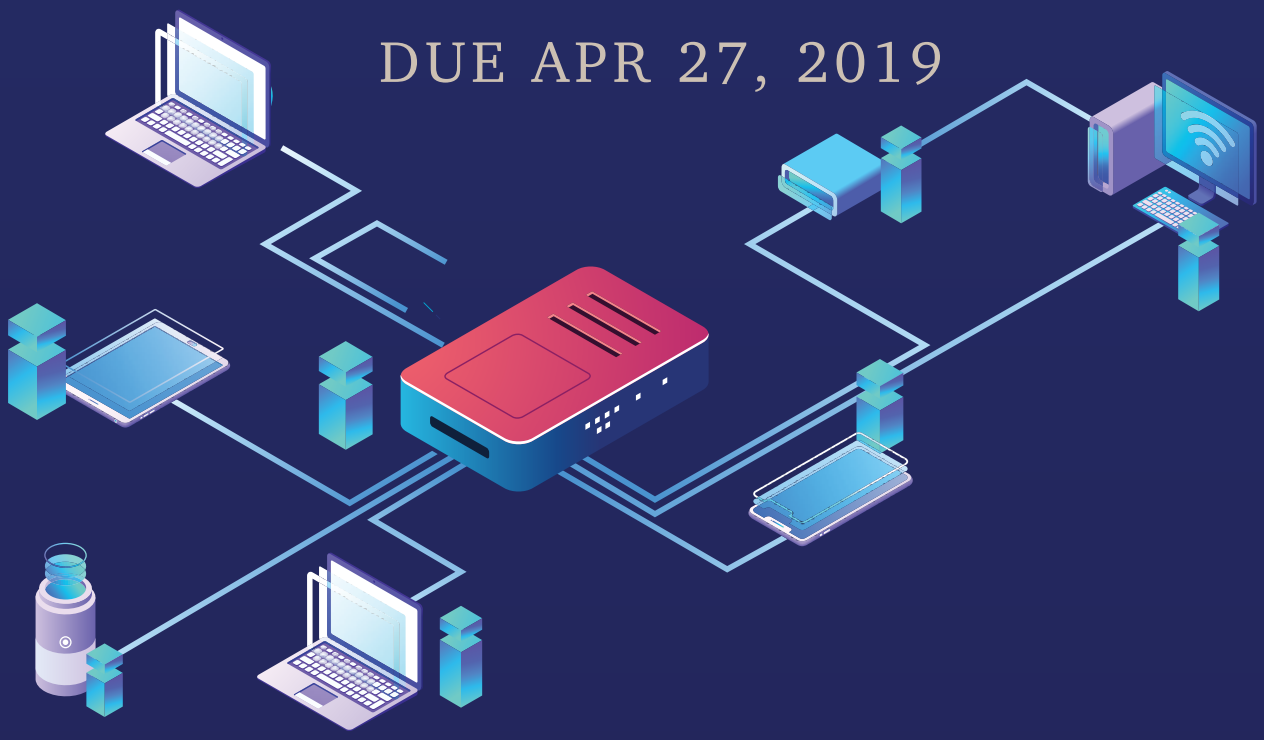
FINAL PROJECT REPORT

SIMPLE ROUTER USING FBGA

ABDULRHMAN ALHAJ ALI
ID : 1637521

ABDULRHMAN SOBHY
ID : 1647817

DUE APR 27, 2019



EE 460:

Digital Design II

Final Project

Table of Contents

Introduction

A router is a networking device that forwards data packets between other devices networks based on the destination address in the packet. So, the function of any router is receiving a packet from the sender device (that contain the data message and the IP of the receiver device and the check error bits) then send the data message to the destination device. Our project is designing a simple router by using FPGA.

The description of the project

Our project is designing a simple router and our router can receive a packet. After the router receives the packet it will check if there any error occurs through the journey of the packet if there is no error the router will send the data message which in the packet to the destination port depending on the destination byte which in the packet. If there is an error occur during the journey of the packet the router will detect the error and it will not send the data message to the destination port and the error flag will be asserted.

The packet

The packet consists from 25-bits and these 25 bits and it divided as follow:

- From 0 to 8 (nine bits): check sum bits for error detection and it is 9 bit to avoid the overflow.
- From 9 to 16 (eight bits): data bits.
- From 17 to 24 (eight bits): destination bits. if the data less than 128 then it should be routed to port1 (From 00000000 To 01111111) and if it equal to 128 or greater than 128 should be routed to port2 (From 01111111 To 11111111)

The design and implementation

The design and implementation stages

Stage1:

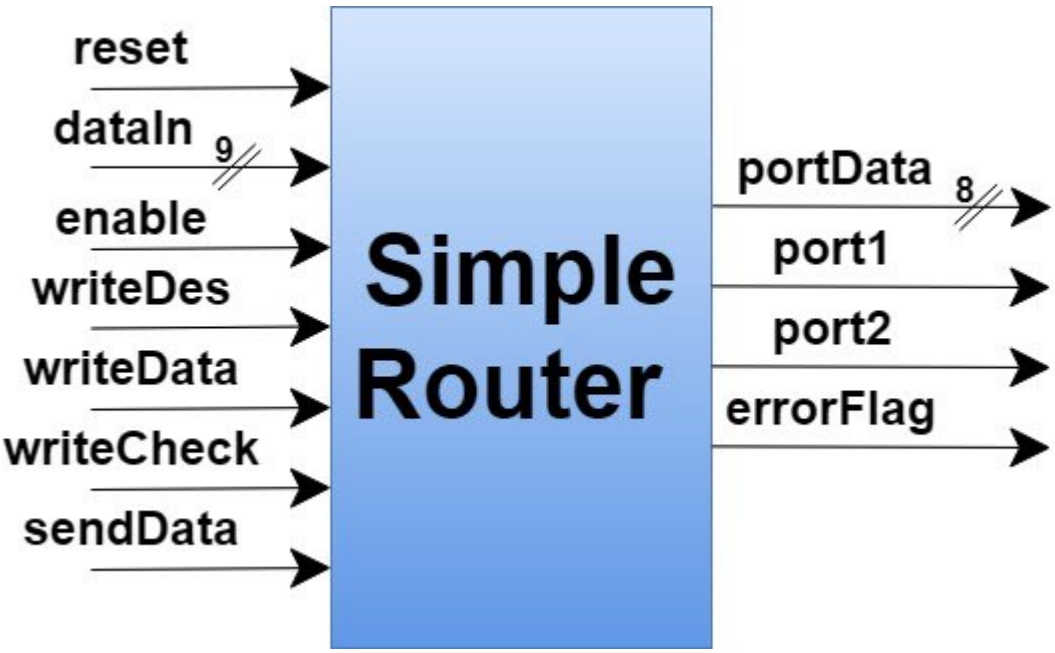
We design a simple router that its packet input from the FPGA's switches and the output data display in the FPGA's four 7-segment display and the output port number display by the FPGA's LED.

Stage2:

We develop the design in stage 1. Since the simple router will receive the packet from the Arduino by SPI protocol and the output data display in the FPGA's four 7-segment display and the output port number display by the FPGA's led.

Stage 1

Top Level Block Diagram



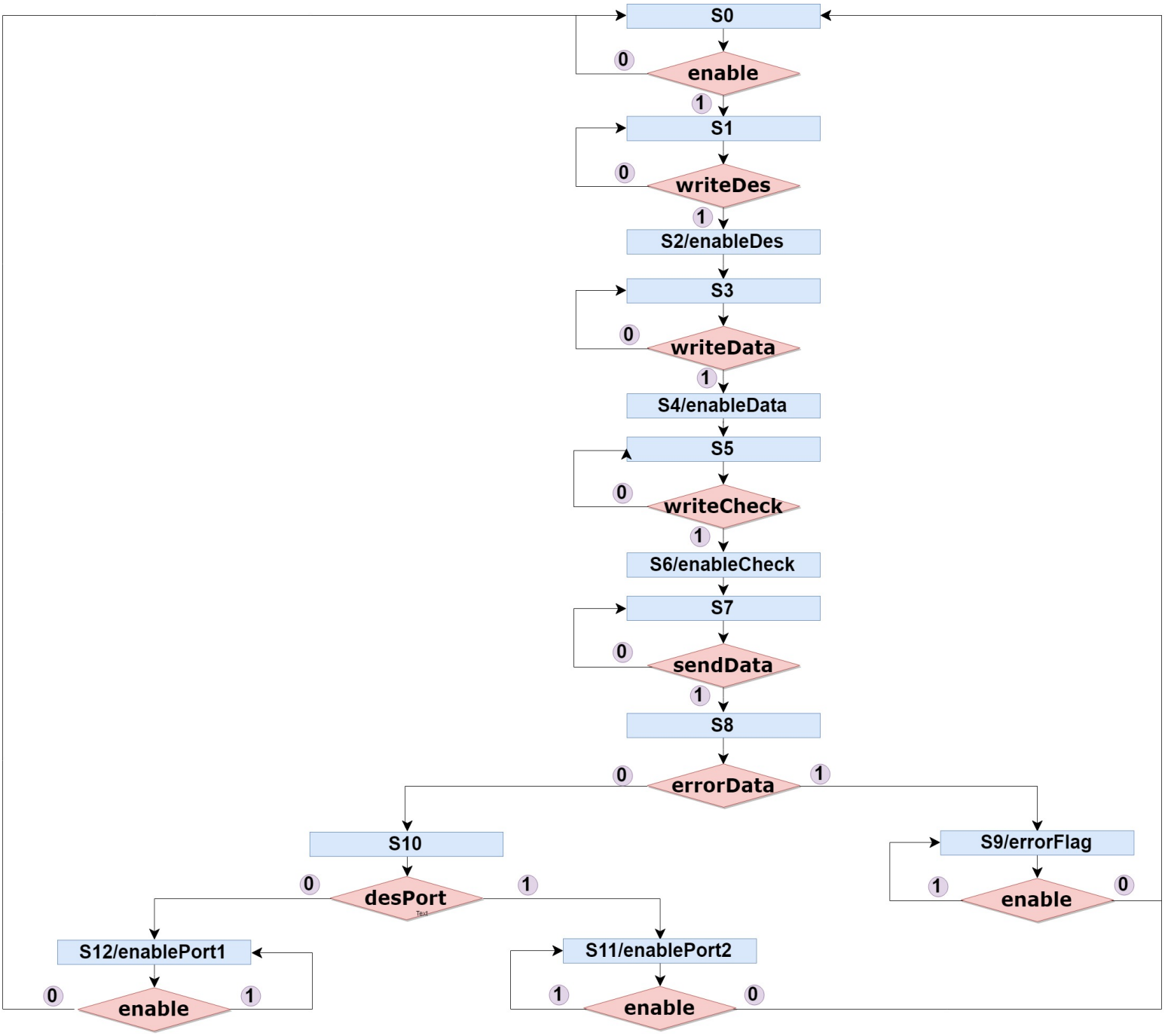
The input signals

1. **Reset:** to reset the router
2. **Enable:** to enable the router to inter the packet
3. **dataIn(9 bits):** the input data for the router and it can be the destination bits or the data bits or the check sum bits depend on other inputs
4. **WriteDes:** to define the dataIn as a destination bits
5. **WriteData:** to define the dataIn as a data bits
6. **WriteCheck:** to define the dataIn as a check sum bits.
7. **SendData:** to send the packet after entering the destination, data, and check sum

The output signals

1. **PortData:** the data sent from the sender to the port
2. **Port1:** to will be asserted if the data rout to port 1
3. **Port2:** to will be asserted if the data rout to port 2
4. **errorFlag:** it will be asserted if there is an error occur during the packet journey.

The Control Unit Design



The input signals

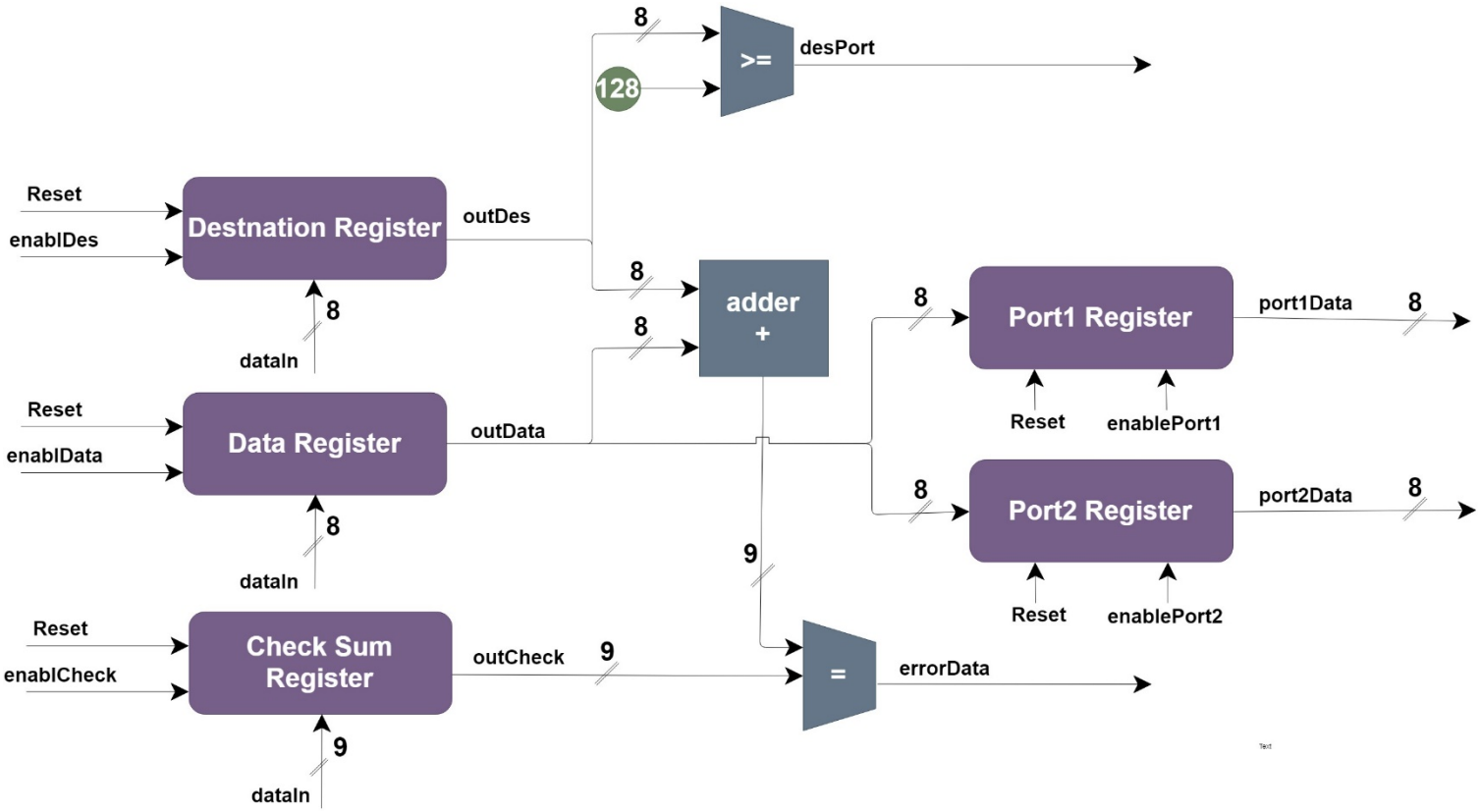
- Input signals from the user:
  1. **enable**
  2. **writeDes**
  3. **writeData**
  4. **writeCheck**
  5. **sendData**

- The input signal from the Datapath:
  1. **errorData**
  2. **desPort**

The output signals

1. **enableDes:** to enable the destination register.
2. **enableData:** to enable the data register.
3. **enableCheck:** to enable the check sum register.
4. **errorFlag:** to tell the user there is an error.
5. **enablePort1:** to enable port1 register.
6. **enablePort2:** to enable port2 register.

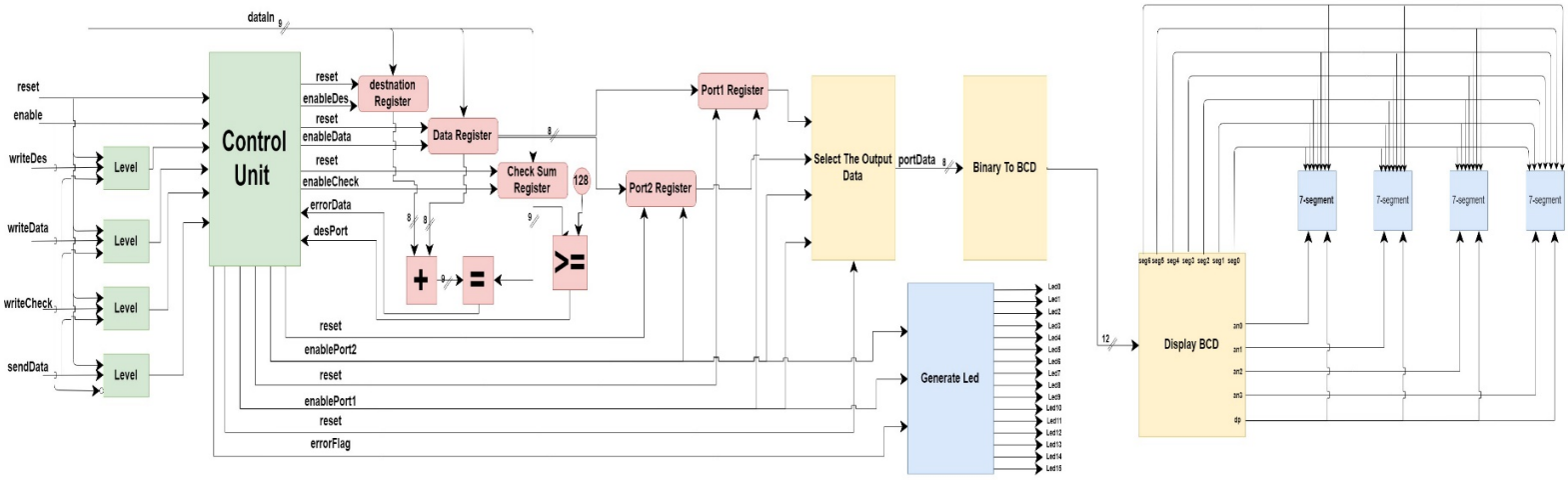
The Datapath design



The Datapath component

The component	The input	The output	The function
Destination register	1. dataIn(8 bit) 2. reset 3. enableDes	outDes	Store destination bits when enableDes asserted
Data register	1. dataIn(8 bit) 2. reset 3. enableData	outData	Store data bits when enableData asserted
Check sum register	1. dataIn(9 bit) 2. reset 3. enableCheck	outCheck	Store check sum bits when enableCheck asserted
Adder	1. outDes 2. outData	adderValue	To add the out from destination and data registers
equalComparator	1. adderValue 2. outCheck	errorData	To compare the out from adder with the check sum register to check the error
GreaterCompatrtor	1. outDes 2. 128 constant	desPort	To compare the out from destination register with 128 to define the output port
Port1 register	1. outData 2. reset 3. enablePort1	Port1Data	Store the out from the data register when enablePort1 asserted
Port2 register	1. outData 2. reset 3. enablePort2	Port2Data	Store the out from the data register when enablePort2 asserted

The complete design of stage1



- The image is small and not clear because its size. We will attach the image with the report

As you can see from the complete design image that we added some component when we implement the design to display the results and to enable the user to enter the packet without any error.

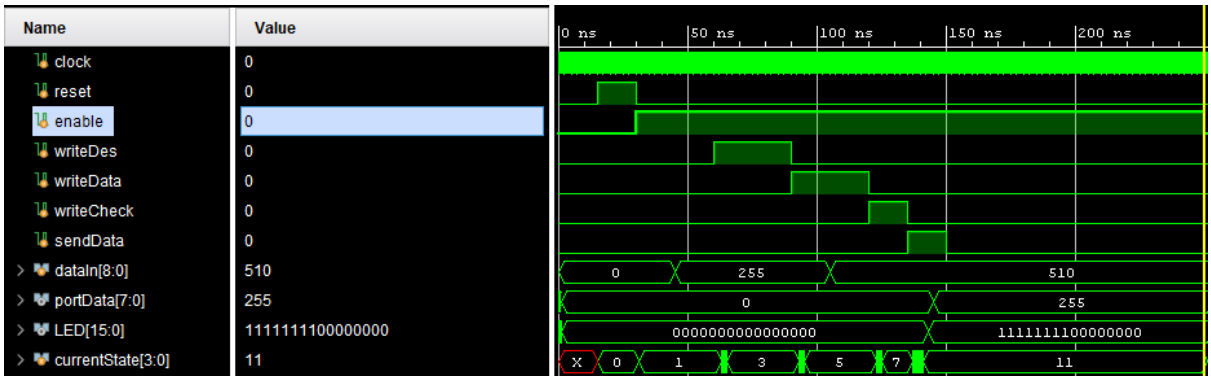
The added component

The component	The input	The output	The function
Level	1. reset 2. neg 3. pos	Level	To control the push button in FPGA. If pos is high the level output be high until the neg be high
Select the output data	1. outPort1 2. outPort2 3. enablePort1 4. enablePort2	portData[7:0]	To select the output data of the router depend on the activate port
Binary to BCD	PortData[7:0]	outBCD[11:0]	To convert the output from binary form to BCD form to use it in 7-segment
Display BCD	outBCD[11:0]	1. an[3:0] 2. dp 3. seg[6:0]	Generate the appropriate wire to operate the 7-segment
7-segment	1. an[3:0] 2. dp 3. seg[6:0]		Display the output
Generate led	1. enablePort1 2. enablePort2 3. errorFalg	Led[15:0]	Display either the error flag or the activate port

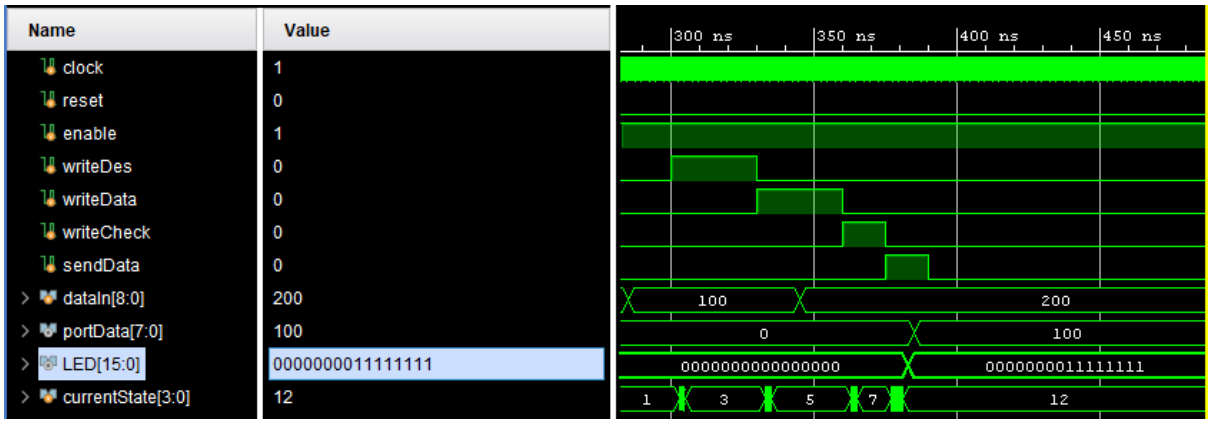
The implementation

We used a Vivado 2018.1 and Verilog language to implement our and the code attach with the report

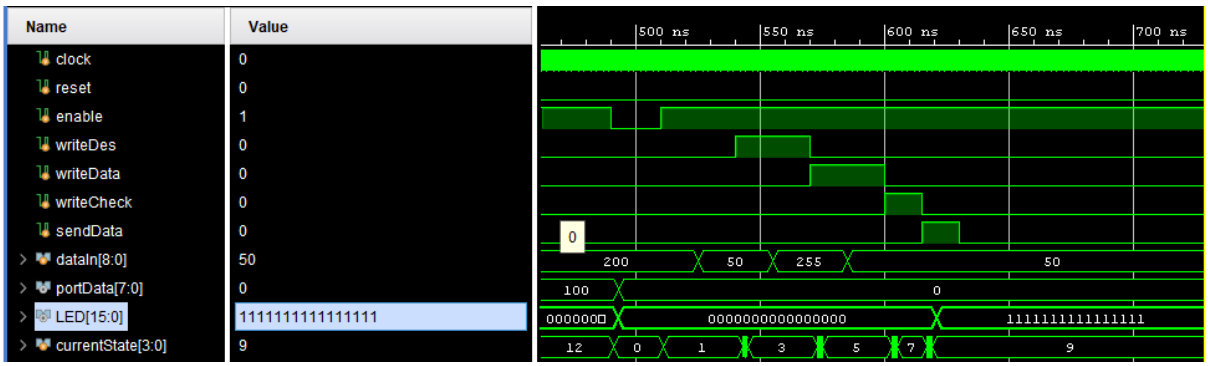
The simulation results



As we can see from the above figure when the destination and the data equal to 255 and the check sum equal 510, the 16-bit of the led output refer to the port is port2 and the portData equal to the outData register 255



In this figure the destination and data register equal to 100 and the check sum register equal to 200. The output led refer to the port is port1 and the portData equal to the outData register 100

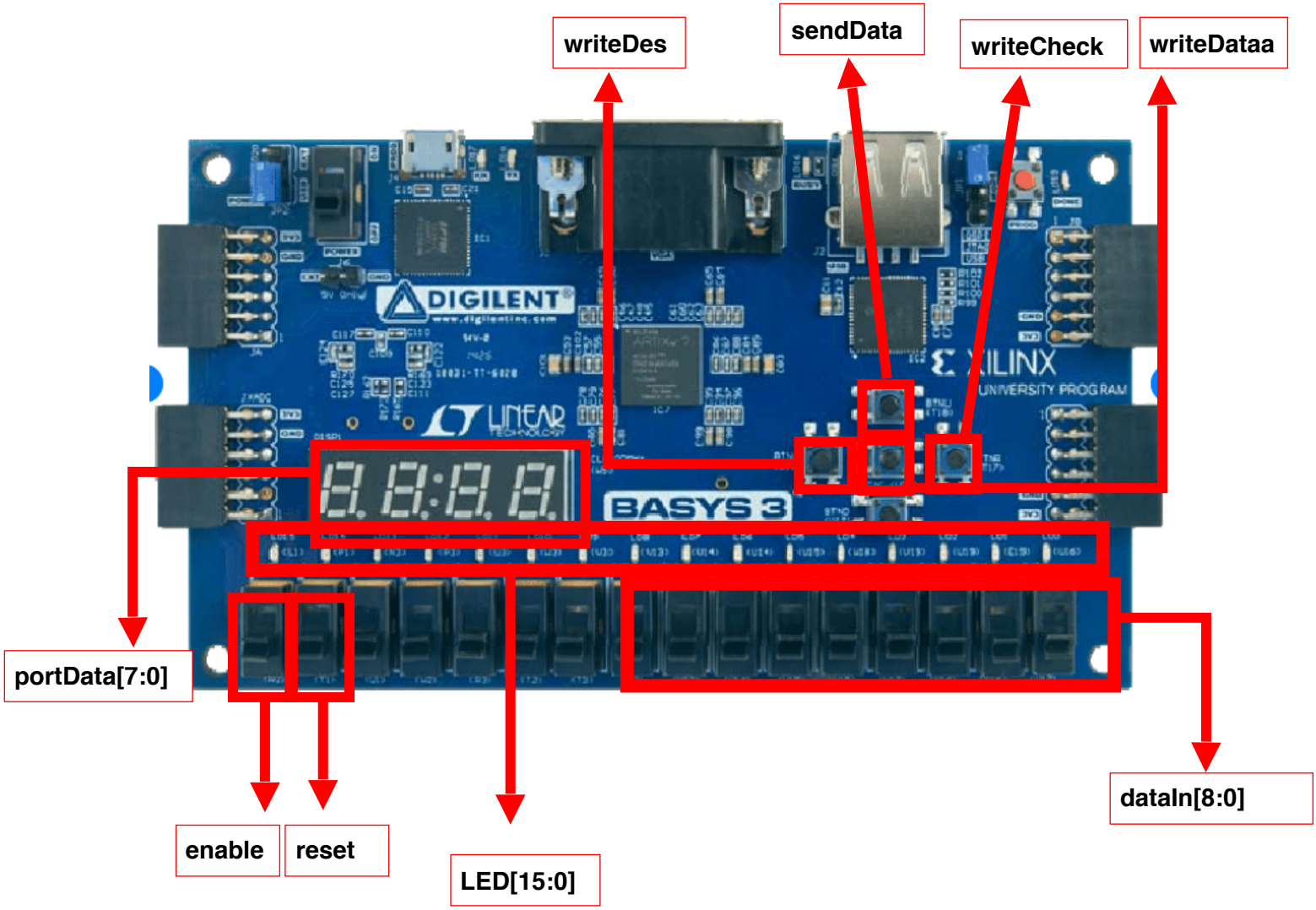


in this figure the destination register equal 50 and the data register equal 255 and the check sum register equal 50. The output led refer to there is an error where outDes+outData != outCheck.

Test the implementation

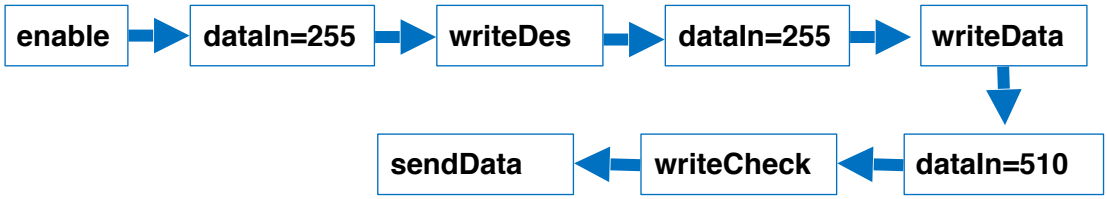
To test our implantation, we used a Basys 3 FPGA and we used the FPGA component to implement the input and output of our system.

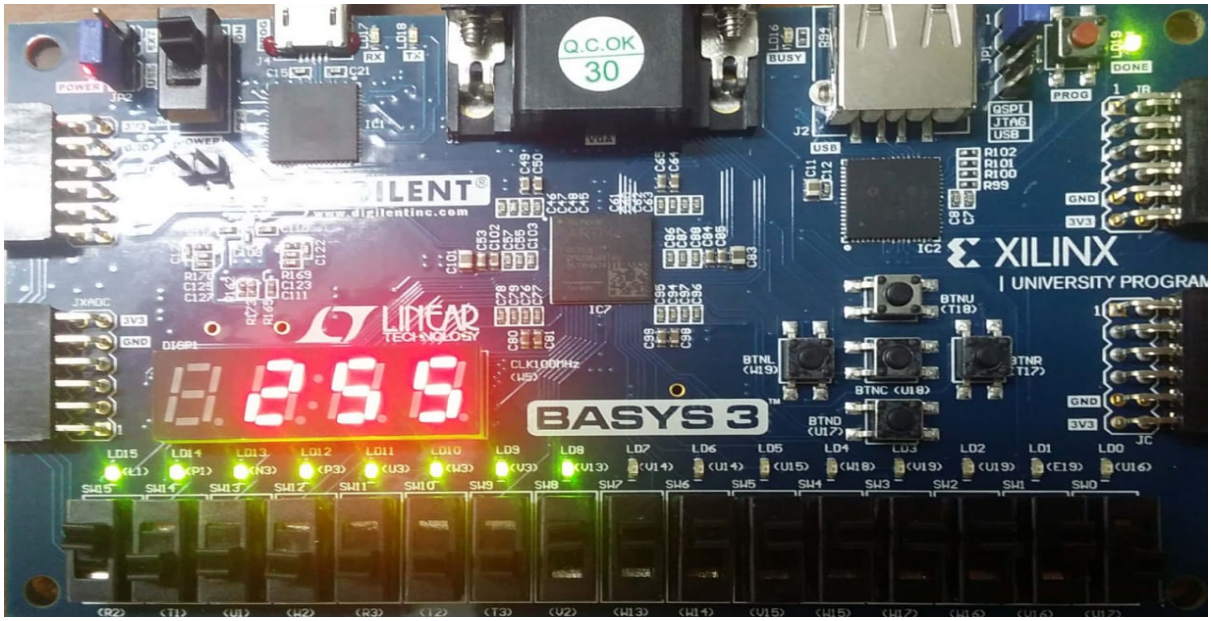
The input and outputs on the FPGA



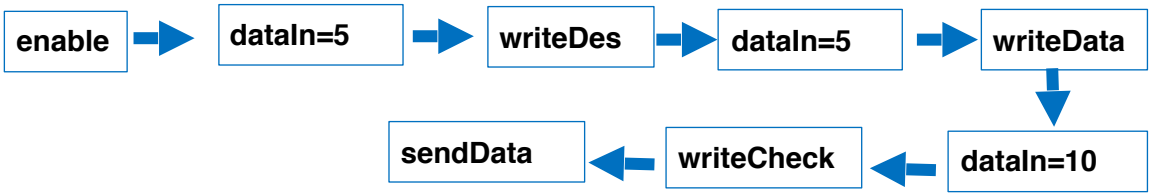
The test results

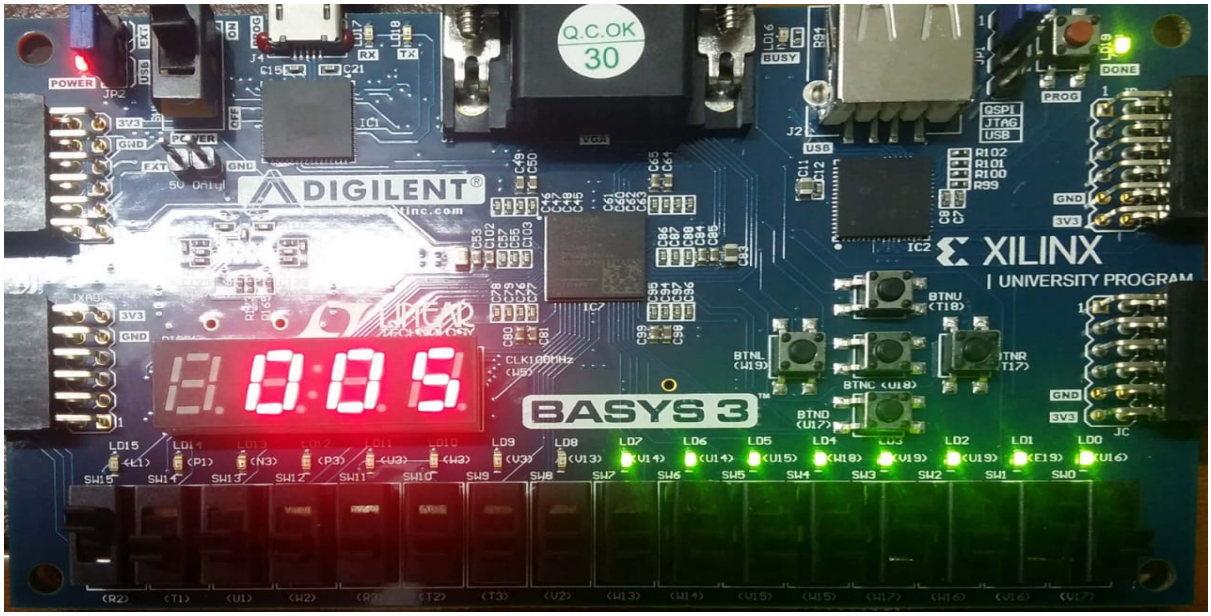
- First test:



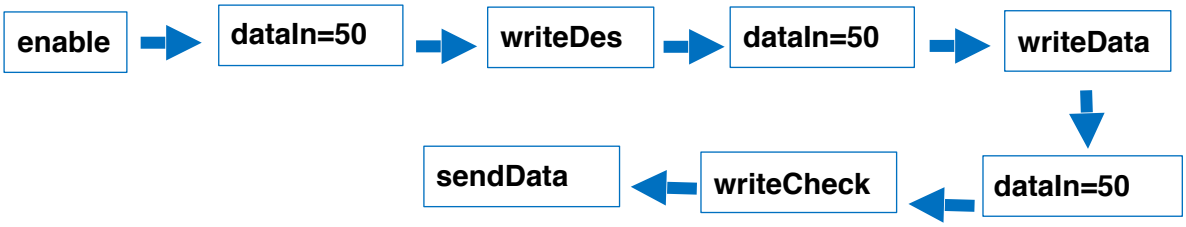

Q.C.OK
30
XILINX
UNIVERSITY PROGRAM
BASYS 3

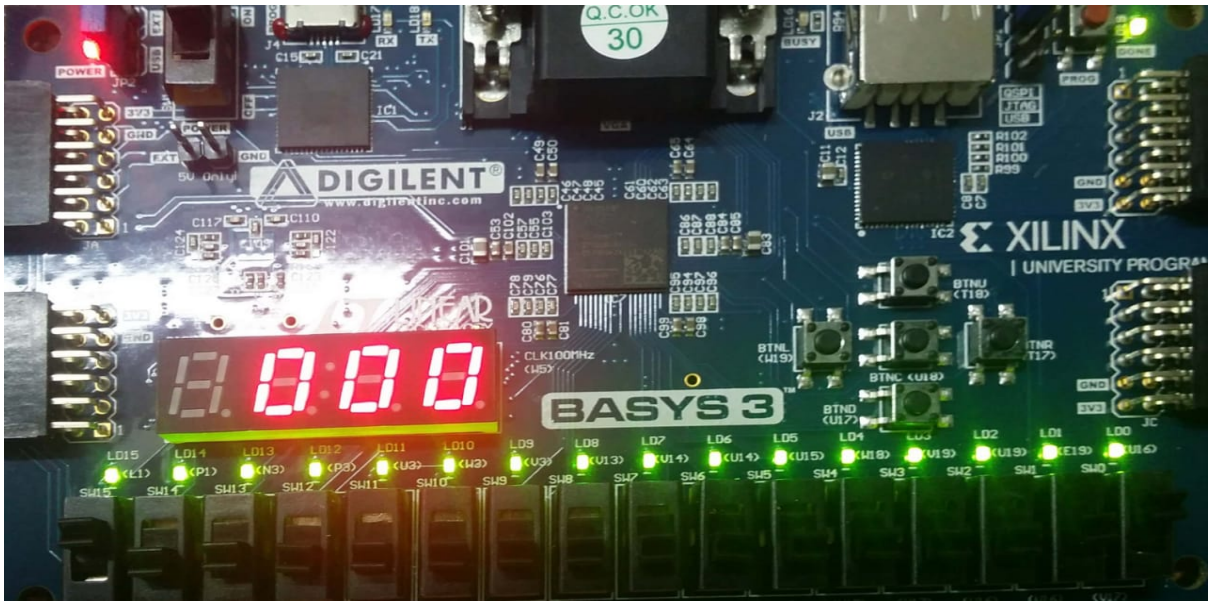
- Second test


enable
dataIn=5
writeDes
dataIn=5
writeData
dataIn=10
writeCheck
sendData


Q.C.OK
30
DIGILENT
XILINX
UNIVERSITY PROGRAM
BASYS 3

Third test


enable
dataIn=50
writeDes
dataIn=50
writeData
dataIn=50
writeCheck
sendData


DIGILENT
XILINX
BASYS 3

Stage 2

Top level block diagram

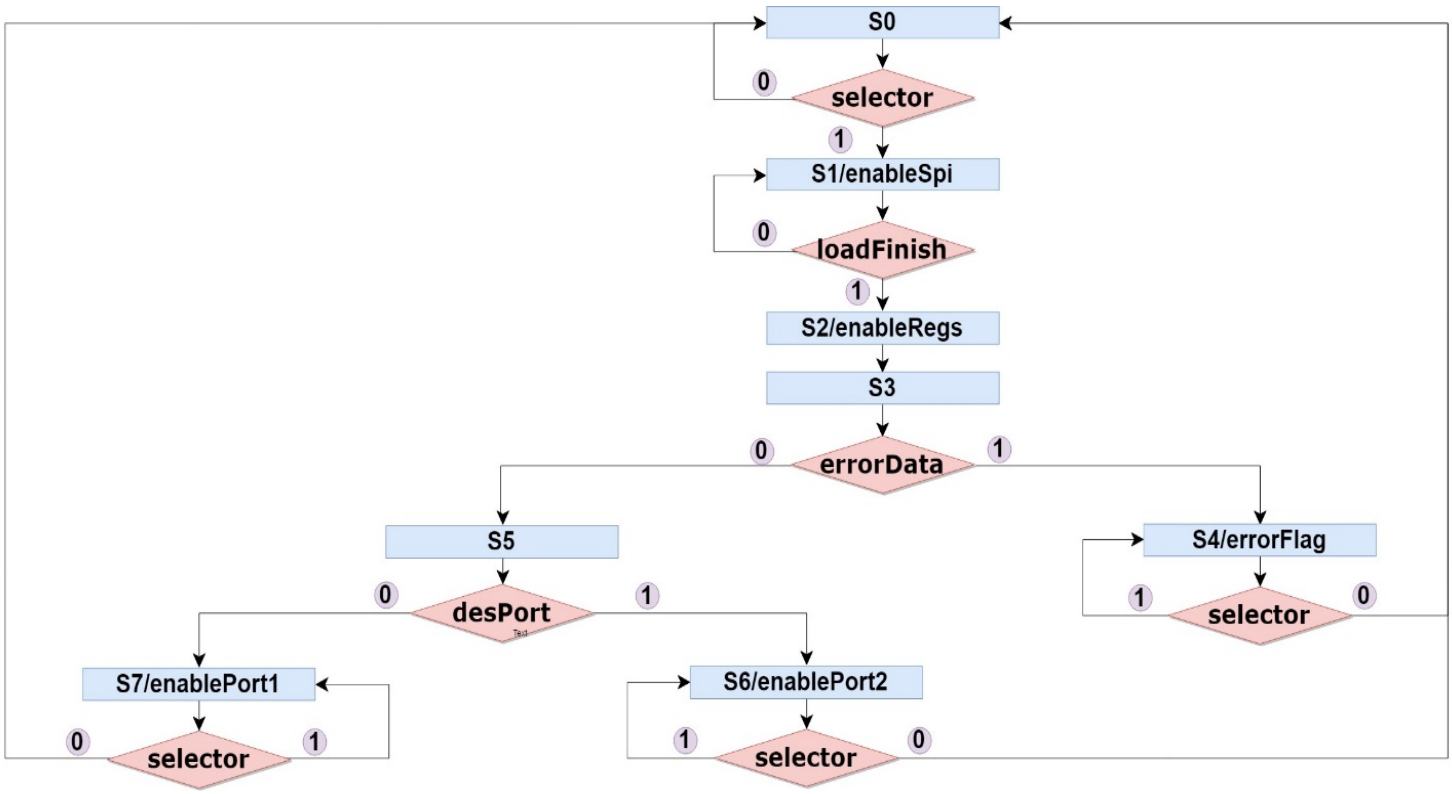
The inputs signals:

1. **bitIn:** the input bit from the master in the serial communication
2. **selector:** to begin send data from master to slave
3. **masterClock:** the clock generated by the master to control the serial communication
4. **reset:** to reset the system

The output signals:

1. **PortData:** the data sent from the sender to the port
2. **Port1:** to will be asserted if the data rout to port 1
3. **Port2:** to will be asserted if the data rout to port 2
4. **errorFlag:** it will be asserted if there is an error occur during the packet journey and sent to the master.

The Control Unit Design



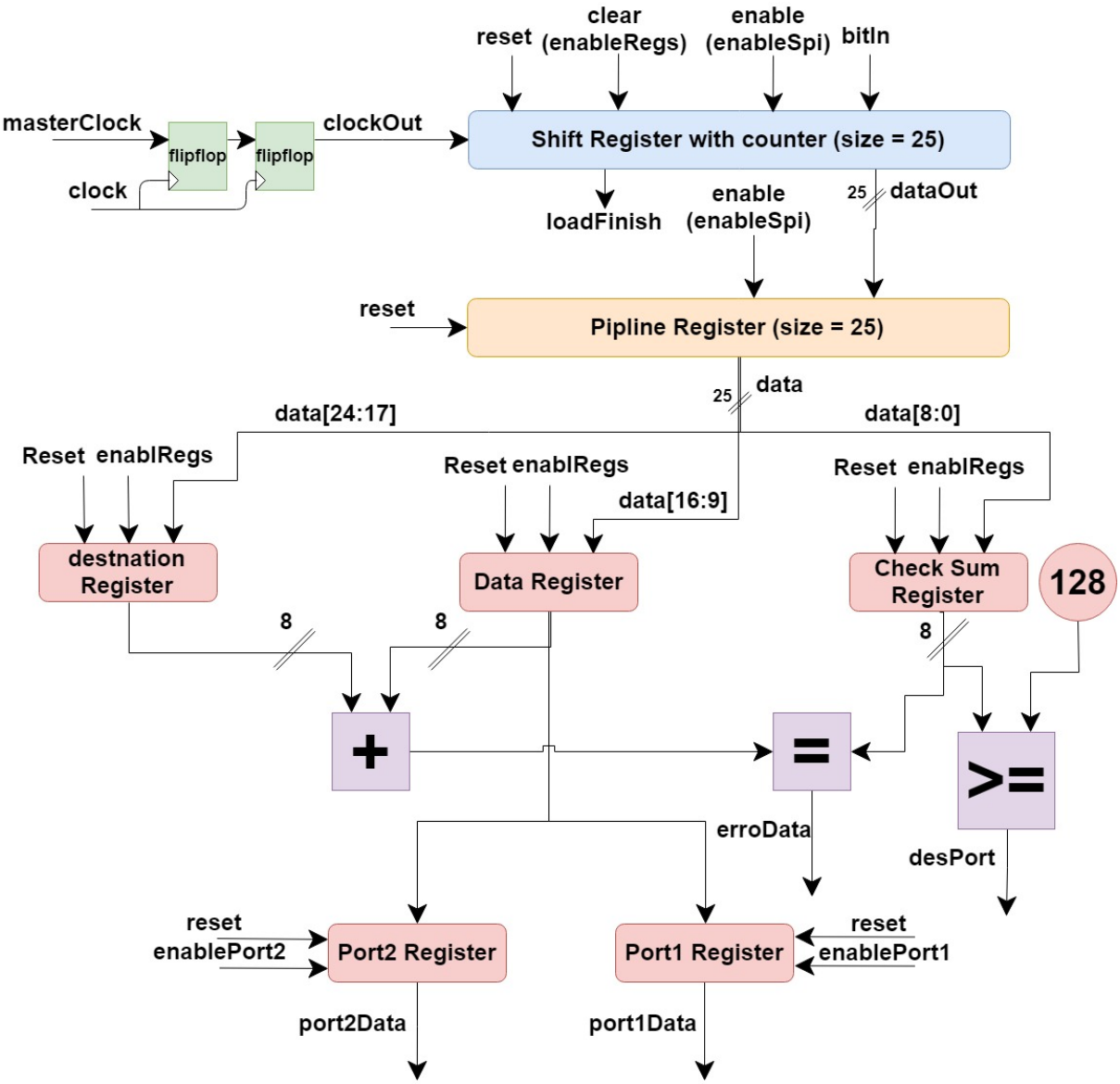
The input signals:

- the input signals from the master device: **selector**
- The input signals from the Datapath:
  1. **loadFinish**: it refer to the serial communication finish
  2. **errorData**
  3. **desPort**

The output signals:

1. **enableSpi:** to enable the shift register to receive the data from the master
2. **enableRegs:** to enable the destination , data and check sum register to store the data
3. **errorFlag**
4. **enablePort1**
5. **enablePort2**

The Datapath Design



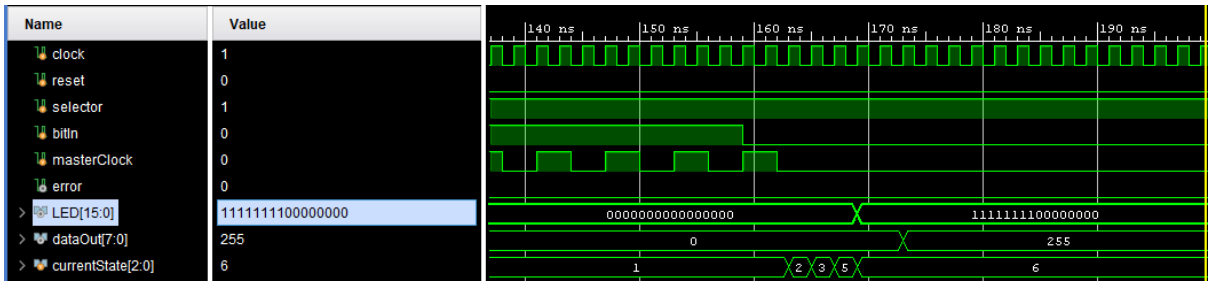
The Datapath component

Component	Inputs	Outputs	Function
Two flipflop	1. masterClock 2. FPGA clock	clockOut	To synchronize the master clock and the FPGA clock
Shift register	1. Reset 2. enableSpi 3. enableRegs 4. clockOut 5. bitIn	1. loadFinish 2. dataOut[24:0]	Receiving the bits from the master device when its enable asserted
Pipeline register	1. reset 2. enableSpi 3. dataOut[24:0]	Data[24:0]	To store the data from the shift register to use it
Destination register	1. reset 2. enableRegs 3. data[24:17]	outDes	Store destination bits when enableRegs asserted
Data register	1. reset 2. enableRegs 3. data[16:9]	outData	Store data bits when enableRegs asserted
Check sum register	1. reset 2. enableRegs 3. data[8:0]	outCheck	Store check sum bits when enableRegs asserted
Adder	1. outDes 2. outData	adderValue	To add the out from destination and data registers
equalComparator	1. adderValue 2. outCheck	errorData	To compare the out from adder with the check sum register to check the error
GreaterCompartor	1. outDes 2. 128 constant	desPort	To compare the out from destination register with 128 to define the output port
Port1 register	1. outData 2. reset 3. enablePort1	Port1Data	Store the out from the data register when enablePort1 asserted
Port2 register	1. outData 2. reset 3. enablePort2	Port2Data	Store the out from the data register when enablePort2 asserted

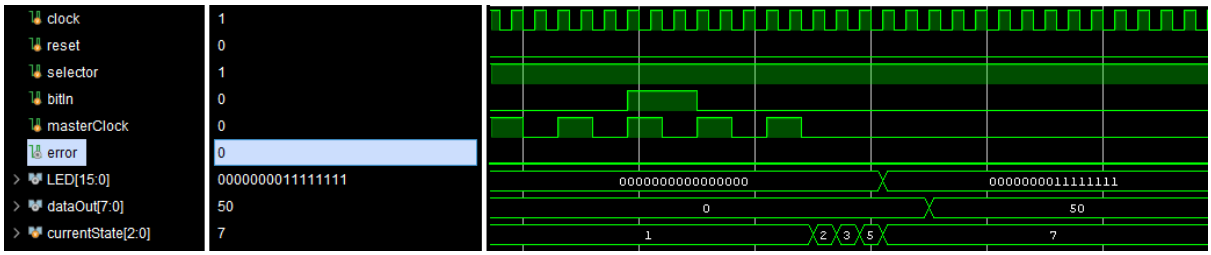
The implementation

We used a Vivado 2018.1 and Verilog language to implement our and the code attach with the report

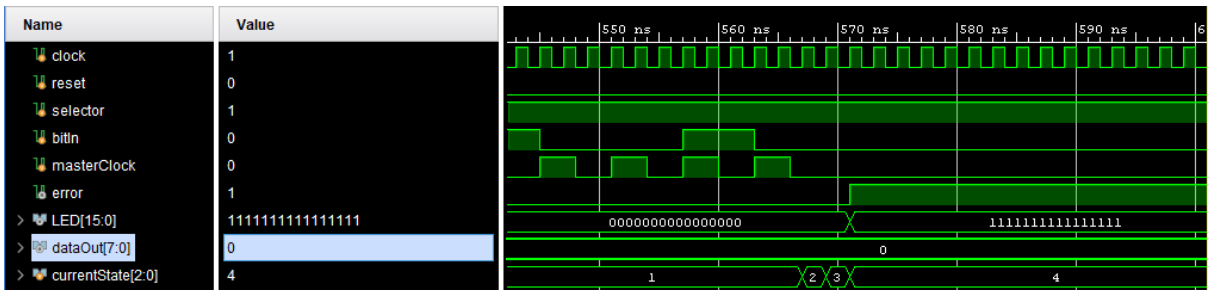
The simulation results



As we can see from the above figure when the destination and the data equal to 255 and the check sum equal 510, the 16-bit of the led output refer to the port is port2 and the portData equal to the outData register 255. The packet entered bit by bit and the packet bit equal 25'b1111111111111111111111110



In this figure the destination and data register equal to 50 and the check sum register equal to 100. The output led refer to the port is port1 and the portData equal to the outData register 50. The packet entered bit by bit and the packet bit equal 25'b00110010001100100010001100100

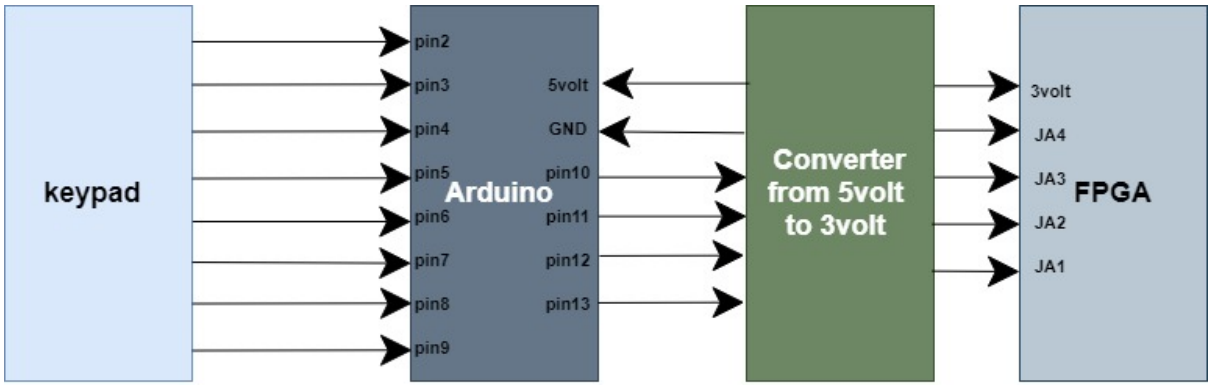


In this figure the destination register equal 50 and the data register equal 50 and the check sum register equal 50. The output led refer to there is an error where outDes+outData != outCheck. The packet entered bit by bit and the packet bit equal 25'b0011001000110010000110010

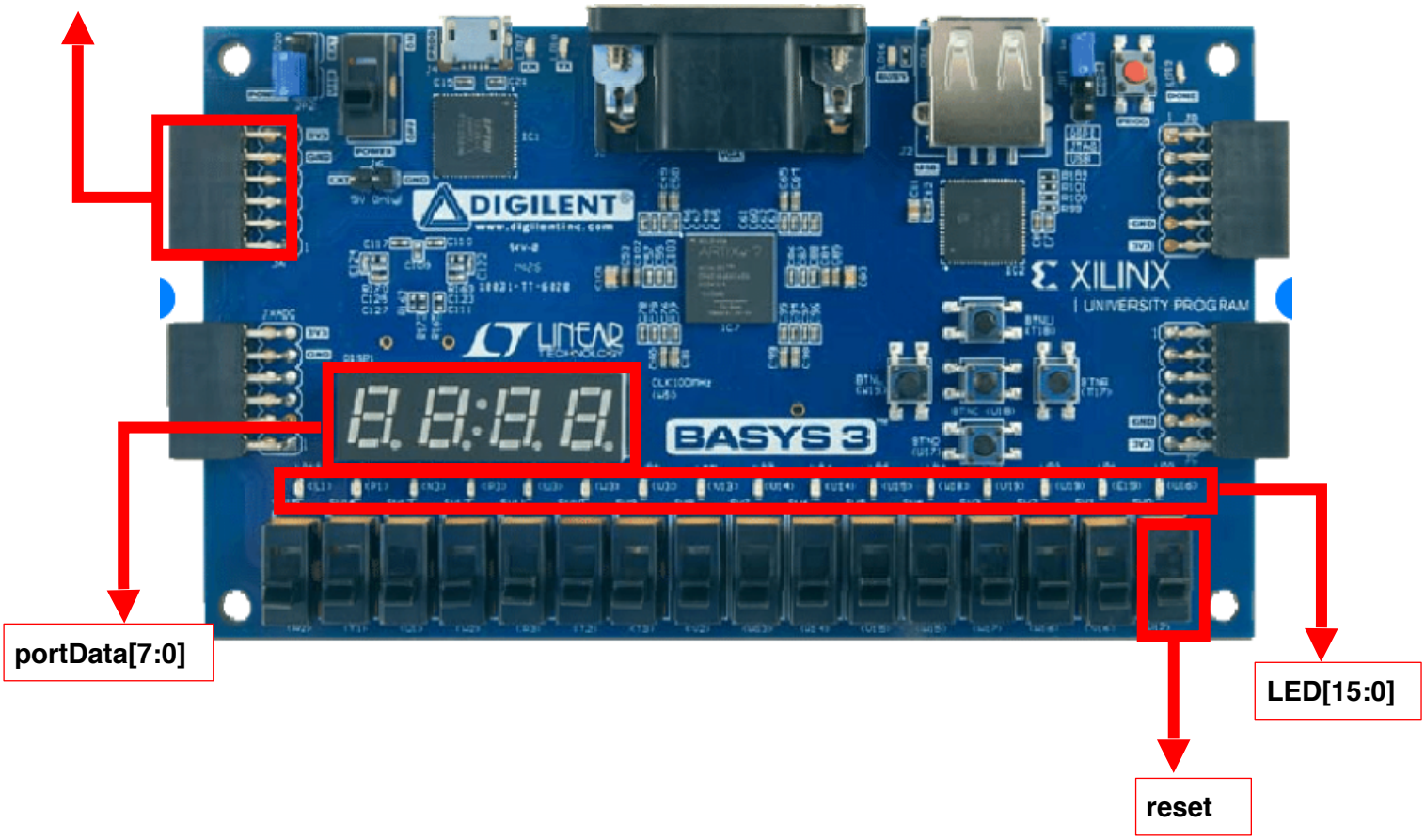
Test the implementation

To test our implantation, we used a Basys 3 FPGA as a slave device and an Arduino UNO as a master device and the user will enter the packet by the keypad then the Arduino will send the packet to the FPGA.

The connection between Arduino and FPGA:

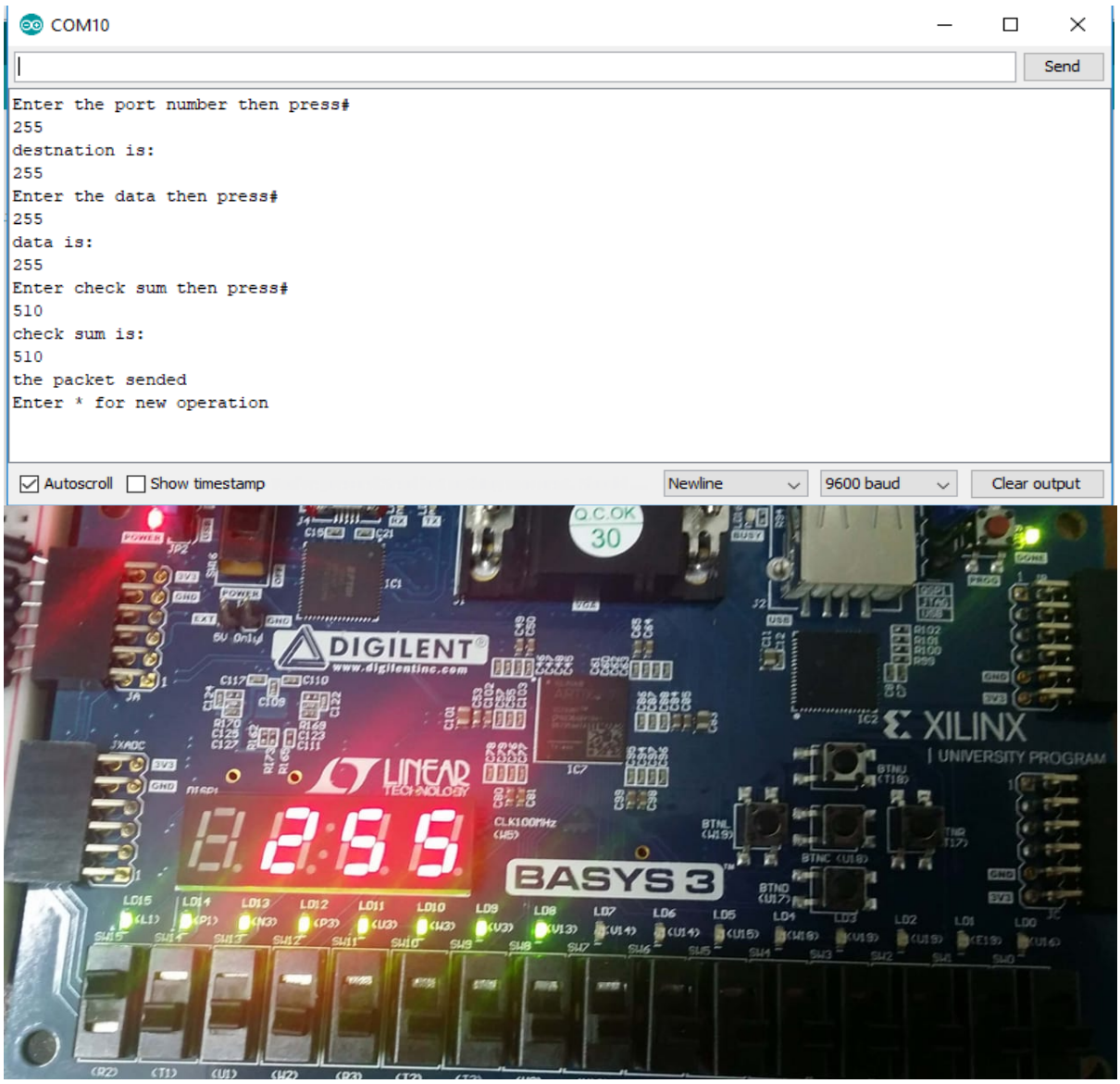


Connected with Arduino

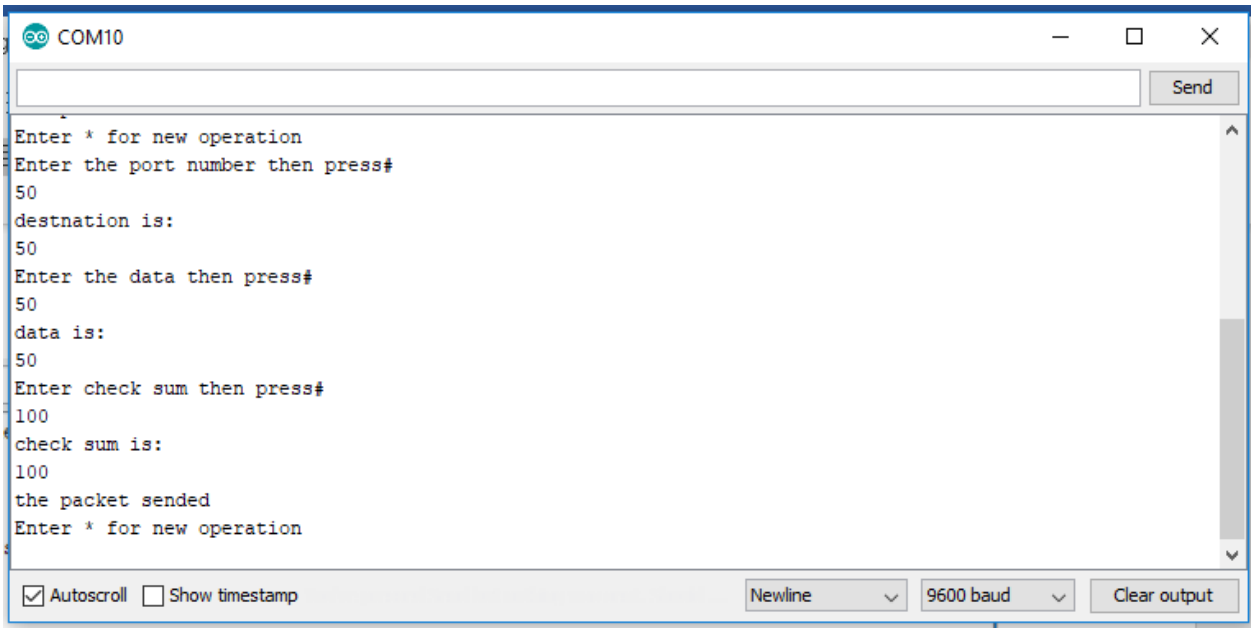


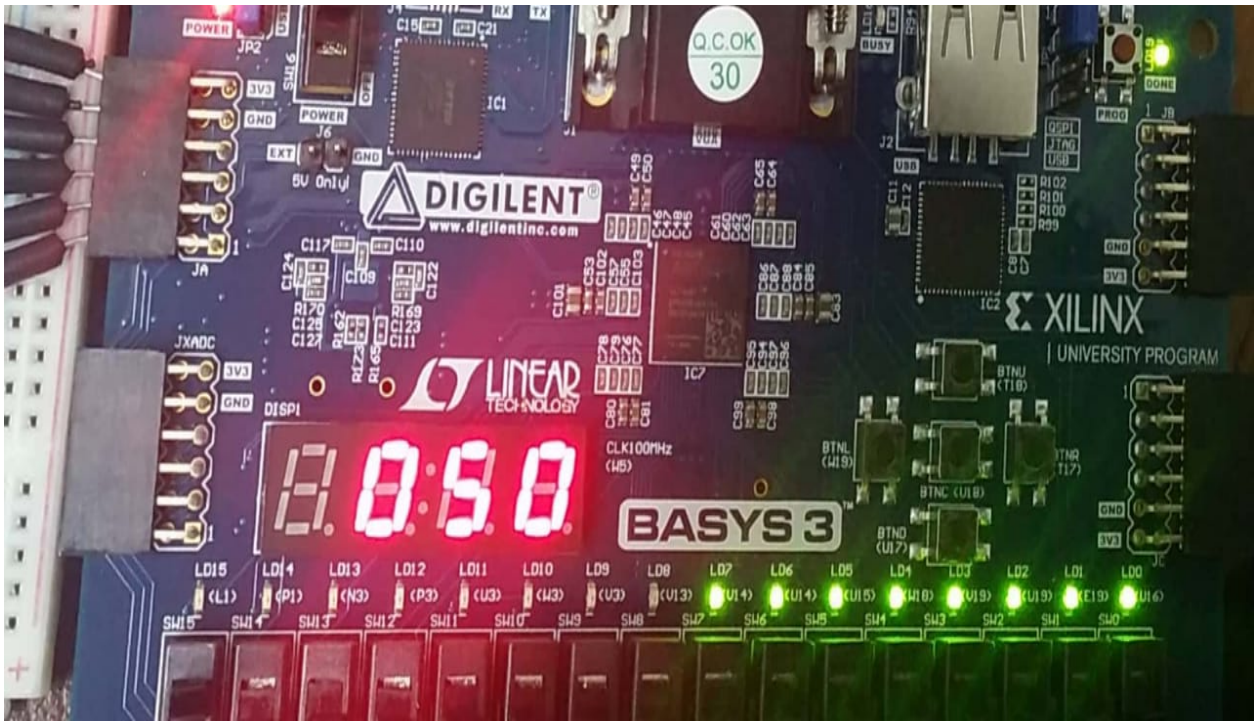
The test result

- First test

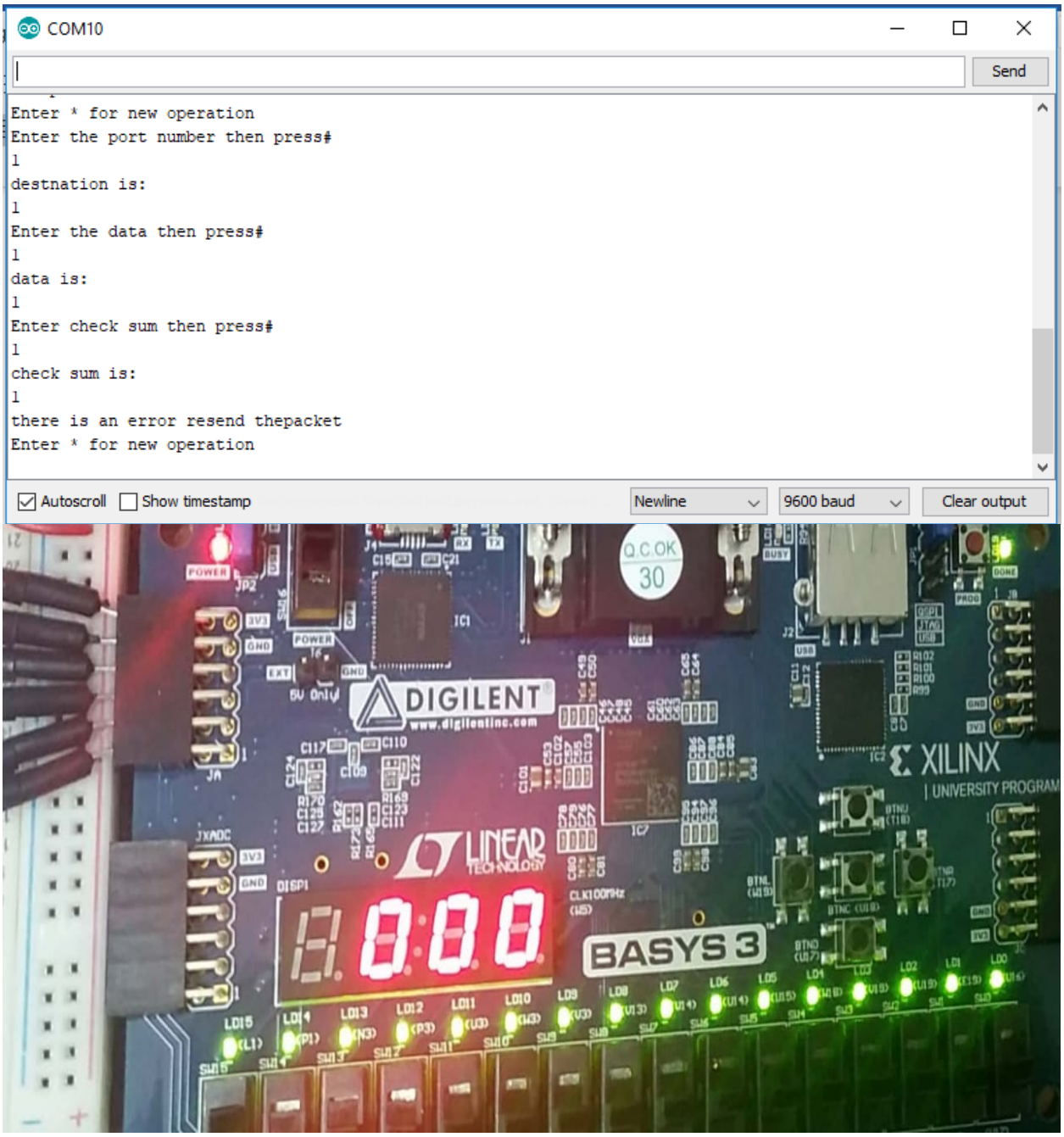


- Second test:


COM10
Send
Enter * for new operation
Enter the port number then press#
50
destnation is:
50
Enter the data then press#
50
data is:
50
Enter check sum then press#
100
check sum is:
100
the packet sended
Enter * for new operation
Autoscroll
Show timestamp
Newline
9600 baud
Clear output


DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
Q.C.OK
30

- Third test:


COM10
Send
Enter * for new operation
Enter the port number then press#
1
destnation is:
1
Enter the data then press#
1
data is:
1
Enter check sum then press#
1
check sum is:
1
there is an error resend thepacket
Enter * for new operation
Autoscroll
Show timestamp
Newline
9600 baud
Clear output
DIGILENT
XILINX
BASYS 3

The final project