Implementing a Half-Wave Rectifier signal using a **Single Port ROM** in Verilog involves two main steps: pre-calculating the sine wave values (where negative values are clipped to zero) and creating the Verilog module to read these values.

Instead of calculating sine values in real-time (which is computationally expensive in hardware), I used a **Look-Up Table (LUT)** stored in ROM.

- **Positive Cycle:** The ROM stores standard sine wave values.
- **Negative Cycle:** The ROM stores zeros.
- **Address:** Acts as the "phase" of the signal. As the address increments, the output follows the shape of a half-wave rectified signal.

I first need a file (e.g., `rectifier_values.mem`) containing the data.

The verilog module uses `$readmemh` to load the rectified sine data into a single-port ROM.

Used a simple Python script to generate the `.mem` file for the Verilog simulation:

To verify the design, a testbench that acts as a stimulus. This testbench will generate a clock signal and a counter to cycle through the ROM addresses, observe the half-wave rectified output in the waveform viewer.

Phase Precision: For a smoother wave, I increased the address width (e.g., 10-bit for 1024 points) and the data width (e.g., 12-bit or 16-bit).