8-bit Calculator

Team Exemplary
24.06.2022

1 Team members

- Asm Nurussafa
- Tasawar Siddiquy
- Nirojan Navaratnarajah
- Arfat Kamal

2 Introduction

The goal of our project this semester is to build an 8-bit unsigned calculator. Our project design goal will perform the following arithmetic operations: 8-bit binary addition, subtraction, multiplication and division. All these operations are performed by combining different modules such as Adder, Subtractor, Multiplier and Divider using sub modules in which basic gates were used. Then, we design and code the following in VHDL and convert the schematics to PCB layout and then, fabricate it. All the software used for each step will be discussed further in the paper. We have used FPGA for the project, since FPGAs can be programmed multiple times and tested. The programming language used to program the components is VHDL.

3 Concept description

1. Block diagram of our application:

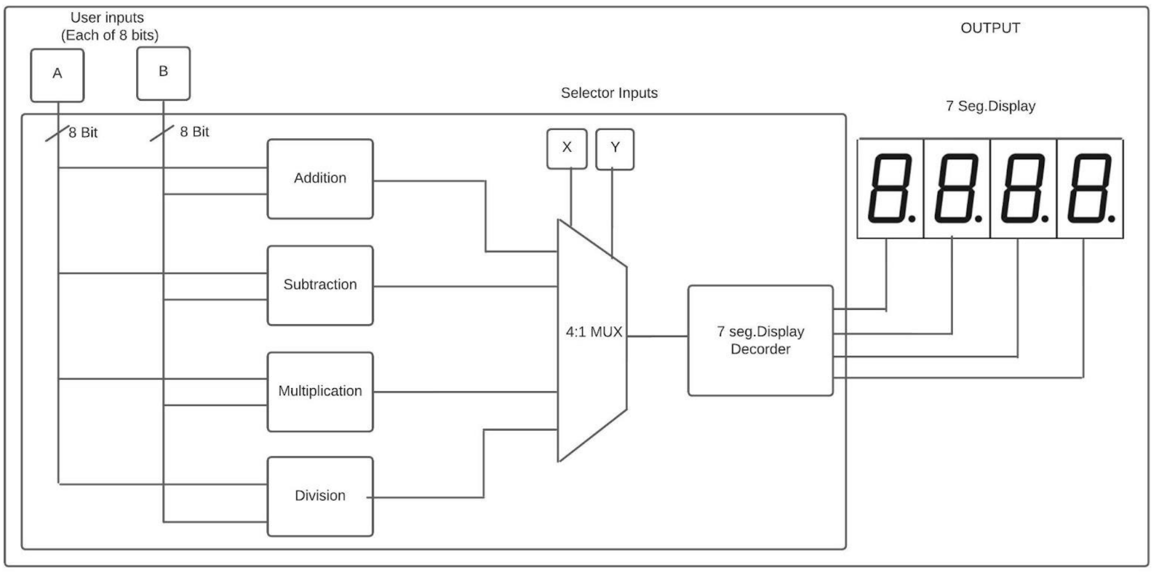


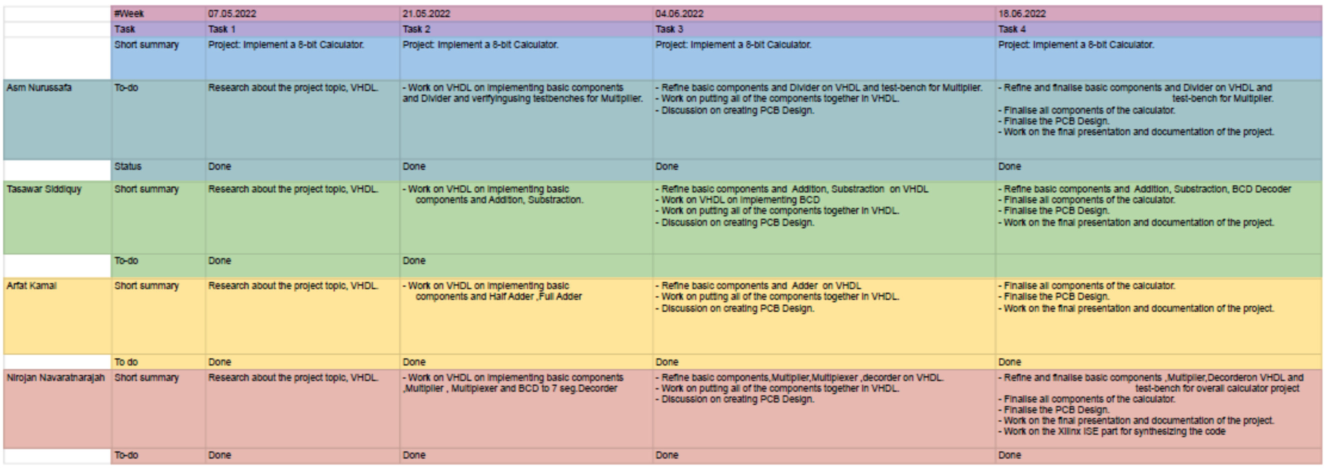
Figure 1

2. Application domain of our project:

The inputs of the System are fed into the first four blocks in the form of binary values using the slide switches found in the PCB, they will then produce the outputs of the two numbers multiplied, added, subtracted and divided using the individual blocks shown. These values from the individual components are then the inputs to the 4:1 Multiplexer. Then 1-bit binary values are fed to the multiplexer which is called the selector inputs. These selector inputs decide on which operation is intended by the user and select the appropriate driver line and send to the decoder. It is here where the output from the multiplexer is decoded to values which is able to display decimal values on the 7 segment displays. The displays are four separate devices and therefore a decoding algorithm which is using a bit shifting and adding mechanism is implemented.
The main application of our prototype is to do simple addition, subtraction, multiplication and division.

4 Project/Team management

To realize our project, we have decided to use a sort of the iterative development model. This will allow us to work on our project and test our scenario iteratively, meaning it lets us go back to previous steps and work on it and then, move on forward as we seem necessary. To organize the project in an orderly manner and to avoid miscommunication, we divided specific works for ourselves and held weekly meetings to comment on our progress and to decide on future tasks. When implementing, we got together in person as often as we could and worked on to realize our target scenario. A documentation of each group member`s progress is shown below. The dates stated indicate we had a meeting in each of these dates.



	#Week	07.05.2022	21.05.2022	04.06.2022	18.06.2022
	Task	Task 1	Task 2	Task 3	Task 4
	Short summary	Project: Implement a 8-bit Calculator.	Project: Implement a 8-bit Calculator.	Project: Implement a 8-bit Calculator.	Project: Implement a 8-bit Calculator.
Asm Nurussafa	To-do	Research about the project topic, VHDL.	- Work on VHDL on implementing basic components and Divider and verifyingusing testbenches for Multiplier.	- Refine basic components and Divider on VHDL and test-bench for Multiplier. - Work on putting all of the components together in VHDL. - Discussion on creating PCB Design.	- Refine and finalise basic components and Divider on VHDL and test-bench for Multiplier. - Finalise all components of the calculator. - Finalise the PCB Design. - Work on the final presentation and documentation of the project.
	Status	Done	Done	Done	Done
Tasawar Siddiquy	Short summary	Research about the project topic, VHDL.	- Work on VHDL on implementing basic components and Addition, Substraction.	- Refine basic components and Addition, Substraction on VHDL - Work on VHDL on implementing BCD - Work on putting all of the components together in VHDL. - Discussion on creating PCB Design.	- Refine basic components and Addition, Substraction, BCD Decoder - Finalise all components of the calculator. - Finalise the PCB Design. - Work on the final presentation and documentation of the project.
	To-do	Done	Done		
Arfat Kamal	Short summary	Research about the project topic, VHDL.	- Work on VHDL on implementing basic components and Half Adder ,Full Adder	- Refine basic components and Adder on VHDL - Work on putting all of the components together in VHDL. - Discussion on creating PCB Design.	- Finalise all components of the calculator. - Finalise the PCB Design. - Work on the final presentation and documentation of the project.
	To do	Done	Done	Done	Done
Nirojan Navaratnarajah	Short summary	Research about the project topic, VHDL.	- Work on VHDL on implementing basic components ,Multiplier , Multiplexer and BCD to 7 seg.Decorder	- Refine basic components,Multiplier,Multiplexer ,decorder on VHDL. - Work on putting all of the components together in VHDL. - Discussion on creating PCB Design.	- Refine and finalise basic components ,Multiplier,Decorderon VHDL and test-bench for overall calculator project - Finalise all components of the calculator. - Finalise the PCB Design. - Work on the final presentation and documentation of the project. - Work on the Xilinx ISE part for synthesizing the code
	To-do	Done	Done	Done	Done

Figure 2

5 Technologies

1. VHDL:

The **VHSIC Hardware Description Language** (**VHDL**) is a hardware description language (HDL) that can model the behavior and structure of digital systems at multiple levels of abstraction, ranging from the system level down to that of logic gates, for design entry, documentation, and verification purposes.
To begin implementing the code the Software Modelsim is used, it is a software targeted for Intel® FPGAs devices, which is a multi-language environment by Siemens, previously developed by Mentor Graphics.
For our project is helps mainly for developing a code, checking syntax, compiling and the main advantage is that it provides the facility to simulate the code, with a very user friendly user interface.

2. AutoCAD Eagle:

EAGLE is electronic design automation (EDA) software that lets printed circuit board (PCB) designers easily connect schematic diagrams, component placement, PCB routing, and comprehensive library content. For this project it was a requirement to design a PCB which includes integration of FPGA module ,switches for inputs, and 7 Segment displays.

A two-layer PCB is designed which will be described in the upcoming sections.

3. FPGA:

FPGA (Field Programmable Gate Array) is a hardware circuit that a user can program to carry out one or more logical operations. Taken a step further, FPGAs are integrated circuits, or ICs, which are sets of circuits on a chip—that's the “array” part. For this calculator project, the FPGA used is EP4CE22E22C8N from Intel. The library used is from Ultralibrarian.com.
For our project we have chosen FPGAs because the developed code can be flashed in several times, to see if it is working in reality. If we had used other components for example like ASICS, it is only possible to flash one time. Since the code is still in the development process, it is very much necessary to have this multiple time code flashing capability. The FPGA used for the PCB design for the project is as described and shown below in Figure 3.

Some of the specifications are as listed below:

- FGPA CyloneR IV E Family
- 22320 Cells 60nm technology
- 1.2 V
- 144-Pin EQFP EP

The Symbol Footprint and the 3D model are shown below.

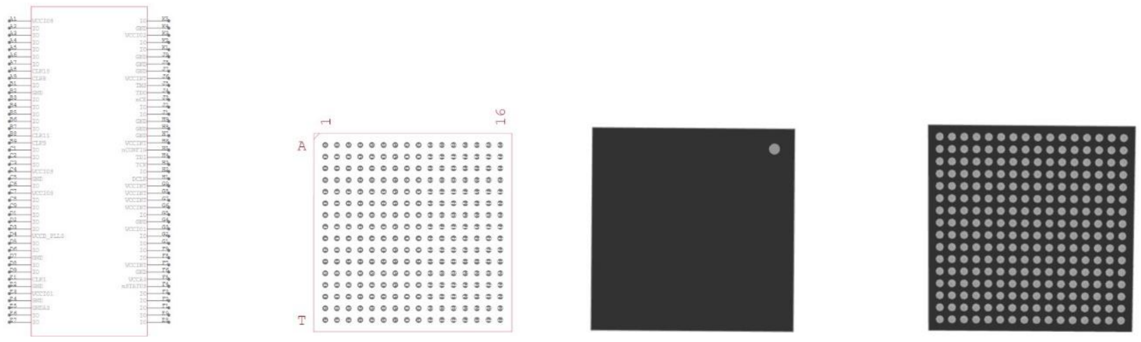


Figure 3

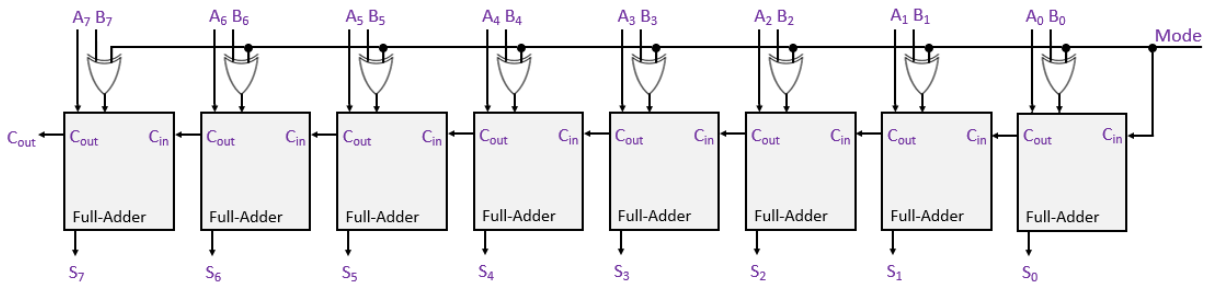
4. Xilinx ISE

The Xilinx ISE is primarily used for circuit synthesis and design.
Starting with checking the syntax of the code until generating the Bitstream file this software is used. The User constraint file (.ucf) is used for mapping the ports of the VHDL code with the physical pins of the FPGA.For this the use of keywords such as “NET” and “LOC” are used.

6 VHDL Implementation

As an initial step, basic components such as the Half adder, Full adder, ripple-carry adder were coded. Testbenches for each module are also made and the components are tested for valid output. In the next step, relatively larger modules like, 8-Bit adder, subtractor, multiplier, divider, decoder, and multiplexer were created. Finally all these sub modules were connected together in such a manner that the expected output is obtained from the overall code.

i. 8-bit adder/subtractor:



After creating all the basic components in VHDL we started our first function. We made an 8-bit Adder using the ripple-carry adder concept. Then the Adder has been integrated with the subtractor by using the mode selector. We used the XOR gate to convert the Input B in 2's complement then added it to the Input A for the subtractor. The carry bit also counted accurately. In the Figure, We can see how the components have been connected including the inputs and output.

ii. Multiplier:

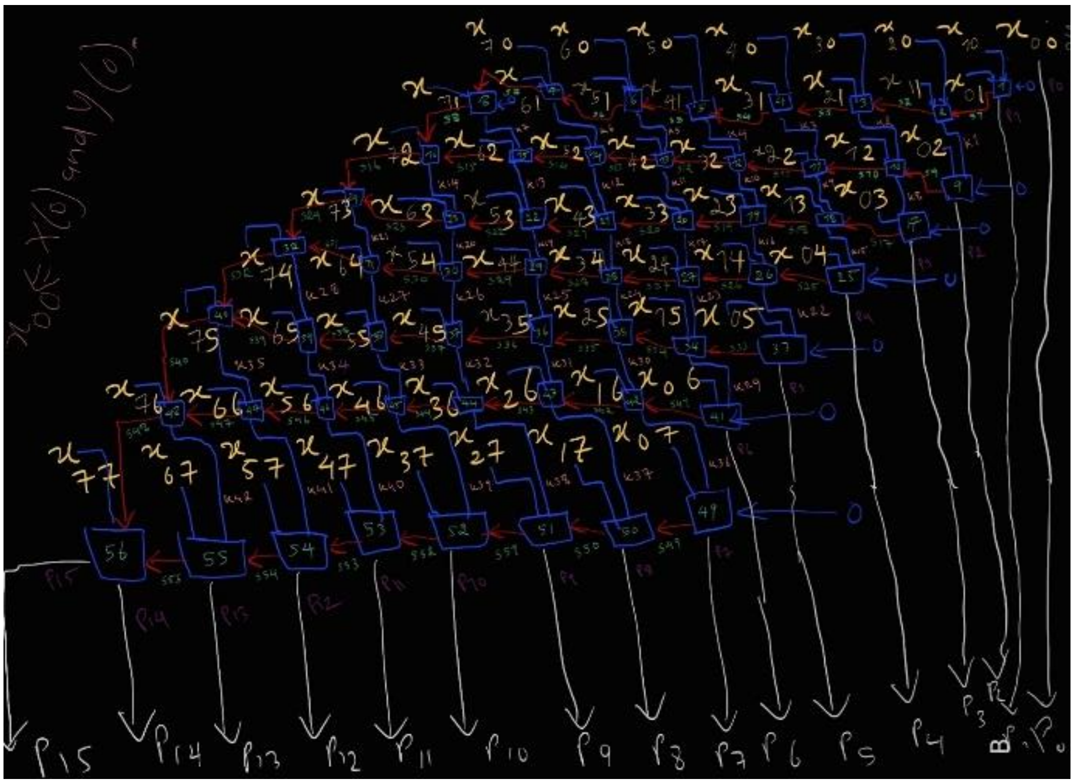


Figure 4

After small research on how the multiplier functions with VHDL, a rough sketch was drawn digitally by hand and this diagram was used as the reference for coding the multiplier component in VHDL in Modelism. To realise the overall multiplier component 56 full adders were used in total. Several signals had to be used for this purpose. The std_logic_vector "P" is the final output of this individual multiplier component.

iii. Divider:

To realise the divider, the following block diagram was taken into account.

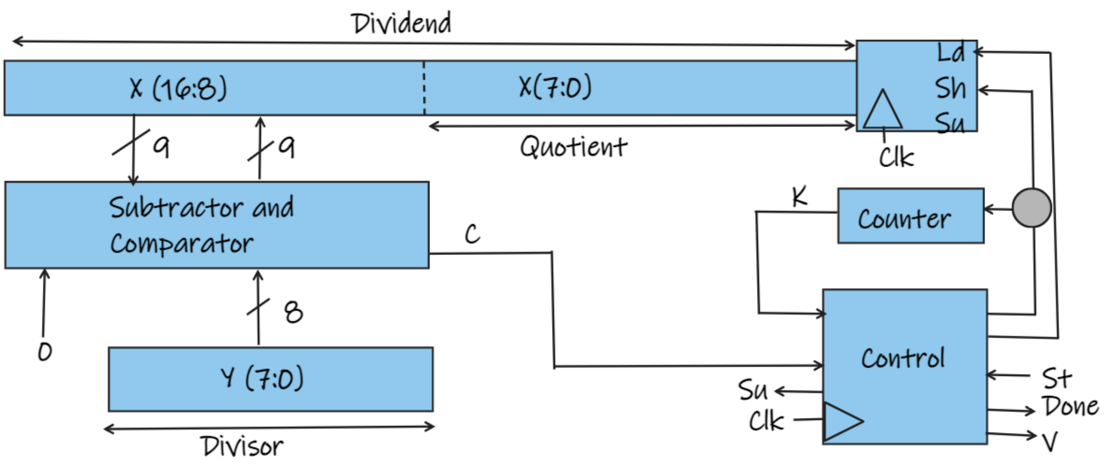


Figure 5

The division can be carried out by a series of subtraction and shift operations. To construct the divider, we will use a 17-bit dividend register and a 8-bit divisor register, as shown in Figure. During the division process, instead of shifting the divisor to the right before each subtraction like in the multiplication, we will shift the dividend to the left. Note that an extra bit is required on the left end of the dividend register so that a bit is not lost when the dividend is shifted left. Instead of using a separate register to store the quotient, we will enter the quotient bit-by-bit into the right end of the dividend register as the dividend is shifted left.

The following was finite state machine was created in order to realize the control circuit:

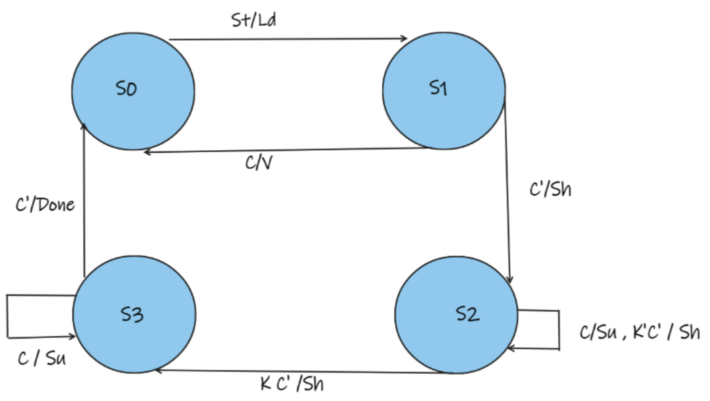
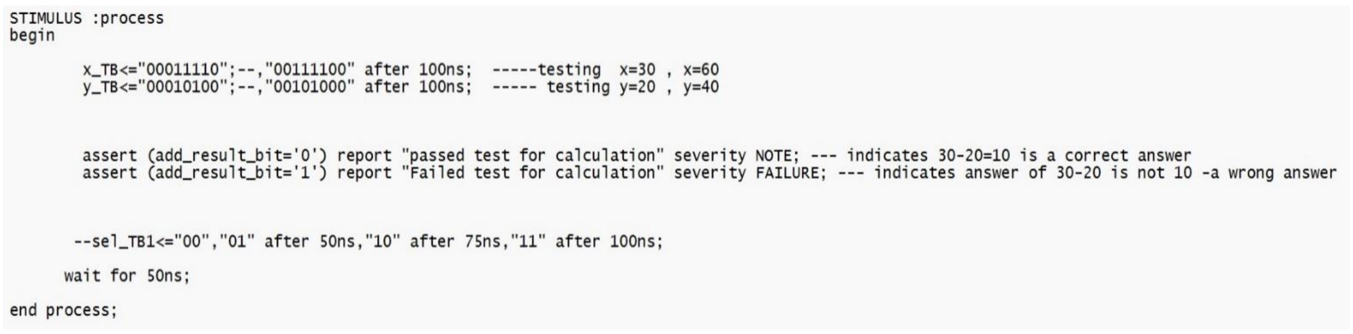


Figure 6

When a start signal (*St*) occurs, the 16-bit dividend and 8-bit divisor are loaded into the appropriate registers. If *C* is 1, the quotient would require 9 or more bits. Because space is only provided for a 8-bit quotient, this condition constitutes an overflow, so the divider is stopped, and the overflow indicator is set by the *V* output. Normally, the initial value of *C* is 0, so a shift will occur first, and the control circuit will go to state *S*2. Then, if *C* = 1, subtraction occurs. After the subtraction is completed, *C* will always be 0, so the next active clock edge will produce a shift. This process continues until four shifts have occurred, and the control is in state *S*3. Then, a final subtraction occurs if *C* =1, and no subtraction occurs if *C* = 0. No further shifting is required, and the control goes to the stop state. For this example, we will assume that when the start signal (*St*) occurs, it will be 1 for one clock time, and, then, it will remain 0 until the control circuit is back in state *S*0. Therefore, *St* will always be 0 in states *S*1 through *S*3.

iv. The overall calculator

A screenshot of the process of the testbench for the overall project is shown below, this is how the test bench is used to validate and check if the final vhdl code is functioning as desired.



```
STIMULUS :process
begin

        X_TB<="00011110";--,"00111100" after 100ns;  -----testing  x=30 , x=60
        y_TB<="00010100";--,"00101000" after 100ns;  ----- testing y=20 , y=40


        assert (add_result_bit='0') report "passed test for calculation" severity NOTE; --- indicates 30-20=10 is a correct answer
        assert (add_result_bit='1') report "Failed test for calculation" severity FAILURE; --- indicates answer of 30-20 is not 10 -a wrong answer


       --sel_TB1<="00","01" after 50ns,"10" after 75ns,"11" after 100ns;

      wait for 50ns;

end process;
```

Figure 7

Finally, these modules were used as components in the final project file, and they were all connected in a manner which produces the expected results as shown in the block diagram in section 3. With the final Testbench for the final project file, it is clear that the code is working as expected. Still the code needs more bug fixed and improvements, which can be solved by working more on the project. A picture illustrating the simulation of the Test Bench is shown below:

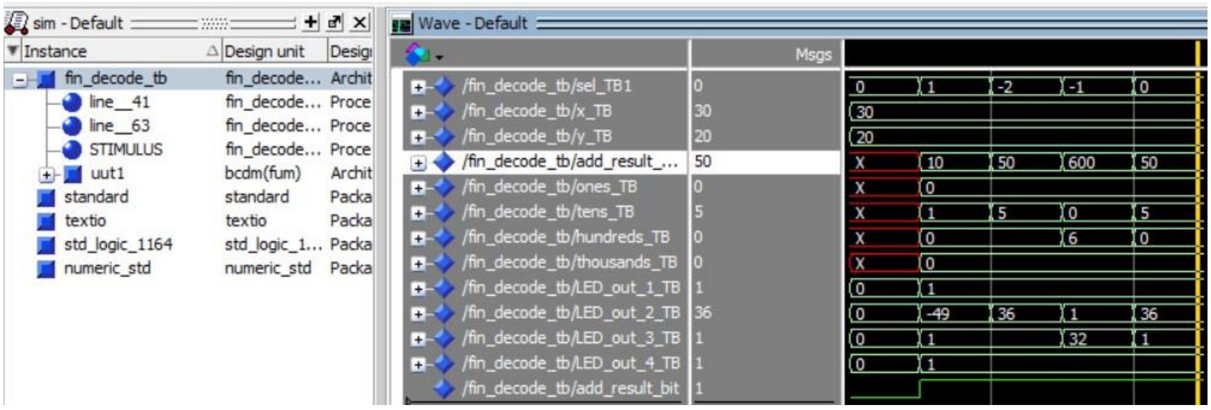

sim - Default
Instance | Design unit | Design
fin_decode_tb | fin_decode... | Archit
line__41 | fin_decode... | Proce
line__63 | fin_decode... | Proce
STIMULUS | fin_decode... | Proce
uut1 | bcdm(fum) | Archit
standard | standard | Packa
textio | textio | Packa
std_logic_1164 | std_logic_1... | Packa
numeric_std | numeric_std | Packa
Wave - Default
Msgs
/fin_decode_tb/sel_TB1 0 | 0 1 -2 -1 0
/fin_decode_tb/x_TB 30 | 30
/fin_decode_tb/y_TB 20 | 20
/fin_decode_tb/add_result_... 50 | X 10 50 600 50
/fin_decode_tb/ones_TB 0 | X 0
/fin_decode_tb/tens_TB 5 | X 1 5 0 5
/fin_decode_tb/hundreds_TB 0 | X 0 6 0
/fin_decode_tb/thousands_TB 0 | X 0
/fin_decode_tb/LED_out_1_TB 1 | 0 1
/fin_decode_tb/LED_out_2_TB 36 | 0 -49 36 1 36
/fin_decode_tb/LED_out_3_TB 1 | 0 1 32 1
/fin_decode_tb/LED_out_4_TB 1 | 0 1
/fin_decode_tb/add_result_bit 1

Figure 8

In the above shown diagram the Testbench is simulated, and as it is visible, inputs of "30" and "20" are given, each time when the selector changes, the answer (Outputs) in the 4th row also change. The second column indicates subtraction, third - addition, fourth- multiplication, and fifth- Division, (needs to be included)

RTL Schematic

RTL Schematic of Memory Chip RTL stands for **Register Transfer Language**. It shows the implementation logic of the circuit that how data flows in and out from the circuit.
Figure 9.shows the RTL schematic which is generated from the XILINX ISE Software.

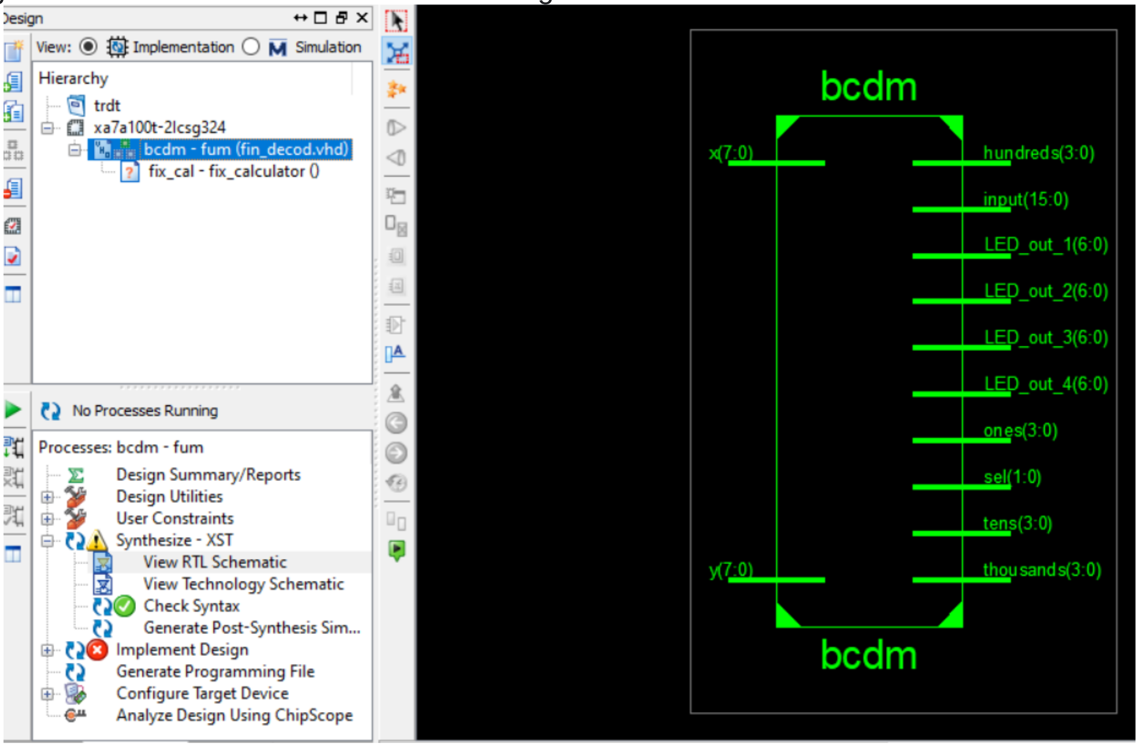


Figure 9

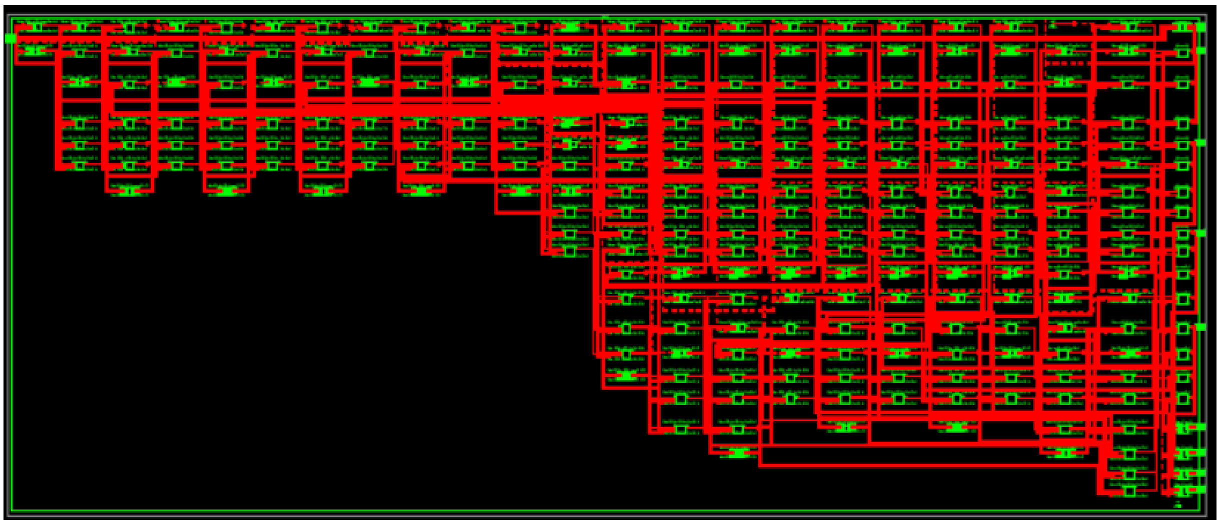


Figure 10

Zoomed view of the RTL schematic

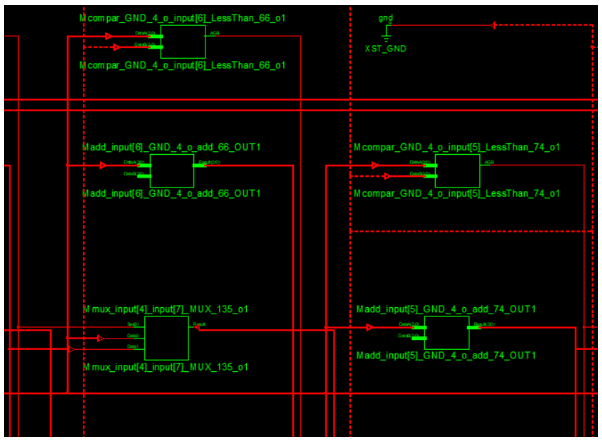

Mcompar_GND_4_o_input[6]_LessThan_66_o1
gnd
XST_GND
Mcompar_GND_4_o_input[6]_LessThan_66_o1
Madd_input[6]_GND_4_o_add_66_OUT1
Mcompar_GND_4_o_input[5]_LessThan_74_o1
Madd_input[6]_GND_4_o_add_66_OUT1
Mcompar_GND_4_o_input[5]_LessThan_74_o1
Mmux_input[4]_input[7]_MUX_135_o1
Madd_input[5]_GND_4_o_add_74_OUT1
Mmux_input[4]_input[7]_MUX_135_o1
Madd_input[5]_GND_4_o_add_74_OUT1

Figure 11

7 PCB Design

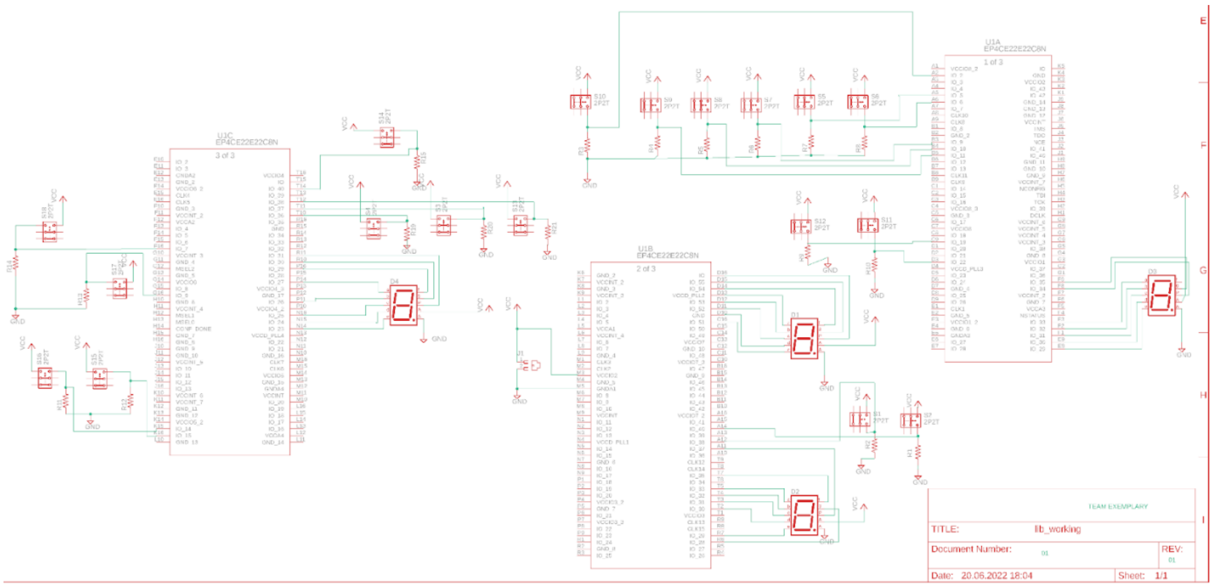


Figure 12

An image of the printed circuit board is shown below. (2 Layer PCB)

As mentioned earlier the software AutoCAD eagle is chosen for this PCB design.
Initially a research of which FPGA to be used for this project was carried out. From the Ultra-librarian website, the EP4CE22E22C8N library from Intel was available with the footprints and symbol, therefore this library was included in the software.
For the users to be able to enter the first and second numbers (in this case binary digits), one switch for each binary digit is chosen, and to simulate this with hardware, 8 sliding switches from the Sparkfun library were chosen for each input. In addition, the user is also able to

select which operation to perform and therefore two additional sliding switches are included for this purpose. Resistors are also used in this case as pull down the voltage when no input is provided. After placing the components in the schematic area, they were all wired up using the net tool. Next, the option for board is switched from schematic and the placing of components were done in a manner that makes sense to use. After the completion of the components placing step, the auto router option was used to wire up each individual component. Since there were too many connections, it was easier to start with AutoRoute and then manually alter some connection which did not fulfil the commands, like 90° bending of nets. The top and bottom layer is covered with a ground plate which makes it easier to make the GND connections and at the same time acts as a heat sink which helps to prevent heating up of the circuit board. A solder mask layer was applied to both the layers to protect the copper from oxidation and shorts during operation. As an additional step, ”vias” were placed in many locations which ensures good connection of the top and bottom layer Ground plates. To be able to mount the PCB on any surface or housing, four holes in the edges of the PCB are also provided. As a final step, a silk layer is added, where the certain texts such as Switch numbers, Team name, and other related texts were to be printed.
With this, the main part of the PCB for the project is ready, with inclusion of certain components such as power supply and J-Tag programmers and the clock (oscillator) the PCB will be fully complete and ready to be manufactured. Gerber files can then be generated and sent to the manufacturers for producing the physical PCB. (When complete and ready to be manufactured).

The figure below shows the PCB Layout

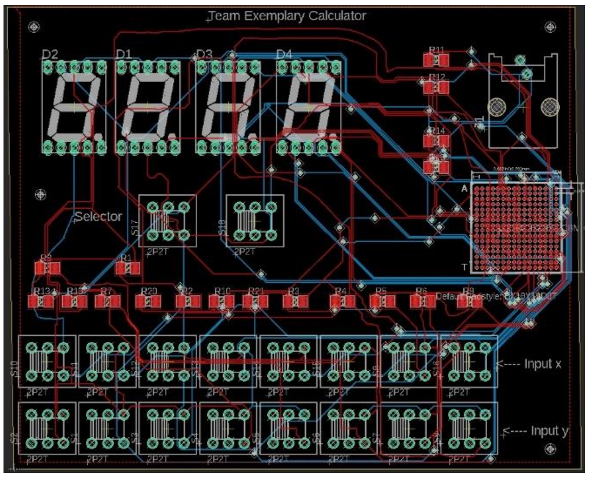


Figure 13

3D view of´the PCB using the Altium 365 Viewer (online version).

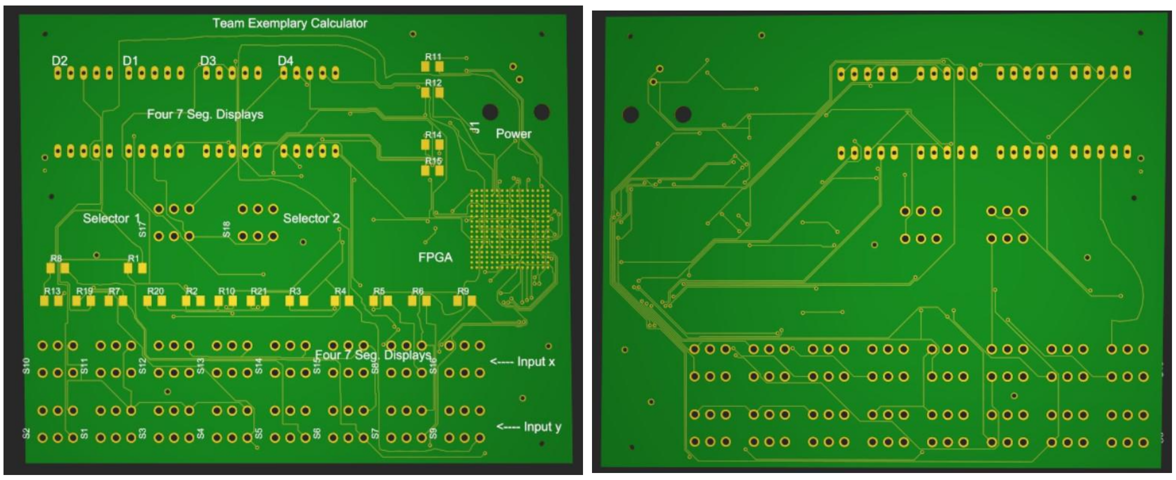
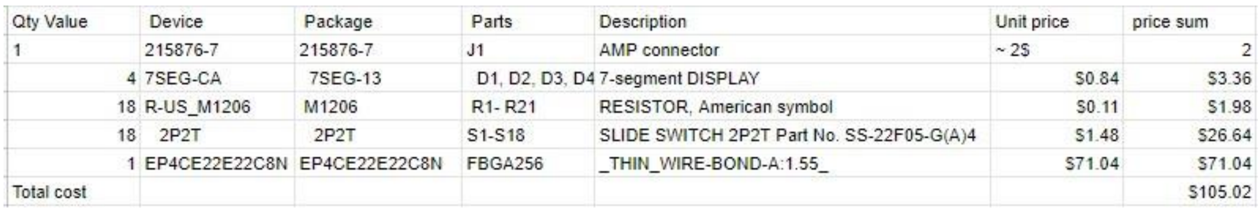


Figure 14

BOM List ,Cost ,size
Size of PCB -> 106 x 87mm
Total estimated cost : - $105.02

BOM list known as the Bill Of Materials gives an overview of the components used and their respective prices. The BOM list is automatically generated from the Eagle CAD software and then edited in an excel sheet adding the prices referring to the online library



Qty Value	Device	Package	Parts	Description	Unit price	price sum
1	215876-7	215876-7	J1	AMP connector	~ 2$	2
4	7SEG-CA	7SEG-13	D1, D2, D3, D4	7-segment DISPLAY	$0.84	$3.36
18	R-US_M1206	M1206	R1- R21	RESISTOR, American symbol	$0.11	$1.98
18	2P2T	2P2T	S1-S18	SLIDE SWITCH 2P2T Part No. SS-22F05-G(A)4	$1.48	$26.64
1	EP4CE22E22C8N	EP4CE22E22C8N	FBGA256	_THIN_WIRE-BOND-A:1.55_	$71.04	$71.04
Total cost						$105.02

Figure 15

1. **Limitations and restrictions of the implemented Calculator** :

- For subtraction – first number should be greater than second.(since signed numbers are not supported by the code.
- For multiplication answer can only be max 14 bits
- Input’s total should not be more than 14 bits in total
  Eg : 7 bits and 7 bits, or 8 bits and 6 bits, etc.

8 Sources/References

Github Repository - https://github.com/Asm-Nurussafa/Advanced-Embedded-System--Team-Exemplary

[1]Implementation and simulation of MC68HC11 a thesis ... - researchgate. (n.d.). Retrieved May 17, 2022, from https://www.researchgate.net/profile/Cumhur-Tuncali-2/publication/301804902_Implementation_and_Simulation_of_MC68HC11_Microcontroller_Unit_Using_SystemC_for_Co-design_Studies/links/57290d1908aef5d48d2c907a/Implementation-and-Simulation-of-MC68HC11-Microcontroller-Unit-Using-SystemC-for-Co-design-Studies.pdf

[2]https://app.ultralibrarian.com/details/229a8561-a1bb-11ea-b5d0-0aebb021a1ea/Intel/EP4CE22E22C8N?uid=15679901