular interval and thus a pseudo-random no. is generated. which will be fed to the comparator according to which comparator gives either 1 or 0. If the output of the LFSR is less than the binary no. and has the same period as LFSR then the output of the comparator will be 1 else it will be 0.

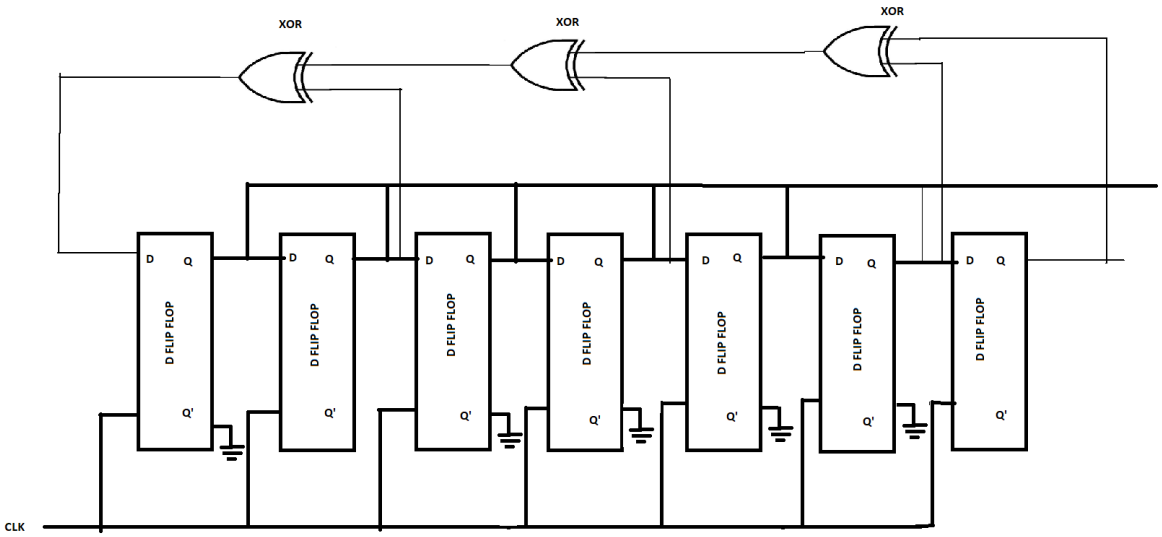


Fig1. B2S binary to stochastic converter consisting of n bit LFSR

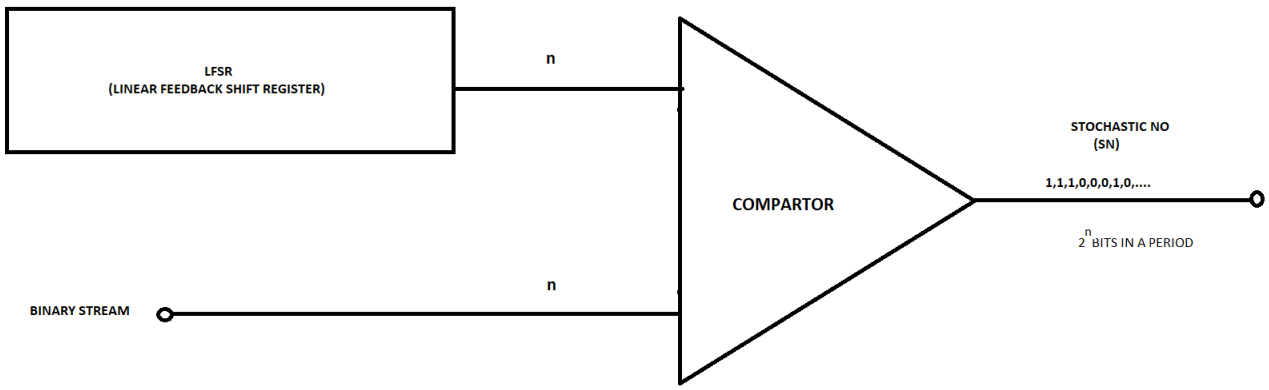


Fig. Binary to stochastic converter consisting of n-bit LFSR

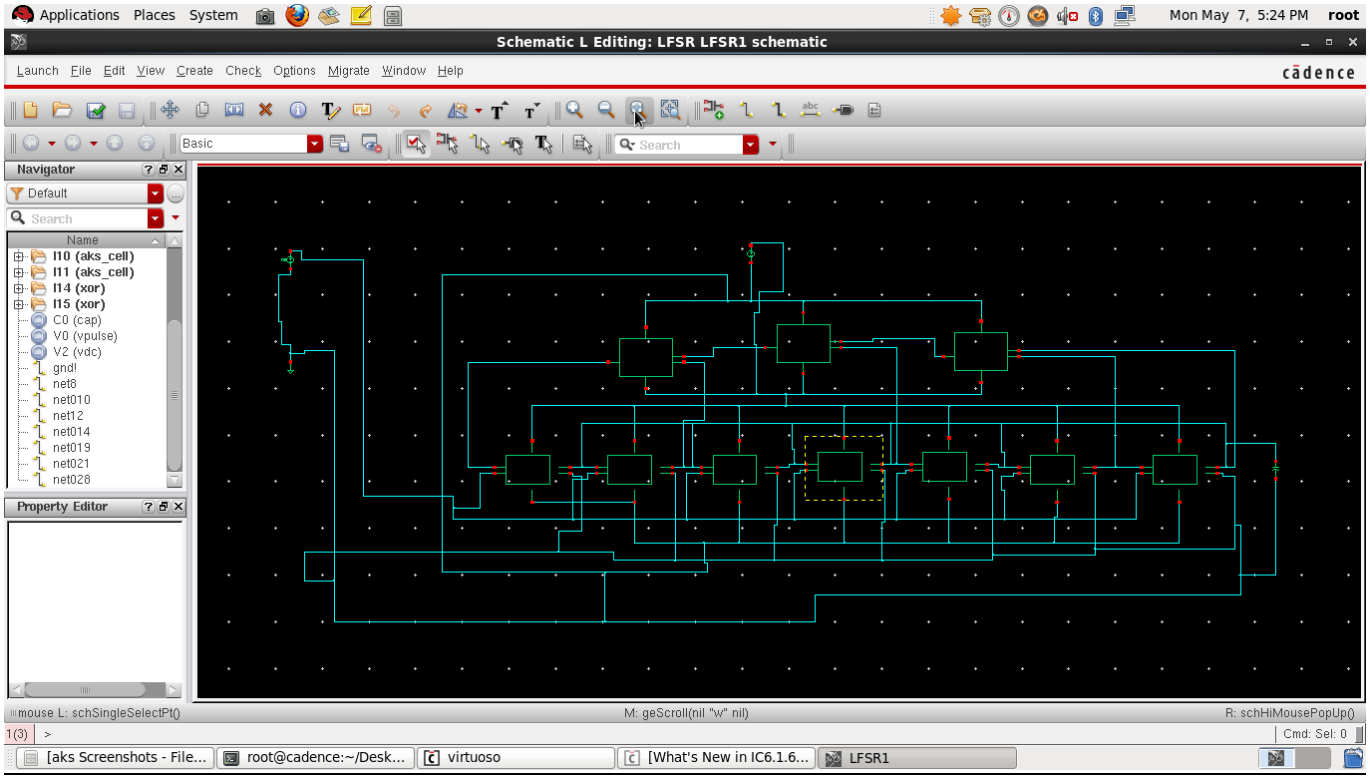
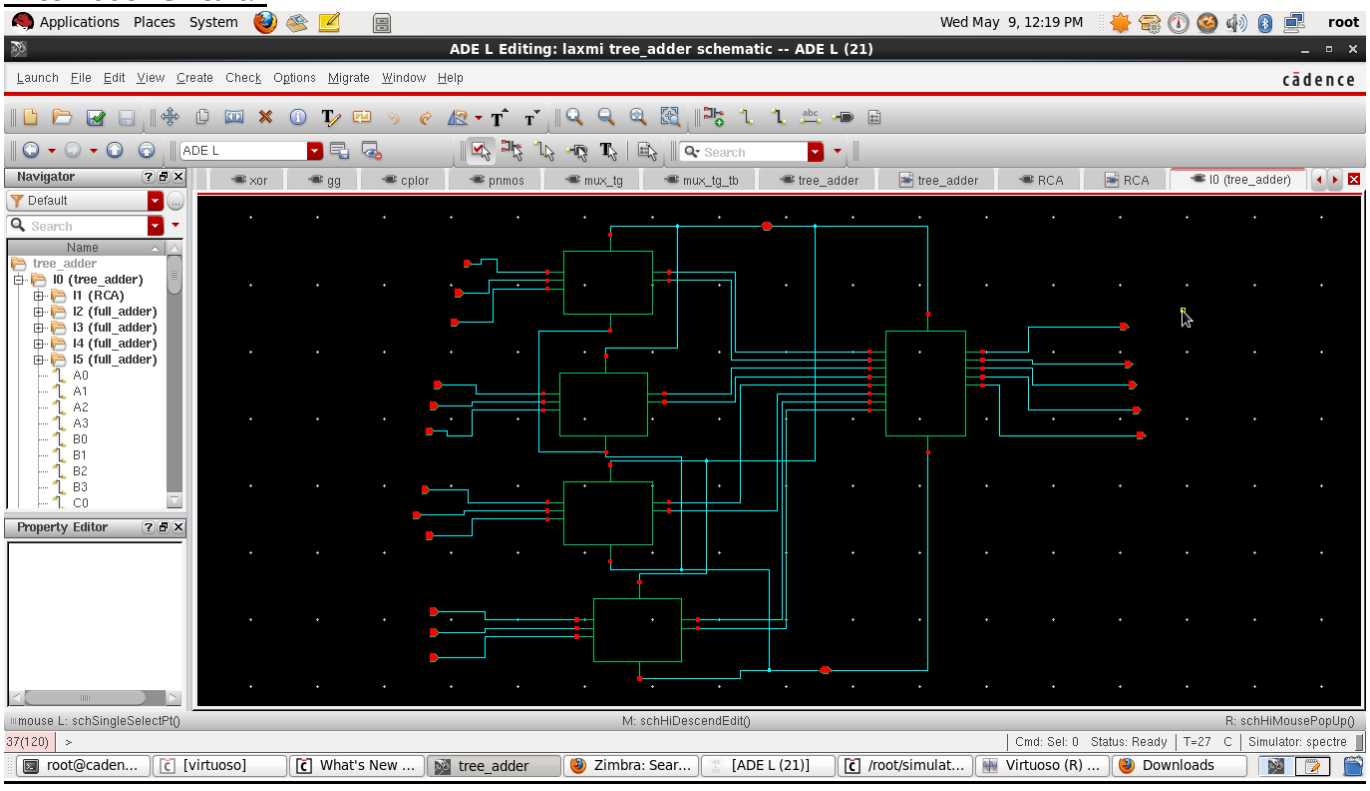
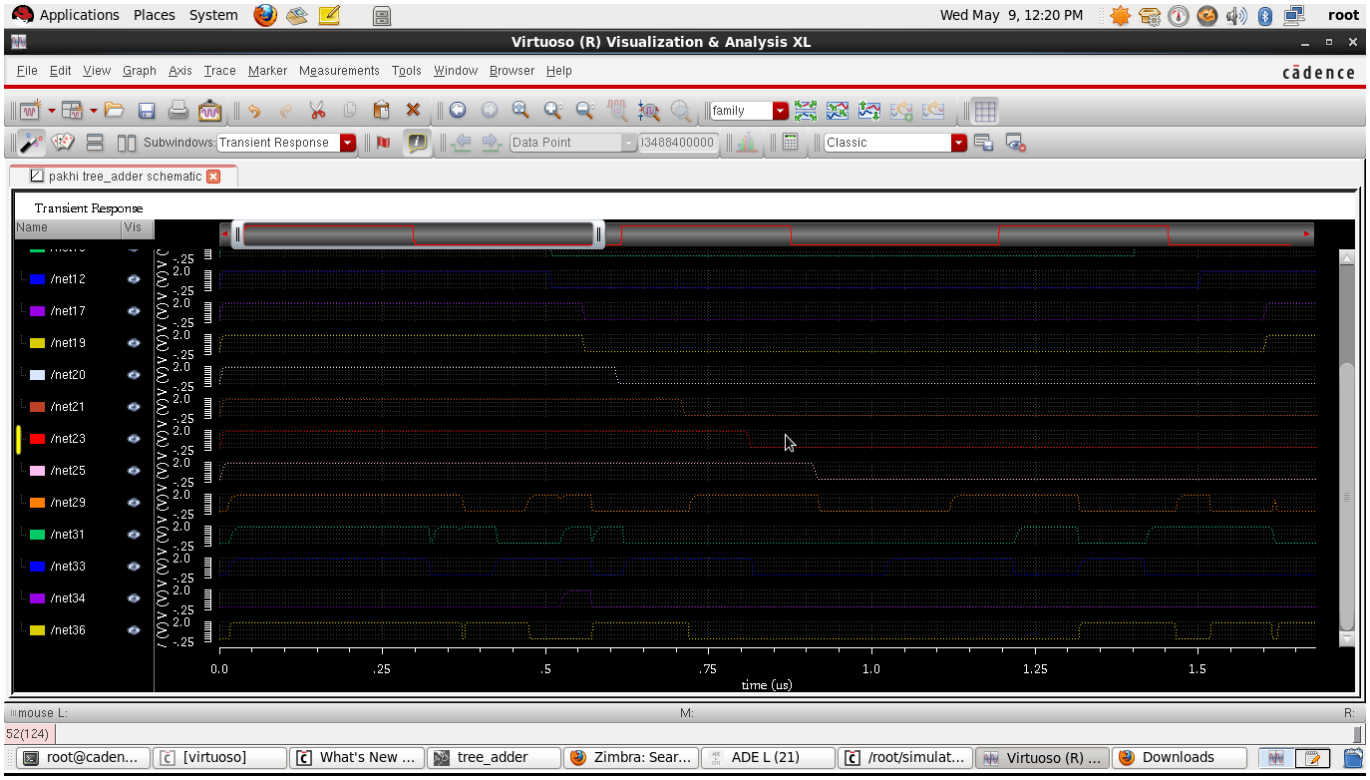


Fig. Cadence (180 nm CMOS) implementation of 7 bit LFSR

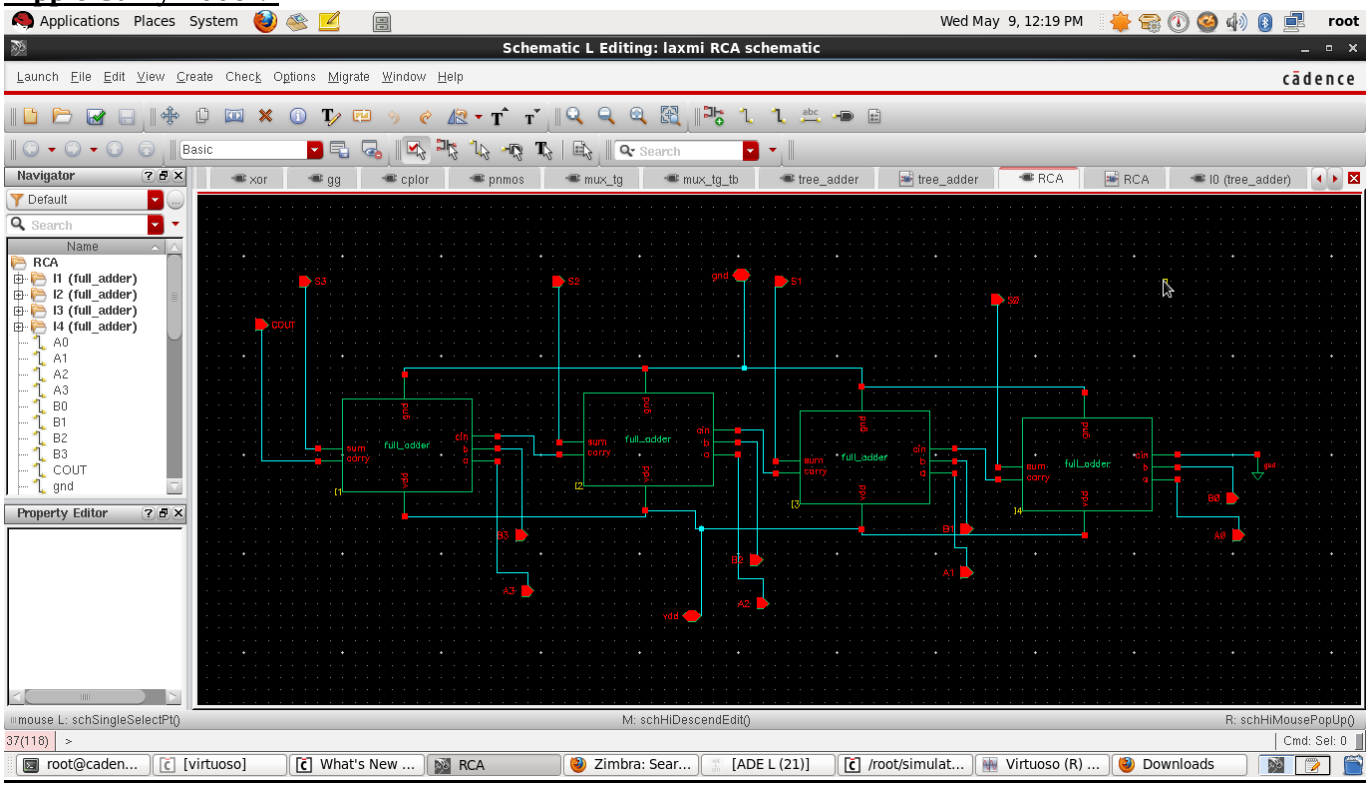
Tree Adder Circuit:-



Tree Adder Output:-



Ripple Carry Adder:-



Rough Diagram to implement NStanh Function using FSM

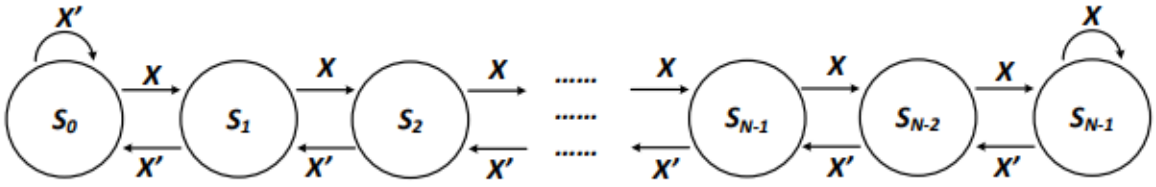


Fig. 2. A generic linear state transition diagram.

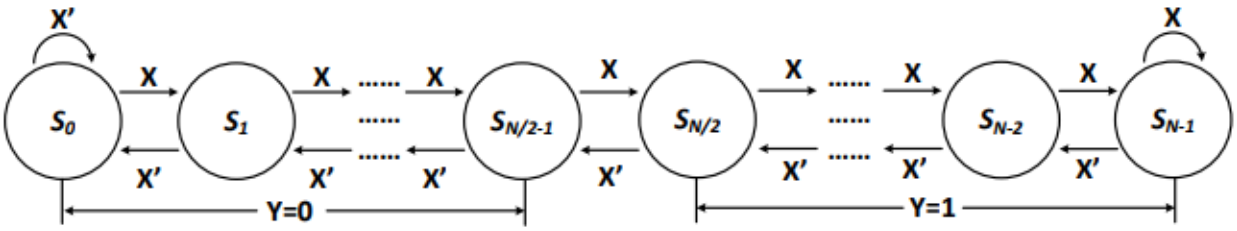


Fig. State Transition Diagram for Implementation of tanh Function in FSM

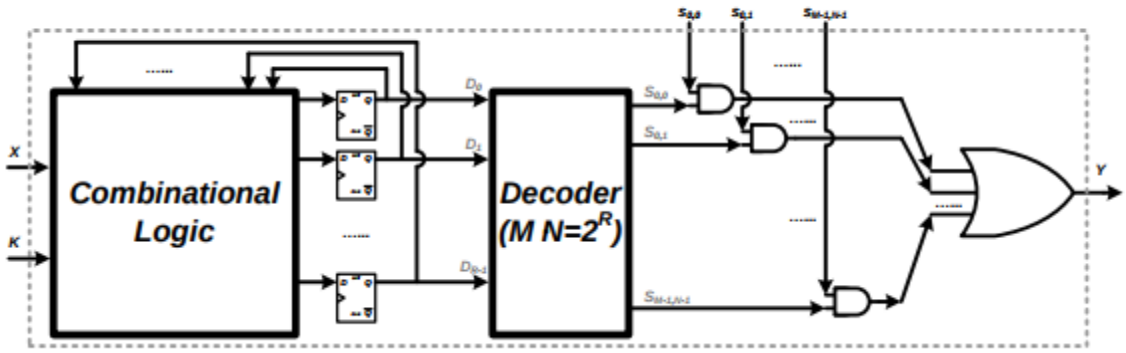


Fig. Proposed model to Implement NStanh Function for Stochastic Computation

Problems faced

1) Unable to provide desirable sequence of the binary input (*an image to be passed as i/p to neuron*)
2) Implementation of NStanh function using FSM.
3) Generation of stochastic stream using FSM Functions.
4) Implementation of State Transition Diagram for making NStanh function.

References

[1] Aidyn Zhakatayev , Kyounghoon Kim , Kiyoung Choi and Jongeun Lee "An Efficient and Accurate Stochastic Number Generator Using Even-distribution Coding". *School of Electrical and Computer Engineering, UNIST, Ulsan 44919, Korea †DMC R&D Center, Samsung Electronics, Seoul 06765, Korea ‡Dept. of Electrical and Computer Engineering, Seoul National University, Seoul 08826, Korea*

Hardware-Driven Nonlinear Activation for Stochastic Computing Based Deep Convolutional Neural Networks, Ji Li, Zihao Yuan, Zhe Li, Caiwen Ding, Ao Ren, Qinru Qiu, Jeffrey Draper, Yanzhi Wang

[3]P. Li, W. Qian, and D. Lilja, “A stochastic reconfigurable architecture for fault-tolerant computation with sequential logic,” inIEEE 30th International Conference on Computer Design (ICCD)Sept 2012, pp.303–308