Polytech Dijon

5A Informatique et Electronique

Projet System On Chips (SoCs)

Static Random Access Memories(SRAM) Controller

Done by :

MAMBO MOTSOU JUNIOR D.
FONKOUA CHARLE LUCE SORIANE

Supervised by :

Mr. Julien DUBOIS

Year 2025–2026

Sommaire

Table des Figures

Chapter 1

Introduction

Static Random-Access Memories (SRAM) are memories commonly used in embedded systems and System-on-Chip (SoC) architectures due to their fast access time and simple control interface. They are particularly well suited for applications requiring deterministic and high-speed data access, such as control units, buffering, and real-time processing systems.

This projet work focuses on the design and validation of a Zero Bus Turnaround (ZBT) SRAM controller described in VHDL. ZBT SRAM memories allow consecutive read and write operations without introducing idle cycles between bus direction changes, thereby improving memory throughput and system performance.

Chapter 2

Project Objectives

The main objective of this practical work is to design a VHDL-based controller capable of interfacing with the Cypress MT55L512Y36F ZBT SRAM memory. The controller must correctly manage all control signals, as well as the address and bidirectional data buses, in accordance with the memory timing specifications. Special attention is given to the implementation of read and write operations without cycle loss, ensuring full compliance with the ZBT memory behavior.

To validate the proposed design, a complete simulation environment is developed. The implementation is written in VHDL, using the manufacturer-provided SRAM VHDL model for functional verification. Simulations are carried out using industry-standard tools such as Vivado, allowing the verification of correct functionality and timing behavior of the SRAM controller.

Chapter 3

ZBT SRAM Memory Overview

This section presents the main characteristics of ZBT SRAM memories and describes the hardware features of the Cypress MT55L512Y36F device used in this project. Understanding the internal behavior and control interface of the memory is essential for the correct design of the SRAM controller.

3.1 ZBT SRAM Principle

ZBT SRAM (Zero Bus Turnaround Static Random-Access Memory) allows direct transitions between read and write operations without requiring an idle or turnaround cycle on the bidirectional data bus. Unlike conventional synchronous SRAMs, where a dead cycle is needed to prevent bus contention when switching bus direction, ZBT SRAMs are designed to support consecutive read and write operations in adjacent clock cycles.

This behavior, illustrated in the timing diagram of FIGURE 1, enables a write operation to immediately follow a read operation and vice versa, without any loss of clock cycles. The Zero Bus Turnaround feature significantly improves memory bandwidth and reduces access latency, provided that the memory controller strictly respects the required read/write and output enable timing constraints.

3.2 MT55L512Y36F Memory Description

The memory used in this project is the Cypress MT55L512Y36F, a synchronous ZBT SRAM designed for high-performance applications.

The device has a large memory capacity with a 36-bit wide data bus, allowing high-throughput data transfers. Memory is accessed through an external address bus, which is used to select the memory location for read or write operations. Address generation is performed externally to the controller, as specified in the TP requirements.

The data bus is bidirectional, meaning that it is used both for data input during write operations and data output during read operations. As a consequence, special care must be taken to avoid bus contention. In this project, this is handled using tri-state buffers, allowing the controller to correctly control the direction of the data flow.

The main control signals of the MT55L512Y36F memory include:

Signal	Description
CKE	Clock enable signal for synchronous operation.
CE	Chip enable signal, used to activate the memory.
OE	Output enable signal, which controls data output during read operations.
R/W	Read/Write control signal, selecting the operation type.
ADV/LD	Address advance/load signal, used to control address loading.
CE2 and $\overline{\text{CE2}}$	Additional chip enable signals required for proper memory activation.

Table 3.1: Memory Control Signal Descriptions

Correct management of these control signals is critical to ensuring reliable memory access and maintaining the ZBT behavior of the device.

3.3 Controller Specifications

This section defines the main functional specifications of the SRAM controller developed for the MT55L512Y36F ZBT SRAM. The controller operates synchronously with the system clock and interfaces directly with the SRAM memory, while address generation is handled externally. Its role is therefore limited to managing the control signals and the bidirectional data path.

The controller supports two input commands, READ and WRITE, which initiate memory access operations. In case both signals are asserted simultaneously, read operations are given priority, ensuring deterministic and unambiguous behavior. This priority mechanism is a key design characteristic of the controller.

To simplify the design and ensure stable operation, several memory control signals (nBWA, nBWB, nBWC, nBWD, ZZ, MODE, LBOn, and nCKE) are fixed to constant logic levels, thereby disabling unused memory features.

A critical feature of the controller is the safe management of the bidirectional data bus. The controller drives the data bus during write operations and releases it during read operations, allowing the memory to provide output data. This approach prevents bus contention and ensures correct Zero Bus Turnaround operation.

3.4 VHDL Architecture of the Controller

This section presents the internal architecture of the SRAM controller and the key VHDL design choices made to ensure correct timing, reliable memory control, and compliance with ZBT SRAM operating principles.

3.4.1 Global Architecture

The SRAM controller is implemented as a synchronous digital system driven by a single clock signal. Its architecture can be divided into three main functional blocks:

- **Control logic**: Responsible for generating the memory control signals based on the READ and WRITE requests.
- **Address interface**: Which forwards the externally generated address to the SRAM memory.
- **Data interface**: Which manages the bidirectional data bus between the controller and the memory.

A block diagram representation of the controller highlights the interaction between these blocks and the SRAM memory. The control logic supervises the operation mode (read or write), while the address and data interfaces ensure proper communication with the memory device.

3.4.2 Control Signal Management

The control logic generates all required SRAM control signals, including chip enable, output enable, read/write selection, and address load signals. These signals are generated synchronously with the clock to respect the timing constraints of the ZBT SRAM.

During a write operation, the controller asserts the appropriate chip enable signals, sets the R/W signal to indicate a write cycle, and disables the memory output drivers. The data to be written is driven onto the data bus by the controller.

During a read operation, the controller sets the R/W signal to indicate a read cycle and enables the memory output drivers through the OE signal. In this case, the controller places the data bus in a high-impedance state and samples the data provided by the memory.

The priority mechanism ensures that if both READ and WRITE requests are active, the controller always performs the read operation.

3.4.3 Bidirectional Data Bus Handling

The bidirectional nature of the data bus requires careful handling to avoid contention between the controller and the memory. To address this, the data interface is implemented using IOBUF primitives from the Xilinx unisim library.

Each data bit is connected to an IOBUF component, which allows the direction of the data flow to be dynamically controlled. During write operations, the IOBUF is configured to drive data from the controller to the memory. During read operations, the IOBUF is placed in tri-state mode, allowing the memory to drive the data bus while the controller captures the output.

This approach ensures safe and reliable operation of the bidirectional data bus while fully respecting the ZBT timing requirements.

3.5 VHDL Implementation

The SRAM controller is implemented in VHDL. The design is structured to ensure clarity, modularity, and compliance with the functional specifications defined in the previous sections.

The controller entity defines all external interfaces, including the system clock, the READ and WRITE control signals, the address bus, and the bidirectional data bus connected to the SRAM memory. In addition, all required memory control signals are declared as outputs of the entity.

The internal architecture relies on synchronous processes triggered on the rising edge of the clock. These processes are responsible for interpreting the READ and WRITE requests, applying the priority rule in favor of read operations, and generating the appropriate control signals for the SRAM memory.

The values of the control signal are determined on the basis of the current operating mode. For a write operation, the controller drives the data bus and asserts the write-related control signals. For a read operation, the controller releases the data bus and enables the memory output, allowing the data to be captured internally. Default signal assignments are used to ensure a safe idle state when no operation is requested.

The bidirectional data bus is implemented using IOBUF components. Each bit of the data bus is associated with one IOBUF instance, allowing independent tri-state control. The direction control signal of the IOBUF is derived directly from the READ and WRITE signals, ensuring the correct data flow during each operation.

To simplify the design and comply with the TP recommendations, unused memory features are disabled by fixing several control signals to constant logic levels. This choice reduces complexity and ensures the predictable behavior of the SRAM.

Overall, the VHDL implementation provides a clear and robust realization of the SRAM controller, suitable for functional simulation and timing analysis.

To ensure proper functioning of system unit tests and interventions were made to minimize error making and proper validation techniques.

3.5.1 Test of the SRAM

To validate the functionning of the SRAM to be sure that we could do a read and write as expected we tested simply the SRAM and obtained the result below:

We can clearly observe that the SRAM does a simultaneous write of 4 datas and read them after.The code use for this functionnal test is found in(Figure A.1)

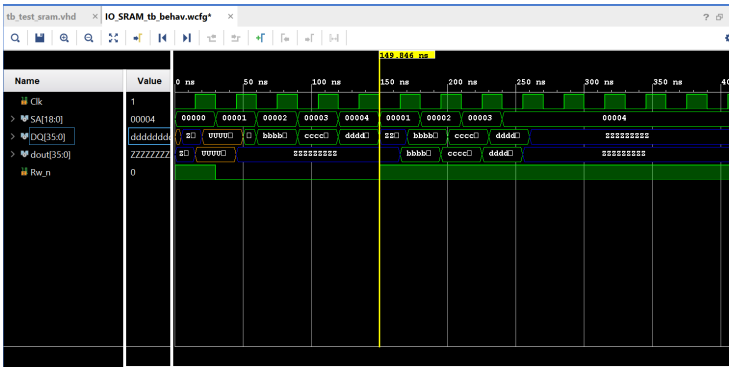

Figure 3.1: SRAM simple read and write

3.5.2 Controller Functioning

After initially performing direct read and write operations on the SRAM, a dedicated controller was designed using a finite state machine (FSM). This FSM describes all stages and possible states involved in the SRAM read and write processes. The controller is structured to prevent any idle conditions of the memory while ensuring a strict alternation between read and write operations, without any loss of clock cycles. The complete implementation of the controller is available on GitHub under the name **sram_ctrl1**.

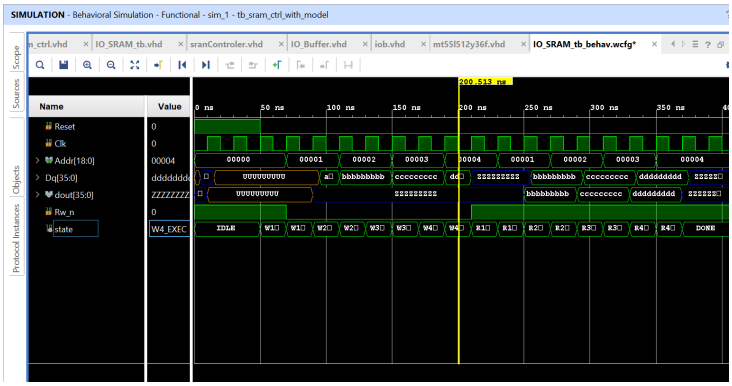

Figure 3.2: Simple read and write with data in FSM

This approach was limited by the fact that the data were directly embedded within the finite state machine, thereby restricting the user's ability to control and manipulate the data. To address this limitation, the design was revised by delegating data control to the user, allowing the data to be entered externally through a dedicated testbench, named **sram_ctrl2**, available on GitHub.

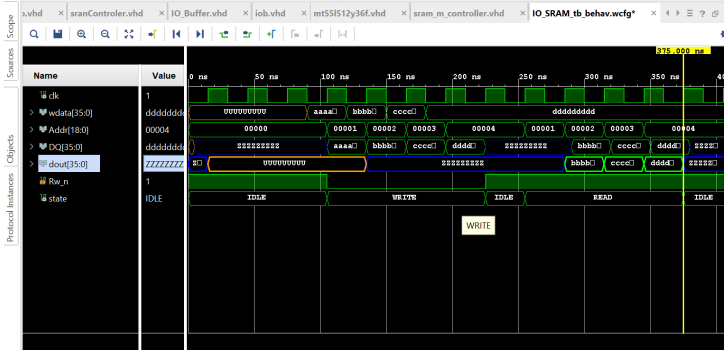

Figure 3.3: Simple read and write and manual data handle

3.5.3 Input Output Buffer (IOBUF)

An IOBUF is inserted to ensure the bidirectionality of the SRAM output.It is first tested individually to ensure its correct functionning. To write the trigger is drive to 0 then 1 to read as shown below:

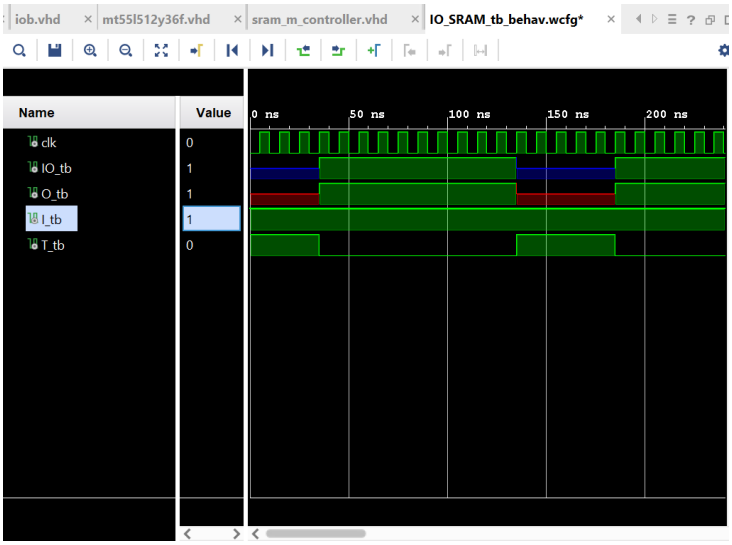

Figure 3.4: IOBUF Test

After this unit test we integrated the IOBUF into the FSM to validate a correct functionning.

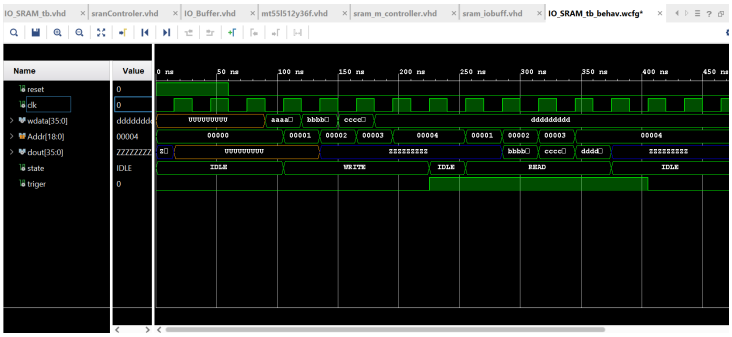

Figure 3.5: IOBUF intergrated into the FSM

3.6 Burst Mode Operation

3.6.1 Overview

Burst mode is used to transfer multiple consecutive data words to or from the SRAM using a single address phase. Instead of supplying a new address for each access, the SRAM internally increments the address using an internal burst counter. This reduces address bus activity and improves overall memory bandwidth.

The implemented controller supports *linear burst mode* as specified in the Micron MT55L ZBT SRAM datasheet. Burst operation is controlled using the `ADV/LD#` signal:

- `ADV/LD# = 0`: Loads the initial address into the SRAM
- `ADV/LD# = 1`: Enables the internal address counter to increment on each clock cycle

Both burst read and burst write operations follow the same address timing, differing only in data direction.

3.6.2 Burst Write Operation

Figure 3.6 illustrates the timing waveform of a burst write operation.

At the start of the burst, the controller places the initial address on the address bus and asserts `ADV/LD#` low for one clock cycle. During this cycle, the SRAM samples the address and the write command (`RW# = 0`) on the rising edge of the clock.

On the following clock cycle, `ADV/LD#` is deasserted high, enabling the internal burst counter. The first data word is driven onto the `DQ` bus by the controller and is sampled by the SRAM on the rising edge of the clock. Each subsequent clock cycle transfers a new data word while the SRAM automatically increments the internal address.

The controller does not buffer burst data internally. Instead, the user provides a new data value on every clock cycle during the burst. The burst continues as long as the burst enable signal remains asserted. When the burst signal is deasserted, the controller releases the data bus and returns to the idle state.

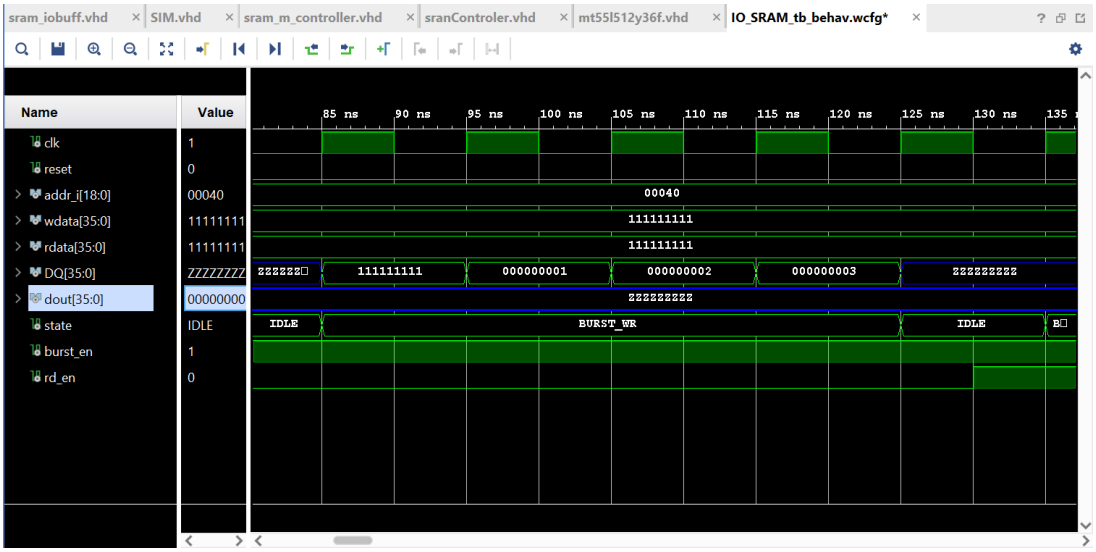

Figure 3.6: Burst write timing waveform showing address load, internal address increment, and consecutive data transfers

3.6.3 Burst Read Operation

Figure 3.7 shows the timing waveform of a burst read operation.

As with burst write, the operation begins with an address phase. The controller drives the initial address and asserts `ADV/LD#` low for one clock cycle while setting `RW# = 1`. The SRAM registers the address and read command on the rising clock edge.

During the next clock cycle, `ADV/LD#` is set high to enable the internal address counter. The first data word becomes valid on the `DQ` bus one clock cycle after the address phase, consistent with zero bus turnaround (ZBT) timing requirements. The controller samples the read data on each rising edge of the clock.

For the duration of the burst, the SRAM outputs a new data word on every clock cycle while incrementing the internal address automatically. No additional address updates are required from the controller. When the burst enable signal is deasserted, the controller disables the output enable signal and transitions back to the idle state.

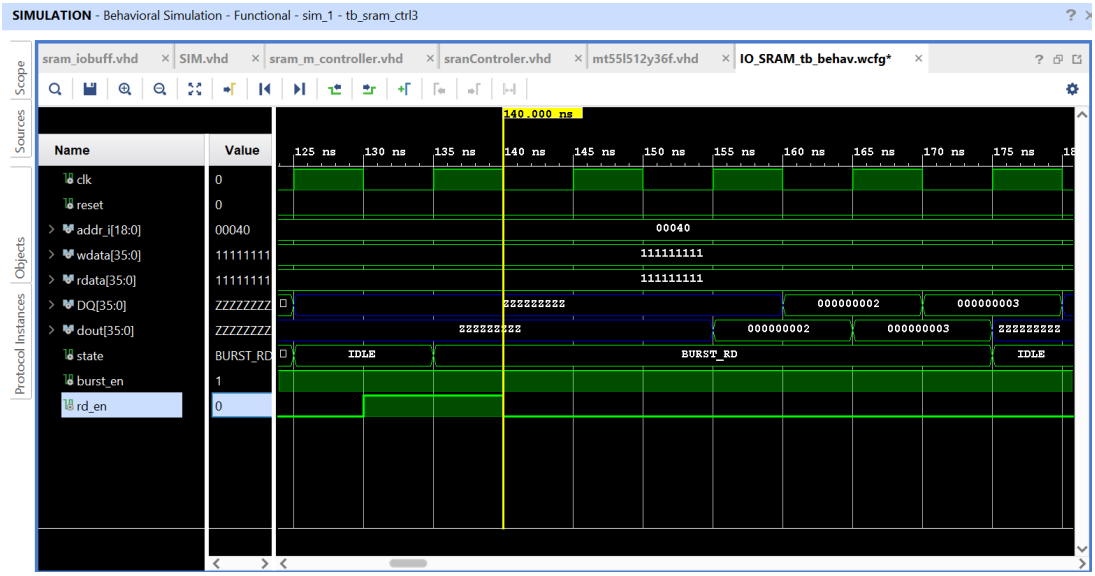

Figure 3.7: Burst read timing waveform illustrating address phase latency and continuous data output from sequential addresses

3.6.4 Burst Mode Timing Summary

Burst mode operation follows these key principles:

- The start address is loaded only once at the beginning of the burst
- The first data word is transferred one clock cycle after the address phase
- Subsequent data words are transferred on every clock cycle
- Address incrementation is handled internally by the SRAM
- No idle cycles are required between consecutive burst data transfers

This timing behavior complies with the Micron MT55L ZBT SRAM specification while providing high-throughput memory access.

3.7 Timing Constraints and Performance

This section analyzes the timing behavior of the SRAM controller, applying timing constraints to determine the maximum achievable operating frequency and identify performance bottlenecks.

3.7.1 Maximum Operating Frequency

Post-synthesis and post-implementation timing analyses are performed to determine the maximum achievable operating frequency of the controller. The critical path is typically located within the control logic responsible for generating the SRAM control signals and managing the read/write arbitration.

Based on the timing reports, the maximum operating frequency corresponds to the inverse of the longest combinational delay between two sequential elements. This frequency represents the highest clock rate at which the controller can operate while still meeting all timing constraints.

3.7.2 Optimization Considerations

To improve the maximum operating frequency, several optimization strategies can be considered. These include simplifying combinational logic, reducing the depth of decision paths in the control logic, and ensuring efficient use of synchronous processes.

Additionally, fixing unused memory control signals to constant values helps reduce logic complexity and contributes to improved timing performance. Overall, the optimized controller achieves reliable operation while respecting the ZBT SRAM timing requirements.

3.8 Conclusion

In this laboratory work, a ZBT SRAM controller was successfully designed and validated using the VHDL hardware description language. The controller was developed to interface with the Cypress MT55L512Y36F memory and to manage all required control, address, and data signals in accordance with the memory specifications.

The controller supports both read and write operations, with a defined priority given to read requests. Particular attention was paid to the management of the bidirectional data bus, which was safely handled using tri-state buffers implemented with Xilinx IOBUF primitives. Simulation results confirmed correct functional behavior, including the absence of idle cycles between consecutive read and write operations, thus validating the Zero Bus Turnaround property of the memory.

Timing analysis was performed to evaluate the maximum operating frequency of the controller and to identify potential optimization opportunities. The applied constraints and design choices ensured reliable synchronous operation while maintaining good performance.

Overall, the objectives of the TP were fully achieved. The developed controller provides a clear and efficient solution for interfacing with a ZBT SRAM memory and constitutes a solid basis for further extensions, such as sequential memory access or integration into a larger SoC architecture.

Appendix A

appendix

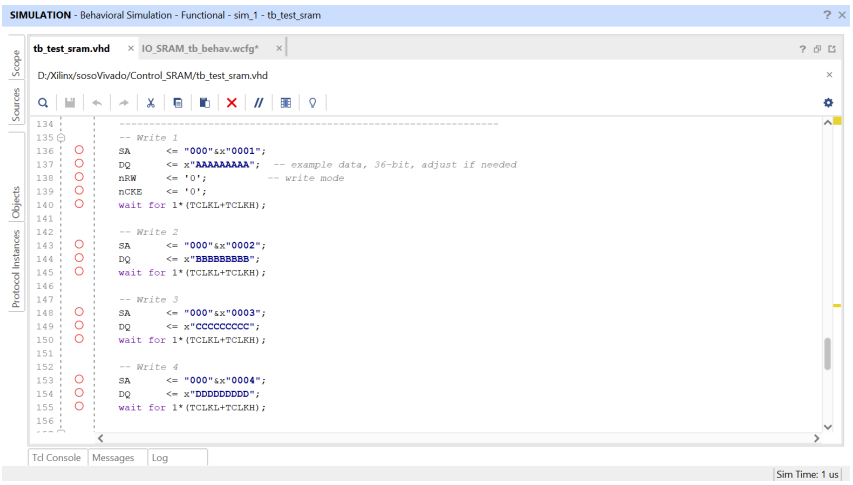

Figure A.1: SRAM test bench simple read and write code