**Project Outline**

**GENERATOR**

**Stochasticity can be used in:**

* **Noise vector**
* **Weight initialization**

**Theory Part:**

Implementing a GAN model on a hardware device using RRAM (Resistive Random-Access Memory) for stochasticity requires careful hardware architecture design to accommodate the specific requirements of GAN training. Here, I'll provide a high-level overview of the key components and their functions in the hardware architecture:

1. **Resistive Random-Access Memory (RRAM):**

• RRAM will be used to introduce stochasticity in the GAN model. It can act as a source of randomness, which is essential for certain parts of the GAN training process, such as the noise injection during the generator's input and the noise introduced during the discriminator's training.

2. **Processing Elements (PEs):**

• PEs are the core computational units responsible for executing the GAN model's matrix multiplication and convolution operations.

• Each PE is equipped with ReRAM for storing weights and activation ZAvalues during the computation.

3. **On-Chip Mesh Interconnect:**

• The PEs are connected through an on-chip mesh interconnect to facilitate communication and data exchange between the PEs during computations.

• The mesh interconnect enables efficient parallel processing and data flow across the hardware architecture.

4. **Vector-Matrix Multiplication Engines (VMMEs):**

• Each PE contains multiple VMMEs, specialized for performing vector-matrix multiplication efficiently.

• VMMEs leverage the RRAM-based stochasticity to introduce randomness during the GAN training process.

5. **Buffers and Registers:**

• Each PE is equipped with buffers for caching temporal data, reducing the need for frequent data access from external memory.

• Registers are used to aggregate the output of the computations performed by the VMMEs.

6. **Analog-to-Digital Converters (ADCs) and Digital-to-Analog Converters (DACs):**

• ADCs are used to convert analog signals from the RRAM-based stochasticity into digital values, which can be processed by the hardware components.

• DACs are employed to convert digital values back to analog signals when interacting with the RRAM.

7. **Common Driver and MUX:**

• In the VMMEs, multiple MCAs (Memristive Crossbar Arrays) share a common driver, which controls their operations and facilitates data flow.

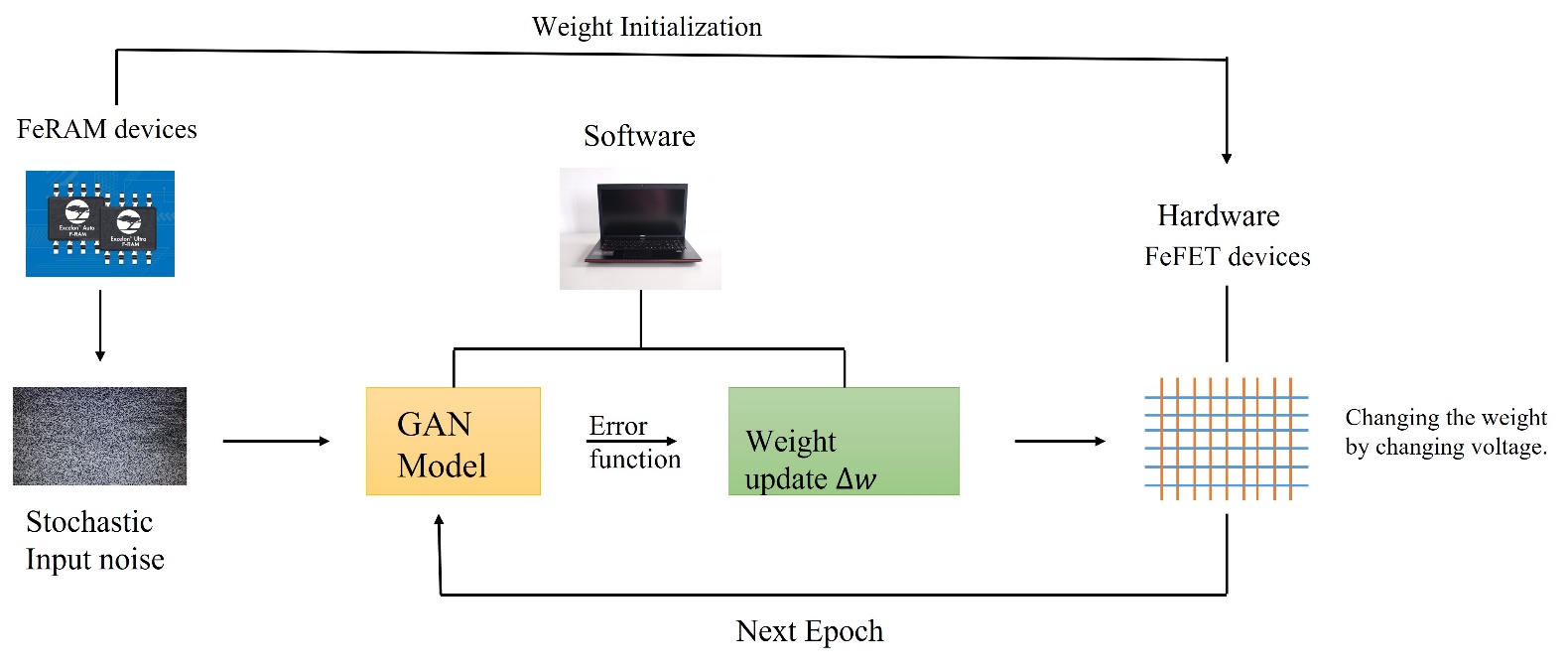
• A bitline MUX (Multiplexer) is utilized to select the results from the active MCA during the matrix multiplication operations.

8. **Scheduler and Mapper:**

• The scheduler controls the computation flow, ensuring the execution of computations in the correct order and managing resources efficiently.

• The mapper is responsible for mapping the dataflow to the PEs, determining how data is distributed among the processing elements.

The hardware architecture leverages RRAM-based stochasticity for introducing randomness into the GAN model. The PEs, VMMEs, and on-chip mesh enable efficient parallel processing and communication among the components. The use of buffers, registers, ADCs, and DACs helps manage data and facilitate data flow within the hardware device. The scheduler and mapper coordinate the computation and data distribution, optimizing the GAN training process.



**Algorithm Part:**

A GAN model was created using the MNIST dataset to create fake images. The model parameters are as follows:

* Batch size = 128
* Epochs = 40
* Total images = 60,000
* Optimizer = Adam

The model was simulated in google colab.

**Experiment Part:**

The experiment is to test the model in hybrid mode, where the errors will be estimated in software mode and the weights will be updated using an array of FeFET devices. The network will be run again with the updated weight until 40 epochs. A comparison will be done between the software mode and the hybrid mode to test GAN accuracy at generating fake images.

The stochasticity required for noise vector and weight initialization will be given by the stochastic distribution in Ferro electric devices.