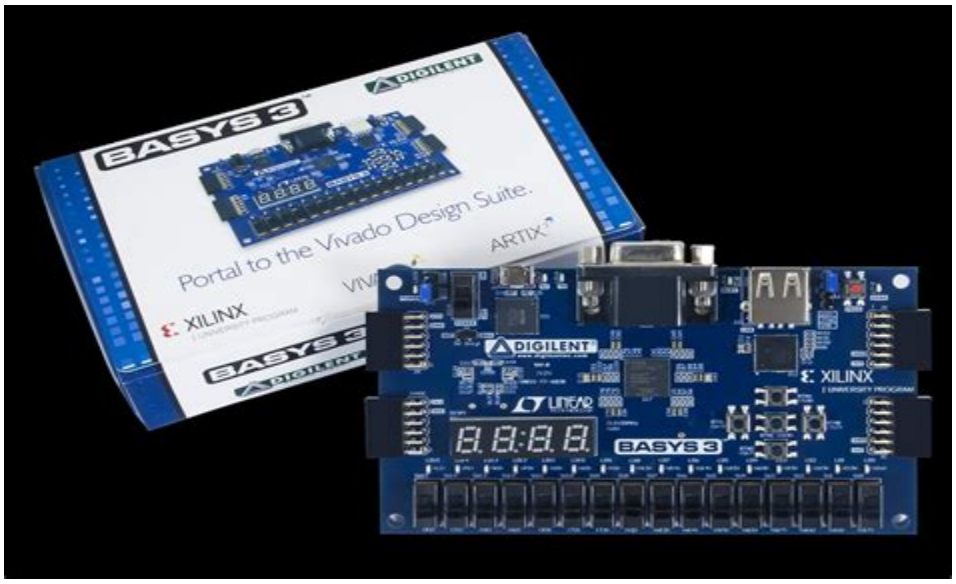
A Series of Experiments on Basys 3 FPGA

Sanath N U, Dheraj S K, Shashank Bhat, Nagarjun Kulkarni



Introduction

The Basys 3 board is a beginner friendly FPGA (Field Programmable Gate Array) which can be used to learn the concepts of Embedded Systems and Controllers. This is a documentation of our experimentation and learning process. Currently we were able to execute five programs, which are basic beginners programs. We have written the code in verilog and used Vivado software to synthesize and program the FPGA. We will discuss each program in more detail. The programs we executed were:

4 Bit Arithmetic Logic Unit, J K Flip Flop, Button Counter, UART Serial Communication, Pseudo Random Number Generation using LFSR

The codes and projects for all these programs are available online via this link:

https://github.com/sanathNU/Verilog-Projects.git

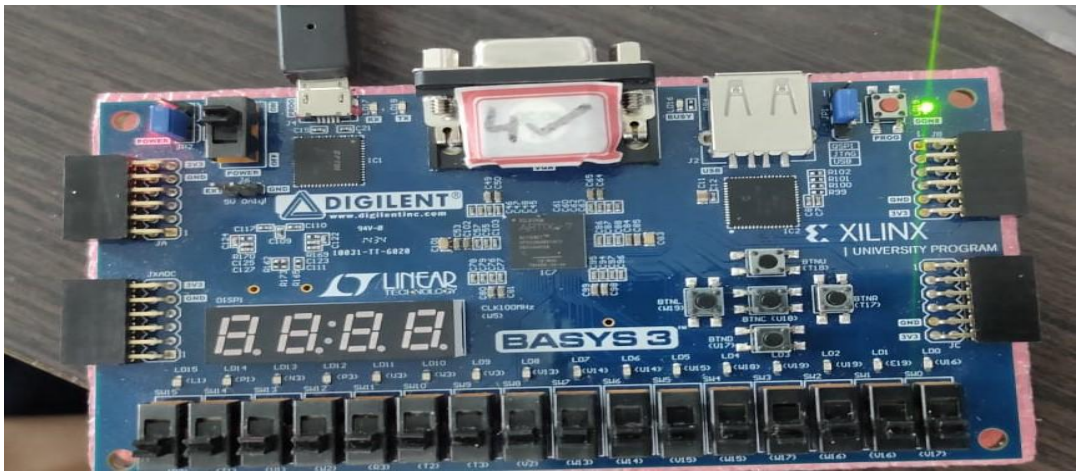
Please note that this github repo is still a work in progress.

Procedure

The procedure for programming in FPGA is fairly simple and there are various excellent guides online that are extremely good and informative. Here, we would like to restrict our discussion to a very crude overview of the process. All other websites are linked in the resources section.

We use Vivado Software (version ranging from 2014 to 2018) [1] to create the project and all the required modules. We use the Basys 3 FPGA board [2] for testing out the code. The code is written in Verilog Hardware Language.

1. Initially, we should create a project in Verilog.We select in the initial stage that our device is a Artix 7 Basys 3 FPGA board. We then write the required code in different modules.
2. A testbench is written for the code and a simulation is done to test our initial results with the code.
3. We then complete the synthesis and implementation of the code which is an option available in verilog. Finally, if there are no errors in the above mentioned steps, we then proceed to generate the bitstream.
4. After the generation of bitstream, we then connected the FPGA board to the computer via a micro-USB to USB connector.
5. The hardware manager is opened in the vivado software, where our FPGA is detected by the computer. The device is then programmed using the bitstream file generated in the earlier steps.
6. After the programming of the FPGA is done, we work on the FPGA device, verifying the correct working of our code. If there is any inconsistency or a problem with the working, we go back to step 1, change the functionality of the code, and repeat the process again. If the device works, the objective is achieved.



An FPGA board after successful bit generation and being programmed

Projects

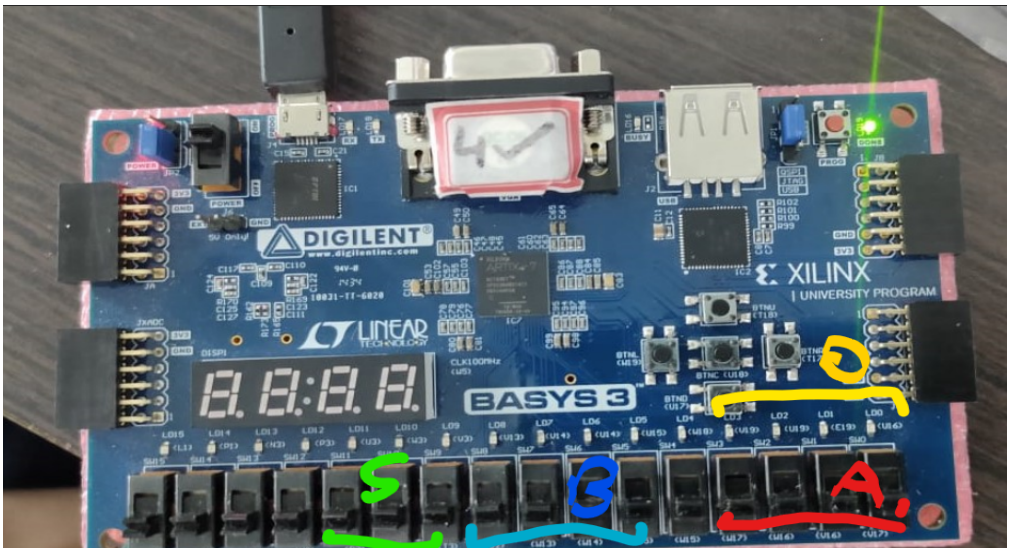
4 Bit Arithmetic Logic Unit

This is a very simple program that tries to emulate a 4 bit ALU on an FPGA board. This was done to learn how to program an FPGA. It was like the 'namaste world' of FPGA designing (yes I know 'namaste' xD). The ALU itself is a simple combinational circuit that takes in 2 4-bit inputs A and B. Then it does operations on it depending on the 3-bit Select Lines.

```
module FourBalu(
    input [3:0] A,
    input [3:0] B,
    input [2:0] Sel,
    input clk,
    output [4:0] Out
    );
    reg [4:0] Out;

    initial Out=5'b0;
    always @(*)
    case(Sel)

    3'b000 :  Out = A;
    3'b001 :  Out = A+B;
    3'b010 :  Out = A-B;
    3'b011 :  Out = A/B;
    3'b100 :  Out = A%B;
    3'b101 :  Out = A<<1;
    3'b110 :  Out = A>>1;
    3'b111 :  Out =(A>B);
    endcase

endmodule
```

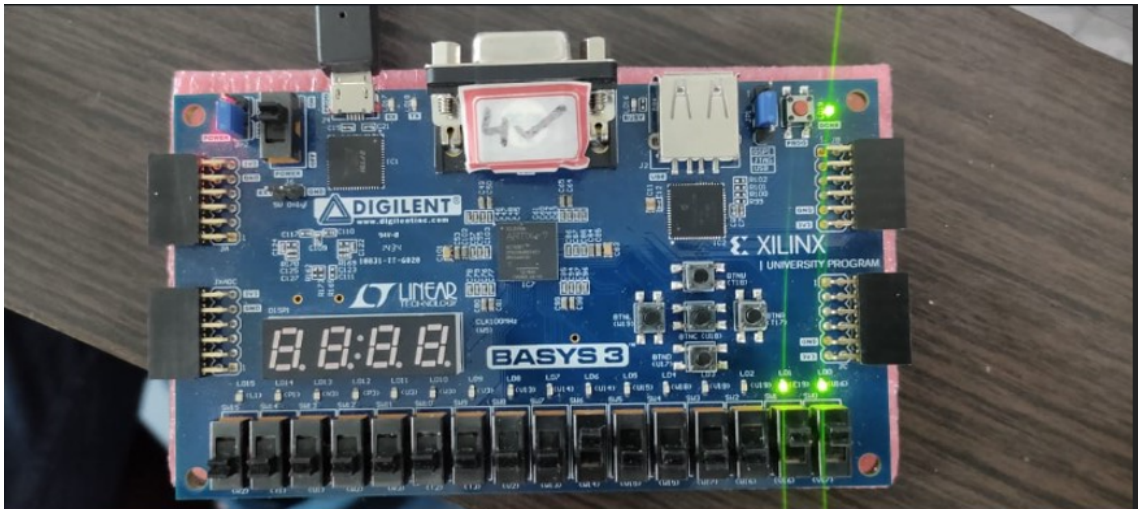
Verilog code for 4 Bit ALU

The 4 bit inputs A and B are mapped to switches in the constraint file of the FPGA board for setting the input manually as shown in the picture. The output is mapped to 4 LEDs just above the switches. The select lines are also mapped to 3 switches.

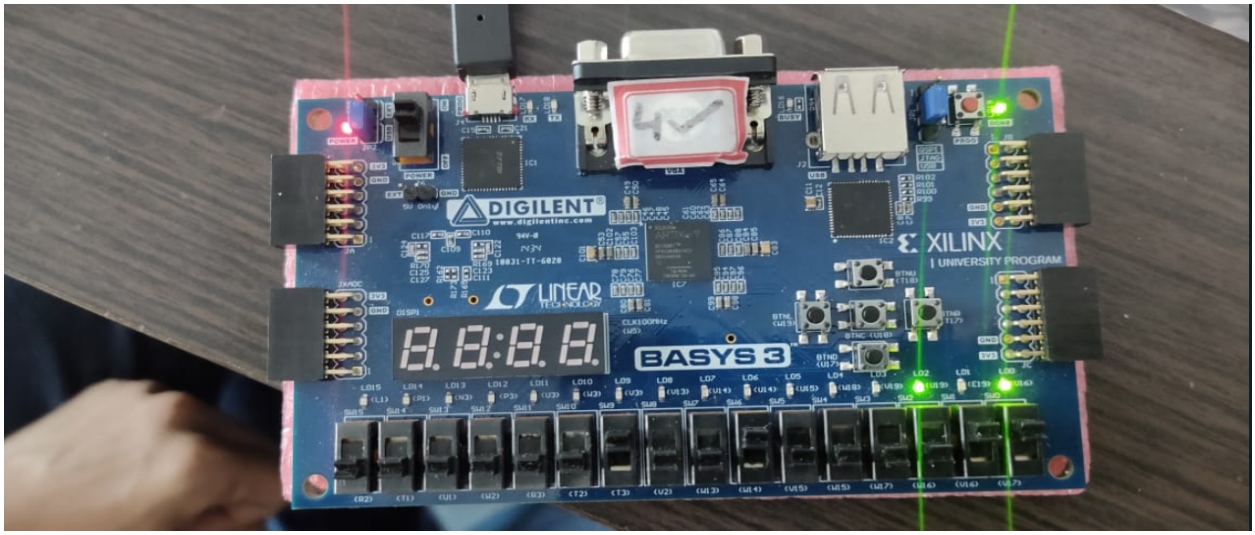


Working

Here, the input A is chosen as 0011, B is chosen as 0000 and Select is chosen as 000. As we can see from the code, the output Out = A is chosen in LEDs which are displaying 0011.



In another trial, we set A = 0011 (binary 3) and B = 0010 (binary 2) and chose Sel=001. We should get output as binary 5 or 0101 as displayed by the LEDs.



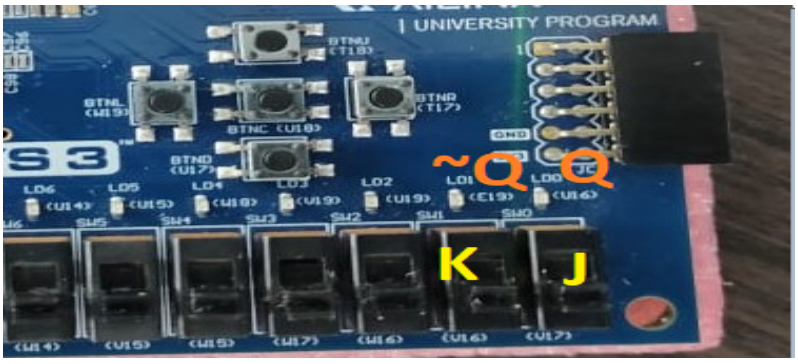
By implementing this, we were successfully able to program the FPGA to simulate a combinational circuit.

The Github link can be found in the Github repository mentioned above.

J K Flip Flop

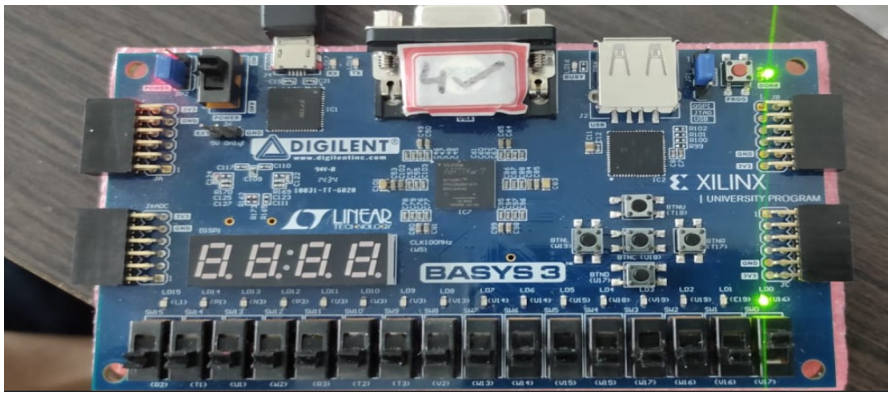
This is another simple verilog script that emulates the working of a simple JK flip flop [3]. We decided to do this circuit as an introduction to work on sequential circuits on the FPGA board. The code is readable and self understandable.

Switch V17 is set as J. Switch V16 is set as K. LED U16 is set as Q. LED E19 is set as Qbar.

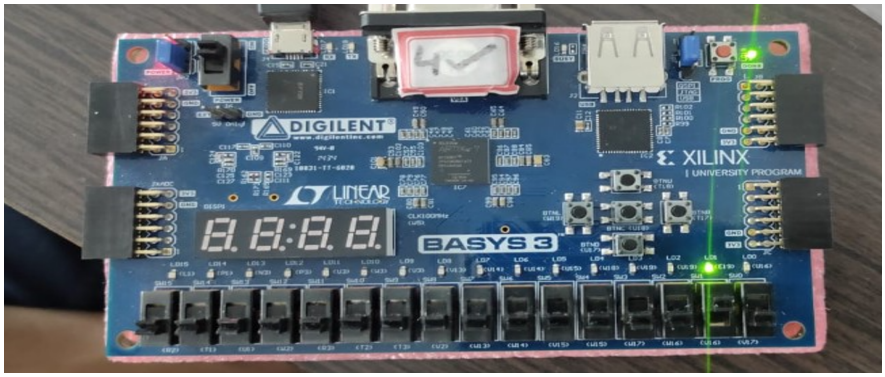


Working

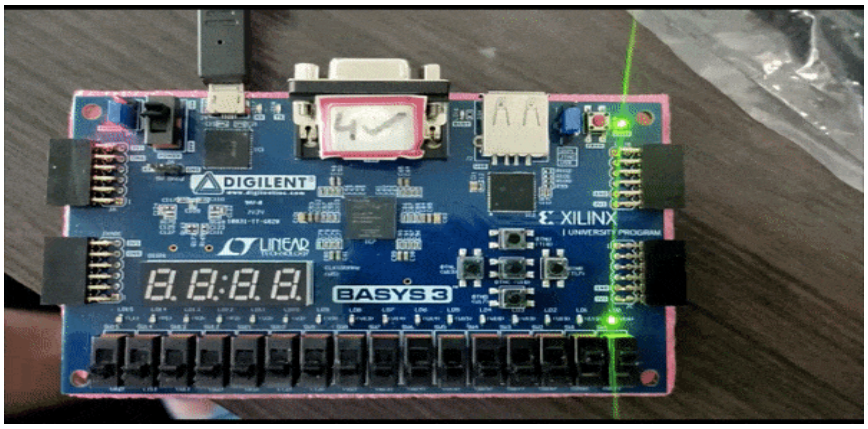
The work is fairly simple. If we turn the J switch, the output Q LED lights up. If we turn the K switch. The output Q~ turns on. If we turn both J and K switches on, it will go into toggle condition and keep on toggling.



Q LED is on for J=1,K=0



QBar LED is on for K=1,J=0



The toggling of the Flip Flop

Hence we were successfully able to simulate a sequential circuit in the FPGA board.

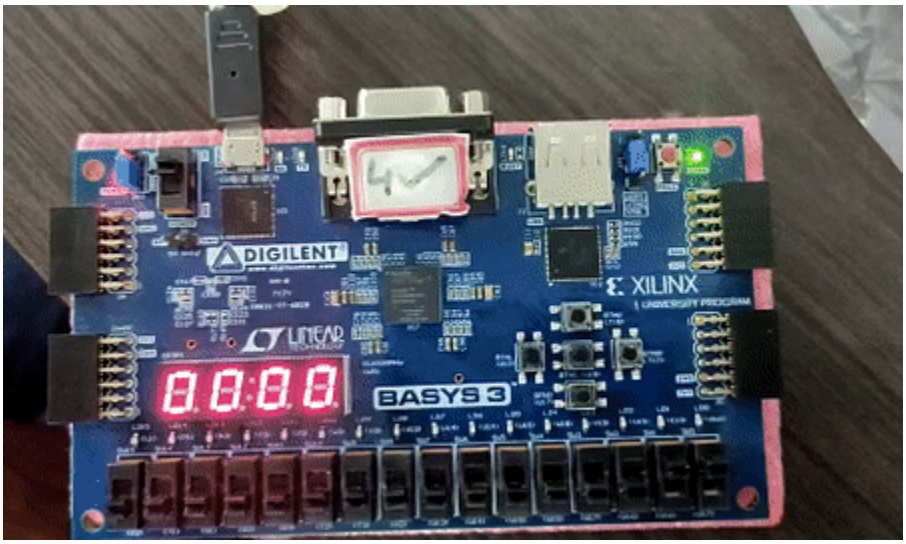
Button Counter

We have implemented a simple button Counter program that starts counting from 0000 till 9999 whenever a specific button on the FPGA is pressed. This project was done to test the 7 segment display which was quite tricky to program in the Basys 3 FPGA board. This is because of common datalines for each of 7 segment displays. We also output the number on the LED's present on the board.

Working

The number of button pushes of the btn0 of Basys 3 is counted and stored in a 16 bit register. Our task is to display this number using the 4 7-segment displays present in Basys 3 FPGA board. To display this number, a separate counter is used to loop through each of the 4 LED displays. This is implemented using the clock divider logic mentioned in the code. The correct explanation of the code is given in the references. [4]

Initially, the displays show the number 0000.



Everytime a button is pressed it starts counting upwards. A reset button is present which resets the counter back to 0.

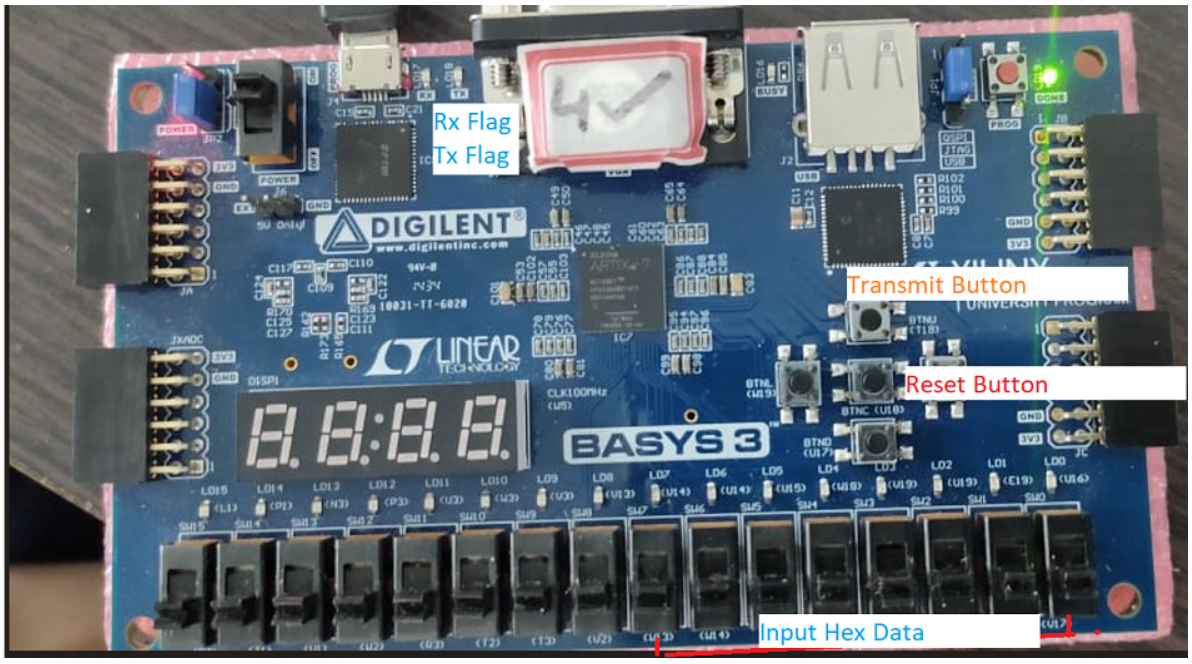
Hence, we were able to successfully implement a button counter on FPGA

UART Communication

This is a working implementation of UART communication [5][7] in Basys 3 FPGA. We can transmit 1 byte from the virtual terminal of the PC which is received by the UART and the Hex value of ASCII is displayed on the LED lights. Also, by setting the switches to the hex value of ASCII of the character we want to send and pushing the transmit button, we are able to transmit that data to the virtual terminal of the PC. The virtual terminal app we have used is open source software Tera Term [6]. The Baud Rate was set to 9600 bps.

We have written 3 main modules in verilog :- Transmitter, Receiver and Debouncer (for debouncing the button). We have used switches V17 to W1 3 as 8 input hex input data whose ASCII is transmitted via UART.

After setting the switches, the transmit button is pressed which activates the transmit module which is responsible for transmitting the data to the PC. There is also a reset button which prevents both transmission and reception of data.

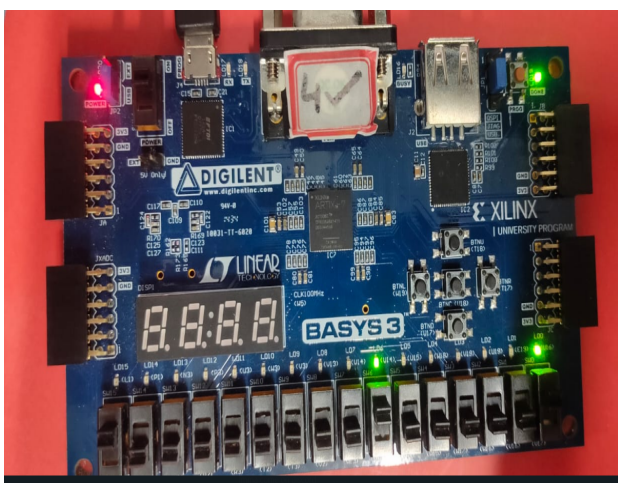


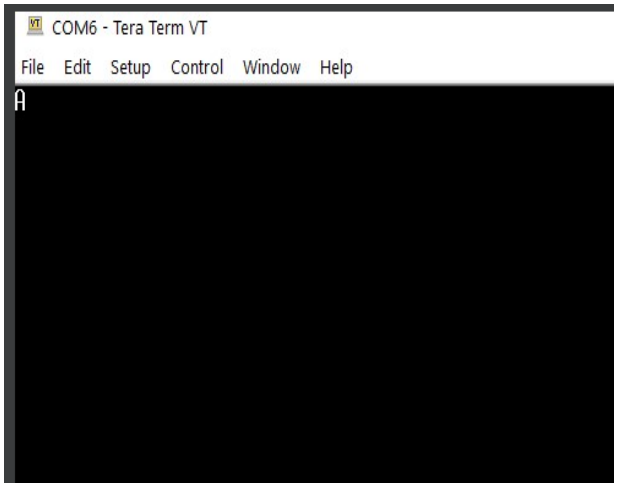
Working

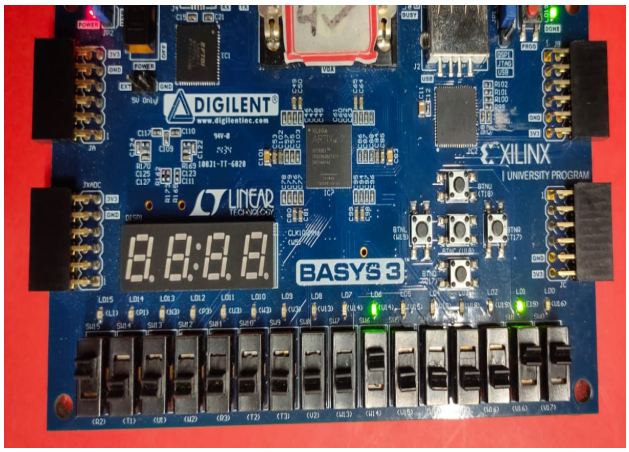
We have implemented a Finite State Machine to realize the circuit of UART. The FSM has 2 states in each module: IDLE and Transmit/Receive respectively. Our code was inspired by these references. [8][9][10]

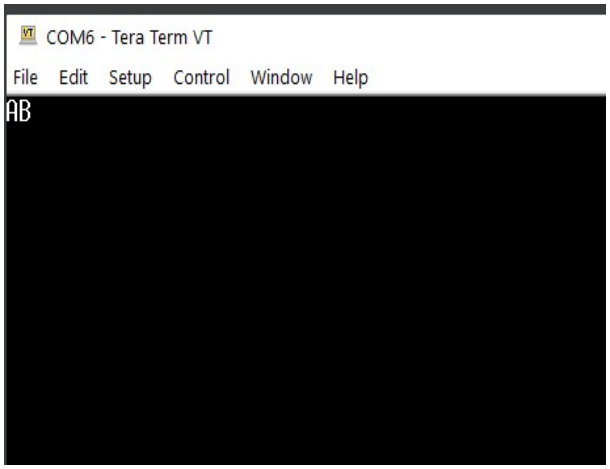
The transmitter and receiver module work in parallel.

For the demonstration of the code, we transmit the characters ‘A’ and ‘B’ in succession to the FPGA which is output on the LEDs.









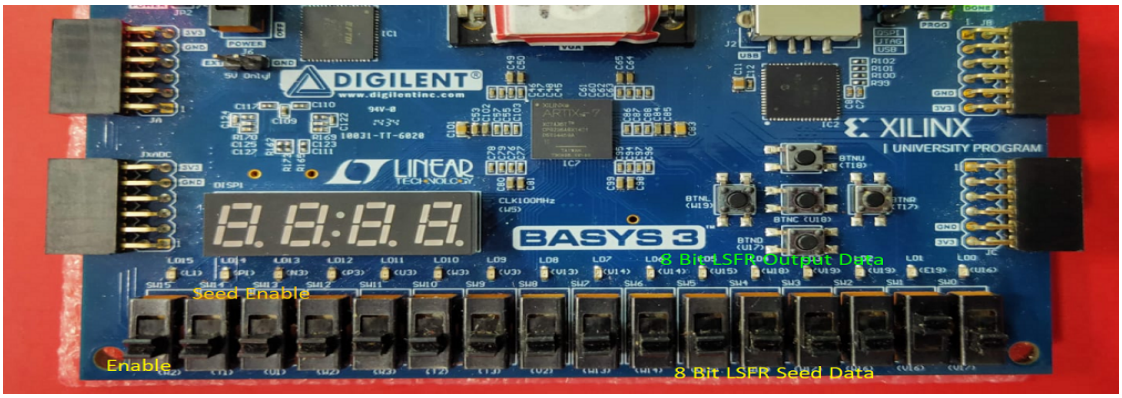
Then we transmitted the letter ‘C’ by setting the hex values on the switches shown in the above pictures. We can see the received data on the virtual terminal.

We have successfully demonstrated the working of UART on a Basys 3 FPGA board.

Linear Feedback Shift Register

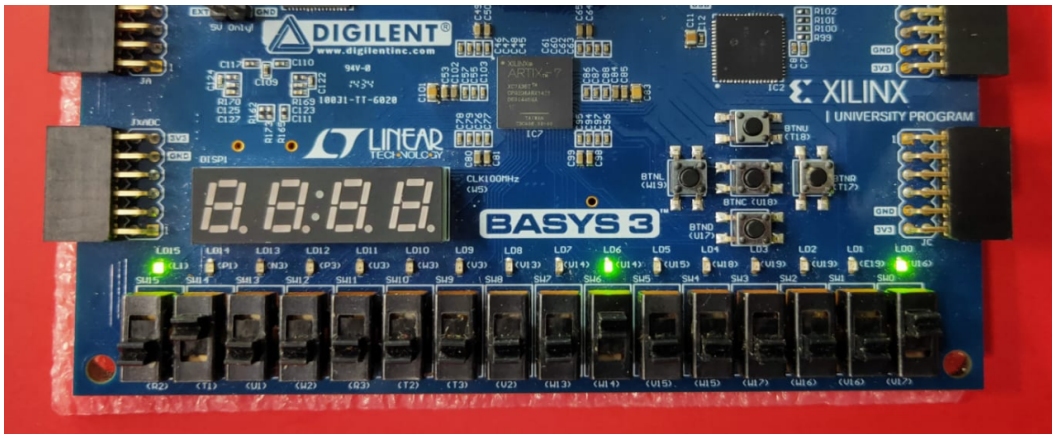
A Linear Feedback Shift Register is a device that can be used to generate Pseudo Random Numbers [11]. In this project we have implemented a LFSR which generates 1 byte random numbers that is then transmitted to the virtual terminal via UART. Code from the transmitter module of previous project UART is used to transmit the random numbers generated.

We have used an 'Enable' and a 'Seed Enable' inputs to the LFSR in which the former controls the LFSR and the latter controls the Seed Input through the switches.

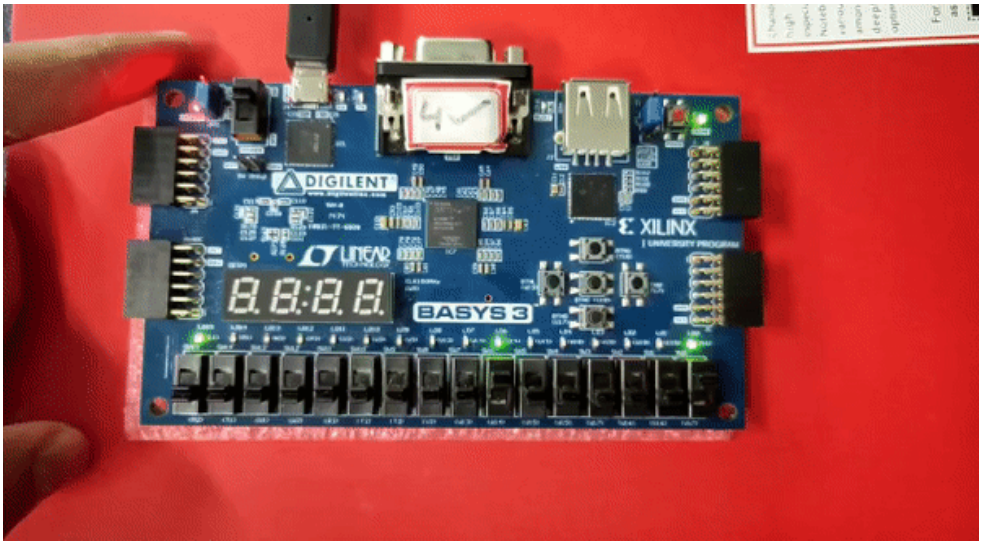


Working

For the working of the LFSR, we can either set the seed data through external switches . By default, the seed value is set to 0. Initially, the seed_enable button is set high. Then, we can see the Switch Data reflected on the LEDs.



After setting seed_enable, the initial seed for the random number is set. Then we should set the seed_enable low and enable the 'Enable' switch. This starts the random number generation.



Hence an LFSR was successfully implemented on Basys 3 FPGA board.

Resources

[1] https://digilent.com/reference/_media/basys3:basys3_rm.pdf

[2]https://www.xilinx.com/developer/products/vivado.html

[3] https://www.tutorialspoint.com/what-is-j-k-flip-flop

[4] http://www.physics.umd.edu/hep/drew/programmable/#serial

[5]https://www.circuitbasics.com/basics-uart-communication/

[6]https://ttssh2.osdn.jp/index.html.en

[7]https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter

[8]https://www.nandland.com/vhdl/modules/module-uart-serial-port-rs232.html

[9]https://www.instructables.com/UART-Communication-on-Basys-3-FPGA-Dev-Board-Power/

[10]https://www.hackster.io/alexey-sudbin/uart-interface-in-vhdl-for-basys3-board-eef170#toc-tera-term-setup-9

[11] https://en.wikipedia.org/wiki/Pseudorandomness