VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Dinh Bao Tuan Phan

CONTROL STEPPER MOTOR BY USING FPGA/VHDL

Technology and Communication
2022

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

ABSTRACT

Author	Dinh Bao Tuan Phan
Title	Control stepper motor by using FPGA/VHDL
Year	2022
Language	English
Pages	31
Name of Supervisor	Santiago Chavez

This thesis aims to develop a system that can control a stepper motor using a computer keyboard and allow the user to rotate the stepper motor with expected rotations. Furthermore, this thesis can be used as material for teaching and learning FPGA/ VHDL.

This project uses Altera DE2 Development and Education board, Stepper motor, RS-232 UART, and VHDL.

The RS-232 UART is a connection between the computer and the DE2 board, allowing communication between two modules. Altera DE2 board was used to store data received from the RS-232 UART, output the data and control the stepper motor. VHDL is the programming language used to develop this project.

Keywords FPGA, VHDL, Altera DE2 board, RS-232, and Stepper Motor

CONTENTS

LIST OF FIGURES AND TABLES

LIST OF CODE SNIPPETS

1 INTRODUCTION

In contemporary society, the stepper motor is found widely in daily lives from family appliances to commercial and industrial applications. By taking advantage of the accurate positioning of the step motor, it plays a vital role in the development of the industry and making the human lifestyle become more comfortable.

The field-programmable gate array (FPGA) allows users to have the ability to custom its functionality after manufacturing. The other advantages of FPGA are low latency and connectivity which do not require a host computer.

The concept of this project is to implement the VHDL program language into the FPGA board to control the stepper motor rotates with an expected rotation.

2 USED TECHNOLOGIES

2.1 Altera DE2 Development and Education Board

The Altera DE2 board is a product of Terasic company that is used for providing the ideal vehicle for advanced design prototyping in the multimedia, storage, and networking.

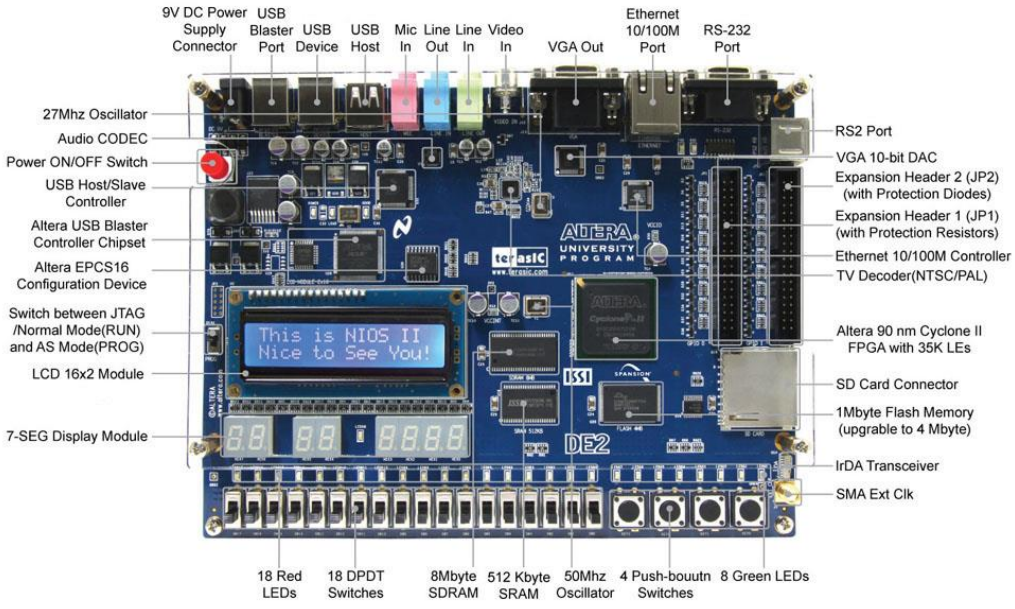

Figure 1. Altera DE2 Board /1/

Feature of Altera DE2 board: /1/

- Altera Cyclone II 2C35 FPGA with 35000 LEs
- Altera Serial Configuration devices (EPCS16) for Cyclone II 2C35
- USB Blaster built-in onboard for programming and user API controlling
- JTAG Mode and AS Mode are supported
- 8Mbyte (1M x 4 x 16) SDRAM
- 512K byte (256K X16) SRAM
- 4Mbyte Flash Memory (upgradeable to 4Mbyte)
- SD Card Socket
- 4 Push-button switches
- 18 DPDT switches
- 9 Green User LEDs
- 18 Red User LEDs
- 16 x 2 LCD Module
- 50-MHz oscillator and 27-MHz (from TV decoder) for clock sources
- 24-bit CD-Quality Audio CODEC with line-in, line-out, and microphone-in jacks

- VGA DAC (10-bit high-speed triple DACs) with VGA out connector
- TV Decoder (NTSC/PAL) and TV in connector
- 10/100 Ethernet Controller with socket.
- USB Host/Slave Controller with USB type A and type B connectors.
- RS-232 Transceiver and 9-pin connector
- PS/2 mouse/keyboard connector
- IrDA transceiver
- Two 40-pin Expansion Headers with diode protection
- DE2 Lab CD-ROM which contains many examples with source code to exercise the boards, including SDRAM and Flash Controller, CD-Quality Music Player, VGA, and TV Labs, SD Card reader, RS-232/PS-2 Communication Labs, NIOSII, and Control Panel API

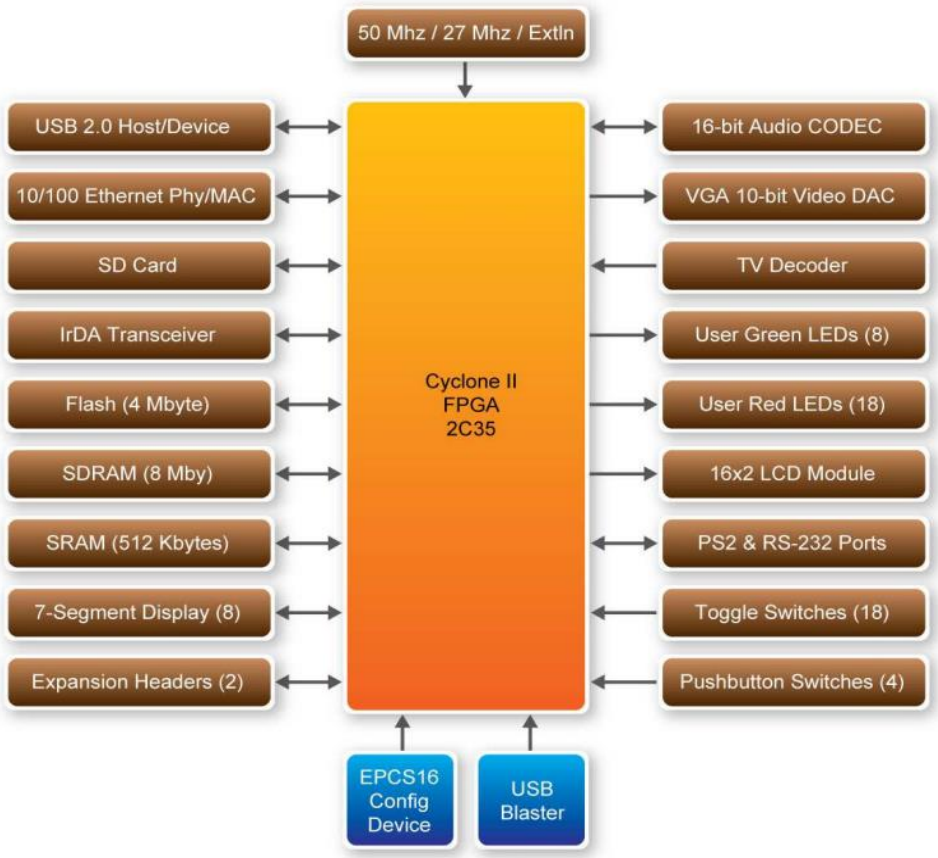

Figure 2. DE2 board block diagram

2.2 FPGA and VHDL Programming Language

A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence the term *field-programmable*. The FPGA configuration is generally specified using a hardware

description language (HDL), similar to that used for an application-specific integrated circuit (ASIC). FPGAs contain a cluster of programmable rationale pieces and pecking order of recon-figurable intercontinental permitting pieces to be wired together. Rationale squares can be designed to perform complex combinational capacities, or act as straightforward rationale entryways like AND and XOR. /2/

The VHSIC Hardware Description Language (VHDL) could be a hardware description language (HDL) that can show the behaviour and structure of advanced frameworks at different levels of reflection, extending from the framework level down to that of rationale doors, for plan section, documentation, and confirmation purposes. /3/

VHDL is generally used to write text models that describe a logic circuit. Such a model is processed by a synthesis program, only if it is part of the logic design.

2.3 RS-232 UART

The serial harbour inside the DE2 Media Computer executes a UART that's associated with an RS-232 chip on the DE2 board. This UART is orchestrated for 8-bit in-formation, one end bit, odd correspondence, and works at a baud rate from 9600 to 115,200.

2.4 Altera Quartus II and RealTerm application

Intel Quartus II is programmable logic device design software produced by Intel; earlier to Intel's procurement of Altera, the apparatus was called Altera Quartus Prime, prior to Altera Quartus II. Quartus Prime empowers the investigation and blends of Hardware description language (HDL) plans, which empowers the designer to compile their plans, perform timing examination, look at Register-transfer level (RTL) graphs, recreate a design's response to diverse boosts, and arrange the target gadget with the software engineer. Quartus Prime incorporates the usage of VHDL and Verilog for equipment depiction, visual altering of rationale circuits, and vector waveform re-enactment. /4/

RealTerm is a terminal program specially designed for capturing and sending data through various protocols (UART, Raw TCP sockets, etc). The purpose is similar

to that of TeraTerm or PuTTY or GTKTerm or Serial Monitor (Arduino), but RealTerm is way more feature-rich than any other serial console program. /5/

3 STEPPER MOTOR

3.1 Can-Stack step motor

The motor was used in this project is the Can-Stack step motor, which has a 2-coil permanent magnet. When poles are magnetized by electric pulses applied to the coils, they attract the permanent magnet rotor core in reverse polarities, thereby starting rotation /6/. The rotation of the motor depends on the number of pulses sent to the motor and the speed can be controlled by changing the frequency of the pulse signal.

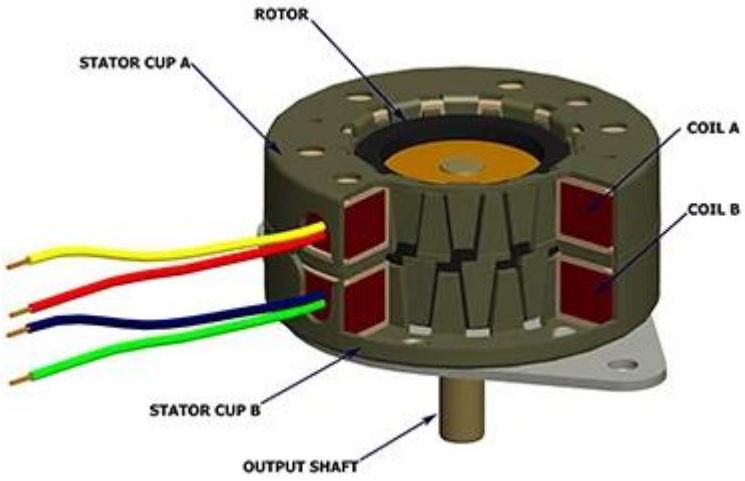

Figure 3. Can-Stack step motor

The formula for the speed of step motor and used in the code:

$$Count = (Main\ clock / Desired\ frequency) - 1$$

where

- Count : The value input into code to get the desired speed
- Main clock : Clock of DE2 board (50 MHz)
- Desired frequency : User expected motor frequency

3.2 Motor driver

Normally, the stepper motor requires a 5V input power to rotate. However, the voltage capability of the FPGA GPIO is only 3.3V, so the stepper motor cannot be driven directly by the FPGA board. To solve this problem, the driver circuit which can increase the voltage of the FPGA output from 3.3V to 5V is applied. The ULN2803A IC is used to drive the stepper motor with an individual electrical pulse for each winding of the stepper motor. /7/

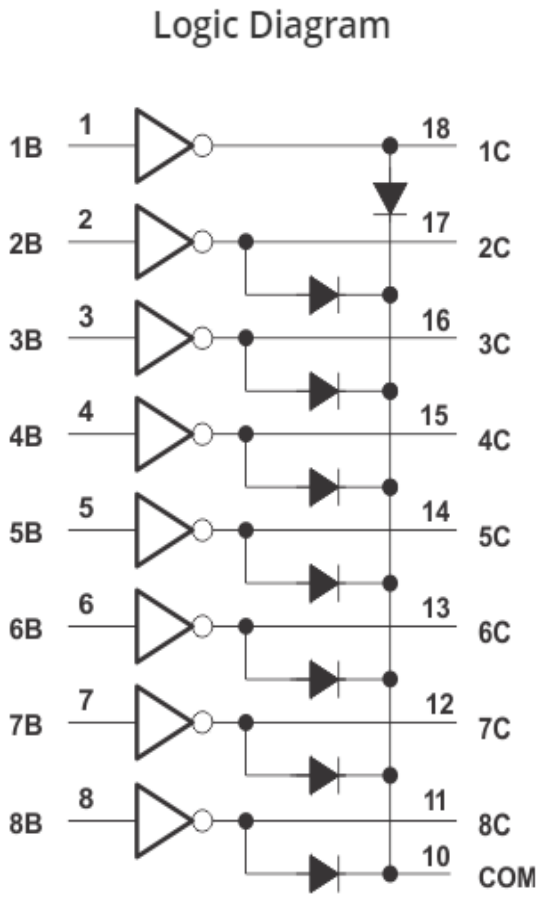

Figure 4. ULN2803A Logic diagram /8/

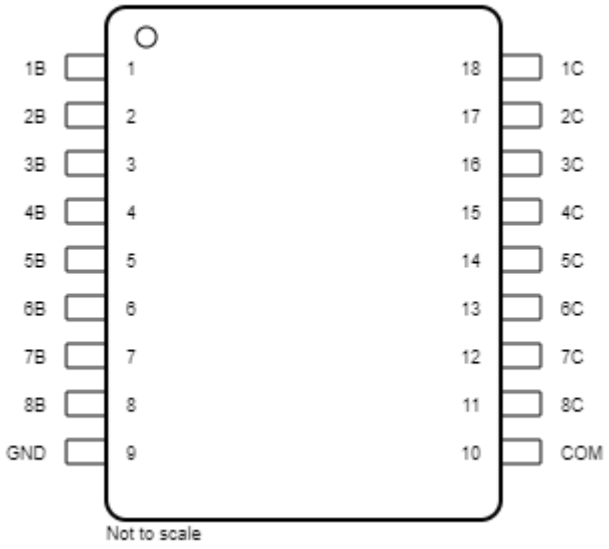

Figure 5. ULN2803A Pin configuration /8/

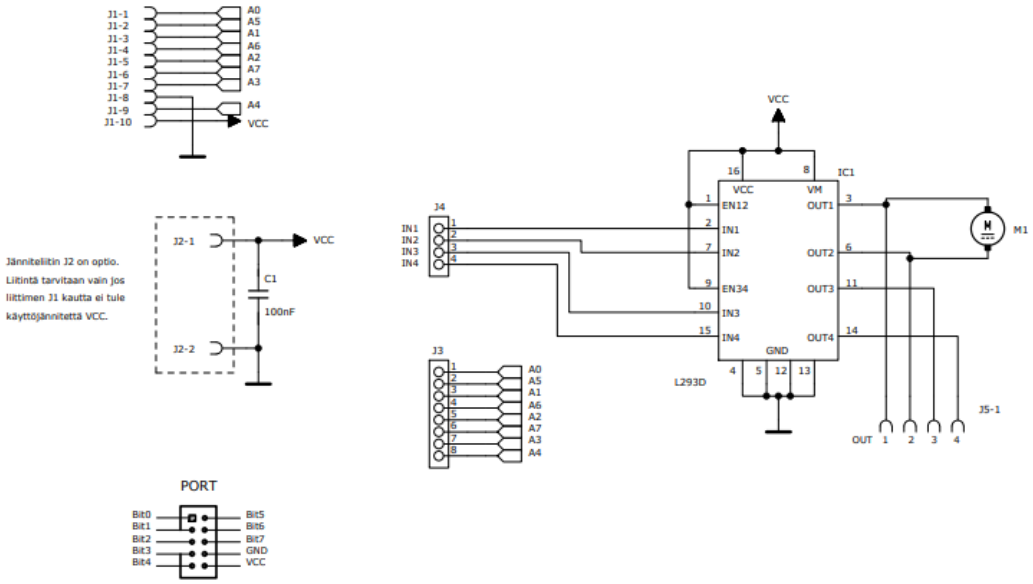

Figure 6. Motor driver schematic /9/

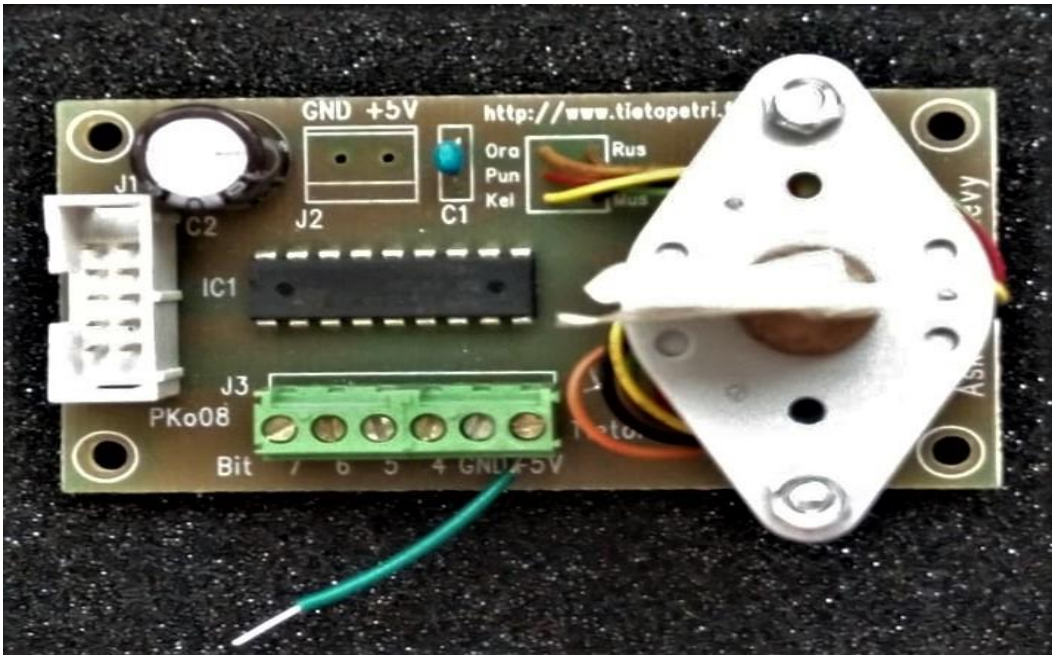

Figure 7. Stepper Motor Driver

4 SYSTEM DESIGN AND IMPLEMENTATION

This chapter will explain the data flow and each module used in the design.

4.1 System Design

As shown in Figure 8, the hardware diagram of the design is working step by step. Firstly, the keyboard from the computer plays a role as the transmitter which sends the data to RealTerm and goes through the RS-232 serial connection to the DE2 board. Next step, the DE2 board that received data will go on sending a string of bit combinations to GPIO ports connected to the motor and start driving the motor.

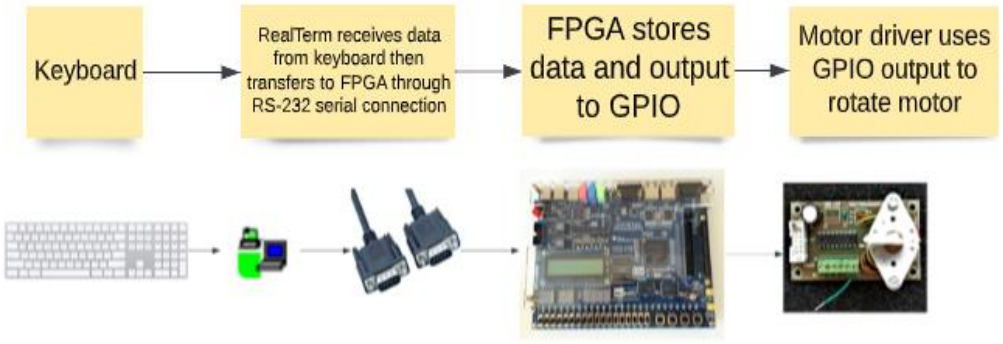

Figure 8. Hardware diagram

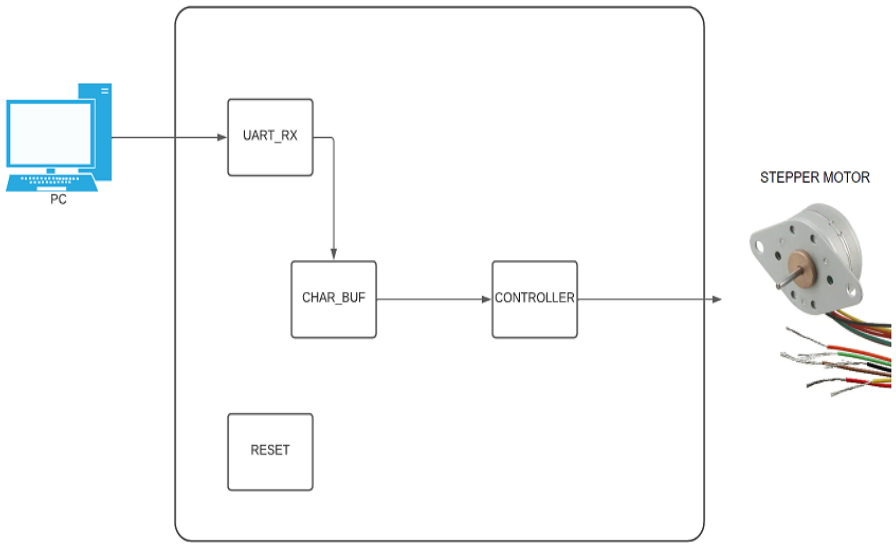

Figure 9. Dataflow software diagram

Figure 9 is shown the dataflow software diagram. From the diagram, it is easy to see that to control the step motor, the ASCII character was sent from the PC keyboard through the RealTerm terminal to the UART_RX module. The data that is transferred from RealTerm go through the RS-232 serial cable to the UART_RX module. After that, UART_RX outputted that data to the CHAR_BUF module to store it in the DE2 board and outputted to the next module. CONTROLLER module received data stored in DE2 and then sent to GPIO to rotate the motor.

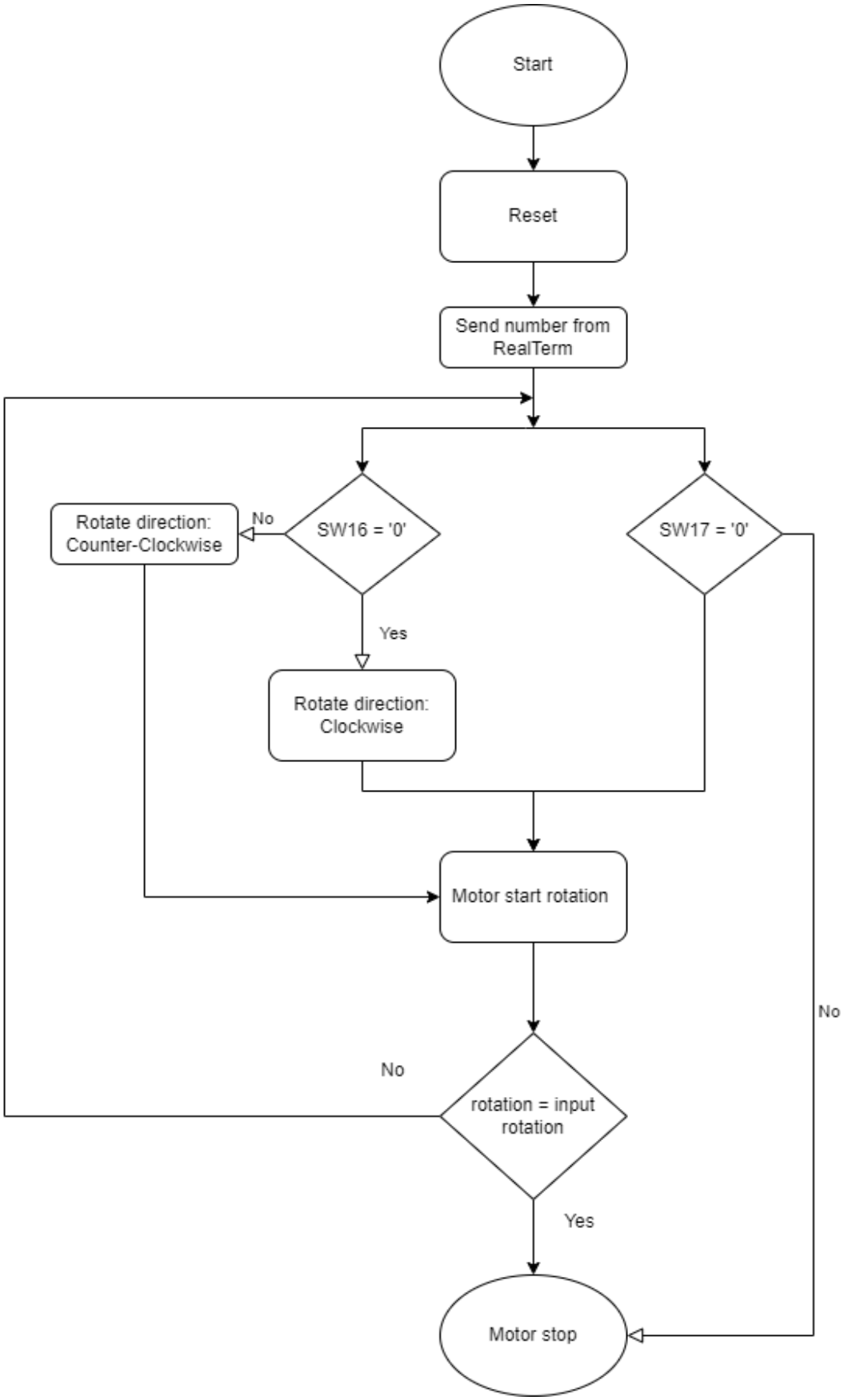

Figure 10. High-level diagram

As figure 7 shown, when the DE2 board is powered up, it first reset all the designs. RealTerm application is used to send the ASCII number to control the motor rotation. After the data is transferred to DE2 by RealTerm, the design will face to 2 conditions SW16 and SW17 that have a direct impact to the stepper motor. SW16 is used to decide which direction the motor will rotate. If the SW16 is turned off, the motor will rotate in a clockwise direction and will rotate counter-clockwise if SW16 is turned on. SW17 plays an important role in this design as a hardware interrupt. When the motor is rotating, SW17 will stop all the activities of the motor if enabled even in the input state. Next, when meeting the two conditions above, the motor starts the rotation. Every time the motor completes one round, the design will ask if the number of rotations of the motor is equal to the expected value or not, they will decide to stop the motor or continue the process.

4.2 Motor System Implementation

Project creation, design implementation, and work theory of product modules are presented in this chapter.

4.2.1 Quartus project creation

The Quartus II (version 13.0sp1) was used in the project, this is the last version of Quartus II to support the DE2 board. To start the project with this application, follow the procedure below:

1. Launch Quartus II and create a new project
2. In **New Project Wizard**, give the name for the project and choose the device family for the board (Cyclone II, EP2C35F672C6)
3. Create a new VHDL file and start programming
4. Download from the Internet file DE2_pin_assignments.csv and save it in the project folder
5. Select **Assignments → Import Assignments →** Find the folder store file DE2_pin_assignments.csv that was downloaded before and select it.
6. Compile the code and select **Tools → Programmer → Hardware Setup** and select **USB-Blaster**

7. Run the **.sof** file to program the FPGA device

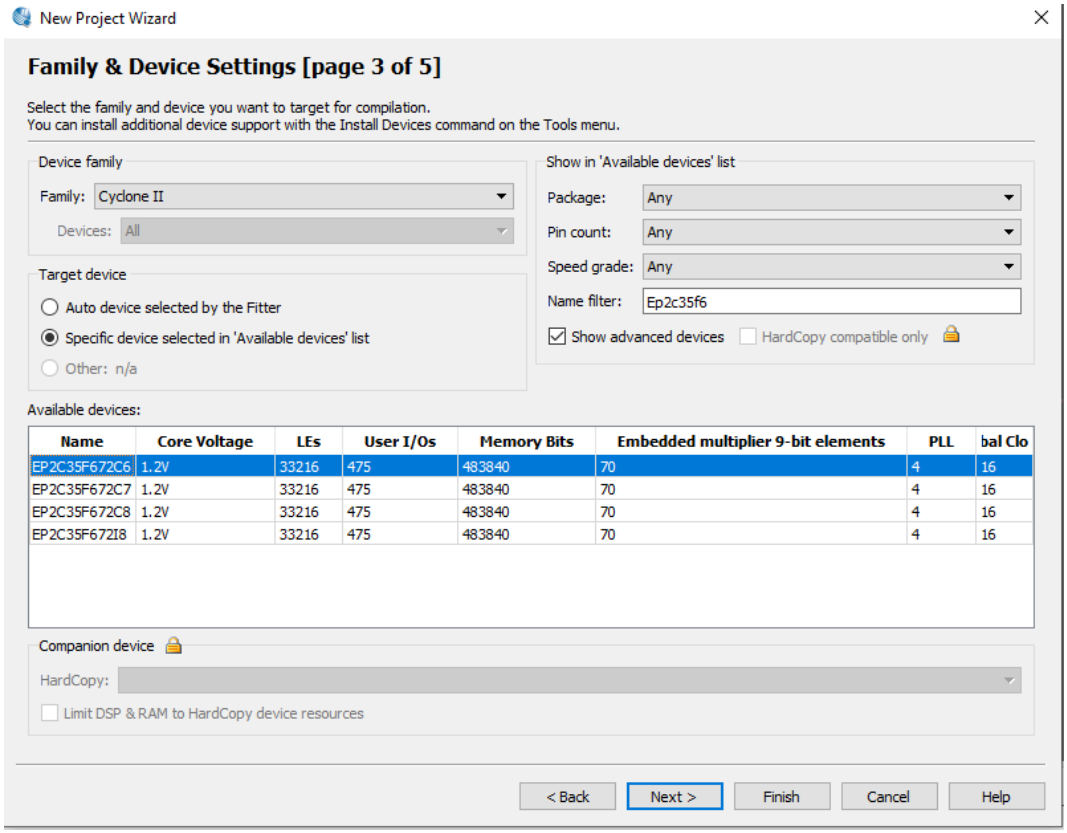

Figure 11. Cyclone II Family & Device Settings

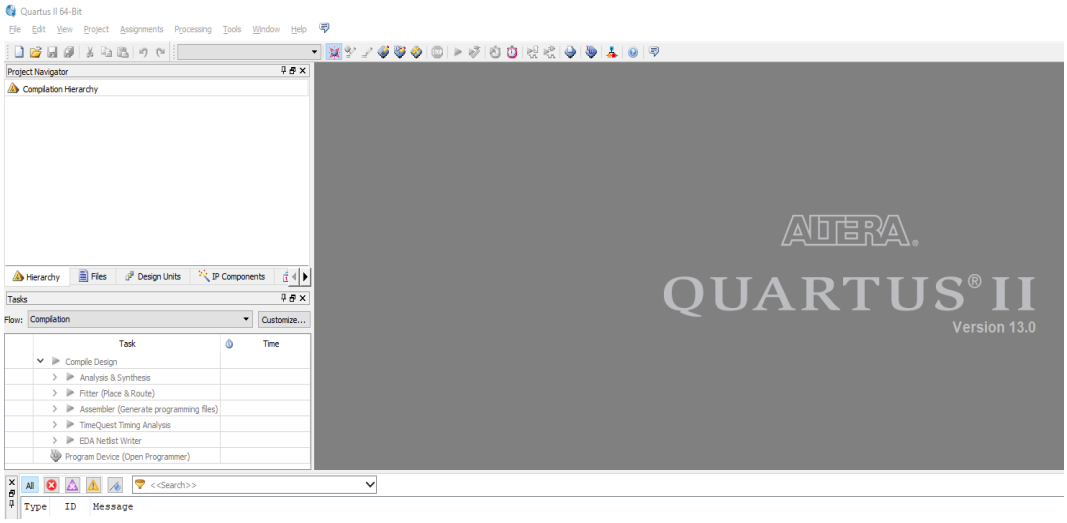

Figure 12. Quartus II user interface

4.2.2 UART_RX Module

The Universal Asynchronous Receiver-Transmitter (UART) is a computer hardware device for asynchronous serial communication in which the data format and transmission speeds are configurable. /10/

The goal of this module was to communicate between FPGA and computer with RS-232 serial communication. In this project, there was no transmitter module (TX) because FPGA did not send anything to the computer in return. FPGA only acted as the receiver to fetch the data from the keyboard. The idea of the module design is based on the UART timing diagram shown in Figure 13.

Figure 13. UART timing diagram /11/

```
type state_type is (
  DETECT_START,
  WAIT_START,
  WAIT_HALF_BIT,
  SAMPLE_DATA,
  WAIT_STOP,
  CHECK_STOP);
signal state : state_type;
```

Code Snippet 1. UART receiver state

Firstly, the module detected where were start bit, stop bit, and half bits of transferred data from the serial terminal. By using a finite state machine, whenever the new data comes to the UART_RX module on every clock pulse and the reset button is not pressed, the DETECT_START state will be enabled and is set as the initial state then move to the WAIT_START state. In this state, the module will find the start bit of the data and wait until the rising edge to move to the next state. The vital purpose of the WAIT_HALF_BIT state is to make sure that the data is stable and

clear. After making sure the process is at half bit of the first bit of the data, the state will move to the next state called SAMPLE_DATA.

```
case state is

  -- Wait for the falling edge on rx
  when DETECT_START =>
    if rx = '0' and rx_p1 = '1' then
      state <= WAIT_START;
      clk_counter <= 1;
      stop_bit_error <= '0';
    end if;

  -- Wait for the duration of the start bit
  when WAIT_START =>
    if clk_counter = clk_counter_type'high then
      state <= WAIT_HALF_BIT;
      clk_counter <= 0;
    else
      clk_counter <= clk_counter + 1;
    end if;

  -- Wait until we are at the middle of data bit 0
  when WAIT_HALF_BIT =>
    if clk_counter = clk_counter_type'high / 2 then
      state <= SAMPLE_DATA;
      clk_counter <= clk_counter_type'high;
    else
      clk_counter <= clk_counter + 1;
    end if;
```

Code Snippet 2. Finite State Machine's first 3 states

Coming to the fourth state, the UART_RX module will sample all data bits from bit 0 to bit 7 by shifting the data from high to low index. Whenever the data in bit 7 is read, the WAIT_STOP state will be enabled to wait for the duration of the stop bit then move to CHECK_STOP to check and output the data.

```
      -- Sample all data bits
      when SAMPLE_DATA =>
        if clk_counter = clk_counter_type'high then
          clk_counter <= 0;

          -- Shift the data in from high to low index
          shift_reg(shift_reg'high) <= rx;
          for i in shift_reg'high downto shift_reg'low + 1 loop
            shift_reg(i - 1) <= shift_reg(i);
          end loop;

          if bit_counter = data'high then
            state <= WAIT_STOP;
            bit_counter <= 0;
          else
            bit_counter <= bit_counter + 1;
          end if;

        else
          clk_counter <= clk_counter + 1;
        end if;

      -- Wait for the duration of the stop bit
      when WAIT_STOP =>
        if clk_counter = clk_counter_type'high then
          state <= CHECK_STOP;
          clk_counter <= 0;
        else
          clk_counter <= clk_counter + 1;
        end if;

      -- Check that the stop bit is '1' and output data
      when CHECK_STOP =>
        state <= DETECT_START;
        data <= shift_reg;
        valid <= '1';
        shift_reg <= (others => '0');

        if rx = '0' then
          stop_bit_error <= '1';
        end if;

    end case;
```

Code Snippet 3. Finite State Machine's last 3 states

4.2.3 CHAR_BUF Module

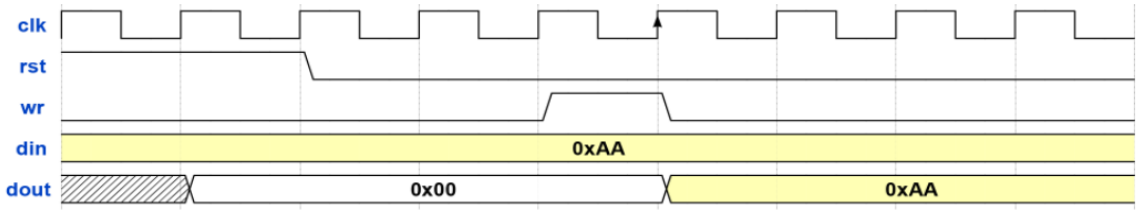

Figure 14. CHAR_BUF timing diagram

```
begin
  if rising_edge(clk) then
    if rst = '1' then
      dout <= (others => '0');

    else
      if wr = '1' then
        dout <= din;
      end if;

    end if;
  end if;
```

Code Snippet 4. CHAR_BUF

When the DE2 was initialized, the reset always stay at a high value, which means resetting the whole system. The reset remained in the high state for several clock cycles then turn to a low state and allow the system to start to work.

After the last state of UART_RX was completed, the value valid was set to high. Therefore, the data was transferred from the RX module to data in (din) and waited for the write signal (wr). This led to trigger the wr value from ‘0’ to ‘1’ state.

Whenever the wr was triggered, data out (dout) detected the failing edge of wr to begin to receive the din value before transmitting to the CONTROLLER module.

4.2.4 CONTROLLER Module

The idea of this part is to send a 4-bit combination of data through General Port Input Output (GPIO) from the Altera DE2-115 board. The board receives data from the CHAR_BUF module, stores it, and then transfers to GPIO to control the step motor.

a. Connection

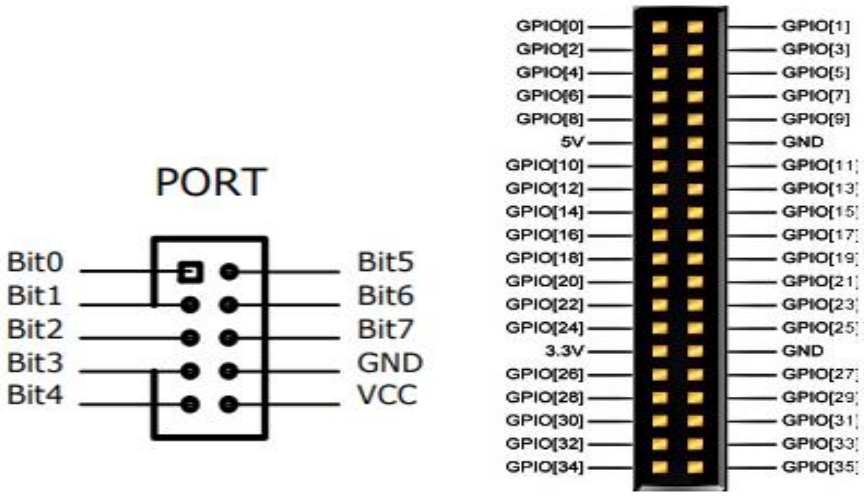

Figure 15. Motor driver port and GPIO pins of DE2

Motor board (J1 and J3 on board)	De2 (JP2 GPIO_1)
Bit0 (J1)	IO_B0
Bit1 (J1)	IO_B1
Bit2 (J1)	IO_B2
Bit3 (J1)	IO_B3
+5V (J3)	VCC5 (11)
GND (J1)	GND (12)

Figure 16. The connection between motor and Altera DE2

To connect the Altera DE2 board with the stepper motor, follow by Figure 15 and Figure 16

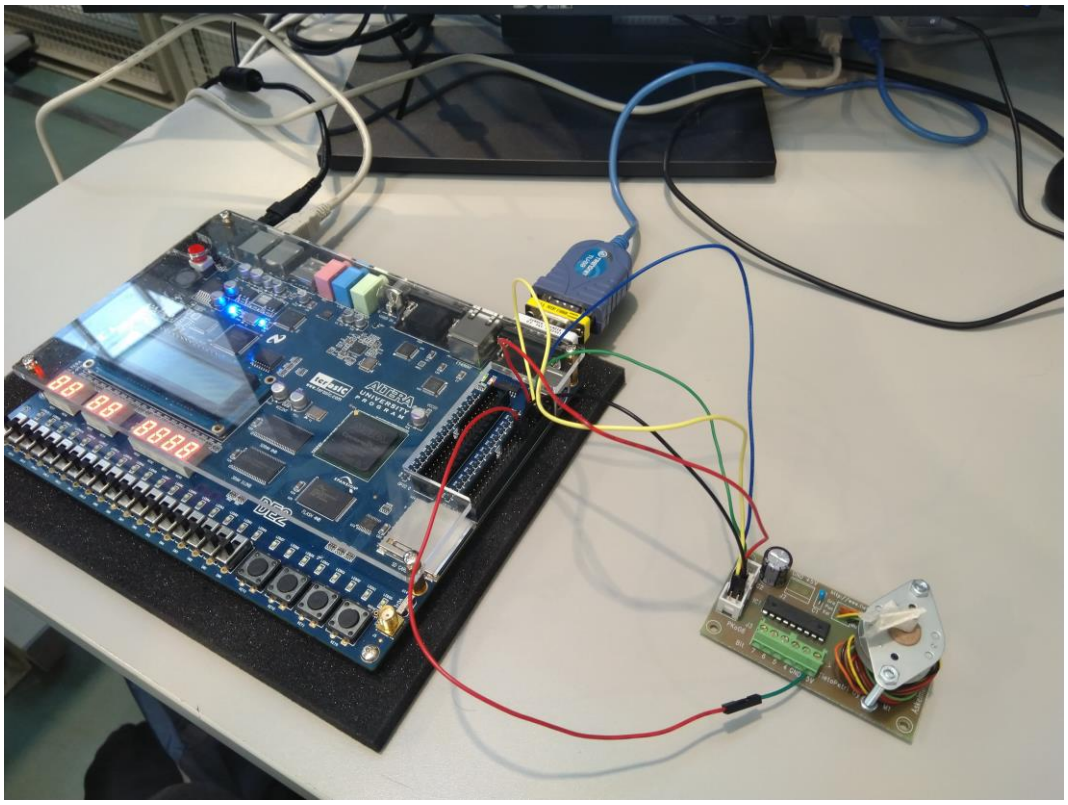

Figure 17. Hardware connection

b. Step sequence of the motor

The motor is comprised of 2 stator cups with claw tooth poles shaped around a winding making each half of the motor. The rotor has the same number of poles sets as the stator, whereas the posts on each stator container are developed to be half a post pitch separated. With two coils, this implies there can be 4 discrete positions per pole pitch. A 2-phase engine, for case, with 12-pole sets in each stator/coil division, will subsequently have 48 steps per transformation or 7.5 degrees per step. /12/

The data coming from the CHAR_BUF module will be converted from std_logic_vector to integer value then subtract 48 from that value (compare to the ASCII table shown in Figure 18). The reason why subtracts 48 is that if character ‘1’ is sent from the keyboard, it means that a string of binary comes to uart_data under std_logic_vector type with the value “0011 0001” will be converted to 49 as integer type. However, the value that needs to be assigned to

step_left is “1”, so 48 will be subtracted from 49. After that, the step_left value now is “1” and will be multiplied by 48 so the motor will rotate exactly 1 round (1 rotation contains 48 steps).

Dec	Hx	Oct	Html	Chr
48	30	060	0	0
49	31	061	1	1
50	32	062	2	2
51	33	063	3	3
52	34	064	4	4
53	35	065	5	5
54	36	066	6	6
55	37	067	7	7
56	38	070	8	8
57	39	071	9	9

Figure 18. ASCII table /13/

```
step_left <= std_logic_vector(to_unsigned(to_integer(unsigned(uart_data)) - 48, step_left'length));


i <= to_integer(unsigned(step_left)) * 48;     -- 1 rotation contains 48 steps
```

Code Snippet 5. Convert data into step sequence

Figure 20 is shown the step sequences of the step motor. Assign the step sequences to the GPIO output combination to control the rotation of the motor. The step sequence going from step 1 to 4 will make the motor rotate in a clockwise direction, and vice versa.

STEP	A	B	C	D
1	0	1	1	0
2	0	1	0	1
3	1	0	0	1
4	1	0	1	0
1	0	1	1	0

Figure 19. Step sequences of motor

```
-- ROM for step sequence ---------------------------------------
type step_type is array (0 to 3) of std_logic_vector(3 downto 0);
  constant step : step_type := (
  ("0110"),
  ("0101"),
  ("1001"),
  ("1010")
);
-----------------------------------------------------------------
```

Code Snippet 6. Step sequences stored in ROM

4.2.5 RESET Module

The RESET module is an important subject and it connects to all modules in the design. The way this module work is that when the FPGA power-up or the reset button is pressed, it will reset the whole design by providing a reset signal that will last for several clock cycles.

```
  signal counter : unsigned(7 downto 0) := (others => '0');

begin

  rst_out <= not counter(counter'high);

  PROC_RESET : process(clk)
  begin
    if rising_edge(clk) then
      if rst_in = '0' then
        counter <= (others => '0');

      else

        if counter(counter'high) = '0' then
          counter <= counter + 1;
        end if;

      end if;
    end if;
  end process; -- PROC_RESET
```

Code Snippet 7. RESET module

4.2.6 TOP Module

The TOP level structure is the most important module that connects all modules in the design. All the signals of previous modules will be mapped in the entity of TOP and assigned the input or output value based on the dataflow diagram.

```
39  RESET : entity Motor.reset(rtl)
40   port map (
41    clk => clk,
42    rst_in => KEY(0),
43    rst_out => rst
44  );
45  UART_RX_INST : entity Motor.RX(rtl)
46  port map (
47   clk  => CLK,
48   rst  => rst,
49    rx  => UART_RXD,
50    data => RX_DATA,
51    valid => RX_VALID,
52    stop_bit_error => RX_STOP
53  );
54
55  CHAR_BUF : entity Motor.char_buf(rtl)
56  port map (
57    clk => CLK,
58    rst => rst,
59    wr  => RX_VALID,
60    din => RX_DATA,
61    dout => char_buf_out
62  );
63
64  CONTROLLER : entity Motor.controller(rtl)
65  port map (
66  clk => CLK,
67  rst => rst,
68  sw17  => SW(17),
69  sw16  => SW(16),
70  uart_data => RX_DATA,
71  ledr17  => LEDR(17),
72  ledr16  => LEDR(16),
73  gpio_output => GPIO_1,
74  ledr_output => LEDR(7 downto 0)
75  );
```

Code Snippet 8. Mapping and assigning signals in TOP

5 CONCLUSION

The final product satisfied the requirements following the theory proposed:

- The communication protocol between the end-user and the FPGA board is based on UART.
- The total of rotations that the motor has rotated is equal to the input number from the keyboard to the serial terminal.
- The speed of the motor is the same as the frequency that the user input into the code.
- The motor operated smoothly in both directions clockwise and counter-clockwise.

In the future, this product can be developed by controlling the speed of the motor by the arrow button on the keyboard The user can interface with the motor behavior without using the switch and key of the FPGA board. Furthermore, the LCD will be used to display the input rotation and motor speed to help the user follow the working process easier.

REFERENCES

/1/ Altera DE2 Board. Terasic. Access 20 April 2022.http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=183&No=30&PartNo=2#heading

/2/ Field-programmable gate array. 2022. Wikipedia. Access 20 April 2022.https://en.wikipedia.org/wiki/Field-programmable_gate_array

/3/ VHDL. 2022. Wikipedia. Access 20 April 2022. https://en.wikipedia.org/wiki/VHDL

/4/Intel Quartus Prime. 2022. Wikipedia. Access 20 April 2022. https://en.wikipedia.org/wiki/Intel_Quartus_Prime

/5/ Rajesh Chandrasekhara Panicker. 2022. RealTerm. Access 18 April 2022.https://wiki.nus.edu.sg/display/ee4218/RealTerm

/6/ Tin-Can Motors. Electromate. Access 19 April 2022. https://www.electromate.com/products/stepper-motors/tin-can-motors/

/7/ Pantech. 2020. VHDL code for Stepper Motor. Pantechsolution .Access 20 April 2022. https://www.pantechsolutions.net/vhdl-code-for-stepper-motor

/8/ ULN2803A Darlington Transistor Arrays (Rev. H). Texas Instruments.Access 20 April 2022. https://www.ti.com/document-viewer/ULN2803A/datasheet/abstract#SLRS0495116

/9/ Moottorilevy. TietoPetri Oy.Access 19 April 2022. http://www.tietopetri.fi/data/motor.pdf

/10/ Universal asynchronous receiver-transmitter. 2022. Wikipedia. Access 18 April 2022.https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter

/11/ UART timing diagram. 2016. Wikipedia. Access 20 April 2022. https://commons.wikimedia.org/wiki/File:UART_timing_diagram.svg

/12/ Can Stack motors. Portescap. Access 20 April 2022.https://www.portescap.com/en/products/stepper-motors/can-stack-motors

/13/ ASCII Table. Access on 20 April 2022. https://www.asciitable.com/