Introduction to Teaching the Digital Electronics Design using FPGA

Richard Ravasz
Institute of Electronics and Photonics
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology
Bratislava, Slovakia
email: richard.ravasz@stuba.sk

Adam Hudec
Institute of Electronics and Photonics
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology
Bratislava, Slovakia
email: adam.hudec@stuba.sk

David Maljar
Institute of Electronics and Photonics
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology
Bratislava, Slovakia
email: david.maljar@stuba.sk

Robert Ondica
Institute of Electronics and Photonics
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology
Bratislava, Slovakia
email: robert.ondica@stuba.sk

Viera Stopjakova
Institute of Electronics and Photonics
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology
Bratislava, Slovakia
email: viera.stopjakova@stuba.sk

***Abstract*—This paper is focused on the design of digital electronics using Field Programmable Gate Arrays (FPGA) that can be easily applicable to school education process. At the beginning, paper is aimed at comparison of digital and analog electrical signals, and a brief introduction to digital electronics. The next section addresses an introduction and brief characterization of FPGA board Basys3. This board is suitable for education purposes as a teaching aid for students who would like to test their gained skills in hardware description languages or school subjects focused on the digital circuits design. Then, the design flow for digital circuits using Vivado design environment is described. The final section concludes the paper.**

***Keywords*—Field-Programmable Gate Array (FPGA), Hardware Description Language (HDL), Digital Electronics, Verilog**

I. Introduction

There are two basic ways of representing the electrical signal. The first way is analogue electrical signal, which is to express a numerical value as a continuous range of values between the two expected extreme values [1]. An example of an analog signal is the transmission of radio waves or sound waves [2]. The other possible way is digital electrical signal, which must have a finite set of possible values. Here, the numerical values are mostly represented by two limit values - logic zero (VSS) and logic one (VDD). The comparison of analog and digital electrical signals is shown in Fig.1. Generally, not all audio and video signals are analog. For example, HDMI for video and MIDI for audio are all digitally transmitted. Most communication between an integrated circuit is digital as well [3]. Interfaces like serial I2C and SPI, all transmit data via a coded sequence of square waves [4]. For digital circuits, the primary concern, in terms of functionality, is related to timing. In other words, digital design is concerned with the amount of additional delay due to the parasitic impedance associated with transistors and interconnects, as well as the effect this additional delay will have on operational frequency or meeting timing requirements [5].

The trend in digital design (and its teaching) is to use so-called Hardware Description Language (HDL), such as Verilog or VHDL to design logic circuits and to verify the design by downloading it into a FPGA development board [6].

II. Digital Circuit Design

A. Introduction to digital electronic

The basic elements of digital circuits are logic gates composed of transistor devices. Logic gates are generally designed using either BJT (Bipolar Junction Transistors) or FET (Field-Effect Transistors) devices. Logic circuits can be fabricated in various technologies but the most spread is CMOS (Complementary Metal Oxide Semiconductor) technology. Usually, logic circuits have several inputs and a single output. The digital information is processed and stored in binary form, where output from a digital circuit will be either logic 1 (True) or logic 0 (False) depending on the combination of its inputs. Generally, digital circuits can be of two main types – **Combinatorial logic** or **Sequential logic**.

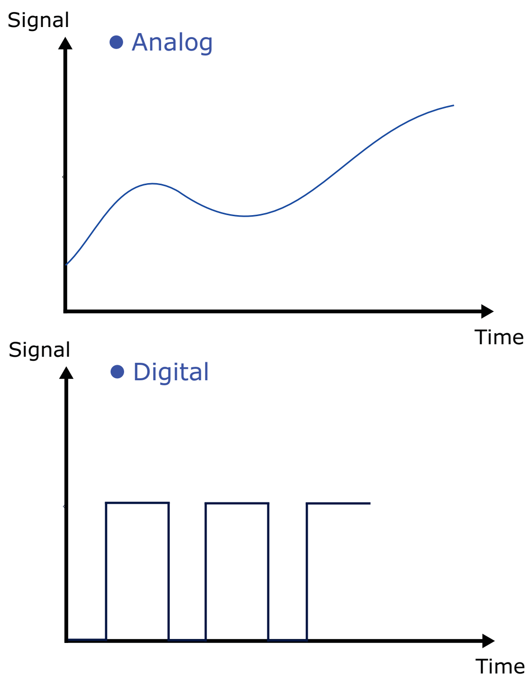


Fig. 1: Digital vs analog electrical signal.

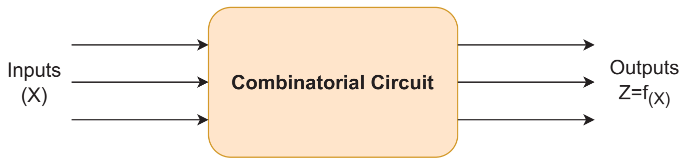


Fig. 2: Combinatorial logic

1) Combination logic: Combinatorial or combinational logic circuits are usually expressed by Boolean algebra and truth table [7]. Basic logic operations (as AND, OR, NAND, etc.) can be physically represented as logic gates, which are essentially a specific connection of transistors. Logic gates have ability to realize every combinational logic functions. The functional behavior of a logic gates can be described by the following:

1) **Truth table** - is prescribed specification table that explains the input–output relation for all possible combination of inputs,
2) **Logic Equation** - in logic equation, the output is expressed as a Boolean function of inputs according to the truth table. Logic equation are generally unique as truth table because they depend on the proposed applications,
3) **Timing diagram** - timing diagram of logic gate indicates the variation of output waveform with respect to the input waveform.

An example of a logic gate is depicted in Fig.3. NAND gate is expressed using its truth table, where inputs are labeled as A and B, while the output is labeled as Z. Fig.3 also shows a schematic diagram of NAND gate as well as physical implementation of NAND gate on transistor level in CMOS technology [8].

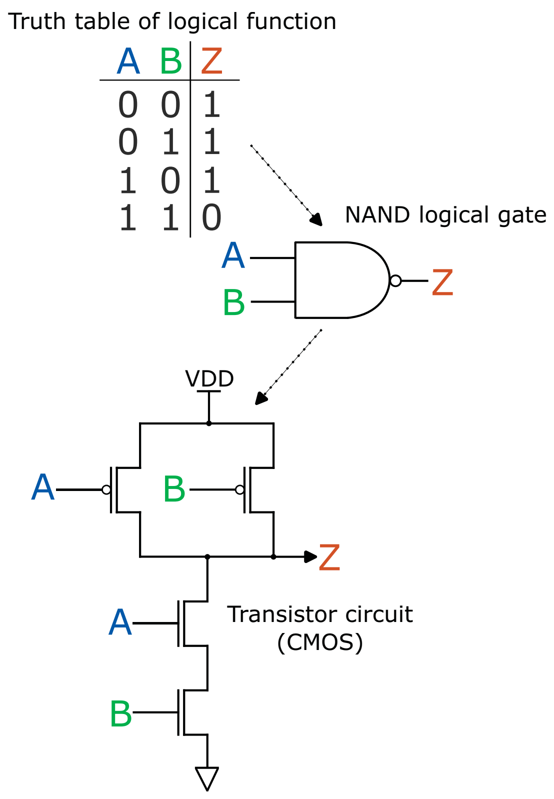


Fig. 3: Combinational logic

2) Sequential logic: Unlike combinational logic circuits that change its state only depending on the actual logic values being applied to their inputs at the time, the output of sequential logic is given by the actual state of its inputs and also by the current state of the circuit. The current state is stored in memory elements [9]. The schematic diagram of a sequential circuit is depicted in Fig.4.

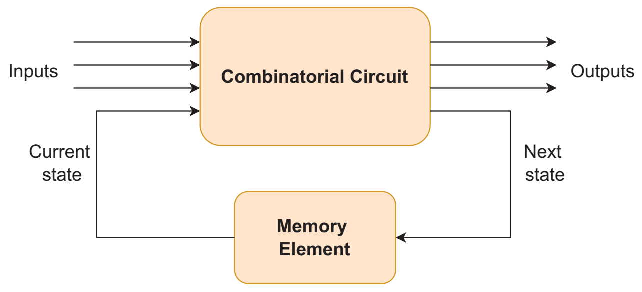
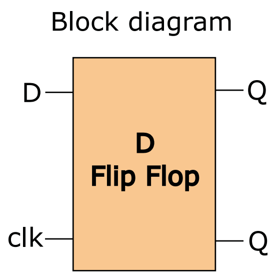


Fig. 4: General block diagram of a sequential logic

There are two types of sequential circuits: Asynchronous sequential circuits and Synchronous sequential ones. **Asynchronous sequential circuits** do not use a global synchronizing signal but uses a local synchronization between sender and receiver blocks. These circuit are faster than synchronous sequential circuits. Therefore, they are used in applications where speed of operation is the highest priority and independent of internal clock pulse. However, design tools and support for asynchronous circuits are not so wide-spread, these circuits are more difficult to design and analyze, and they usually require more design knowledge and effort, and overall fabrication expenses are rather high. On the other hand, **synchronous sequential circuit** uses a global synchronizing signal called clock. The output is changed depending on speed of the clock signal. Memory elements can be sensitive either on edge (positive/rising or negative/falling) of the clock signal or level (high or low) of the clock signal [10]. Therefore, we distinguish two types of the memory elements: Flip-Flops or Latches. There is a variety of flip-flops and latches available that differ in their behavior depending on its construction, the value on its inputs and the way of using the clock signal [11]. For example, RS Latch, RS Flip-Flop, JK Flip-Flop, T Latch, D Flip-Flop (DFF) and others are available. There are many parameters that specify the flip-flop circuit performance, such as setup time and hold time, delay, layout area, power dissipation or leakage current [12]. The block diagram and truth table of D Flip-Flop is depicted in Fig.5. In this case, DFF is sensitive to the rising edge of clock signal. If a rise edge of the clock signal occurs, the output of DFF changes depending on the value of its input. In other case, the output of DFF does not change and keeps the previous state [13].



Truth table

clk	D	Q	Description
↓	X	Q	*Memory no change*
↑	0	0	Q → 0
↑	1	1	Q → 1

Fig. 5: Description of DFF operation

B. *HDL - Hardware Description Languages*

Digital circuits can be analyzed at different levels of abstraction. For example, a central processing unit (CPU) can be interpreted as a sea of transistor devices but much easier way how to describe more complex digital blocks is organize transistors to logic gates, then, logic gates to adders or registers, then registers to memory, and so forth. This virtual approach to describing digital circuits is more intuitive but it becomes impractical as complexity of digital systems increases. Another way to describe digital circuits is to use a textual language called Hardware Description Language (HDL). The most popular hardware description language are **Verilog** and **VHDL** [14]. HDL design methodology is extremely important for designers when developing modern digital systems. Once you have learned Verilog (System Verilog) or VHDL, you will be able to design a digital system much faster than by manual drawing the complete schematic of the system in a schematic editor. The most important thing to remember when you are writing a HDL code is that you are describing real hardware, not writing a computer program [15]. Compared to programming languages (like Python, C, C++, etc.), where created programs are execute line by line, HDLs describe the behavioral digital hardware. Although the code of described circuit is written top-to-bottom, the hardware and its processes can operate simultaneously.

C. *Field-Programmable Gate Arrays*

Field-Programmable Gate Arrays (FPGAs) are semiconductor pre-created structures that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reconfigured to the required application functionality after manufacturing process. This feature makes FPGAs different from Application Specific Integrated Circuits (ASICs), which are full-custom designed and manufactured for a specific application [16]. The basic architecture of FPGA with the Input/Output Block, Programmable interconnects and (configurable) Logic blocks is shown in Fig.6 [17]. FPGAs are widely used for different application purposes such as circuit prototyping, automotive, aerospace and defense, medical devices, video/image processing, etc.

In the next paragraphs, the main block components of FPGA are shortly described.

Configurable Logic Blocks : A CLB is the fundamental component of each FPGA, allowing the user to implement any logical functionality within the chip [18]. CLB includes combination logic circuits, flip -flops and look-up tables (LUTs) that can be configured into several different logic functions. CLB contains arithmetic carry logic and multiplexers to create logic functions. The basic structure of CLB is shown in Fig. 7.

Input/Output Blocks : I/O block is a programmable input and output unit, which is an interface between FPGA and external circuits. Hence, data from the outside world is acquired through inputs pins. On the ather hand, the output from FPGAs is fed to the outside word using output pins. For examle, the ARTIX-7 XC7A35T FPGA has I/O pins that can operate on standard voltage values from 1.2 V to 3.3 V. The FPGA Basys3 board contains 106 such input/output pins.

Programmable interconnects : Programmable interconnects are collection of wires and programmable switches. These are responsible for interconnecting the CLBs and other building blocks within FPGA. Interconnect blocks is also called routing channels.

D. *Development boards*

Important part of the digital circuit design process based on HDL methodology is design verification using a development board. One of the available board suitable for education purposes is Basys 3. The Basys3 board is developed by Digilent Inc. and have the Xiling Artix-7 FPGA chip onboard which has 33.280 logic cells in 5200 slices (each slice contains four 6-input LUTs and 8 flip-flops). This board is suitable for education purposes since it has peripheral module (Pmod) connector, four-digit seven-segment display, 16 slide switches, 16 LEDs, and five push buttons. The FPGA is programmed to required function. Then, the configuration file will be generated by design environment, e.g. Vivado design suite. The generated

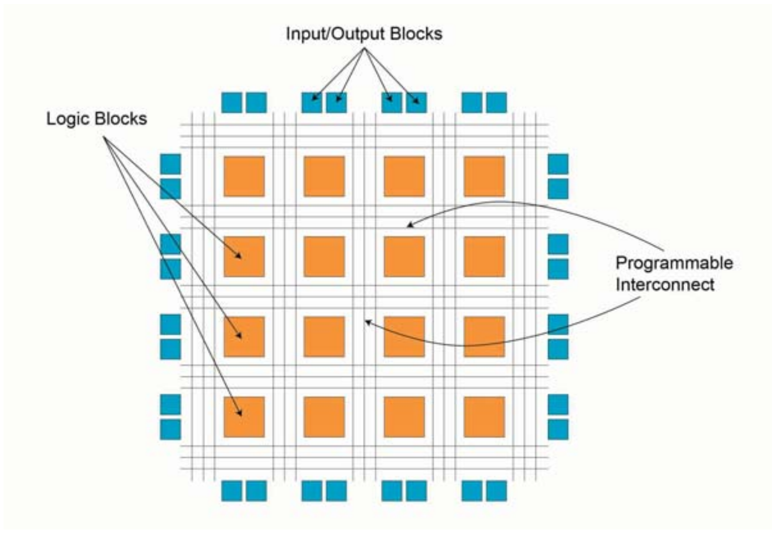


Fig. 6: General architecture of FPGA [19]

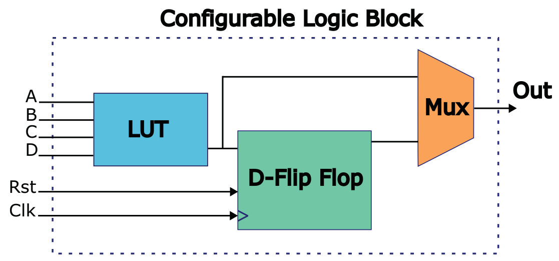


Fig. 7: Block diagram of CLB

configuration file can be fed to Basys3 in three ways. The first method is using the shared UART/JTAG USB port. We will use this method while configuration the FPGA through Vivado. The second method is using the SPI flash, where the configuration file should have been stored in the flash. Finally, in the third method, the configuration file is stored in a USB stick through the USB host connector. There is also a VGA connector available on the board that allows 12-bit data transfer to a VGA display device. The Basys3 board has also an onboard oscillator working at 100 MHz. The clock signal is generated by the oscillator that is fed to the Artix-7 FPGA through a dedicated pin [20].

III. FPGA Design Flow

Vivado Design Suite is a software tool produced by Xilinx for synthesis and analysis of HDL-based designs. Vivado environment includes the in-built logic simulator as well. Moreover, Vivado also introduced a high-level synthesis tool with a tool-chain that converts C code into a programmable logic. Student, professors and researchers can download the HL WebPack Edition for free.

The FPGA design flow includes of several different steps like: design source file creation, design synthesis, design implementation, a bit-stream generation and a process uploading the bit-stream into FPGA board. The block diagram of the whole design flow is shown in Fig.9.

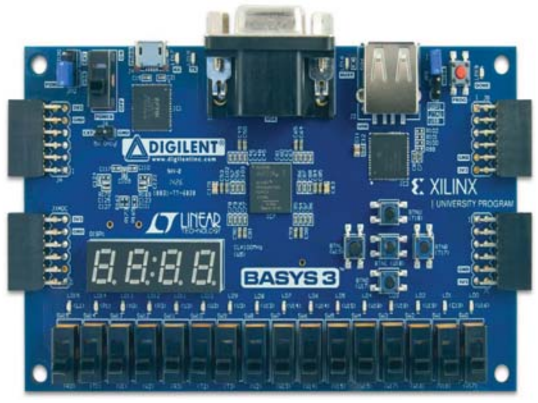


Fig. 8: Basys 3 FPGA development board

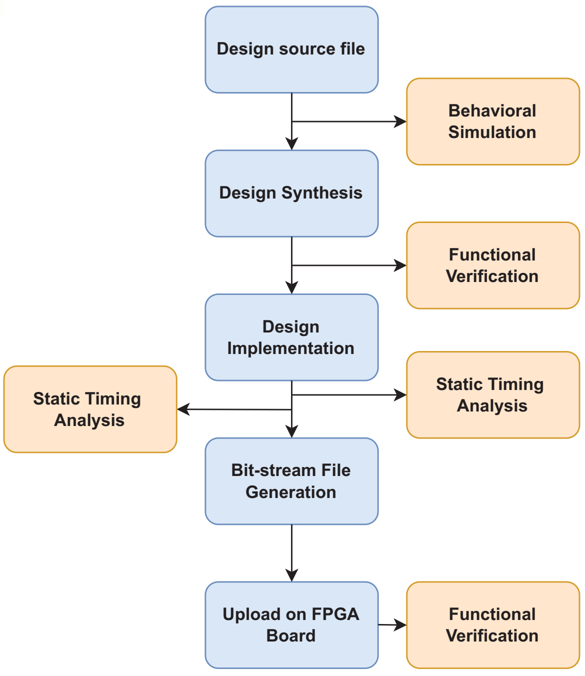
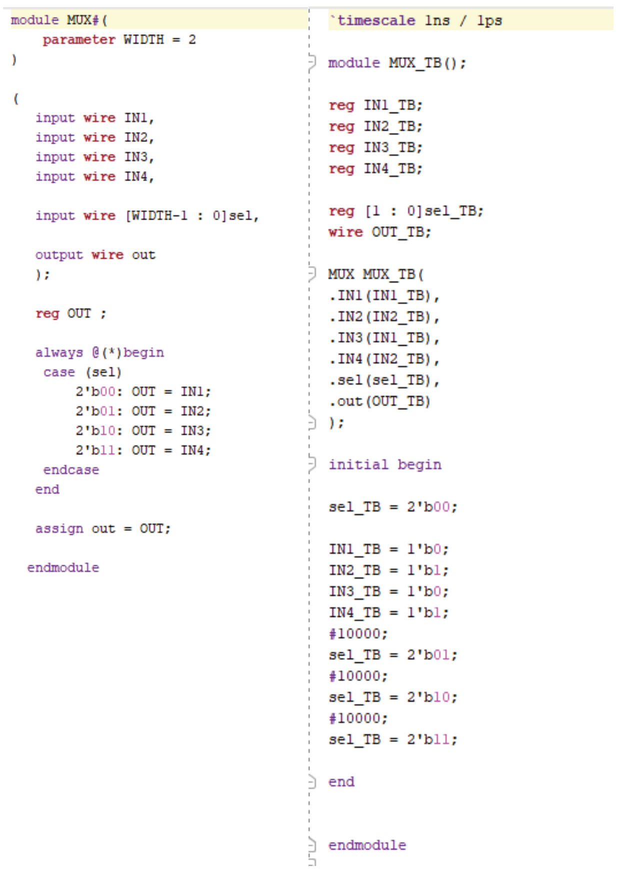


Fig. 9: Digital design flow for FPGA

Design source files can be created by various techniques, such as schematic editing or through HDL coding [21]. An example of a design source file representing a 4-channel multiplexer written in Verilog HDL is depicted in Fig. 10 (a). The multiplexer schematic diagram is shown in Fig.11. Every digital block described by Verilog must starts with keyword *module* and end with keyword *endmodule*. An *always* block in one of the procedural blocks in Verilog. Statement inside an always block are executed sequentially. In order to be able to use module with a different specifications, we parametrized this module. In order to be able simulating the design described in HDL Verilog, it is necessary to create a Testbanch. As an example, a testbench for simulation of MUX is depicted in Fig. 10 (b). In this file, we will instantiate the design source file (MUX) and specify the input stimuli that will help us verify the functionality of designed digital block. After creating the testbench, design is ready for simulation process.



```
module MUX#(
    parameter WIDTH = 2
)

(
  input wire IN1,
  input wire IN2,
  input wire IN3,
  input wire IN4,

  input wire [WIDTH-1 : 0]sel,

  output wire out
  );

  reg OUT ;

  always @(*)begin
   case (sel)
       2'b00: OUT = IN1;
       2'b01: OUT = IN2;
       2'b10: OUT = IN3;
       2'b11: OUT = IN4;
   endcase
  end

  assign out = OUT;

 endmodule
```

```
`timescale 1ns / 1ps

module MUX_TB();

reg IN1_TB;
reg IN2_TB;
reg IN3_TB;
reg IN4_TB;

reg [1 : 0]sel_TB;
wire OUT_TB;

MUX MUX_TB(
.IN1(IN1_TB),
.IN2(IN2_TB),
.IN3(IN1_TB),
.IN4(IN2_TB),
.sel(sel_TB),
.out(OUT_TB)
);

initial begin

sel_TB = 2'b00;

IN1_TB = 1'b0;
IN2_TB = 1'b1;
IN3_TB = 1'b0;
IN4_TB = 1'b1;
#10000;
sel_TB = 2'b01;
#10000;
sel_TB = 2'b10;
#10000;
sel_TB = 2'b11;

end


endmodule
```

Fig. 10: Source file in Verilog HDL

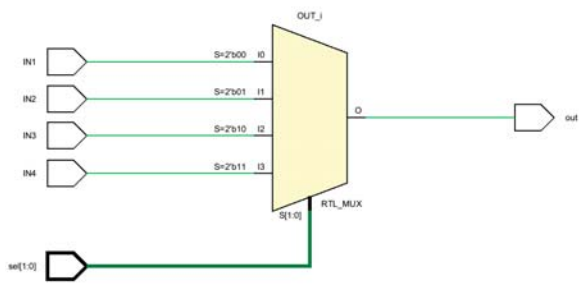


Fig. 11: Schematic diagram of MUX

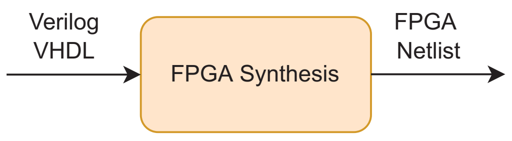


Fig. 12: Synthesis in HDL-based digital design

Simulation is a process of applying input stimuli to the designed digital block at time in order to check if the code behaves in the intended way. Simulation is a suitable technique to verify the robustness of design as well. The example of simulation data (input and output waveforms/signals) of designed MUX is depicted in Fig.12. The inputs are set to logic 1 or logic 0 according to a respective MUX truth table, and these stimuli are fed to MUX inputs. After that, the value on selected input pins labeled as $selTB$ is set, which switches one of the MUX inputs of the output pin. It can be observed that MUX works as expected.

Synthesis and functional verification is a process of converting high-level logic design written in HDL into a gate-level representation or logic component 13 schematics in FPGA. Synthesis tool typically generates netlist and a bitstream for FPGA code upload [22]. The netlist is device-independent and can be available in the standard form like electronic design interchangeable format (EDIF). The functional verification is the process of ensuring that RTL description conforms to a specification [23].

Implementation During the design implementation, the EDA tool translates the design into the required format and maps it on to the FPGA according to a set of optimization constraints, e.g. required area or operation speed/timing. The mapping is performed by the EDA tool uses the actual logic cells or macro-cells. Designed MUX after the implementation process is shown in Fig.14. During the mapping, the EDA tool uses the macro-cells, programmable interconnects and IO blocks. All the necessary blocks are placed on the predefined geometry inside the FPGA and routed using the programmable interconnects for the intended functionality. This step is called Place&Route.

A. *Design Constraints*

Constraints are used to influence the FPGA design implementation process including the synthesizer and Place&Route tool. They allow the designer to specify the design performance requirements and guide the implementation process toward meeting the requirements. We know four primary types of constrains: synthesis, I/O, timing and are/location constraints [24].

Synthesis constraints influence how the synthesis tool translate HDL design source into Register-Transfer-Level (RTL) representation. I/O constraints (also commonly referred as pin assignment) are used to assign the signal to a specific I/O pin or I/O bank.

Timing constraints are used to specify the timing characteristic of the design. Timing constraints may affect to all internal timing interconnections, delays through logic and LUTs, and timing between flip-flops or registers.

Area constraints are used to map specific circuitry to a range of resources within the FPGA. Location constraints specify the location either relative to another design element or to a specific fixed resources within the FPGA.

IV. Conclusion

The main purpose of this paper was to present an introduction to teaching the design of digital circuits and systems using HDL methodology that offers a possibility to describe logic circuits at higher levels of abstraction and consequently, synthesize and implement them into FPGA. This design approach brings several advantages especially interesting for education purposes. Advantages of HDL-based design methodology mainly include: design of different blocks on various levels of abstraction, technology and tool independence, easy re-design

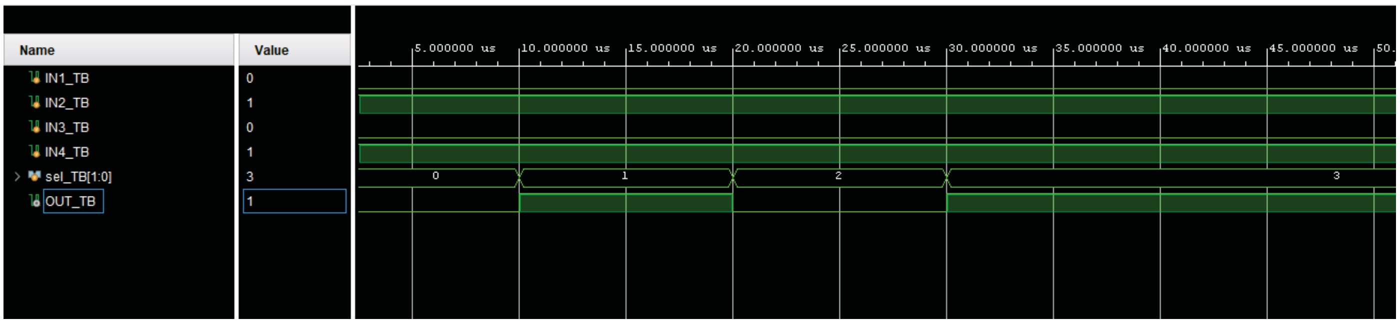


Fig. 13: Simulation of MUX circuit

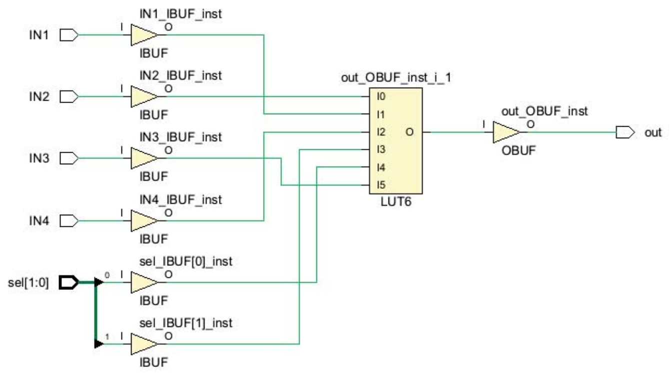


Fig. 14: MUX after implementation into FPGA

and re-use, modular design, productivity, fast examination of design alternatives, and many others. The main part of the paper is focused on digital design flow using Vivado design environment with link to FPGA Basys 3 development board. This board is suitable for education purposes thanks to a teaching aid for students who would like to test their skills in HDL-based design of digital circuits or other related subjects. Using the development board, students can learn how to work with 7-segment display, LEDs, switches and many other interfaces that can lighten up teaching the digital circuit design at schools.

ACKNOWLEDGMENT

This work was supported in part by the Slovak Research and Development Agency under grant APVV-19-0392 and the Ministry of Education, Science, Research and Sport of the Slovak Republic under grants VEGA 1/0760/21 and VEGA 1/0731/20. This work has also received funding from the Electronic Components and Systems for European Leadership Joint Undertaking under grant agreement No 876868. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Slovakia, Netherlands, Spain, Italy.

REFERENCES

[1] B. Razavi, *Design of Analog CMOS Integrated Circuits*. McGraw-Hill, first ed., 2001.
[2] S. J. Ferguson and R. Hebels, *Computers for librarians: An introduction to the electronic library*. Elsevier, 2003.
[3] JIMBLOM.
[4] A. K. Maini, *Digital electronics: principles, devices and applications*. John Wiley & Sons, 2007.
[5] J. C. Kemerling, R. Greenwell, and B. Bharath, "Analog-and mixed-signal fabrics," *Proceedings of the IEEE*, vol. 103, no. 7, pp. 1087–1101, 2015.
[6] G. Donzellini and D. Ponta, "A bottom-up approach to digital design with fpga," in *2011 IEEE International Conference on Microelectronic Systems Education*, pp. 31–34, IEEE, 2011.
[7] A. Agarwal and J. Lang, *Foundations of analog and digital electronic circuits*. Elsevier, 2005.
[8] D. De, *Basic electronics*. Pearson Education India, 2010.
[9] W. Storr, "Sequential logic circuits and the sr flip-flop," Aug 2022.
[10] "Introduction of sequential circuits," May 2022.
[11] D. Astels, "Digital circuits 4: Sequential circuits."
[12] O. Bondoq, K. Abugharbieh, and A. Hasan, "A d-type flip-flop with enhanced timing using low supply voltage," in *2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–4, IEEE, 2020.
[13] P. Scherz and S. Monk, *Practical electronics for inventors*. McGraw-Hill Education, 2013.
[14] "What is a hardware description language (hdl)? - technical articles."
[15] D. M. Harris and S. L. Harris, "Hardware description languages," *Digital Design and Computer Architecture*, pp. 170–235, 2022.
[16] "What is an fpga? field programmable gate array."
[17] V. Taraate, "Digital logic design using verilog," *Pune: Springer India*, 2016.
[18] N. Eastland, "Fpga – configurable logic block," Jun 2021.
[19] Admin, "Fpga design, architecture and applications [2022]," Jul 2022.
[20] C. Ünsalan and B. Tar, *Digital system design with FPGA: Implementation using Verilog and VHDL*. McGraw-Hill Education, 2017.
[21] "The ultimate guide to fpga design flow," May 2022.
[22] "Understanding fpga synthesis," Dec 2020.
[23] "Functional design and verification," Oct 2019.
[24] R. Cofer and B. F. Harding, *Rapid System Prototyping with FPGAs: Accelerating the Design Process*. Elsevier, 2006.