**EE599 Project Phase III:**

**Final Project Report**

**PART A:**

1. **Project Details:**
2. **Project Topic:** Intermittent Computing
3. **Team Members:**

Bilkish Ara Naikodi Phalguni Bhangod Siddharth Gupta

[naikodi@usc.edu](mailto:naikodi@usc.edu) [bhangod@usc.edu](mailto:bhangod@usc.edu) [gupt232@usc.edu](mailto:gupt232@usc.edu)

1. **Problem Description:**

There are many modern applications where there is no continuous supply of power, and hence stored energy is used to perform computations. Such applications are usually tolerant towards errors and require fast and low power computations. Intermittent computing is used in these cases to perform required tasks by taking relevant parts of the data and making tolerable approximate calculations, in limited time and power constraints.

1. Mathematical Challenges:

The challenges are to determine the advantage in terms of time and power, over existing Intermittent Computing technology by replacing the standard main memory with a memristor and exploring new approximation techniques mainly for multiplication. Multiplication is a basic operation executed maximum number of times in a piece of code in several applications and is time consuming. Hence, we have chosen to approximate multiplication operation.

1. Software Challenges:

Implementation of flow of Intermittent Computing architectural model in Python, using access times/delays calculated and obtained from NVSim and Cacti, for memristor based memories, and Xilinx ISE design suite for approximate multiplier modelled on FPGA.

1. Algorithmic Challenges:

Design of flow of Intermittent computing processor stages including the instruction fetch, decode, execute, memory and write-back stages and checkpointing in Python. Conversion of high-level source code in Python to Assembly level Language with replacement of the conventional multiplication operation with operations that perform approximate multiplication (replacement of MULT operand with SHIFT and ADD operands).

1. Motivation:

Intermittent Computing applications must be performed with limited energy and time for execution and are hence more accepting of errors in computations. So, we have chosen to design an approximate multiplier, which gives an acceptable result, to replace the frequently occurring multiplication operations, and replace the main memory with a memristor because of huge access times gap (appx. 100 times faster) in terms of read and write access times.

1. Novelty:

We are replacing the conventional multiplier with an approximate multiplier design, and the main memory with a memristor based ReRAM. A memristor has been chosen because of large savings in time required for memory read and write access, as compared to standard SRAM/DRAM. Because the frequency of multiplication operations is high, we have chosen to approximate it in a time and power constrained environment.

1. **Project Timeline:**

|  |  |
| --- | --- |
| Stage I | Read the following papers to understand Intermittent Computing and its challenges:   1. Intermittent Computing - Challenges and Opportunities 2. The What’s Next Intermittent Computing Architecture |
| Stage II | We decided to improve upon the Paper 2, so we started reading the following papers on approximate multiplier designs:   1. SiMul: An algorithm Driven approximate multiplier Design for machine learning. 2. RoBA Multiplier: A Rounding-based approximate multiplier for high speed yet energy efficient Digital Signal processing.   We decided to use a memristor in place of main memory to determine its advantages and went through the following paper:   1. A Novel Design for memristor-based logic switch and crossbar circuits. |
| Stage III | 1. Designed a Python model of Memristor cell for understanding. 2. Learned to use Cacti to determine the read/write access times and latencies for cache and RAM. |
| Stage IV | 1. Learned to use NVSim because Cacti does not support non-volatile memory modelling, to calculate the read/write access times and latencies. 2. Simulated basic memristor based RERAM memory model on NVSim. 3. Developed Verilog code and the testbench code for an approximate multiplier in ModelSim and tested it for accuracy. |
| Stage V | Simulated several RERAM models on NVSim to determine the advantage of using a RERAM over the conventional main memory and compared the read/write access times using RERAM v/s conventional SRAM, by modelling it for both Cache and RAM. |
| Stage VI | Simulated the approximate multiplier code on an FPGA model using Xilinx ISE Design suite to obtain the power consumed and area occupied for the multiplier. |
| **Phase III**  Stage VII | To test the ‘shift and add’ approximate multiplier code for accuracy v/s power and latency for different shift value parameters. |
| Stage VIII | Test and understand the checkpointing mechanism implemented in What’s Next Intermittent Computing paper. |
| Stage IX | Implement the Intermittent computing processor stages including the instruction fetch, decode, execute, memory and write-back stages and checkpointing in Python, along with an assembly code to replace the MULT operand with SHIFT and ADD operands, with NVSim access time and latency parameters input in the Python code for RERAM as main memory. |
| Stage X | 1. Check the percentage of improvement for our approximate multiplier design over the What’s Next Intermittent Computing paper design by comparing the multiplication results accuracy. 2. Check the percentage of improvement for the replacement of RERAM over conventional main memory in terms of read/write access times and latencies, by running benchmark programs for each one of them. |

Intermittent Computing

*Bilkish Ara Naikodi, Phalguni Bhangod, Siddharth Gupta*

University of Southern California, Ming Hsieh Department of Electrical Engineering [naikodi@usc.com, bhangod@usc.com,](mailto:ranipaga@gmail.com) [gupt232@usc.com](mailto:gupt232@usc.com)

***Abstract*—** *Intermittent Computing is applicable to devices that harvest energy from their surroundings when it is available, and store these bursts of energy for computations, thereby eliminating the need for a power source. The energy is not always available continuously and is therefore termed as intermittent. Nowadays, many applications such as medical sensors, small satellites etc., are expected to perform computations with small amounts of power and time. Therefore, energy distribution becomes critical and we allot this limited energy and time to more frequently occurring and meaningful computations. These devices comprise of hardware elements such as a CPU, sensors, transceivers, volatile memory, in which data is lost on power exhaustion and non-volatile memory, in which data is retained even on power exhaustion. To cater to this time constrained operation, we are replacing the conventional main memory with a memristor based memory device, which has access times that are approximately 100 times faster. These energy-starved devices are also more acceptable of approximate results for the mathematical computations. In this project we aim to generate approximate, yet acceptable results in a shorter duration of time and for smaller amounts of energy, for which we propose an architecture that is like What’s Next Intermittent Computing Architecture paper but improving upon it by choosing a Memristor device as opposed to a conventional main memory, by designing another approximate multiplier design.*

***Keywords-****energy harvesting, intermittent computing, approximate computing, memristor****;***

I. *INTRODUCTION*

In recent years, small and low-power computing devices are seeing increased shifts in technology trends, towards numerous application domains such as medical sensors, implantable devices, satellites and computer vision. These devices operate using energy exclusively from the environment. These battery-less devices are powered entirely by energy gathered from environmental sources such as radio waves, solar light or vibration.

These devices present a unique challenge in terms of energy consumption as energy is not continuously available, but it is available intermittently in bursts, and all the meaningful computations must be done in the time during which the energy is available for. Such devices are also more accepting of approximate results and are error tolerant.

After referring to several papers, we have summarized the following ones as being significant to our project:

1. Intermittent Computing – Challenges and Opportunities
2. A Reconfigurable Energy Storage Architecture for Energy-harvesting Devices
3. The What’s Next Intermittent Computing Architecture.

In this paper we have implemented the “The What’s Next Intermittent Computing Architecture” paper and we propose two improvements over this architecture which are:

* 1. Replacement of a conventional multiplier with an approximate multiplier design and measure the accuracy rates for multiplication results and,
  2. Replacement of conventional memory with a non- volatile memristor based ReRAM.

The motivation behind these two changes is:

1. Multiplication was chosen as the operation to be approximated because of its high frequency of occurrence and high time latency.
2. Memristor was chosen over conventional main memory because of its lower area and power consumption, and higher speed of read/write access.