Raul-Mihai Acu
30432

FPGA & Arduino communication

1. Purpose

The objective of this project is to design a communication protocol between a Basys3 board and an Arduino board, in order to make use of Arduino`s capabilities of interaction with the environment and the performance of the FPGA board.

2. Specification

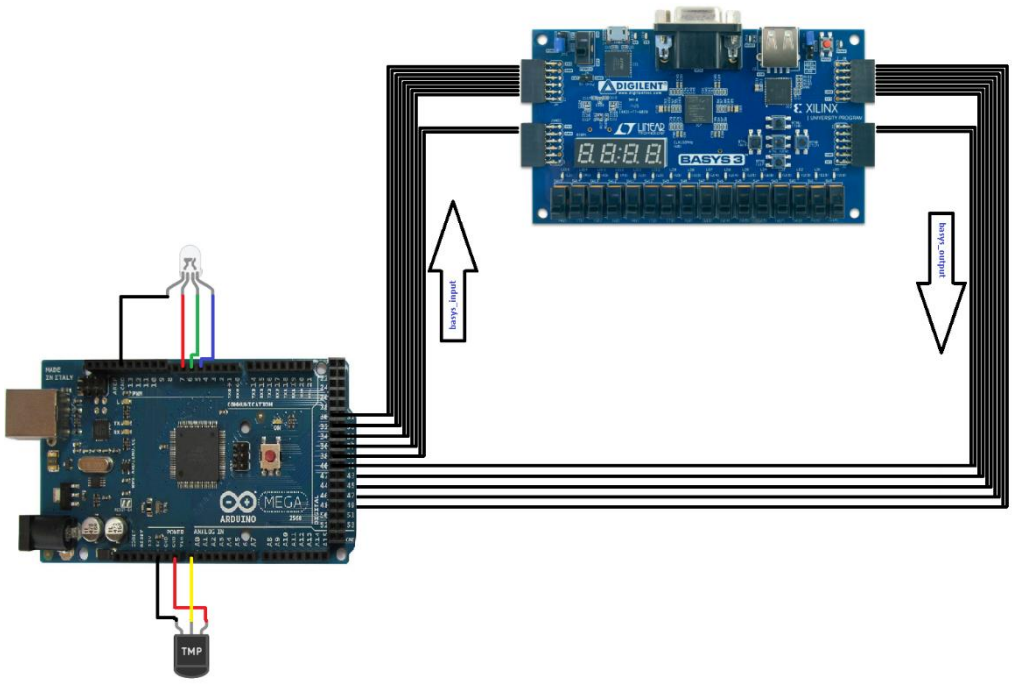
Required materials

- 4x breadboard jumper cables for the RGB-Led
- 3x breadboard jumper cables for the analog temperature sensor
- 20x breadboard jumper cables for communication
- 1x breadboard
- 1x Arduino Mega2560 board
- 1x Basys3 FPGA board

Flow description

The temperature sensor sends an analog value to the Arduino board, where it is converted to a binary array, ready to be sent to the FPGA board.
The transmission of data is obtained by using the communication cables that connect the Arduino and Basys3 board.
The internal configuration of the FPGA board allows us to convert the analog value received to a readable temperature, represented as Celsius, Kelvin or Fahrenheit degrees. The value is displayed on the seven-segment display and then sent back to the Arduino, lighting the led depending on the value as: cold (blue), medium (green), hot (red).

3. Design



4. Implementation

Arduino

At first, we will set up the pinMode for each pin of the Arduino board.

```
pinMode(basys_inputX, OUTPUT); - digital pins 30-39
pinMode(basys_outputX, INPUT); - digital pins 40-49
pinMode(ledRed, OUTPUT); - digital pin 7
pinMode(ledGreen, OUTPUT); - digital pin 6
pinMode(ledBlue, OUTPUT); - digital pin 5
pinMode(analogTemp, INPUT); - analog pin A0
```

The communication process begins after reading the analog temperature value and converting it into a binary array.

```
long analogTemperature = analogRead(analogTemperaturePin);
long i = 9;
while(analogTemperature != 0){
   binaryArray[i--] = analogTemperature%2;
   analogTemperature /= 2;
 }
sendDataToBasys(){
        digitalWrite(basys_input0 , binaryArray[0]);
        digitalWrite(basys_input1 , binaryArray[1]);
        digitalWrite(basys_input2 , binaryArray[2]);
        digitalWrite(basys_input3 , binaryArray[3]);
        digitalWrite(basys_input4 , binaryArray[4]);
        digitalWrite(basys_input5 , binaryArray[5]);
        digitalWrite(basys_input6 , binaryArray[6]);
        digitalWrite(basys_input7 , binaryArray[7]);
        digitalWrite(basys_input8 , binaryArray[8]);
        digitalWrite(basys_input9 , binaryArray[9]);
}
```

After sending the data to the FPGA board, it is computed a digital value for the temperature, and read back into the Arduino, then the RGB led is lit depending on the converted binary received array.

```
readDataFromBasys(){
            recievedArray[0] = digitalRead(basys_output0);
            recievedArray[1] = digitalRead(basys_output1);
            recievedArray[2] = digitalRead(basys_output2);
            recievedArray[3] = digitalRead(basys_output3);
            recievedArray[4] = digitalRead(basys_output4);
            recievedArray[5] = digitalRead(basys_output 5);
            recievedArray[6] = digitalRead(basys_output 6);
            recievedArray[7] = digitalRead basys_output 7);
            recievedArray[8] = digitalRead(basys_output 8);
            recievedArray[9] = digitalRead(basys_output 9);
        }
binaryArrayToDecimal(int binary[10]){
  recievedDecimal = 0;
  for(int i = 0; i < 10; i++)
    recievedDecimal = recievedDecimal*2 + binary[i];
};
```

```
color(int red, int green, int blue){
        analogWrite(redPin, red);
        analogWrite(greenPin, green);
        analogWrite(bluePin, blue);
}
lightLedDependingOnTemperature(int temperature){
        if(temperature < 28){
         color(12,120,165);
        }
        else if(temperature >= 28 && temperature <= 30){

          color(255,88,0);
        }
        else if(temperature > 30){
          color(255,0 ,10);
        }
}
```

Basys3

Modules:

Temperature_Control – main module
Seven segment display – module for displaying the digital value

Temperature_Control

- input:
  - clock: std_logic;
  - reset: std_logic;
  - basys_input: std_logic_vector(9 downto 0);
- output:
  - basys_output: std_logic_vector(9 downto 0);
  - anode_activate: std_logic_vector(3 downto 0);
  - cathode_activate: std_logic_vector(6 downto 0);

Seven segment display

- input:
  - clock: std_logic;
  - reset: std_logic;
  - display_number: std_logic_vector(9 downto 0);
- output:
  - anode_activate: std_logic_vector(3 downto 0);
  - cathode_activate: std_logic_vector(6 downto 0);

After defining the signals required in the architecture of the top-module, a process of converting the received array into a decimal value to be sent to the Arduino is described as it follows:

```
Display: seven_segment_display port map(clock_100Mhz, value_to_display , reset, Anode_Activate, LED_out);
    process(Recieved_value)
      begin
       auxIntRecieved <= to_integer(unsigned("000000" & Recieved_Value));
       auxIntMilivolts <= auxIntRecieved * (5000 / 1024);
       auxResult <= (auxIntRecieved - 500) / 10;
       value_to_display <= std_logic_vector(to_unsigned(auxResult, 16));
        Transmitted_Value <= value_to_display(9 downto 0);
    end process;
```