Spintronics Logic in Cache (SLIC) Based Adders

A Thesis
SUBMITTED TO THE FACULTY OF
UNIVERSITY OF MINNESOTA
BY

Shamayal Raza

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE
IN
ELECTRICAL ENGINEERING

David J. Lilja, Adviser

December, 2014

Acknowledgements

Research as we all know is a long and tedious process and it requires efforts from more than an individual to take it to its appropriate end. The case here is no different where I got all the help that I needed from a wonderful group of people that I worked with.

First and foremost, I want to thank my adviser Prof. David J. Lilja for giving me the opportunity to work under him and for his continuous support and motivation throughout the thesis. I also want to thank him for the suggestions that he gave me whenever I was in doubt or when I got stuck at some problem.

I want to thank our meeting group which includes Bill Tuohy, Nishant Borse, Pushkar Nandkar and Cong Ma for their support and suggestions during our weekly meetings.

I also want to thank my friends, Vaibhav Sharma and Vijay Panda for listening to me talk about the project and for having discussions on the same. I came up with solutions to some of the problems during these discussions.

Last but not the least; I want to thank the graduate school for providing the resources that made this project possible.

Abstract

MOSFET is the technology on which the electronics industry is based on for the last few decades. But as the device size keeps getting smaller, some difficulties arises that result in degradation of the device performance which can no longer be dealt with. So, we need to have a look at the novel alternatives that can not only maintain the current standards of the industry but also have the potential to take it forward.

An emerging technology called Spintronics which uses spin of an electron as well as the charge as opposed to MOSFET which only uses the charge, opens up avenues that were not possible with the CMOS technology.

Specifically, Spintronics Logic in Cache (SLIC) is discussed in this thesis where in addition to being serving as a proper cache; it can also serve as a full adder (Ripple Carry Adder) that can perform a 4 bit operation on the bits stored in the cache. We also designed a Carry Look-ahead Adder and compared its performance on the basis of delay, power consumption and device count to the Ripple Carry Adder. We also tried to analyze how these SLIC based adders do as compared to their CMOS counterparts. During this process, we also generalized a way to design any digital circuit using SLIC that could be designed in CMOS technology.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Gordon Moore in his 1965 publication "Cramming more components onto integrated circuits" came up with a very important observation that drove the course of semiconductor industry for decades and is still as much relevant as it was then. He observed that the number of transistors in an Integrated Chip would double approximately every 2 years. This prediction helped the industries to plan ahead and set targets for their R&D teams to achieve. This in turn, improved the performance of the circuits exponentially, enhanced the impact of digital electronics in every world market and brought in an era of miniaturization.

It was expected that this growth will continue until at least 2015 or 2020. However, ITRS in their 2010 update predicted that the growth will slow down at the end of 2013 and the transistor count, then, will double every 3 years [1].

MOSFET is the technology on which the electronics industry is based for the last few decades as it provides an excellent device for digital switching with low power consumption and high fan-out capacity. Moreover, these devices are scalable too which is one of the main reasons for amazing advances in the electronics industry. But as the device size keep getting smaller, some difficulties arise that result in degradation of the device performance. These challenges include sub-threshold leakage reduction, interconnect scaling and the minimization of the adverse effects of process variations on

the design. Dealing with these challenges is becoming more difficult with further scaling of the device feature sizes which basically means that the MOSFET based technology may not be able to provide the kind of push to the semiconductor industry in the future that it has been providing for the last several decades. So, we need to have a look at the novel alternatives that can not only maintain the current standards of the industry but also have the potential to take it forward.

In this thesis, we will focus broadly on an emerging technology called Spintronics and specifically on Spintronics Logic in Cache (SLIC) [3]. As opposed to the MOSFET based technology where the charge of an electron is responsible for all the operations, Spintronics uses spin of an electron as well as the charge which makes it a very attractive field and opens up avenues that were not possible with the CMOS technology. Magnetic Tunnel Junction (MTJ) is the basic device that makes use of Spintronics technology where, the non-volatility of spin, of the material used to make the device can be leveraged for building non-volatile memories and the charge can be used for transferring information from one device to the other, thereby, allowing us to design logic [2]. Our main focus in this thesis is to take advantage of both of these features simultaneously resulting in a logic-in-memory concept of Spintronics Logic in Cache [3]. One of the main motivations for this work is that this kind of design has the potential to provide a solution for the memory bandwidth bottleneck. Here, the same device (MTJ) is being used for storing and processing data as opposed to the conventional CMOS design of having different devices for that purpose. In addition to the MTJs, we also need switches to set/reset these MTJs and to connect them to form circuits, thereby, providing us great

control and maneuverability over the complete design. Moreover, Using the SLIC [3] approach, we first focused on the design of a 4-bit Ripple-Carry Adder (RCA) and tried to compare its performance, in terms of delay and power consumption, with conventional CMOS 4-bit RCA. In addition to this, we also designed a 4-bit Carry-Look Ahead-Adder (CLA) using the same switch-based SLIC [3] architecture and compare the pros and cons of this design over the 4-bit RCA. The important thing to note about the comparison of 4-bit RCA with the conventional CMOS design is that we have to take into account the time taken, in the case of conventional CMOS design, to fetch the data or operands from the memory which is not relevant in SLIC [3] based Adder as the operation is being performed by the memory itself. This has the potential to provide some speed advantage over the CMOS design. Moreover, having memory and logic as a single module should also provide an area advantage over separate memory and logic modules in case of CMOS but analysis in this thesis is restricted to comparing SLIC's circuit performance with the CMOS based circuits as well as other SLIC based design of the same circuit.

The methodology used in the design of this SLIC [3] based Adder is actually quite similar to the conventional CMOS adder design where we use cascading of different gates to realize the logic. Here, instead of that, we used cascades of basic SLIC [3] based NAND gate to realize every function. While designing this adder, we also realized that this methodology can be applied to design any conventional CMOS circuit. So, in addition to the adder, we tried designing a 2-bit comparator too.

So, in general, this thesis makes the following contributions:

1. It demonstrates the ability of SLIC [3] to perform complex logic operations like 4

bit addition as demonstrated by the designs of Ripple Carry and Carry Look-ahead adders while acting as a memory.

2. It also, as a result, demonstrates that this new approach to the design can in fact be used to realize any CMOS circuit.
3. It gives a comparison of the SLIC based adders against the conventional CMOS adders based on the parameters of delay, power consumption and device count.

The thesis is organized as follows: **Chapter 1** introduces the motivation and the goal of the thesis. **Chapter 2** gives the background where some related work and specifically the Phd work of Shruti Patil from which this thesis is inspired has been talked about and it basically includes the description of MTJ devices and their working in switch based SLIC settings. **Chapter 3** discusses the method that we used to come up with the switch-based SLIC [3] circuit design in general and specifically for circuits of 4-bit Ripple Carry Adder (RCA), 4-bit Carry Look Ahead Adder (CLA) and 2-bit comparator. **Chapter 4** explains the experimental setup and how HSPICE is used to get the results along with the explanation of the simulated waveforms. **Chapter 5** analyzes the results of this new design against the conventional CMOS design and in addition, gives a comparison of this architecture with other architectures that have been tried before. **Chapter 6** gives conclusion of the thesis describing the promise of this new design, areas where it needs improvement and a brief discussion about possible future work.

Chapter 2

Background

2.1 Spintronics: Basic Concept and Potential Applications

Conventional electronics is based on the use of charge of electrons. Usually, electrons in non-magnetic materials are aligned in such a way that their magnetic moments cancel out each other. Magnetism in a material is a result of electron spin which can either be up or down. Material's magnetism depends on the fact that the magnetic moment is not getting cancelled out due to an inequality in the number of spin-up and spin-down electrons. Spintronics (Spin Transport Electronics) is a field that leverages this spin aspect of an electron and this is what differentiates it from traditional electronics and makes it more attractive as a technology for the future. With the recent research advances in this field, we are now more efficiently able to control and manipulate the spin of the electron in the materials and thereby, can use it to our advantage. The phenomenon of Giant Magneto Resistance (GMR) in magnetic multi-layers is the direct result of the research that went in this area and is an important reason for the renewed interest in this field. For practical purposes and for getting the most out of this new phenomenon, large GMR values should be attainable at room temperature with the help of small magnetic fields and this was achieved by Parkin et al in 1991 where they showed that saturation magnetoresistance of

more than 65% exists in antiferromagnetic Co/Cu multilayers at 300k [5]. Further advances in this field came with the discovery of spin valves [6] where B. Dieny et al observed very large changes in resistance (8.7%) in a sandwich like structure consisting of two magnetically soft ferromagnetic layers with a noble metal layer in the middle. This was observed at room temperature and at a low magnetic field (< 20 Oe) [6]. Magnetization of one of the ferromagnetic layers is pinned using exchange bias to measure the relative variation in the orientation of the two layers [6]. The phenomenon of GMR in these sandwiches like structures depends on the fact that whether the two ferromagnetic layers are magnetically aligned or not. In a ferromagnetic-metal-ferromagnetic structure, if the two ferromagnetic layers are magnetically aligned, electrons pass through the structure easily attributing low resistance value to the structure in contrast to the case where they have opposite magnetizations, in which case the resistance value of the structure is relatively higher. Over the years, the main application of GMR came in the form of Magnetic Random Access Memories (MRAM) which got even a greater boost from the fact that these magnetoresistive devices have very low sensitivity to radiation [7]. Further, the use of GMR concept in the design of read head for magnetic disks revolutionized the magnetic disk industry by increasing the memory density to unprecedented levels.

A concept very similar to GMR, Tunneling between ferromagnetic films, was first put forward by M. Julliere [8] in 1975 where he observed the spin based tunneling phenomenon in Fe-Ge-Co structure at temperatures less than 4.2K. Terunobu Miyazaki

and Nobuki Tezukwas [9] and J.S. Moodera et al [10] working independently came up with a structure where they replaced the sandwiched noble metal used in spin valves with an insulator like Al_2O_3 or AlN. The thin film structure of the formation of ferromagnetic/insulator/ferromagnetic showed magnetoresistance of about 14% or higher at room temperature [10]. The basic working principle for these structures is quite similar to the spin valves where if the two ferromagnetic layers are aligned magnetically, the tunneling probability is higher and, therefore, the tunneling current is also higher whereas if they are anti-aligned, the tunneling probability is relatively lower which results in relatively higher resistance. Some of the novel features of these devices as explained by J.S. Moodera in his paper [10] include their use in MRAM, read-heads and in magnetic sensors. In addition to that, they have non-volatile memory properties, high sensitivity to magnetic fields; radiation resistance and gives out a large output signal so that there is no need for further amplification. These are the reasons which give "spin based tunneling" devices an advantage over GMR based devices. There are many other spin based devices which have been proposed over the years like Spin filters, Spin diodes and Spin transistors where spin transistors especially, attracted attention of many researchers and different types of transistors were proposed like Spin Field Effect Transistor, Magnetic Bipolar Transistor and Hot-Electron Spin Transistor but our focus in this chapter will be mainly on the Magnetic Tunnel Junction (MTJ) devices which are based on the concept of spin based tunneling.

2.2 Magnetic Tunnel Junction Device

Magnetic Tunnel junction is a tri-layer structure where a thin insulating tunneling barrier

material like aluminum oxide is sandwiched between two ferromagnetic materials. The device, whether, will have a high resistance or low resistance depends on the magnetic alignment of its two ferromagnetic layers. If the two layers, as explained in the previous section, have same magnetic alignment, the resistance will be low and if they are anti-aligned the resistance of the device will be high. The tunneling magnetoresistance in this device can be attributed to the majority and minority spins in these layers. The spin of an electron can either be up or down and in a ferromagnetic layer; the spin of the majority of electrons will also be either up or down. For making the explanation easier, let's assume that the spin of the majority of electrons is "up" for the layer from where the transport of electron is taking place. Now, if the two layers are aligned, the spin up electrons from a layer will be passing into the layer where they will encounter a majority of spin-up states. If the layers are anti aligned, the spin-up electrons will be forced to enter a layer where spin-down state is in majority. Spin-up electrons not being able to find sufficient number of spin-up states in the other layer results in large tunneling resistance; as compared to the case, where the layers are aligned [11]. These low and high resistance values serve as low-0 and high-1 logic states respectively for storing data. Ideally, the difference between the resistances in these states should be considerably high as then, the reading and writing of these devices will be relatively fault free and they will be more noise-resistant during their operation.

Fig. 2.1 shows the basic structure of an MTJ device with two ferromagnetic layers which can either be fixed or free and an interspersed insulator layer. Usually, one of the layers is made to stick to a particular alignment so that the alignment of the other layer can be

modified to set the device to low or high resistance value. The fixed layer is usually pinned using a process called anisotropic exchange where a hard magnetized behavior of anti-ferromagnetic substance is used to fix the magnetization of the soft ferromagnetic layer which serves as fixed layer in our device. Once a layer is fixed to a particular magnetic alignment, the other layer can be made to switch its alignment either using

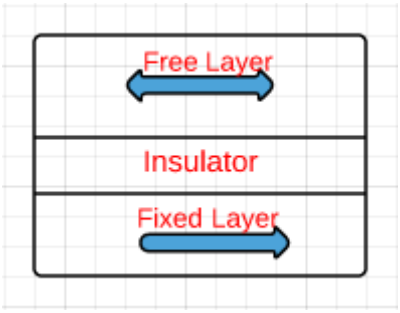

Fig 2.1 Basic MTJ device

an externally applied magnetic field or by passing a value of current above a certain value through the device perpendicular to the tri-layer structure. If external magnetic field is used to switch the magnetization of the free layer, it is called Field Induced Magnetization Switching (FIMS) and if current is used for the same purpose, it is called Current Induced Magnetization Switching (CIMS). In CIMS, the direction of the current through the device (top to bottom or vice versa) decides the magnetization of the free layer. CIMS is based on a technique called Spin Transfer Torque (STT) where a spin-polarized current can modify the magnetization of a magnetic layer in an MTJ device.

2.3 Related work

The idea to implement 4 bit SLIC switch based adders came from one of the basic works done by Shruti Patil in her Ph.d thesis on Spintronic Processing in Memory [3] where she was able to use the MTJ devices and NEMS based switches to perform bit-wise logic

operations. The next logical step, after the realization of these bit-wise operations is to somehow use them to perform a series of logic operations so that we are able to realize some useful functions which are required in normal day-to-day computing these days. That gave us the motivation to realize an adder circuit using the same Spintronic Logic In Cache (SLIC) architecture.

In a conventional processor, the memory and logic functions are performed by separate units and the difference in operational speeds of memory and logic units created bandwidth problems for many applications. As a result, there was a conscious effort in the past few decades to build Processor-In-Memory structures where processors were integrated in the DRAM memory to increase bandwidth between memory and processor. But even with these types of structures, the memory and processor units were separate. But with SLIC approach, the memory and logic operations can really be performed by the same unit. Fig. 2.2 shows the basic structure used by Patil et al [3] in realizing the NAND operation with MTJ devices and NEMS switches.

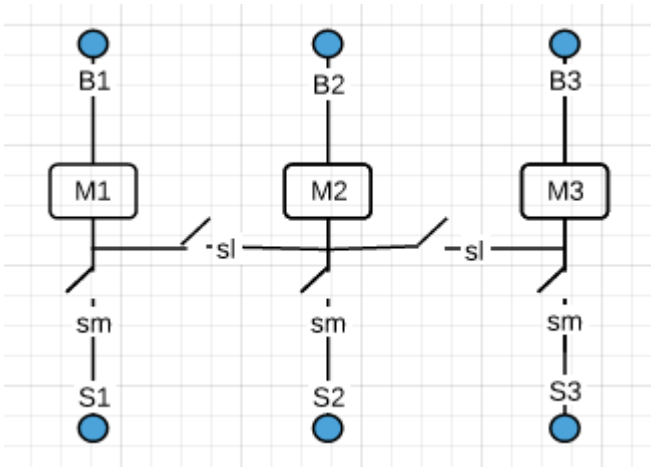

Fig 2.2 NAND Function Realization

Three MTJ devices were used where M1 and M2 hold the initial values or the operands

and the final result of the NAND operation is stored in M3. Five switches were used; 3 sm (switch memory) switches and 2 sl (switch logic) switches. Here, two types (sl and sm) of switches were used so that the same device can be used to store values as well as to perform logic operation on those values. These switches will help differentiate between the 2 modes of the circuit: Memory and Logic. Complete NAND operation can be broken in 4 steps: Write M1, Write M2, Preset M3 to 0 and Perform the NAND operation. For writing M1 and M2 and for presetting M3, "sm" switches are turned on and suitable bias voltage is applied to S1, S2 and S3 to attribute initial values to MTJ devices. Once the MTJ devices are initialized, "sl" switches are turned on and the bias voltages at which the circuit performs a NAND operation are applied on B1 and B2 terminals with the ground voltage being applied on B3.

To make a decision on the types of switches that can be used with MTJ devices, simulations were performed to check the sensitivity (Rate of increase of bias voltages to perform 4 steps in a NAND operation as explained above) of the circuit depending on the non-ideality of the switches where non-ideality refers to the switches having finite off and on resistance and finite switching time. Different combinations of Off and On resistances were used from (0, infinity) till (5K, 10M) and it was concluded that from (0, infinity) till (1K, 100M), the rate of increase of bias voltage doesn't vary much with different combinations of Off and On resistances which led Patil et al [3] to make an assumption that any existing technology can be used to realize switches for these MTJ based circuits. Finally, they used NEMS switches to implement basic bit-wise operations

in these SLIC units and that is what we will be using in our design going forward. In addition to the SLIC based full-adder design that has been proposed in this thesis, there are some other interesting MTJ based adder designs. A Spintronics full adder was proposed by Hao Meng et al [13] where they were able to realize a full adder using just seven MTJ elements. A full adder design using serially connected MTJ devices has been shown to have low power consumption, high speed and high density [14]. [15] shows MTJ devices used along with CMOS devices to study the advantages and implications of having MTJ integrated with CMOS technology. In addition to that, other devices like MTJ based Multiplexer and De-multiplexer [16], 3-bit gray counter [17] and a non-volatile flip-flop [18] were also proposed and considering the novel features like non-volatility, high density, high speed etc, Spintronics field will continue to gain traction among researchers.

Chapter 3

Circuit Design

After the successful realization of bit wise operations using SLIC architecture by Patil et al [3], the obvious next effort has to be in the direction of answering whether the same approach can be used to perform a chain of logic operations and that is what a part of this thesis tries to answer. We started with the basic NAND circuit as shown in Fig. 2.2 and tried to come up with more complex circuits like XNOR that can be directly used in the circuits to get useful functionalities like 2-bit comparator, 4 bit adder etc. With the CMOS based structures of adders, comparators etc in mind, we tried to build spintronics components on the gate level first and then, as there are different levels of logic in CMOS, we tried to achieve the same logic structure in spintronics.

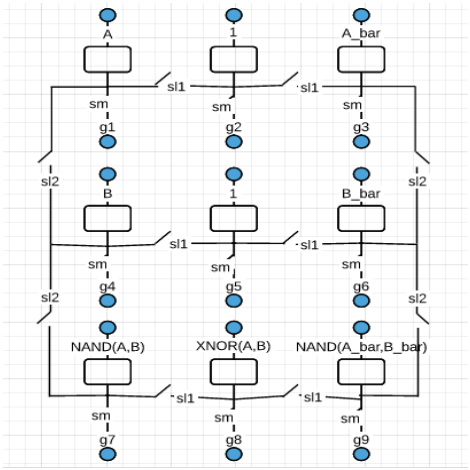

Fig. 3.1 XNOR SLIC circuit

Fig. 3.1 shows the XNOR SLIC circuit that is basically a combination of NAND SLIC [3] circuits in which an MTJ in one NAND circuit can be used as a part of other NAND

circuit depending on the positioning of the switches. Here is a brief explanation of how this XNOR circuit actually works. Let's break this up in terms of NAND circuits. In all, there are 5 NAND circuits in this XNOR circuit, 3 NAND circuits arranged horizontally and 2 NAND circuits arranged vertically on the vertical edges of the Fig. 3.1. Horizontal NAND circuits are connected together with 'sl1' switches and vertical circuits with 'sl2' so as to make sure that only one type of circuit is actually functioning at any moment. This is made sure by switching on either 'sl1' or 'sl2' at a time. Firstly, every MTJ has to be set to its initial value by switching on 'sm' switches, and then in the top 2 horizontal rows which are activated by switching on 'sl1', values of A and B are inverted by NANDing them with 1's. Once we have A, B, A_bar and B_bar values, 'sl2's' are switched on to activate the vertical rows which again initializes the values in 2 MTJs at the bottom edges. 'sl1' switches are again turned on to get the final XNOR result in the middle MTJ. In this way, if we have to use the XNOR (A, B) value anywhere else, we will just add more switches to the MTJ holding the XNOR value and again NAND it with some other value.

For example, if we need to realize $\overline{A \oplus B}.C$, we first get the value $\overline{A \oplus B}$ as explained above. The MTJ holding the value of $\overline{A \oplus B}$ becomes an input MTJ for another NAND circuit where the other input MTJ will be set to value 'C'; the NAND operation (by switching on 'sl2') on these 2 values will give us $\overline{\overline{A \oplus B}.C}$ in the resultant MTJ. In turn, this MTJ will again be used to NAND with an MTJ holding value '1' (by switching on 'sl1' for the third time) which will invert the function we got in the previous step to give

$\overline{A \oplus B}.C$ in the final MTJ. Fig. 3.2 shows the circuit diagram to realize this function. It is quite evident from this

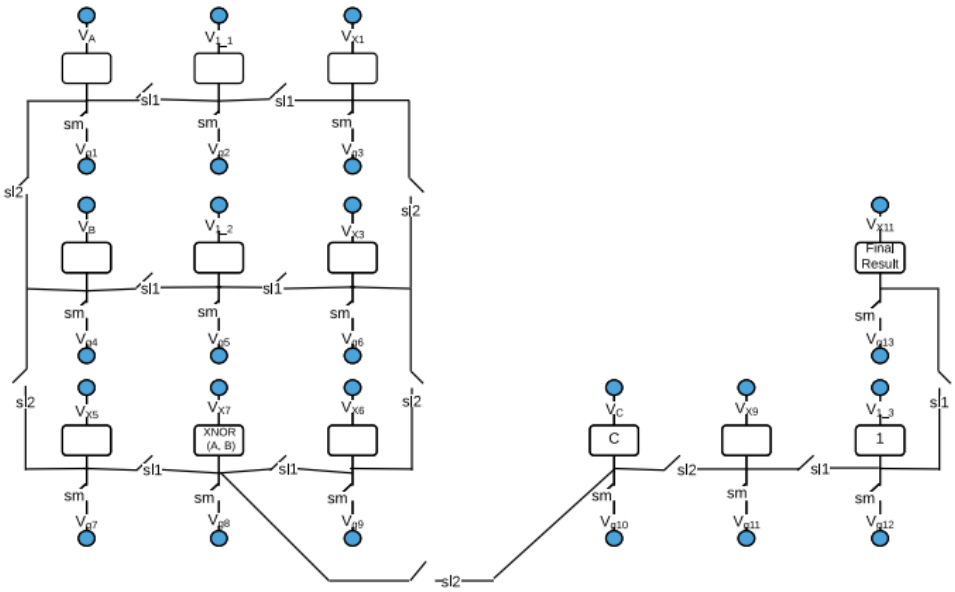

Fig. 3.2 Circuit diagram for $\overline{A \oplus B}.C$

example that this methodology can be applied to realize any CMOS based circuit. Although, there is an added cost to this approach because of the fact that we are using just one function (NAND) to realize every other function as we have to invert the results to get the actual results. This can be solved by using the same 3 MTJs as we used in a NAND SLIC circuit to perform, say an OR function but in that case, the bias voltage required for this to happen will be different from the one that we used in the case of NAND SLIC circuit. So, to keep things uniform, we tried to realize every function using NAND.

Another important thing that can be mentioned here is that using the basic NAND circuit shown in Fig. 2.2, all the other basic gates like AND, OR, NOR and XOR can be

designed as demonstrated by Patil et al [3]. Here, the same voltage biases were used but some extra number of MTJs is added to the design to get all other basic gates. Effectively, it is like designing all other basic gates using the NAND gate. For e.g.: To build an AND gate, we just invert the output of the NAND gate; to build an OR gate (A+B), we actually implement $\overline{\overline{A}.\overline{B}}$ where we first invert A by NANDing it with 1, then, invert B in the same way and then, NAND $\overline{A}$ and $\overline{B}$ to get the OR functionality (Refer to [3] for more details). Once we have all these basic gates, we can actually formalize the process to build any circuit based on this SLIC [3] approach. First, we need to have the gate based CMOS description of the design. This design will consist of one or more of NAND, AND, OR, NOR, XOR and XNOR gates. Replace these gates with the SLIC based gates described in [3]. Make sure that the gates are isolated from each other using different switches so that they work independently and don't interfere with each other.

Following example of circuit design will be able to demonstrate this process clearly. Instead of straightaway designing a complex circuit like 4-bit full adder, we started with a comparatively simpler 2-bit comparator circuit. We first focused on the gate level design of the 2 bit comparator which can be seen in the Fig. 3.3 given below.

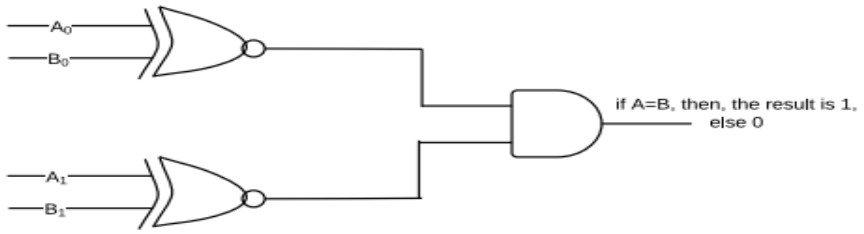

Fig. 3.3 2-bit Comparator Circuit

As the figure demonstrates, the design has 2 XNOR gates and an AND gate. So, we have 2 XNOR SLIC circuits that we have directly used in the design and the AND functionality is achieved in the same way as we did it in the last example, by first NANDing the outputs from 2 XNORs and then, again NANDing it with '1'. If the value in the final MTJ comes out to be '1', then A=B, otherwise the 2 bits of 'A' are not equal to the 2 bits of 'B'. Fig 3.4 shows the circuit design of a SLIC based 2 bit comparator.

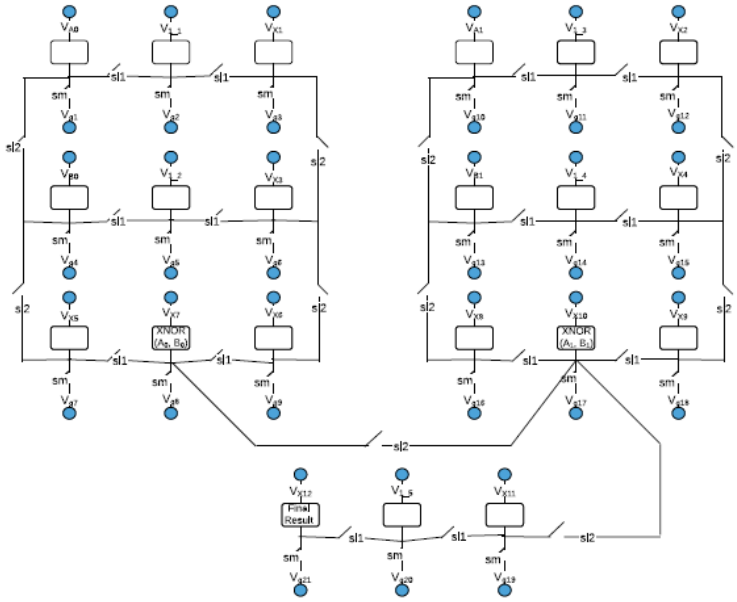

Fig. 3.4 2-bit SLIC Comparator Circuit

We were able to verify the functionality of this design by seeing the resistance waveforms (Resistance Value of an MTJ decides if it is in '1- High' state or '0-Low' state) for every MTJ that we used in the circuit. Two big blocks of XNOR will calculate their values simultaneously after which 'sl2' will be switched on again, which will give out the result in 'X_{11}' MTJ which again will be inverted to give the final result in 'X_{12}' MTJ.

After the successful verification of 2-bit SLIC based comparator design, we went on to design 4-bit SLIC based Ripple Carry Adder (RCA) and Carry Look-ahead Adder (CLA) with the same approach as we used in the previous designs, focusing on the gate level CMOS design of 4-bit RCA and then, realizing it by using basic SLIC gates.

As the design of a 4 bit adder is quite complex, we have broken it down bit by bit. Firstly, we will have a look at the 4-bit Ripple Carry Adder (RCA) which is shown in Figs. 3.5, 3.6, 3.7, 3.8. Each figure here shows the design for a single bit which has been broken in 2 parts which are highlighted using Blue and Red colors.

The part of the circuit shown in Blue is generating carry for the next bit. MTJs holding important results that leads to the final carry being generated are highlighted in light blue color. Each such MTJ is marked with the result that it is holding. For the generation of carry for next bit, we first have XOR ($A_{x,}$ B_x) which is NANDed with the incoming carry and the result is held in the next MTJ also highlighted in light blue. From the other end, we generate NAND ($A_{x,}$ B_x) which is NANDed with the result that we got in the previous MTJ to get the carry for the next bit (C_{x+1}) which is highlighted in Green color. This same carry is shown in the next bit diagram with dashed lines attached to it showing the switch connections and highlighted in Red color.

The part of the circuit shown in Red is generating the sum bit. The first block in that part of the circuit is performing XOR operation on ($C_{x-1,}$ B_x) and the result is held in the MTJ marked in Red color. This MTJ is again XORed with A_x to get to the Sum bit (S_x) which is highlighted in Red color too.

The explanation of the 1st bit of RCA based on the functioning of switches is as follows:

Considering the **Carry** circuitry (see Blue highlighted part of Fig. 3.5), When V_{sm} =1, we set the values for MTJs A0_1, B0_1, C_{in}, 1_1, 1_2, A0_3 and B0_3 and preset X7, X8, X1, X2 and X14 (MTJs that are going to hold results for NAND operations) to 0. When V_{sm} = 0, firstly, V_{sl1} =1 which gives us results in X7, X8 and X14. As we now have input values to perform further NAND operations and V_{sm} is still 0, V_{sl1} is brought down to 0 and V_{sl2} = 1, these settings of switches will give us results in X1 and X2. This competes 1 cycle of V_{sm}.

When V_{sm} is again made 1, we preset X3 and X16 to 0. Now, when V_{sm} = 0, V_{sl1}=1 will give us the result in X3. As V_{sm} is still 0, V_{sl1}=0 and V_{sl3} =1 which performs a NAND operation on C_{in} and X3 to give us the result in X16. This completes the second cycle for V_{sm}. Again, V_{sm} goes to 1 and we preset X15 (MTJ that is going to hold the next carry bit) to 0. Next, V_{sm} is brought down to 0 and V_{sl2} = 1 because of which a NAND operation takes place between X16 and X14 to provide us with the final result for this carry circuitry i.e. the carry for the next bit in X15 MTJ. This means that we get the carry for the next bit in 2 complete and a ¾ of V_{sm} cycle.

Now moving on to the **Sum** circuitry (see Red highlighted part of Fig. 3.5), When V_{sm} =1, we set the values for MTJs C_{in}, 1_3, B0_2, 1_4, 1_5, 1_6 and A0_2 and preset X17, X18, X4, X6 and X5 (MTJs that are going to hold results for NAND operations) to 0. When V_{sm} is brought down to 0, firstly, V_{sl1} =1 which gives us results in X17 and X18. As we now have values to perform further NAND operations on input MTJ pairs ((C_{in}, X18) and (X17, B0_2)) and V_{sm} is still 0, V_{sl1} is brought down to 0 and V_{sl2} = 1, this setting of switches will give us results in X4 and X5. This competes 1 cycle of V_{sm} which

is the same one as we used in Carry calculation which means that Sum and Carry calculation is being done in parallel in this cycle.

When V_{sm} is again made 1, we preset X5, X9 and X10 to 0. Now, when V_{sm} = 0, V_{sl1}=1 will give us the result in X5 due to a NAND operation on X4 and X6. As V_{sm} is still 0, V_{sl1}=0 and V_{sl3} =1, NAND operation on X5 and 1_5 gives us the result in X9 and another NAND operation on A0_2 and 1_6 gives us the result in X10. This completes the second cycle for V_{sm} again in parallel to carry calculation. Again, V_{sm} goes to 1 and we preset X11, X13 and X12 (MTJ that is going to hold the sum bit) to 0. Next, V_{sm} is brought down to 0 and V_{sl2} = 1 because of which a NAND operation takes place between X5 and X10 to provide us the value in X11 and another NAND operation between X9 and A0_2 to give us the value in X13. As V_{sm} is still 0, V_{sl2} is brought down to 0 and V_{sl3}=1, this switch setting performs a final NAND operation on X11 and X13 to give us the first sum bit result in X12 MTJ. This effectively means that the calculation of one sum bit takes 3 complete V_{sm} cycles and it happens in parallel to the carry calculation. Sum and carry calculations for 2^{nd}, 3^{rd} and 4^{th} bits happen in the same way.

The next bit calculations start only when we have the sum and carry results from the previous bit. Fig. 4.2 shows the waveforms for all the switches and signals that we talked about and might help in giving a clear idea of the process that we followed.

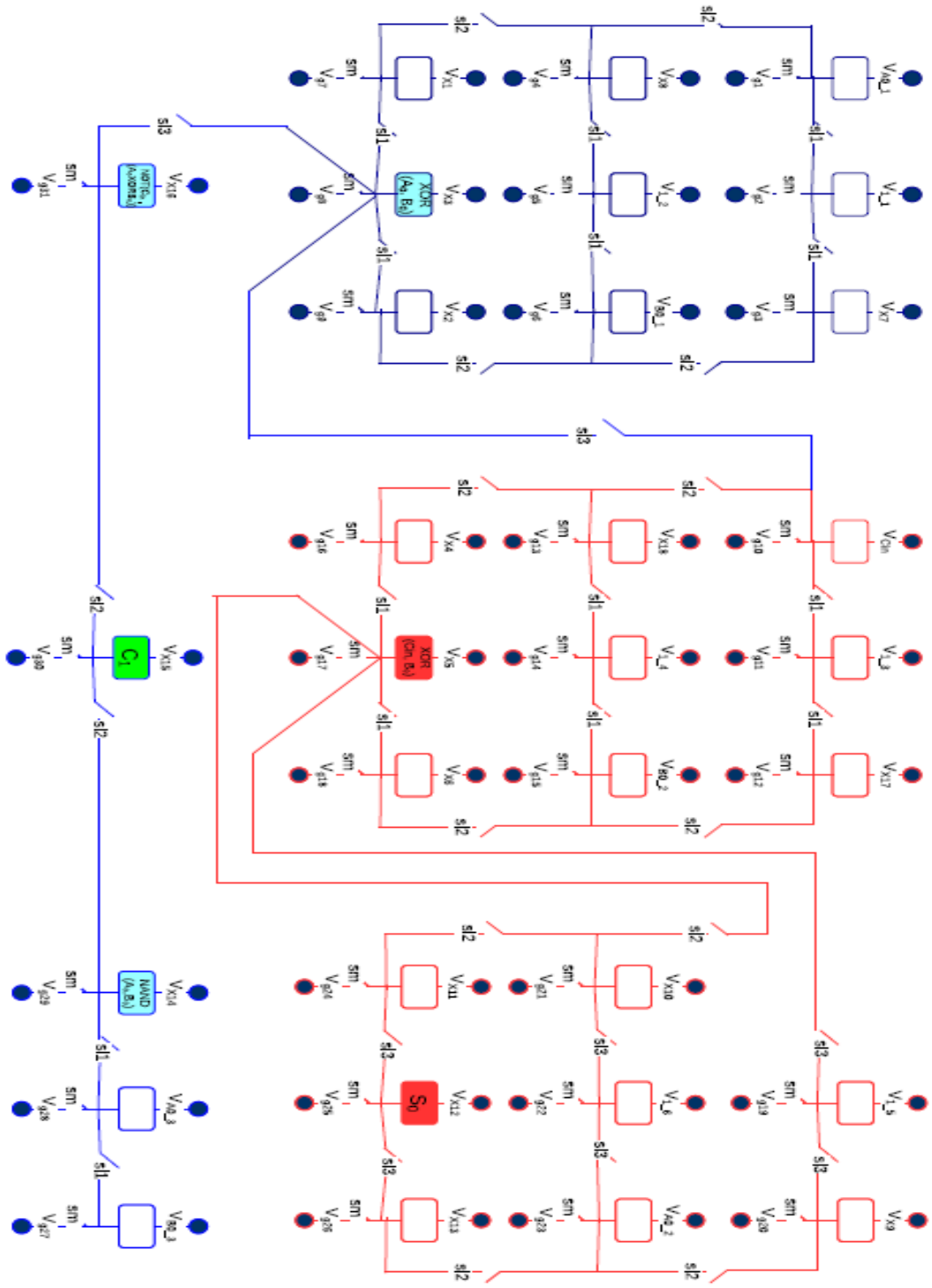

Fig. 3.5 RCA - 1st bit

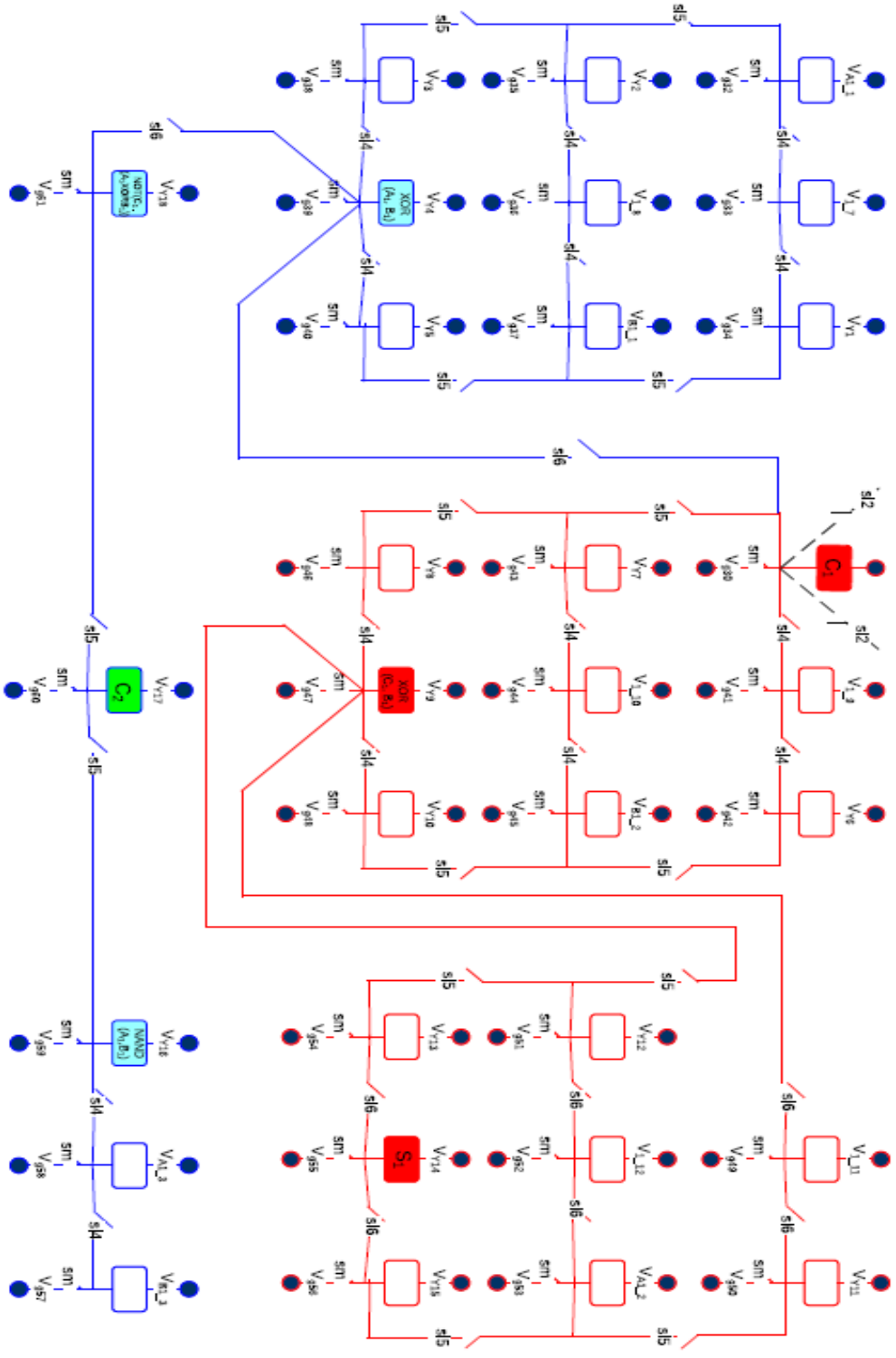

Fig. 3.6 RCA – 2nd bit

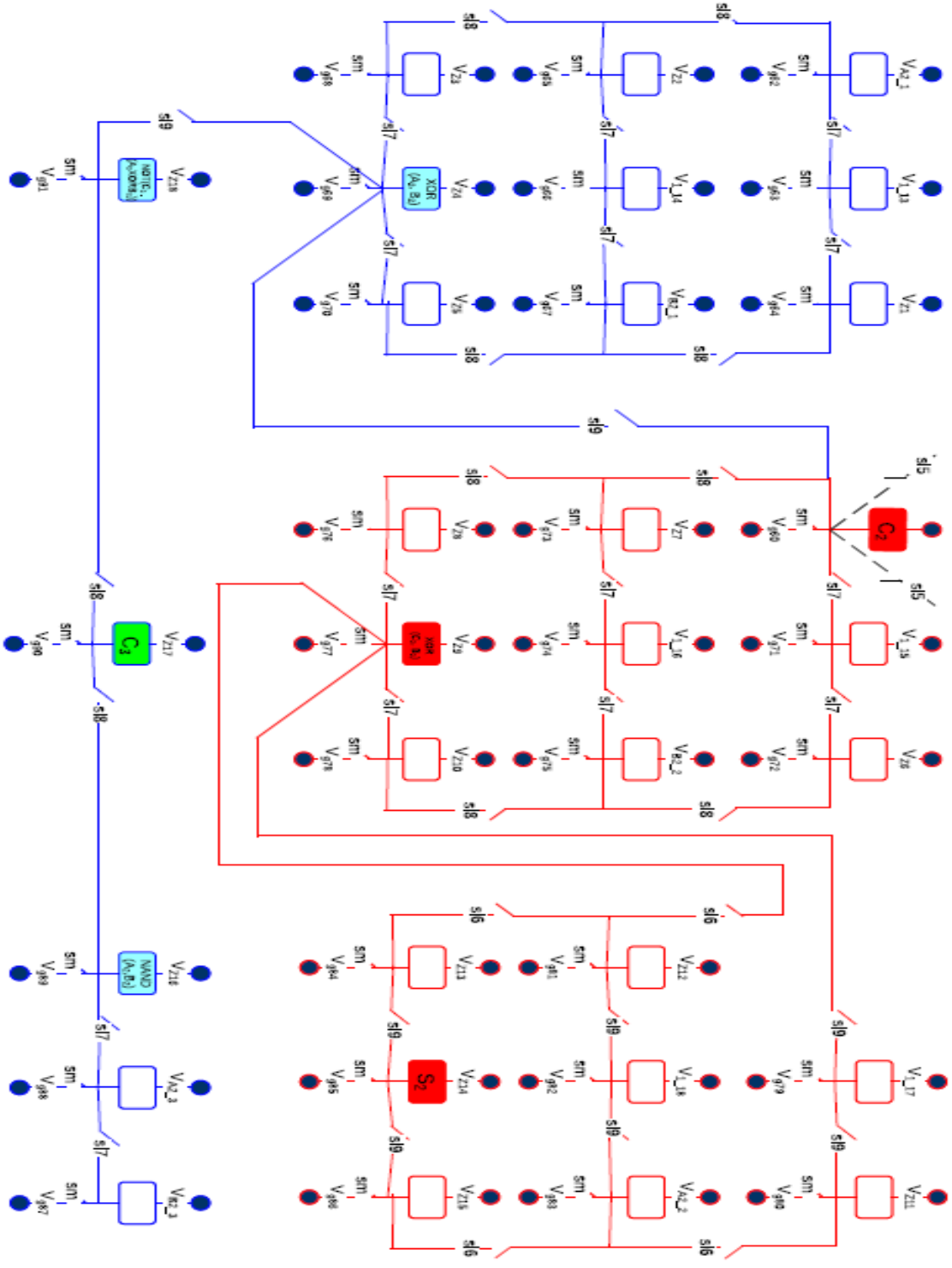

Fig. 3.7 RCA – 3rd bit

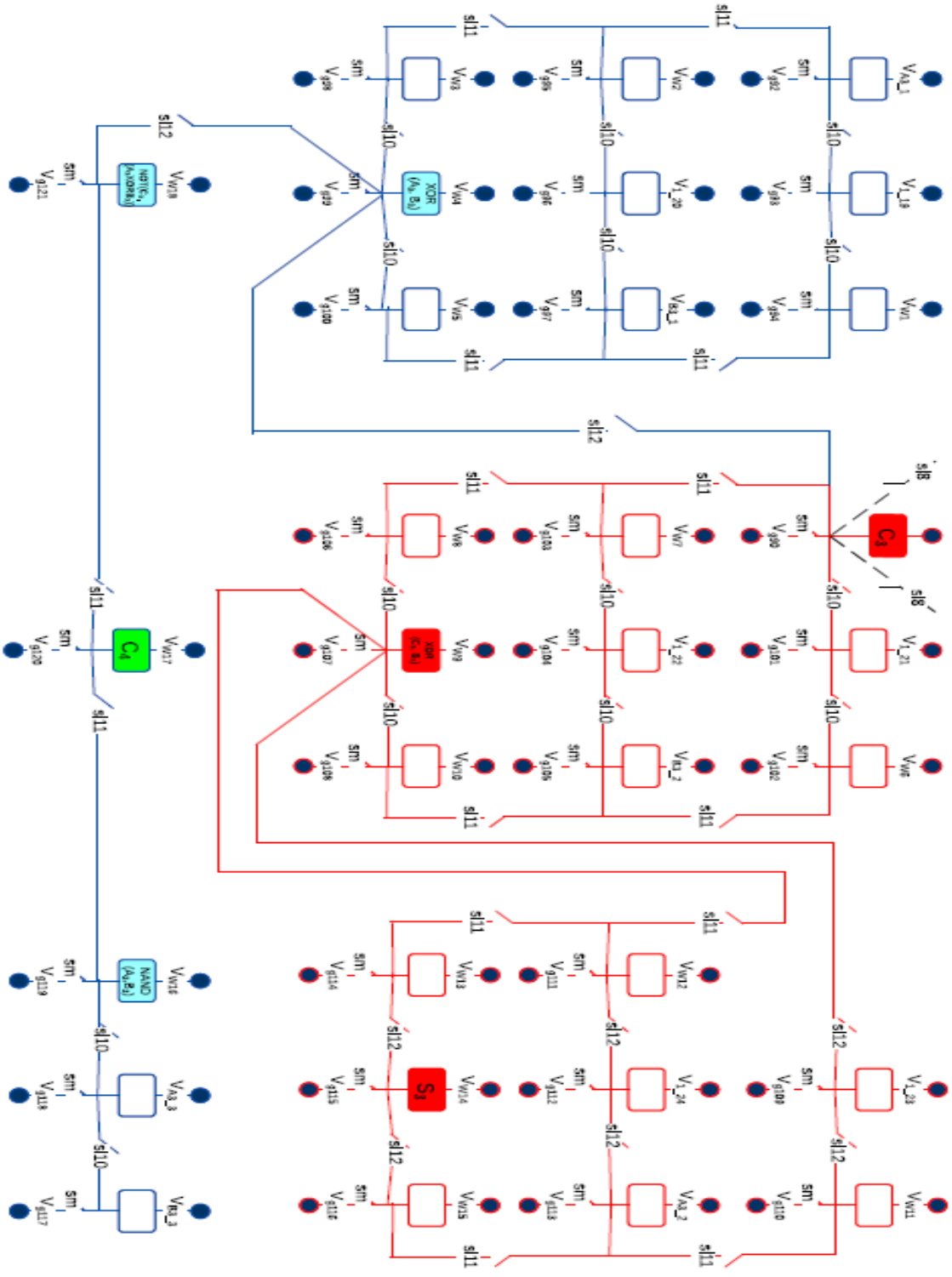

Fig. 3.8 RCA – 4th bit

Figures 3.9, 3.10, 3.11 and 3.12 show the 1st, 2nd, 3rd and 4th bits respectively for the 4-bit Carry Look-ahead Adder (CLA). Here, the complexity of the design increases with the bit as we are trying to generate the next carry for each bit without waiting for the carry to be generated from the previous bit which is what Carry Look-ahead Adder is supposed to do. This makes the design a lot faster as compared to the Ripple Carry Adder design.

In these figures, the circuit for each bit design is shown in 3 different colors. The part of the circuit shown in Blue is exclusively for carry generation; part in Red is exclusively for sum generation and the part in Black is the circuitry that is common to both sum and carry generation. This optimization saved us quite a few numbers of MTJs in the overall design. The circuitry for sum generation is exactly similar to the circuitry that we have in the RCA design.

First, the common circuitry shown in Black generates the Propagate term (Px- Highlighted in gray color) or in other words XOR (Ax, Bx) which is again XORed with Cx to give out the sum bit (S_x) which is highlighted in Red color. $S_x = P_x \oplus C_x$ is the expression that we realized to get the sum bit where $P_x = A_x \oplus B_x$.

Carry generation uses the expression $C_{x+1} = G_x + P_x C_x$ where the Generate term is $G_x = A_x . B_x$. Contrary to the sum generation which depends on the previous carry bit to be generated, carry for each bit is generated independently of any value from the previous bit. As we need to have all the previous Generate (G_x) and Propagate (P_x) terms to generate carry for that term, that is the main reason for the exponential growth in the carry generation circuitry with each bit. MTJs holding important results are highlighted in

Light Blue color again with the results marked inside them which finally converge to give out the carry term highlighted in Green color.

While passing the values from the common circuitry (marked in Black) to either carry or sum circuitry, preference is always given to the carry circuitry so that parallel carry generation can be as fast as possible which also lies on the critical path of the circuit and decides the overall performance of the circuit. The value generated in the common circuitry is first passed on to the carry circuitry by turning 'on' the switches connecting them and then, it is passed on to the sum circuitry.

Now, if we look at the 1^{st} bit of CLA from the point of view of sequence in which switches are turned ON, it is almost the same as the 1^{st} bit of RCA with the only difference that here we are sharing the circuitry for Sum and Carry due to which we are able to get rid of one XOR circuit. The difference between RCA and CLA designs shows up only in the later bits that too only in the Carry circuitry. Switches are turned ON in a sequence so as to propagate the value to the final Sum and Carry MTJ as explained in the RCA section above. Here too, in one V_{sm} cycle, we are performing 2 NAND operations and the results from those 2 operations becomes the inputs for further NAND operations in the next V_{sm} cycle and this continues until we get the value in the Sum and Carry MTJ for that bit.

The common circuitries for all 4 CLA bits start their calculations simultaneously as they have the same combinations of switches in them. While Carry circuitries don't require any values from the previous bits and function independently, the Sum circuitries do depend on the carry values from the previous bits and therefore, produce the result only

after that carry value becomes available. Fig 4.4 shows the switch and MTJ waveforms for all 4 bits of CLA. As calculations in CLA are happening simultaneously, we can't describe the duration of Sum and Carry calculations in terms of **fixed** number of V_{sm} cycles.

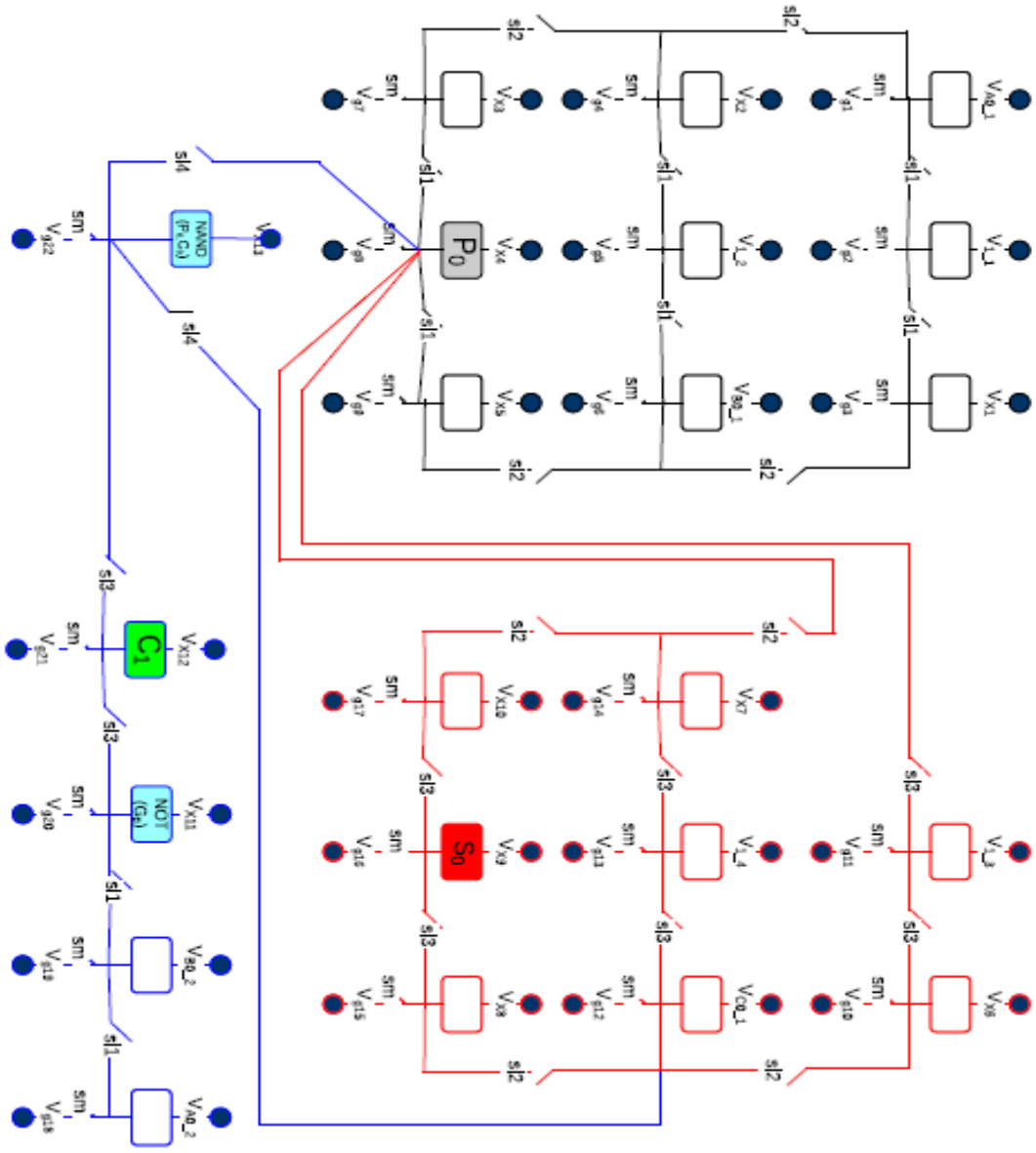

Fig. 3.9 CLA – 1st bit

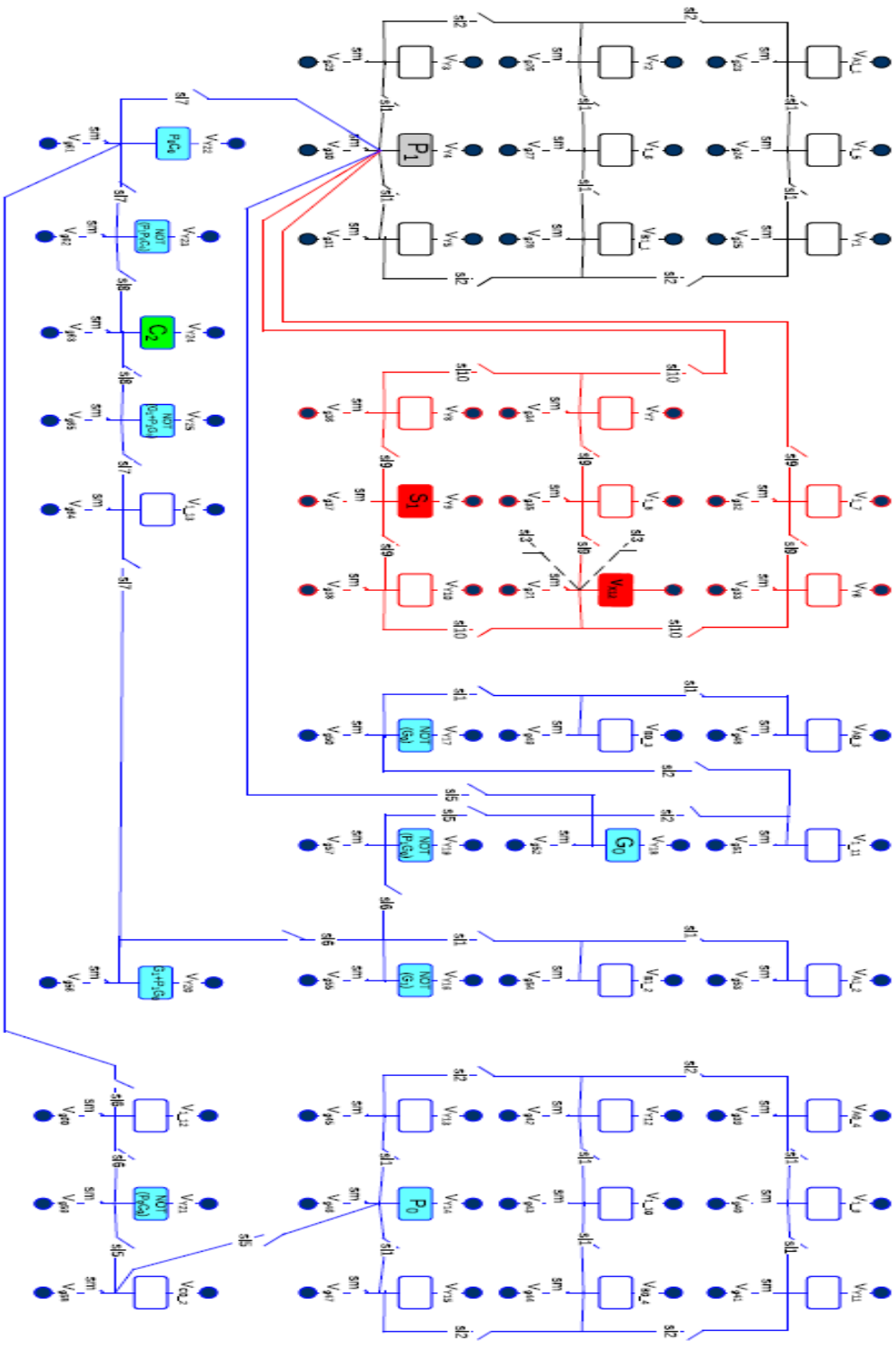

Fig. 3.10 CLA – 2nd bit

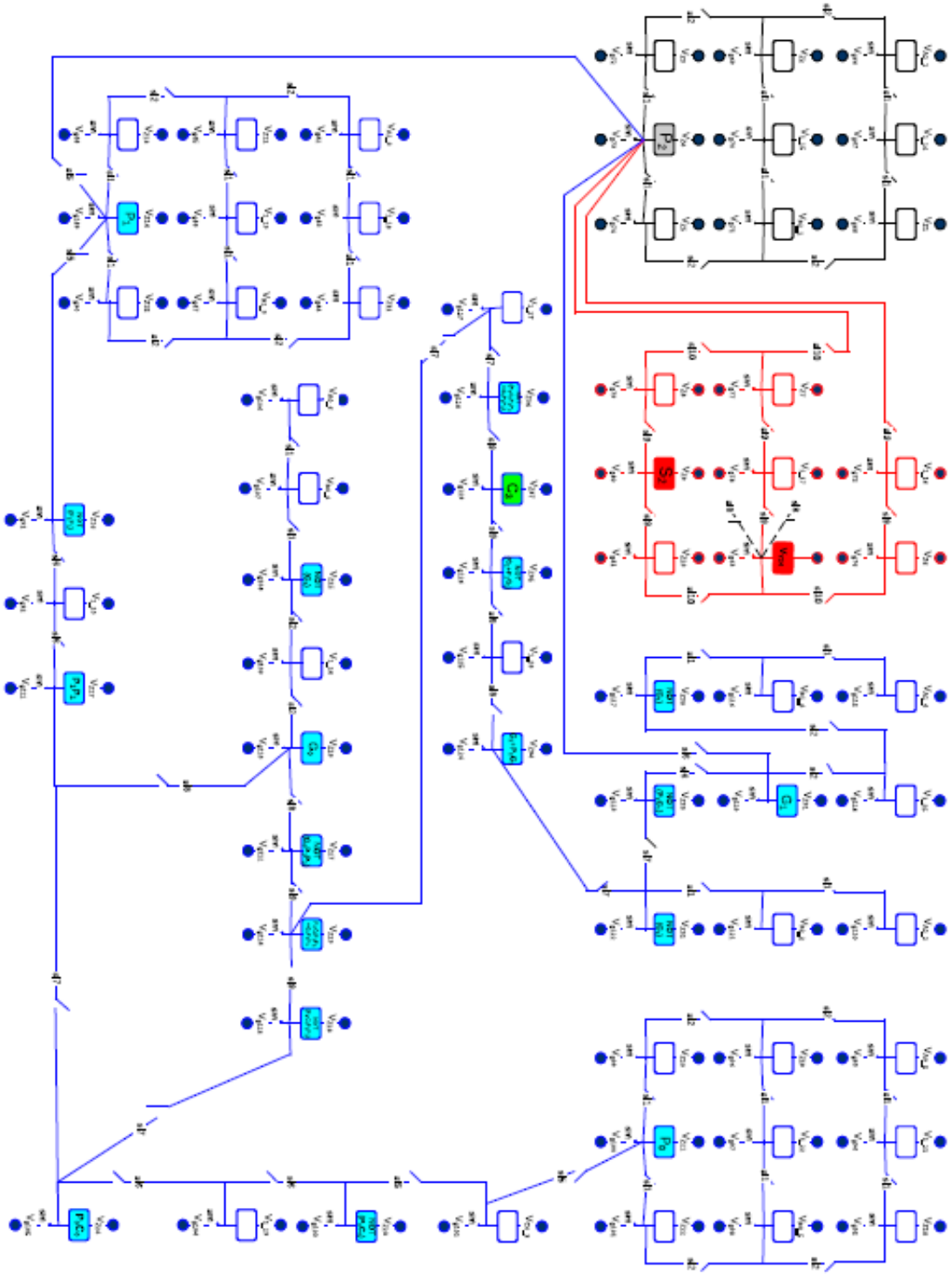

Fig. 3.11 CLA – 3rd bit

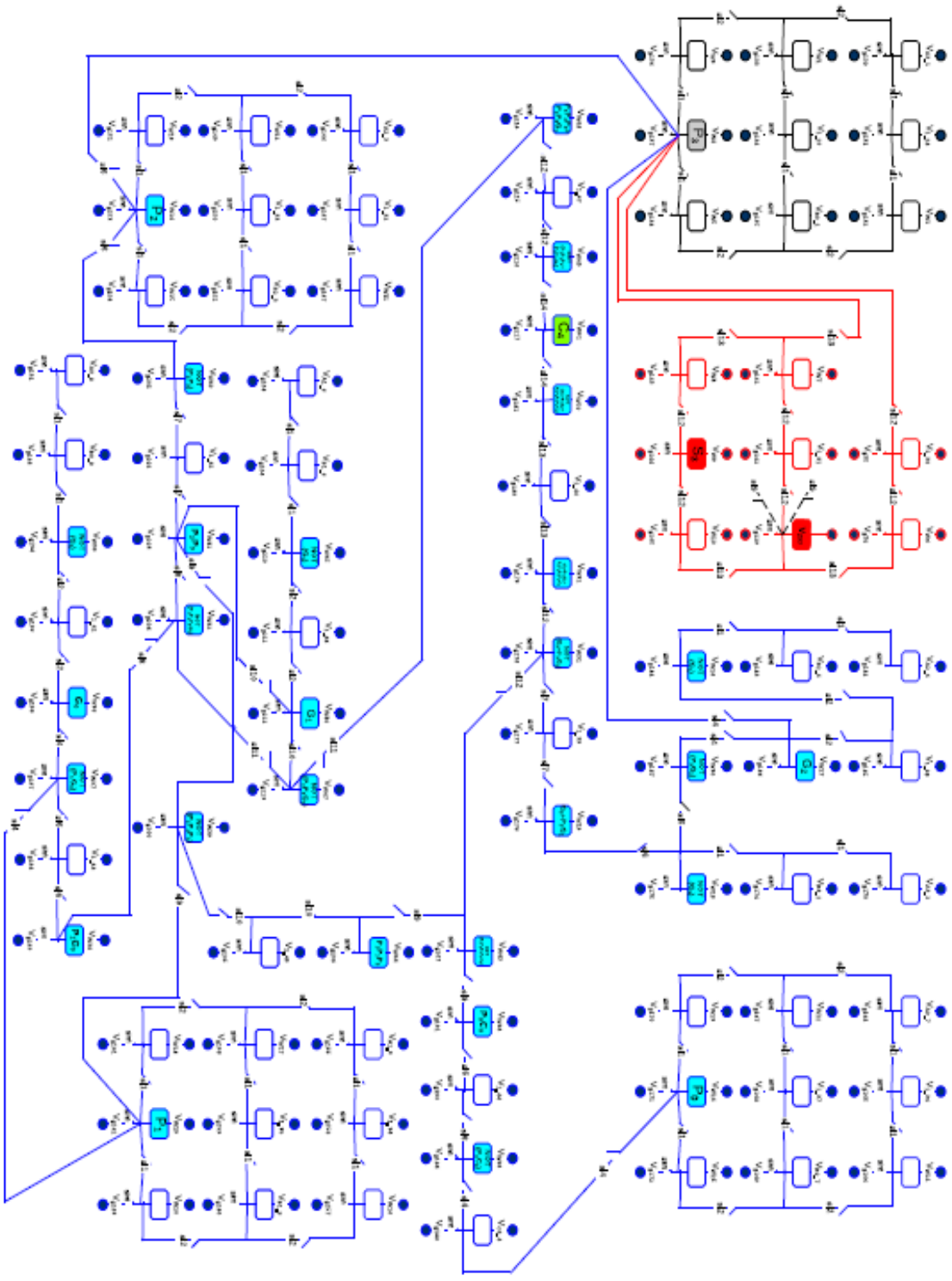

Fig. 3.12 CLA – 4th bit

Chapter 4

Simulations and Results

In order to verify the functionalities of 4-bit Ripple Carry adder and 4-bit Carry Look-ahead adder circuits as described in the previous chapter, these circuits were simulated using HSPICE. Every SLIC circuit effectively includes MTJs, NEMS switches and interconnects. The other important aspect is the bias voltage that we apply to each MTJ to get the required operation from the circuit.

The MTJ macro-model used in the simulation of the adder circuitry is the SPICE model created by Harms et al [19] which is a two-terminal device whose device parameters include R_{low} or R_P (Resistance of the MTJ device in Parallel state), R_{high} or R_{AP} (Resistance of the MTJ device in Anti-Parallel state), I_{CP} (Critical current for switching to Parallel state) and I_{CAP} (Critical current for switching to Anti-Parallel state) among other parameters. The values for these parameters were used from a fabricated device of size 120nm * 240nm described in [20] where R_{low} or R_P =3472Ω, R_{high} or R_{AP}= 5902Ω and I_{CP}=I_{CAP}=319μA.

The NEMS switch used in the circuit to initialize the values of MTJ and to enable logic operation between them was modeled by Patil et al [3] and was simulated using VERILOG-A. The R_{OFF} (Open Switch Resistance) for the NEMS switch is modeled at 10e8 Ω and the R_{ON} (Closed Switch Resistance) is kept at 100 Ω; V_t (Threshold Voltage for switching) is modeled at 1V.

The voltage that needs to be applied to turn on the switch is 1.8V. Considering the NAND function realization in Fig. 2.2, the bias voltages for writing or initializing MTJs (i.e. V_{S1}, $V_{S2,}$ V_{S3}) and the bias voltages for performing NAND operation (i.e. V_{B1} and V_{B2}) are all 3.8V. V_{B3}, the bias voltage for the MTJ holding the result will always be 0V during the NAND operation. V_{SL} (switch for logic) = V_{SM} (switch for memory) = 1.8V is the voltage at which the switches will be in closed position or will be ON.

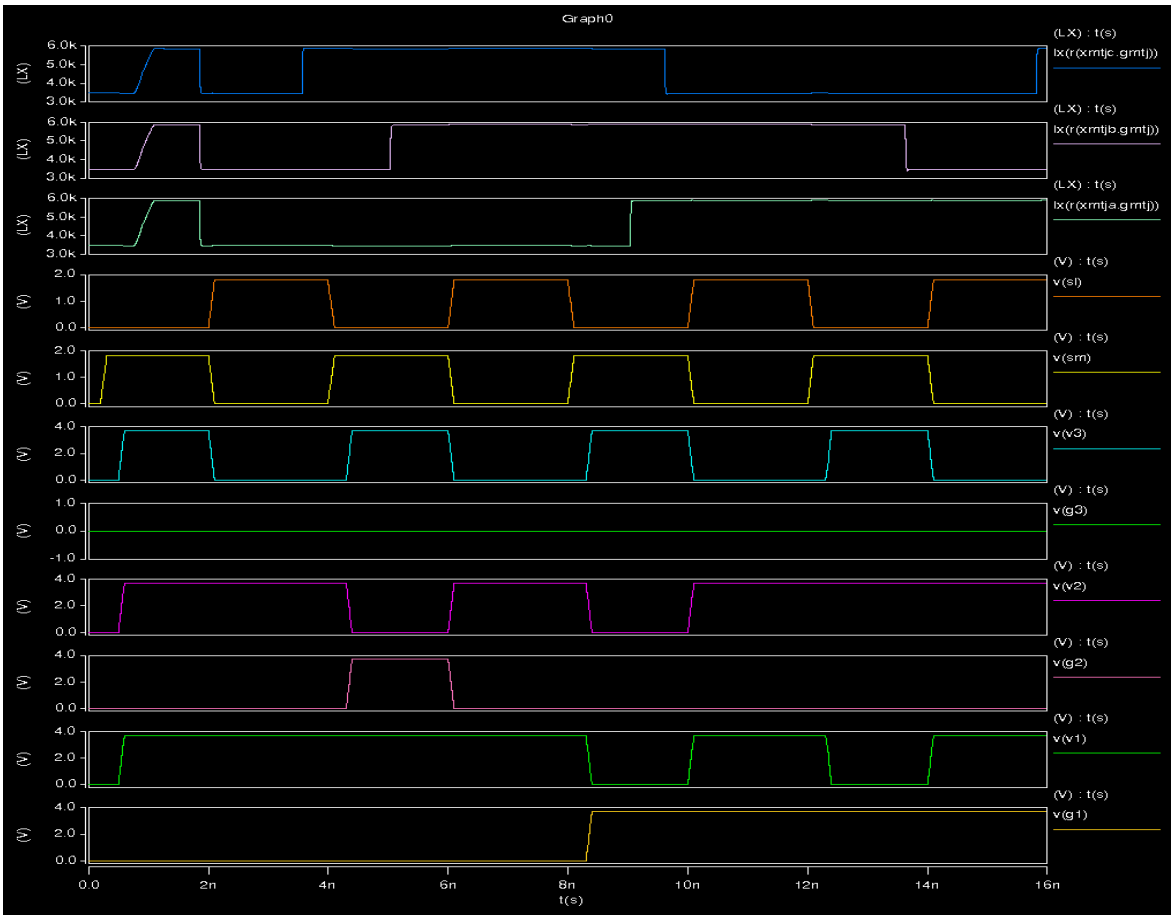

Fig. 4.1 NAND SLIC Circuit Waveforms

As the complete adder circuitry, whether it is Ripple Carry adder or Carry Look-ahead adder (Fig. 3.5 through Fig. 3.12), is made up of a combination of individual NAND circuits depicted in Fig. 2.2 attached together using switches, we used the same bias and

switch voltage values for all the circuits as we used in this simple NAND circuit. Fig. 4.1 shows exactly what kind of voltage waveforms are applied to the NAND circuit and how resistances of MTJs respond to these voltage waveforms. All the applied voltage waveforms are square-shaped.

As mentioned earlier, HSPICE was used to generate these square waveforms. Comparing these waveforms to the voltage signals in Fig. 2.2, we can establish the correspondence as follows:

Circuit	≡	Waveforms
(M1, M2, M3)	≡	(MTJ_A, MTJ_B, MTJ_C)
(V_{B1}, $V_{B2,}$ V_{B3})	≡	(V_{v1}, $V_{v2,}$ V_{v3})
(V_{S1}, $V_{S2,}$ V_{S3})	≡	(V_{g1}, $V_{g2,}$ V_{g3})
(V_{SL}, V_{SM})	≡	(V_{sl}, V_{sm})

One cycle for the NAND circuit is decided by the waveform of the V_{sm} signal. When V_{sm} (memory switch) is high, circuit is ready to be written anywhere within that duration which can be done by making (V_{v1}, $V_{v2,}$ V_{v3}) or (V_{g1}, $V_{g2,}$ V_{g3}) high. If either of V_{v1}, V_{v2} or V_{v3} is high when V_{sm} is high, it will set the MTJ corresponding to that signal to 1 (High Resistance value) as shown in Fig. 4.1. If V_{g1}, V_{g2} or V_{g3} is high when V_{sm} is high, it will set the MTJ corresponding to that signal to 0 (Low Resistance value). All the logic calculations are performed when V_{sm} is low and V_{sl} (logic switch) is high. NOTE: V_{sl} is high only during low phase of V_{sm} waveform. Once the values of MTJs are initialized, during the phase where V_{sl} = 0 (0V) and V_{sm} =1 (1.8V), we set

(V_{B1}, $V_{B2,}$ V_{B3}) ≡ (V_{v1}, $V_{v2,}$ V_{v3}) ≡ (3.8V, 3.8V, 0V)

So, depending on the initial values of MTJ_A and MTJ_B, the total amount of current generated by the branches containing MTJ_A and MTJ_B will be enough to set the value of MTJ_C to 1 (High resistance). If it is not, then, the value of MTJ_C will remain 0 (Low resistance) at which it was initially set.

As mentioned earlier, a cycle for the NAND circuit is decided by the waveform of the V_{sm} signal. Let's analyze this whole process by considering a sample cycle of V_{sm} starting from t(s) = 8n till t(s) = 12n in Fig. 4.1. Here are the values of different signals during this time duration and their interpretations.

For duration t(s) = 8n till t(s) = 10n :

V_{sm}= 1 (1.8V), V_{sl}= 0 (0V) →Write Mode

(V_{v1})= 0 (0V), (V_{g1}) = 1 (3.8V) → (Resistance of MTJ_A) = r(xmtja)= 5.9KΩ (MTJ_A is set to 1)

(V_{v2})= 0 (0V), (V_{g2}) = 0 (0V) → (Resistance of MTJ_B) = r(xmtjb)= 5.9KΩ (MTJ_B was already set to 1 in the previous cycle, we just kept the same value of MTJ_B. In the previous cycle, the value of the signals were (V_{v2}) = 0 (0V), (V_{g2}) = 1 (3.8V))

(V_{v3})= 1 (3.8V), (V_{g3}) = 0 (0V) → (Resistance of MTJ_C) = r(xmtjc)= 3.5KΩ (MTJ_C, which will hold the final result, is initialized to 0)

For duration t(s) = 10n till t(s) = 12n :

V_{sm}= 0 (0V), V_{sl}= 1 (1.8V) →Logical operation Mode

(V_{v1})= 3.8V → Supplying the bias voltage for MTJ_A to enable logic operation

(V_{v2})= 3.8V → Supplying the bias voltage for MTJ_B to enable logic operation

(V_{v3})= 0V → Applying ground to MTJ_C so that it can act as a sink to the current being

generated from the other 2 MTJs, thus completing the circuit.

NOTE: Voltage values at V_{g1}, V_{g2} or V_{g3} are irrelevant in this case as V_{sm}= 0 (0V), which means that these three signals are not connected to the circuit.

So, for this case, MTJ_A =1, MTJ_B =1

MTJ_C = NAND (MTJ_A, MTJ_B) = NAND (1, 1) = 0 which can be validated in Fig.4.1 from the resistance waveform of MTJ_C (r (xmtjc)) going down to 3.5KΩ representing a 0 (digital low) value.

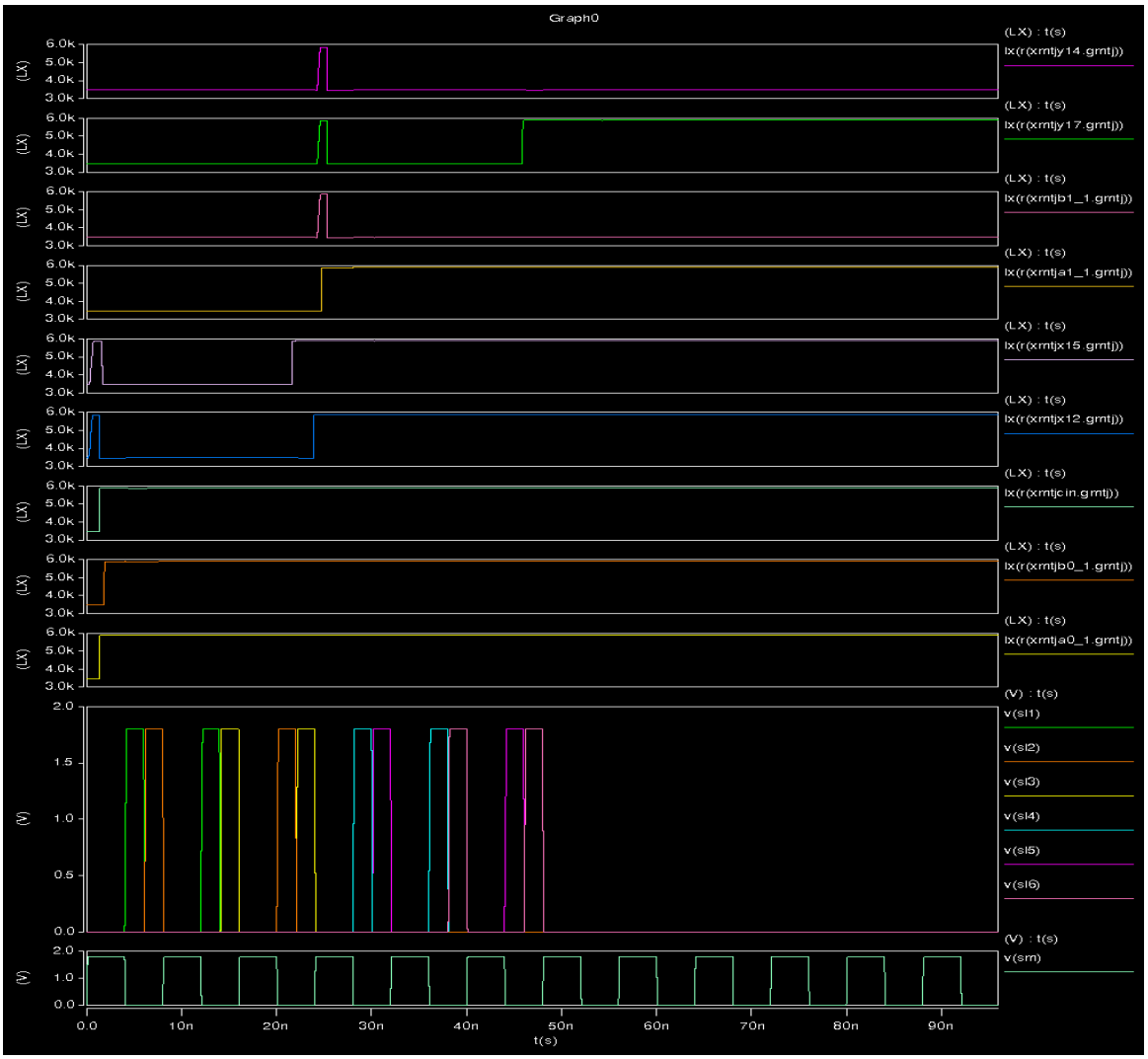

Fig. 4.2 1st and 2nd bit waveforms for 4-bit RCA

Fig. 4.2 and 4.3 depict waveforms for 4-bit RCA, 2 bits at a time. Fig. 4.2 shows 1st and 2nd bits waveforms while Fig. 4.3 shows 3rd and 4th bits waveforms.

In Fig. 4.2, V_{sm} cycle duration is 8ns as opposed to the V_{sm} cycle duration in Fig. 4.1 which is 4ns. The reason is that, here, we are trying to accommodate 2 NAND operations in a single V_{sm} cycle which is evident from the two logic-switch pulses on every V_{sm} cycle when V_{sm} = 1 (1.8V). As already explained in the RCA section of circuit design chapter, it takes 2 ¾ V_{sm} cycles to calculate Carry bit depicted by r (xmtjx15) waveform and 3 V_{sm} cycles to calculate Sum bit depicted by r (xmtjx12) waveform after we have set the initial values for a0_1, b0_1 and cin shown in r (xmtja0_1), r (xmtjb0_1) and r (xmtjcin) waveforms respectively (Fig. 4.2).

Initial values were set when V_{sm}=1 and the Sum and Carry bits get calculated when V_{sm}=0 and the logic-switch waveform is high. In Fig. 4.2, a0_1 = b0_1 = cin =1, as a result r (xmtjx12) (Sum waveform) and r (xmtjx15) (Carry waveform) finally go high. Next, a1_1 = 1, b1_1 = 0, x15 = 1 (carry waveform r (xmtjx15) from previous bit), this gives Sum (r (xmtjy14)) = 0 and Carry (r (xmtjy17)) = 1. Now, in Fig. 4.3, a2_1 = 1, b2_1 = 0, y17= 1 (carry waveform r (xmtjy17) from previous bit), so Sum (r (xmtjz14)) = 0 and Carry (r (xmtjz17)) = 1 and finally, as we have set a3_1 = 1, b3_1 = 1 and z17= 1 (carry waveform r (xmtjz17) from previous bit), Sum (r (xmtjw14)) = 1 and Carry (r (xmtjw17)) = 1. Calculations for each bit take 24ns and as next-bit calculation for RCA starts only when the previous calculation is done, the whole 4 bit operation takes **96ns** as shown in Fig. 4.3.

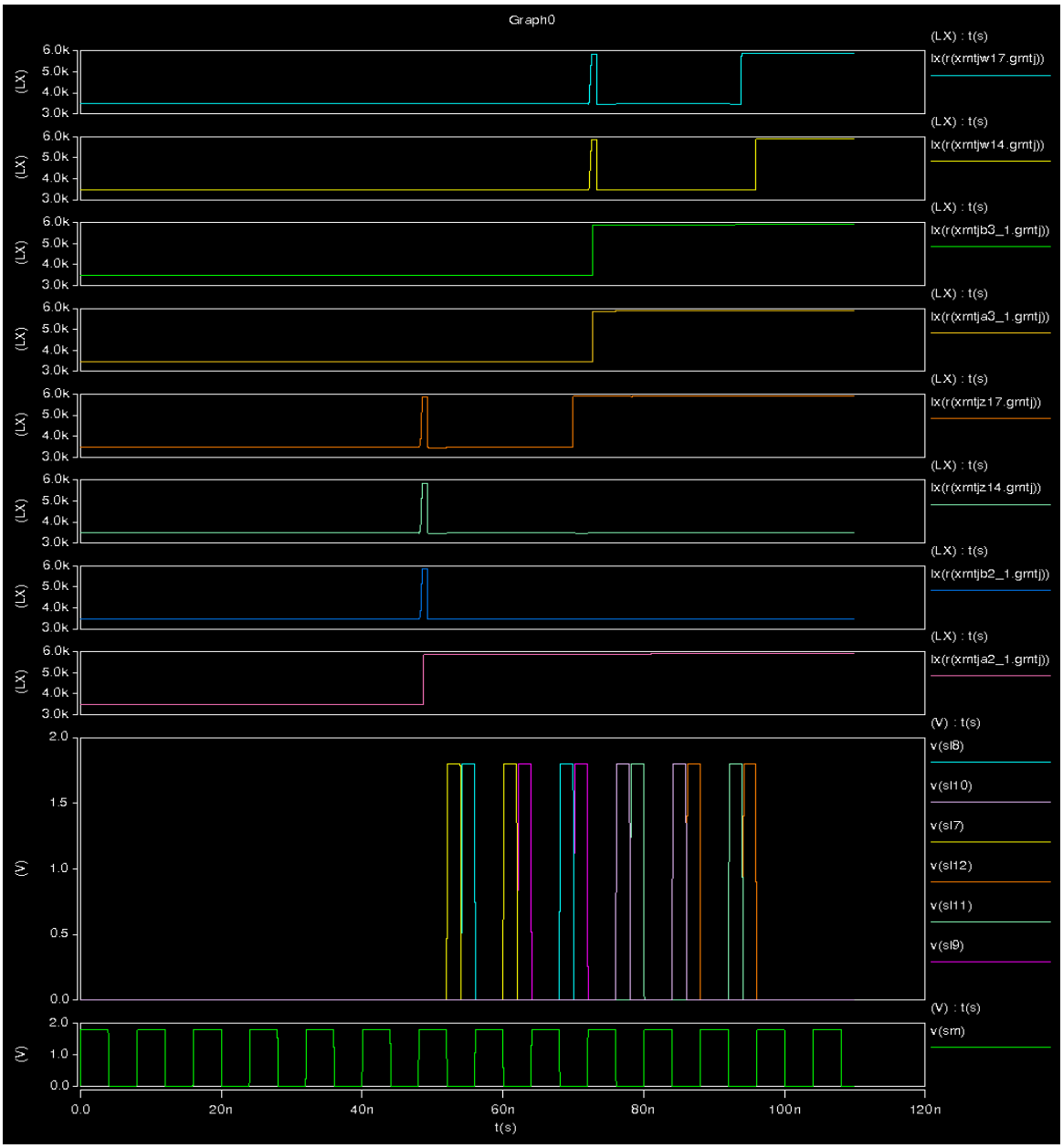

Fig. 4.3 3rd and 4th bit waveforms for 4-bit RCA

Fig 4.4 shows the switch and MTJ waveforms for all 4 bits of CLA. Here too, each V_{sm} waveform is of 8ns and accommodates 2 NAND operations in a single cycle. Initial values are set only when V_{sm} = 1 and we get results when V_{sm} = 0 and logic-switches are high. In Fig. 4.4, a0_1 = b0_1 = cin =1, as a result r (xmtjx9) (Sum waveform) and r

(xmtjx12) (Carry waveform) finally go high. Here, Carry gets calculated before the Sum due to the sequence in which we have turned ON the switches which prioritizes Carry calculation over Sum. Next, a1_1 = 0, b1_1 = 0, x12 = 1 (carry waveform r (xmtjx12) from previous bit), this gives Sum (r (xmtjy9)) = 1 and Carry (r (xmtjy24)) = 0. Now, a2_1 = 0, b2_1 = 1, y24= 0 (carry waveform r (xmtjy24) from previous bit), so Sum (r (xmtjz9)) = 1 and Carry (r (xmtjz37)) = 0 and finally, as we have set a3_1 = 0, b3_1 = 1 and z37= 0 (carry waveform r (xmtjz37) from previous bit), Sum (r (xmtjw9)) = 1 and Carry (r (xmtjw51)) = 0.

There is no fixed time duration for the calculation of Sum and Carry in CLA. Calculations for whole 4 bit operation take **54ns** (see Fig. 4.4). Here, calculation for each bit is being done in parallel, so, all the bits start calculating simultaneously and they arrive at the results at different time durations.

Carry calculation in each bit is completely independent but the next Carry bit always takes more time as compared to the previous Carry due to the increase in Carry logic with the increasing bit. The Sum calculation depends on the Carry value from the previous bit so even if we are done with almost all the calculations, we still have to wait for the previous Carry bit to get the final Sum value for that bit.

Note: In all the adder waveforms (Fig. 4.2, 4.3 and 4.4) that we have put here, we have just included the waveforms of the MTJs holding the input values and the bit-wise results and omitted all the waveforms of the intermediate MTJs for the sake of simplicity.

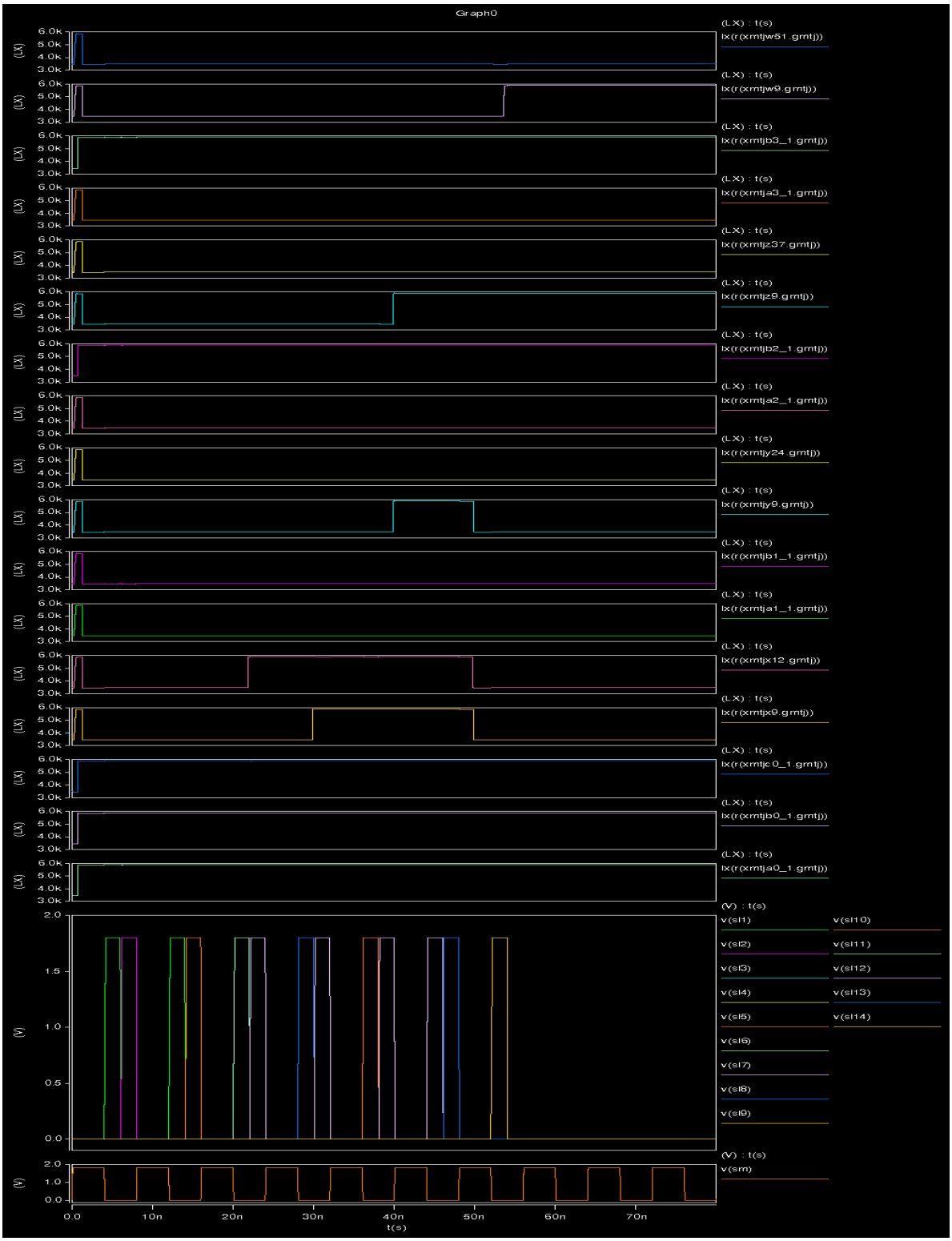

Fig. 4.4 Waveforms for 4-bit CLA

We also calculated power consumptions for 4-bit RCA and 4-bit CLA circuits. Each MTJ has two terminals and two different waveforms are applied to each MTJ. So, to calculate the power consumption of the whole circuit, we first calculated the average power consumption of the two waveforms for every MTJ that is there in the design.

HSPICE was used to first calculate average power consumption for each of these waveforms for the total time duration of 4-bit operation using the MEASURE command present in HSPICE. In the same way, we calculated the power consumption of all the switch (memory and logic) signals for the total time duration of the 4-bit operation. Next, we added power consumption values for all these waveforms to get the average power consumption value for the whole circuit. Average power consumption for the 4-bit RCA came out to be **39.98 mw** and average power consumption for the 4-bit CLA circuit came out to be **79.65 mw**.

Chapter 5

Analysis

As confirmed from the results and waveforms in the previous chapter, it is in fact possible to realize complex functions using the SLIC [3] approach. In this chapter, we first tried to analyze the designs of adders that we implemented by trying to compare 4-bit RCA and 4-bit CLA in their speed, power consumption and design and how these SLIC [3] based designs fit in an overall architectural scheme of things. We also analyzed how these SLIC designs fare as compared to their CMOS counterpart. It also includes discussion about some inherent shortcomings in the design mainly because of the functioning of MTJ.

If we consider a single bit in the RCA design, Carry and Sum circuitries do not share any MTJs except for the initial Carry MTJ and each of these have 15 and 17 MTJs respectively. So, a single bit in RCA design uses 31 MTJs {15+17-1(common carry-bit MTJ)}. 4 bit RCA design, therefore, uses **121** MTJs {(31*4) -3(Carry MTJs from previous bits)}.

CLA design consists of exclusive Carry circuitry, exclusive Sum circuitry and a small part which is common to both. The common circuit which is highlighted in black in Fig. 3.9 to 3.12 is basically the realization of the expression $A \oplus B$ for each bit which is used to realize Sum expression ($A \oplus B \oplus C$) as well as the expression to calculate Carry for each bit. As a result this common circuitry saved us 9 MTJs per bit in the CLA design as

compared to a single bit in RCA design. The thing that we realized later was that this could have been done in the RCA design as well which would have brought down the total count of MTJ in RCA to 85 MTJs {(22 (MTJs in a single bit) *4)-3 (Carry MTJs from previous bits)}. Total number of MTJs in CLA design comes out to be **220** with MTJ in each bit increasing progressively due to the increase in Carry circuitry (22+44+65+89). We can conclude that the CLA circuit is almost twice in size as compared to the RCA circuit and this is what we have expected.

Let's focus now on the speed of the circuits. The time it took to complete one 4-bit addition for the RCA design was **96ns** as compared to the 4-bit CLA addition which took **54ns**. So, we conclude that CLA design is approximately twice as faster as the RCA design.

The average power consumption is also along the same lines, where the CLA design because of being faster and larger in size is almost twice as compared to the RCA design. The average power consumption for CLA is **79.65** mw as compared to the RCA power consumption value of **39.98** mw. Following table summarizes all the results from both SLIC based adders for one 4 bit operation.

Table 5.1: SLIC based Adder Results

	MTJs Count	Time (ns)	Avg. Power (mW)
4-bit Ripple Carry SLIC based Adder	121	96	39.98
4-bit Carry Look-ahead SLIC based Adder	220	54	79.65

We know that in a Ripple Carry Adder, its size, delay and power consumption values increase proportionally to the number of bits in the design. In Carry Look-ahead Adder, the time varies logarithmically (log (n)) with the number of bits and size and average power consumption on the other hand, vary according to the equation n*log(n), if we consider 'n' to be the number of bits in the adder. Nagendra et al in [21] compared gate counts, performance and power consumption of different types of 16-bit, 32-bit and 64-bit Adders. These values from [21] for Ripple Carry Adder and Carry Look-ahead Adder along with the corresponding adder values for 1, 2, 3 and 4 bits for SLIC are tabulated in Table 5.2 and Table 5.3.

Table 5.2: Values for SLIC based RCA and CMOS based RCA

Number Of bits	SLIC Timings (ns)	CMOS Timings (ns)	SLIC MTJ count	CMOS Transistor count	SLIC Avg. Power consumption (mW)	CMOS Avg. Power consumption (mW)
1	24	-	31	-	25.24	-
2	48	-	61	-	30.21	-
3	72	-	91	-	35.21	-
4	96	-	121	-	39.98	-
16	-	28	-	596	-	0.4
32	-	56	-	1204	-	0.9
64	-	110	-	2420	-	1.8

We extrapolated the values that we got from [21] to get the delay, device count and average power consumption trends for CMOS adders and compared it to the trends that

we got from the results for SLIC based adders. These trends are shown in figures from 5.1 through 5.6.

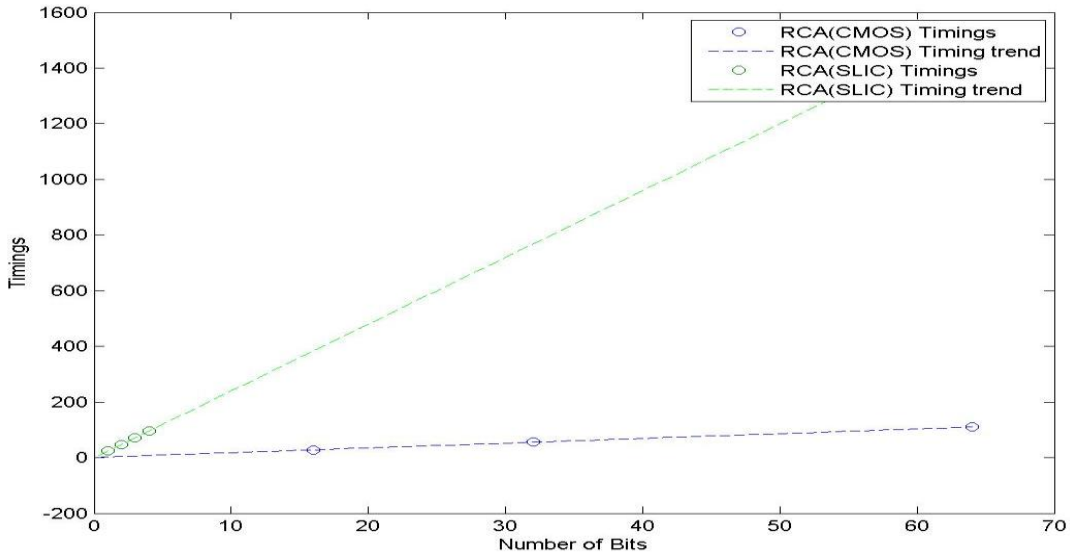

Fig. 5.1 Plot showing Delay trends for SLIC RCA and CMOS RCA

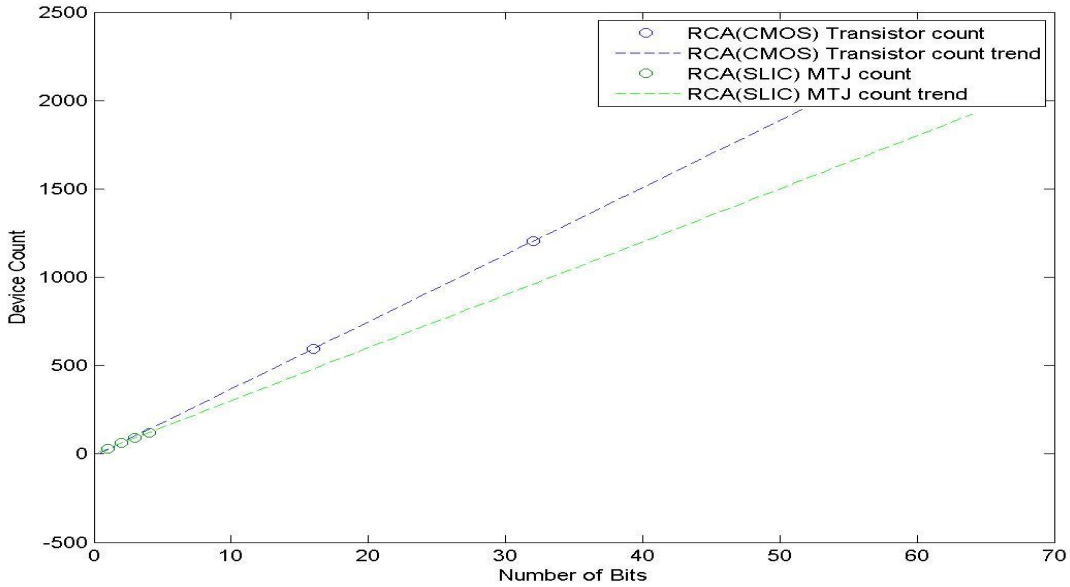

Fig. 5.2 Plot showing Device-count trends for SLIC RCA and CMOS RCA

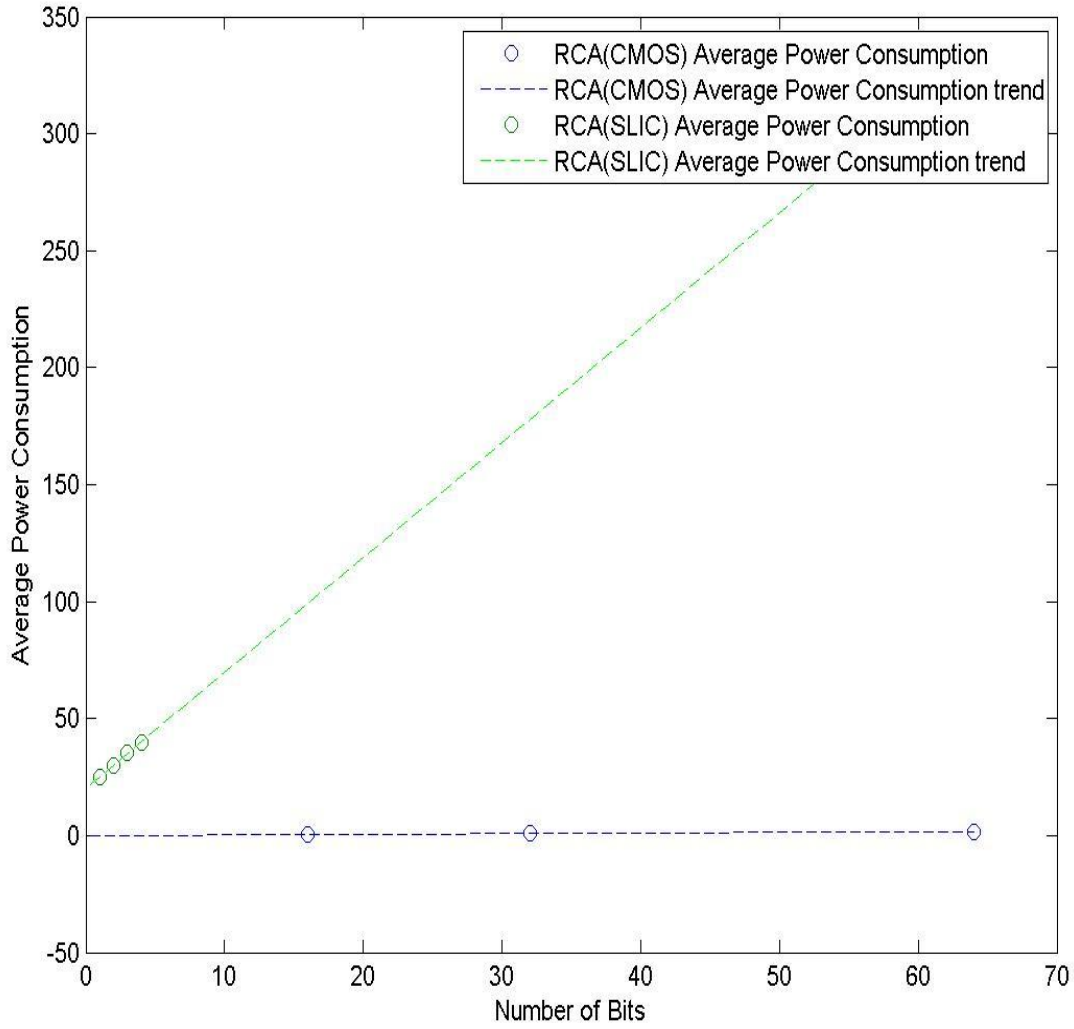

Fig. 5.3 Plot showing Avg. Power Consumption trends for SLIC RCA and CMOS RCA

Fig. 5.1, 5.2 and 5.3 show the plots depicting the comparison of SLIC trends against CMOS trends for Ripple Carry Adder. These graphs are based on the data shown in the Table 5.2.

Table 5.3: Values for SLIC based CLA and CMOS based CLA

Number Of bits	SLIC Timings (ns)	CMOS Timings (ns)	SLIC MTJ count	CMOS Transistor count	SLIC Avg. Power consumption (mW)	CMOS Avg. Power consumption (mW)
1	30	-	22	-	10.24	-
2	40	-	66	-	32.16	-
3	40	-	131	-	51.5	-
4	54	-	220	-	79.65	-
16	-	10	-	1038	-	0.6
32	-	15	-	2132	-	1.3
64	-	16	-	4348	-	2.6

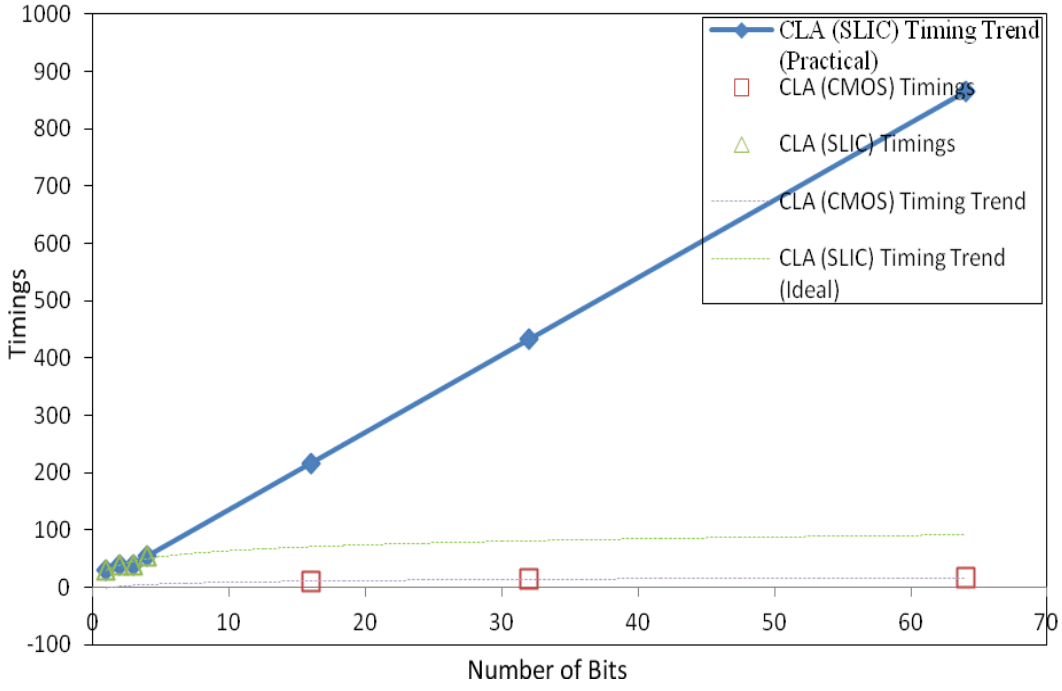

Fig. 5.4 Plot showing Delay trends for SLIC CLA and CMOS CLA

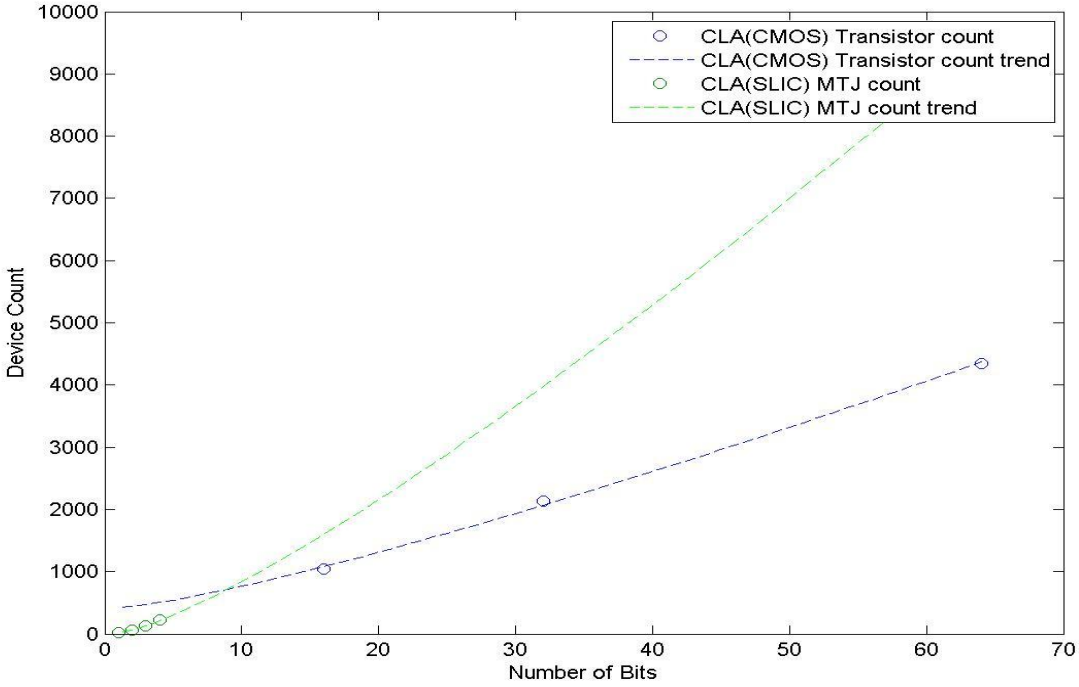

Fig. 5.5 Plot showing Device-count trends for SLIC CLA and CMOS CLA

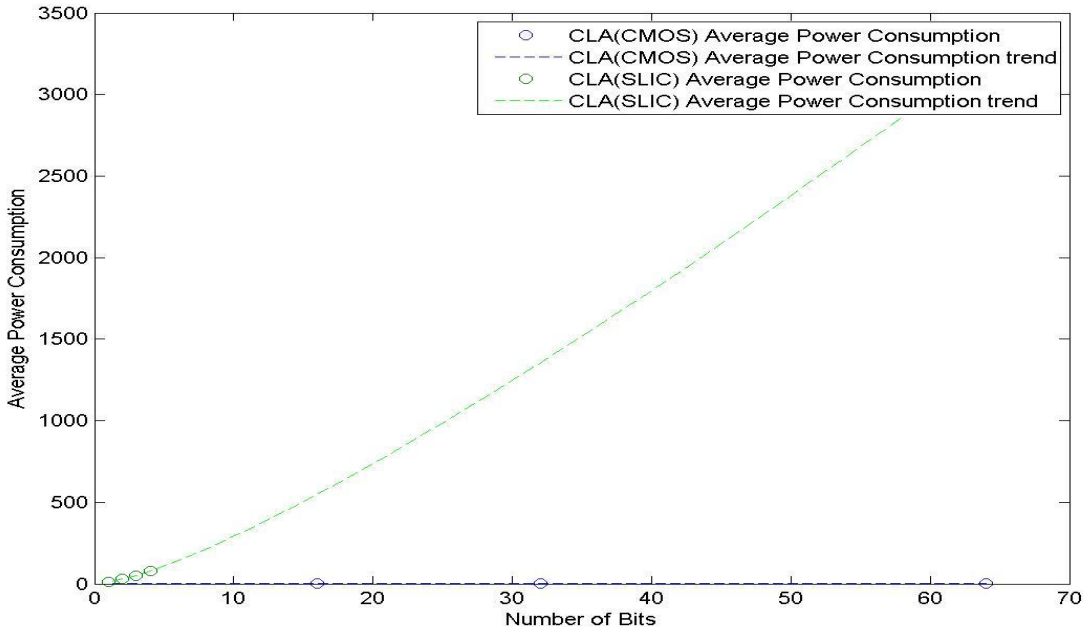

Fig. 5.6 Plot showing Avg. power Consumption trends for SLIC CLA and CMOS CLA

Fig. 5.4, 5.5 and 5.6 show the plots depicting the comparison of SLIC trends against CMOS trends for Carry Look-ahead Adder. These graphs are based on the data shown in the Table 5.3. While analyzing the Fig. 5.4, an important thing to note is that we know in theory, the delay trend in CLA is logarithmic in relation to the number of bits, but in practice, the fan out of CLA keeps on increasing as we increase the number of bits and that large fan-out might create problem in the effective functioning of the circuit. So, an additional curve is included in Fig. 5.4, where we assumed that we are using our 4 bit SLIC based CLA adder as a macro to build higher bit circuits. Using that reasoning, the delays for 16, 32 and 64 bit adders will be 216ns, 432ns and 864ns respectively and we plotted a line curve for these values to show the practical trend for this SLIC based CLA circuit.

These graphs clearly show that the results for CMOS adders are far better than SLIC based adders except for the device-count trend in Ripple Carry Adder. High power consumption values in SLIC based design can be ascribed to the high voltage biases that we need to operate the circuits (V_{bias}= 3.8V). The duty cycle for V_{sm} signal is kept at 50% where one half is used to write the values into MTJs while in the other half, we perform logic calculation. Each SLIC based NAND circuit takes at least 2ns for the value to be written into the MTJ and another 2 ns after that to process the inputs and generate the output in the resultant MTJ, once the logic switch is turned ON. So, due to the involvement of switches in the design which are basically there to make the circuit more controllable and because of the inherent delay in an MTJ to register a value in this type of design, the delay is bound to be higher. Large delay values, therefore, can be attributed to

the inherent design of the circuit.

In this thesis, although we haven't done much on the architectural aspect of the design, it could be quite an interesting exercise to compare how processor with Spintronics Logic in Cache (SLIC) [3] differs from the one with a normal cache. First, we fetch the instruction from the program counter which after being decoded turned out to be let's say, an addition operation. While performing an addition operation in a conventional processor, if the operands are present in the cache, they will be directly taken from it to the processor and if they are not, they will be taken from the main memory and cache will be updated with these values. Once we have the operands, the processor performs the addition operation and store the result back into the cache.

Performing the same operation using a processor with SLIC [3], we can save time as unlike in the previous case, here; we have the operands present in the cache. So, we can directly perform the addition operation in the cache itself and the result is again stored back in the cache. In the case where operands are not present in the cache, we need to hit the main memory to get them but here too, we are just getting the value in the cache from the main memory and it still saves us the time that we used in the above case to get the operands from the cache to the processor and also the time taken to write the final result from the processor to the cache. In addition to that, as the addition operation in this case has been delegated to SLIC, the main processor is free to perform any other complex operation. So, provided, we are able to bring down the time taken by an operation using SLIC approach to the levels that we have in CMOS these days, it has the potential to speed-up the overall processor and as explained above, it has a double advantage of

speeding up the operation that we are performing in the SLIC as well as providing us with the freedom to simultaneously utilize the main processor for some other complex operation.

Another thing that can be compared to this SLIC design is Processing In Memory (PIM) Technology. Conceptually, PIM is trying to achieve the same objective as SLIC which is to find a solution to the inherent latency present in the standard computer architecture (Von Neumann architecture). The processor and memory are separate and as data has to move between the two, there is bound to be some delay in accessing the data and in storing back the result. Moreover, the advancement in processor speeds as compared to the memory speeds is also a factor in limiting the transfer rates as the processor has to wait more and more to fetch the data. As a solution to this problem, PIM was proposed where although the logic (CMOS) and memory (DRAM) are still separate [23], they are integrated very close to each other so as to increase the bandwidth of data that can be transferred between the two and thus, also help in reducing the power consumption. Suh et al [22] in their paper 'A PIM-based Multiprocessor System' devised a system where PIM can be operated in three modes. ACTIVE MODE where both logic and memory are active. SLEEP MODE where only memory is active and it can used to read from and write into by an external device. STANDBY MODE where all components are disabled but the memory can still retain the data that it was holding. These PIM systems are very useful for applications that require a lot of parallel computations. SLIC design too is especially good at parallel computations as demonstrated by Patil [2] in her thesis by comparing its performance against a classic processor on 2 separate microbenchmarks

namely IMAGE NEGATION and AND MASKING. On the other hand, SLIC architecture unlike PIM is just one MTJ based circuit that acts as a memory device when the memory switch is turned ON and performs logic calculations when the logic switch is turned ON. So in SLIC, we have effectively replaced both CMOS logic and DRAM memory by a single MTJ based circuit that can work as memory or logic depending on the switch settings.

Chapter 6

Conclusion

SLIC as demonstrated by Patil [3] et al can be used to perform simple logic operations like NAND, NOR and other basic gate operations. The question that arises after this is if SLIC can also be used to perform complex logic operations that we are actually going to need if we want this design to be useful in our day to day logic calculations. This thesis is an attempt to answer that question. The main goal of this thesis is to demonstrate the ability of Spintronics Logic in Cache (SLIC) to perform complex logic operations. The operation that we chose to demonstrate is a 4-bit addition operation using full adders.

As evident from the explanation of simulated waveforms in Chapter 4 on Simulations and Results, it is in fact possible to use this design to perform addition of 4 bit operands. We first, attempted it using an ordinary Ripple Carry Adder and then, demonstrated the same operation using a faster Carry Look-ahead Adder. We saw an improvement in the performance of Carry Look-ahead Adder over Ripple Carry Adder but at the expense of power consumption and number of devices used. Though, we started with the objective to realize only a 4 bit adder, we also realized as we delved deeper into the design that the same method that we used to design adders can be used to design any digital circuit as demonstrated in Chapter 3 of Circuit Design.

We went on to compare the SLIC design against the CMOS design for the corresponding

adders on the parameters of delay, power consumption and device count and found out that except for slight advantage on the parameter of number of devices used, CMOS performs way better than SLIC as demonstrated in Chapter 5 on Analysis. The inherent switch-based design which uses high operational voltages of the order of 3.8V causes the overall design to consume large amounts of power and because of the use of NEMS switches in the design to separate memory and logic operations, it takes extra time for the MTJ to attain a stable value. This is because the switch pulse needs to be applied for longer duration (2ns at least) for the MTJ to reach the stable desired value in the cases where we are storing a value into an MTJ and when we are waiting for the result to appear in an MTJ due to some logic operation.

Based on the work that is done in this thesis, this SLIC architecture in the future can be used to realize other important functions like Multiplier, Divider etc. Other future work can also be on the lines of designing a complete ALU in cache where as we discussed before, small tasks can be delegated to SLIC, leaving the main processor free to perform other complex operations. This thing can have a great impact on the overall performance of the processor.

Most of the work in this thesis is focused towards the circuit design aspect of SLIC. Architectural aspect of SLIC design can also be explored by analyzing the advantage of this type of cache design over a processor with a normal cache. That kind of work can give valuable insight into the practical usage of Spintronics Logic in Cache.

REFERENCES

[1] "Overall Technology Roadmap Characteristics," International Technology Roadmap for Semiconductors, 2010

[2] Shruti Patil, "Development of Next Generation Computing Elements Fabricated with Emerging Technologies," Ph.D. thesis, Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, 2011

[3] Shruti Patil and David J. Lilja, "Performing bitwise logic operations in cache using spintronics-based magnetic tunnel junctions." In Proceedings of the 8th ACM International Conference on Computing Frontiers, CF '11, pages 33:1-33:9, 2011.

[4] S. Patil, A. Lyle, J. Harms, D. Lilja, and J.P. Wang, "Spintronic logic gates for spintronic data using magnetic tunnel junctions," IEEE International Conference on Computer Design (ICCD 2010), pages 125-131, 2010.

[5] S. S. P. Parkin, Z. G. Li, and D. J. Smith,''Giant Magnetoresistance in Antiferromagnetic Co/Cu Multilayers,'' Appl. Phys. Lett. 58, 2710 (1991);

[6] B. Dieny, V. S. Speriosu, S. Metin, S. S. P. Parkin, B. A. Gurney, P. Baumgart, and D. FL Wilhoit, "Magnetotransport properties of magnetically soft spin-valve structures," Journal of Applied Physics 69, 4774 (1991)

[7] "Spintronics," IBM J. Res. Develop., vol. 50, no. 1, 2006

[8] M. Julliere, Physics Letters, vol. 54A, No.3, pp.225-226, 1975

[9] T. Miyazaki and N. Tezuka,''Giant Magnetic Tunneling Effect in Fe/Al2O3/Fe Junction,'' J. Magn. Magn. Mater. 139, L21 (1995).

[10] J. S. Moodera, L. R. Kinder, T. M. Wong, and R. Meservey,''Large Magnetoresistance at Room Temperature in Ferromagnetic Thin Film Tunnel Junctions,'' Phys. Rev. Lett.74, 3273 (1995)

[11] J.M. Slaughter , E.Y. Chen, R. Whig, B.N. Engel, J. Janesky, and S. Tehrani, Magnetic Tunnel Junction Materials for Electronic Applications, JOM-e, 52 (6) (2000)

[12] Shruti Patil and David J. Lilja, "Performing bitwise logic operations in cache using spintronics-based magnetic tunnel junctions," In Proceedings of the 8th ACM International Conference on Computing Frontiers, CF '11, pages 33:1-33:9, 2011.

[13] Hao Meng, Jianguo Wang, and Jian-Ping Wang, "A Spintronics Full Adder For Magnetic CPU," IEEE ELECTRON DEVICE LETTERS, VOL. 26, NO. 6, JUNE 2005

[14] Seungyeon Lee, Sunae Seo, Seungjun Lee and Hyungsoon Shin, "A Full Adder Design Using Serially Connected Single-Layer Magnetic Tunnel Junction Elements," IEEE TRANSACTIONS ON ELECTRON DEVICES, VOL. 55, NO. 3, MARCH 2008

[15] Hiwa Mahmoudi, Thomas Windbacher, Viktor Sverdlov, and Siegfried Selberherr, "Design and Applications of Magnetic Tunnel Junction Based Logic Circuits," 978-1-4673-4581-1/13, IEEE, 2013.

[16] Dhruva Kumari, Monisha SaW, Aminul Islam, "Design of 2: 1 Multiplexer and 1:2 Demultiplexer Using Magnetic Tunnel Junction Elements," 978-1-4673-5301-4/13, IEEE,2013

[17] Seungyeon Lee, Nakmyeong Kim, Heejung Yang, Gamyoung Lee, Seungjun Lee, and Hyungsoon Shin, "The 3-Bit Gray Counter Based on Magnetic-Tunnel-Junction Elements," IEEE TRANSACTIONS ON MAGNETICS, VOL. 43, NO. 6, JUNE 2007

[18] W. Zhao, E. Belhaire, and C. Chappert,"Spin-MTJ based non-volatile flip-flop," 7th IEEE Conference on Nanotechnology (IEEE-NANO 2007), pages 399-402, 2007.

[19] J. Harms, F. Ebrahimi, X. Yao, and J.-P. Wang,"Spice macromodel of spin-torque-transfer operated magnetic tunnel junctions," IEEE Transactions on Electronic Devices, 2010

[20] Z. Diao, A. Panchula, Y. Ding, M. Pakala, S. Wang, Z. Li, D. Apalkov, H. Nagai, A. Driskill-Smith, L-C.Wang, E. Chen, and Y. Huai," Spin transfer switching in dual mgo magnetic tunnel junctions," Applied Physics Letters, 90(13):132508, 2007.

[21] Chetana Nagendra, Mary Jane Irwin and Robert Michael Owens, "Area-Time-Power Tradeoffs in Parallel Adders", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-11: ANALOG AND DIGITAL SIGNAL PROCESSING, VOL. 43, NO. 10, OCTOBER 1996

[22] Jinwoo Suh, Changping Li, Stephen P. Crago, and Robert Parker "A PIM-based Multiprocessor System", 0-7695-0990-8/01, IEEE, 2001

[23] Peter M. Kogge, Toshio Sunaga, Hisatada Miyataka, Koji Kitamura "Combined DRAM and Logic Chip for Massively Parallel Systems", 0-8186-7047-9/95, IEEE, 1995