Conglomerate of Practices using FPGA's

Jorge Alejandro González Díaz

March 20, 2024

Abstract

This paper presents a comprehensive examination of various digital design practices implemented on the Altera MAX 10 10M50DAF484C7G FPGA using Verilog. The focus is on leveraging the specific features of this FPGA to optimize digital circuits for speed, power, and resource utilization. Following this, we delve into a series of design case studies, including signal processing algorithms, digital controllers, and communication interfaces, all tailored to exploit the FPGA's architecture efficiently. Each case study follows a structured approach, starting with design specification, through simulation and synthesis, to the final implementation and testing on the FPGA. Verilog its employed for its versatility and industry relevance, providing detailed insights into coding strategies and optimization techniques that enhance performance and scalability. The paper concludes with a discussion on the implications of these practices for future FPGA designs and potential areas for further research.

1 Introduction

A Field Programmable Gate Array, also known as FPGA, is a technology which allows users to interact directly with hardware through Hardware Description Languages (HDL) such as Verilog, VHDL and ABEL. Its main features are related to the design flexibility they provide, as they can be reprogrammed multiple times which reduces development costs, allows for easier correction of errors in designs and facilitates circuit testing before launching the final product into the market. In this paper, I will include the main practices done during the "Programmable Logic Design" course along with their simulations and results, for these purposes I will be relying in the Intel Quartus development environment (IDE) to ensure logic integrity and reliability.

One thing I should clarify before embarking on the practices explanations if that the FPGA board model being used in these projects is the Altera MAX 10 10M50DAF484C7G designed by Intel, which is known for its versatility and practicality for the varied peripherals which are included in the board.

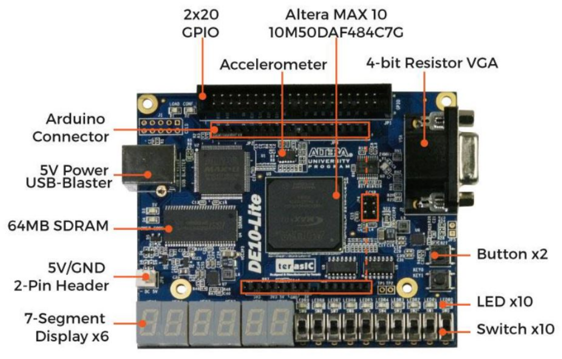


Figure 1: Intel Altera 10M50DAF484C7G

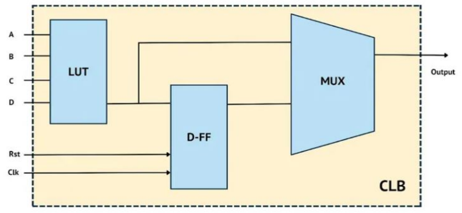


Figure 2: Architecture of a Configurable Logic Block (CLB)

2 Clock Divider

2.1 Functionality

This practice consists of creating a clock divider module to adjust the timescale of a virtual oscillator leveraging the internal 50 Mhz clock in the Altera Max 10 according to the user needs. To demonstrate the correct operation of the device, the board's 7 segment displays were used to display the count of a timer which triggers every second and can count up or down according to the state of the first switch (SW[0]) in the board. Additionally, the user can reset the current count or pause the clock when needed.

2.2 Register Transfer Level (RTL) View

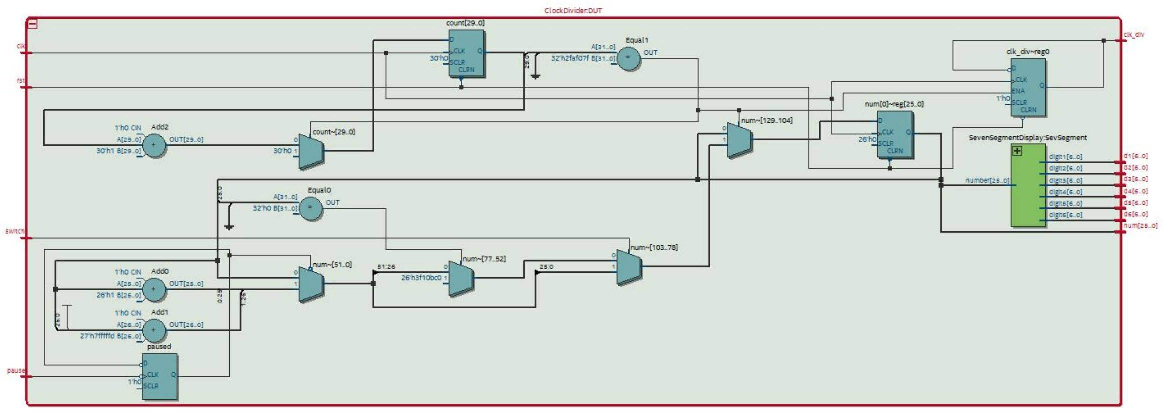


Figure 3: RTL view of Clock Divider

2.3 Physical Tests

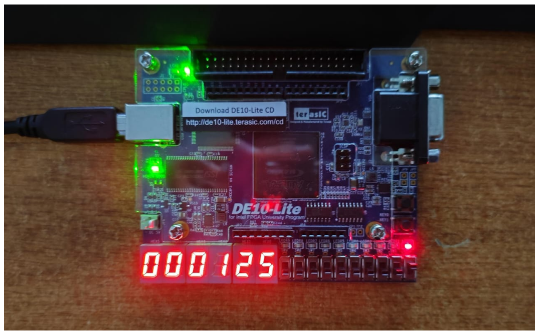


Figure 4: Counter Physical Testing

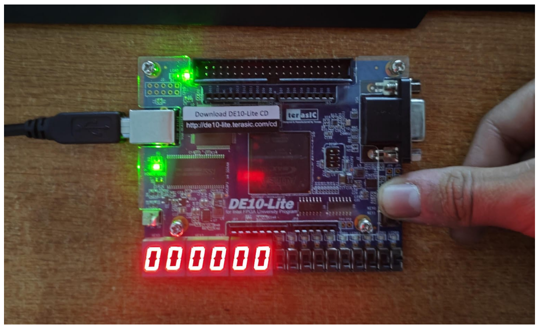


Figure 5: Reset State Testing

In the previous images it is shown that the module works as intended and in this case is updating a counter in a constant time lapse of 1 second. Next, a test bench of the circuit design is shown, as it is generally a good practice to keep track of the design signals responding to different stimuli.

2.4 Clock Divider Test Bench

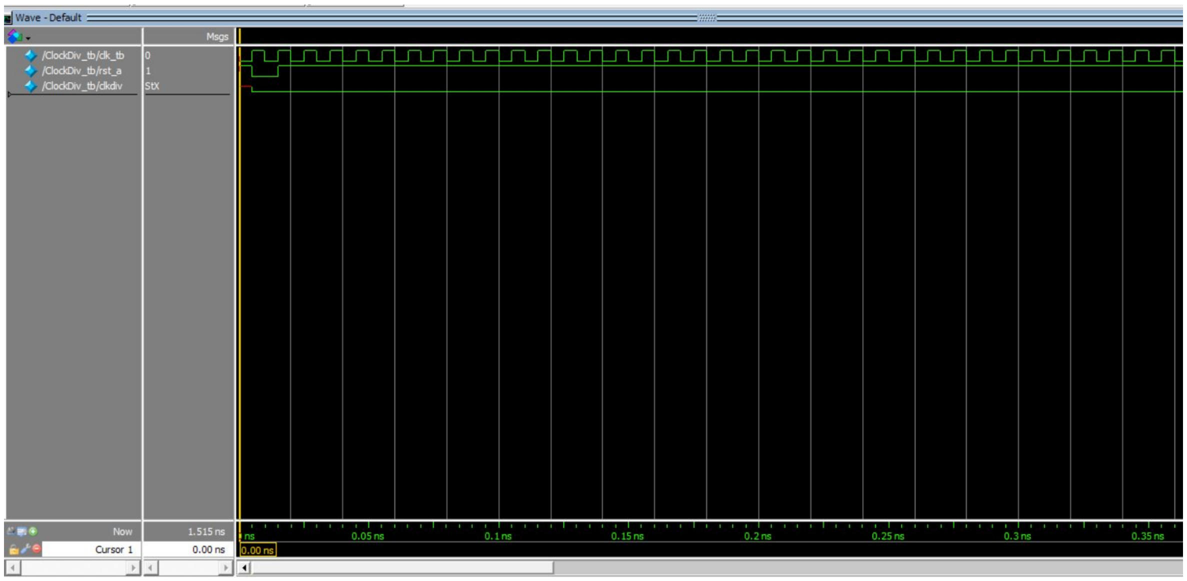


Figure 6: Clock Divider Test Bench

The test benches are used to simulate the logic design under different conditions. In this image it is shown that the clock input is toggled recurrently and the reset signal is toggled at the beginning of the simulation. On the other hand, the clock output is only toggled near the start of the simulation, but the timescale which is being used does not give enough time for it to toggle, as it only changes its current state once every 50 million clock input cycles.

3 Password Module

3.1 Functionality

This practice consists of a password module which was implemented using a custom state machine which is updated accordingly with the user input. The module stores a fixed password number and compares each digit with the current value of the board built-in switches. If the value and the current digit are equal to one another, the state machine takes the next state, if not, it returns to IDLE. Ultimately the module checks if the passwords match, if so, it turns on a CORRECT state LED signal, if not, it turns on an INCORRECT state signal.

3.2 RTL View

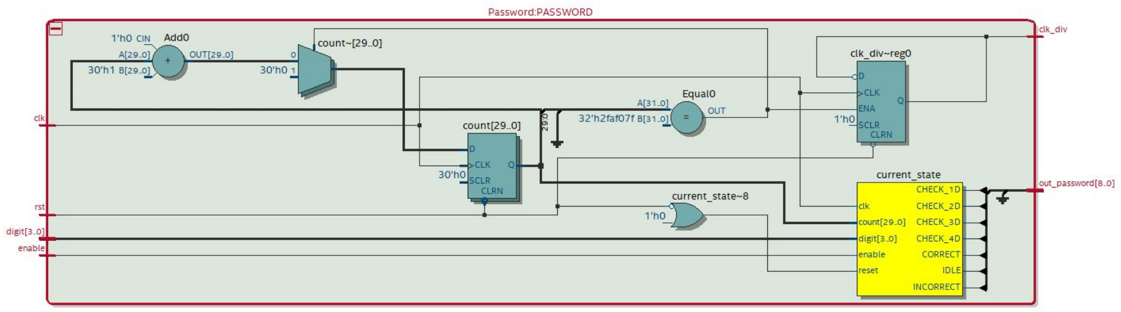


Figure 7: RTL view of Password Module

3.3 State Machine View

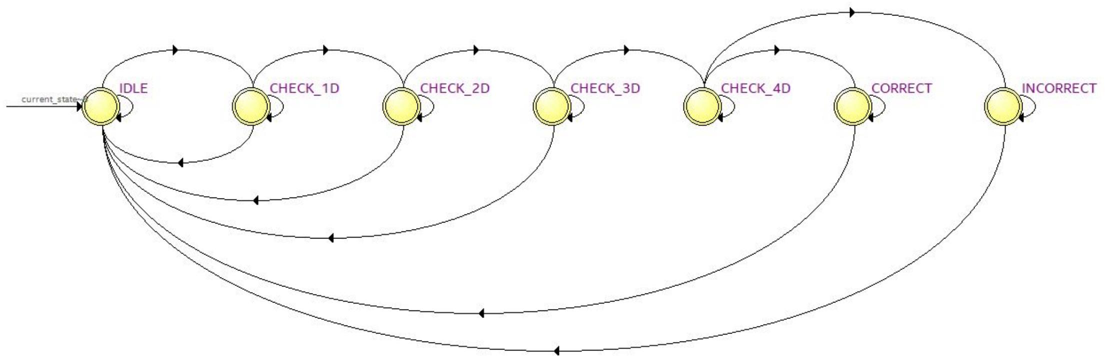


Figure 8: State Machine of Password Module

3.4 Physical Tests

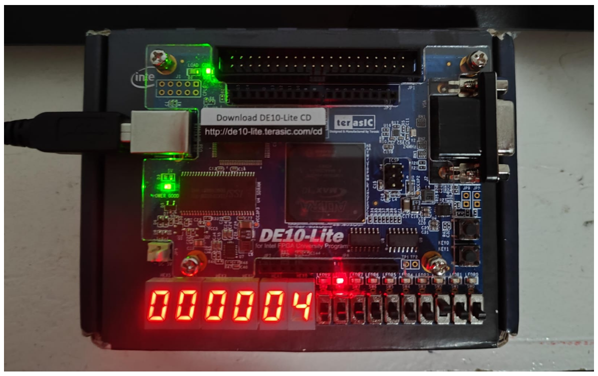


Figure 9: Correct LED signal on

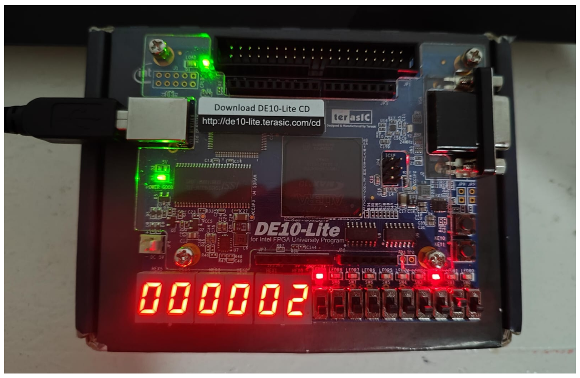


Figure 10: State Machine index shown in LED

3.5 Test bench

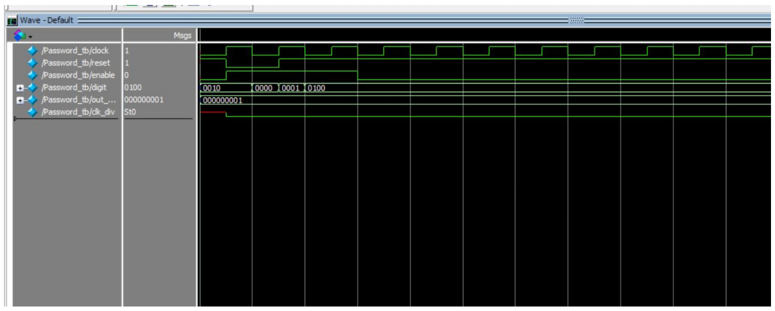


Figure 11: Password Module Test Bench

Personally, I considered this practice to be very educative since it was the first encounter I had with state machines in Verilog. In the same way, a lot was learnt about user inputs and wrapper development to allow these kinds of functionalities.

4 Pulse Width Modulation (PWM)

4.1 Functionality

This practice consists of creating various modules and combining them to generate pulse width modulated signal to control a servo motor. Its important to keep in mind that servos are controlled with pulses of different duration relative to a 20 ms constant signal period. For setting the motor position in 180° a high pulse of 2 ms is needed, on the other hand, for setting the servo position in 0°, no high pulse is needed, so the high operation range for them is from 0 - 2 ms, while the remaining milliseconds must be in a low state. This application allows the user to specify the number of degrees they want to rotate the motor in binary via built-in switches and update the position with an enable push button. For the integration of this module, it is necessary a PWM Handler along with a clock divider module to adjust pulse frequency resolution to several pulses per millisecond, to operate in the high pulse range of 0 - 2 ms. In the same way, a top wrapper to interact with the physical inputs of the board was included.

4.2 RTL View

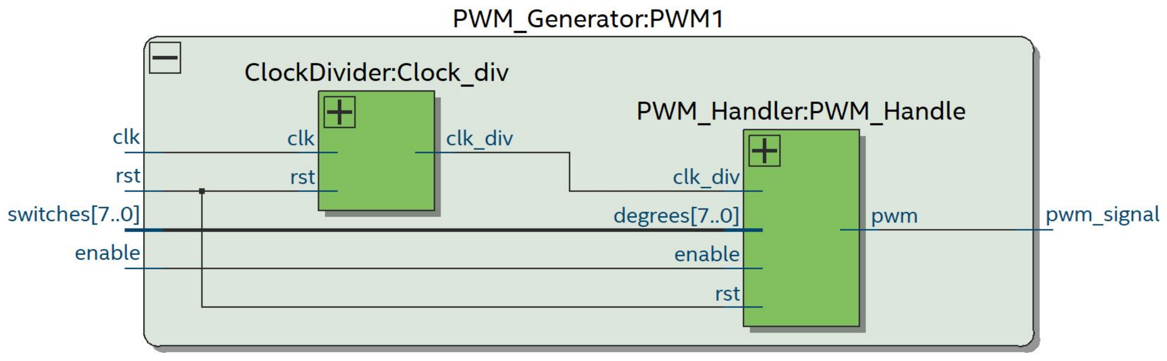


Figure 12: RTL view of PWM Module

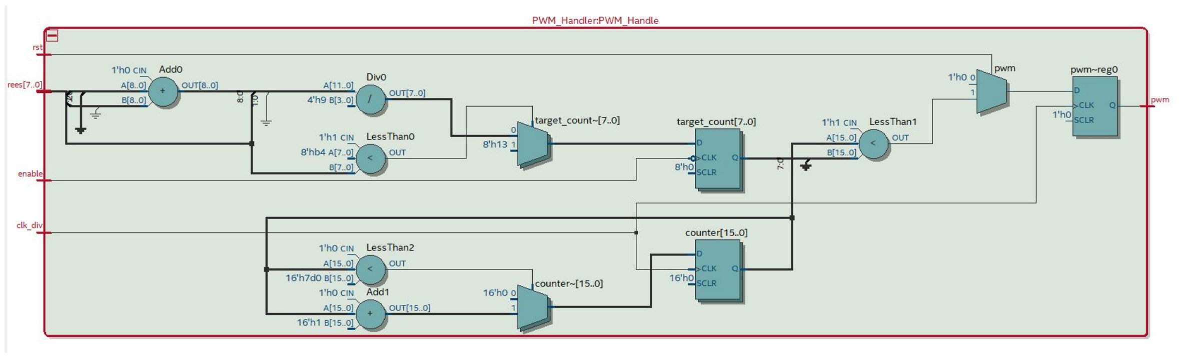


Figure 13: RTL view of PWM Handler

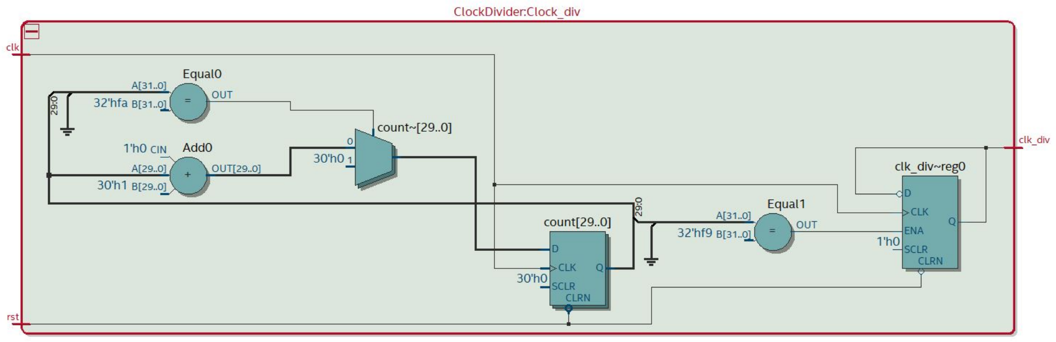


Figure 14: RTL view of Clock Divider Instance

4.3 Physical Tests

To demonstrate the correct operation of the device, I decided it was better to attach a photo of the generated signal using an oscilloscope, as it allows to see the duration of each pulse with high accuracy and also the waveform of the PWM signal.

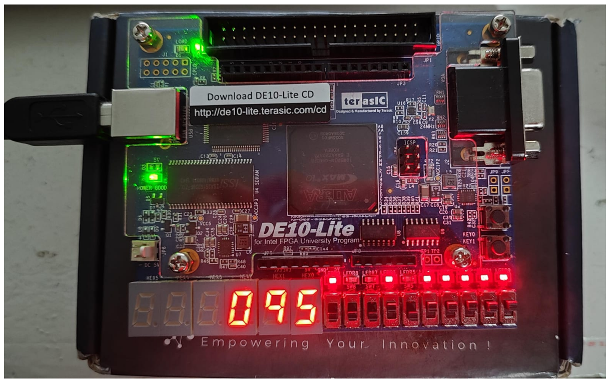

Download DE10-Lite CD
http://de10-lite.terasic.com/cd
terasIC
DE10-Lite
for Intel FPGA University Program
095
Empowering Your Innovation !

Figure 15: Switches to degrees conversion

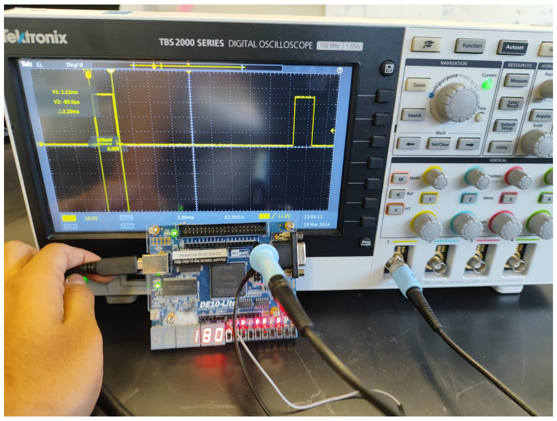
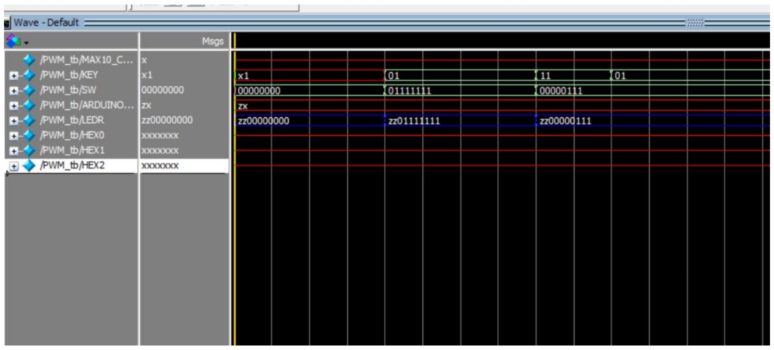


Figure 16: Oscilloscope testing)

The module was evaluated using an oscilloscope because this way, the timing of the PWM signal can be easily evaluated what was observed is that it does work correctly, since PWM scales proportionately limiting the high pulse to 2 ms. In the same way, the complete signal period was measured to be exactly 20 ms, which meets the PWM requirements to control accurately a servo motor.

4.4 Test Bench



```
/PWM_tb/MAX10_C...   x
/PWM_tb/KEY          x1          x1          01          11    01
/PWM_tb/SW           00000000    00000000    01111111    00000111
/PWM_tb/ARDUINO...   zx          zx
/PWM_tb/LEDR         zz00000000  zz00000000  zz01111111  zz00000111
/PWM_tb/HEX0         xxxxxxx
/PWM_tb/HEX1         xxxxxxx
/PWM_tb/HEX2         xxxxxxx
```

Figure 17: PWM Module Test Bench

5 UART

5.1 Functionality

Universal Asynchronous Receiver Transmitter, also known as UART, is a serial data communication protocol which operates using 2 channels, one for data transmission and the other one for data reception. As its name describes, there is not a common clock between parts, so a baud rate must be provided to synchronize them, which basically specifies how many bits are sent per second. Typical baud rates are 4800, 9600 and 115200. A clock divider must be included to adjust clock signal frequency to these timing requirements. It is necessary to clarify that the base code was taken from [1]

5.2 RTL View

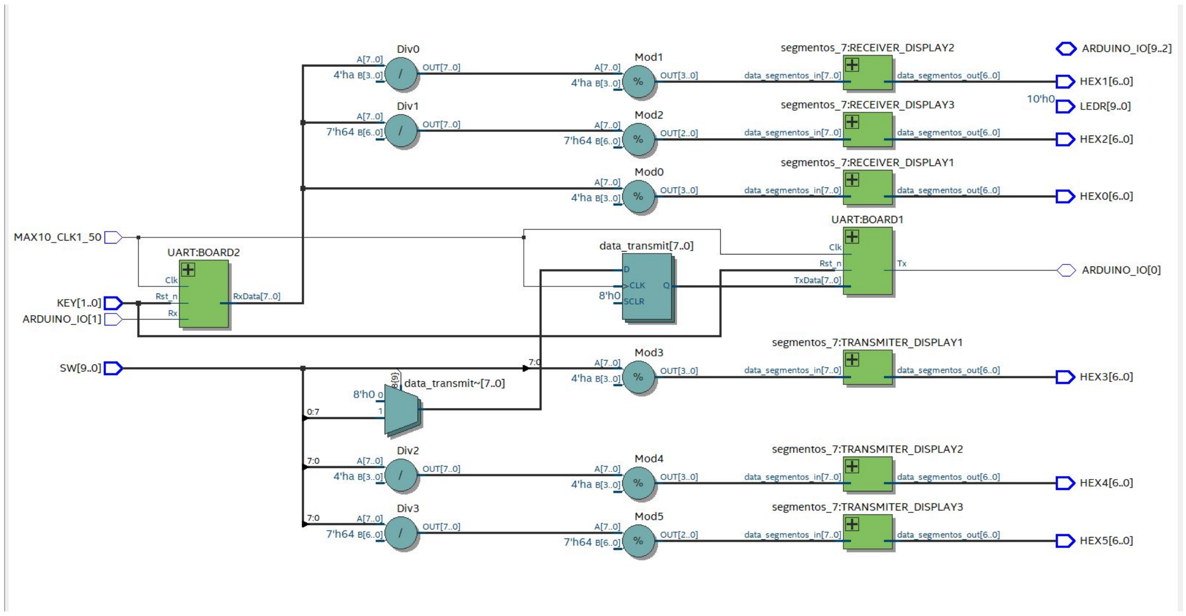


Figure 18: RTL view of top wrapper UART Module

In the previous image, it can be observed that to make the UART module work correctly in a single board, it is crucial to instantiate two UART modules, one specifically aimed at data transmission and the other one its used for data reception. In the program, interaction between modules is carried out with a physical wire or jumper which connects transmission (TX) and reception (RX) buses and the user is allowed to specify the numeric data they want to send via the board switches. To prove the integrity of the sent data, 3 displays are used to check the sent value and the other 3 are used to check the received value. If the value received matches the value transmitted, it means that the application is working correctly.

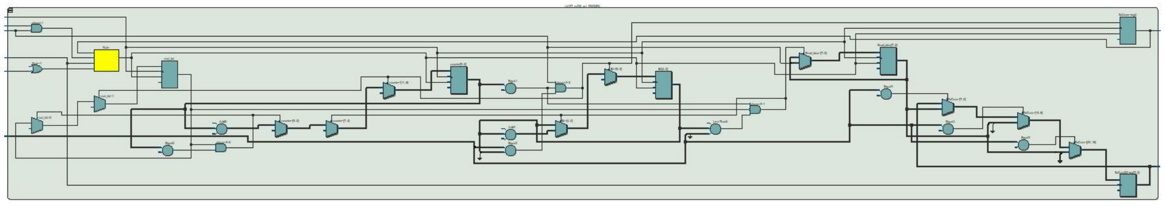


Figure 19: RTL view of RX module

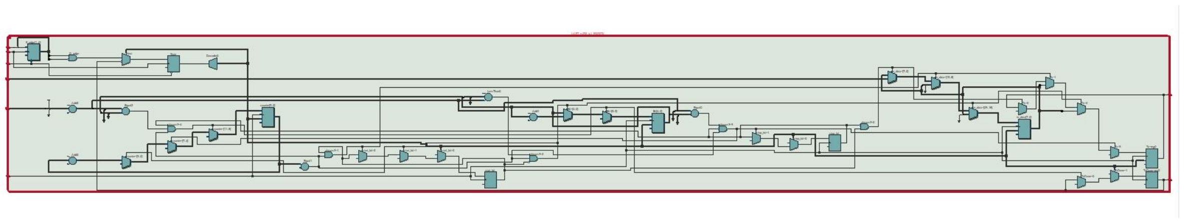


Figure 20: RTL view of TX module

Basically these circuit designs describe the logic behind the complete UART module, additionally a clock divider must be included in order to scale down the Altera Max 10 clock which runs at 50 Mhz to the required baud rate for synchronization purposes, which in this case is equal to 9600 bits per second, but a different baud rate could be chosen and the module integration should work appropriately.

5.3 Physical Tests

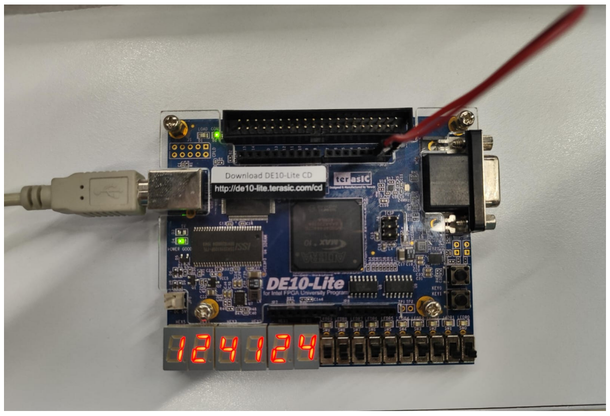


Figure 21: UART data not sent yet

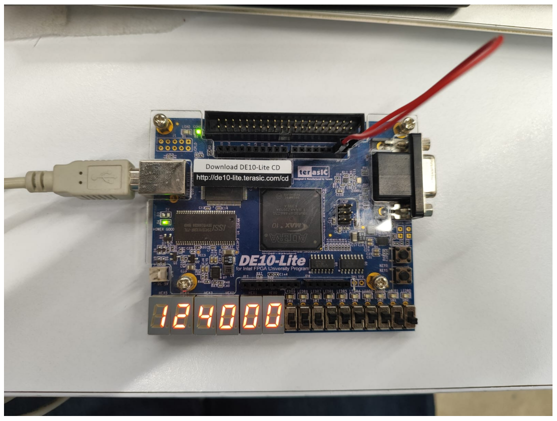


Figure 22: UART data sent & received

6 Video Graphics Array (VGA)

6.1 Functionality

This Verilog module generates a VGA signal for a 640x480 resolution display. It uses a 25 MHz clock and provides synchronous reset, horizontal and vertical sync signals, and RGB color outputs. The script defines video timing and dimensions, and utilizes counters to manage pixel positioning and screen refresh cycles. Additionally, it creates a test pattern of vertical color bars to demonstrate VGA output functionality, making it a great demonstrative example for digital video applications.

6.2 RTL View

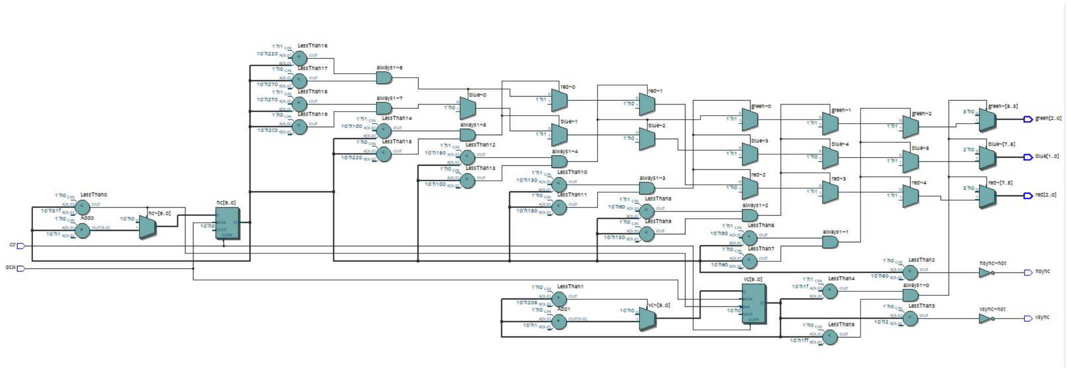


Figure 23: VGA RTL View

6.3 Physical Test

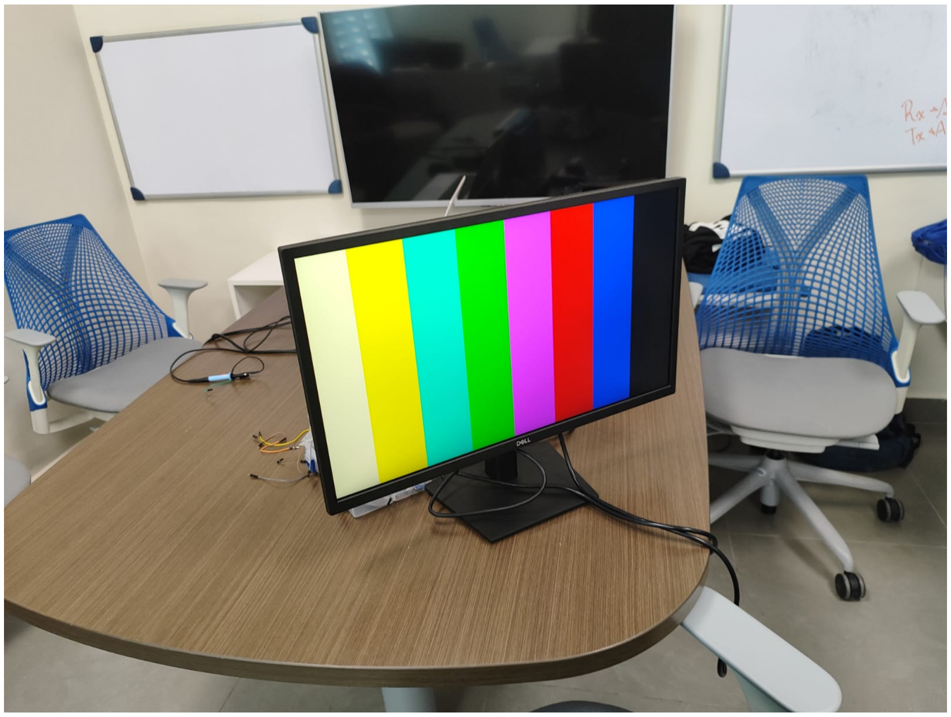


Figure 24: VGA Demonstration in PC screen

7 Accelerometer

7.1 Functionality

This module takes the raw output of the board built-in accelerometer which changes rapidly and is very prone to noise and calculates an average value taking into account 128 samples, and then displays these values (considering X and Y axes) using the 7 segment displays. Finally, the module its using a signed value signal to consider the current sign of the accelerometer output. If the sign is negative, it turns on a LED signal for the corresponding axis, respectively.

7.2 RTL View

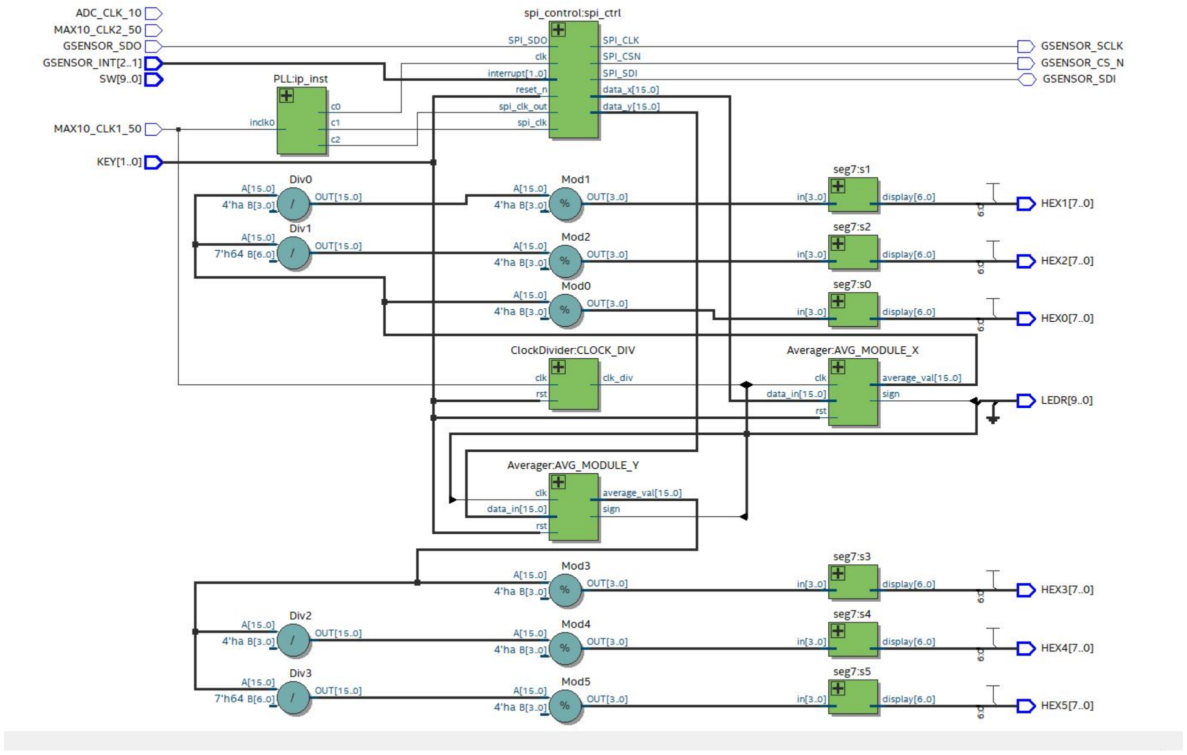


Figure 25: RTL view Top Accelerometer Module

In this RTL view it can be appreciated that to read accelerometer data there are many modules involved. There is the SPI module for interacting with the sensor, clock dividers and PLL to match operating and communication frequencies, 7 segment display handlers and finally the custom modules to calculate average value of output data from the accelerometer in the X and Y axes. Below, an image that shows the internal configuration of the averager module is shown.

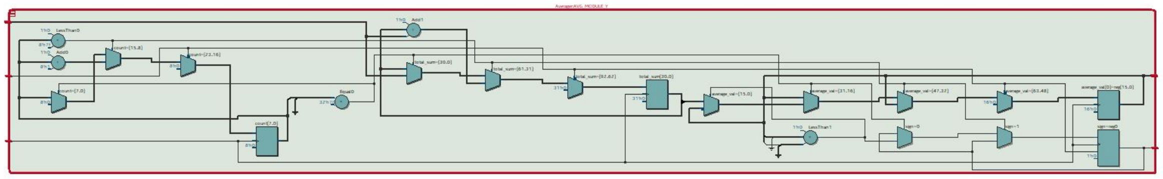


Figure 26: RTL view Averager Module

7.3 Physical Tests

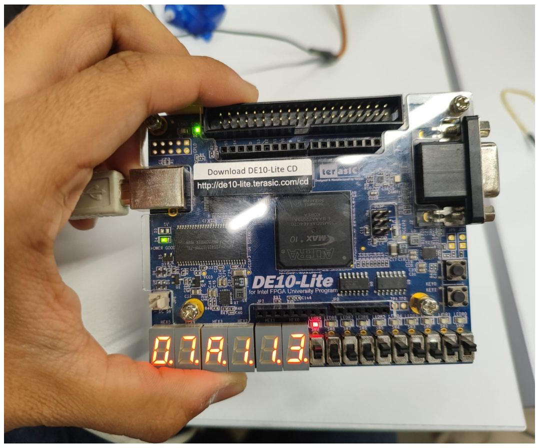


Figure 27: Physical Test 1 - Positive Output

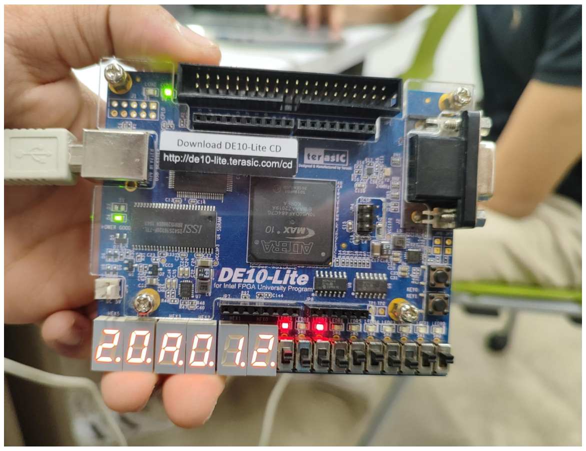


Figure 28: Test 2 - Negative Output

8 Conclusions

During the development of these practices, I observed that working with FPGAs can be challenging because its quite different from working with other platforms like micro controllers, where a layer of abstraction is provided, so the user does not interact with hardware at the level that working with FPGAs require. I realized that one of the best practices while working with hardware description languages is to divide all the logic into specific modules that are in charge of a single job, "modularize" everything. Personally, I think I may prefer working directly with microcontrollers, but am aware that FPGA's offer some advantages that are crucial to some applications, regarding quickness, low energy use and high logic reliability, so I consider very important learning more about this technology to leverage its best qualities.

9 GitHub Repo

To access the course codes, click on this link to the GitHub Repo

References

[1] Electronoobs. Uart on fpgas. Online. https://electronoobs.com/eng_circuitos_tut26.php.