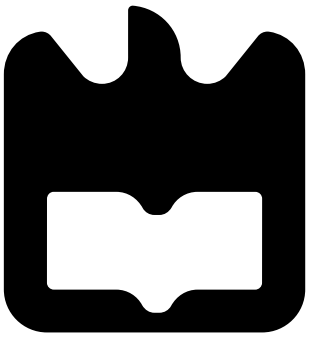

AAD - Assignment 2

107474-Joseane Pereira
109050-Gabriel Costa
Universidade de Aveiro, DETI

January 4, 2025

Contents

1 Introduction

This assignment focuses on the implementation and analysis of different accumulator designs in VHDL, including a single-cycle version, a pipelined version, and versions with a shift functionality.

2 Implementations

2.1 Single-Cycle Accumulator

As suggested with the provided starting point, the basic accumulator was implemented in `accumulator_single_cycle.vhd` with the following features:

- Simple write and read ports
- Basic addition functionality
- Synchronous reset to initialize the accumulator
- Parameterizable data width for flexibility

The accumulator operates by reading an input value, adding it to the current value stored in the **ram** at the address **write_addr**, and writing the resulting sum back to the storage register, all within a single clock cycle.

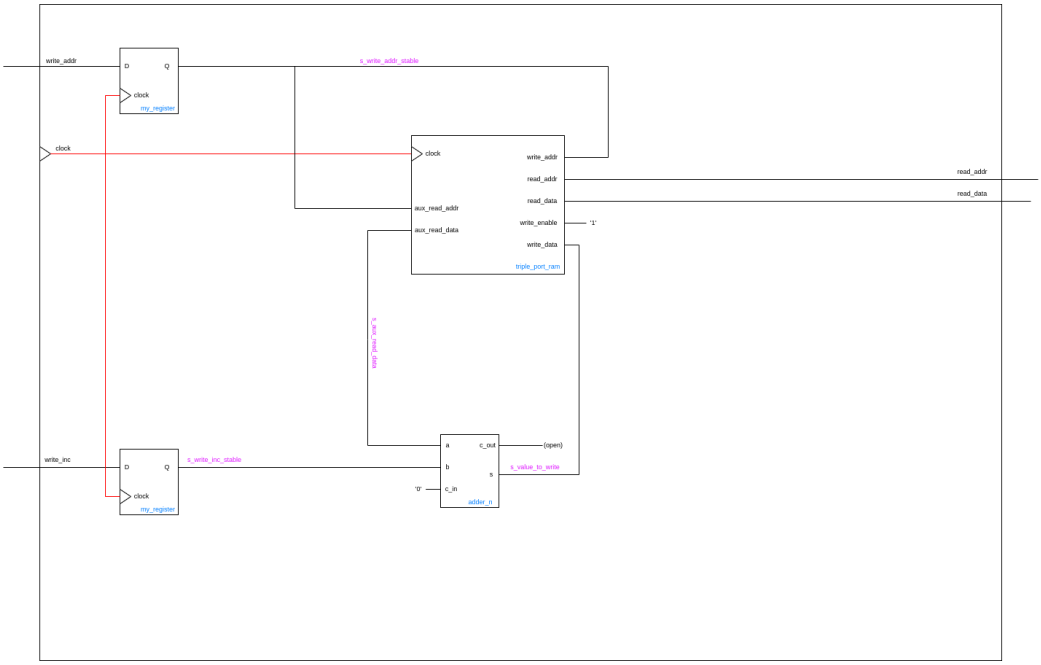

Figure 1: Single-cycle accumulator block diagram

2.2 Pipelined Accumulator

The pipelined version, `accumulator_pipeline.vhd`, requires two clock cycles to complete the objective.

- **First clock cycle:** The module reads the value stored in the specified memory location.
- **Second clock cycle:** It computes the sum of the input value and the previously read value, then writes the result back to the memory.

If the next input value needs to be added to the same **write_addr** used in the previous addition, the output from the previous clock cycle (**s_value_to_write**) is directly reused as the starting value for the new increment operation, as if it were read from memory. This eliminates the need to wait an additional clock cycle to get the updated value from memory.

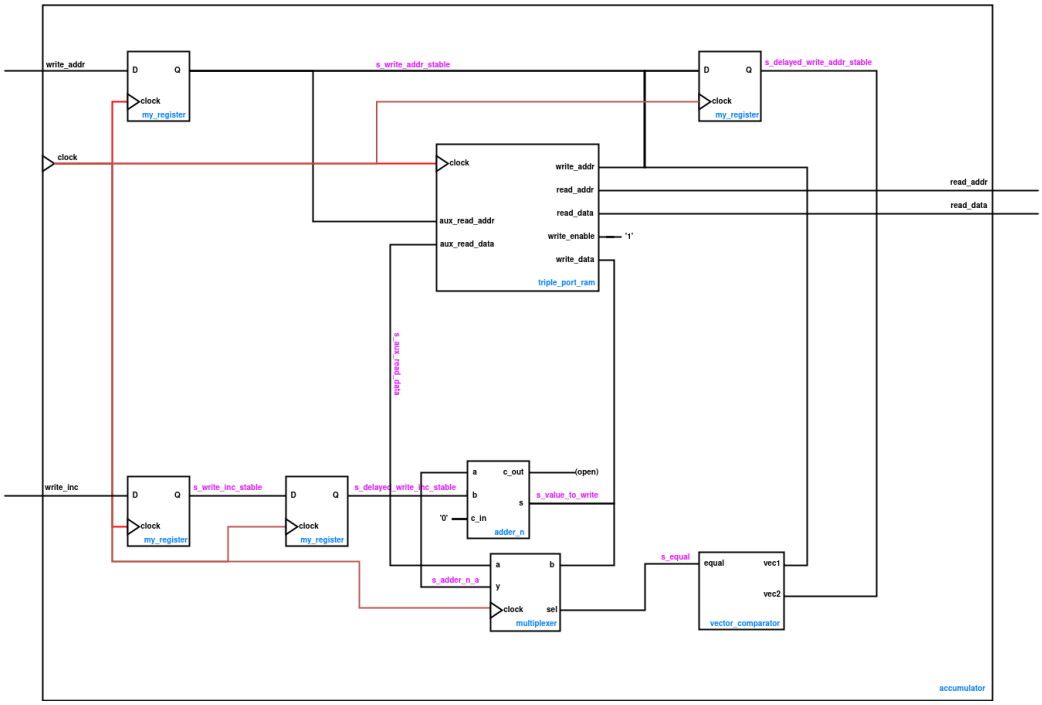

Figure 2: Pipelined accumulator block diagram

2.3 Shift Accumulator

We implemented a version of the accumulator using a barrel shifter for both the pipelined and single cycle versions (`shift_accumulator_pipeline.vhd` and `shift_accumulator.vhd`), both of which improve upon the basic design by:

- Dynamic shift operations on input data
- Configurable shift amounts
- Integration with the accumulation logic

The shift accumulator design leverages a barrel shifter to perform dynamic shift operations on the input data.

The accumulator operates by reading an input value, adding it to the current value stored in the ram at the address **write_addr**, and writing the resulting sum back to the storage register, all within a single clock cycle.

In the single cycle version, the shift and accumulation operations are performed within a single clock cycle. The **write_inc** is shifted a **write_shift**

amount and it's value is added to the current value stored in the **write_addr** address. The result is then stored back to the same register. The block diagram in Figure 3 illustrates the architecture of the single cycle shift accumulator.

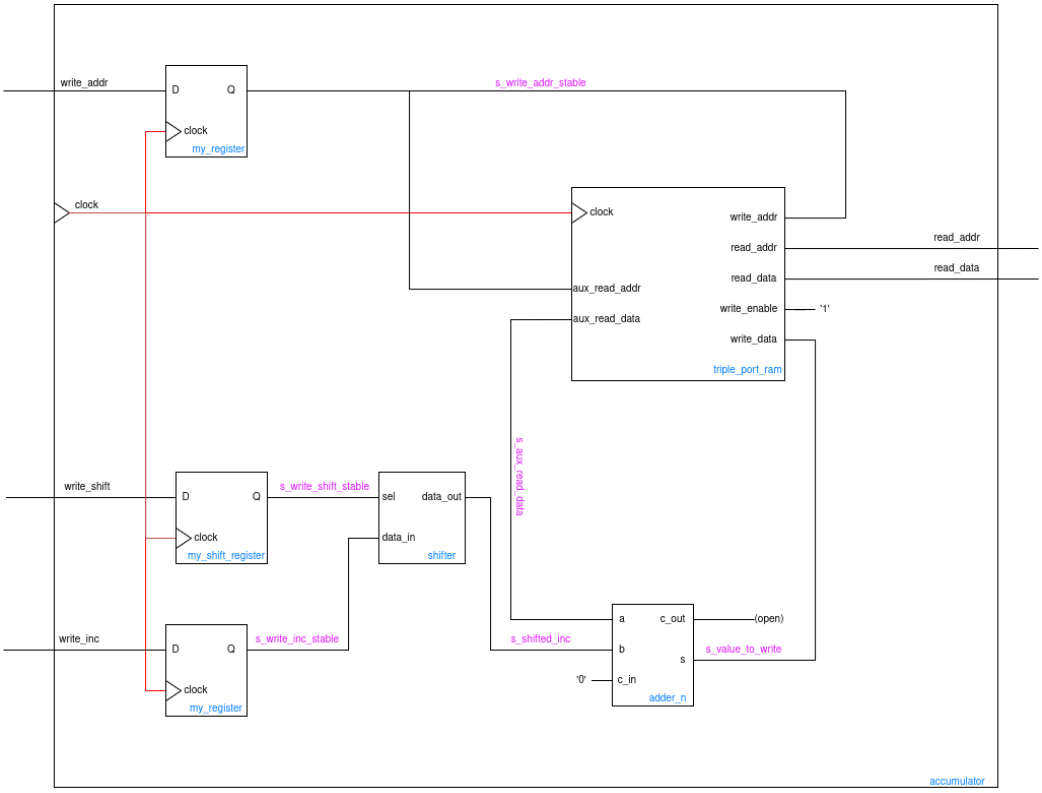

Figure 3: Shift accumulator single cycle block diagram

For the pipelined version, the design is divided into multiple stages, allowing for higher clock frequencies and improved throughput. Each stage of the pipeline performs a portion of the shift and accumulation operations, enabling the design to handle higher data rates. This is particularly beneficial in applications where large volumes of data need to be processed efficiently. The block diagram in Figure 4 shows the architecture of the pipelined shift accumulator.

And similarly to the non shift version, the pipelined version, this time in `accumulator_shift_pipeline.vhd`, requires two clock cycles to complete the objective.

- **First clock cycle:** The module reads the value stored in the specified

memory location.

- **Second clock cycle:** It computes the sum of the input value and the previously read value, then writes the result back to the memory, only this time the sum is shifted by the amount specified in the **write_shift** input.

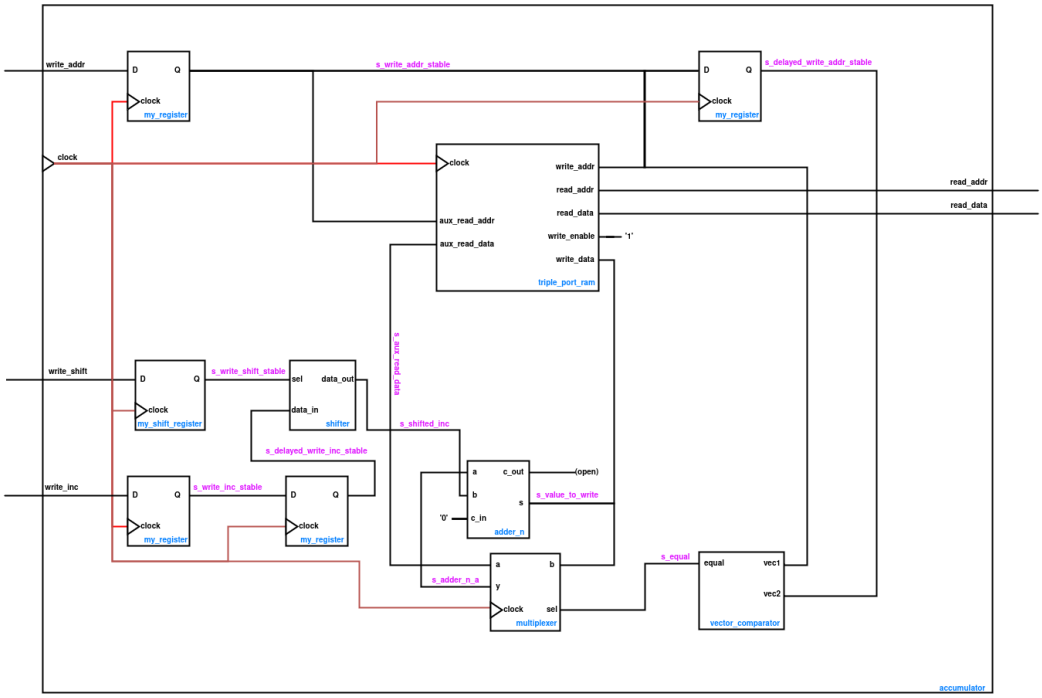

Figure 4: Shift accumulator pipeline block diagram

Overall, the shift accumulator designs provide a robust and flexible solution for data manipulation and accumulation tasks, with the pipelined version offering enhanced performance for high-speed applications.

3 Components

Some of the Components were provided by the professor, and some were implemented by us. Key components used in the designs include:

- **Triple Port RAM** - for memory storage
- **Registers** - for signal stabilization

- **Adder N** - for arithmetic operations
- **Shifter** - for shift operations
- **Vector Comparator** - for address comparison

Note that the **aux_read** port from **Triple Port RAM** is assyncronous in all implementations.

4 Testing

Testing was performed using:

- GHDL simulator
- VCD waveform generation
- A provided testbench

5 Results and Analysis

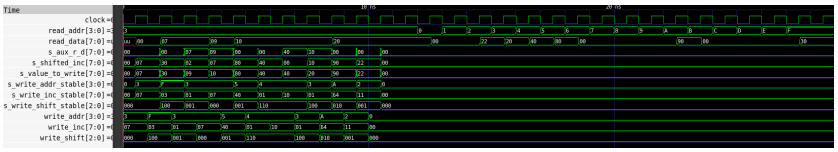

Figure 5: Waveform of the single-cycle accumulator testbench

5 shows the waveform generated by the testbench for the single-cycle accumulator. The testbench was adapted to the different versions and the results were similar. The shift accumulator was tested with different shift amounts, and the results were as expected. For both Accumulators the smallestworking clock period observed was 10ps.

6 Conclusions

This project demonstrated the design and analysis of various accumulator architectures in VHDL: single-cycle, pipelined, and shift-based. The single-cycle design offered simplicity and minimal latency but was limited in scalability for high-speed operations. The pipelined accumulator addressed this

by improving throughput and enabling higher clock frequencies. The shift accumulator added flexibility with dynamic shift operations, making it suitable for data-intensive applications.

Testing validated the functionality of all designs, with each meeting expected performance criteria. Overall, the project highlighted the trade-offs between simplicity, speed, and flexibility in digital system design. Future improvements could focus on optimizing power and area efficiency while exploring additional features.