FBU-CPU_Project

Marouan Boumchahate, Ousama Jamal Eddin, Norris Nishimwe, MHD Labib Almohammad

Istanbul Türkiye

e-mail: {marouanboumchahate@gmail.com - ossama.jamal.aldien@gmail.com - Nishimwenorris18@gmail.com - Moe.almhmd.business@outlook.com }

Summary:

Verilog langauge is an intresting language and such a powerful one, CPU's and FPGA's were built using this specefic language. For students and specially University students that are trying to learn Verilog language, using FPGAs is the way, but since FPGAs can be super expensive and time consuming, a simulators were developed to help people attain their learning objectives efficiently such as implementing CPUs/FPGAs. And our task (Project) for this term is to implement a CPU that mirrors the functionality of a real CPU. The key objectives include ensuring seamless operation and meticulous execution of the CPU's components – from fetching to decoding and executing – all in the correct order. This project reinforces the understanding of Verilog language and how CPUs work.

[Project's Link] -: https://github.com/OssamaJamalEddin/FB-CPU-Project-MONM-GROUP-.git

[Project's Slides Link] -: https://github.com/OssamaJamalEddin/Project_Slides.git

[Project's Youtube Video] -: https://youtu.be/FGRtR2mGu44

I - Introduction

* The project is about CPUs and how they really work. We made a little one using an FPGA simulator that our lecturer, Mr. Levent, built. We used the Verilog language to construct a CPU and its parts. The whole purpose of this project is having eventually the knowledge of how CPUs work and how they are built with knowing what are the CPUs parts ofcourse and their task.

II - System Architecture

* We have only used an FPGA simulator that Mr.Levent developed. in addition to that, FPGA simulators are crucial tools for verifying and debugging designs before actual hardware implementation. They come in two types—behavioral, focusing on functionality, and timing, considering timing characteristics—supporting languages like VHDL, **Verilog**, and SystemVerilog. Popular tools include ModelSim, XSIM, and VCS. Key features include **waveform viewing**, interactive debugging, code coverage, and performance profiling. Simulators often support co-simulation and enable virtual prototyping for system-level testing. The choice depends on design language, required features, and the FPGA vendor.

III - Software Used

This is the main screen of the simulator used to create our project:

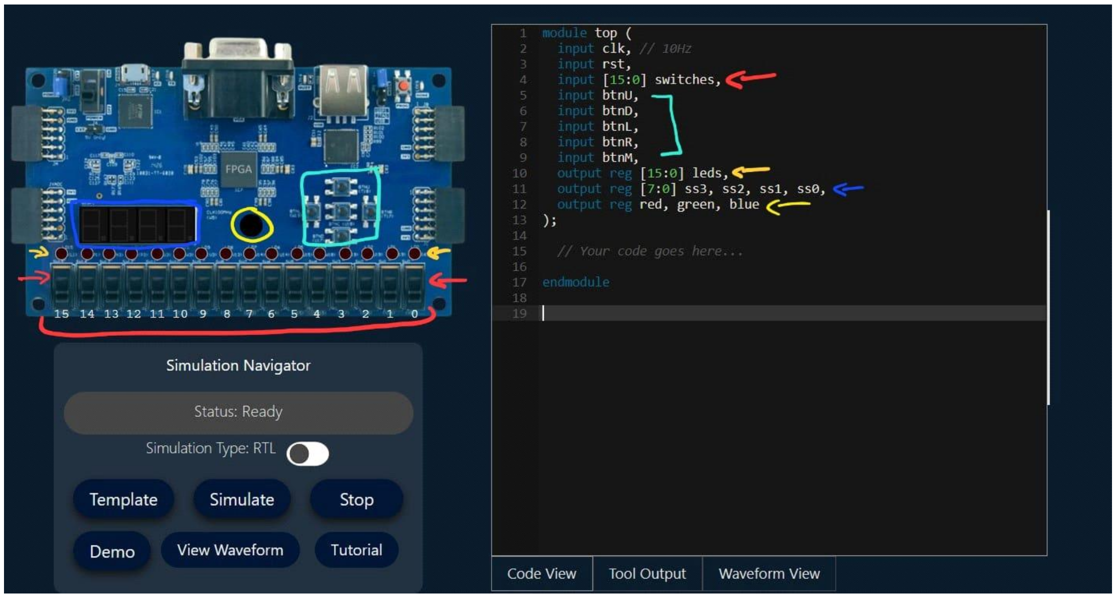
Red -> 16 Switches used as inputs

Cyan -> 5 Buttons used as inputs

Blue -> Numbers of 8-bits can be shown on the screen

Yellow -> RGB output lights

Orange -> 16 leds used most likely as outputs



```
module top (
  input clk, // 10Hz
  input rst,
  input [15:0] switches,
  input btnU,
  input btnD,
  input btnL,
  input btnR,
  input btnM,
  output reg [15:0] leds,
  output reg [7:0] ss3, ss2, ss1, ss0,
  output reg red, green, blue
);

  // Your code goes here...

endmodule
```

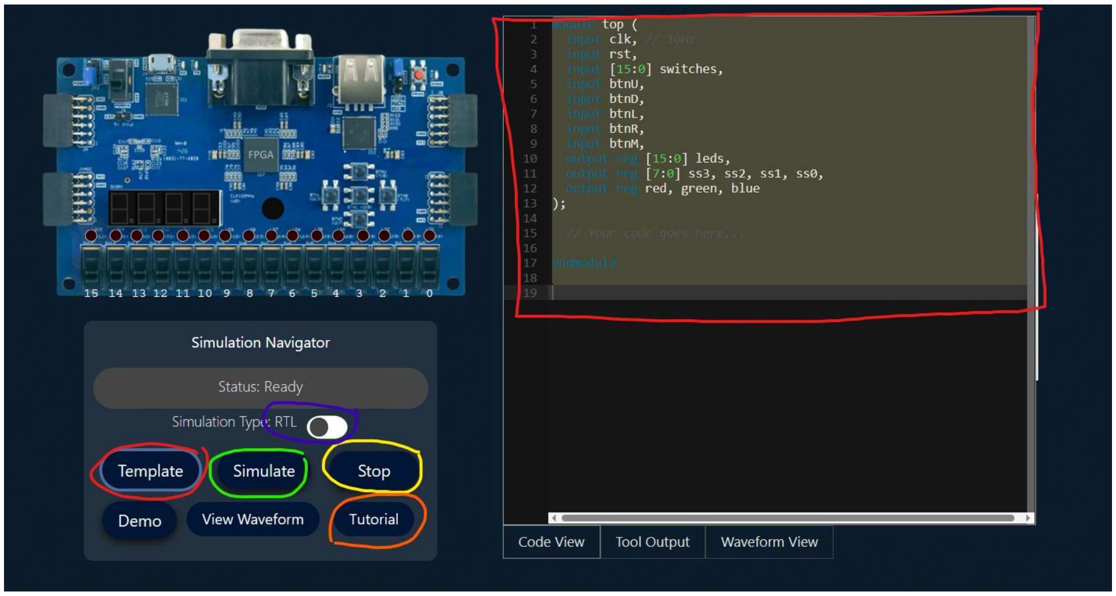
Red -> if we want to reset the simulator after some coding, we can hit (**Template**), and it will bring back the main module (**module top**) of the simulator and we can write again our new code exactly right where (**Your code goes here**)

Green -> starts our simulator by clicking on (**Simulate**)

Yellow -> **Stops** the simulator

Orange -> Starts a **Tutorial** of how to use the simulator

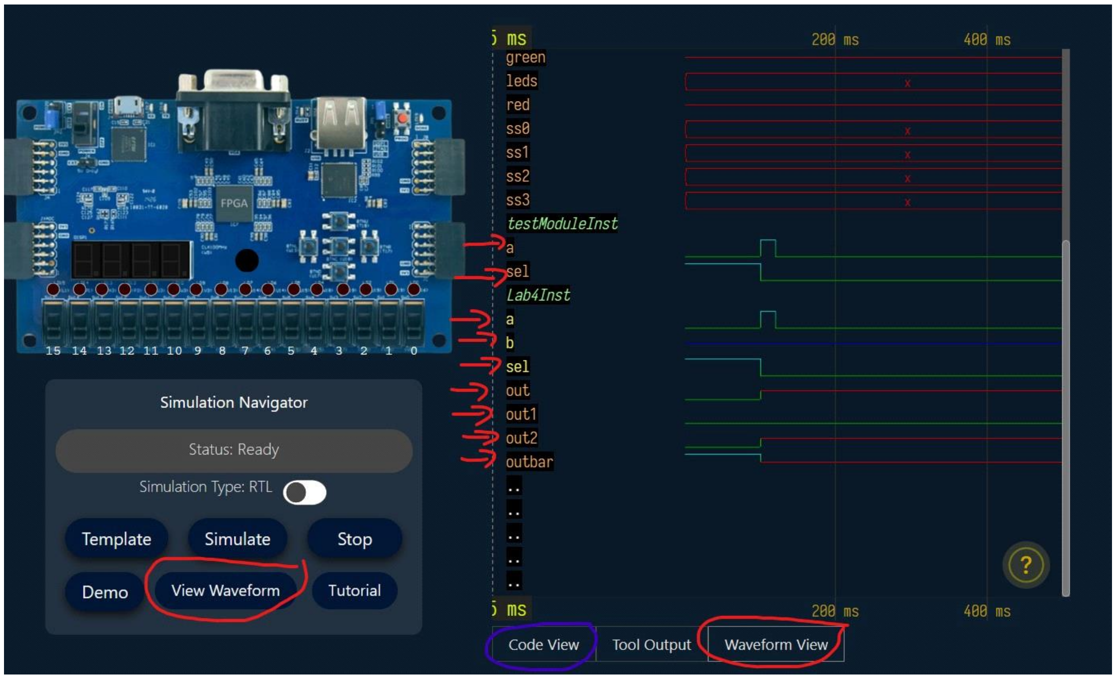
Blue -> swapping between the simulatio types -(**RTL** - **Mapped**)-



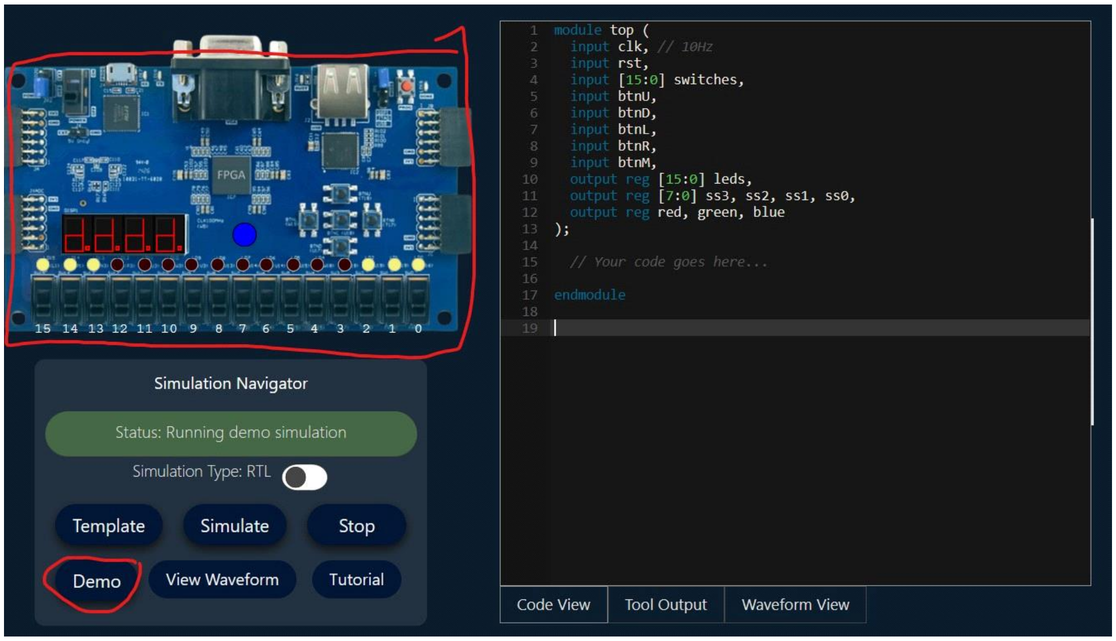
```verilog
module top (
  input clk, // 10Hz
  input rst,
  input [15:0] switches,
  input btnU,
  input btnD,
  input btnL,
  input btnR,
  input btnM,
  output reg [15:0] leds,
  output reg [7:0] ss3, ss2, ss1, ss0,
  output reg red, green, blue
);

  // Your code goes here...

endmodule

```

Red -> After writing some code, we can check our inputs and outputs behavior by clicking on (**View Waveform**), as shown, we **can monitor our values**

Blue -> By clicking on (**Code View**), we can go back to our code



Red -> Lastly, by clicking on (**Demo**) we can check how the simulator works when we use its **tools**, and we can press as well (***Stop***) to stop it.



```
module top (
  input clk, // 10Hz
  input rst,
  input [15:0] switches,
  input btnU,
  input btnD,
  input btnL,
  input btnR,
  input btnM,
  output reg [15:0] leds,
  output reg [7:0] ss3, ss2, ss1, ss0,
  output reg red, green, blue
);

  // Your code goes here...

endmodule

```

IV - Results

* Our project has provided us with a profound understanding of the intricate workings of a CPU, unraveling the functionalities of crucial components like the PC, IR, MAR, MDR, C-Unit, ACC, and AUL. In addition to having a decent understanding about Verilog language, FPGA and FPGA simulators.

Project Team:

1- Marouan Boumchahate's **CV**: https://github.com/marouan-boumchahate/CV.git

2- Ousama Jamal Eddin's **CV**: https://github.com/OssamaJamalEddin/CV.git

3- Norris Nishimwe' **CV**: https://github.com/OssamaJamalEddin/Norris-CV.git

4- MHD Labib Almohammad' **CV**: https://github.com/OssamaJamalEddin/Moe-CV.git

Refrence Files and Links:

1- { **Youtube Videos** }

I - First: https://www.youtube.com/watch?v=jFDMZpkUWCw

II - Second: https://youtu.be/04UGopESS6A?si=xo_Yzm50hYdOhMid

2- { **Logical System Design Classes's Slides** }

I - < Verification >

II - < Memories (BRAM) >

III - < Sequential Circuit >

IV - < Machine State >

-> Github Link for the slides:- https://github.com/OssamaJamalEddin/SystemDesignSlides.git

3- { **Fundemental of Comp Engineering Slides as -> Pictures** (LAST YEAR'S SUBJECT) }

-> Github File's Link: https://github.com/OssamaJamalEddin/FundementalSlides-OLD-.git