Politecnico di Milano

Digital Logic Final Project

Authors:
Edoardo Gennaretti (CP: 10743751, NM: 955326),
Samuele Pietro Galli (CP: 10710025, NM: 955426)

Supervisor:
Prof. Fabio Salice

20 maggio 2023

Indice

1 Specs

1.1 Project Description

The objective of the project is to realize a HW component that, having received as input a memory address and information regarding the required output channel, prints the contents of the address on the specified channel.

Seven interfaces are presented, including 2 primary inputs (W and START), both 1 bit, and 5 outputs (Z0, Z1, Z2, Z3, DONE), of which, the first 4 (8 bits), on which all bits of the memory word are to be reported, and DONE 1 bit. There is also a reset signal (RESET) and a clock signal (CLK), unique to the component.

The specification calls for implementing a hardware module in VHDL that interfaces with a memory and receives information via a one-bit serial input about a memory location whose contents are to be routed to one of four available output channels.

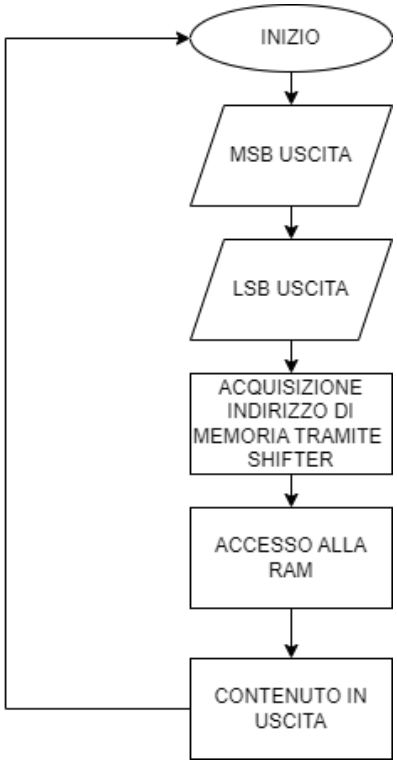
The module must be able to read the sequence on the primary serial input W, when the START signal is '1'. The sequence is organized into 2 header bits, followed by N bits ($0 \leqslant N \leqslant 16$). The two header bits identify the output channel on which the message is to be addressed. The N bits reconstruct the memory address always consisting of 16 bits. The START signal remains active for at least 2 clock cycles and no more than 18 clock cycles.

Outputs Z0, Z1, Z2, and Z3 are initialized to zero, and remain at that value as long as the DONE signal remains low. When DONE becomes 1, their value changes, showing the data just read from RAM on the associated channel, while the last saved value is retained on all other outputs. The DONE signal remains active for only one clock cycle, after which it goes low again.

1.2 Description of design choices

The description of the HW component involves the use of 14 signals, and operation is divided into three main phases:

- **OUTPUT SELECTION**: Initial step in which the descriptive bits of the output channel are acquired.
- **TRANSCRIPTION OF MEMORY ADDRESS**: Second stage in which the N bits of the address are received in a register. Storage takes place with a shift to the left of all the address acquired up to that time, and of the storage of the W signal at the least significant position. In this phase the correction of the reception also takes place in that the bit sequence is "shifted" one position too far to the left.
- **CONCLUSION AND MESSAGE PRESENTATION**: Final stage in which the message is routed and presented on the output corresponding to the one specified.



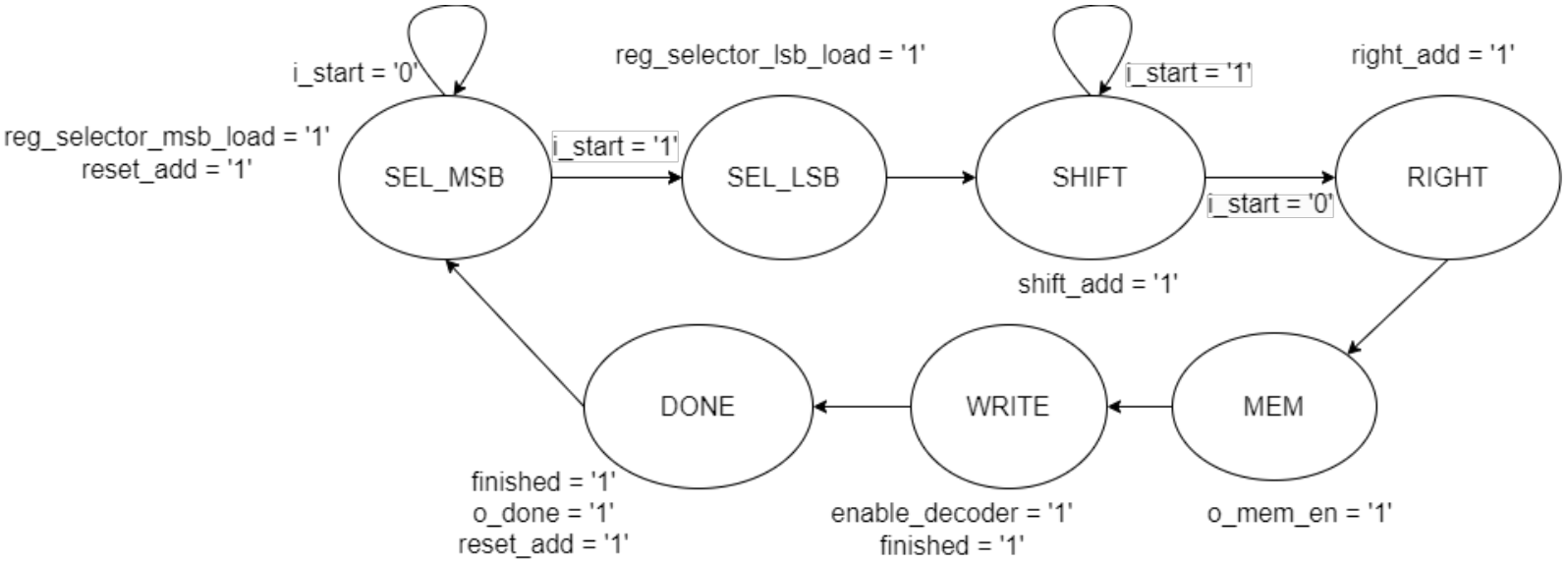
2 Architecture

A simplified version of the FSM (Final State Machine) is shown below; all arcs that return to **SEL_MSB** from any state are omitted when **i_rst='1'**. All unspecified signals are set to '0'.

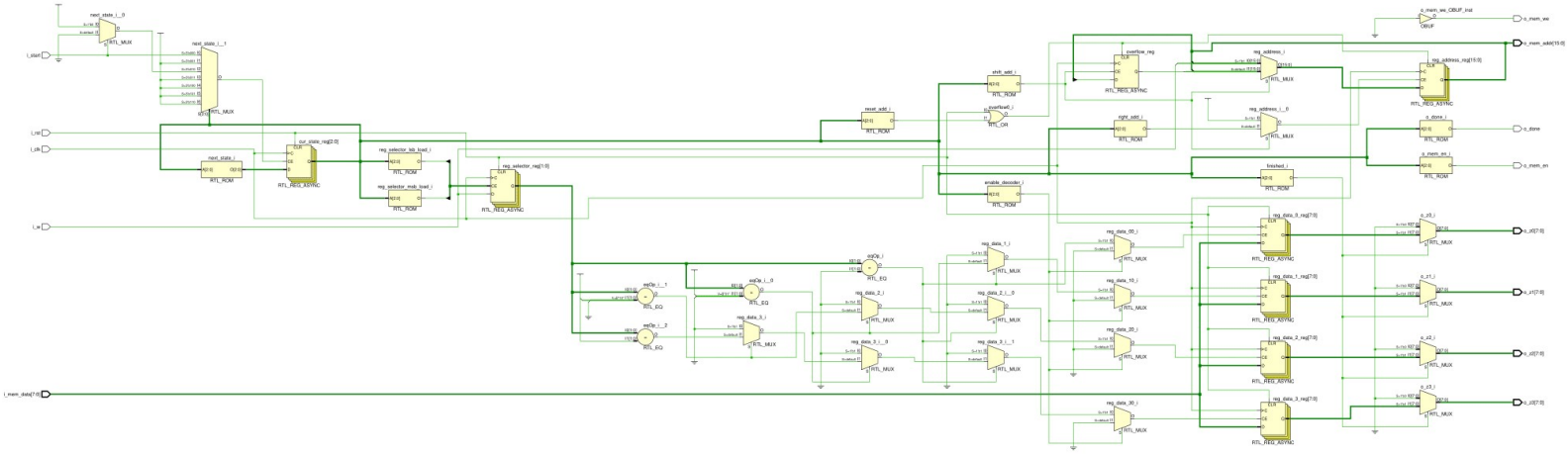
Description of each state:

- **SEL_MSB**: initialization state of the registers in which the most significant bit of the output is acquired. At the moment when **i_start** becomes high it switches to the SEL_LSB state, otherwise nothing happens. The signal **reg_selector_msb_load** is set to 1.
- **SEL_LSB**: acquisition status of the least significant bit of the output. The signal **reg_selector_lsb_load** is set to 1.
- **SHIFT**: memory address storage state by using the **reg_address** register. It remains in this state as long as **i_start** is high, the moment it goes low it switches to the RIGHT state. The signal **shift_add** is set to 1.
- **RIGHT**: state of completion of the acquired address, in which we shifted **reg_address** by one position to the right, inserting the register **overflow** in the most significant position.
- **MEM**: state in which the completed memory address is saved and the data is expected from RAM. The signal **o_mem_en** is set to 1, so as to enable reading into RAM.
- **WRITE**: write status of the data received from RAM to the appropriate output. **enable_decoder** and **finished** are set to '1', the former to save the RAM data to the proper output, the latter to show what is saved in the output registers.
- **DONE**: final state in which the signal **finished** remains high. The signals **o_done** and **reset_add** are set to '1', the former to show that the entire process has been completed, the latter to reinitialize the **reg_address** and **overflow** registers.

2.1 FSM scheme



2.2 Schematic of the entire project



3 Experimental Results

3.1 Synthesis

The project was implemented and tested on the version of **Vivado 2019.1**, specifying Artix-7 xc7a200tfbg484-1 as the target FPGA. The following summary report was generated:

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	37	0	134600	0.03
LUT as Logic	37	0	134600	0.03
LUT as Memory	0	0	46200	0.00
Slice Registers	54	0	269200	0.02
Register as Flip Flop	54	0	269200	0.02
Register as Latch	0	0	269200	0.00
F7 Muxes	0	0	67300	0.00
F8 Muxes	0	0	33650	0.00

For states and consumption instead:

Type	Power	Util%
Signals	0.448 W	16%
Logic	0.285 W	10%
I/O	2.059 W	74%
Device Static	0.142 W	5%

4 Testing

Having completed the description and synthesis phase of the component, we moved on to the testing phase, in which the component responded correctly to all testbenches to which it was subposed. Some Behavioural simulations are given below, showing in particular borderline cases.

4.1 Test 1

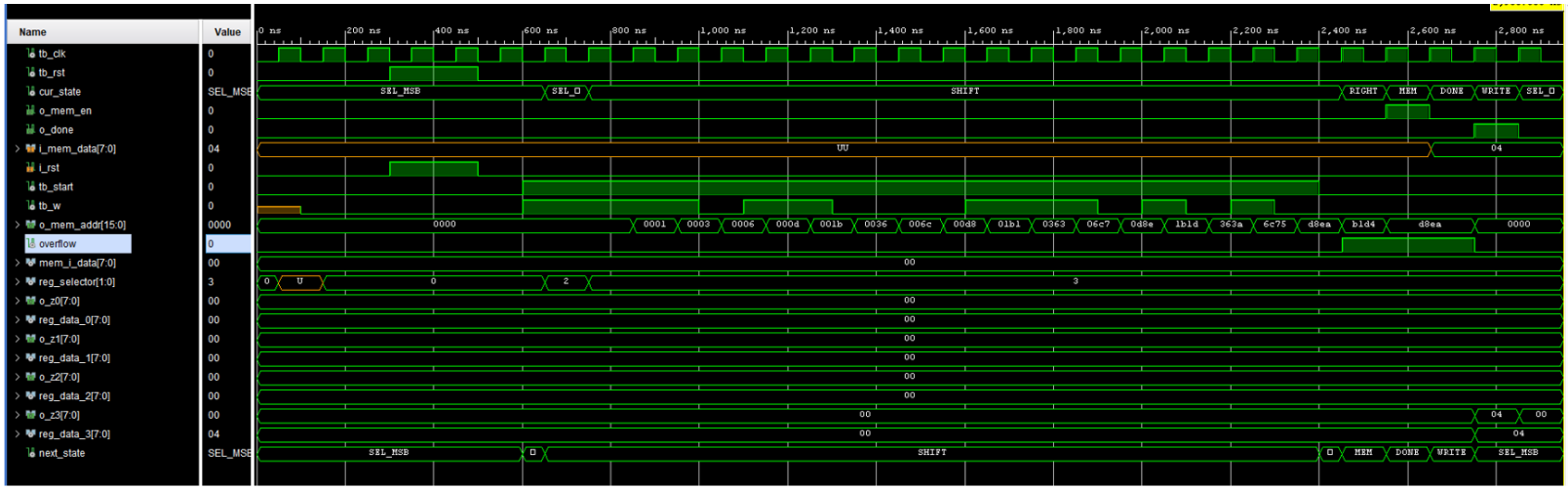


Figura 1: Behavioural simulation of the first test.

This test is reported as all 16 bits of the address are used, and the overflow register is set to '1', showing its usefulness.

4.2 Test 2

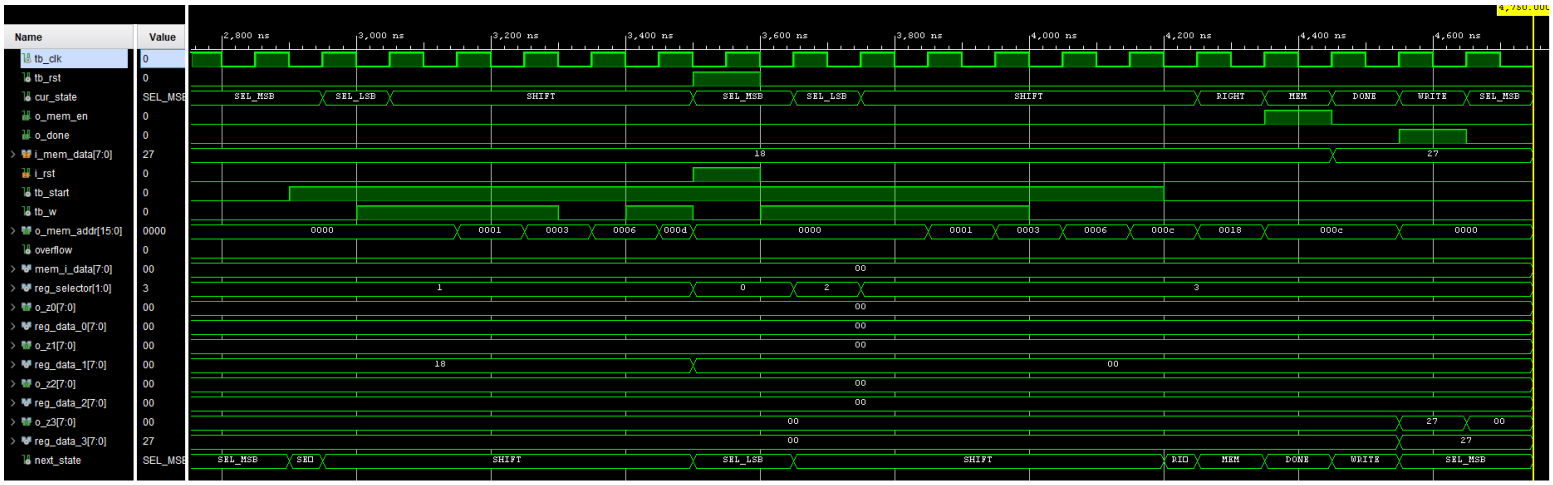


Figura 2: Behavioural simulation of the second test.

The simulation of this test was reported as involving a reset case that is set to '1' while the component is being read.

4.3 Test 3

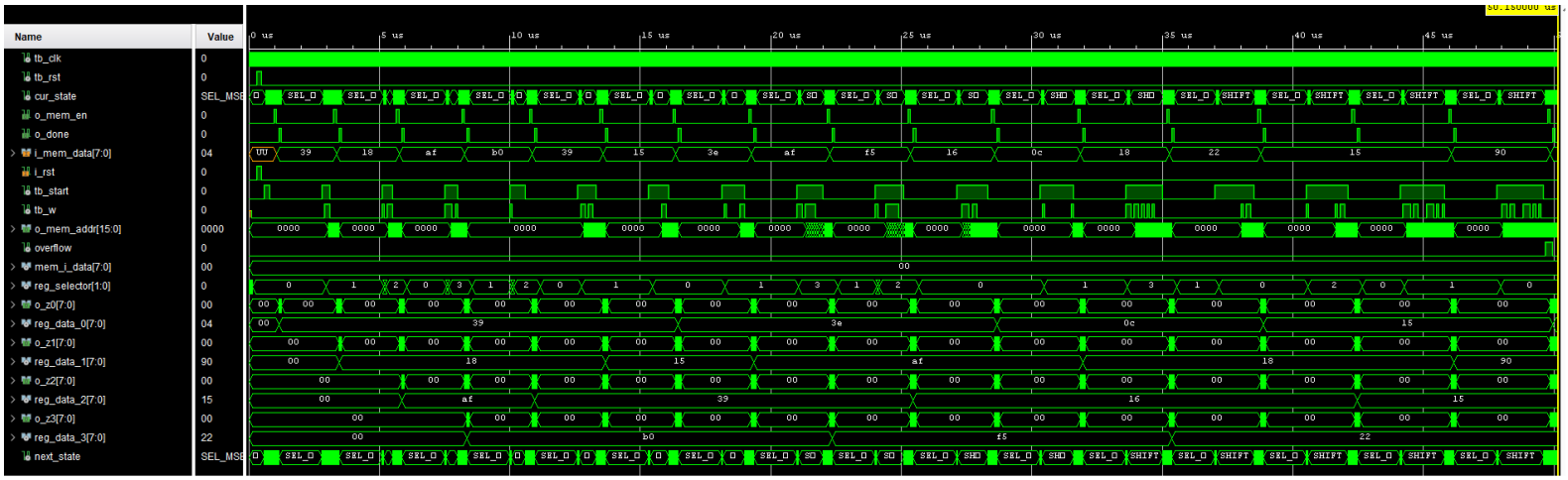


Figura 3: Behavioural simulation of the third test.

The simulation of this test was reported as it can be considered a 'stress' simulation with repeated operation.

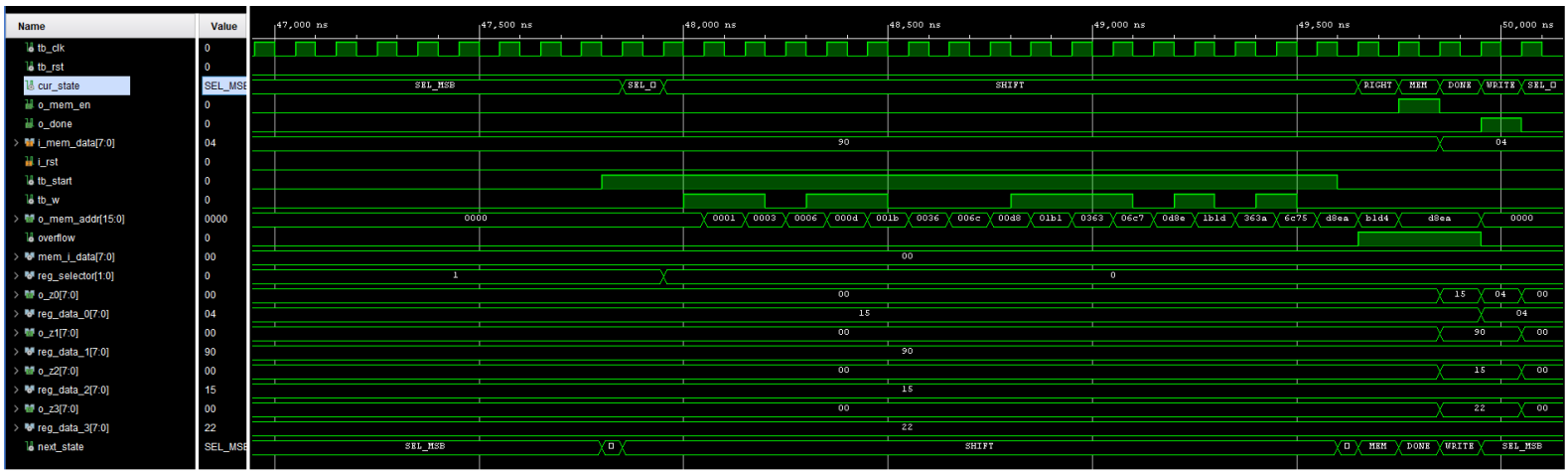


Figura 4: Zoom of the simulation of the third test reported as the overflow register is activated.

5 Conclusion

Studying and implementing this project allowed us to understand how memory addressing works at a very low abstract level, going deeper into an aspect widely used in a great many programming languages.
We were also able to analyze and handle different cases, trying to get total coverage for all extreme cases. Collaboration was a very important aspect as it brought different ideas both to identify a solution and to find cases that could block the proper functioning of the component.