NEW PCM BASED FPGA ARCHITECTURE AND GRAPHENE MEMORY CELL DESIGN

BY

CHUNAN WEI

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Adviser:

Associate Professor Deming Chen

Abstract

In this work, we introduce a new look-up table (LUT) implementation using phase change memory (PCM) cells. Utilizing the fact that PCM cells store data using resistance values, unlike traditional memory cells, the memory cell we propose can store up to 2 bits per cell. This is achieved by controlling the 1T2R PCM cell resistances between the on resistance and the off resistance. Taking advantage of this property, a new LUT is designed that can either store two functions or a single function with an additional input. Thus the new design greatly improves FPGA logic density and reduces the area and timing cost. In order to access this new type of memory cell, a sense circuit is designed. We also propose a new hybrid architecture that uses LUTs with single and independent twin-outputs. The architecture is evaluated over delay and area with different LUT configurations. Experimental results show that on average 50 percent area reduction and 10 percent delay reduction can be achieved with the use of the new PCM based LUTs over SRAM based designs. Also, further savings can be achieved by incorporating hybrid architecture.

Also, we introduce a new device design based on the SYMFET. This new type of device can be implemented to a memory cell utilizing the graphene sheet-to-sheet tunneling effect to store data. The SYMFET sheet-to-sheet tunneling can generate high current density when the Dirac points of graphene sheets are aligned; low current density is generated when Dirac points are not aligned. This new SYMFET memory has a GIGIG (graphene-insulator-graphene-insulator-graphene) structure. With this new type of device structure, a new mechanism for controlling data stored in the memory cell is introduced. Based on self-limiting tunneling, a third piece of graphene sheet is inserted in-between the source and drain graphene sheets serving as the tunneling current control mechanism. We also developed methods to calculate the on/off currents and gate potentials for this device.

To Mom and Dad.
To my friends.

Acknowledgments

This thesis is made possible by the support from many people. My special thanks to Ashutosh Dhar, who helped me on the PCM-based FPGA design. My thanks to Christine Chen, who helped me on the graphene transistor research.

Also my greatest regard to Professor Deming Chen, who advised my research from my senior year and helped me all these years.

Finally, my thanks to my parents and my friends, for without your support, I cannot walk this far.

Table of Contents

Chapter 1: New PCM Based FPGA Architecture

1.1 INTRODUCTION

Field-programmable gate arrays (FPGAs) represent a promising computing platform for parallel and low power applications [1]. More importantly, compared to ASIC chips, FPGAs are programmable and can be reconfigured for different applications. With improved technology scaling resulting in shrinking die sizes, FPGAs are getting smaller and their logic density is growing. However, due to characteristics of silicon MOSFETs and their scaling limit, to further improve the density of SRAM based FPGAs is getting harder. Thus, a lot of effort has been invested towards finding a substitute for SRAM based memory cells that are the key components to build the look-up table (LUT) within the configurable logic blocks of the FPGA. As the work [2] shows, emerging non-volatile memories (NVM) are gaining acceptance as future memory. Recent works such as introducing nano-electromechanical (NEM) switches in FPGA architecture [3], [4], R-RAM FPGA [5] which utilize resistive RAM as replacement for SRAM, PCM based FPGA [6] in which SRAM is substituted by PCM cells and [7] which develop a 3D PCM FPGA architecture suggest that the leading trend for FPGA development is to also introduce emerging non-volatile memories (NVM) into the FPGA architecture.

Emerging NVMs have gained a significant amount of interest in recent years, with several types of NVMs being proposed and developed. Among them, PCM is considered to be one of the most promising candidates for future storage elements. This is because PCM has excellent scalability [8] with a single cell being potentially scaled to a few nanometers in size. It also has a moderate amount of read/write cycles.

Incorporating PCM into FPGA designs has been attempted previously in a rather simplistic fashion. By simply replacing SRAM cells with 1T2R PCM cells [6], the FPGA architecture developed in [6] can achieve an average of 40 percent delay reduction and 13 percent area. However, most of the work focused on replacing SRAM cells with the PCM cells directly without exploring new architecture capabilities.

In this work, we propose a new FPGA architecture with a different strategy. Utilizing the 1T2R memory cell design, we expand the single cell memory capacity to 2-bit by controlling PCM set/reset process to achieve different resistance levels between PCM On resistance and Off resistance values. Based on this feature of PCM cells, we develop a new LUT architecture with the capability to be configured to achieve better logic density. We show that this type of new architecture not only saves a significant amount of area, but also delivers improved timing performance when compared to traditional SRAM based FPGA.

We summarize our contributions as follows:

- We develop a novel use of PCM cells to store 2 bits per cell for the LUT design in FPGA.
- We design the LUT to support either two functions or a single function through simple configuration that improves the flexibility of using the LUT.

The rest of the chapter is organized as follows. The architecture and design are discussed in Section 1.2, following which we outline our experimental setup and methodology in Section 1.3. Finally, we discuss our results in Section 1.4 and present our conclusions in Section 1.5.

1.2 Architecture and Design

In this section we discuss the architecture and design of the new proposed LUT. Figure 1 illustrates the new LUT architecture. The salient feature of the new LUT is the use of a PCM cell in a novel fashion such that a single PCM cell can store 2 bits of configuration data. Figure 1 traces the design for a single PCM cell associated with the LUT. As is the case with SRAM based LUTs, each decoder line is associated with a single memory cell. The ability to store 2 bits per PCM cell allows us to use the LUT to either store two functions per LUT by using both outputs or use an N-1 input LUT to implement an N-input LUT by introducing one more input signal to make a selection between the two outputs.

We now discuss the architecture and design of components within the LUT: the memory cell and the sense amplifier circuit. We also discuss the switch box and connection box designs along with the architecture of the FPGA we use to evaluate the new LUT.

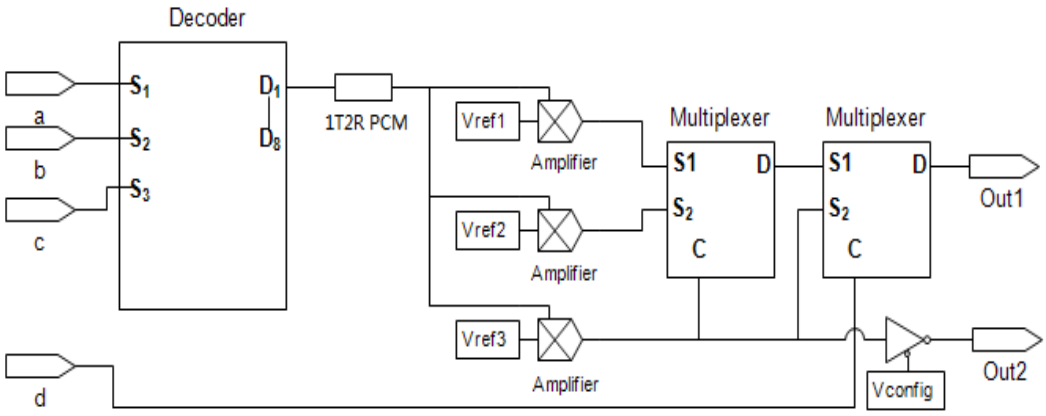

Figure 1. Design of the new LUT architecture. Based on input logic drives, we use an N-1 LUT to implement an N-LUT or to store two (N-1)-input functions. The inputs a, b, c are used for address decoding while the input d will only be used for one-output case. Vconfig will determine whether this LUT has a single output or two outputs.

1.2.1 Memory cell design

The memory cell model for the new PCM cell uses the 1T2R model as shown in Figure 2.

We select this design for two reasons. Firstly, the 1T2R model enables us to control the relative resistance values for the 2 resistors, which are actually PCM cells. By controlling these resistance values, we can set the output voltage to be any value between VDD and GND. Secondly, 1T2R cells have small leakage currents as the sum of the resistances offered by the two cells are high for any type of cell programming. To program the memory cell, the supplies, VDD and GND, are attached to different current sources and the transistor (1T) is turned on to allow current to pass through the PCM cells.

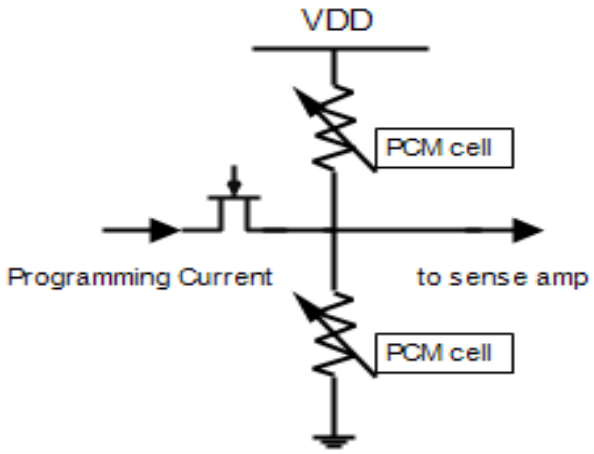

Figure 2. Transistor level 1T2R PCM memory cell.

Traditional single-bit PCM cells, which store only one bit per cell, have their output swing from 0V to VDD. So, we program the two PCM cells to a "one-on one-off" state. For example, to store logic 1, we can program the top memory cell in Figure 2 to be in an on state and the bottom memory cell to be in the off state. This way, not only have we created a low resistance between the output and the source, but also a high resistance across the memory cells between VDD and GND has been achieved. This design has fast read speed and low leakage power [6].

For our design, we aim to store two bits in a single memory cell. Using the same memory cell design, we set the output voltages of the cell to be 0, VDD/3, 2VDD/3 and VDD representing the bit vectors 00, 01, 10 and 11 respectively. This is done by controlling the programming current density or programming time. In order to reduce leakage power, the sum of the resistances from the two cells must be high. Thus, on an average, the read speed has been sacrificed since the resistances between output and the source are expected to be higher in the case of 01 and 10 as compared to 00 and 11. For leakage power concern, an extra transistor will be added in series with two PCM cells to limit the leakage current.

1.2.2 Sense amplifier circuit design

Since the output of our memory cell ranges between VDD and GND, we need a sense amplifier circuit to determine the actual outputs from the memory cell. We have employed a sense amplifier circuit design based on three latched voltage comparators (Figure 3).

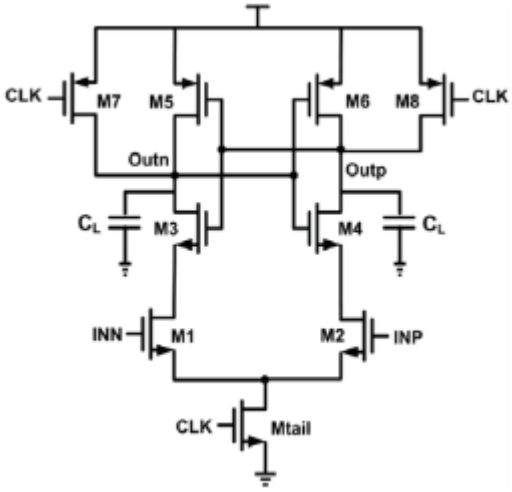

Figure 3. Latched voltage comparator compares INN with INP. The output is based on whether INN or INP has higher voltage [10].

We use three voltage comparators and provide three different reference voltages: VDD/2 (vref3), VDD/4 (vref1) and 3VDD/4 (vref2) as shown in Figure 1. Based on the relation between output voltage and half-VDD, we can directly determine the most significant bit of our output. For example, if the output voltage is greater than half VDD, then the most significant bit of the output can only be 1 since the data stored is either 10 or 11. And based on the most significant output, we can choose which output to use from the other two sense amplifiers. This results in a small area overhead when compared to a traditional SRAM-based LUT. However, since every LUT memory cell can share one sense amplifier circuit, the area overhead is diminished.

An outstanding feature of our LUT design is its ability to either store two functions, by providing two independent outputs, or to store a single function and behave like a normal LUT with a single output but with an additional bit available for decoding, i.e. a larger LUT size.

When operating in the twin-output mode, the first function is stored using the more significant bit while the second function is stored the using less significant bit. While operating in the single-output mode, the LUT utilizes an additional input signal to select either to use the more significant bit or the less significant bit.

A slight routing area overhead will be introduced when operating in the twin-output mode since we need to consider the connection from the additional output to the channels.

1.2.3 Connection box and switch box

For the connection box, the CLB input ports are connected to routing channels through MUXs. The mux size is determined by the number of channels a single LUT's input port is connected to. The mux we use is a pass transistor based mux. While tri-state buffers are used between CLB outputs and routing channels. The number of routing channels one CLB output can be connected to is 10 percent of total routing channels.

For the switch box, uni-directional switching is used with subset switches. So, for one specific channel, the signal can only go one direction and cannot communicate with wires on other channels.

1.2.4 CLB and LUT sizes

For our study, we focused on configurable logic block (CLB) of size 10; i.e. a single CLB contains 10 LUTs. We evaluated multiple LUT sizes including 4, 5, 6, 7, 8 and 10.

1.3 EVALUATION METHODOLOGY

In this section we present the approach we adopt. A flow-diagram of the approach is provided via Figure 4.

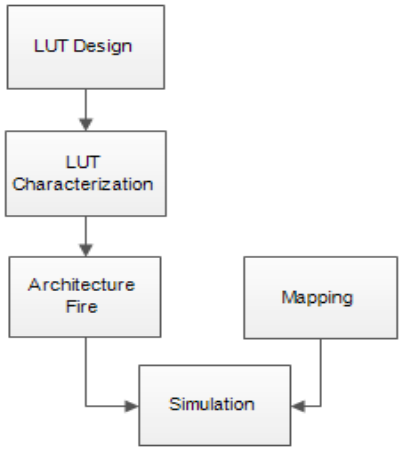

Figure 4. The research approach for the new architecture performance evaluation.

The LUT design stage focuses on the design of the memory cells and sense amplifier circuit. This step is done with the help of HSPICE. SPICE models of the LUT are used to determine whether or not the LUT functions as desired. The following LUT characterization step is also done with the help of HSPICE. Upon verifying the functionality of the model, we collected data concerning the LUT delay and estimated area. These will be used in later stages of the flow. After obtaining the results from mapping and LUT simulations, we used the VPR and T-PACK tool chain [11] to evaluate the performance of the design.

1.4 Experimental Results and Discussion

In this section, we present our evaluation of the newly designed PCM based FPGA. Since the purpose of our study is to develop a new FPGA architecture capable of better performance, we evaluate its performance based on area and timing. First, the LUT performance will be evaluated. Then, the single output LUT based FPGA architecture performance will be evaluated.

A. Single LUT evaluation

1) Area:

The PCM based LUT and SRAM based LUTs are implemented by Hspice and their areas are evaluated by transistor count. Since PCM cells are capable of storing two bits per cell and the fact that smaller decoder is needed, even with the overhead of sense amplifiers, we still achieve area reduction around 50 percent. At LUT size 4, the area ratio between PCM LUT and SRAM LUT is 54.88%. As the LUT size increases, the area ratio drops to 48.6% for size 8. This is because the decoder size has a quadratic growth and starts to dominate the LUT area while the area of sense amplifiers stays constant.

2) Delay (Timing):

Since the PCM cells have relatively larger read resistance than SRAM cells, the worst case scenario for PCM cell read delay is worse than the SRAM cell. We evaluated the delay value by Hspice simulation. The PCM cell delay value is represented by the worst case scenario that an 11 read is followed by a 01 read. From our experiment, the PCM cell has the worst case delay of 112ps while SRAM cell has 80ps delay. Considering other LUT components, sense amplifiers have a relative constant delay of 21.2ps while decoder has a quadratic delay growth. So, for smaller LUT sizes, SRAM based LUTs have almost half of the delay of PCM LUTs. For example, for LUT size 5, delay for PCM LUT is 334.9ps and 182.4ps for SRAM LUT. However, as the LUT size grows, the decoder delay starts to dominate the total LUT delay.

3) Leakage Power

According to our simulation, if 1T2R cell design is used, then the resistive loads between VDD and GND are the two PCM cells. This means the resistance is around 1 MΩ. This will introduce leakage current with amplitude around 1uA. This amount of leakage current is not acceptable. So, when the cell is not accessed, we need to reduce the leakage current dramatically. This leads to a 2T2R design. The extra transistor is added in series with

two PCM cells. The extra transistor is only turned on when the memory cell is accessed. So, the leakage current is dominated by the extra transistor.

From our experiment, the leakage current of a single NMOS transistor is 48.2nA. Since the PCM memory cells only have one leakage path compared to two leakage paths in SRAM, we expect less leakage power from the PCM based architecture. Also, the number of memory cells needed is halved, so a further leakage reduction can be expected.

We build our test SRAM and PCM by Hspice. All the cells are designed with minimal width nmos and pmos for the convenience. The SRAM cell has a standard 6T design with access delay of 82 ps. We know that SRAM cells contain two inverters, which leads to two leakage paths, while PCM cell only has one leakage path. However, the sum of the SRAM leakage current is around 92.005 nA and this is more than two times of PCM leakage current (31.119nA). This is because the sizing of the SRAM cell requires one inverter to be larger than the other in order to flip logics. So, we can see that for a single memory cell, SRAM will consume 3x more leakage current, and thus leakage power, than PCM cell. The total leakage power can be calculated by multiplying the total number of memory cell used with the leakage power of a single memory cell.

B. Single output LUT FPGA evaluation

For the purpose of evaluation, we used the MCNC benchmarks and tested the architecture with LUT sizes of 4, 5, 6, 7, 8 and 10. We passed each benchmark through identical mappers and place-and-route tools for PCM-based LUTs and SRAM-based LUTs.

1) Area:

Figure 5 presents the relative area of PCM based design with respect to the SRAM counterparts. For example, for LUT size 4, we see that PCM-based design occupies only 55 percent area with respect to SRAM

based architecture if implemented in our new architecture. As the LUT size increases, the area ratio reaches the best value at LUT size of 8 (48 percent). After size 8, the area ratio between PCM based design and SRAM based design returns to around 50 percent.

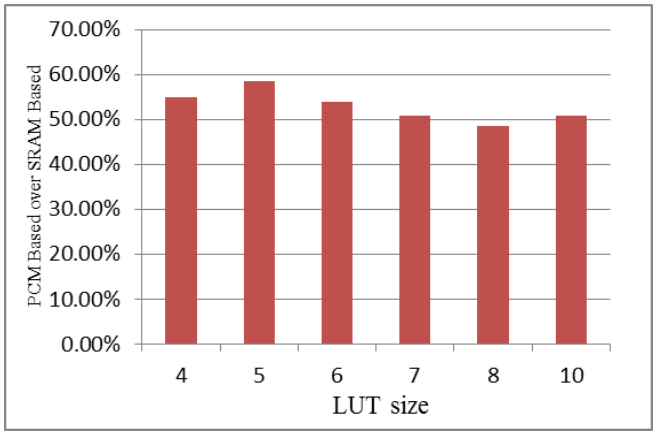

Figure 5. Relative area occupied by SRAM-based designs with respect to PCM-based design. It is clear that the area saving diminishes in percentage as the size increases. Thus size 8 has the best saving.

The dramatic area advantage is the combination of three factors. Firstly, since our design has the capability of storing two bits per cell, we require only half the number of memory cells as compared to traditional SRAM based FPGA. Secondly, the decoder area dominates the area of the LUT as the LUT size increases. Since our design uses half of the memory cells needed by SRAM based FPGA, the decoder needed for our design is one size smaller than the decoder needed for SRAM based FPGA. For example, to implement a size 4 LUT, only a 3 to 8 decoder is needed for our design. And the third factor, with smaller LUT areas, the routing channels can be made shorter, which counts toward area saving as well.

2) Timing

We now proceed to discuss the timing performance of the new architecture. Figure 6 presents the average relative critical delay of an SRAM-based FPGA design with respect to our PCM-based design.

From Figure 6, we notice that the PCM-based design offered about 10 percent performance advantage over SRAM with respect to delay from the small LUT sizes such as size 4 and 5. However, as the LUT size increases to 6, 7 and 8, the performance of SRAM-based cells starts to deteriorate with respect to PCM-based cells. At last, when the LUT size increased to 10, there is no performance advantage from either PCM based architecture or SRAM based.

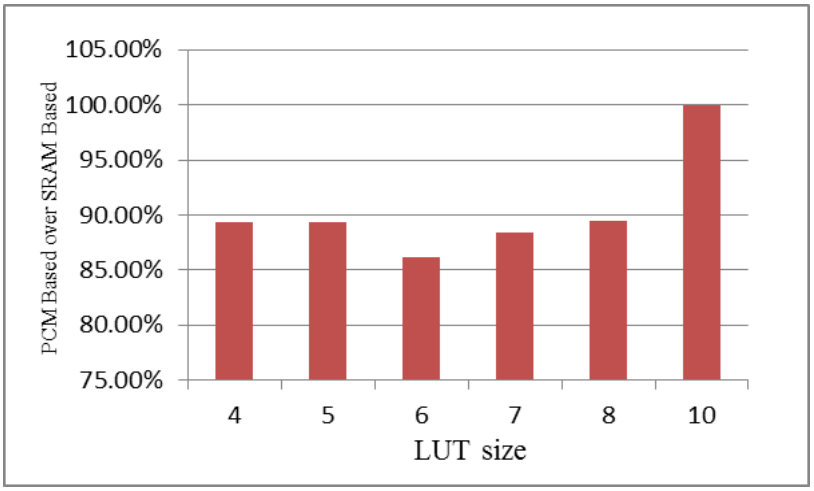

Figure 6. Relative critical path delays of SRAM-based designs with respect to PCM-based design. Again, the maximum relative delay reduction is reached for LUT size 8.

To understand the nature of this result, we should note that the critical path delay is comprised of 2 components: the delay through the LUT (logic delay) and the routing delay. Based on our architecture, the delay of the LUT can be expressed as:

$$T_{LUT} = T_{decoder} + T_{Bit\ line} + T_{sense\ circuit} + T_{mux}$$

After completing the characterization of each LUT, we found that for a 4-input LUT, the logic delay of PCM LUT is approximately two times the logic delay from SRAM LUT in the worst case scenario. This is because of

the high read resistance from the LUT cell. So, in order to have better timing performance, we expected the PCM FPGA to have better routing delay than SRAM FPGA due to its area advantage.

From the relative critical path comparison shown in Figure 6, we can see that starting from 4-LUT, PCM FPGA has the advantage over SRAM FPGA of about 10 percent average critical path delay saving. When the LUT size increases to 5, there is only a tiny difference over PCM FPGA. This is because the actual tile size did not grow that much, indicating the saving from routing and the overhead from logic canceled out. Starting from 6-LUT, where the area growth starts to accelerate, we can see increasing critical path delay saving from 6-LUT, which means routing delay is dominating. Starting from LUT size 7, since the MCNC benchmarks are relatively small, logic delay starts to dominate once again because the number of levels of logic is decreasing. At LUT size 10, we can see that the average delay performance is the same for PCM-based and SRAM-based even they have different logic delay and routing delay.

At this point, we proceed to make a conclusion concerning the best LUT size for this new FPGA architecture (single output LUT only). Based on previous work [12], LUT size 6-8 has shown the best overall timing performance while demonstrating relatively high area cost. In our experiment, the best area reduction comes from size 8 while the best delay reduction comes from size 6. In Figure 7 we consider both area and delay via area-delay percentage product.

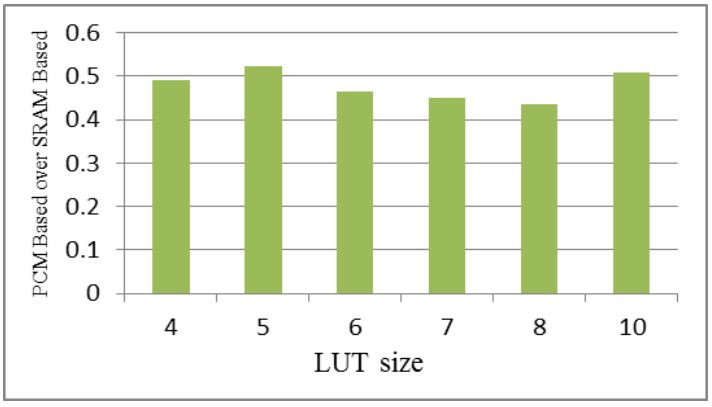

Figure 7. The area delay percentage product.

From Figure 7 we see that size 8 now has the best overall area and delay saving relative to SRAM based FPGA if we give equal weight to area and timing. So we conclude that as compared to SRAM based FPGA [12], the best area-delay performance values can be obtained for LUT sizes of 6 to 8 for this new architecture.

1.5 PROCESS VARIATION

The PCM cells suffer from process variation and also resistance drift [13]. This means that the resistance of PCM cells after a write operation may not be exactly the desired resistance value. For a large PCM memory system, the program-and-verify (P&V) technique is deployed. After every write operation, a verification of the resistance value is launched. If any memory cell failed to be written into a particular resistance range, then another write operation will be issued [14]. This type of technique is also popular for multi-level NAND flash memory. We can also use this technique to overcome resistance drift. The disadvantage of P&V is that it introduces area overhead to the circuit. A better way to solve these issues is to design the memory cell to tolerate the errors. Generally, the different resistance values of a PCM cell are modeled by log scale Gaussian distribution with standard deviation equal to $\frac{1}{6}$ [15]. Here, we notice the difference of assigning resistance values to logic levels. In [15], the resistance levels are assigned with the same ratio of 10. For example, logic level 00 has resistance of 1000Ω, logic level 01 has 10000Ω and so on. In our case, resistance levels are assigned with the same difference. So by applying this model to our design, the resistance distribution can be found in Figure 8.

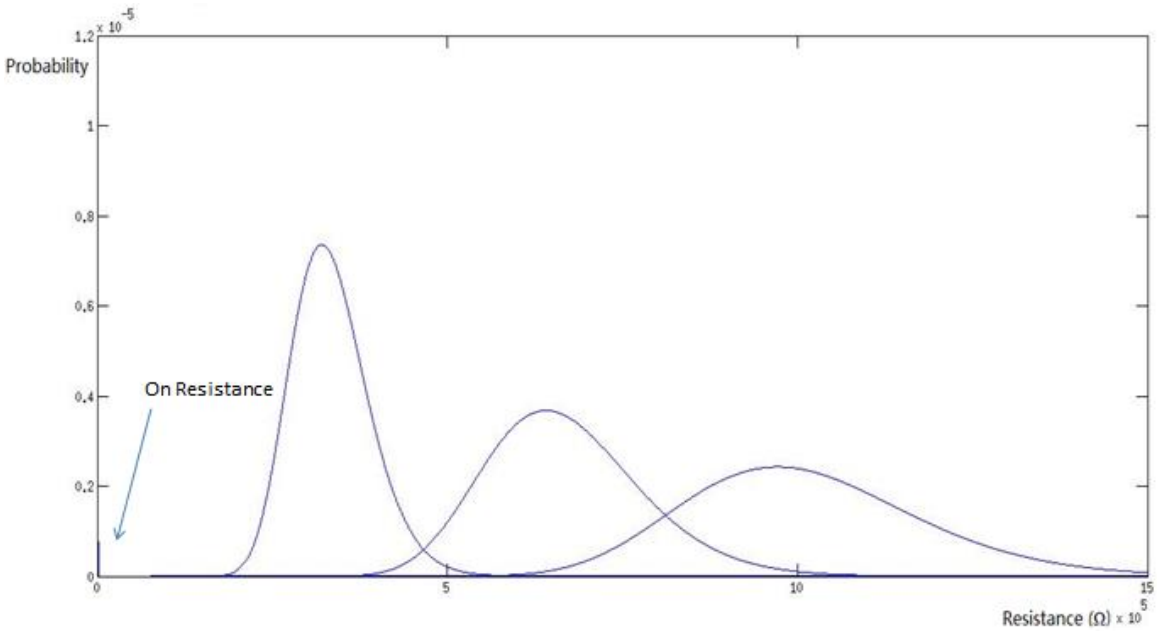

Figure 8. The resistance distribution for 4-level PCM cell.

As we can see, there is a relatively large area overlap for 10 and 11 resistances. But since our cell uses the ratio of two resistances to determine the output, this will not destroy data stored inside the memory cell. In the worst case scenario, the highest two resistances will have considerable overlapping area. However, even with just three distinguishable resistance levels, our memory cell still works. This will be illustrated in the next section with the design of 3-level cell.

According to [15], 3-level PCM cells have better read-out ratio and are more stable than 4-level PCM cells. But how can we use 3-level PCM cells to represent four logic levels? By our design, in order to reduce the read current, two programmable resistors are used for a single memory cell. When storing 11, the top resistor, which is connected to Vdd, will have low resistance and the bottom resistor, which is connected to Gnd, will have high resistance. When storing 00, the resistances of the two resistors are opposite of the case when storing 11. However when storing 01 and 10, one of the resistors has to have some intermediate resistance. For example, when storing 10, the top resistor should be programmed to the intermediate resistance level and the bottom

resistor should be off completely. So, if we merge the top two resistance levels, the resistance distribution can be found in Figure 9.

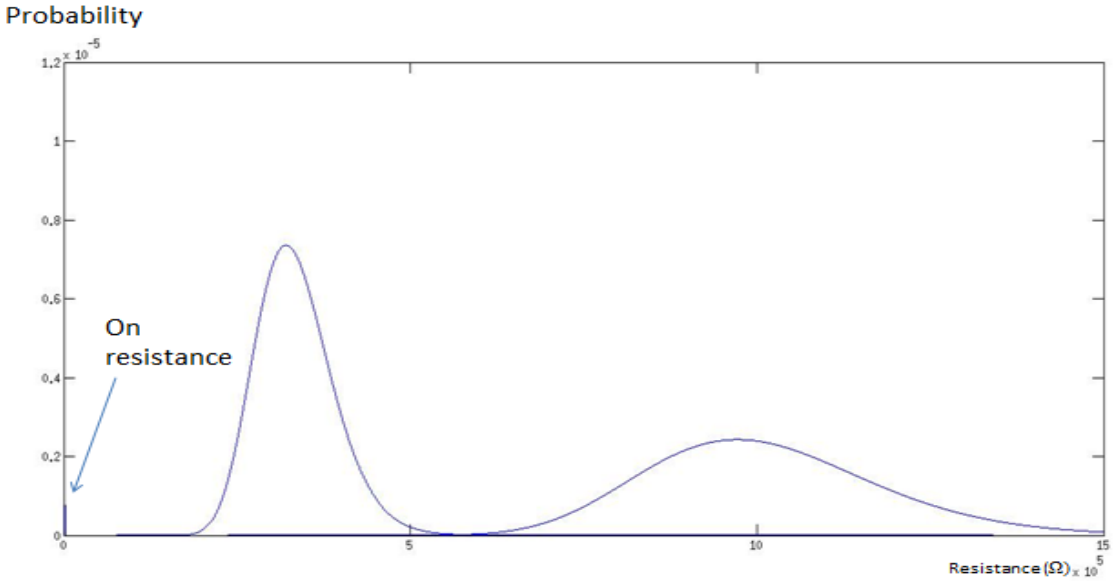

Figure 9. The resistance distribution of a 3-level PCM cell.

In this case, however, our assumption of equal leakage current no longer holds since the total resistance values for various logic levels are not the same anymore.

From Hspice simulation, we found that the access PMOS has leakage current of 31.2nA with 1V supply voltage. The equivalent off resistance can then be calculated, which is around 32.1MΩ. This is much greater than the off-state resistance of a PCM cell, which is around 1MΩ. After the Hspice simulation, the leakage current for storing 00 and 11 is 27.639nA. The leakage current when storing 01 or 01 is 26.262nA. So, when the memory cell stores 01 or 10, it consumes 5.24% less leakage power than 00 and 11.

When the memory cell suffers from resistance drift, the leakage current will change accordingly. The variation of the leakage current can also be modeled by Gaussian distribution. However, since the access transistor has much higher resistance than PCM cell resistance when turned off, the leakage current will generally remain the same.

1.6 CONCLUSION

From this study, we can conclude that the new PCM based FPGA architecture has a significant advantage over SRAM based FPGA in terms of area cost. For the single-output LUT case, the best area saving happens for LUTs of size 8 and the average saving is around 50 percent. With regards to timing, we also see moderate amount of saving with the best delay-saving LUT size to be 6, which delivers 14 percent delay reduction.

Also, the advantage of our new architecture over traditional SRAM based architecture can be further improved by the use of the hybrid single-output and twin-output LUT architecture.

In conclusion, we believe this multi-bit PCM based FPGA architecture provides better area and timing than to SRAM based FPGA architectures.

Chapter 2: SYMFET Based Graphene Memory Design

2.1 INTRODUCTION

As the CMOS technology scales down fast, the transistors are operating at higher and higher frequencies. Processor and chip operating frequencies are reaching several GHz. While the overclocked components are up to the challenge to meet the needs of computation power, the memory access speed starts to limit the overall computation power. The memory usage can be split into two distinct parts in a computer system: computing memory and storage. The computing memory can be further split into two parts: cache and main memory. As we know, the cache memory needs to be fast to be able to work with the processor chip. The choice for cache memory cell is the static random access memory (SRAM) cell. The SRAM cell has the fastest read/write speed; however, relatively large cell area and high leakage power consumption make it expensive to use. That is why for main memory, the dynamic random access memory (DRAM) cell is more widely used. DRAM memory has better cell density than SRAM, which allows DRAM to store more data per unit area. However, DRAM cells are generally slower and the cells need to be refreshed periodically.

Various emerging non-volatile memory (NVM) cells have been proposed recently. One common and good point of those memory cells is the data retention, which means the NVM cells can store data even when there is no power supply. This means that the leakage current will no longer exist in NVM cells since the leakage current happens when we supply voltage to the traditional volatile memory cell in order to maintain the data stored inside. Those attracting new memories are studied in [16]. According to the study, those NVMs all share a common problem in replacing the current memory: They are not as fast as SRAM while not as compact as the NAND flash memory. In other words, their timing performance and memory density are not within the top tier.

Recently, a new type of transistor structure called SYMFET is developed [17]. This type of memory, as we can see in Figure 10, uses the graphene sheet-to-sheet tunneling effect introduced in [18] to generate the on/off currents. When the Dirac points of two pieces of Graphene sheet align, we know the energy bands of those two

pieces of graphene match. In this case, the electrons from the filled state can tunnel to the empty state in the same energy level. Since the current is generated by the tunneling effect, the access speed is expected to be fast. However, the on/off ratio reported in [17] is not so promising since the ratio is inversely proportional to the cell size. In this work, we extend the GIG structure to a GIGIG structure to serve as a memory cell. The new GIGIG structure has better control of the on/off current and thus improves the on/off current ratio. The rest of the sections are arranged as follows: First, the memory cell structure will be introduced. Then, the read and write operation will be discussed separately. Finally, device simulation result will be presented and a conclusion will be made.

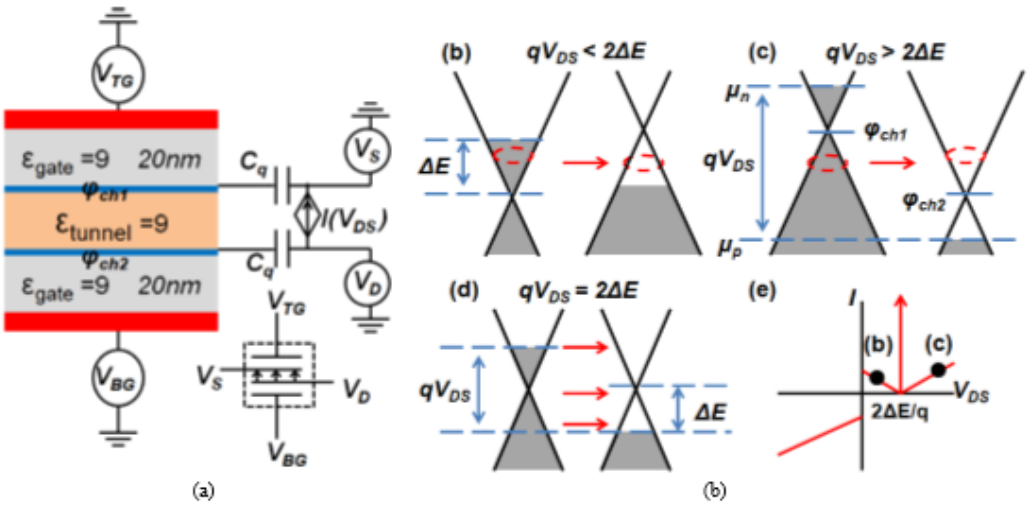

Figure 10. (a) The SYMFET device diagram. (b-c) Small tunneling current when Dirac points misalign. Only one matched energy level exists and supports tunneling current. (d) Large tunneling current when Dirac points align. There is a range of energy levels support tunneling current. (e) The I-V curve. [17]

2.2 Memory Cell Structure

In this section, the detailed memory structure will be introduced. The memory cell has a similar structure as the SYMFET, which has a graphene-insulator-graphene structure. In our design, we insert another graphene layer in between the SYMFET's two graphene layers to form a GIGIG structure as we can see in Figure 11. The purpose of the outside two graphene layers is the same as SYMFET, that is, to act as source and drain. The middle layer of graphene, however, is the floating gate node of this memory cell. The middle floating graphene

can be accessed by tunneling current either from source or drain. The drain-source voltage controls the potential difference and thus the alignment, or misalignment, of the drain-source Dirac points. Since the graphene sheet is capable of holding charges, the voltage potential of the middle floating gate depends on the tunneling currents from both drain and source. From the tunneling current equation we know:

$$I = G_1(V_{DS} - \frac{2\Delta E}{q})\,\mathrm{sgn}(V_{DS} - \frac{2\Delta E}{q})\tanh(\frac{qV_{DS}}{4k_BT})\tanh(\frac{LqV_{DS}}{\pi\hbar v_F}) + \left\{\frac{1.6}{\sqrt{2\pi}}G_1\frac{L\Delta E^2(2u_{11}^4+u_{12}^4)}{u_{12}^4 q\hbar v_F}\exp[-\frac{A}{4\pi}(\frac{qV_{DS}-2\Delta E}{\hbar v_F})^2] \times\frac{N_s(T)}{N_s(0)}\tanh(\frac{LqV_{DS}}{2\pi\hbar v_F})\right\} \quad [17]$$

The tunneling current will reach the maximum value when the Dirac points of two pieces of graphene sheet align. So, by controlling the alignment between the three graphene sheets, we can control the data stored inside the memory cell. If the Dirac points are aligned or close to aligned, high current density across the cell is expected and we can say that the memory cell stores logic 1. If the Dirac points are not aligned, in contrast, low current density is expected and the memory cell stores logic 0. Since the memory cell has a sandwiched structure, it is sufficient to use the middle layer to control the current across the entire cell. So, we can set the access voltage (drain-source) to some fixed voltage value that ensures the drain-source graphene sheets have near perfect Dirac points alignment. We also discovered from our experiment that, if the size of the graphene sheet is small (L is less than 100 nm), the current density does not drop very fast beyond Dirac point alignment voltage. And this is the reason why we can set the source-drain current to near alignment but not full alignment. The benefits of this design will be discussed in the next section.

Figure 11. The GIGIG structure of the memory cell. Blue strips represent graphene sheets; red blocks represent insulators.

2.3 READ OPERATION

2.3.1 Reading logic one from memory cell

The benefit of setting the drain-source voltage to near alignment but not full alignment is that in this case, if we put the Dirac points of the middle Graphene layer in between the source-drain Dirac-points, as shown in Figure 12, during write operation, the Dirac point of the middle floating gate will stabilize itself to a position in between the Dirac points of source and drain during read operations. This is because the closeness of two Dirac points affects the tunneling current density dramatically. If symmetrical cell structure is assumed (same tunneling current environment between floating gate to drain and to source), the stabilized Dirac point location for the middle gate is expected to be right in the middle of drain and source Dirac points. As we can see from Figure 12, the Dirac point level on the drain side is the highest among the three Dirac points. If the Dirac point of the middle gate is too close to the Dirac point of the drain, then an increase of the tunneling electrons from the drain to the gate is expected. The increasing tunneling electrons will introduce an imbalance between the drain-gate and gate-source tunneling current and thus excessive tunneling electrons from the drain will accumulate on the gate and thus bring down the gate Dirac point. With the Dirac-points of the drain and the gate get farther away, the drain-gate tunneling current density will decrease accordingly. So, due to this self-limiting tunneling effect, during a "1" reading, the Dirac point of the middle gate will always be "stuck" in the middle of the other two Dirac points. This design has two major benefits. Firstly, during write operation, it is hard to bring the gate potential exactly to a certain potential level. Secondly, during reading or writing, this design is more noise-tolerant.

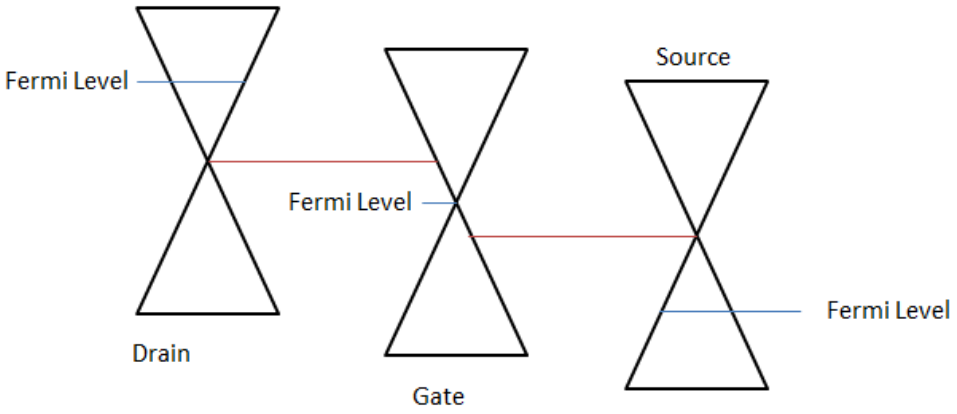

Figure 12. The GIGIG memory cell band diagram when reading a 1.

2.3.2 Reading logic zero from memory cell

When reading a 0, however, if we program the middle gate to have a low potential value, then the Dirac point of the gate will be far below the other two Dirac points as shown in Figure 13. In this case, as we can see, electrons can only tunnel from drain and source to the gate since the there is no empty state in the drain and source below the Dirac point of the middle gate. Thus, the tunneling electrons from drain and source will further bring down the gate potential. Thus, the gate potential will remain low and thus destroy the Dirac point alignment from drain to gate to source, thus limiting the total current through the entire cell.

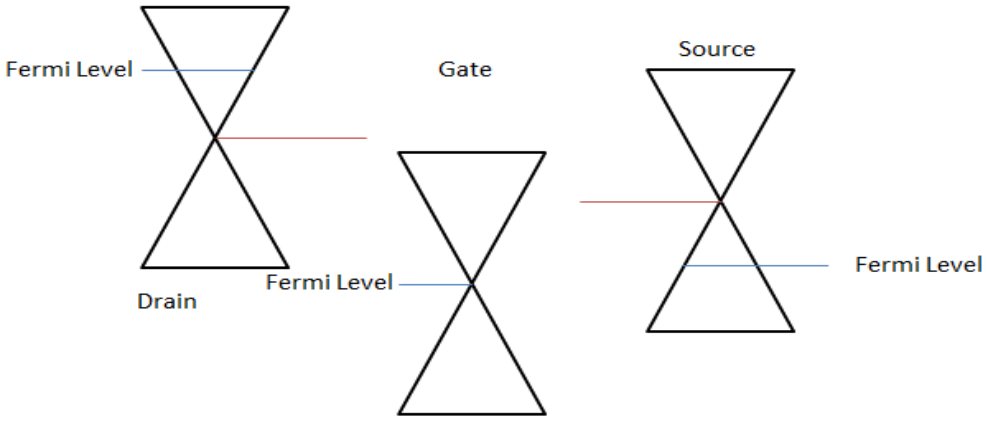

Figure 13. The GIGIG memory cell band diagram when reading a 0.

However, the gate voltage will not keep dropping when reading a zero. When the write circuit is designed, there is leakage current from the gate node. This will be discussed in the next section.

2.4 WRITE OPERATION

We know that the middle graphene layer should be floating and accessible by tunneling current from drain and source during read operation. We also know from the previous section that the potential on the middle graphene layer plays a critical role in determining the value stored inside the memory cell. So to program the memory cell, we need to change the potential on the middle graphene layer.

One simple way to do this is to make the middle gate not floating during the programming stage. By adding an accessing circuit to the middle gate, we can pull the voltage potential of the middle graphene layer to the desired value during write operation. The middle graphene layer will become floating if we turn off the access circuit. The leakage current from the gate through the access circuit can be compensated by the tunneling currents. By KCL, the drain-gate, source-gate and gate leakage will sum to zero. This will bring the Dirac point of the middle graphene layer to a stable position.

So, when writing a one to the memory cell, we would pull the middle gate potential to a value so that the Dirac point of the middle gate is between the Dirac points of source and gate. The middle gate potential will stabilize itself during read operation due to self-limiting tunneling. If writing a zero is desired, we can simply bring down the middle gate potential to ground. In this case, electrons will be able to tunnel from drain and source to the gate. This will bring down the middle gate voltage while increasing Vgs for the access NMOS. So if the word line is kept low when not writing, the access device will be slowly turned on to balance the tunneling current from drain and source. By KCL, we know that $I_{gate(leakage)} = It_{(Gate\ to\ Source)} + It_{(Gate\ to\ Drain)}$.

At this balance point, if we access the data from the source side, we will find that the current direction varies when storing a one or a zero. When storing a one, the electrons tunnel from drain to gate and gate to source, meaning that the current direction is from source to drain with relatively large current density. However if a zero is stored, the electrons tunnel from drain and source to the gate. At the source side, we will find the current direction from gate to the source with low current density. Figure 14 shows the memory cell with the write

circuit. This particular memory cell behavior makes it very simple for us to distinguish one from zero. And the current on/off ratio here is not important since the current direction changes.

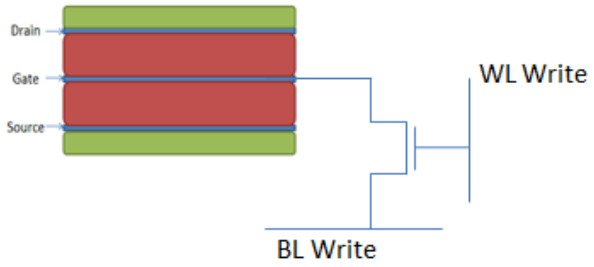

Figure 14. The memory programming circuit.

2.5 PERFORMANCE CALCULATION AND SIMULATION

Since little work has been done on this type of device, our results are obtained by equations derived from previous sections. Here we present on/off current density and stable potential level for gate during reading.

We here present the device with Vds = 2*Delta E + 0.2V. The 0.2V here is the designated tunneling gap between drain and source as discussed in the previous section. The device parameters we used for simulation are: L = 100nm, V(drain gate) = 2V, Delta (intrinsic) = 0.1eV, T_{tunnel} = 0.5nm, relative permittivity = 9. Vds = 0.8V. The I-V simulation curve can be found in Figure 15.

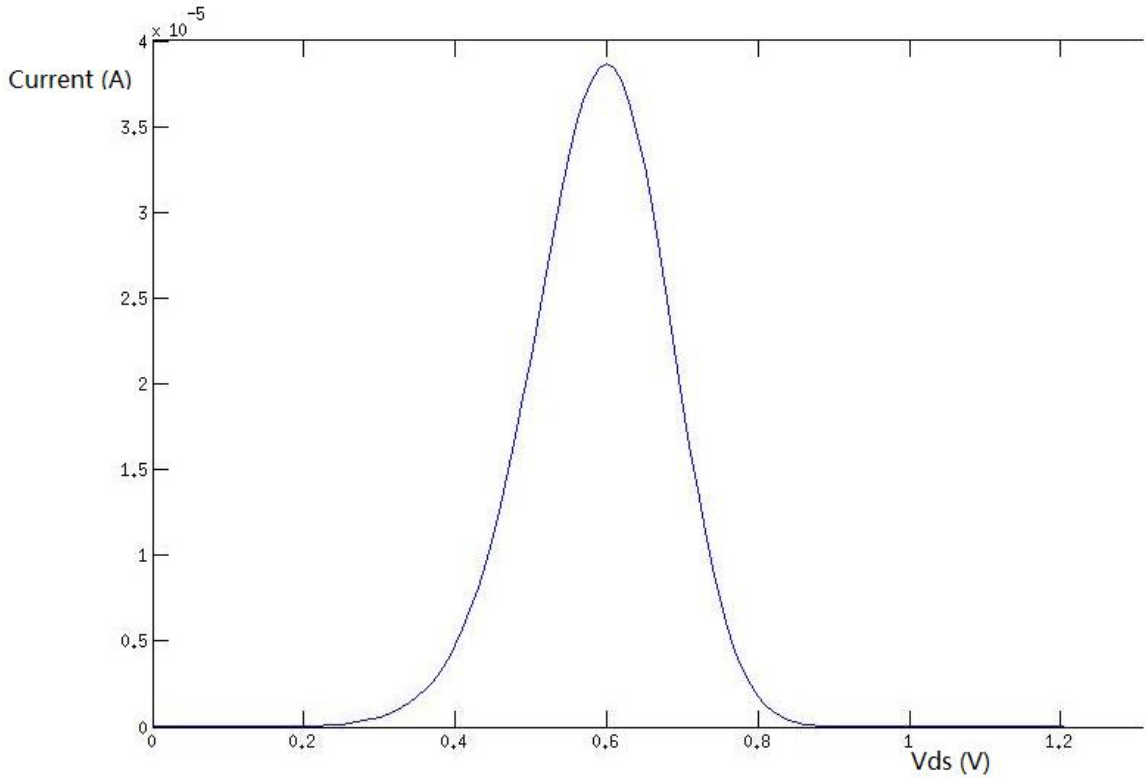

Figure 15. The I-V curve for L=100nm SYMFET.

While we know that for our device, if the Dirac points match between two graphene sheets, the tunneling current is half of the max tunneling current of a SYMFET. Since we set Vds to be 2*DeltaE + 0.2V, then by KCL we know for reading a logic one:

$$It(_{\text{Source to Gate}}) + I_{\text{gate(leakage)}} = It_{\text{(Gate to Drain)}} \quad (1)$$

$$Vd\text{-}Vs = 2*DeltaE + 0.2V \quad (2)$$

Since the I-V curve of SYMFET is symmetric, we can then use half of the I-V curve to represent the I-V relation between source and gate while the other half represents the I-V relation between drain and gate as Figure 16 shows.

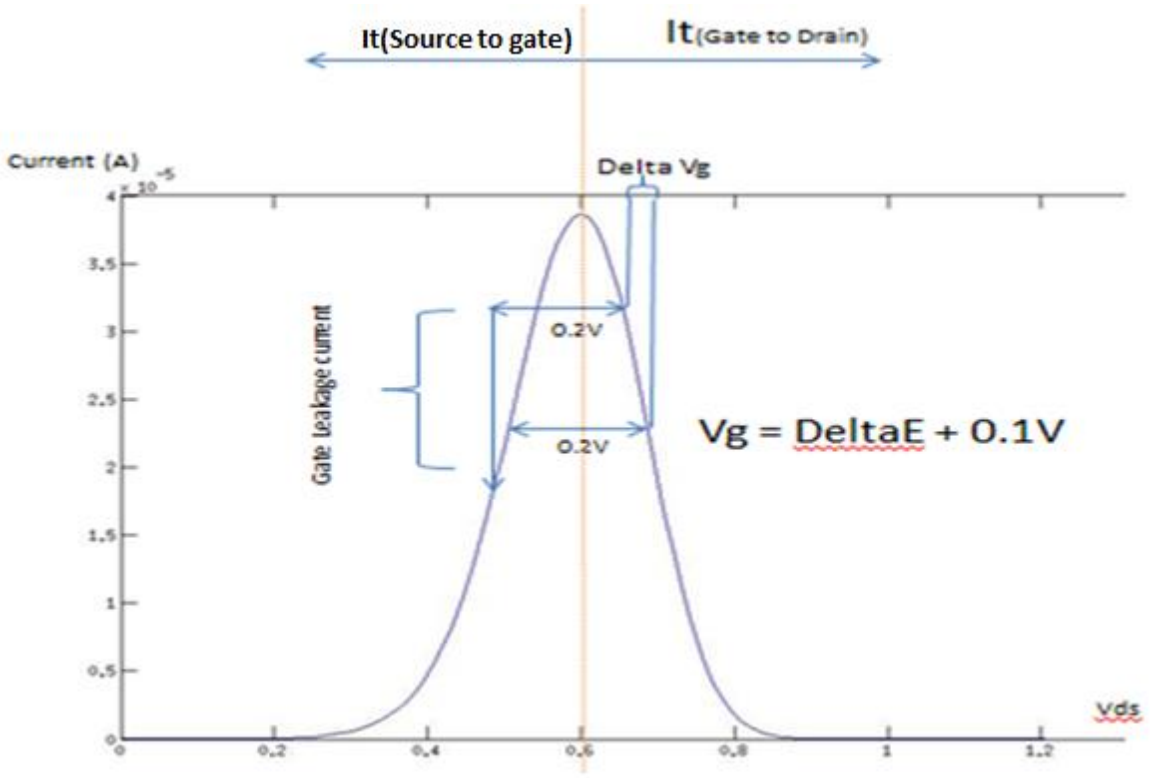

Figure 16. The I-V curve for storing/reading a 1. The stable gate potential and current density can also be found using this graph.

In Figure 16, if no gate leakage current is present, then we would assume:

$$It(\text{Gate to Drain}) = It(\text{Source to Gate})$$

We also know that the two tunneling currents to have the same current density, which means the gate potential should be:

$$Vg = Vds + 0.2V - \text{Delta E}$$

If the gate potential is shifted by the amount of delta Vg as shown in Figure 16, we just shift the 0.2V line along with the I-V curve. This is valid because the Dirac point of the Gate is in the middle of the source-drain Dirac points and the 0.2V gap between drain-source alignment still holds. We can then find the corresponding tunneling current. If, on the other hand, there is leakage through the gate access device, then we have to find a gate potential that satisfies the KCL equation we derived previously. The leakage portion can be also found in Figure 16 and the corresponding gate potential can be easily calculated. In our simulation case, the middle gate

potential = 0.7V and the tunneling current = 11μA. Since the leakage current is much smaller, it is ignored. When storing a zero, as we can see from Figure 17, the I-V curves represent the sum of drain-gate tunneling current and source-gate tunneling. However, this time, the gate potential is set to ground so that the sum of the two tunneling currents equals the leakage current. Thus, the gate potential for storing a zero can be calculated accordingly.

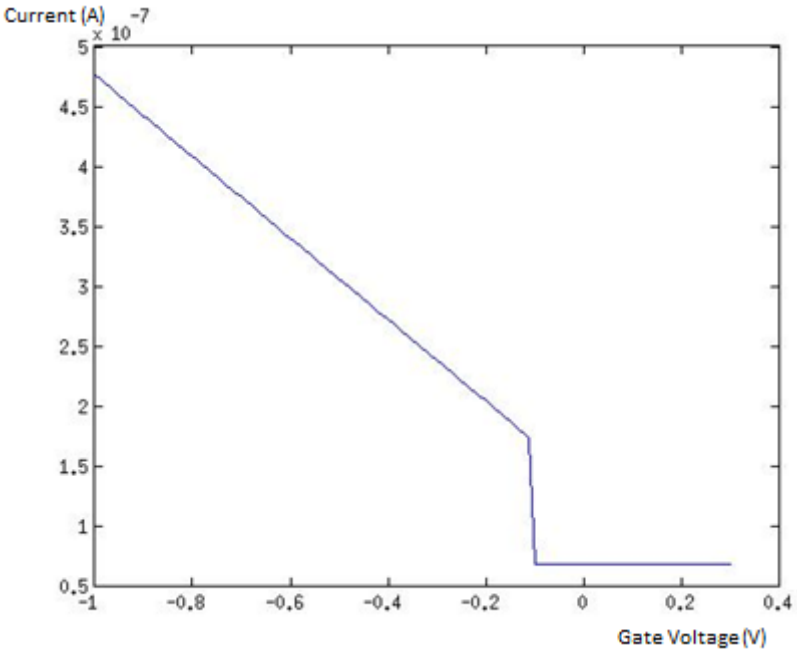

Figure 17. The I-V curve for storing/reading a zero.

So from our simulation, the current when reading a one is 11μA and the direction is from source to gate at gate potential 0.7V. The current for reading a zero is -16.7nA at gate potential -0.23V. The output current ratio is 658.7.

2.6 FUTURE WORK

Another novel way to program the cell is to use the drain-gate or source gate tunneling. As indicated in [18], the graphene sheet has quantum capacitance and this is studied in [19]. So, by ramping up/down the drain

potential and preventing the source-drain tunneling current, the gate potential is expected to follow the potential change of the drain side. In this way, we can get rid of the extra access transistor and make this memory device function solely on tunneling. Also, by now, this memory is volatile. This is because the gate leakage current will destroy the potential on the middle gate by time. However, if the middle gate is truly floating, then it may become non-volatile.

2.7 CONCLUSION

From this work, we can see that while the SYMFET based memory is not as compact as the NAND flash memory, its high speed performance makes it a very attractive substitute for SRAM. Compared to SRAM, SYMFET based memory consumes much less area than SRAM cells. However, since only conceptual work has been done for this newly purposed device, more simulation and design modification or a device fabrication should be done in the future.

References

[1] D. Chen, J. Cong, and P. Pan, "FPGA Design Automation: A Survey, Foundations and Trends in Electronic Design Automation," NOW Publishers, November 2006, 137 pages.

[2] Y. Xie, "Future memory and interconnect technologies," *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2013, March 2013, pp. 964,969.

[3] C. Dong, C. Chen, S. Mitra, and D. Chen, "Architecture and Performance Evaluation of 3D CMOS-NEM FPGA," *Proceedings of IEEE/ACM International Workshop on System Level Interconnect Prediction*, June 2011.

[4] C. Chen, R. Parsa, N. Patil, S. Chong, K. Akarvardar, J. Provine, D. Lewis, J. Watt, R.T. Howe, H.-S.P. Wong, and S. Mitra, "Efficient FPGAs using nanoelectromechanical relays," *ACM/SIGDA International Symposium on Field Programmable Gate Arrays - FPGA*, 2010, pp. 273-282.

[5] T. Ogun et al. 'RRAM-based FPGA for "Normally Off, Instantly On" applications,' *Journal of Parallel and Distributed Computing*, June 2014, pp. 2441-2451.

[6] G. Pierre-Emmanuel et al., "Phase-change-memory-based storage elements for configurable logic," *2010 International Conference on Field-Programmable Technology, FPT*, 2010, pp. 17-20.

[7] Y. Chen, J. Zhao, and Y. Xie, "3D-NonFAR: Three-dimensional non-volatile FPGA architecture using phase change memory," *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on*, 2010, pp.55-60.

[8] S. Raoux et al., "Phase-change random access memory: A scalable technology," *IBM. J. Res & Dev.*, 2008, 52(4/5), pp. 465-4.

[9] Berkeley Logic Synthesis and Verification Group, ABC: A System for Sequential Synthesis and Verification, Release 131218. http://www.eecs.berkeley.edu/~alanmi/abc/.

[10] S. Babayan-Mashhadi, and R. Lotfi, "Analysis and Design of a Low-Voltage Low-Power Double-Tail Comparator," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 2014, pp. 343-352.

[11] V. Betz and J. Rose, "VPR: A new packing, placement and routing tool for FPGA research," presented at International Workshop on Field Programmable Logic and Applications, 1997.

[12] E. Ahmed and J. Rose, "The effect of LUT and cluster size on deep-submicron FPGA performance and density," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, March 2004, pp. 288-298.

[13] J. Li, B. Luan and C. Lam, "Resistance drift in phase change memory," *Reliability Physics Symposium (IRPS), 2012 IEEE International*, April 2012, pp. 6C.1.1.

[14] N. Papandreou, H. Pozidis, A. Pantazi, A. Sebastian, M. Breitwisch, C. Lam and E. Eleftheriou, "Programming algorithms for multilevel phase-change memory," *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, May 2011, pp. 329-332.

[15] S. Nak Hee et al., "Tri-level-cell phase change memory: toward an efficient and reliable memory system," *Proceedings of the 40th Annual International Symposium on Computer Architecture*, 2013, pp. 440-451.

[16] Yuan Xie, "Modeling, Architecture, and Applications for Emerging Memory Technologies," *Design & Test of Computers, IEEE*, 2011, pp. 44-51

[17] P. Zhao, R. M. Feenstra, G. Gu, and D. Jena, "SymFET: A Proposed Symmetric Graphene Tunneling Field-Effect Transistor," *Electron Devices, IEEE Transactions on*, March 2013, pp. 951-957.

[18] R. M. Feenstra, D. Jena, and G. Gu, "Single-particle tunneling in doped graphene-insulator-graphene junctions," *J. Appl. Phys.*, vol. 111, 2012, pp. 043711.

[19] T. Fang, A. Konar, X. Huili, and D. Jena, "Carrier statistics and quantum capacitance of graphene sheets and ribbons," *Applied Physics Letters*, vol. 91, 2007, pp. 092109.