Real Time Clock

Prasad Gaikwad EE18MTECH11001
Nisha Akole EE18MTECH11002

April 27, 2019

Instructor: Professor GVV Sharma

1 FPGA Introduction

Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing. This feature distinguishes FPGAs from Application Specific Integrated Circuits (ASICs), which are custom manufactured for specific design tasks. Although one-time programmable (OTP) FPGAs are available, the dominant types are SRAM based which can be reprogrammed as the design evolves.
The structure of the chip is a series of tiles of programmable logic gates connected together with wires and switches .One can program this chip to become any kind of digital circuit just by programming switches and gates. It turns any kind of hardware problem in software problem

2 Hardware

2.1 IcoBoard

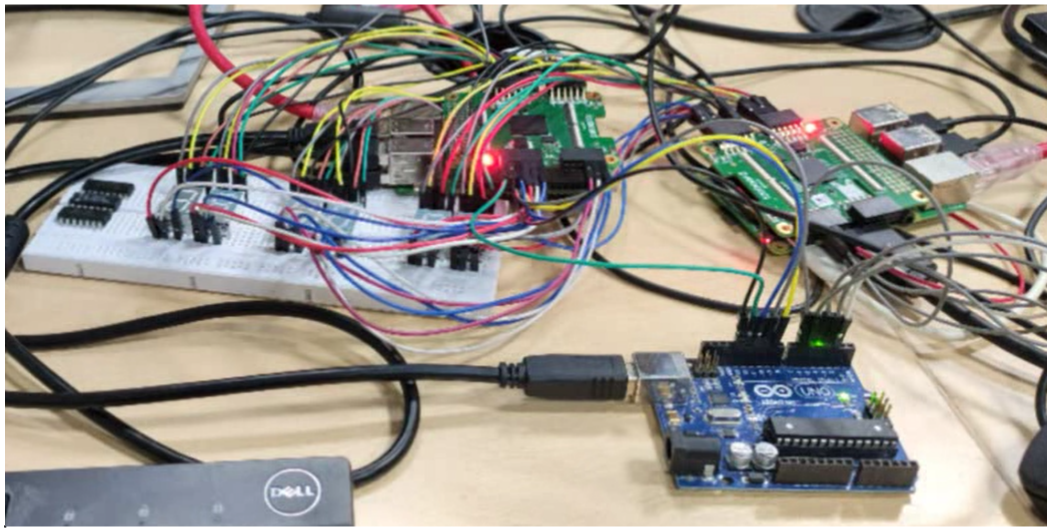
The icoBOARD contains a Lattice FPGA with 8k LUT, 100MHz max clock, up to 8 MBit of SRAM and is programmable in Verilog by a complete open source FPGA toolchain.
The icoTC (toolchain consisting of Yosys and ArachnePnR and icetools) for the Lattice ICE40 series does support all chip components like PLLs, Block RAMs, the WARMBOOT macro, dedicated carry logic, and IO blocks. The icoBoard is pincompatible with the RaspberryPi 2B, newer versions and any board using the same pinout. The icoTC can generate bitstream-files directly on the RaspberryPi.
The icoBoard can also be operated standalone without RaspberrPi. Up to 200 IO Pins or 20 PMOD modules can be connected to the icoBOARD. 3 flatflex connectors are available to connect additional boards.

2.2 Arduino Uno

The Arduino UNO is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 Digital pins, 6 Analog pins, and programmable with the Arduino IDE (Integrated Development Environment) via a type B USB cable. It can be powered by a USB cable or by an external 9 volt battery, though it accepts voltages between 7 and 20 volts.

2.3 Seven Segment Display

A seven-segment display (SSD), or seven-segment indicator, is a form of electronic display device for displaying decimal numerals that is an alternative to the more complex dot matrix displays.
Seven-segment displays are widely used in digital clocks, electronic meters, basic calculators, and other electronic devices that display numerical information



3 Software

3.1 Code from .v file

```
module rtc(clk,rst,second,minute,hour,rx);
input clk,rst;
output [6:0] sec_l,sec_u,min_l,min_u,hour_l,hour_u;
reg [5:0] second;
reg [5:0] minute;
reg [5:0] hour;
input [5:0] rx;
reg [26:0] delay;
reg [31:0] delay1;

Sevenseg ss0(.cnt(second mod 10),.out(sec_l));
Sevenseg ss1(.cnt(second/10),.out(sec_u));
Sevenseg ss2(.cnt(minute mod 10),.out(min_l));
Sevenseg ss3(.cnt(minute/10),.out(min_u));
Sevenseg ss4(.cnt(hour mod 10),.out(hour_l));
Sevenseg ss5(.cnt(hour/10),.out(hour_u));
```

```
always@(posedge clk)
begin
delay = delay + 1;
delay1 = delay1 + 1;
if(delay1 ≤ 32′b00000101111101011110000100000000)
begin
second[0] = rx[0];
second[1] = rx[1];
second[2] = rx[2];
second[3] = rx[3];
second[4] = rx[4];
second[5] = rx[5];
end
if(delay1 ≥ 32′b000001011111010111100001000000000anddelay1 ≤ 32′b0000101111101011110000100000000 00)
begin
minute[0] = rx[0];
minute[1] = rx[1];
minute[2] = rx[2];
minute[3] = rx[3];
minute[4] = rx[4];
minute[5] = rx[5];
end
if(delay1 ≥ 32′b0000101111101011110000100000000 00anddelay1 ≤ 32′b00010001111000011010001100000000)
begin
hour[0] = rx[0];
hour[1] = rx[1];
hour[2] = rx[2];
hour[3] = rx[3];
hour[4] = rx[4];
hour[5] = rx[5];
end
if(delay == 27’b101111101011110000100000000)
begin
delay = 27’b0;
second = second + 1’b1;
if(second == 6’b111011)
begin
minute = minute +1’b1;
second = 6’b0;
if (minute == 6’b111011)
begin
hour = hour + 1’b1;
minute = 6’b0;
if (hour == 6’b010111)
begin
```

```
hour = 6'b0;
end
end
end
end
end
endmodule

module Sevenseg(cnt,out);
input [5:0]cnt;
output reg [6:0]out;
always@ (*)
begin
case(cnt)
(6'b000000): out = 7'b1000000;
(6'b000001): out = 7'b1111001;
(6'b000010): out = 7'b0100100;
(6'b000011): out = 7'b0110000;
(6'b000100): out = 7'b0011001;
(6'b000101): out = 7'b0010010;
(6'b000110): out = 7'b0000010;
(6'b000111): out = 7'b1111000;
(6'b001000): out = 7'b0000000;
(6'b001001): out = 7'b0011000;
default: out = 7'bxxxxxxx;
endcase
end
endmodule
```

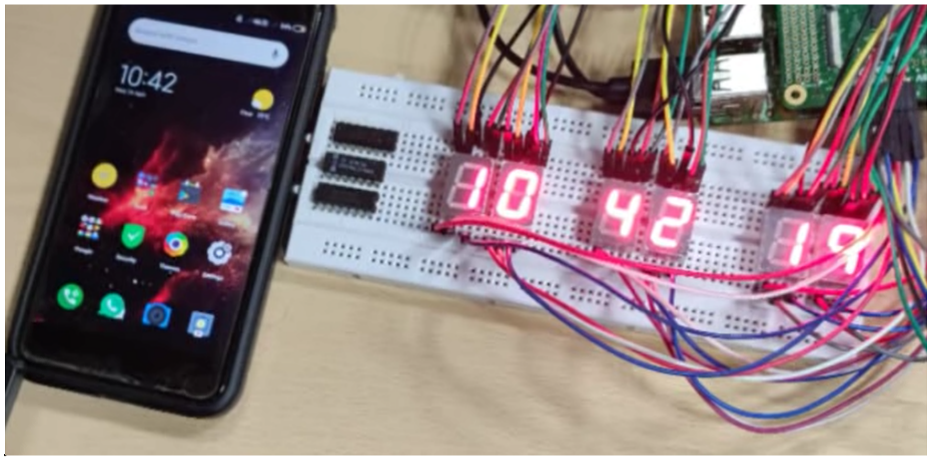
3.2 Pinout Connection from .pcf file

```
set_io clk R9

set_io sec_l[0] C13
set_io sec_l[1] A15
set_io sec_l[2] L12
set_io sec_l[3] J16
set_io sec_l[4] E9
set_io sec_l[5] C11
set_io sec_l[6] P14

set_io sec_h[0] B14
set_io sec_h[1] e10
set_io sec_h[2] T16
set_io sec_h[3] J14
set_io sec_h[4] A16
set_io sec_h[5] F9
set_io sec_h[6] F14

set_io min_l[0] M8
set_io min_l[1] N7
set_io min_l[2] B7
set_io min_l[3] B6
set_io min_l[4] B3
set_io min_l[5] P9
set_io min_l[6] N9

set_io min_h[0] L7
set_io min_h[1] N6
set_io min_h[2] A5
set_io min_h[3] A2
set_io min_h[4] C3
set_io min_h[5] G5
set_io min_h[6] L9

set_io hour_l[0] T11
set_io hour_l[1] R10
set_io hour_l[2] B5
set_io hour_l[3] B8
set_io hour_l[4] A9
set_io hour_l[5] T14
set_io hour_l[6] T15

set_io hour_h[0] T10
set_io hour_h[1] T9
set_io hour_h[2] B4
set_io hour_h[3] D8
set_io hour_h[4] B9
set_io hour_h[5] T13
set_io hour_h[6] R14

set_io Rx[0] D15
set_io Rx[1] E13
set_io Rx[2] G13
set_io Rx[3] G14
set_io Rx[4] D16
set_io Rx[5] E13
```

4 Algorithm

- Initialise all the six registers (each two for sec, min, hour) to zero.
- second and minute ranges from 0 to 59 and hour ranges from 0 to 24.
- Second register is up counted by one for every one second which is equal to 100Mhz.
- For every 60 second count, increment minute register by one. Similarly for every 60 minute count, update hour with count equal to previous value plus one.
- Arduino is parallely communicating with FPGA.
- Initially Real time is taken from Arduino and run the above algorithm continuously.

5 Simulation Results



6 Conclusion

We have learned

- How to use counters and registers in FPGA?
- Arduino interfacing with FPGA
- Successfully built and run the Real Time Clock

References

- http://icoboard.org
- https://www.arduino.cc