National Institute of Technology Rourkela

Synopsis Seminar

Seminar Title	: In-memory Computing on Memristor Crossbar: Architecture and Algorithms
Speaker	: Kajal Kishori (Rollno : 519cs1008)
Supervisor	: Prof. Sumanta Pyne
Venue	: New Conference Hall (CS 323)
Date and Time	: 06 Feb 2025 (11:00 AM)
Abstract	: In traditional von Neumann computers, data-intensive applications often suffer from performance degradation and higher power dissipation primarily because of the frequent data transfers between the memory and the processor. This movement of data back and forth creates bottlenecks, slowing computation and increasing energy consumption. To mitigate these issues, non-von Neumann architectures have been proposed, which leverage in-memory computing to perform computations directly within the memory itself, thereby reducing the need for frequent data transfers. One key component enabling this shift is using non-volatile memory built with memristors, an alternative building block allowing efficient data storage and processing. Its unique characteristic lies in its ability to alter resistance in response to the flow of electrical current. The resistance of a memristor lies between two logical states - low resistance, RON (logic 1 or set) and high resistance, ROFF (logic 0 or reset) where, RON << ROFF. Various logic design strategies have emerged for memristor crossbars, each offering unique advantages and capabilities. One such approach is memristive-only logic, where stable evaluation of logic operations can be achieved by applying a single voltage pulse at the circuit's gateway. However, recent advancements have introduced a more sophisticated stateful logic design known as Memristor Aided LoGIC (MAGIC), which surpasses traditional approaches like IMPLY. The concept of MAGIC involves the utilization of memristors to enable logic operations directly within memory arrays, thereby facilitating in-memory computing. By incorporating memristors, it becomes feasible to implement logic gates such as Boolean NOT and NOR gates directly within a crossbar memory array, which holds the potential to significantly improve processing speed and energy efficiency. This capability enables arithmetic and logic operations to be performed directly within the memory, eliminating the need for data movement between the memory and the processor. The way its behavior differs from other conventional circuit components like resistors, capacitors, and inductors, creates new opportunities for applications in neuromorphic computing, memory storage, hardware security, and other cutting-edge fields. To support in-memory computing efficiently, architectural support has been proposed in this work. These include the addition of specialized instructions, such as FILL and MCMP, to the instruction set. These instructions facilitate constant-time in-memory comparisons, which are crucial for various computational tasks. By leveraging these architectural advancements, in-memory set operations are implemented using the proposed instruction set. The considered set operations are union, cartesian product, transitive closure, and power set. The performance benefits of these in-memory operations are substantial, with significant reductions in energy consumption compared to traditional CPU-based approaches. On average, the energy reduction achieved by in-memory operations is impressive: 168× for union, 249× for cartesian product, 134× for transitive closure, and 264× for power set generation. However, it's important to note that the superiority of in-memory computing varies depending on the input size. For larger input sizes, CPU-based approaches for union, cartesian product, and transitive closure outperform in-memory operations by factors of 1.58×, 1.23×, and 2.07×, respectively. Nonetheless, in-memory power set generation still achieves an average speed-up of 1.14×, showcasing its efficiency even for larger datasets. The next work introduces a novel approach for conducting in-memory comparisons of integers (signed and unsigned) and floating-point numbers using memristor crossbars, guided by specific instructions. By leveraging voltage comparators, two-bit sequences of ones can be accurately compared in the analog domain. For the comparison of m-bit integers and 32-bit floating-point numbers, the crossbar space required is 7×m and 27×32, respectively. Comparisons made in-memory prove to be as effective as, or sometimes even better than, CPU-based comparisons. This approach significantly reduces energy consumption by a factor of 50-400×. The efficacy of CPU and mMPU-based comparisons is tested on programs, such as set union and bubble sort algorithms. For unsigned integers, in-memory union and bubble sort algorithms exhibit an average energy reduction of 171.76× and 150.49×, respectively. Similarly, for signed integers, the energy savings are notable, with in-memory union and bubble sort algorithms requiring 155.49× and 146.55× less energy on average, respectively. When it comes to floating-point numbers, the energy consumption reduction achieved by in-memory union and bubble sort algorithms is also significant, with an average reduction of 42.41× and 45.94×, respectively. However, it's worth noting that while in-memory comparisons drastically reduce energy consumption, there is a trade-off in terms of delay. On average, CPU-based union and bubble sort algorithms exhibit lower delays, with reductions of 1.59× and 1.91× for unsigned integers, 1.83× and 2.15× for signed integers, and 5.92× and 5.27× for floating-point numbers, respectively. The memristor boasts an array of impressive qualities, including high-density integration, high-speed switching capabilities, low power consumption, and rapid read/write times of less than 1 nanosecond. These attributes position it as a promising option for future non-volatile memory systems. However, one of the significant obstacles in the construction of a crossbar memory design is the sneak-path problem. The sneak-path problem arises when attempting to read a memristor in a high-resistance state while a series of memristors in a low-resistance state exist in parallel to it. This situation can lead to an erroneous interpretation of the high-resistance memristor as being in a low-resistance state due to the influence of the adjacent low-resistance memristors. When performing arithmetic logic unit (ALU) operations on a memristor crossbar, the presence of sneak-path current can distort the accurate outcomes. Addressing this challenge typically involves implementing various techniques at both the hardware and algorithmic levels. Hardware solutions may include incorporating isolation elements such as diodes or transistors between memory cells to minimize sneak paths or employing advanced sensing and signal processing circuits to distinguish between genuine and

spurious signals. Algorithmic approaches may involve optimizing read and write algorithms to mitigate the effects of sneak paths or implementing error correction mechanisms to detect and rectify erroneous data. Therefore, the next work conducts a mathematical analysis to assess the impact of sneak-path current during the execution of in-memory instructions on the memristor crossbar array. Notably, for a given instruction set designed for in-memory operations, it is observed that the sneak-path current is non-destructive. Apart from this, an algorithm is designed to find all possible sneak-paths on the n×m crossbar between input voltages and the ground. This algorithm considers one/two-input operations as given in the instruction set. Furthermore, it is found that the sneak-path current consumes significantly less energy per bit compared to single and two parallel in-memory operations. Specifically, it consumes 5.02× to 12.92× times less energy per bit than single in-memory operations and 35.3× to 36.78× times less energy per bit than two parallel in-memory operations.