AAD Project 2

Tiago Santos 89356, Leandro Rito 92975

January 5, 2025

1 Single Cycle

This project involved creating a VHDL description of a sequential logic circuit that performs operations based on a C function. The function reads a value from a 16-element array using a read address, increments the array element at a specified write address, and returns the original read value.

The design implements the single-cycle version, where the read, addition, and write operations occur within the same clock cycle. The main components used in the VHDL implementation are:

1. **Accumulator**: Handles the overall operation by interacting with internal registers;
2. **Triple Port RAM**: Stores the data array and facilitates read and write operations;
3. **Vector Register**: Holds the stable values of the write address and write increment.
4. **Adder**: Performs the addition operation for updating the array elements.

1.1 Accumulator

The accumulator handles the overall operation by interacting with internal registers by coordinating the read, add, and write stages of the implementation. It utilizes ports for the clock input and the ports write_addr, write_inc, read_data and read_addr. Its architecture consists of the registers addr_reg

and addr_inc, for the write address and write increment respectably, using the Vector Register component, ram, for the triple port ram, using the Triple Port RAM component and adder, for the addition, using the Adder component.

1.2 Triple Port RAM

The triple port ram entity implements a memory array with a write port and two read ports. It manages data writing and reading operations through these ports. The architecture consists of a RAM array initialized to zero, with processes to update it on the clock's rising edge and provide data from the specified addresses.

1.3 Vector Register

The vector_register entity is a basic register that stores stable values for the write address and increment. It has ports for clock input, data input, and data output. The process within the entity assigns input data to the output on the rising edge of the clock, simulating data transfer with a slight delay.

1.4 Adder

The adder entity is a generic N-bit adder using a structural approach. It instantiates multiple full_adder components within a loop to handle each bit of the input operands, ensuring proper carry propagation between them.

1.5 Results

As a result we have the following results:

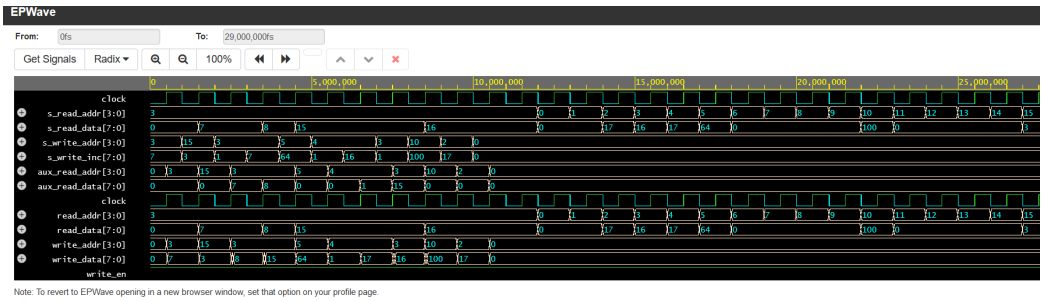

Figure 1: Results from the Single Cycle implementation

The results show the ports present in the Triple Port RAM component, and in the overall Accumulator. By observing the results, we can observe that initially, the read_data value is incremented by the value 7, present in the s_write_inc, resulting in read_data being given the value 7. Following the next clock cycle, the value is incremented by 1, resulting in the value 8. This shows that the implementation is working as instead.

2 Multi Cycle

This section introduces the concept of multi-cycle processing into our implementation. The task was to implement a multi-cycle design for the read, addition and write operations, in order for them to work in a combined three clock cycle, being one clock cycle dedicated for each of the operations.

In order to do so, we implemented a finite state machine within the accumulator, comprised of four states: IDLE, READ, ADD and WRITE. These states transition on every rising edge of the clock, with each clock cycle corresponding to a different operation. The state sequence begins with IDLE, then follows READ, ADD, WRITE before returning to READ, repeating the cycle. IDLE would only be used in the initial state and in abnormal situations.

As a result we have the following results:

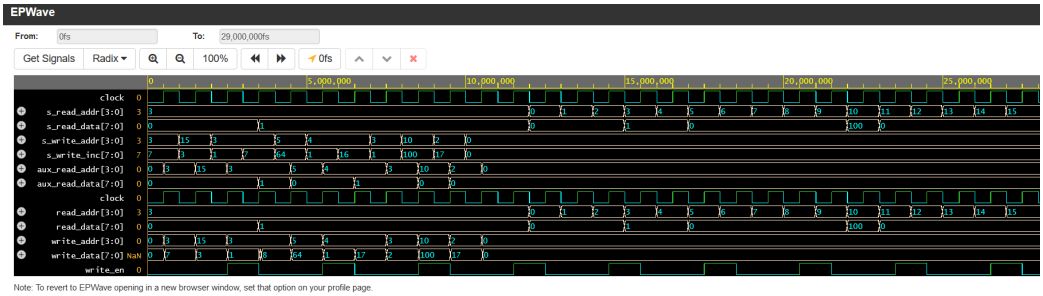

Figure 2: Results from the Multi Cycle implementation

As it can be observed, the writing is done in following the triggering of the write_en, leading to the incrementation of 1 into the read_data, which is the value select in the previous clock cycle, where the ADD state was active. This way we can conclude that it is working has expected.