9th International scientific conference
Technics and Informatics in Education – TIE 2022
16-18 September 2022

Session: *Engineering Education and Practice*
Professional paper
DOI: 10.46793/TIE22.137K

Use of electronic design automation tools in computer engineering courses

Mihailo Knežević, Slađana Đurašević, Vanja Luković, Željko Jovanović and Uroš Pešović*
University of Kragujevac, Faculty of technical sciences Čačak, Serbia
* uros.pesovic@ftn.kg.ac.rs

Abstract: *Electronic Design Automation (EDA) tools are widely used semiconductor industry to support the ever-growing complexity of computer design, and they became an important skill which is required by computer engineers. In this paper, we present the usage of the EDA Playground web platform for practical exercise in the Digital systems design course at the computer engineering department of the Faculty of technical sciences. This platform was used in process of designing and testing of program counter register of a simple processor.*

Keywords: *EDA; Verilog; design; simulation; HDL*

1. INTRODUCTION

Computer engineering is a branch of electrical engineering which integrates knowledge from the field of computer science to design new computer hardware and software. This discipline is responsible that the software part of the computer system being integrated seamlessly with the hardware part of the computer system. To achieve this goal, computer engineering requires a solid foundation in computer architecture, programming languages, operating systems, electrical engineering, electronics, and a variety of subdisciplines [1].

Due to the high complexity of current computer systems, which include billions of transistors and millions of lines of code, computer engineering relies on a set of automated tools which significantly increase design productivity. In the case of computer software, programmers write the programs in high-level languages, which syntax is very abstract for the computer hardware to be executed directly. Therefore, a set of tools is developed which automatically translates this high-level code into machine executable code. These tools include compilers, linkers, debuggers and assemblers.

The design of computer hardware relies on a set of EDA (Electronic Design Automation) tools which are used through the various design phases. These tools are used in the design and testing of integrating circuits, automating the placement of logic elements and interconnections on silicon wafer of integrated circuit, as well as designing PCB (Printed Circuit Boards) boards which are found in almost every computer system.

Different software tools are used for designing and simulating of digital circuits where some of the most common are Logisim, LogicWorks, LOGiX, Digital-ProfiLab, Cedar Logic Simulator, Deeds-DcS and Proteus. These simulators represent design in form of the schematic diagram, which is very convenient for simple designs, and can only stimulate design signals with values manually set by the user. On the other hand complex designs are today mostly represented using the HDL (Hardware Description Language) in which the structure or functionality of digital circuits are represented by a specific type of program language with a high level of abstraction.

EDA Playground is a free web platform that provides the possibility of designing, simulating, and verifying hardware described in HDL language. It supports most HDL languages such as Verilog, VHDL, C++, and SystemC, which can be run on various free and commercial simulators. This platform enables easy design and development of functional tests for small prototypes. Easy sharing of project code and simplified access to simulators and libraries makes it especially suitable for educational purposes.

EDA Playground platform is used as a tool for practical exercises for two computer engineering courses at the Faculty of technical sciences: Digital systems design [2] and VLSI systems design [3]. In this paper, we present a design and testing methodology for the practical exercise of a program counter register of a central processor of Hack computer [4]. This design is used as part of the design of the digital systems course.

2. PROGRAM COUNTER DESIGN

The program counter (PC) is one of the most important registers of the central processor. Its role is to point the address of the next instruction to be executed. When the computer is powered on for the first time or rebooted, the value of the program counter is set to zero, so this register points to the first instruction in the program located at the address 0. When the current instruction is executed, program counter is incremented for the length of the current instruction to point to the address of the next instruction. The only exceptions to this rule are the branching instructions, for which program counter can point to the address of jump instruction, depending on the certain condition.

The most widely used HDL languages in the industry are VHDL and Verilog. In Verilog circuit is represented as a block, called a module which implements certain functionality between its input and output interfaces, called ports.

The final circuit, called the top module, can be composed of lower-level modules which are embedded inside the top module of it and the top module communicates with them using their input/output ports. This kind of design hierarchy enables module reuse and helps the designer in tackling with the design complexity.

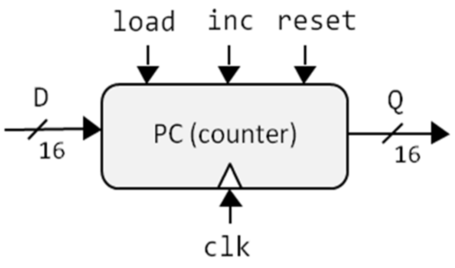


Figure 1. *Interfaces of program counter module*

Interfaces of the program counter include the following ports:

- Q – represents 16-bit wide output value of program counter which points to the address of the next program instruction.
- clk – represents the input signal which oscillates between a high and a low state and is used to coordinate actions of digital circuits.
- inc – represents the input signal used to increment the current program counter value after every positive transition of clk signal. In this design, each instruction has a length of one word, so inc input is represented by the one bit wide signal.
- D - represents 16-bit wide input value of address of the jump instruction.
- load – represents the input signal which loads the value of input in to program counter when jump condition is true.
- reset – represents the input signal which resets the value of the program counter to zero.

The structural view of the program counter module is composed of several lower-level modules. Register module r0 is responsible for holding the current 16-bit program counter value. This value can be reset by the reset input, or it will always load the value from the output of the adjacent multiplexer m1 at the positive transition of clk signal. This multiplexer selects one of two 16-bit inputs depending on the value of the Select input which is connected to the load signal. This multiplexer will load the address of jump instruction present at D input or the output of the adjacent multiplexer m0. Multiplexer m0 will select the current value of the program counter or the value of the program counter incremented with one by the adder a0, depending on the value of inc input.

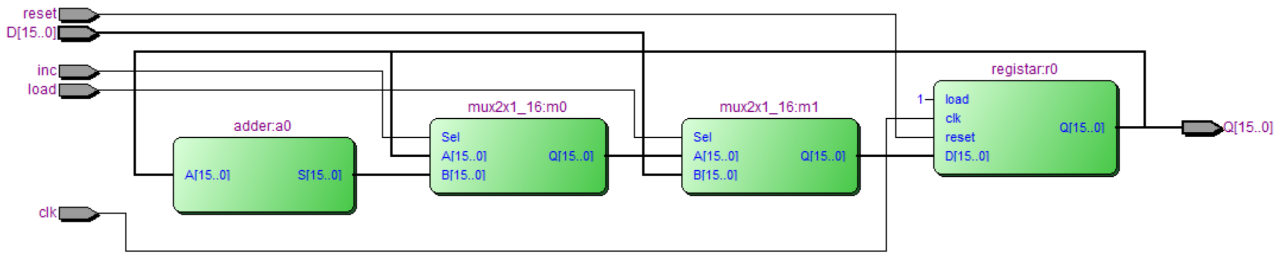
3. EDA PLAYGROUND PROJECT

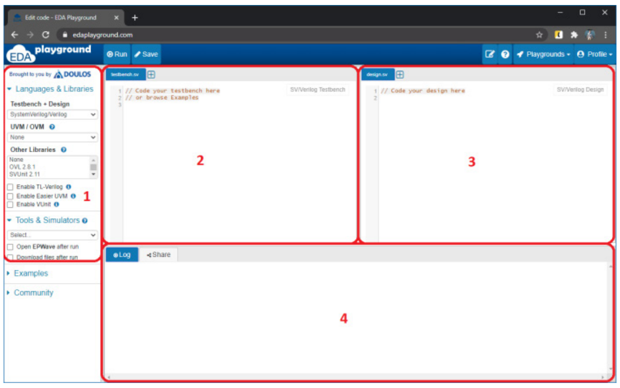
The EDA Playground platform requires the creation of a user account, for which it is recommended to use an institutional e-mail address, to be able to use all types of simulators [5]. Within its account, the user has a personal directory for storing the active projects. For each project, the user can adjust its visibility on the EDA Playground platform. The project can be

- private – project is invisible to other EDA Playground users.
- public - project is visible to users which are provided with the project URL.
- published - project is visible to all users of the EDA platform via the search.

The EDA Playground workspace, shown in Fig 3. is divided into 4 main areas:

1. Language, libraries and simulator settings
2. Testbench editor
3. Design editor
4. Log output panel





```
// Code your testbench here
// or browse Examples
```

```
// Code your design here
```

Figure 3. *Layout of EDA Playground workspace*

The right part of the EDA Playground workspace, marked with 3 in Fig 3., is reserved for the design editor in which the digital component is described by Verilog HDL code. The design can include several components that can be found in single or multiple files, whereby the largest component in the hierarchy must be in the mandatory design.sv file. Additional design files must be appended to the design.sv file using the include command. The structural description in Verilog HDL of the program counter module, called PC, is shown in Fig 4. The top-level module includes several lower-level modules which are also described in the Verilog HDL code.

```
module PC (input [15:0] D,input
inc,load,clk,reset,output [15:0] Q);
   wire [15:0] S;
   wire [15:0] Q_inc;
   wire [15:0] Q_load;
   supply1 vcc;
   adder a0 (Q,S);
   mux2x1_16 m0 (Q,S,inc,Q_inc);
   mux2x1_16 m1 (Q_inc,D,load,Q_load);
   registar r0(Q_load,vcc,clk,reset,Q);
endmodule
```

Figure 4. *Verilog PC module of program counter*

The left part of the EDA Playground workspace, marked with 2 in Fig 3., is used for writing the tests which are used to check design functionality. These tests are known as testbench and are placed in the testbench.sv file. The testbench is represented as a module without external interfaces and its role is to instantiate the top-level design component, called DUT (Design Under Test). The main role of the testbench is to generate test signals for a certain scenario, pass these signals to the top-level design component, collect design output and present it to the user in the output log panel of the EDA Playground workspace marked with 4 in Fig 3.

In this exercise, testbench generated several test signals of register type, which all have the suffix "_t" which indicated that they are generated by the testbench. The top-level module PC is instantiated as a dut component and connected with test signals, while the output of dut is connected to the Q_t signal.

```
module testbench;
  reg [15:0]D_t;
  reg clk_t,load_t,inc_t,reset_t;
  wire [15:0]Q_t;
  PC dut(D_t,inc_t,load_t,clk_t,reset_t,Q_t);
  .......
endmodule
```

Figure 5. *Internal signals of testbench module*

The following step is the generation of test signals, which are carried out in discrete time instances. First, clock signal clk_t is generated, using the initial statement when is set to zero, after which is inverted every 5 ns throughout the simulation. After the generation of the clock signal other signals are generated in the initial block where a certain time delay is added between changes of a certain signal as shown by the testbench code represented in Fig 6.

```
initial
   clk_t = 0;
always
   #5 clk_t = ~clk_t;
initial
   begin
        D_t=16'h0000;
        load_t=1'b0;
        inc_t=1'b0;
        reset_t=1'b1;
   #10 inc_t=1'b1;
        reset_t=1'b0;
   #20 D_t=16'h0100;
        load_t=1'b1;
   #10 D_t=16'h0000;
        load_t=1'b0;
   #10 reset_t=1'b1;
   #10 reset_t=1'b0;
   #50 $finish;
   end
```

Figure 6. *Generation of test signals for testbench module*

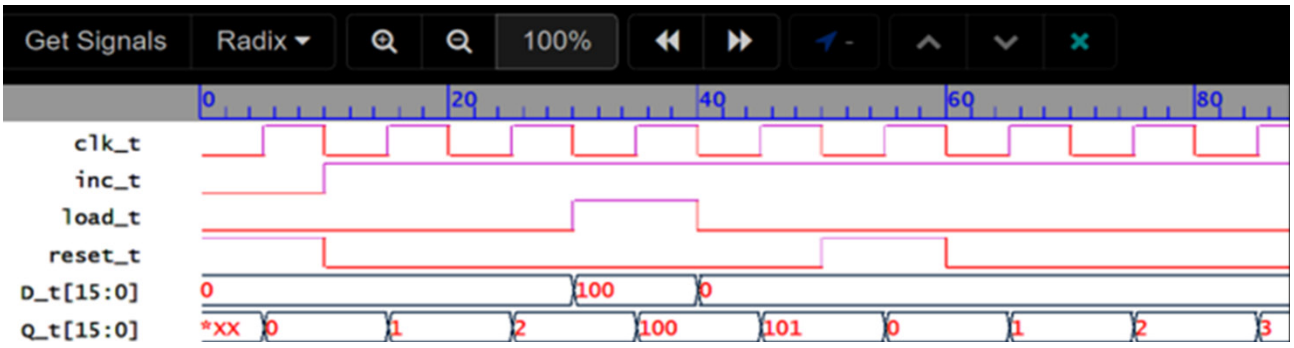


Figure 8. *EPWave testbench waveform for program counter example*

Simulation is ended after the 50ns of the last change in test signals. The simulation starts by resetting the program counter by the reset signal, after which it begins to increment its value by one at every positive transition of the clock signal. After three successive instructions, the program counter loads the jump address 0100_{16} from the D input and continues to execute the next instruction at address 0101_{16}. Next, a reset is performed after which the program counter continues to increment its value.

```
initial
  begin
         $dumpfile("dump.vcd");
         $dumpvars(0);
  end
```

Figure 7. *Collecting signals for EPWave diagram*

Results displayed in Fig 8, show that the design behaves as expected by the functional description of the program counter. Observation of program counter output, shows that is first reset to 0, after which starts to increment twice to value 2, after which it jumps to address 010016, then increments by one. Then, reset returns the value of the program counter to 0, after which it starts to increment. A manually driven testbench is only sufficient for testing simple designs. Complex design requires the use of test automation that can be loaded using test-vector files, or by using constrained random verification. EDA Playground also provides several verification methodologies, the most useful of which is UVM, which ensures complete automation of verification processes.

4. CONCLUSION

Usage of EDA tools significantly increases the productivity of the design of computer systems. These tools are becoming an important piece of technology with which future computer engineers should be familiar. Usage of EDA tools in education significantly improves the quality of practical exercises and improves students' skills which will be needed in further professional development. In this paper, we presented the methodology of design end testing of the program counter using the EDA Playground platform.

ACKNOWLEDGEMENTS

The research in this paper is part of the project 451-03-68/2022-14/200132 funded by the Ministry of Education, Science and Technological Development of the Republic of Serbia.

REFERENCES

[1] Association for Computing Machinery (ACM), IEEE Computer Society (2016), *Computer Engineering Curricula 2016, Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering*, ISBN: 978-1-4503-4875-1, DOI: 10.1145/3025098

[2] Lukovic, V, Pesovic U, (2018), Digital systems design course curriculum for Erasmus course, https://en.kg.ac.rs/courses/ftn/Digital%20Systems%20Design.pdf

[3] Pesovic U, (2018), VLSI systems design course curriculum for Erasmus course, https://en.kg.ac.rs/courses/ftn/VLSI%20System%20Design.pdf

[4] Nisan, N, Schocken, S, (2021), *The Elements of Computing Systems, second edition: Building a Modern Computer from First Principles 2nd Edition*, ISBN: 978-0262539807

[5] Doulos (2022), *EDA Playground Documentation*,

[6] EDA Playground Program counter example, https://www.edaplayground.com/x/CHk5 last access 15th July 2022.