**Introduction and Overview**

In modern digital systems, Static Random-Access Memory (SRAM) is widely used for high-speed data storage. SRAM plays a critical role in various applications, including cache memory and embedded systems, due to its low latency and reliability. For this project, I designed a simple 4x4 SRAM module using Verilog, a hardware description language. The module was simulated and verified using Vivado software to validate its functionality and understand how read and write operations are handled in a memory array. This project deepened my understanding of memory operations and digital logic design.

A computer screen shot of a computer

Description automatically generated

**RTL Design Overview**

The SRAM design focuses on two main operations:

1. **Write Operation**: When the write enable (we) signal is high, data is written to the specified memory location.
2. **Read Operation**: When we is low, data is read from the memory location and presented on the data\_out signal.

The module utilizes a clock signal (clk) for synchronization, ensuring that memory operations occur at the correct time. The design allows for simple and efficient control of memory access through a combination of address, control, and data signals. Key signals include:

* clk: Synchronizes all operations.
* we: Controls write operations.
* address: Specifies the memory location for read/write operations.
* data\_in: Provides data to be written into the memory.
* data\_out: Outputs data stored in the specified memory location during read operations.

A screenshot of a computer

Description automatically generated

**Waveform Analysis and Simulation**

Simulation and waveform analysis were conducted in Vivado to validate the SRAM module's functionality. Key observations from the simulation include:

* **Write Operation (we = 1)**:
  + Data is stored at the specified memory address.
  + The data\_out signal remains unchanged, as no read operation occurs simultaneously.
* **Read Operation (we = 0)**:
  + The data\_out signal reflects the value stored at the specified memory address.

The waveform analysis confirmed the correct behavior of the SRAM module, demonstrating proper synchronization of signals and accurate handling of read and write operations.

**Device Utilization Summary**

After synthesizing the design in Vivado, the device utilization report indicated the following:

* **Logic Elements**: The SRAM design utilized minimal logic elements, given the small memory size (4x4 array).
* **Flip-flops**: A small number of flip-flops were used to store the 4-bit data for each memory address.
* **Lookup Tables (LUTs)**: The design employed a minimal number of LUTs to implement basic read and write operations efficiently.
* **I/O Pins**: The module used:
  + 1 input for the clock.
  + 1 input for the write enable signal.
  + 2 inputs for the address.
  + 4 pins for data input/output.

The lightweight design can be easily scaled for larger memory configurations by modifying the address and data widths.

**Key Insights**

1. **Simulation Tools**: Using Vivado’s simulation and waveform analysis tools is essential for validating the correctness of a design before physical implementation.
2. **Control and Timing**: Ensuring proper timing and control of signals is crucial for accurate memory operations in digital systems.
3. **Design Verification**: Waveform analysis provides a clear representation of logic behavior, helping identify and fix issues early in the design phase.
4. **Scalability**: The 4x4 SRAM module demonstrates a foundational design that can be expanded to accommodate larger memory arrays with appropriate modifications.

**Conclusion**

This project significantly enhanced my knowledge of digital memory systems and RTL coding. It reinforced the importance of simulation and verification in hardware design, particularly using Vivado. The insights gained from this project will aid me in future endeavors involving VLSI design and digital systems.