IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 7, July 2025

DOI: 10.17148/IJIREEICE.2025.13706

A touch-sensing system for precise robotic arm control, using force-sensing resistors (FSR), VHDL and FPGAs

Dr Evangelos I. Dimitriadis[1], Leonidas Dimitriadis[2], Aristeidis Grigoriadis[3]

Department of Computer, Informatics and Telecommunications Engineering, International Hellenic University, End of Magnisias Str, 62124 Serres Greece[1]

Undergraduate student, Department of Information and Electronic Engineering, International Hellenic University, 57400, Sindos Thessaloniki, Greece[2]

Undergraduate student, Department of Information and Electronic Engineering, International Hellenic University, 57400, Sindos Thessaloniki, Greece[3]

Abstract: A touch-sensing system, based on force-sensing resistances, FPGAs and VHDL, is presented here, capable of providing precise control of a robotic arm model, thus making it move up or down depending on which of two sensors is pressed. The robotic arm is connected to a step motor able to rotate clockwise or counterclockwise, in order to provide lowering or rising of the arm, respectively. In case that both sensors are stopped being pressed, robotic arm remains to its last obtained position. Both sensors use voltage divider circuit and their analog voltage values act as input to FPGA's ADC converter and both values are presented to seven-segment displays. Three different rotation speed scales are used for both sensors, depending on the exerted pressure and respective LED lights up to present the scale. Additionally, buzzer alarm and half of FPGA's board LEDs are activated if one of the sensors is pressed over a critical value which is dangerous for damaging sensors. Another control implemented in our system, makes the other half of FPGA's board LEDs light up, when a critical time value of continuous step motor operation is exceeded, in order to protect motor from overheating. The system uses DE10-Lite FPGA board with two FSR 402 sensors connected to it. Our system can work with a variety of force sensors and critical input voltage limits can be set by the programmer depending on the application that the system is implemented.

Keywords: Force-sensing resistances, Step motor, FPGA, VHDL, Buzzer, LEDs, robotic arm.

I. INTRODUCTION

FPGAs have the main advantage of combining software and hardware, thus having the ability of hardware programming for a series of applications. Languages used for FPGAs' programing are VHDL and Verilog and VHDL is the one used in our work.

An interesting application field of FPGAs is robotic arms control. [1-5] Presented works deal with topics such as robot arm controller using FPGA, vision guided dual arms robotic system with DSP and FPGA integrated system structure, FPGA components for integrating FPGAs into robot systems, FPGA-based robotic computing and also neuromorphic FPGA-based infrastructures for a robotic arm. All the above works use complicated systems and also expensive, but the problem of handling robotic arm moving is not clearly solved. Our system uses force-sensitive resistors providing a touch sensitive and precise control solution, for robotic arm handling. It also uses safety control system for avoiding large forces exerted on sensors and a warning system for protecting step motor from overuse. Two FSR 402 sensors are used for controlling rising up or moving down the arm. Seven-segment displays present input voltage values from both sensors and three external LEDs are used for each sensor to show the scale of step motor rotating speed. Another benefit of our system is that it can work with a variety of force sensors and also its cost is remarkably low.

II. DESIGN OVERVIEW AND OPERATION OF THE SYSTEM

Figure1 presents device overview and operational units of our system, using FPGA DE10-Lite board, while Figure 2 presents circuit diagram of the system.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 7, July 2025

DOI: 10.17148/IJIREEICE.2025.13706

It is obvious from both of the above figures that our system, except from DE10-Lite FPGA board, contains also some basic circuit parts. We can see two LED systems, one for each sensor, responsible for providing a visualized view of step motor rotating speed scale. Those scales are determined by force exerted on sensors and consequent input voltage obtained. A larger force leads to greater input voltage values. Although each sensor causes different step motor rotation (sensor1 counterclockwise causing rising of robotic arm – sensor2 clockwise causing lowering of robotic arm), both of them use the same rotation speed scales. Input sensor voltage values from 0.2V to 0.5V determine first rotating speed scale, turning blue external LED ON. Values from 0.51V to 0.9V determine second rotating speed scale turning yellow external LED ON, while values from 0.91 to 1.2V determine third rotating speed scale turning red external LED ON. All the above LEDs are OFF if no force is exerted on sensors. Another control system is buzzer circuit containing a transistor and diode and it is connected in I/O pins of the FPGA board. It is the circuit that controls buzzer's operation. The transistor is used for amplifying signal bit 1 sent by the I/O pins of FPGA board, in order to provide sufficient voltage supply for buzzer operation. It is used together with half of FPGA's board LEDs and they are both turned ON whenever a critical input voltage value of 1.5V is exceeded from one of the sensors. This system prevents hard sensor pressure avoiding sensor damaging.

Another system of great importance used here, is step motor unit. It is connected with a thread to our simple robotic arm model, allowing the arm move up and down depending on which sensor is pressed. As we mentioned above motor rotating speed is a result of sensor force value.

Time is the other input value used here and determines turning ON of second half FPGA's board LEDs, in case that time value of continuous step motor operation is exceeded.

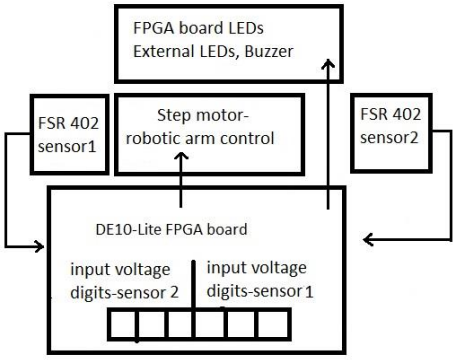

Figure 1: Device overview and operational units of our system.

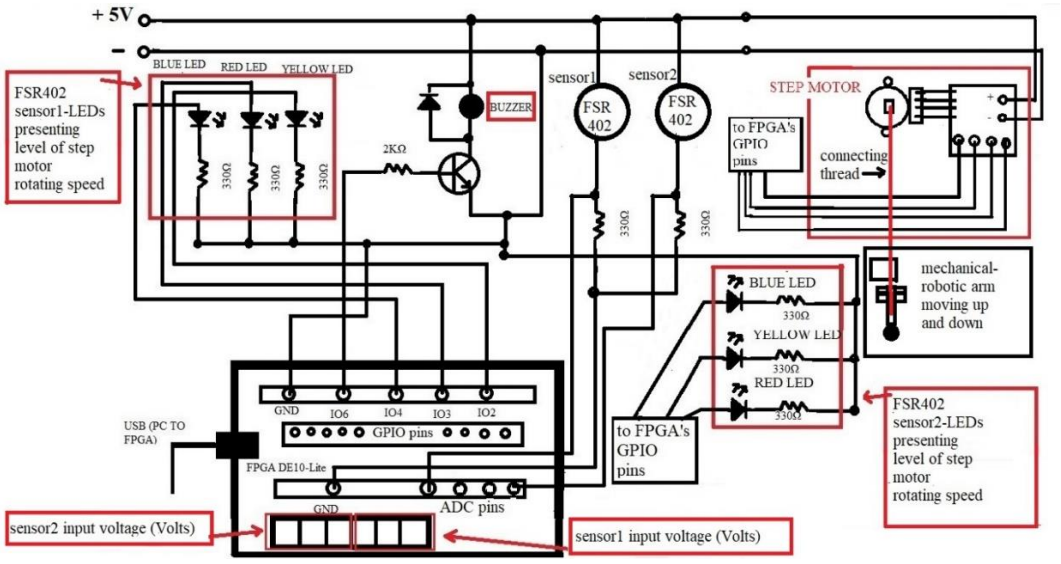

Figure 2: Circuit diagram of our system

DE10-Lite FPGA board used here offers its seven-segment displays for presenting input sensor voltage values and board LEDs mentioned above.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 7, July 2025

DOI: 10.17148/IJIREEICE.2025.13706

FSR 402 force sensor resistors, act as main input units. Both sensors are connected in series with 330Ω resistor, thus making a voltage divider for each sensor. We used this resistor value because it gives more gradual increase in input voltage with force, compared to a 100KΩ resistor. Figure 3 presents input voltage values vs force for FSR 402 sensors used here.

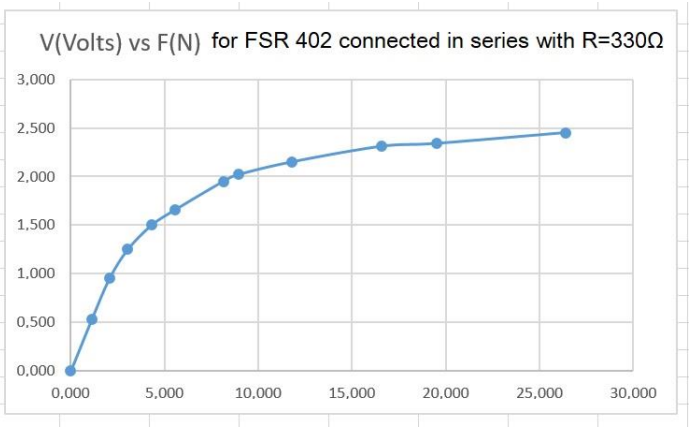

Figure 3: Input voltage vs force for FSR 402 sensor

Our system starts operating as soon as power supply +5V is applied to all circuits and the VHDL program is sent via USB Blaster interface, to FPGA chip, reading at first input voltages from FSR 402 sensors, with simultaneous start of time measurement. The analogue input voltages are converted to digital and presented in seven-segment displays. The system receives input voltage values periodically, ensuring continuous voltage change monitoring.

Consequently, units of step motor and rotating speed scale LEDs, are put in use. LED units receive bit 1 from FPGA, for their different LEDs in order to light them ON.

Figure 4 shows our FSR 402 touch-sensing system for precise robotic arm control.

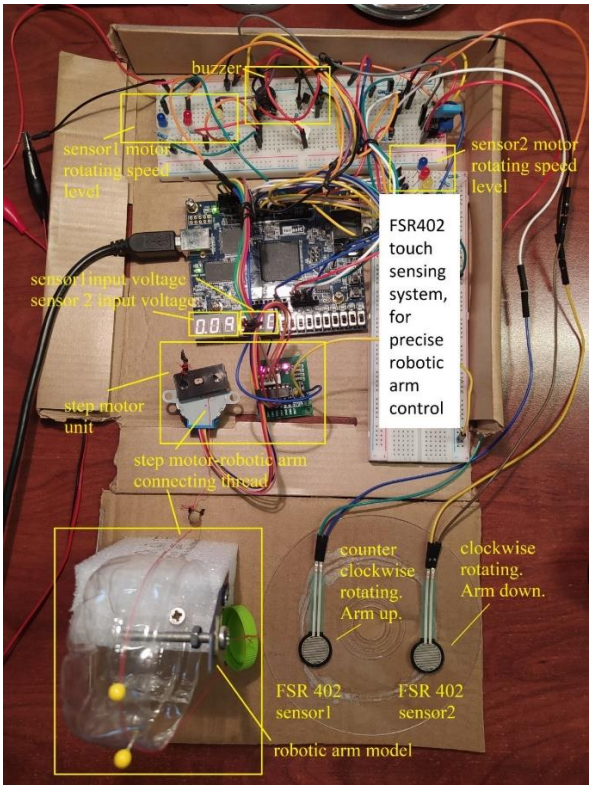

Figure 4: FSR 402 touch-sensing system for precise robotic arm control.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 7, July 2025

DOI: 10.17148/IJIREEICE.2025.13706

If one of FSR sensors is pressed then respective input voltage value is displayed in seven-segment displays and step motor starts rotating clockwise or counterclockwise, depending on which sensor is activated. Simultaneously robotic arm moves down or up, respectively and LED unit lights up specific LED (blue, yellow or red), depending on exerted force and consequent rotating speed scale in which the system operates.

Our system exhibits great sensitivity in sensors exerted force because buzzer starts sounding simultaneously with half board LEDs lighting up, as soon as input voltage of 1.5V is exceeded, meaning that according to Figure 3, a force value of over 5N is exerted. Another operation starts running if the arbitrary time of 60sec continuous operating of step motor, is exceeded. Then the other half board LEDs light up, in order to remind user to protect step motor from overuse and overheating. Needless to point out that 60sec time is not realistic for industrial applications but we used it in our program for simplicity reasons, in order to have a fast checking of our system's operation. All the above controls are periodically operated as long as the system is at the ON state. The system goes to OFF state if external circuit voltage supply is OFF or if FPGA board is unplugged from USB Blaster, or both of them.

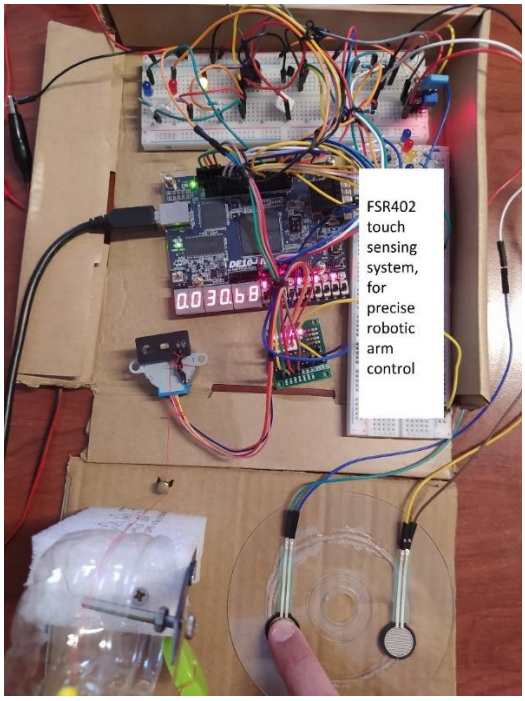

Figure 5a: FSR 402 touch-sensing system for precise robotic arm control, operating.

Figures 5a, 5b and 5c present our system in operation mode. Figures 5a and 5b show the use of sensor1 which results in counterclockwise rotation of step motor, thus leading robotic arm up. In Figure 5a we observe that input voltage value is 0.68V so the system is at second rotating speed scale (0.51-0.9V) and yellow LED lights up (in the upper left of the figure). We can also see that half of FPGA's board LEDs are ON because the critical value of 60sec of step motor operation is exceeded.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 7, July 2025

DOI: 10.17148/IJIREEICE.2025.13706

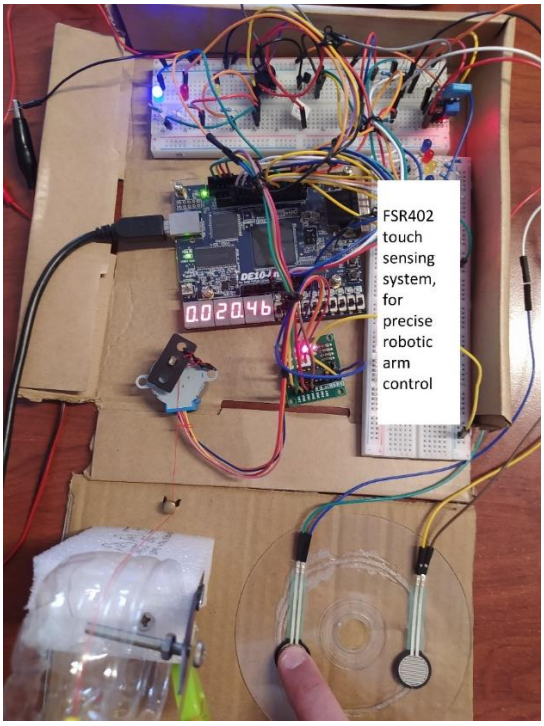

Figure 5b: FSR 402 touch-sensing system for precise robotic arm control, operating.

Figure 5b presents the system in first rotating speed scale (0.2-0.5V) and blue LED lights up (in the upper left of the figure). Input voltage value is 0.46V.

Figure 5c presents the system using sensor2 for clockwise step motor rotation, in second rotating speed scale (0.51-0.9V) and yellow LED lights up (in the upper right of the figure). Input voltage value is 0.58V.

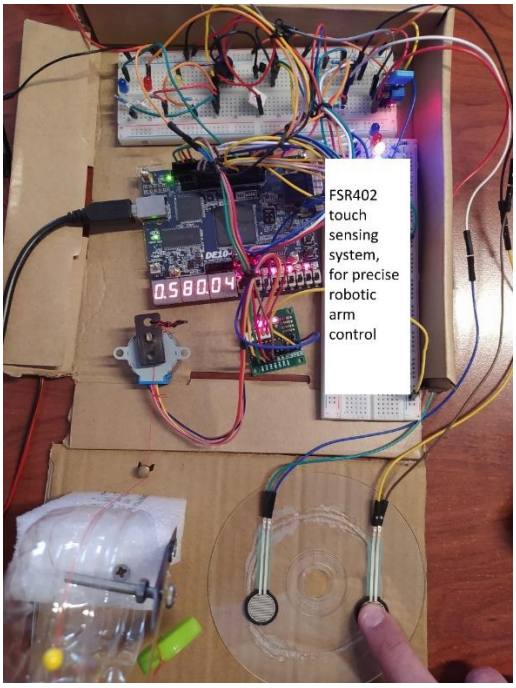

Figure 5c: FSR 402 touch-sensing system for precise robotic arm control, operating.

The result of the above system operation shown in Figures 5a and 5b is presented in Figures 6a, 6b and 6c where we observe that our hand-made robotic arm model, finally has its moving arm risen.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 7, July 2025

DOI: 10.17148/IJIREEICE.2025.13706

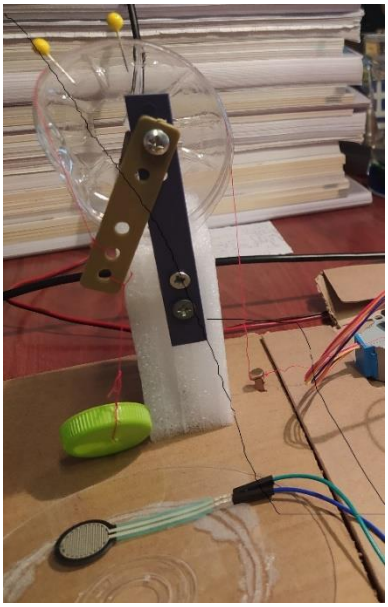

Figure 6a

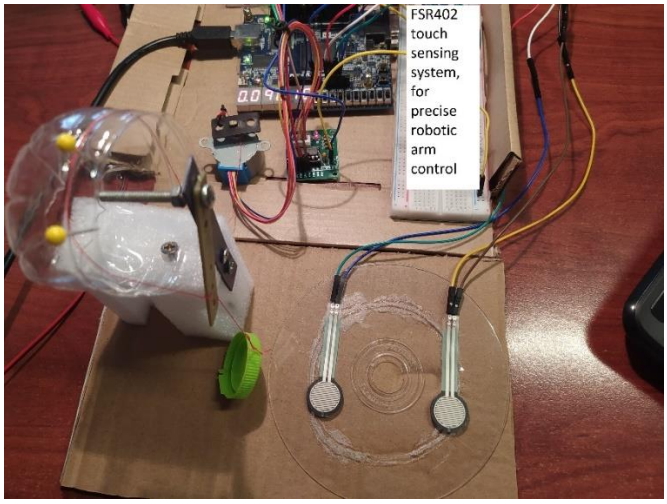

Figure 6b

Figure 6c

Figures 6a, 6b and 6c: Robotic arm model in use.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 7, July 2025

DOI: 10.17148/IJIREEICE.2025.13706

III. PROGRAMING THE SYSTEM

We used Quartus Prime Lite Edition 21.1.1 to create the VHDL programs of our system. It must be mentioned here that before proceeding with the VHDL programming of our system, we had to set a series of parameters controlling the operation of DE10-Lite FPGA's Analog to Digital Converter (ADC). This converter plays a very important role in the whole system operation, since it converts the analogue input voltages from FSR402 sensors connected to FPGA board to digital values, acting as main input of the system. The files created by the above ADC parameters setting are imported into the final project of our system.

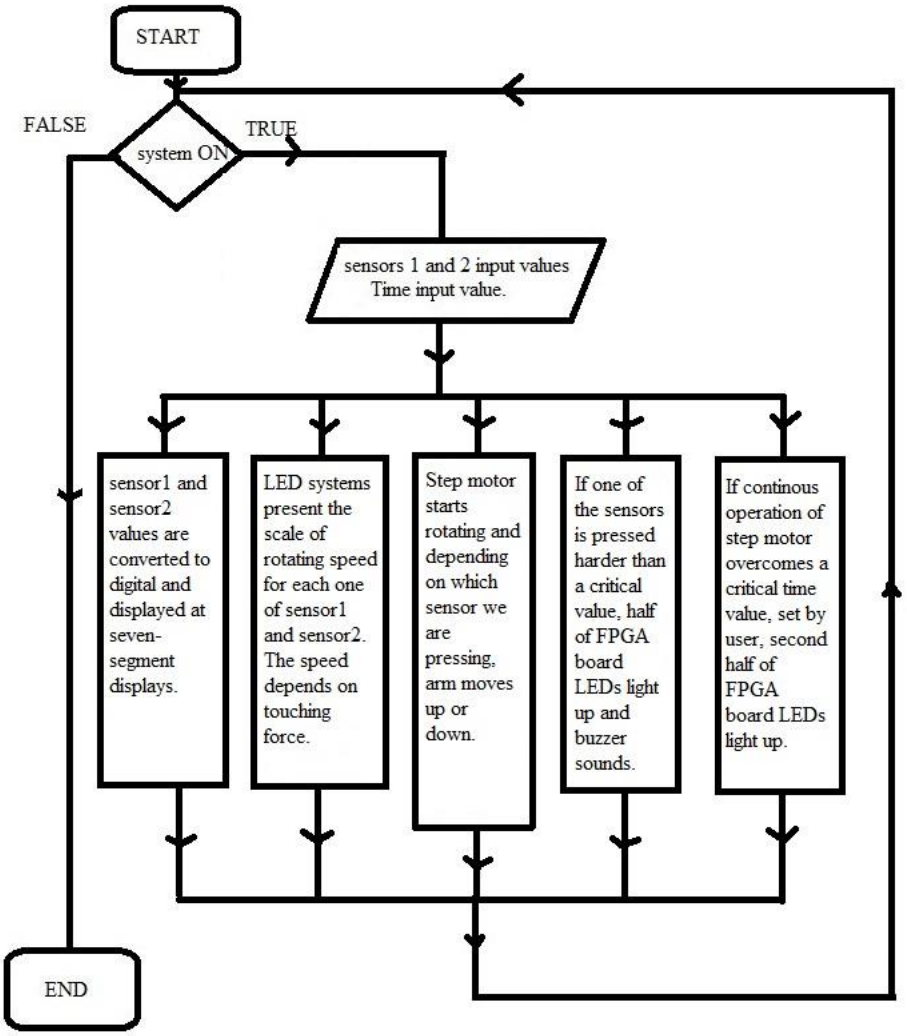

Figure 7: Flowchart diagram, presenting main functions-processes of our system.

A flowchart diagram, presenting main functions of our system is presented in Figure 7, while the APPENDIX contains the whole VHDL program.

It is clear that the system basically operates five functions. All of them use processes in VHDL programming language. The first function playing definitive role in system operation, uses analog input voltages provided from FSR402 voltage dividers shown in Figure2, as main input, convert them to digital values and present the final result in seven-segment displays. These processes are running as long as the system is ON. Calculated values for input sensor voltages are used in many other processes, in order to activate control systems and external or internal LEDs. Second function includes such processes which activate FSR corresponding LED systems, in order to present the scale of step motor rotating speed.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 7, July 2025

DOI: 10.17148/IJIREEICE.2025.13706

Third function of our system deals with step motor operating processes, which control clockwise or counterclockwise step motor rotation, depending on the sensor that is being pressed. Additionally they control step motor rotating speed for both sensors and we must mention that input voltage limits for determining speed scales, can be set and changed by programmer. Higher rotating speed is succeeded by changing specific parameter called "step" in step motor process. Robotic arm model is connected to step motor via a thread, causing its gradual rising or lowering. If no pressure is exerted to sensors, robotic arm remains to its last position.

Fourth function of our system constitutes a safety system, associated with sensors input values. If one of the sensors is pressed hard, thus is at risk of destruction, our program process activates buzzer and half of FPGA board LEDs, informing user to lower sensor pressure. We programed this upper pressure limit at an input voltage of 1.5V corresponding to about 5N force value. Needless to say that programmer could change this value depending on the application of our system.

Finally fifth function is another safety system, which turns ON the other half of FPGA's board LEDs as soon as a critical time value set by programmer is exceeded. Two processes are used here. A time measuring process which counts time in seconds simultaneously with turning system ON and another process which set a critical time value for activating board LEDs.

IV. CONCLUSION

A novel FPGA-based system is presented here, which manages to control a robotic arm model, giving the ability of rise or lower the arm. The system uses two force-sensing resistors FSR402 succeeding precise arm control. It also uses seven-segment displays for presenting input sensor voltage values and manages to rotate a step motor connected to robotic arm, clockwise or counterclockwise depending on the sensor we press. Step motor rotating speed can also be controlled corresponding to sensor exerted force and a LED system can show the scale of rotation speed for each sensor. Our system incorporates two safety controls, one for preventing sensor damage due to hard pressure and second for preventing step motor continuous overuse after exceeding a critical time value. The system can work with a variety of force sensors and it is easy to be manufactured for multiple applications, providing also the benefit of low cost.

REFERENCES

[1]. Urmila Meshram, Pankaj Bande, P. A. Dwaramwar and R. R. Harkare, "Robot arm controller using FPGA", International Multimedia, Signal Processing and Communication Technologies, 2009, DOI: 10.1109/MSPCT.2009.5164161

[2]. Shiuh-Jer Huang and Jian-Cheng Huang, "Vision guided dual arms robotic system with DSP and FPGA integrated system structure", Journal of Mechanical Science and Technology, Volume 25, pages 2067–2076, 2011

[3]. Takeshi OHKAWA, Kazushi YAMASHINA, Hitomi KIMURA, Kanemitsu OOTSU, Takashi YOKOTA, "FPGA Components for Integrating FPGAs into Robot Systems", IEICE TRANSACTIONS on Information Vol. E101-D No.2 pp.363-375, 2018, DOI 10.1587/transinf.2017RCP0011

[4]. Zishen Wan, Bo Yu, Thomas Yuang Li, Jie Tang, Yuhao Zhu, Yu Wang, "A Survey of FPGA-Based Robotic Computing", IEEE Circuits and Systems Magazine, Volume: 21, Issue: 2, 2021, DOI:10.1109/MCAS.2021.3071609

[5]. Salvador Canas-Moreno, Enrique Piñero-Fuentes, Antonio Rios-Navarro, Daniel Cascado-Caballero, Fernando Perez-Peña and Alejandro Linares-Barranco, "Towards neuromorphic FPGA-based infrastructures for a robotic arm", Autonomous Robots, Volume 47, pages 947–961, 2023.

APPENDIX

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.ALL;    -- step = step + 1
entity DE10_Lite_ADC_sensors is
generic(ClockFrequencyHz : integer:=50000000;
wait_count : natural := 1250000);  -- 50000000=1sec wait time for the stepper
port(
rst : in std_logic; --SW8
coils : out std_logic_vector(3 downto 0); -- connected to IN1..IN4
nRst    : in std_logic; -- Negative reset
Seconds : inout integer;
led1: out std_logic;
```

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 7, July 2025

DOI: 10.17148/IJIREEICE.2025.13706

```
led2: out std_logic;
led3: out std_logic;
led4: out std_logic;
led5: out std_logic;
led6: out std_logic;
led7: out std_logic;
led8: out std_logic;
led9: out std_logic;
led10: out std_logic;
led_blue1: buffer std_logic;
led_red1: buffer std_logic;
led_yellow1: buffer std_logic;
led_blue_out1: out std_logic;
led_red_out1: out std_logic;
led_yellow_out1: out std_logic;
led_blue2: buffer std_logic;
led_red2: buffer std_logic;
led_yellow2: buffer std_logic;
led_blue_out2: out std_logic;
led_red_out2: out std_logic;
led_yellow_out2: out std_logic;
buzzer:out std_logic;
Vr1: buffer integer;
Vr2: buffer integer;
d2abuf :buffer integer range 0 to 9;
d1abuf :buffer integer range 0 to 9;
d0abuf :buffer integer range 0 to 9;
d2bbuf :buffer integer range 0 to 9;
d1bbuf :buffer integer range 0 to 9;
d0bbuf :buffer integer range 0 to 9;
SW0 : in std_logic;
-- Clocks
ADC_CLK_10: in std_logic;
MAX10_CLK1_50: in std_logic;
MAX10_CLK2_50: in std_logic;
-- KEYs
KEY: in std_logic_vector(1 downto 0);
-- HEX
HEX0: out std_logic_vector(7 downto 0);
HEX1: out std_logic_vector(7 downto 0);
HEX2: out std_logic_vector(7 downto 0);
HEX3: out std_logic_vector(7 downto 0);
HEX4: out std_logic_vector(7 downto 0);
HEX5: out std_logic_vector(7 downto 0);
ARDUINO_IO: inout std_logic_vector(15 downto 0);
ARDUINO_RESET_N: inout std_logic);
-- GPIO
--GPIO: inout std_logic_vector(35 downto 0);
end entity;
architecture DE10_Lite_ADC_sensors_Arch of DE10_Lite_ADC_sensors is
-- Analog to Digital Converter IP core
component myADC is
port(
clk_clk: in std_logic := 'X';
modular_adc_0_command_valid: in std_logic := 'X';
modular_adc_0_command_channel: in std_logic_vector(4 downto 0) := (others => 'X');
modular_adc_0_command_startofpacket: in std_logic := 'X';
modular_adc_0_command_endofpacket: in std_logic := 'X';
modular_adc_0_command_ready: out std_logic;
modular_adc_0_response_valid: out std_logic;
modular_adc_0_response_channel: out std_logic_vector(4 downto 0);
modular_adc_0_response_data: out std_logic_vector(11 downto 0);
modular_adc_0_response_startofpacket: out std_logic;
modular_adc_0_response_endofpacket: out std_logic;
reset_reset_n: in std_logic
);
end component myADC;
signal modular_adc_0_command_valid: std_logic;
signal modular_adc_0_command_channel: std_logic_vector(4 downto 0);
signal modular_adc_0_command_startofpacket: std_logic;
signal modular_adc_0_command_endofpacket: std_logic;
signal modular_adc_0_command_ready: std_logic;
signal modular_adc_0_response_valid: std_logic;
```

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 7, July 2025

DOI: 10.17148/IJIREEICE.2025.13706

```
signal modular_adc_0_response_channel: std_logic_vector(4 downto 0);
signal modular_adc_0_response_data: std_logic_vector(11 downto 0);
signal modular_adc_0_response_startofpacket: std_logic;
signal modular_adc_0_response_endofpacket: std_logic;
signal clk_clk: std_logic;
signal reset_reset_n: std_logic;
type state_machines is (sm0, sm1, sm2, sm3, sm4);
signal sm: state_machines;
-- signals to store conversion results
signal ADCIN1, ADCIN2, ADCIN3, ADCIN4: std_logic_vector(11 downto 0);
signal AD1, AD2, AD3, AD4: std_logic_vector(11 downto 0);
-- signals for BCD digits
signal digit2a, digit1a, digit0a: std_logic_vector(3 downto 0);
signal digit2b, digit1b, digit0b: std_logic_vector(3 downto 0);
signal digit5, digit4, digit3, digit2, digit1, digit0: std_logic_vector(3 downto 0);
-- signal to determine how fast the
-- 7-seg displays will be updated
signal cnt: integer;
signal state_LED1: std_logic;
signal state_LED2: std_logic;
signal state_Vr1: integer;
signal state_Vr2: integer;
signal Ticks : integer;
-- signal for step motor control
signal count : natural range 0 to wait_count;
begin
-- ADC port map
adc1: myADC port map(
modular_adc_0_command_valid => modular_adc_0_command_valid,
modular_adc_0_command_channel => modular_adc_0_command_channel,
modular_adc_0_command_startofpacket => modular_adc_0_command_startofpacket,
modular_adc_0_command_endofpacket => modular_adc_0_command_endofpacket,
modular_adc_0_command_ready => modular_adc_0_command_ready,
modular_adc_0_response_valid => modular_adc_0_response_valid,
modular_adc_0_response_channel => modular_adc_0_response_channel,
modular_adc_0_response_data => modular_adc_0_response_data,
modular_adc_0_response_startofpacket => modular_adc_0_response_startofpacket,
modular_adc_0_response_endofpacket => modular_adc_0_response_endofpacket,
clk_clk => clk_clk,
reset_reset_n => reset_reset_n
);
clk_clk <= MAX10_CLK1_50;
reset_reset_n <= KEY(0);
-- process for reading new samples
p1: process(reset_reset_n, clk_clk)
begin
if reset_reset_n = '0' then
        sm <= sm0;
elsif rising_edge(clk_clk) then
        case sm is
                when sm0 =>
                        sm <= sm1;
                        modular_adc_0_command_valid <= '1';
                        modular_adc_0_command_channel <= "00001";
                when sm1 =>
                        if modular_adc_0_response_valid = '1' then
                                modular_adc_0_command_channel <= "00010";
                                ADCIN4 <= modular_adc_0_response_data;
                                sm <= sm2;
                        end if;
                when sm2 =>
                        if modular_adc_0_response_valid = '1' then
                                modular_adc_0_command_channel <= "00011";
                                ADCIN1 <= modular_adc_0_response_data;                                   sm <= sm3;
                        end if;
                when sm3 =>
                        if modular_adc_0_response_valid = '1' then
                                modular_adc_0_command_channel <= "00100";
                                ADCIN2 <= modular_adc_0_response_data;
                                sm <= sm4;
                        end if;
                when sm4 =>
                        if modular_adc_0_response_valid = '1' then
```

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 7, July 2025

DOI: 10.17148/IJIREEICE.2025.13706

```
                                        modular_adc_0_command_channel <= "00001";
                                        ADCIN3 <= modular_adc_0_response_data;
                                        sm <= sm1;
                                    end if;
                                when others =>
                    end case;
end if;
end process;
-- process for conversion from binary to BCD (first analog voltage)
p2: process(AD1, d2abuf, d1abuf, d0abuf)
variable vin: integer;
variable d2, d1, d0: integer;
begin
vin := to_integer(unsigned(std_logic_vector(to_unsigned(to_integer(unsigned(AD1)) * 500,
32))(31 downto 12)));
d2 := vin / 100;
d1 := vin mod 100 / 10;
d0 := ((vin mod 100) mod 10);
digit2a <= std_logic_vector(to_unsigned(d2, 4));
digit1a <= std_logic_vector(to_unsigned(d1, 4));
digit0a <= std_logic_vector(to_unsigned(d0, 4));
d2abuf<= d2;
d1abuf<= d1;
d0abuf<= d0;
end process;

-- process for conversion from binary to BCD (second analog voltage)
p3: process(AD2,d2bbuf,d1bbuf,d0bbuf)
variable vin: integer;
variable d2, d1, d0: integer;
begin
vin := to_integer(unsigned(std_logic_vector(to_unsigned(to_integer(unsigned(AD2)) * 500,
32))(31 downto 12)));
d2 := vin / 100;
d1 := vin mod 100 / 10;
d0 := ((vin mod 100) mod 10);
digit2b <= std_logic_vector(to_unsigned(d2, 4));
digit1b <= std_logic_vector(to_unsigned(d1, 4));
digit0b <= std_logic_vector(to_unsigned(d0, 4));
d2bbuf<= d2;
d1bbuf<= d1;
d0bbuf<= d0;
end process;
state_Vr1<= (d2bbuf*100)+(d1bbuf*10)+(d0bbuf);
Vr1<= state_Vr1;
state_Vr2<= (d2abuf*100)+(d1abuf*10)+(d0abuf);
Vr2<= state_Vr2;
-- determine how fast the 7-seg displays will be updated
p4: process(reset_reset_n, clk_clk)
begin
if reset_reset_n = '0' then
cnt <= 0;
elsif rising_edge(clk_clk) then
if cnt < 20_000_000 then
cnt <= cnt + 1;
else
cnt <= 0;
AD1 <= ADCIN1;
AD2 <= ADCIN2;
AD3 <= ADCIN3;
AD4 <= ADCIN4;
end if;
end if;
end process;
--time-seconds
process(MAX10_CLK1_50) is
   begin
      if rising_edge(MAX10_CLK1_50) then --same with “IF CLK'EVENT AND CLK='1' ”
         -- If the negative reset signal is active
         if nRst = '0' then
            Ticks   <= 0;
            Seconds <= 0;
         else
```

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 7, July 2025

DOI: 10.17148/IJIREEICE.2025.13706

```
        -- True once every second
          if Ticks = ClockFrequencyHz - 1 then
             Ticks <= 0;
                Seconds <= Seconds + 1;
          else
             Ticks <= Ticks + 1;
          end if;
        end if;
      end if;
   end process;
 --critical Vr1 or Vr2 value exceeded buzzer sounds
Process(MAX10_CLK1_50,Vr1,Vr2)
variable i : integer := 0;
BEGIN
IF Vr1>=150 OR Vr2>=150 THEN

if MAX10_CLK1_50'event and MAX10_CLK1_50 = '1' then
if i <= 50000000 then
i := i + 1;
buzzer <= '1';
elsif i > 50000000 and i < 100000000 then
i := i + 1;
buzzer <= '0';
elsif i = 100000000 then
i := 0;
end if;
end if;
end if;
end process;
--critical Vr1 or Vr2 value exceeded board LEDs light up
--(100->1Volt)
process(state_LED1,Vr1,Vr2)
begin
  IF Vr1>=150 OR Vr2>=150 THEN
               state_LED1 <= '1';
    else
               state_LED1 <= '0';
  end if;
end process;
led_red_out1<= led_red1;
led1 <= state_LED1;
led3 <= state_LED1;
led5 <= state_LED1;
led7 <= state_LED1;
led9 <= state_LED1;
--Vr1 within the first scale of FSR pressure then external blue led lights up
process(led_blue1,Vr1)
begin
  IF Vr1>=20 AND Vr1<=50 THEN
    led_blue1 <= '1';
    else
    led_blue1 <= '0';
  end if;
end process;
led_blue_out1<= led_blue1;
--Vr2 within the first scale of FSR pressure then external blue led lights up
process(led_blue2,Vr2)
begin
  IF Vr2>=20 AND Vr2<=50 THEN
    led_blue2 <= '1';
    else
    led_blue2 <= '0';
  end if;
end process;
led_blue_out2<= led_blue2;
--Vr1 within the second scale of FSR pressure then external yellow led lights up
process(led_yellow1,Vr1)
begin
  IF Vr1>=51 AND Vr1<=90 THEN
    led_yellow1 <= '1';
    else
    led_yellow1 <= '0';
  end if;
```

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 7, July 2025

DOI: 10.17148/IJIREEICE.2025.13706

```
end process;
led_yellow_out1<= led_yellow1;
--Vr2 within the second scale of FSR pressure then external yellow led lights up
process(led_yellow2,Vr2)
begin
  IF Vr2>=51 AND Vr2<=90 THEN
    led_yellow2 <= '1';
    else
    led_yellow2 <= '0';
  end if;
end process;
led_yellow_out2<= led_yellow2;
--Vr1 within the third scale of FSR pressure then external red led lights up
process(led_red1,Vr1)
begin
  IF Vr1>=91 AND Vr1<=120 THEN
    led_red1 <= '1';
    else
    led_red1 <= '0';
  end if;
end process;
led_red_out1<= led_red1;
--Vr2 within the third scale of FSR pressure then external red led lights up
process(led_red2,Vr2)
begin
  IF Vr2>=91 AND Vr2<=120 THEN
    led_red2 <= '1';
    else
    led_red2 <= '0';
  end if;
end process;
led_red_out2<= led_red2;
--Time-seconds exceeds critical value of system operation then board leds light up
process(state_LED2,Seconds)
begin
  IF Seconds>=60 THEN
    state_LED2 <= '1';
    else
    state_LED2 <= '0';
  end if;
end process;
led2 <= state_LED2;
led4 <= state_LED2;
led6 <= state_LED2;
led8 <= state_LED2;
led10 <= state_LED2;
MICROSTEP_PROC : process(MAX10_CLK1_50, rst, Vr1, Vr2)
      variable step : std_logic_vector(0 to 2) := "111";
   begin
      if rst = '1' then
         coils <= "0000";
         -- we start with a step
                                                count <= wait_count;
      elsif rising_edge(MAX10_CLK1_50) then
         if (count < wait_count) then
            -- wait for the next micro step
                                                          count <= count + 1;
         else
            -- perfom a single micro step
                                                          count <= 0;
            if (Vr1>=20) AND (Vr1<=50) then --1st positive rotating scale
               step := step + 1;
             end if;
              if (Vr1>=51) AND (Vr1<=90) then --2nd positive rotating scale
               step := step + 3;
             end if;
              if (Vr1>=91) AND (Vr1<=120) then --3rd positive rotating scale
               step := step + 5;
              end if;
            if (Vr2>=20) AND (Vr2<=50) then --1st negative rotating scale
               step := step - 1;
              end if;
              if (Vr2>=51) AND (Vr2<=90) then --2nd negative rotating scale
```

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 7, July 2025

DOI: 10.17148/IJIREEICE.2025.13706

```
            step := step - 3;
           end if;
           if (Vr2>=91) AND (Vr2<=120) then --3rd negative rotating scale
            step := step - 5;
          end if;
         case step is
           when "000" => coils <= "0001";
           when "001" => coils <= "0011";
           when "010" => coils <= "0010";
           when "011" => coils <= "0110";
           when "100" => coils <= "0100";
           when "101" => coils <= "1100";
           when "110" => coils <= "1000";
           when "111" => coils <= "1001";
           when others => coils <= "0000";
         end case;
       end if;
     end if;
   end process;
process(digit2a,digit1a,digit0a, digit2b,digit1b,digit0b,SW0)
begin
  IF SW0='0' THEN
  digit5 <= digit2a;
  digit4 <= digit1a;
  digit3 <= digit0a;
  digit2 <= digit2b;
  digit1 <= digit1b;
  digit0 <= digit0b;
  end if;
end process;
  WITH digit5 SELECT
HEX5 <= "01000000" WHEN "0000", -- display 0
"01111001" WHEN "0001", -- display 1
"00100100" WHEN "0010", -- display 2
"00110000" WHEN "0011", -- display 3
"00011001" WHEN "0100", -- display 4
"00010010" WHEN "0101", -- display 5
"00000011" WHEN "0110", -- display 6
"01111000" WHEN "0111", -- display 7
"00000000" WHEN "1000", -- display 8
"00011000" WHEN "1001", -- display 9
"01111111" WHEN OTHERS; -- blank display
WITH digit4 SELECT
HEX4 <= "11000000" WHEN "0000", -- display 0
"11111001" WHEN "0001", -- display 1
"10100100" WHEN "0010", -- display 2
"10110000" WHEN "0011", -- display 3
"10011001" WHEN "0100", -- display 4
"10010010" WHEN "0101", -- display 5
"10000011" WHEN "0110", -- display 6
"11111000" WHEN "0111", -- display 7
"10000000" WHEN "1000", -- display 8
"10011000" WHEN "1001", -- display 9
"11111111" WHEN OTHERS; -- blank display
WITH digit3 SELECT
HEX3 <= "11000000" WHEN "0000", -- display 0
"11111001" WHEN "0001", -- display 1
"10100100" WHEN "0010", -- display 2
"10110000" WHEN "0011", -- display 3
"10011001" WHEN "0100", -- display 4
"10010010" WHEN "0101", -- display 5
"10000011" WHEN "0110", -- display 6
"11111000" WHEN "0111", -- display 7
"10000000" WHEN "1000", -- display 8
"10011000" WHEN "1001", -- display 9
"11111111" WHEN OTHERS; -- blank display

 WITH digit2 SELECT
HEX2 <= "01000000" WHEN "0000", -- display 0
"01111001" WHEN "0001", -- display 1
"00100100" WHEN "0010", -- display 2
"00110000" WHEN "0011", -- display 3
"00011001" WHEN "0100", -- display 4
```

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 7, July 2025

DOI: 10.17148/IJIREEICE.2025.13706

```
"00010010" WHEN "0101", -- display 5
"00000011" WHEN "0110", -- display 6
"01111000" WHEN "0111", -- display 7
"00000000" WHEN "1000", -- display 8
"00011000" WHEN "1001", -- display 9
"01111111" WHEN OTHERS; -- blank display
WITH digit1 SELECT
HEX1 <= "11000000" WHEN "0000", -- display 0
"11111001" WHEN "0001", -- display 1
"10100100" WHEN "0010", -- display 2
"10110000" WHEN "0011", -- display 3
"10011001" WHEN "0100", -- display 4
"10010010" WHEN "0101", -- display 5
"10000011" WHEN "0110", -- display 6
"11111000" WHEN "0111", -- display 7
"10000000" WHEN "1000", -- display 8
"10011000" WHEN "1001", -- display 9
"11111111" WHEN OTHERS; -- blank display
WITH digit0 SELECT
HEX0 <= "11000000" WHEN "0000", -- display 0
"11111001" WHEN "0001", -- display 1
"10100100" WHEN "0010", -- display 2
"10110000" WHEN "0011", -- display 3
"10011001" WHEN "0100", -- display 4
"10010010" WHEN "0101", -- display 5
"10000011" WHEN "0110", -- display 6
"11111000" WHEN "0111", -- display 7
"10000000" WHEN "1000", -- display 8
"10011000" WHEN "1001", -- display 9
"11111111" WHEN OTHERS; -- blank display
end architecture;
```