

# Final Report: Nearest Station Locator

Chinmay Rai(2016CS50615)

Arshdeep Singh(2016CS50625)

- **Introduction**

We designed and implemented a System which returns the code of nearest station from the user. The coordinates of the user are given in terms of latitude° and longitude° with accuracy if two decimal points. The System will find the nearest station from a predefined list of stations and display the code of the nearest station.

- **Structural design**

The VHDL implementation of the design consists of two submodules inside the top module; one of the modules is responsible for making all the computations and the other module displays the output of the computation.

The computation module named “project” comprises of the logic to input the coordinates of the user and compute the distance from each station; thereby finding the minimum distance station and passing its code to the “display” module for display.

The project module is designed to stay in either of the states: Query, Update, do\_nothing . The query state is the one which allows the user to query the nearest station; the update state allows the user to add stations to the pre-existing list of stations, while the do\_nothing state puts the system in idle state.

The display module along with displaying the station code also uses the seven-segment display to instruct the user as to when they are supposed to enter their latitude and longitude.

- **Approach**

The computation of distance of between two given points using their latitude and longitude values is done using the formula:  $a = (\Delta \text{lat})^2 + (\Delta \text{long})^2$ .

The value of a can be seen as a scaled down version of the distance between two points. This is application of Pythagoras theorem in the sense that distance between any two points on earth can be approximated using the formula  $d = r(\Delta \alpha)$ ; where  $\Delta \alpha$  is the difference in latitude/longitude coordinates of the two points.

We store this value for all pairs formed by the user coordinates and the coordinates of the cities. We then compute the minimum of these values and send it to the other module for

display. The display outputs a number between 0 to 50 which is supposed to be the reference code of the nearest station.

- **The Update State**

The update state of the system is used to add coordinates of new station into the predefined list of stations which is used for computation of nearest station. Addition of new station can be done by putting the system in UPDATE state by using the state change button. Thereafter we need to enter and record the value of system as prompted by the system. The addition of new values of station will start from the index 33 as 33 stations (index=0 to index=32) are already added into the list of predefined stations into the code.

- **Input Methodology**

In order to input the latitude and longitude coordinates we use all the 16 slide switches available on the board. Values of LAT and LONG in degrees are input after multiplying by 100. (We assume that 2 point decimal accuracy is sufficient to distinguish between two major stations.) For eg: - 78.23° is input as 7823. Firstly Latitude value is entered, recorded then the longitude value is entered and recorded (Using record\_lat & record\_long buttons). The method of entry is as that we treat each digit of the 4 digit number as a hexadecimal number and use 4 slide switches to assign value to it. Hence 4-digit value can be recorded using (4\*4) switches.

- **Problems encountered and their solutions**

- The initial description of the project prompted us to use UART as the method of input, but in subsequent discussions with the project in charge we were instructed to use the slide switches as a directly as the method of input instead of the UART, due to lot of complications involved in the implementation of UART.
- The ideal method of computing the distance between two points is the haversine formula.  $a = \sin(\Delta \text{lat}/2) * \sin(\Delta \text{lat}/2) + \cos(\text{lat}1) * \cos(\text{lat}2) * \sin(\Delta \text{long}/2) * \sin(\Delta \text{long}/2)$ . This involves the use of sinusoidal function which cannot be implemented on the board. We sought to implement the sine function by storing the values of sinusoidal function for integers and linearly approximating the values of sine at when fractional inputs are given. The code for the function that we tried to implement to compute the sine values is given below:

```
function sine (val: integer range 0 to 10000) return integer is
    type ARRAY_TYPE2 is array (0 to 91) of integer range 0 to 10000;
    constant sine_table : ARRAY_TYPE2
:= (0,174,349,523,697,871,1045,1218,1391,1564,1736,1908,2079,2249,2419,2588,2756,2923,3090,3255,3420,3583,3746,
3907,4067,4226,4383,4539,4694,4848,5000,5150,5299,5446,5591,5735,5877,6018,6156,6293,6427,6560,6691,6820,694
6,7071,7193,7313,7431,7547,7660,7771,7880,7986,8090,8191,8290,8386,8480,8571,8660,8746,8829,8910,8987,9063,9
```

```

135,9205,9271,9335,9396,9455,9510,9563,9612,9659,9703,9743,9781,9816,9848,9876,9902,9925,9945,9961,9975,9986
,9993,9998,10000,9998);
    variable right : integer range 0 to 100;
    variable left : integer range 0 to 100;
    variable final_val : integer range 0 to 10000;
    begin right := (val) mod 100; left := (val-right)/100;
        final_val := sine_table(left)
+ (right*(sine_table(left+1)-sine_table(left)))/100;
    return final_val; end sine;

```

We were not able to implement this function because the time required to output the sine value of a number is very high due to the type of operations used in this sine function. The operations of division and multiplication lead to the formation of large dividers and multipliers which require more than a clock cycle to output the sine value. This compelled us to use some approximations for computing the distance between two points.

- Our initial prototype of design would lead to the exhausting of resources when we tried to synthesize and implement it on the board. Our initial impression was that it happened due to the large amount of data hardcoded into the module; but on analysis of the utilisation report of synthesis we realised that a lot of LUT were being synthesized due to a lot of operations being performed in one cycle of clock. Distributing those operations over multiple clock cycles lead to successful resolution of error.

The list of cities along with their code is as follows :

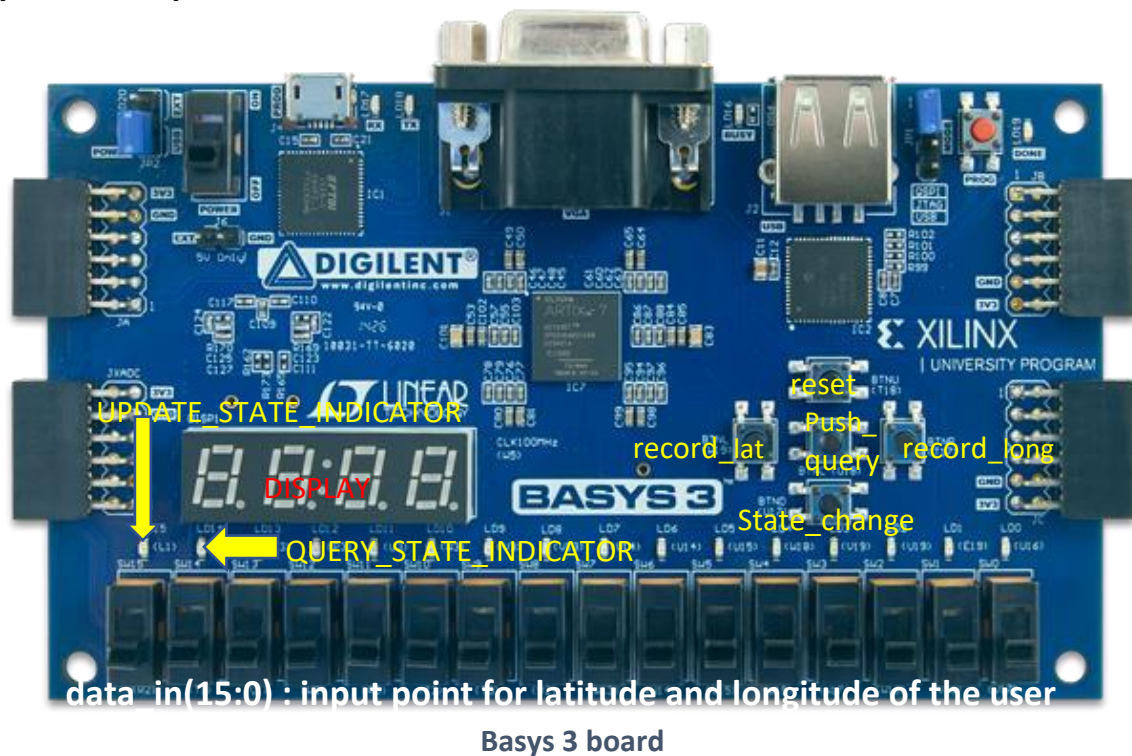
0	Agartala	11	Jaipur	22	Patna
1	Agra	12	Kandla	23	Port Blair
2	Ahmedabad	13	Kanpur	24	Pune
3	Amritsar	14	Kochi	25	Shillong
4	Bangalore	15	Kolkata (Calcutta)	26	Shimla
5	Chandigarh	16	Kulpahar	27	Siliguri
6	Chennai (Madras)	17	Lucknow	28	Srinagar
7	Dibrugarh	18	Ludhiana	29	Surat
8	Gangtok	19	Mumbai (Bombay)	30	Tezpur
9	Guwahati	20	Nagpur	31	Trivandrum
10	Hyderabad	21	Jaipur		

- **Input /Output ports**

Pin Name	TYPE	Description
----------	------	-------------

data_in(16:0)	input	Used to input latitude and longitude data using the slide switches
state_change	input	Pushbutton used to change the state of the system
reset	input	Pushbutton used to reset the state of the system
record_lat	input	Pushbutton used to register the latitude value of user
record_long	input	Pushbutton used to register the longitude value of the user
push_query	input	Pushbutton used to initiate the query of nearest station
update_state_indicator	output	Indicates the presence of system in UPDATE state
Query_state_indicator	output	Indicates the presence of system in QUERY state

- Input Switch positions**



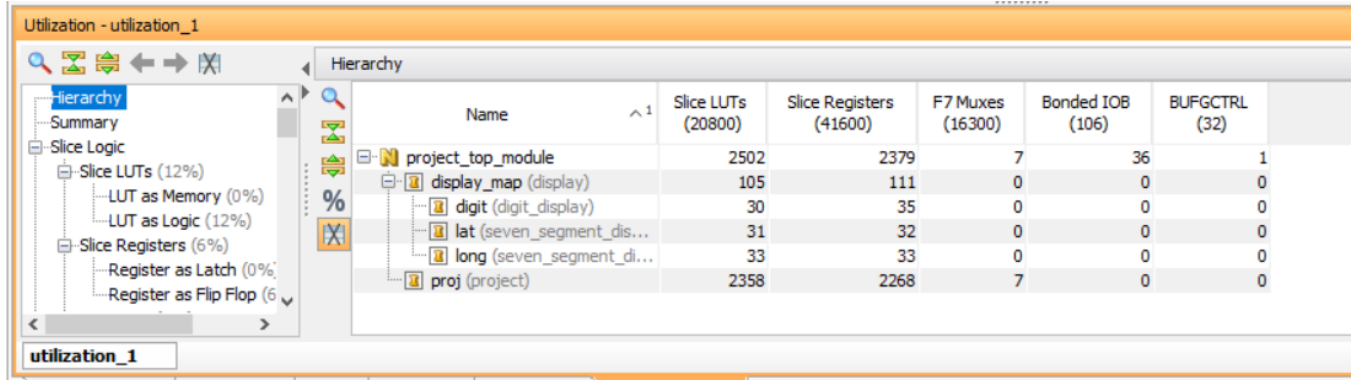
- Validation Methodology**

The validation for the system is done by the unit level testing of both the submodules: project and display; by the means of separate test-bench for each of them. The Respective modules can be tested against a given set of testcases by means of simulating the test bench with the respective module (setting it as top). We also created a test bench for the top module of the system and simulate it against a set of unit test cases. Top level testing of the system was done by updating the database and making query from the board and checking the validity of output on Google maps for over 200 inputs and outputs.

The name of the file used for simulation against the test bench is final\_project\_simulate.vhd.

The code of this file is slightly different from the original vhd file in the sense that It doesn't involve the delay of 1 sec as we assigned the signal "start1sec" to value "0".

Top level testing was done using project\_top\_module\_tb.vhd. Unit level testing was performed on the entity project and entity display was tested on FPGA board.



utilization\_1

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	Bonded IOB (106)	BUFGCTRL (32)
project_top_module	2502	2379	7	36	1
display_map (display)	105	111	0	0	0
digit (digit_display)	30	35	0	0	0
lat (seven_segment_dis...)	31	32	0	0	0
long (seven_segment_di...)	33	33	0	0	0
proj (project)	2358	2268	7	0	0

## UTILISATION REPORT