4-bit Calculator

Team E

16.06.23

Team members:

- Amit Chakma
- Md Limon Apu
- *Yashodhan Vishvesh Deshpande*

1 Introduction

Building a 4-bit unsigned calculator is the subject of our semester's assignment. Various mathematical operations, such as binary addition, subtraction, multiplication, and division, are to be implemented. To do this, we combine many modules, including Adder, Subtractor, Multiplier, and Divider, using sub-modules that use simple gates. Following the design and coding of the functionality in VHDL, the calculator is next constructed by translating the schematics to a PCB layout. We go into detail at every stage about the software used in the related study. Due to its programmability and capacity for numerous testings, we decided to utilize FPGA for this project. The components are configured using VHDL which is a hardware description language.

2 Concept description

1. **Block diagram of our application**

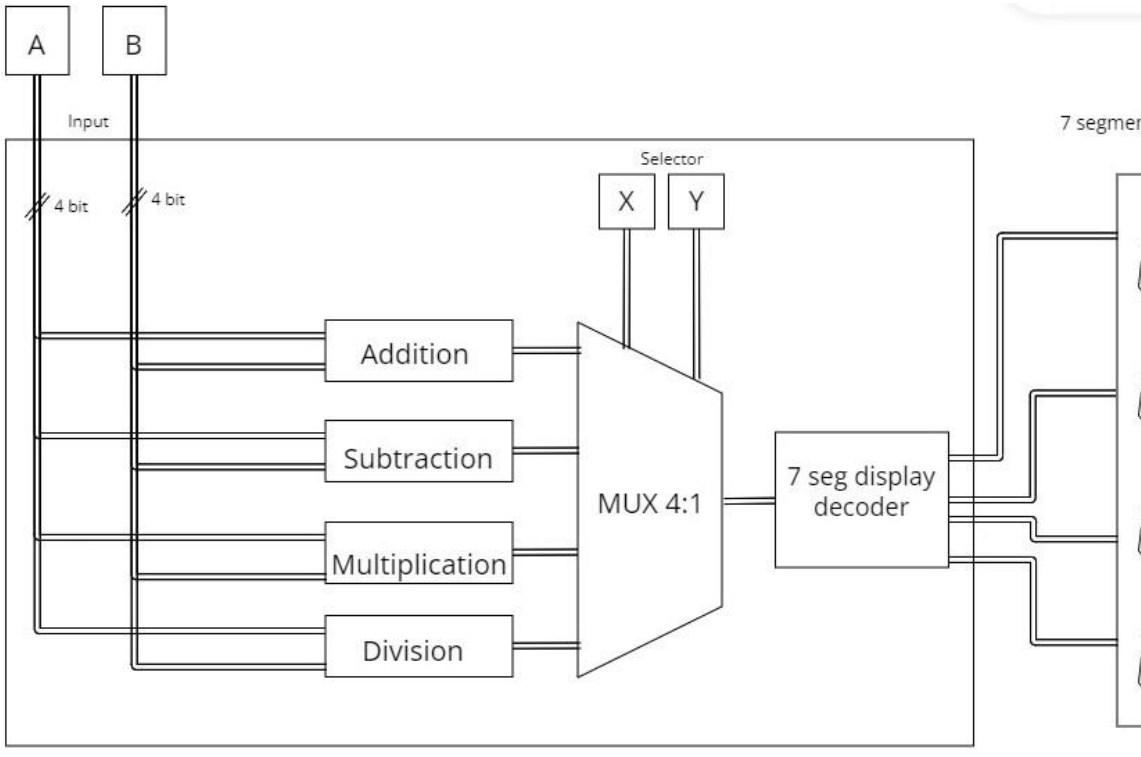


Fig.1: Main function calculator

2. **Application of our project:**

The 4-bit calculator system consists of input slide switches on the PCB, through which binary values are provided. These inputs are then processed by individual blocks for multiplication, addition, subtraction, and division, generating the corresponding results. The outputs from these blocks serve as inputs to a 4:1 multiplexer, which selects the appropriate driver line based on 1-bit binary selector inputs, indicating the desired operation. The selected output is then decoded by a decoder, converting it into decimal values for display on the 7-segment displays. The calculator's main purpose is to perform basic arithmetic operations, including addition, subtraction, multiplication, and division.

3 Project/Team management

Our project follows an iterative development model, allowing us to work on and test our scenario in a step-by-step manner. This approach enables us to revisit previous steps, make necessary adjustments, and move forward based on our findings. To ensure an organized workflow and effective communication, we divided specific tasks among team members and held weekly meetings to discuss progress and plan future steps. During the implementation phase, we prioritized in-person collaboration whenever possible to work collectively towards achieving our project objectives. Below is a documentation of each group member's progress, highlighting the dates when meetings took place. This systematic approach helped us stay on track and successfully accomplish our project goals.

	Date	25.05.23	31.05.23	8.06.23	16.06.23
	Task	Task-1	Task-2	Task-3	Task-4
Md Limon Apu	To-do	Research on project	Research on VHDL	Implement in Xillix	Connect to FPGA
	Status	Done	Done	Done	Done
Yashodhan Vishvesh Deshpande	To-do	Research on project	Research on Schamtic	Implement in Kicad	Schamatics
	Status	Done	Done	Done	Done
Amit Chakma	To-do	Research on project	Research on VHDL	Implement in Modelsim	Simulation
	Status	Done	Done	Done	Done

Fig.2: Workload of group member

4 Technologies

- **VHDL**

VHDL (VHSIC Hardware Description Language) is a powerful hardware description language used for modeling the behavior and structure of digital systems. It allows designers to describe and specify complex digital circuits at various levels of abstraction, from system-level designs to individual logic gates. VHDL is commonly used for design entry, documentation, and verification purposes.

During the implementation phase of our project, we utilized ModelSim, a software tool developed by Siemens (previously Mentor Graphics), which is specifically targeted for Intel® FPGAs devices. ModelSim provides a multi-language environment and serves as a valuable tool for developing VHDL code. It offers features such as syntax checking, compilation, and most importantly, simulation capabilities. The user-friendly interface of ModelSim allows for

efficient code development and simulation, aiding in the verification and testing of our design.

By using VHDL and ModelSim in our project, we were able to effectively design, develop, and test our digital system, ensuring its functionality and correctness before proceeding with further stages of implementation.

- **FPGA**

The Xilinx Nexys A7 FPGA board, powered by the Xilinx Artix-7 FPGA, serves as the core hardware component for our calculator project. The Artix-7 FPGA offers a programmable and versatile platform capable of executing a wide range of logical operations. Unlike fixed components like ASICs, the FPGA provides the advantage of reprograming ability, allowing us to update and test our code multiple times during the development process. The specific FPGA used, EP4CE22E22C8N from Intel's Cyclone IV E family, features 22320 cells utilizing 60nm technology and operates at a voltage of 1.2V. With its extensive capabilities and flexibility, the Xilinx Nexys A7 FPGA board enables us to implement and validate our calculator design efficiently on a physical PCB.

- **KiCAD**

KiCAD is a powerful electronic design automation (EDA) software that we employed for our project to design the printed circuit board (PCB). With its comprehensive set of tools, KiCad facilitated seamless integration of the FPGA module, input switches, and 7-segment displays into our PCB design. We followed a four-layer PCB approach, optimizing signal routing and simplifying the overall design. By utilizing KiCAD's features for schematic capture, component placement, and PCB routing, we were able to create an efficient and well-structured PCB layout. The software's extensive library content further aided in the selection and placement of components, ensuring proper electrical connectivity. Overall, KiCad played a crucial role in enabling the successful design and realization of our project's PCB.

- **Vivado**

In our project, we had utilized Vivado, a software tool provided by Xilinx, which greatly aided us in implementing and programming the FPGA module for our calculator system. Vivado served as a comprehensive and user-friendly development environment, offering a wide range of features that were essential for our FPGA design. We took advantage of its RTL synthesis, simulation, and place-and-route algorithms to optimize our design for optimal performance, power efficiency, and efficient resource utilization. Vivado's intuitive graphical interface allowed us to easily configure the FPGA, assign pins, and generate the necessary programming bitstream. We also greatly appreciated its robust verification and debugging capabilities, which allowed us to thoroughly test and validate our design before deploying it onto the FPGA. Vivado played a pivotal role in the successful development and programming of our FPGA module, ensuring a seamless integration into our calculator project.

5 VHDL and FPGA Implementation

In our project, we embarked on a systematic approach by initially coding and testing essential components, including the Half Adder, Full Adder, and Ripple-Carry Adder. With dedicated testbenches, we ensured their accurate functionality. Progressing further, we developed larger modules such as the 4-bit Adder, Subtractor, Multiplier, Divider, Decoder, and Multiplexer, specifically designed for 4-bit operations. Thoroughly tested and validated, these modules were seamlessly interconnected to form a cohesive calculator system.

I. Adder/Subtractor:

Notably, the integration of the Adder and Subtractor involved a mode selector and XOR gates for 2's complement conversion of input B. Accurate carry bit counting was also incorporated. Illustrated in the figure, our design showcases the interconnected components, encompassing inputs and outputs, forming the robust foundation of our 4-bit calculator system.

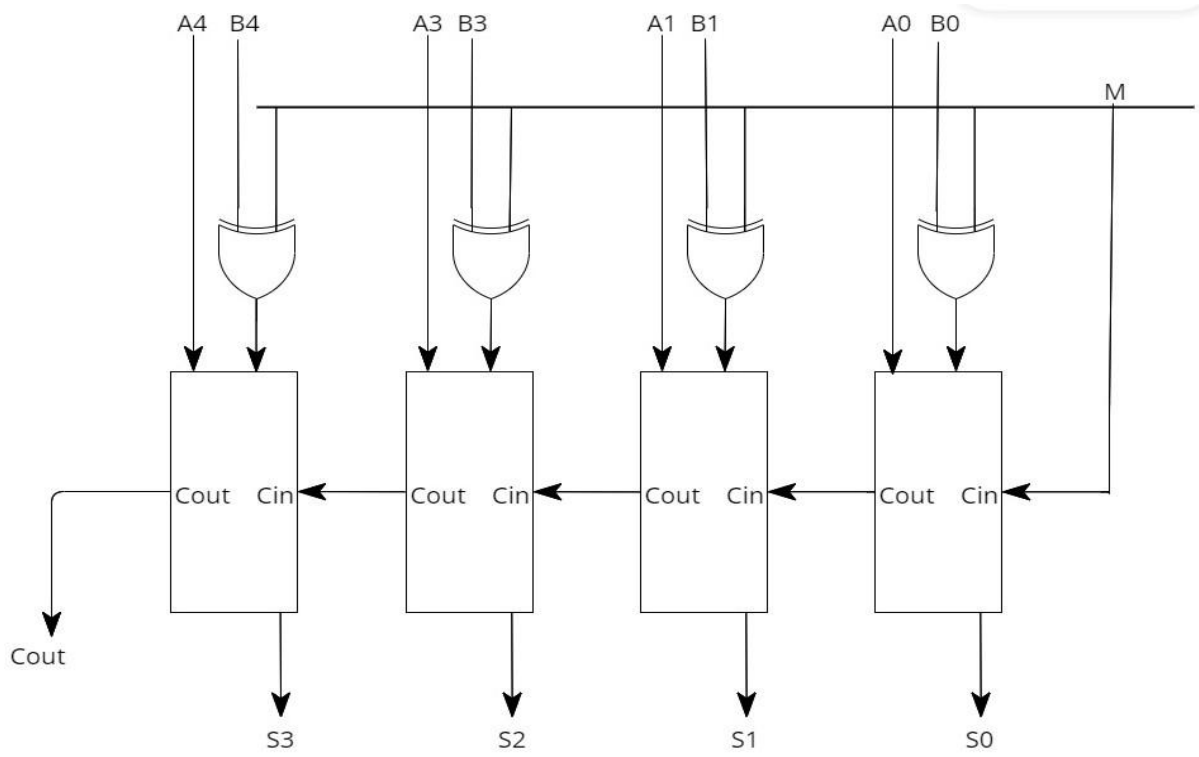
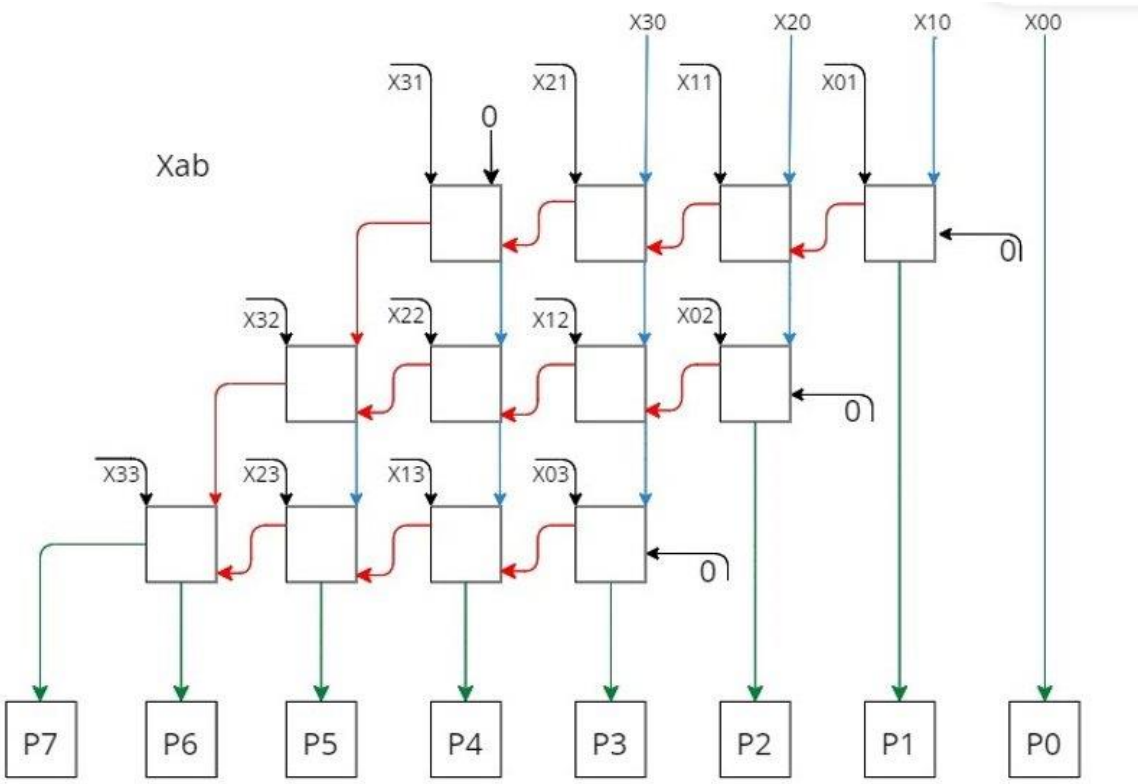


Fig.4: 4-bit adder/subtractor

II. Multiplier:



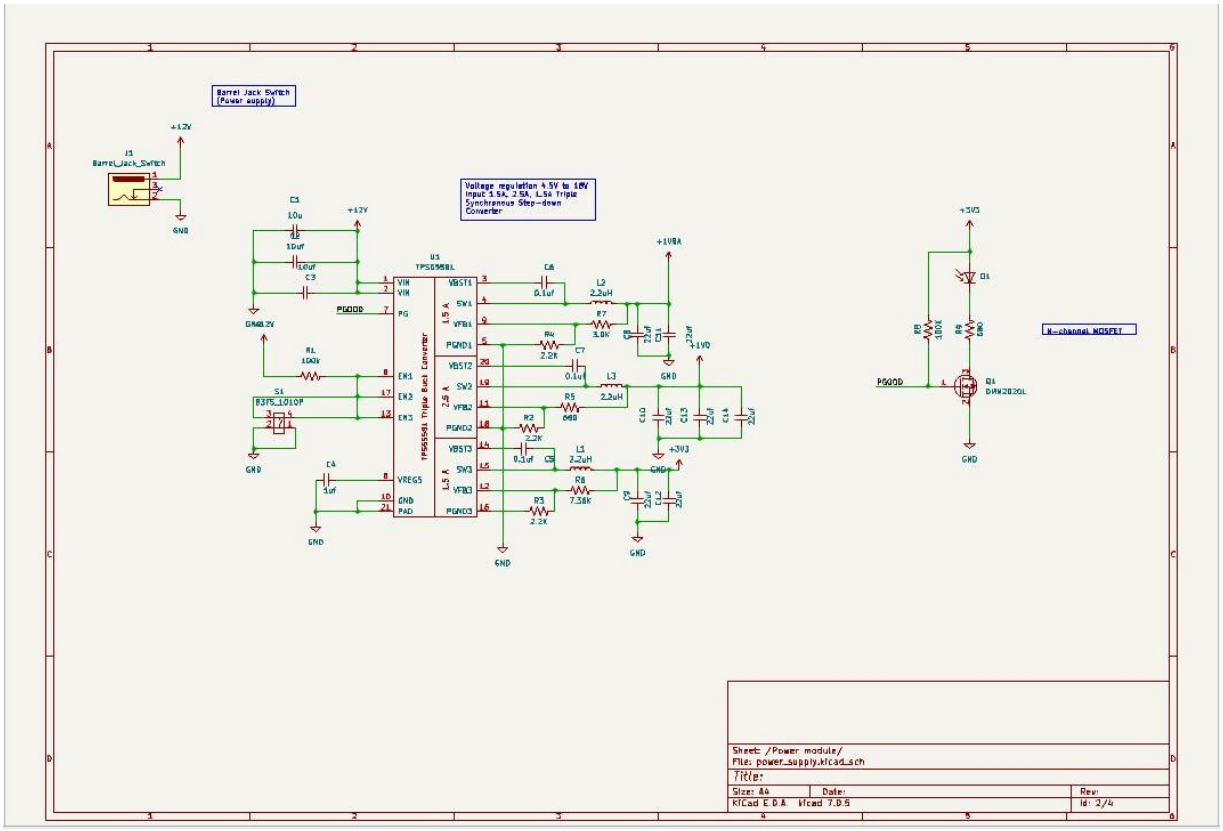
A basic drawing was created digitally by hand and used as a reference for coding the multiplier component in VHDL in Modelsim after some preliminary investigation on how the multiplier works using VHDL. 56 complete adders were utilized in total to realize the overall multiplier component. For this, many signals were required. The result of this specific multiplier component is the std_logic_vector "P".

6 PCB Design

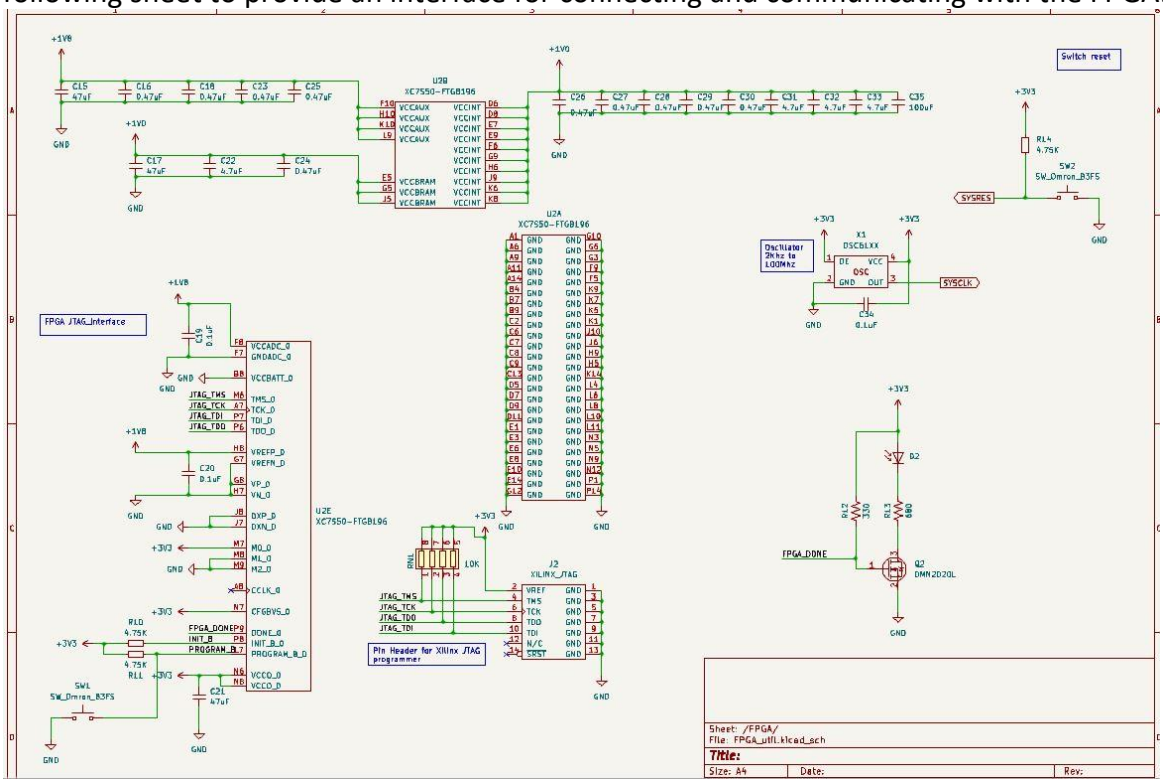
The PCB design and schematic diagram is implemented using Kicad software for the 4-bit calculator to be realized. The design contains Spartan 7 XC7S50- 1FTG196 FPGA chip for realizing the logic of the calculator. This is a programmable chip that is used to interpret hardware description languagesThe schematic diagram is implemented using hierarchy schematic method where the respective sections of the schematic are divided into sheets. The root represents an overall view of all the schematics used.

Figure: Root sheet of the schematic

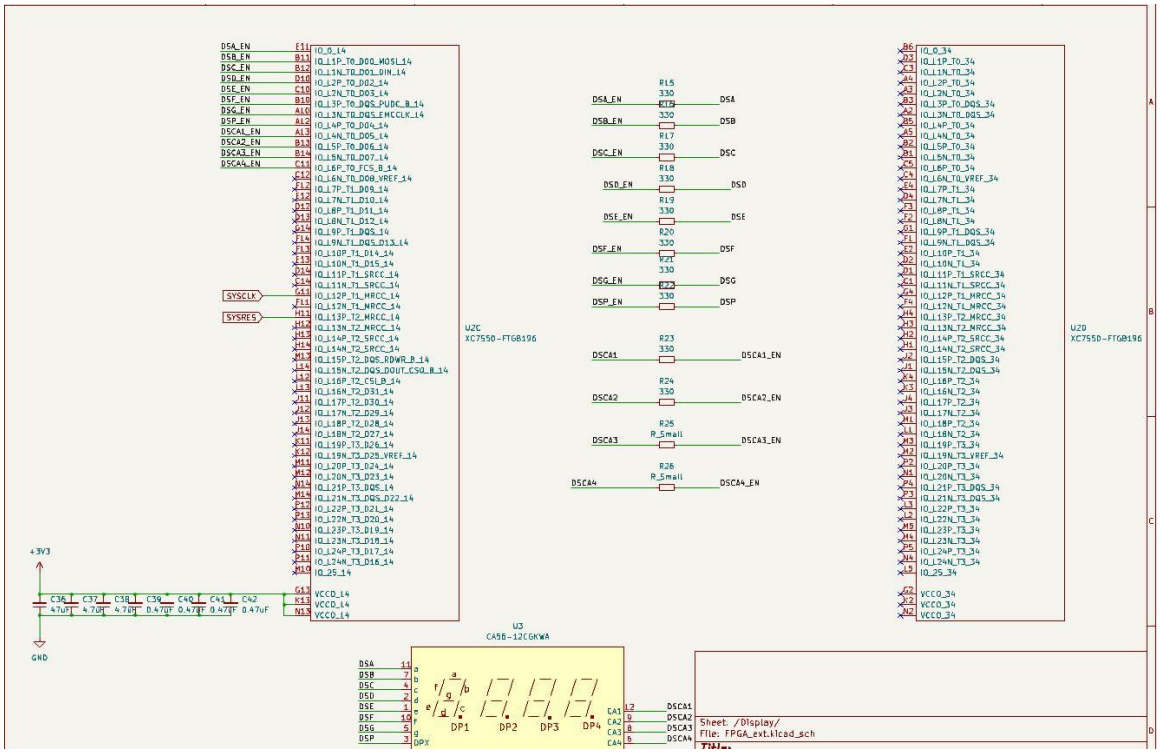
The Power module sheet represents the power circuit for the power supply for the PCB. Here a triple buck converter has been used for regulating the power. External libraries like power and LED libraries have been used for representing certain components like resistors and capacitors.



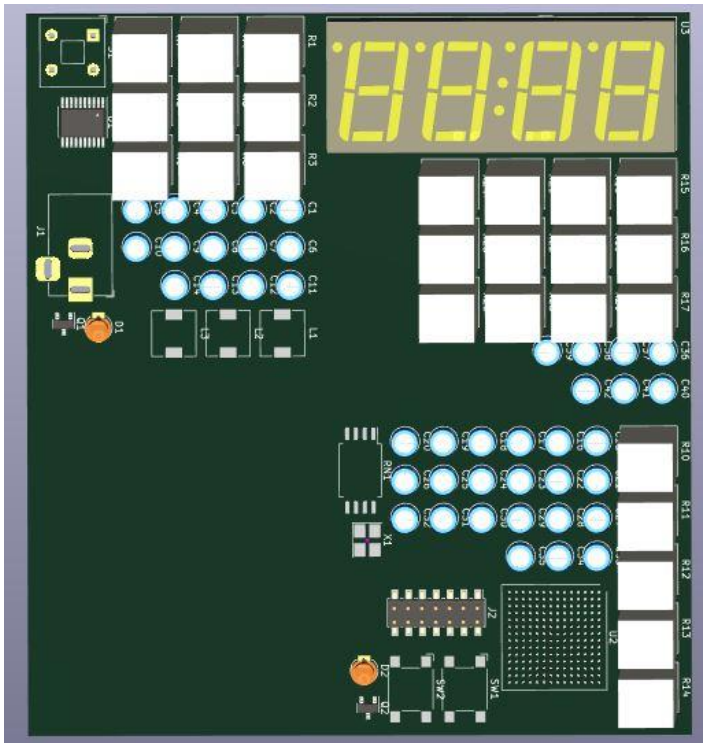
The JTAG programming interface, CLK interface and reset switch are designed in the following sheet to provide an interface for connecting and communicating with the FPGA:



The display and the FPGA symbols are in the following sheet:



Annotations for various components were assigned. An ERC check was carried out to solve numerous errors in the schematics, so the further process of PCB design is carried out in an errorless manner. This was done using the ERC check of the Kicad software itself. After performing a successful error check the PCB board was designed using 4 layers of copper. Several footprints for the components we not available in the existing library hence external libraries for the footprints have been used for components like capacitors and resistors. All the components are placed on the first inner copper layer. The following figures show the PCB boards design and the 3D view of the PCB.



7 Xilinx implementation

In this part we implement the code in the FPGA board (Nexys-A7). In the board we can any 4-bit unsigned number and calculate addition, subtraction, multiplication, division. As show in below figure it done addition to 3+3 which results 6.

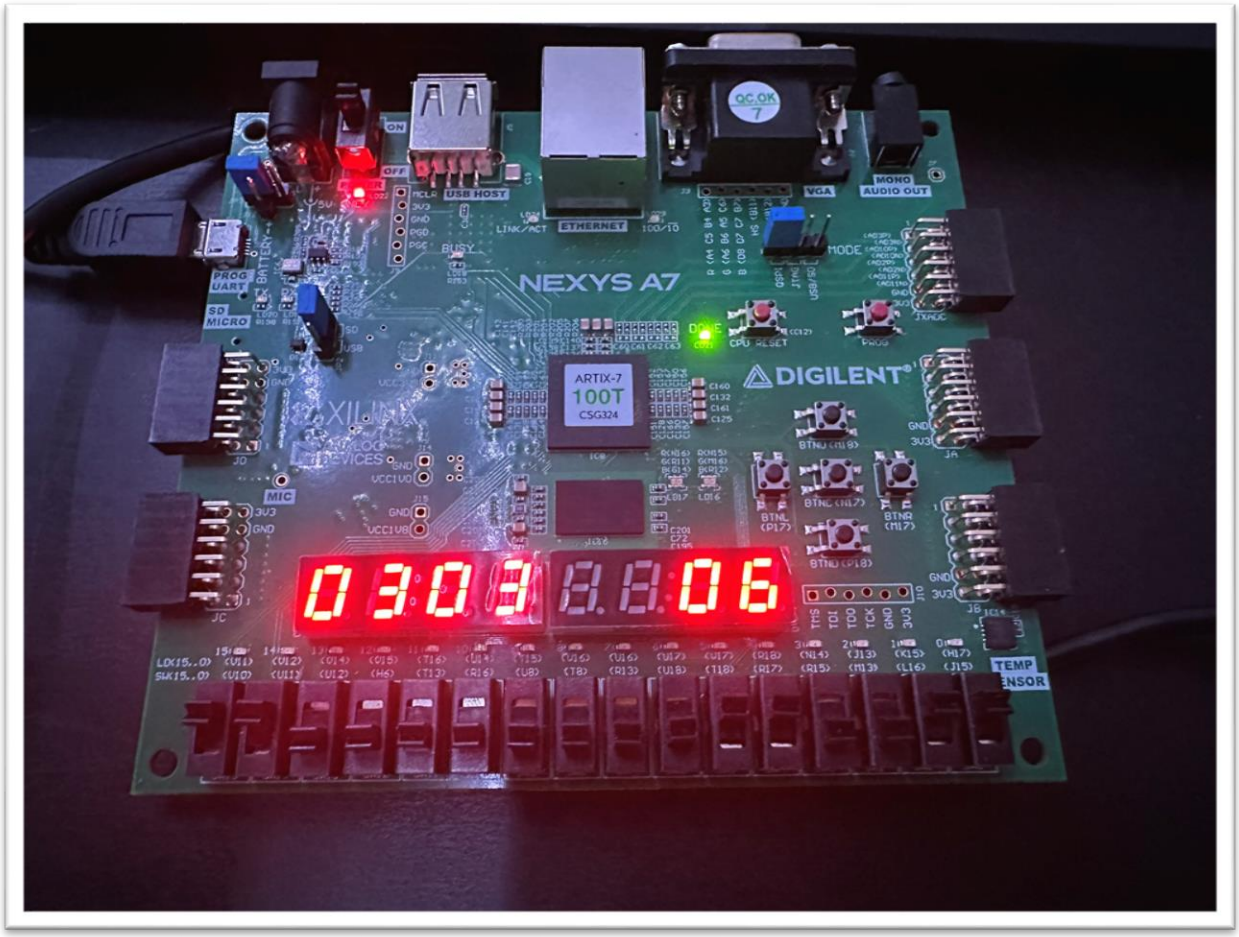


Figure: Calculating addition in Nexys-A7 board.

For the code part it divided in Top file (final_calculator) and other files which use the constrain file to setup and select the pins, buttons, display and switches in the board.

```
architecture Behavioral of main_calculator is
begin
    process(A, B, OP, Equals, Clear)
        variable temp: integer;
    begin
        if Clear = '1' then
            temp := 0; -- Reset temp to zero when Clear is active
        elsif Equals = '1' then
            case OP is
                when "00" =>
                    temp := to_integer(unsigned(A)) + to_integer(unsigned(B));
                when "01" =>
                    temp := to_integer(unsigned(A)) - to_integer(unsigned(B));
                when "10" =>
                    temp := to_integer(unsigned(A)) * to_integer(unsigned(B));
                when "11" =>
                    temp := to_integer(unsigned(A)) / to_integer(unsigned(B));
            end case;
        end if;
```

Figure: main_calculator file where all the operation defined.

Also, in below figure, there are two processes. The first process, ClockDivider, takes a 100MHz clock (CLK100MHZ) as input and generates a divided clock (DividedClock) with a frequency approximately one-hundred times lower, i.e., around 1MHz. It achieves this by toggling the DividedClock output every 10000 cycles. The second process, DisplaySelection, increments a counter (Display_Position_Counter) on the rising edge of the divided clock (DividedClock). The counter cycles through values 0 to 7, effectively creating a loop to control the display position, commonly used in 7-segment displays or similar applications. Once the counter reaches 7, it wraps back to 0, ensuring continuous cycling of the display positions. This clock divider and display position control module provides a reliable and efficient solution for managing display output in digital systems, where a lower frequency clock is required for display operations. The implementation is highly suitable for applications demanding precise timing and synchronization, such as digital clocks, timers, and embedded systems.

```
ClockDivider: process(CLK100MHZ)
begin
    if rising_edge(CLK100MHZ) then
        if Clock_Divider_Counter = 999 then
            Clock_Divider_Counter <= 0;
            DividedClock <= not DividedClock;
            else
                Clock_Divider_Counter <= Clock_Divider_Counter +1;
        end if;
    end if;
end process;
```

```
DisplaySelection : process(DividedClock)
begin
    if rising_edge (DividedClock) then
        if Display_Position_Counter = 7 then
            Display_Position_Counter <= 0;
        else
          Display_Position_Counter <= Display_Position_Counter+1;
          end if;
        end if;

end process;
```

Figure: Two process in Display_file

And the full implementation codes for VHDL, PCB & Xilinx is provided in the GitHub as instructed.

8 Sources/References

GitHub: https://github.com/Yashodhandesh/HW-AEE-Group-E

9 Contributions

Amit Chakma did the following:

- Introductions and block diagram
- VHDL code and testbench implementation for realization of 4-bit calculator in Model Sim software.

Yashodhan Vishvesh Deshpande did the following:

- Kicad implementation, Schematic Diagrams, Schematic symbols, footprint, component library research. And integration.
- PCB design, 3D views of the PCB.

MD Limon Apu

- Xilinx board setup and realization of 4-bit calculator on Xilinx board.
- Xilinx program implementation.