Department of Electronics and Telecommunications
Bachelor degree in Electronic Engineering

Digital Systems Electronics

LAB 5: FSM.

Due date: April 29, 2024
- extended to: May 03, 2024
Delivery date: May 03, 2024

Group: 19

Contributions:

- Valtorta Alexander James
- Tedesco Angelo
- Urgu Sara

The members of the group listed above declare under their own responsibility that no part of this document has been copied from other documents and that the associated code is original and has been developed expressly for the assigned project.

Contents

0 Introduction

0.1 Aim of this project

The purpose of this laboratory is to investigate the operation of Finite State Machines, implementing different FSMs starting from different design hypotheses.

0.2 Content and structure of the document

The report is divided into four sections, each for every exercise of the laboratory assignment. Moreover, each section contains the key parts in order to design and implement each circuit:
- Design Entry, which describes the circuit to be implemented on the FPGA device, in particular it is indicated the overall architecture of the circuit, by reporting the name of the source file or files and the key role of each of them.
- Functional Simulation, which describes the testbench and the approach used to assess the completed design. (provided when requested)
- Synthesis, which shows the results of Modelsim functional simulation and that of the timing simulation.

1 One-Hot Finite state machine

1.1 Design entry

In order to implement a Finite State Machine (FSM) that recognizes four consecutive 1s or four consecutive 0s the following design is implemented:

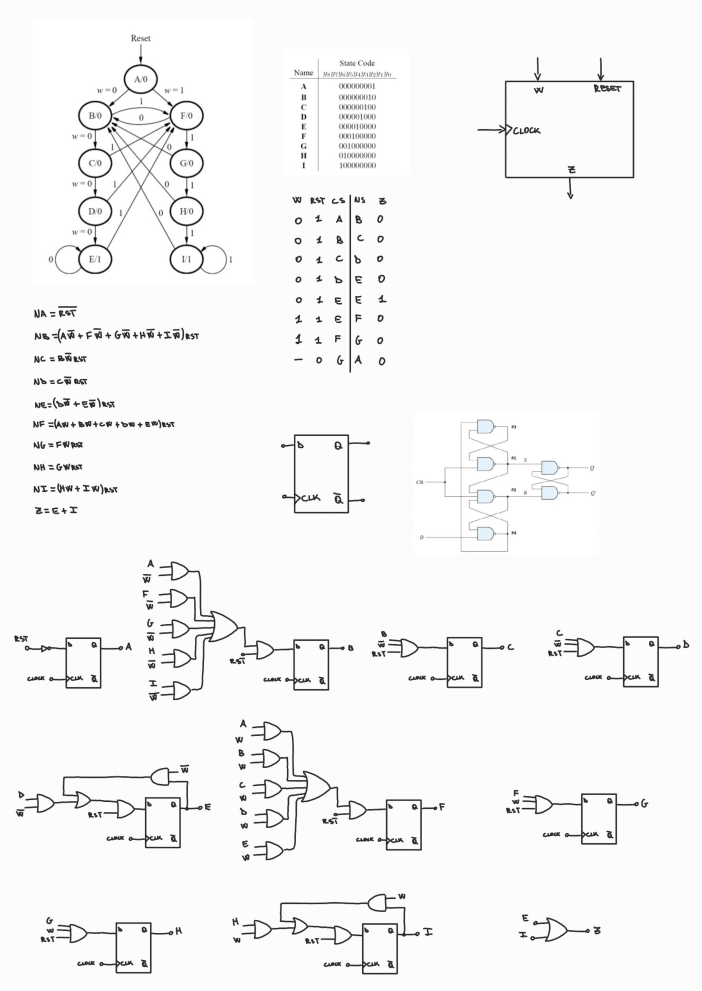

Figure 1: Design of the circuit

In particular it is composed by:

- The D-type flipflop is defined in *D_flipflop.vhd* with a dataflow architecture that specifies the boolean equations that relate each internal signal *s1, s2, s3, s4, s5, s6* to the inputs and output declared in the the entity *D, CLK, Q*.

Since the FF defined with the logic gates seem to have metastability problem, they are also defined with their behavioural approach:
- The D-type flipflop is defined in *D_flipflop2.vhd* with a behavioural architecture that checks for the rising edge of the clock through a process.

- The actual One-Hot FSM is defined in *onehot_fsm.vhd* where, once defined *D_flipflop2.vhd* as component and the internal signals *A, B, C, D, E, F, G, H, I, NA, NB, NC, ND, NE, NF, NG, NH, NI, nW, nReset* it is possible to describe the logic equation for all nine states thus mapping the right signals of the corresponding FF, following the state diagram provided and the resulting boolean equations.

- Lastly in the *onehot_map.vhd* it takes place the mapping of the inputs and outputs of the One-Hot FSM (defined in *onehot_fsm.vhd*) to those assigned to the DE1 board. In particular are assigned

KEY0 as a clock input, switches *SW0, SW1* as Reset and input W respectively and, finally, the LED *LED0* as the output *Z*.

1.2 Functional simulation

With the testbench *tb_onehot_fsm.vhd* it is possible to check the functionalities of the completed design, in particular, after the typical empty entity, in the declarative part of the behavioural architecture is defined the component (specified in *onehot_fsm.vhd*), and the internal signals for the simulation initialized to '0' (*W_t, Reset_t, Clock_t, Z_t*).
Then, in the descriptive part, once the internal signals are properly mapped to the components' ones the simulation is performed through means of a two processes: one for the simulation of the reset and input signal, and the other for the simulation of the clock. The former is performed by assigning to the reset internal signal (*Reset_t*) the logic value '0', then toggling it to '1', than the same is done for the input internal signal (*W_t*), waiting a short amount of time in between each commutation. The latter is performed by assigning to the internal signal (*CLK_t*) first the logic value '0' and then '1', waiting a short amount of time in between.

1.3 Synthesis

Once the circuit is successfully synthesized by Quartus it is possible to run the Modelsim functional simulation.

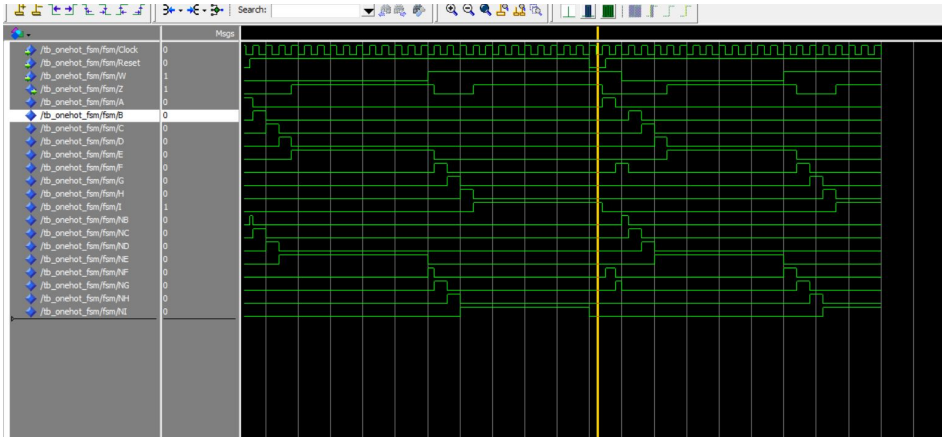

Figure 2: Modelsim functional simulation

2 Modified One-Hot FSM

2.1 Design entry

In order to simplify the circuit it is only a matter of complementing the output of the first FF. This is done by modifying the code in the *onehot_fsm.vhd* (keeping the *D_flipflop2.vhd, onehot_map.vhd* untouched), thus adding the internal signal *An* to manage the correct associations for the equations of the states (in particular that of the B and F next-states).

2.2 Functional simulation

Also the testbench is carried out exactly as that of the first assigment. File *tb_onehot_fsm.vhd*.

2.3 Synthesis

Once the circuit is successfully synthesized by Quartus it is possible to run the Modelsim functional simulation.

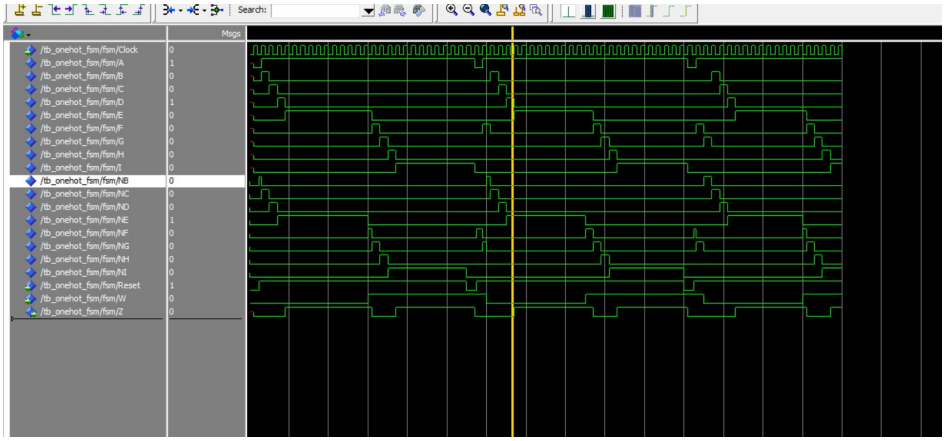

Figure 3: Modelsim functional simulation

3 Two-process FSM

3.1 Design entry

Following the assignment provided it is possible to describe the state table for the FSM by using a VHDL CASE statement in a PROCESS block, another PROCESS block to instantiate the state flip-flops and a simple assignment statements to specify the output z. File *two_process_fms.vhd*.

In particular the implementation follows:

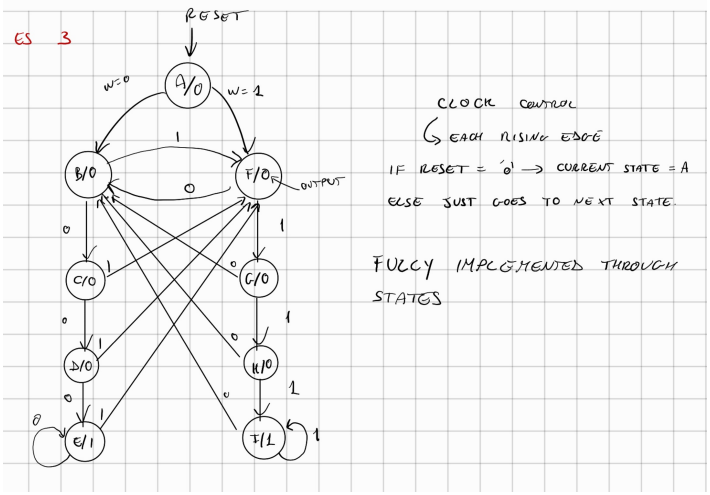

Figure 4: Design of the circuit

3.2 Functional simulation

With the testbench *tb_fsm.vhd* it is possible to check the functionalities of the completed designs. After the typical empty entity, in the declarative part of the behavioural architecture is defined the component (specified in *two_process_fsm.vhd*), and the internal signals for the simulation initialized to '0' (*W_tb, Clk, Rst, Z_tb*).
Then, in the descriptive part, once the internal signals are properly mapped to the components' ones the simulation is performed through means of a two processes: one for the simulation of the reset and input signal, and the other for the simulation of the clock. (in the same way as previously described).

3.3 Synthesis

Once the circuit is successfully synthesized by Quartus it is possible to run the Modelsim functional simulation.

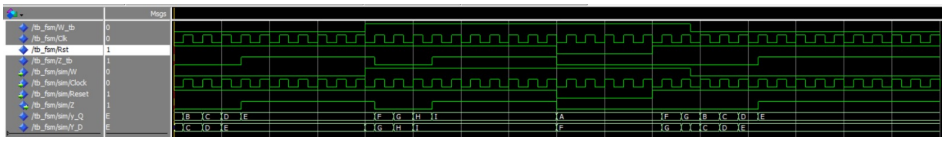

Figure 5: Modelsim functional simulation

4 "HELLO" FSM

4.1 Design entry

To implement the desired circuit the following design is considered:

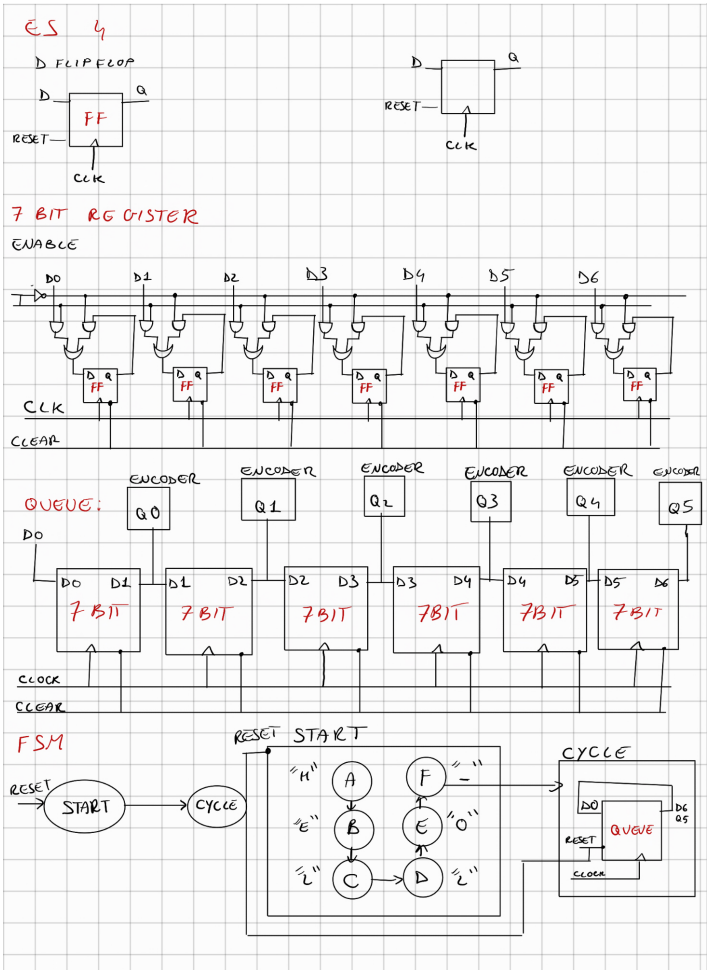

Figure 6: Design of the circuit

In particular it is composed by:

- The D-type flipflop is defined in *d_flipflop.vhd* with a dataflow architecture that uses an asynchronous reset.

- The 7-bit register is defined in *register7bit.vhd* where the FF (defined in *d_flipflop.vhd*) is declared as a component in the declarative part of the dataflow architecture, as well as *Dv, Qv* as initialized internal signals. In the descriptive part the internal signals are properly associated, by means of a for-generate statements, for each of the seven bits, and the inputs and outputs of the components are mapped to those declared in the entity.

- In *queue_reg.vhd* the six 7-bit registers are connected in a queue-like fashion, as requested. In particular the 7-bit register is defined as a component in the declarative part of the dataflow architecture, as well as *D0, D1, D2, D3, D4, D5, D6* as initialized internal signals; then in the descriptive part the inputs and outputs of the components are properly mapped to those defined in the entity and the internal signals are associated to the outputs *Q0, Q1, Q2, Q3, Q4, Q5* so that the registers are connected in such a way that the outputs of the first register feed the inputs of the second one, and so on.

- Finally the whole circuit is implemented in *hello_fsm.vhd*, where in the declarative part of the architecture the 7-bit register is defined as a component (from *queue_reg.vhd*), two types are defined (*top_state, inner_state_type*) with the corresponding lists of states (*start, cycle, A, B, C, D, E, F*) that can be assigned to the signals defined below each type, and initialized internal signals (*enable_reg, count, Dv, Hv0, hv1, Hv2, Hv3, Hv4, Hv5*). In the descriptive part two processes are used. The first one to manage association of the letters of the word 'HELLO', thus, when the proper condition is verified, by means of a case statement, the *A, B, C, D, E, F* sates are connected in a chain-like manner by associating the correct next-state and the 7-bit sequence of the corresponding letter to the *Dv* signal (also considering a blank space after the five letters), so that it is than associated to the *Hv5* signal (the idea is to create a loop by connecting the last register to the first one). The second one checks whether the reset is asserted thus "initializing" signals and the counter to the right states, otherwise checks for the rising edge of the clock to enable the register and associate the signals and the counter to the right states if the counter as reached the correct value. Lastly the internal signals are mapped to those of the component and the to the corresponding ones of the displays, defined in the entity .

4.2 Functional simulation

With the testbenches it is possible to check the functionalities of the completed designs.

- In *reg_tb.vhd*, after the empty entity, in the declarative part of the behavioural architecture is defined the component (specified in *register7bit.vhd*), and the initialized internal signals for the simulation (*Ena, Clk, Rst, Dt, Qt*).
Then, in the descriptive part, once the internal signals are properly mapped to the components' ones the simulation is performed through means of a two processes: one for the simulation of the reset, input signal and enable, and the other for the simulation of the clock. (in an analogous way as previously described).

- The same is done in the *queue_tb.vhd*.

- In *Hello_tb.vhd*, after the empty entity, in the declarative part of the behavioural architecture is defined the component (specified in *hello_fsm.vhd*), and the internal signals for the simulation (*rst, clk, hx, hx2, hx3, hx4, hx5*).
Then, in the descriptive part, once the internal signals are properly mapped to the components' ones the simulation is performed through means of a two processes: one for the simulation of the clock and the other for the simulation of the reset (in an analogous way as previously described).

4.3 Synthesis

Once the circuit is successfully synthesized by Quartus it is possible to run the Modelsim functional simulation.

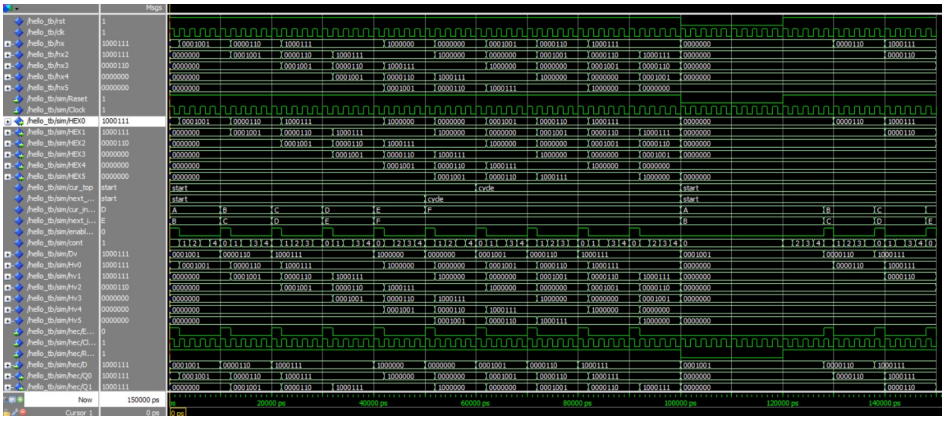

Figure 7: Modelsim functional simulation