REALIZATION OF THE FINAL STATE MACHINE WITH CPLD DEVICE

**Ivan Simeonov Simeonov, Valentina Stoianova Kukenska,
Raycho Todorov Ilarionov**

Computer Systems and Technologies, Technical University of Gabrovo, Hadji Dimitar Str. 4, 5300 Gabrovo, Bulgaria, phone: +359 66 223 479, e-mail: isim@tugab.bg

In this paper is presented practical realization of Mealy determinate final state machine which is set by table. For this purpose here is used the technology for designing with programmable logic - Complex Programmable Logic Devices (CoolRunner-II XC2C128), based on the development system Xilinx ISE WebPACK 7.1i. The project is based on the family CoolRunner-II XC2C128 from the new generation CPLD devices, developed by Xilinx. The realization of the state machine is done according to its graphical description with the help of StateCAD(r). The work of the developed final state machine is simulated with the help of ModelSim XE III 6.0a. The CPLD device is configured in laboratory environment "in-place" (in system programmable). The programming of the scheme is made according to the IEEE 1149.1 JTAG standards, with the help of specialized programmer CPLD CoolRunner-II XC2C128 Design Board, connected by JTAG (Joint Test Action Group) cable for communication with the parallel port of the personal computer.

Keywords: ISE WebPack, VHDL, CPLD, ModelSim, Design Board, JTAG

1. INTRODUCTION

The ability of the user to integrate complicate specialized systems in integrated circuits is deduced in the appearance and the development of programmable logic (CPLD and FPGA) and leads to significant change in the technology for designing of the digital systems.

The processes for designing of the CPLD and FPGA devices have no significant differences each other. With small exceptions the process for designing of programmable and reprogrammable devices offered by different firms, have same consequence – entering, verifying, realizing and programming. The process of designing is made with the help of own development system for every one firm. In this paper is presented simplified the methodology for designing with programmable logic devices by firm Xilinx in the development environment **ISE WebPACK 7.1i** and more specifically, the realization of the Mealy state machine, set by a table. With the synthesizer **Xilinx Synthesis Technology (XST)** the synthesis of the state machine is made. Its realization is made after its graphical description by means of **StateCAD(r)** [1]. As a result the State Diagram file type of the project was created (a diagram of the state machine). The purpose of this paper is to be realized the created before project with the libraries of Xilinx (CoolRunner II) and so the programming of Xilinx CoolRunner-II CPLD.

2. REALIZATION

For the realization of the state machine is used created with the help of **StateCAD** project SMM1.DIA [1] (fig. 1). The graph of the state machine is produced from which after compilation (with the button *Generate HDL*, shown on the fig. 2) the

VHDL description is made [2], preceded by overall statistics for the project and made analyses (fig. 3). If there are syntactical mistakes after the verifying of the VHDL description, in the console window of the ***Project Navigator*** will appear messages for the type and the place of the mistake. If there are such, they will be corrected in the VHDL file, after what the modified file can be saved with the menu **File→Save** [2].

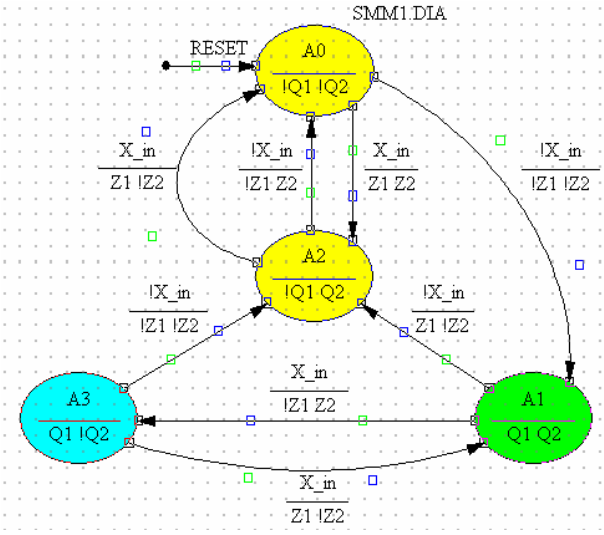

Fig. 1. The graph of the state machine

After cleaning up of the code from mistakes the next stage of designing is made – functional simulation. In the environment **ISE WebPack 7.1i** it is used external simulator – **ModelSim** and after its start the **StateBench** is called too. On the fig. 4 is shown the simulation of the project SMM1.DIA with the help of **StateBench**. The periods and the pauses are set previously in the window Constrain Timing, called by button Constrain Timing.

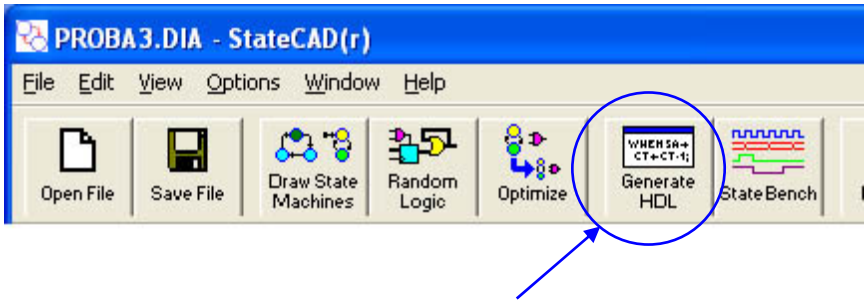

Fig. 2. Button for generating of VHDL code, describing the state machine

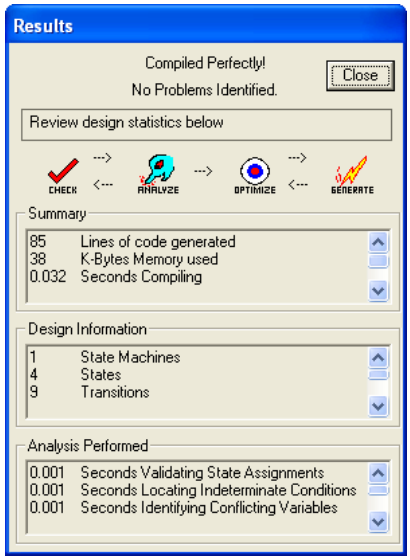

Fig. 3. Results after correct generating of the VHDL code of the state machine

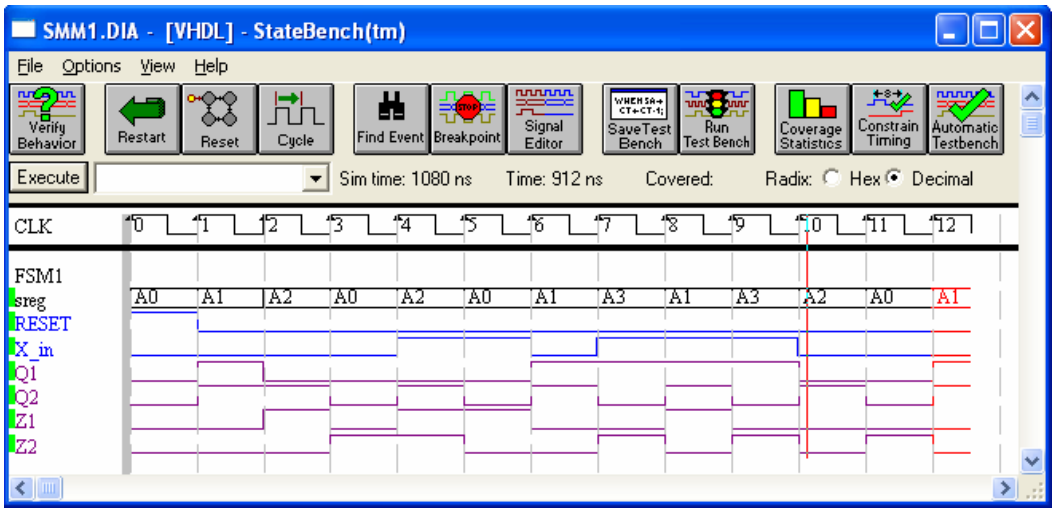

Fig. 4. Simulation of the Project SMM1

With the help of the simulation is verified the behavior of the state machine and the analysis respective with the law for its functioning (Table 1-1).

Table 1-1
Table of the transitions and outputs of the Mealy state machine

$X_i \backslash A_j$	a_0	a_1	a_2	a_3
X_1	a_1/Z_1	a_2/Z_3	a_0/Z_2	a_2/Z_1
X_2	a_2/Z_4	a_3/Z_2	a_0/Z_3	a_1/Z_3

After finishing of the check for correctness for the designed module the work with the **StateCAD** is done and the Project Navigator from the **ISE WebPACK 7.1i** have to be started. From the menu **Project/Add Source** the two created files have to be added. First the file with the extension *.VHD, and after this - with *.DIA.

The next step after the simulation is *the physical designing*. It consists from setting of **User Constraints** and **Implement Design** in the chosen chip.

The available **Design Utilities** are: **Create Schematic Symbol**, **Launch ModelSim Simulator**, **View Command Line Log File** and **View HDL Instantiation Template**.

```
4   -- Notes:
5   -- 1) This instantiation template has been automatically generated using ty
6   -- std_logic and std_logic_vector for the ports of the instantiated module
7   -- 2) To use this template to instantiate this entity, cut-and-paste and th
8
9      COMPONENT smm1
10     PORT(
11        CLK : IN std_logic;
12        RESET : IN std_logic;
13        X_in : IN std_logic;
14        Q1 : OUT std_logic;
15        Q2 : OUT std_logic;
16        Z1 : OUT std_logic;
17        Z2 : OUT std_logic
18        );
19     END COMPONENT;
20
21     Inst_smm1: smm1 PORT MAP(
22        CLK => ,
23        RESET => ,
24        X_in => ,
25        Q1 => ,
26        Q2 => ,
27        Z1 => ,
28        Z2 =>
29     );
```

smm1.vhi (R...

Fig. 5. VHDL description of the ports of the created by StateCAD component

With the help of the tool **View HDL Instantiation Template** an excerpt is made, which is VHDL description of the ports of the created with the help of **StateCAD** component with extension *.vhi (fig. 5).

There is need at the time of designing to be given user parametrical constraints (**User Constraints**). They can be related to timings (they are set by **Create Timing Constraints**), for assigning of the functions between the pins of the chosen chip (they are set by **Assign Package Pins**), and also their editing in text view (with the help of **Edit Constrains (Text)**). The given constraints are saved in Implementation

Constraint File (UCF), which is created by the integrated environment [3]. The implementing of the design in the chosen chip (**Implement Design**) is made by passing through the next settings: **View Synthesis Report** (a report for everything done till this moment for the synthesis of the state machine), **View RTL Schematic** (block-scheme of the synthesized module with an annotation of the input-output pins, fig. 6), **View Technology Schematic** (examination of the technology scheme, fig. 6), **Check Syntax** (For checking of the syntax), **Translate** and **Translation Report** (a report for the made from the environment translation – like a result from this the files are created with the extensions *.ngd, *.bld), **Fit** and **Filter Report** (This repost gives information for the made settings of the project and the result after the implementing of the project in the chip, from it can be viewed the list of errors and warnings in the project (if there are some), and other information for the project too, available by the menu from the appeared window), **Generate Programming File** (Creating of file for programming of the chosen chip (CPLD or FPGA)), **Configure Device (iMPACT)** – for settings about the configuration mode of the connection to JTAG cable (automatically or manually settings). The message from the fig.7 will appear after successfully made communication between PC and the development printed circuit (kit), after which the environment asks for JEDEC Design File (a project file for programming in the chip). The CPLD chip has to be selected from the appeared chain (daisy chain) and with the right button of the mouse the Program have to be chosen. As a result after the finishing of the programming of the design in the chip the environment will visualize the shown on fig. 8 window.

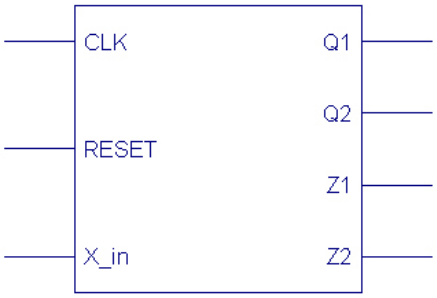

Fig. 6. Block-scheme of the synthesized module with annotation of the input-output pins, and also graphical view of the technological scheme

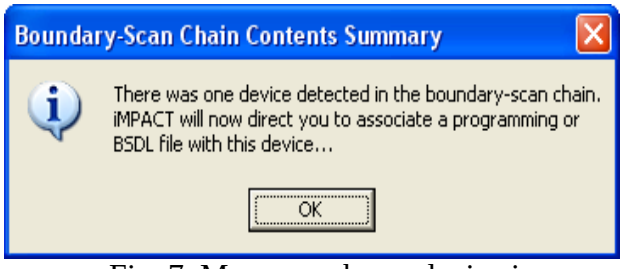

Fig. 7. Message when a device is found

After successful programming of the project in the chosen chip there are additional tools for the implementation (**Optional Implementation Tools**). They are: **Generate Timing, Timing Report** (for visualizing of information for timing characteristics of the implemented project. This information can be viewed from **Filter Report**, one of the **Implementation Design** tools) and **Generate Post-Fit Static Timing (Timing Analyzer)** – perform analyses of time delay during the

transactions between the states of the state machine according to: the set of **User Timing Constraints**, automatically generated timing constraints, calculated time delay for signals spreading between the points, defined by the user and analysis for the relation between timing constraints.

Generate Post-Fit Simulation Model performs simulation of the implemented into the chip project, after which this information is visualized by activating of the tool **Post-Fit Simulation Model Report**.

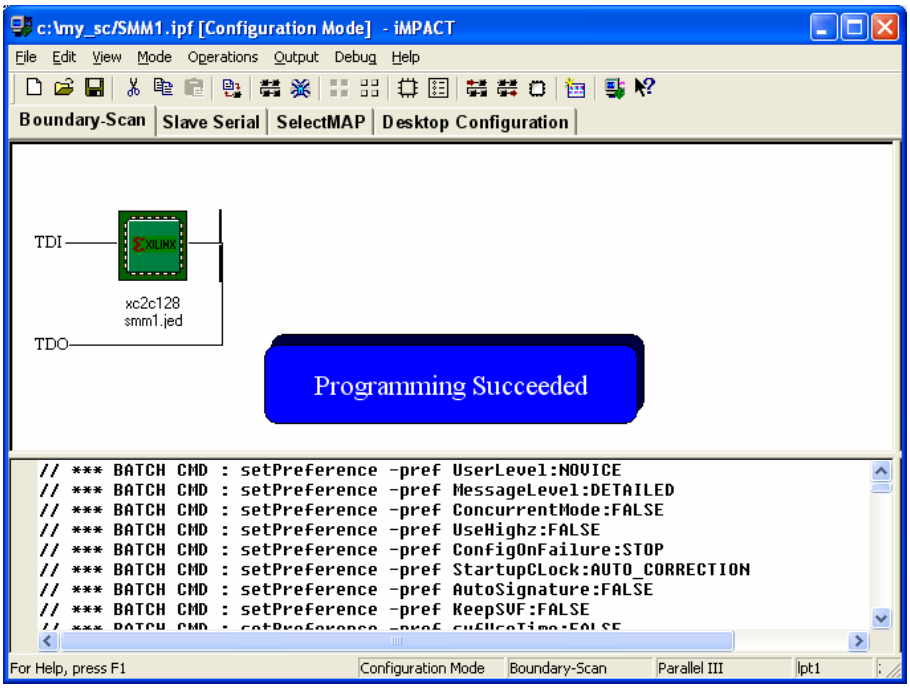

Fig. 8. Message after successful programming of the project in the chosen chip

Generate IBIS Model – performs simulation HSPICE of the project according with the chosen chip from the family CoolRunnerII and the type of the integrated circuit body. This information can be visualized after activating of the tool **View IBIS Model**.

Analyze Power (XPower) - XPower is based on measurements of real designs with active functional elements reflecting real world design scenarios. The analysis and the result from it can be performed by the tools **Generate Power Data** and **View XPower Report**.

Lock Pins – a tool for locking of the chip pins.

Generate SVF/XSVF/STAPL File – for creation of a file for the chip programming.

3. CONCLUSION

With the help of produced results is produced a file for programming of the chosen CPLD chip XC2C128 from the family integrated circuits CoolRunner II. Its physical programming is made with the help of specialized programmer, connected with JTAG cable to PC (Fig. 9) in a laboratory environment. From the made experiments the ability to work of the state machine was proven according to its

graph from fig. 1. It was found that there is correspondence between associated pins from the implementation and the physical pins of the chip (fig. 10). The same approach can be used successfully for the synthesis of the Moore state machine.

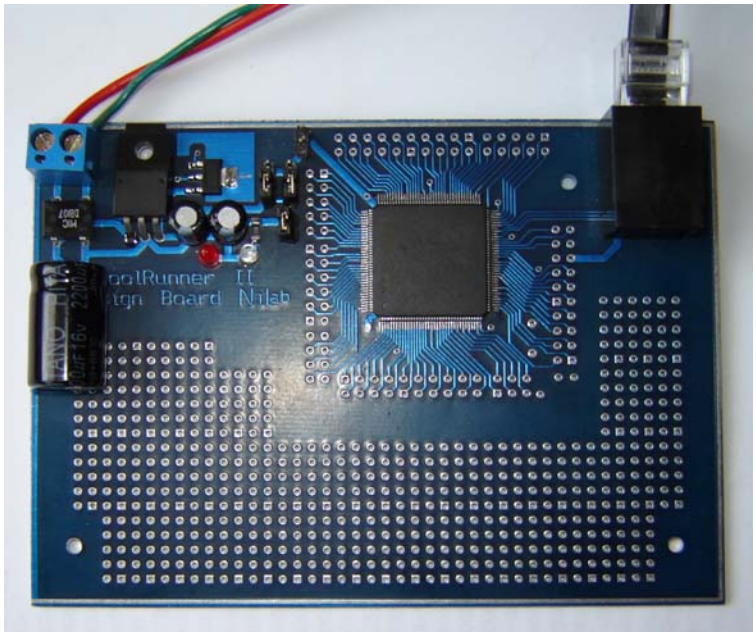

Fig. 9. Kit with a device from the family CoolRunner II

With the help of produced results is produced a file for programming of the chosen CPLD chip XC2C128 from the family integrated circuits CoolRunner II. Its physical programming is made with the help of specialized programmer, connected with JTAG cable to PC (Fig. 9) in a laboratory environment. From the made experiments the ability to work of the state machine was proven according to its graph from fig. 1. It was found that there is correspondence between associated pins from the implementation and the physical pins of the chip (fig. 10). The same approach can be used successfully for the synthesis of the Moore state machine.

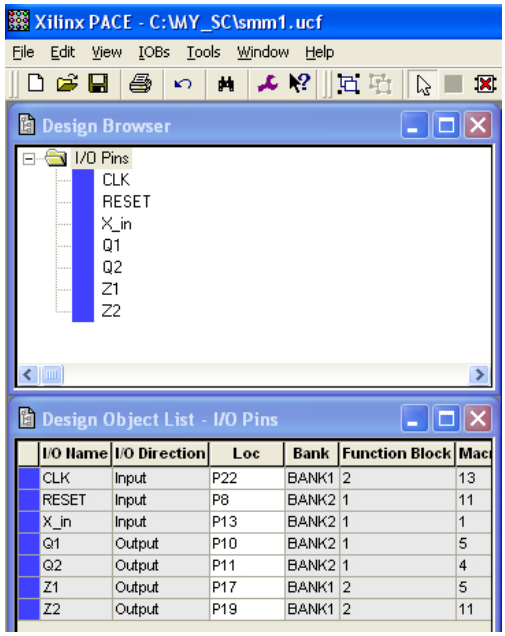

Fig. 10. Associating of the chip pins with the input-output ports

4. REFERENCES

[1] Simeonov, I.. S. *Designing of a State Machine with Programmable Logic*. Proceedings of the Fourteen Int. Conference ELECTRONICS'05, September 21 – 23, 2005, Sozopol, (in the press).

[2] Nancheva-Filipova K., Using (v)HDL for electronic hardware synthesis, Sofia, 2004.

[3] Spartan-II Data Sheet (www.xilinx.com)

[4] Gizdarski E., Designing with a programmable logic, Rousse, 1998.