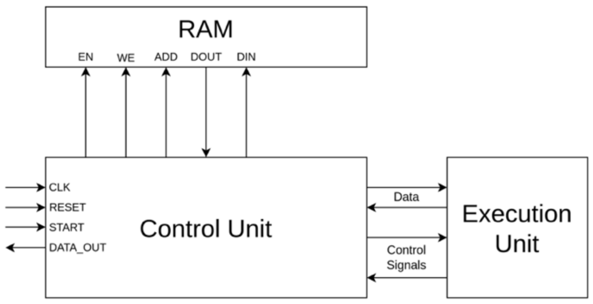
Samuele Pasquale – 328744

Assignment
Specification and Simulation of Digital System

Introduction

Given the circuit shown below, the objective of the assignment is to implement the design of the Control Unit and the Execution Unit for finding the longest increasing sequence stored in the RAM memory and to save the starting address and the length in the last two locations of the RAM memory.



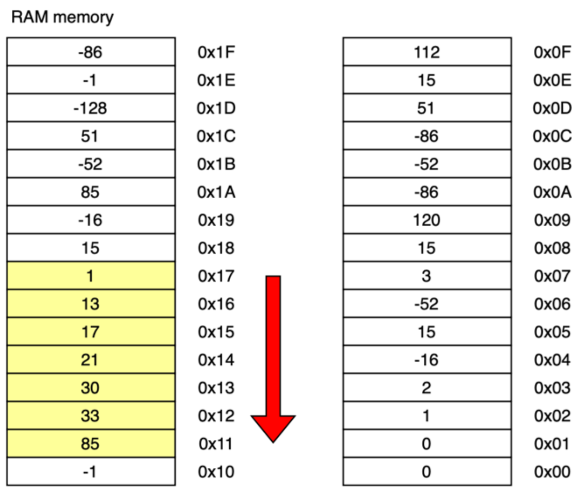
In the Vivado project, you can find the files listed below. In this report, I will describe the project structure, the Control Unit, the Execution Unit, the RAM memory, and the respective testbenches.

- SequenceDetector.vhd
- ControlUnit.vhd
- ExecutionUnit.vhd

- TestbenchSequenceDetector.vhd
- TestbenchControlUnit.vhd
- TestbenchRAM.vhd

RAM memory

The memory is a
RAM with 32 words, 8 bits wide. In the initial phase of the project, I created an initialization file that defines the content of the memory. The state of the RAM memory is shown in the image below. As you can see, the RAM memory contains more than one increasing sequence, but the longest one starts from address 0x17 and has a length of 7 words.

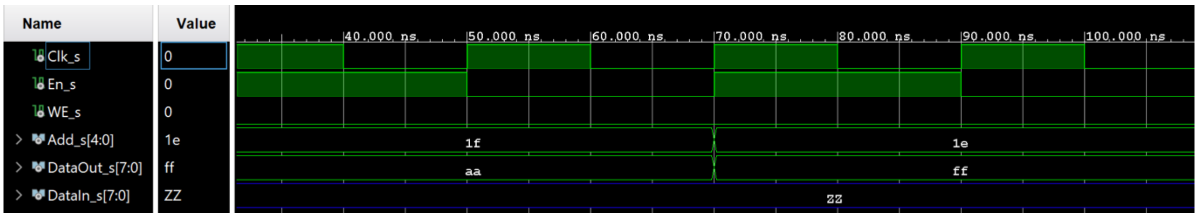


RAM memory

-86	0x1F	112	0x0F
-1	0x1E	15	0x0E
-128	0x1D	51	0x0D
51	0x1C	-86	0x0C
-52	0x1B	-52	0x0B
85	0x1A	-86	0x0A
-16	0x19	120	0x09
15	0x18	15	0x08
1	0x17	3	0x07
13	0x16	-52	0x06
17	0x15	15	0x05
21	0x14	-16	0x04
30	0x13	2	0x03
33	0x12	1	0x02
85	0x11	0	0x01
-1	0x10	0	0x00

RAM memory content, longest increasing sequence

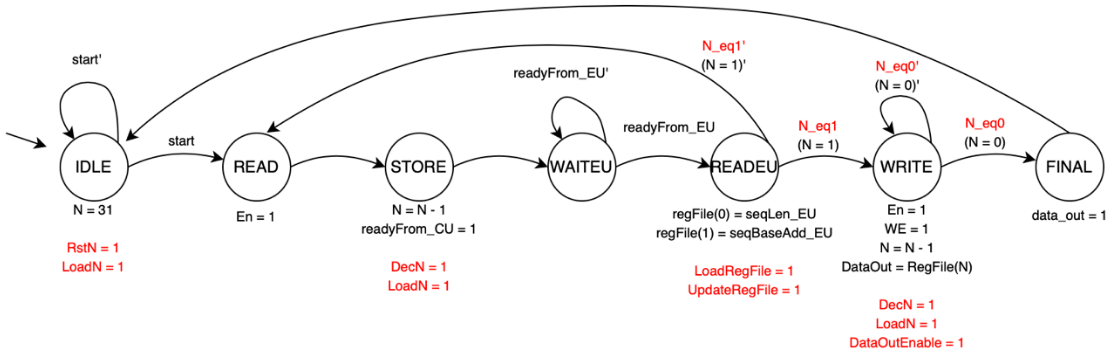
Before starting with the design, I implemented a testbench to verify the correct loading of the memory and simulate read and write cycles. As shown in the waveform below, once the enable is activated and the write enable is set low, the RAM memory will output the value read from the provided address driven onto the address bus. For simplicity, the image below shows only the first two read cycles of the locations at addresses 0x1F and 0x1E.



Waveform simulation RAM memory

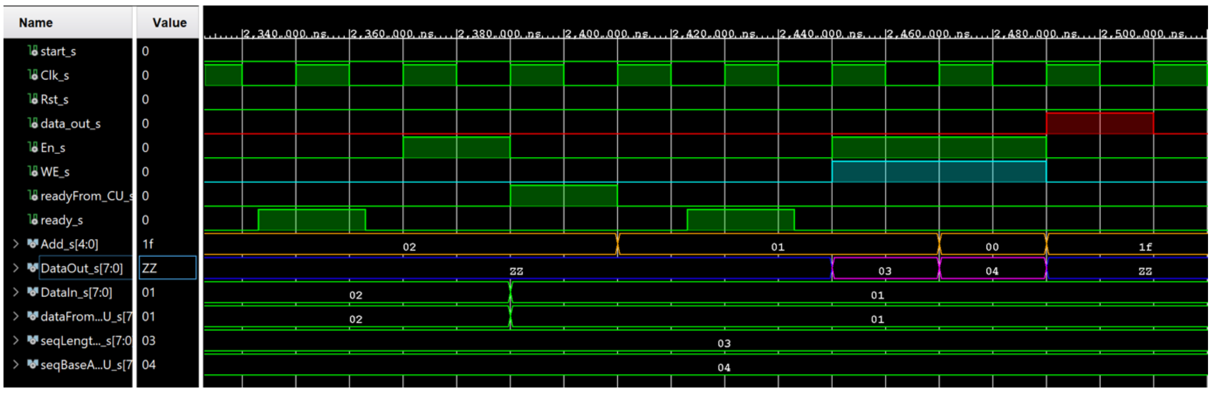
Control Unit

For the design of the Control Unit, I implemented two different architectures, one for the HLSM and the other for the FSM-D. Both architectures are available in the Vivado project, and their operations are shown by the state diagram below. The purpose of the Control Unit is to act as a bridge between the RAM memory and the Execution Unit. In particular, it has been implemented to interface with the RAM memory and perform both read and write cycles. In an initial phase, it reads the memory sequentially, and the value of each memory location is sent, along with its address, to the Execution Unit. Internally, a register file made of only two 8-bit registers has been implemented, containing respectively the address and the length of the current longest increasing sequence. These two information are updated with the result of the Execution Unit's computation. In this way, at the end of the RAM memory read, the register file will contain the final result, which needs to be written to RAM memory into the last two memory locations. Therefore, we proceed with writing to the RAM memory and activate the data_out signal at the end of the write operation. In the state diagram image, you can also see how these operations were subsequently mapped with control signals for the implementation of the FSM-D.



State diagram Control Unit

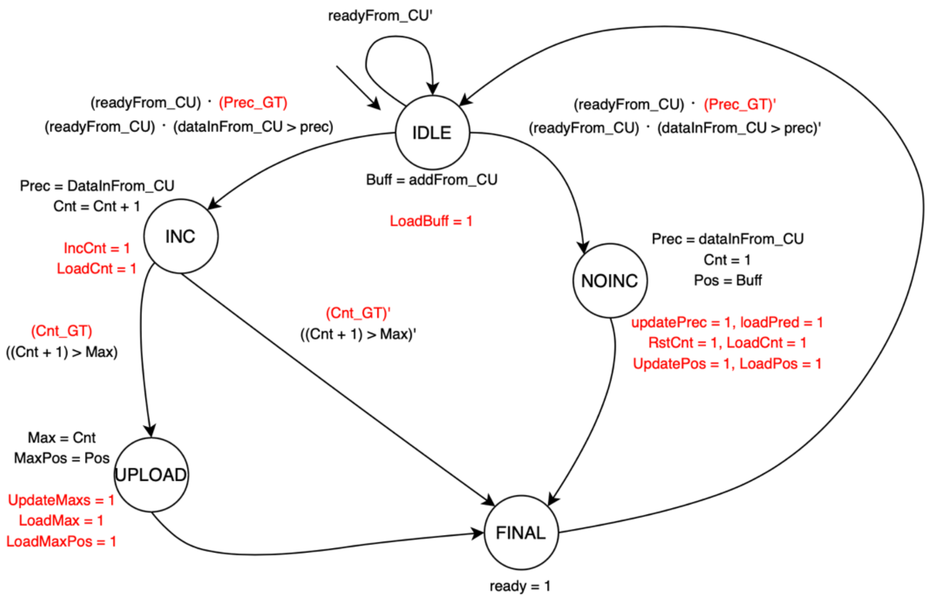
In the testbench implemented for the Control Unit, it is verified that it is capable of correctly reading the memory and writing the final result. To simulate stimuli from the Execution Unit, the ready signal from the Execution Unit is activated with a fixed period, and the value 0x4 is driven as the address and 3 as the length of the longest increasing sequence for test purposes. The image below shows the last read cycles and the two write cycles in RAM memory. During these two clock cycles, the enable signal remains active, and the data and address are driven to the RAM memory to save the final result. Except during the write phase, the data line connected to the RAM memory assumes a high-impedance value. At the end, the Control Unit sets high the data_out signal for one clock cycle.



Waveform simulation Control Unit: data_out signal Red, Address signal Orange, Final result Magenta, Write enable Aqua

Execution Unit

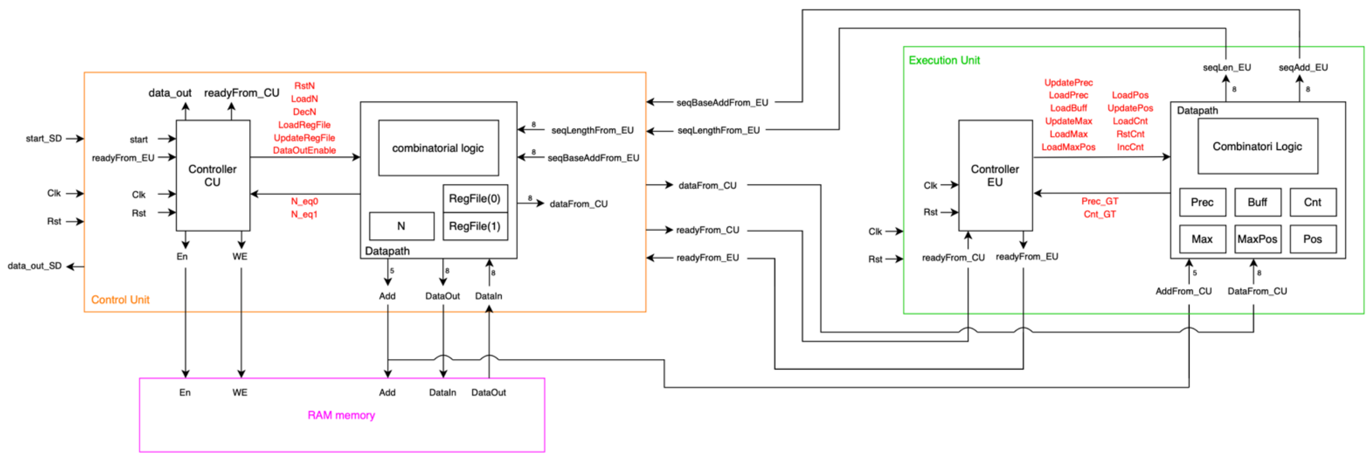
For the design of the Execution Unit, I initially implemented an HLSM and subsequently the architecture of the FSM-D. In the image below, I show the state diagram with the mapped control signals. The implemented algorithm receives an address, a signed value from the Control Unit, and a ready signal. As soon as the ready signal is activated, the address value is saved in a temporary buffer, while the input data is used to check if it is greater than the previous one. During reset, the previous value, saved into a register, is set to the maximum possible value to force the state NOINC during the first comparison of the first value read from memory. In all subsequent reads, the sent data will be compared with the previous one. If the data is greater, then the counter that counts the number of elements of the increasing sequence is incremented by one in the INC state. If it is less or equal than the previous value, an increasing sequence has been interrupted, and the counter is reset to one. Each time the counter is incremented, the Execution Unit checks if it is a new maximum. If so, the value of the maximum counter and the starting address are updated in the UPDATE state. In this way, each time the verification is completed and the Execution Unit activates the ready signal, the updated address and length are provided for the values read from memory up to that point, regardless of how many still need to be read. Therefore, the position and address for the maximum and temporary values are saved in four registers: max, posMax, cnt, and pos. Finally, each time a new increasing sequence begins, the Execution Unit enters the NOINC state and updates the value of pos. The latter is updated only in this case, to ensure that the starting address of the increasing sequence is saved instead of an intermediate address. At the end of the comparisons, the Execution Unit activates the ready signal and updates the prec value with the input that has just been received.



State diagram Execution Unit

Final Circuit

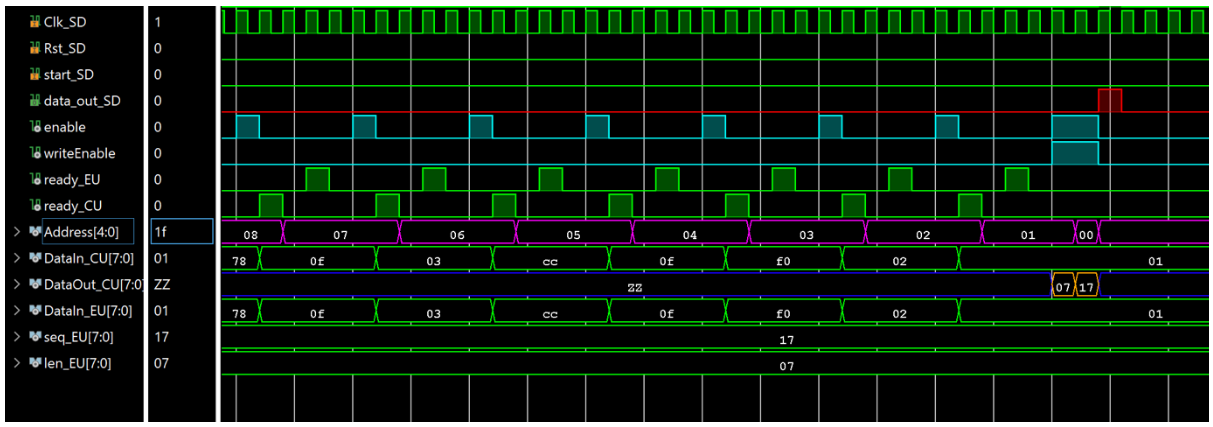
In the image below, I show a high-level abstraction diagram with the two FSM-Ds and their respective control signals.



Final circuit structure connection

In the end, a testbench was implemented to verify the correctness of the entire system. As can be seen during execution, there are three successive enable signals: the first one, called enable, is activated for reading and writing to memory. The second one, called ready_CU, is activated by the Control Unit once data and address are driven into the Execution Unit. The ready_EU signal is activated with a certain delay after ready_CU due to the computation times of max and posMax by the Execution Unit.

Finally, at the end of the processing, there are two clock cycles for writing to memory at addresses 0x1 and 0x0 the values of the length of the longest increasing sequence (7) and the starting address (0x17) respectively.



Waveform simulation full circuit: data_out signal Red, RAM control signals Aqua, Address signal Magenta, Final result Orange