Computer Engineering Department
Module Title: Microprocessor Design
Module Code: 21COMP03H

ARM CORTEX-M MICROPROCESSOR

Simulation and implementation

PRESENTED TO
Dr. Ihab Adly

PRESENTED BY
YOUMNA BASSIOUNY 187552
MARYAM KHALED 185629
FATIMA ELMALLA 192551
SALMA SAMEH 182635

DECEMBER 2021

Table of Contents

List of figures

Introduction

The goal of this project is to exchange characters data with a PC through UART protocol. This will be done be verifying the data coming out of the UART and viewing the output on a compiler.

The two signals in UART are named as:

- Transmitter (Tx)
- Receiver (Rx)

The main goal of a transmitter and receiver line for each device is to transmit and receive serial data intended for serial communication, UART uses only two wires for its transmitting and receiving ends.

Number of wires	2
Speed	9600, 19200, 38400, 57600, 115200, ...
Method of transmission	Asynchronous

Table 1: UART characteristics

System Design

Block diagram design of Cortex-M0 processor components

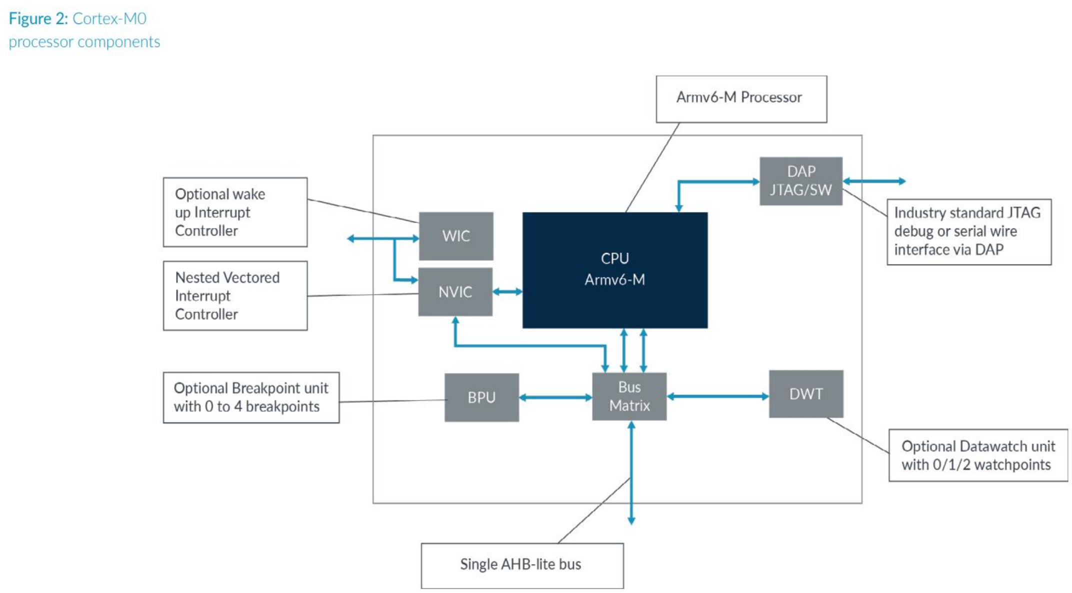

Figure 1: Cortex-M0 processor components

System of Cortex-M0

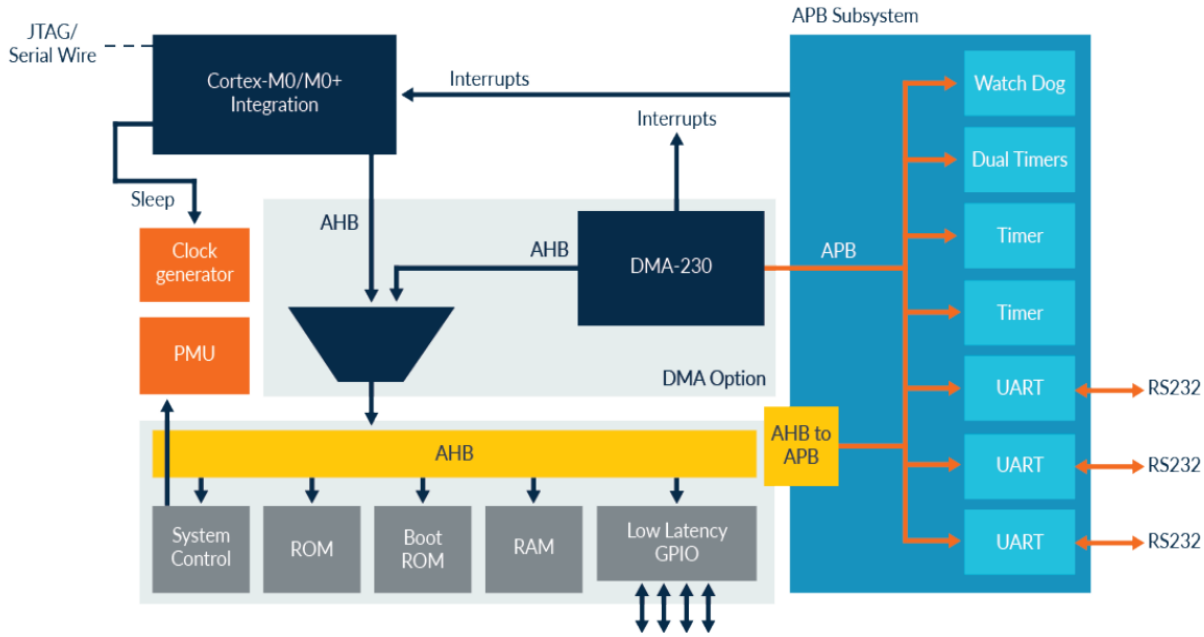

Figure 2: System of Cortex-M0

Design flowchart

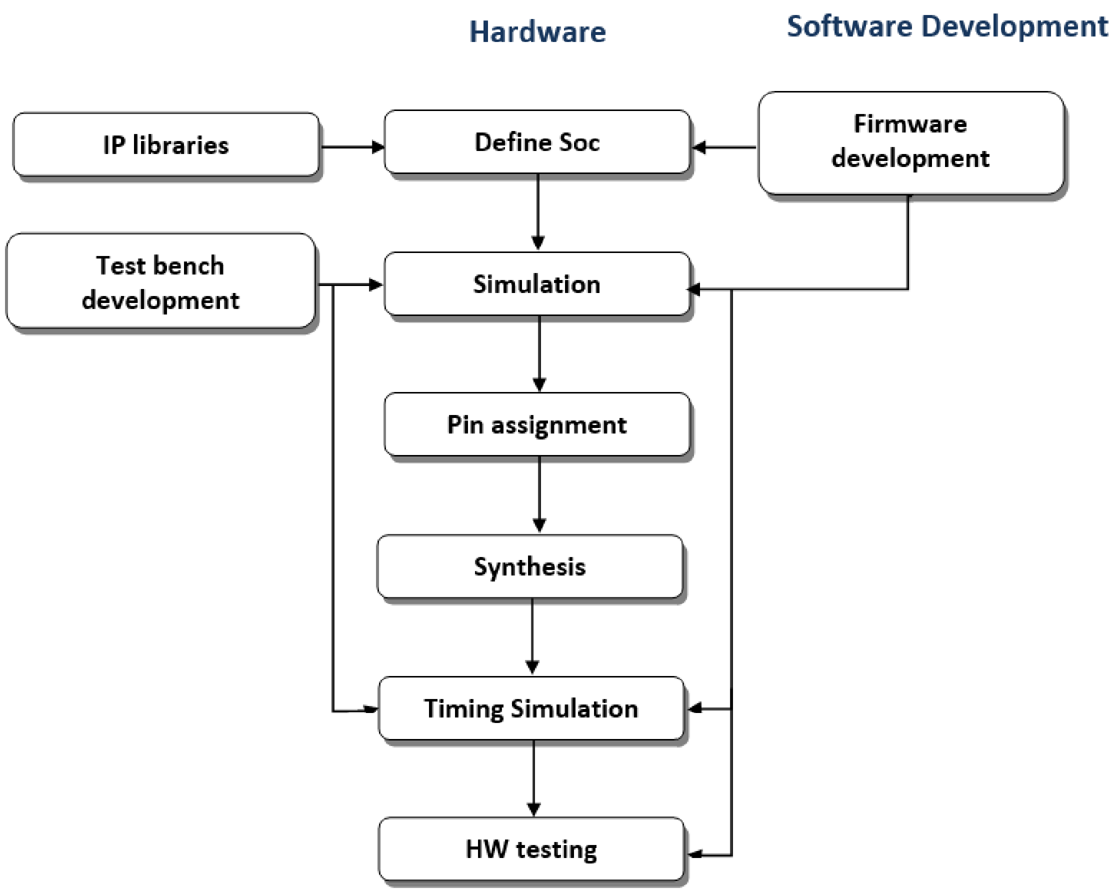

Figure 3 Design flowchart

Simulation and Implementation results:

Keil uVersion5:

- ➢ Keil uVersion5 was used to write the C code and convert it into (hexa-decimal) hex file. The file contains two print statements “hello world” and “test passed” to be transmitted from the FPGA to the computer through the UART port.
- ➢ Next, the file name is changed to image.c to match its name in the files provided kindly by Dr. Ihab, the path will also be changed to: "..\edu_m0_v2\edu_m0\quartus\cmsdk_mcu\image.hex"

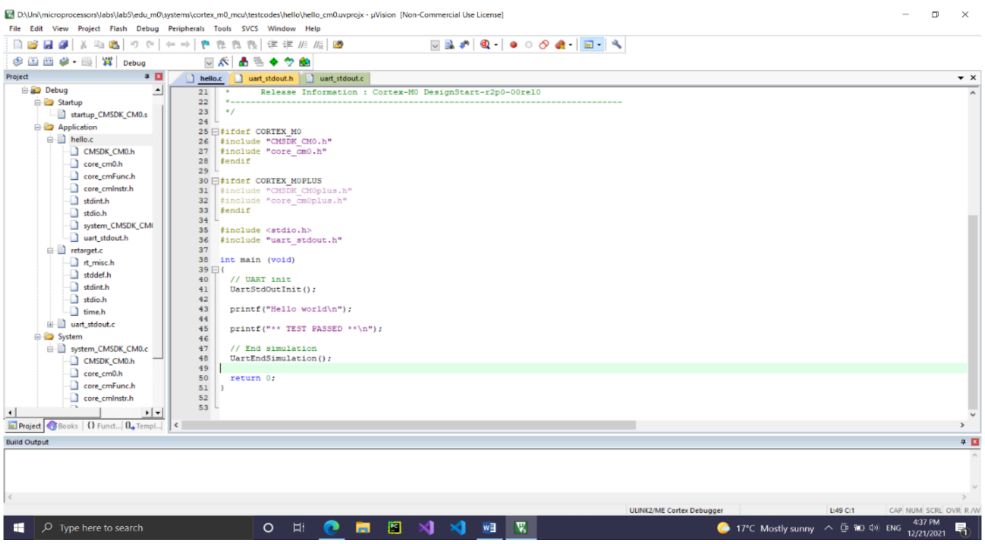

Figure 4 Keil – hello.c

- ➢ This file (uart_stdout.c) was used to change the baud rate. Baud rate is the rate at which information is transferred in a communication channel. Baud rate is commonly used when discussing electronics that use serial communication.

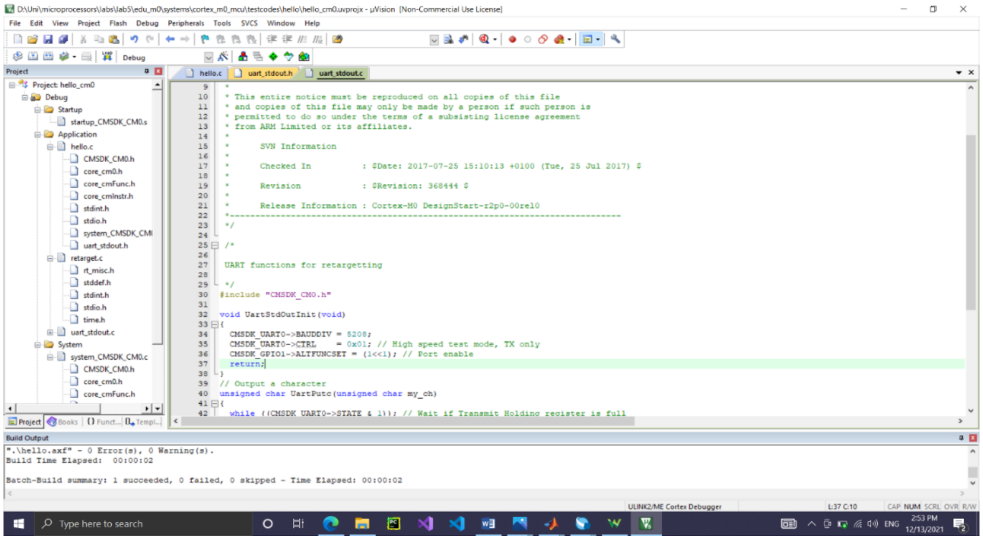

Figure 5 Keil -- uart_stdout.c

ModelSim Simulation

- ➢ Now open Visual Studio Code and create a work library and run this command “”vsim work.tb_cmsdk_mcu””. ModelSim will run the simulation for about 60000ns.

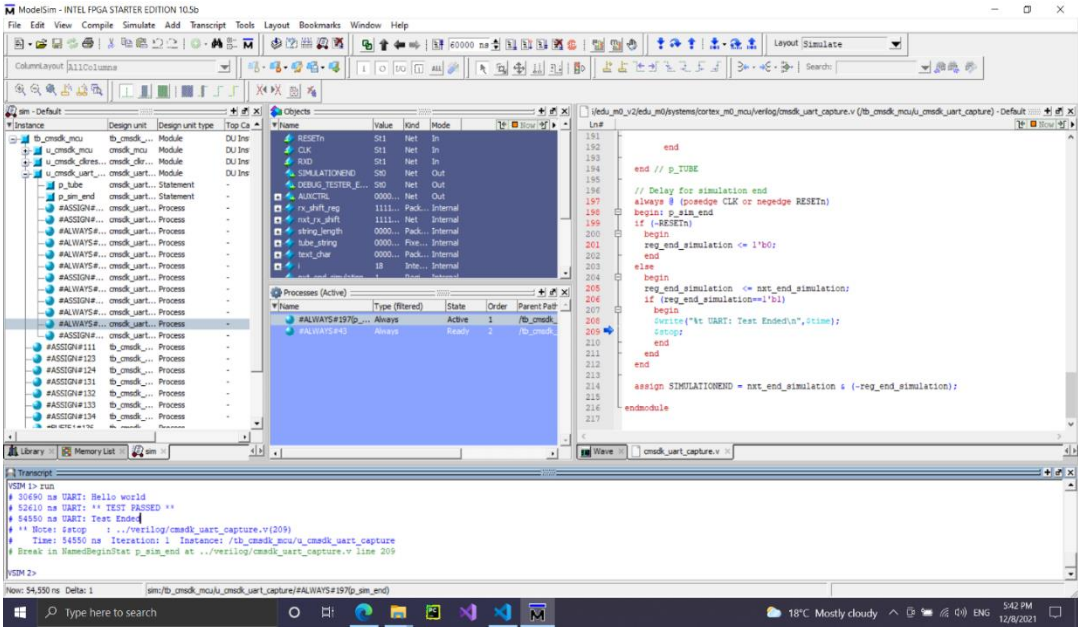

Figure 6 ModelSim simulation

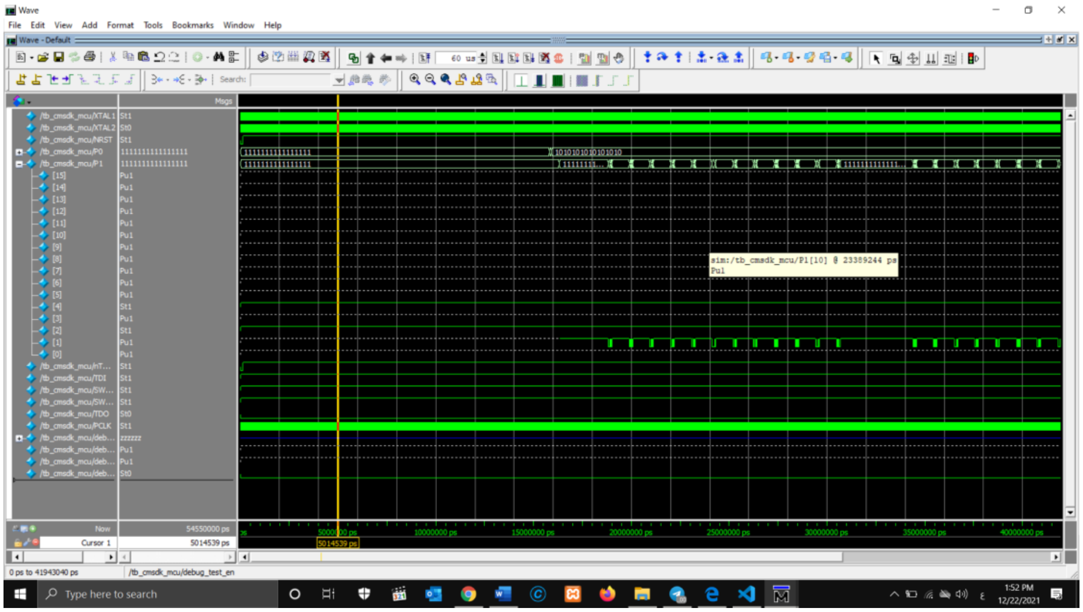

Figure 7 ModelSim simulation waves.

Quartus Simulation

- Create a folder by the name Quartus in the edu_m0 folder. Now create a new wizard project then compile the cmdk_mcu folder. After the compilation was successful with no errors.

- Next connect the FGPA to your computer, clink on programmer, choose the port that connects your computer with the FPGA, then click start. Make sure that the process has been done successfully.

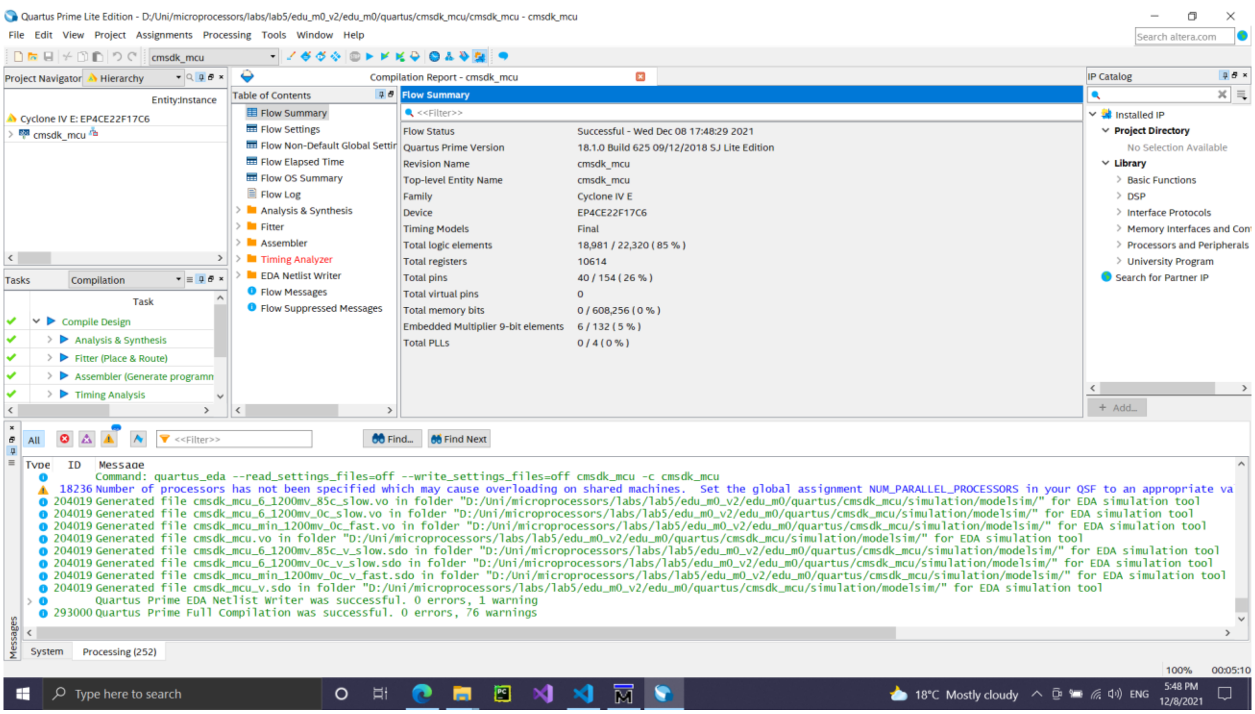

Figure 8 Quartus simulation

Figure 9 FPGA

Waveforms Simulation

- Next, we want to check if the data is transmitted through UART is not corrupted. This will be done by connecting the logic analyzer to the FPGA. A logic analyzer is an electronic instrument that captures and displays multiple signals from a digital system or digital circuit.

Figure 10 Logic analyzer

The connection is done as followed:

- Create a common ground between both devices (FPGA and logic analyzer).
- Connect P1 in the FPGA which represents the UART port to the logic analyzer

Next, recompile your file using Quartus and open Waveforms. Waveforms makes it easy to acquire, visualize, store, analyze, produce, and reuse analog and digital signals, it seamlessly connects to our USB portable oscilloscope.

Waveforms steps:

1. Choose logic button
2. Add channel
3. Add UART
4. Define the trigger

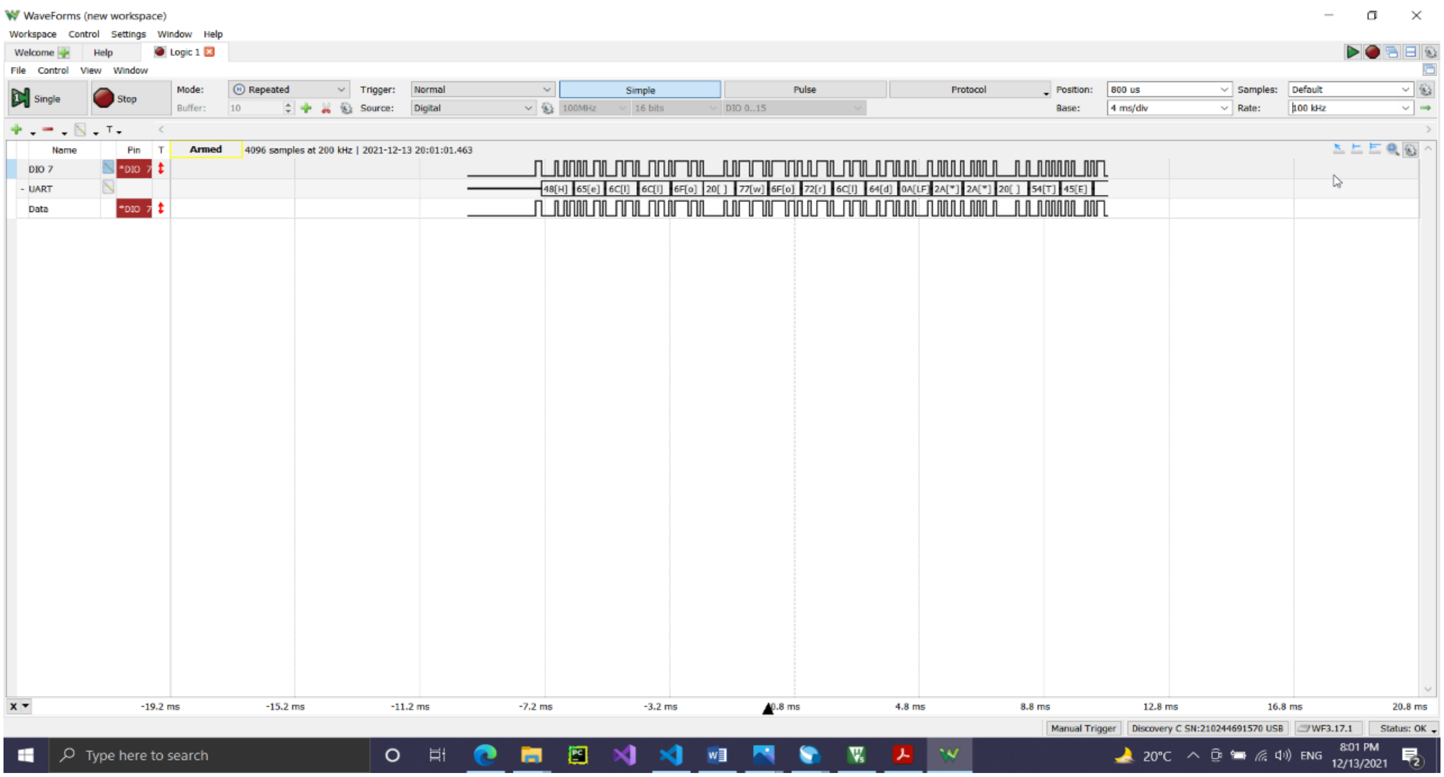

Figure 11 Waveform simulation

- Last step, use Docklight as a compiler to view the output transmitted from the FPGA. First, connect the FPGA to a module USB to serial 3.3 volt.

Figure 12 USB to serial module

The connection will be done as follows:

- Connect a common ground between both devices (FPGA and module USB).
- Connect the receive pin (Rx) in the module USB to the P1 port in the FPGA.
- Connect both devices to the computer.

Next, open Docklight and click start communication to view the output as following:

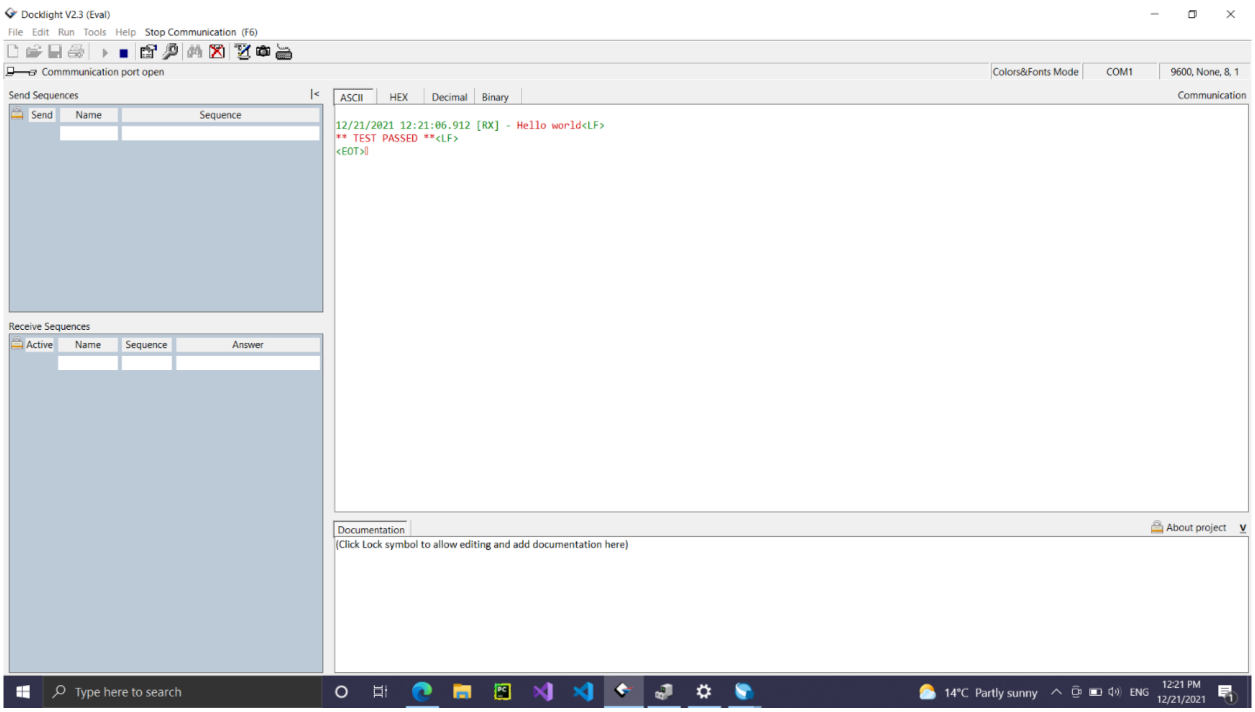

Figure 13 Docklight communication