Indian Institute of Space Science and Technology

Thiruvananthapuram

AV 241 : VLSI LAB

Project Report

on

DIGITAL CLOCK

Submitted by

Sri Aditya Deevi

SC18B080

8 July 2020

Department of Avionics

.

Declaration

This project report titled **“DIGITAL CLOCK"** is a presentation of my original work. Wherever contributions of others are involved, every effort is made to indicate this clearly in the references. .

Sri Aditya Deevi
SC18B080
Roll No. 16
ECE (Avionics)
Indian Institute of Space science and Technology

Date: 8 July , 2020

.

Acknowledgement

First and foremost, I would like to express our deep and sincere gratitude to **Dr. Sheeba Rani**, our Professor, for giving me the opportunity to work on this wonderful project. With the help of her constant guidance and support, I was able to go ahead during this project.

Besides our professor, I would also like to pay our special regards to the TA's and Lab assistants of VLSI Lab IIST, without whose initial directions I would not have been able to finish this project.

I also sincerely thank everyone else who helped me in realizing this project successfully.

.

Abstract

This project is aimed at realizing a simple real-time digital clock with the following inbuilt facilities/modules/features :

- *24 hr Real-time Clock Display*
- *Alarm*
- *Timer*
- *Stopwatch*

For prototyping purpose, an FPGA(Field Programmable Gate Array) can be used for basic implementation and verification.

Users can set the current time into the clock in 24 hr format .

They will be provided a facility to set an alarm time which goes HIGH*, when the clock's time matches the alarm time. Also, the alarm can be reset at any point.*

They will be provided with a facility to set a timer which goes HIGH *when it becomes zero. Also, a stopwatch facility is provided.*

Push button switches, provided in the FPGA, can be used by the users to set the clock time,alarm,timer time and starting/stopping the stopwatch . The outputs of the real-time clock and stopwatch can be realized (for prototype) using Seven Segment Display provided in the FPGA. The output of the indicator can be connected to a buzzer (or) LED's provided in the FPGA. FPGA is planned to be programmed using Verilog code.

Contents

Abstract **II**

1 Basic Block Level Schematic **1**

2 Working Principle **2**
2.1 Real-time Clock . . . 2
2.2 Alarm . . . 3
2.3 Timer . . . 4
2.4 Stopwatch . . . 5

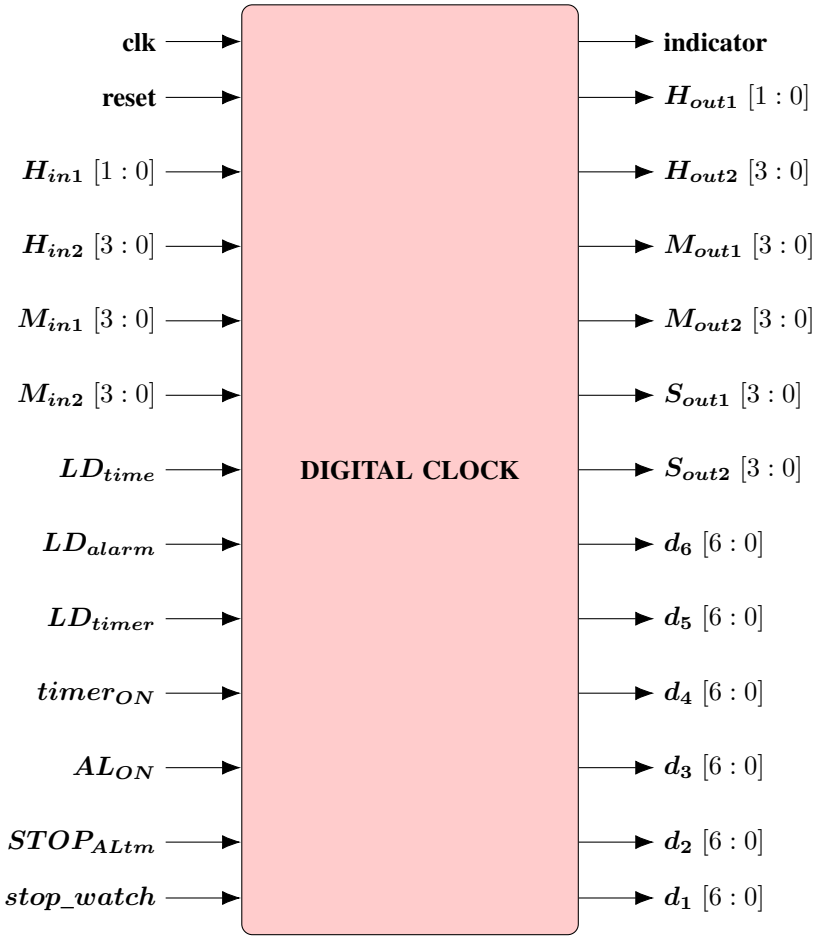
3 Verilog Code **6**

4 Simulation Results **19**
4.1 Reset and Load Time . . . 19
4.2 Real-Time Clock Operation . . . 20
4.3 Loading Alarm Time and Switching ON Alarm . . . 21
4.4 Stop Loading Alarm . . . 21
4.5 Checking Alarm Functionality . . . 22
4.6 Triggering “STOP ALARM" . . . 22
4.7 Checking “STOP ALARM" Functionality . . . 23
4.8 Switching OFF Alarm . . . 23
4.9 Loading Timer Time and Switching to Timer Functionality . . . 24
4.10 Stop Loading Timer and Timer STARTS . . . 24
4.11 Checking Timer Functionality . . . 25
4.12 Triggering “STOP TIMER" . . . 26
4.13 Checking “STOP TIMER" Functionality . . . 26
4.14 Switching OFF Timer . . . 27
4.15 Switching ON Stopwatch . . . 27
4.16 Checking Display-Switch (Real-Time –> Stopwatch) . . . 28
4.17 Starting the Stopwatch . . . 28
4.18 Checking the Stopwatch Functionality (START) . . . 29
4.19 Stopping the Stopwatch . . . 29
4.20 Checking the Stopwatch Functionality (STOP) . . . 30
4.21 Switching OFF Stopwatch . . . 30
4.22 Checking Display-Switch (Stopwatch –> Real-Time) . . . 31
4.23 Back to Real-Time Clock (Functionality Check) . . . 31

5 Concluding Remarks **32**
5.1 A note on Implementation . . . 32
5.2 Conclusion . . . 32

References **33**

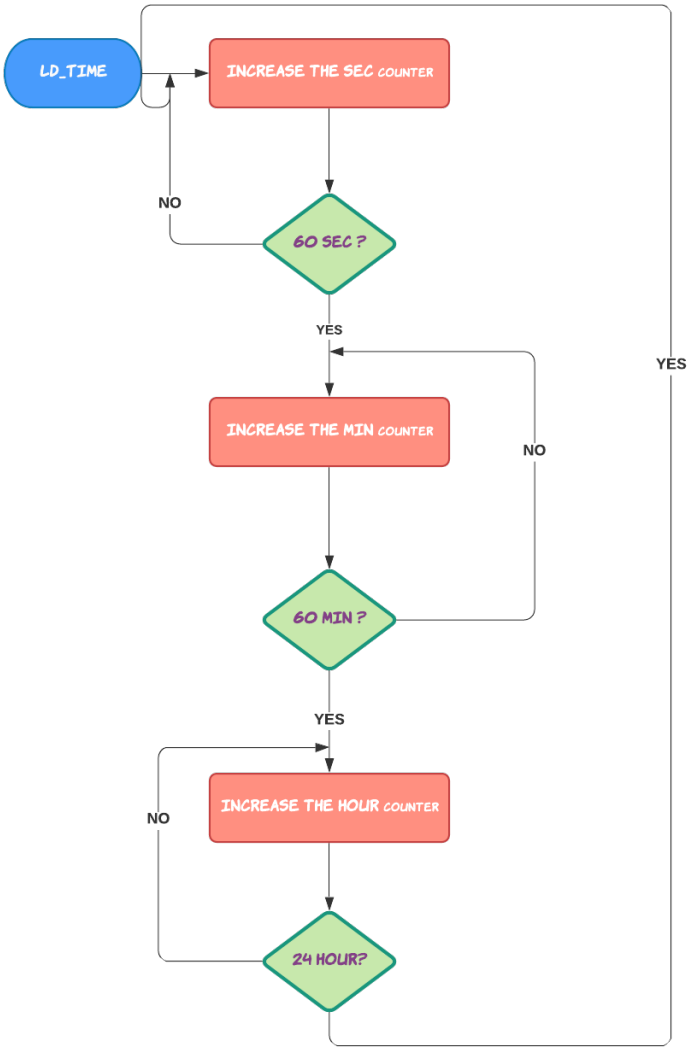
1 | Basic Block Level Schematic



2 | Working Principle

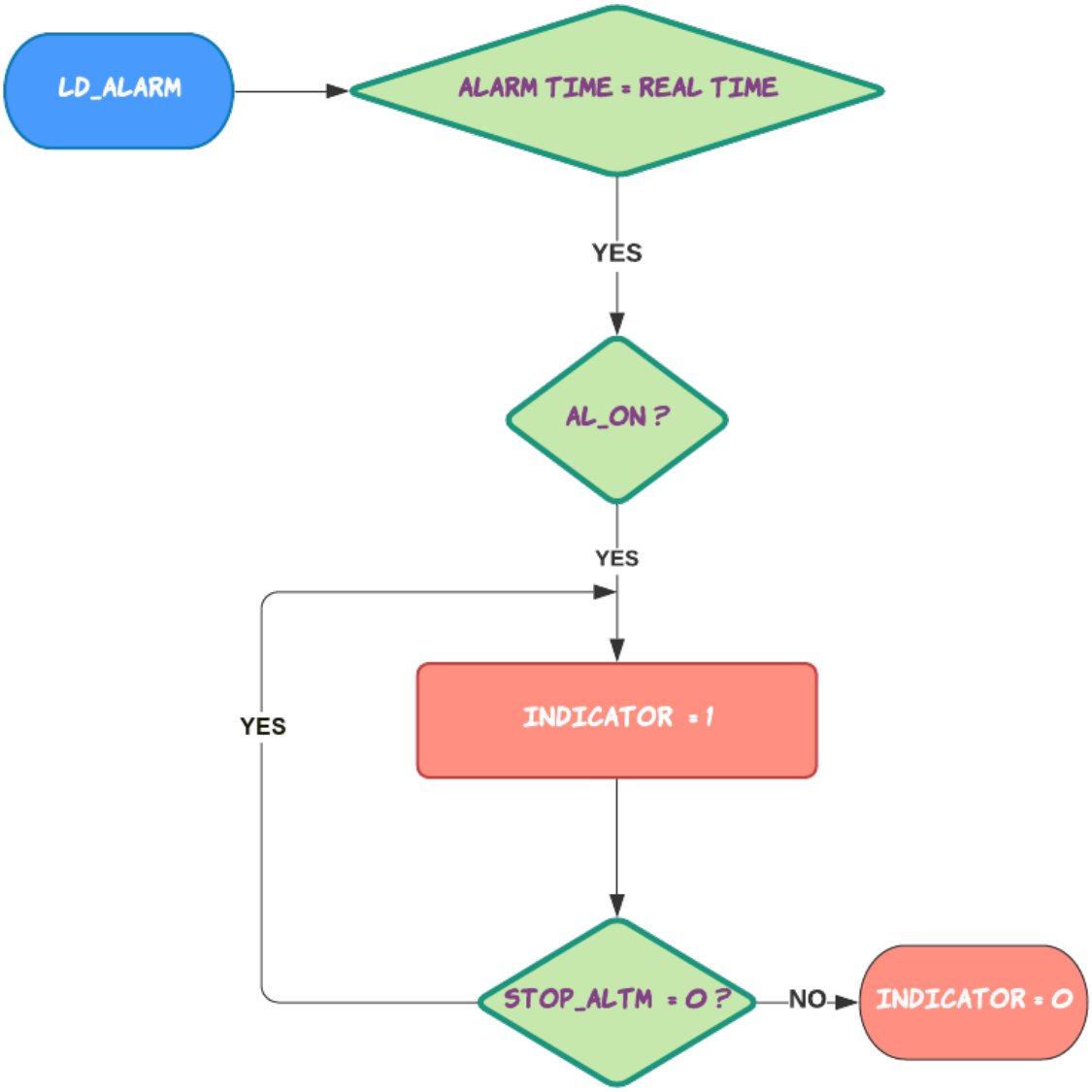
This section illustrates the working principles of various modules involved using simple *algorithmic* flowcharts.

2.1 Real-time Clock



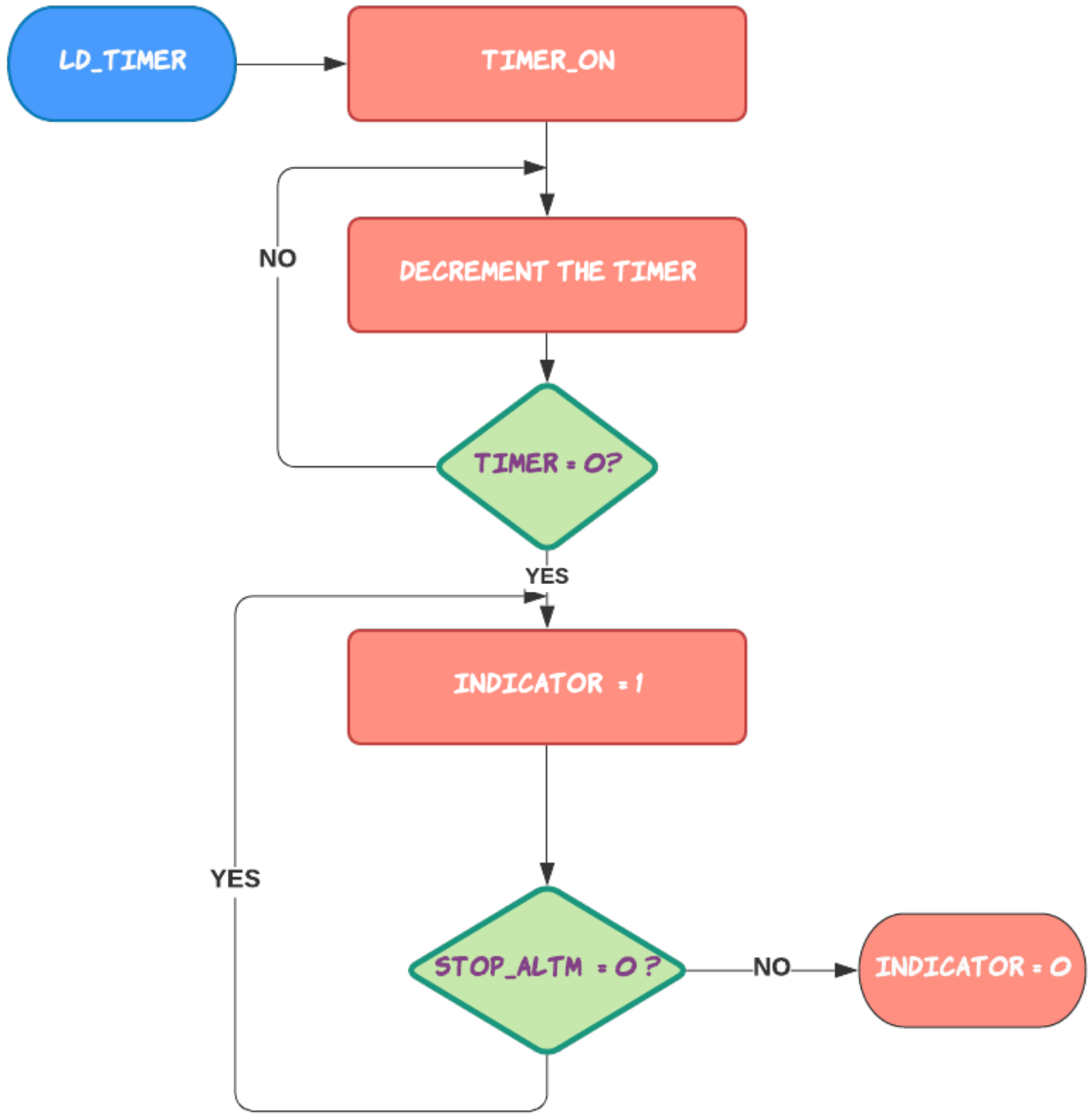
.

2.2 Alarm

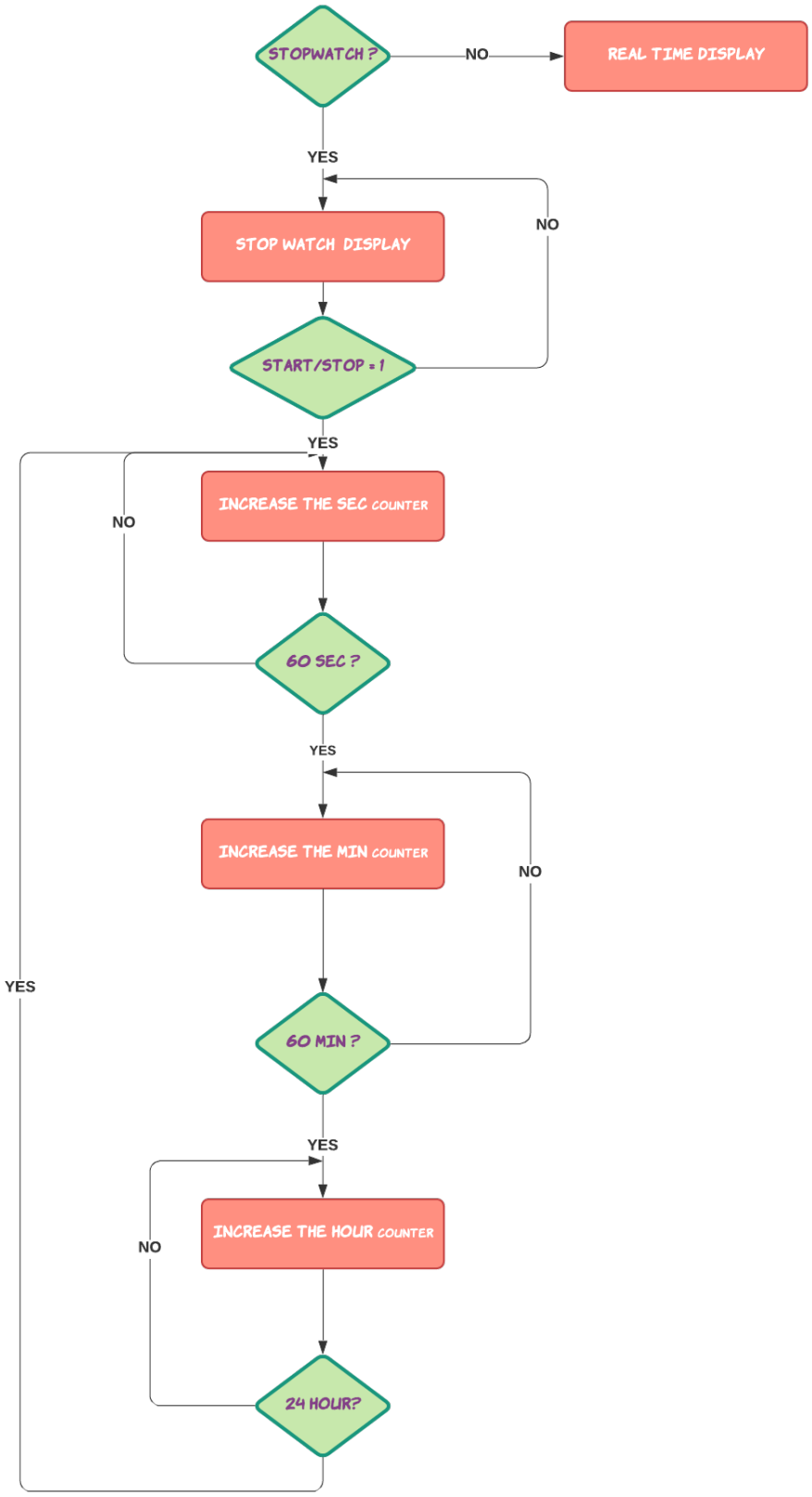

LD_ALARM
ALARM TIME = REAL TIME
YES
AL_ON ?
YES
YES
INDICATOR = 1
STOP_ALTM = 0 ?
NO
INDICATOR = 0

.

2.3 Timer


LD_TIMER
TIMER_ON
DECREMENT THE TIMER
NO
TIMER = 0?
YES
INDICATOR = 1
YES
STOP_ALTM = 0 ?
NO
INDICATOR = 0

2.4 Stopwatch



3 | Verilog Code

```
//==========================================================================================
/*------------------------------------------------MAIN MODULE------------------------------------------*/
//==========================================================================================

`timescale 1ns / 1ps

module digital_clock(

// IO Pins

input reset, //Active high reset pulse which sets the real time according to the input and sets the alarm
//timer's time to 0

input clk, // 10 Hz clock input

input [1:0] H_in1, // MSB of Hour - 2 bit input

input [3:0] H_in2, // LSB of Hour - 4 bit input

input [3:0] M_in1, // MSB of Minute - 4 bit input

input [3:0] M_in2, // LSB of Minute - 4 bit input

input LD_time, // Loads time to real-time clock

input LD_alarm, // Loads time to alarm clock

input LD_timer, // Loads required time to timer (max 15 min)

input timer_ON, // Switches on the timer

input AL_ON, // Switches on the alarm

input STOP_ALtm, // Switches off the timer or alarm output

input stop_watch, // Switches the display from real-time from stop-watch display and viceversa

input start_stop, // Starts the stop-watch and stops it and remains in the stopwatch screen until the stop

output reg indicator, // Gives alarm and timer's output

output [1:0] H_out1, // MSB of Hour - 2 bit output

output [3:0] H_out2, // LSB of Hour - 4 bit output

output [3:0] M_out1, // MSB of Minute - 4 bit output
```

```
47
48  output [3:0] M_out2, // LSB of Minute - 4 bit output
49
50  output [3:0] S_out1, // MSB of Second - 4 bit output
51
52  output [3:0] S_out2, // LSB of Second - 4 bit output
53
54  output [6:0] d_6,d_5,d_4,d_3,d_2,d_1 // 7 segment display for output
55
56
57  );
58
59  //Internal
60
61  reg clk_1s; // The working 1 second real_time clock
62
63  reg [3:0] temp1; // Loop variable for clock generation
64
65  reg [5:0] r_hour,r_min,r_sec; //temporary variables for real-time clock
66
67  reg [9:0] t_timer; // Timer counting variable
68
69  reg [1:0] c_hour1,a_hour1; // MSB of hour digit
70
71  reg [3:0] c_hour0,a_hour0; // LSB of hour digit
72
73  reg [3:0] c_min1,a_min1; // MSB of minute digit
74
75  reg [3:0] c_min0,a_min0; // LSB of minute digit
76
77  reg [3:0] c_sec1,a_sec1; // MSB of second digit
78
79  reg [3:0] c_sec0,a_sec0; // LSB of second digit
80
81  reg [5:0] h_hour,h_min,h_sec; //temporary variables for real-time clock
82
83
84
85  //----------------------------------------------------------------------------
86  //--------------------------Clock 1s GENERATION-----------------------
87  //----------------------------------------------------------------------------
88
89
90  always @(posedge clk or posedge reset)
91  begin
92
93  if(reset)
94  begin
95
96  temp1 <= 0;
97  clk_1s <=0;
98
99  end
100
101 else
102 begin
103
104 temp1 <= temp1+1;
105
```

```
106 if(temp1<=5) clk_1s <= 0;
107
108 else if(temp1>=10)
109 begin
110
111 clk_1s <= 1;
112 temp1 = 1;
113
114 end
115
116 else clk_1s <= 1;
117
118 end
119
120 end
121
122 //-------------------------------------------------------------------------
123 //-------------------------REAL TIME GENERATION---------------------
124 //-------------------------------------------------------------------------
125
126 always @(posedge clk_1s or posedge reset)
127 begin
128
129 if(reset)
130 begin
131
132 r_hour <= H_in1*10 + H_in2;
133 r_min <= M_in1*10 + M_in2;
134 r_sec <= 0;
135
136 end
137
138 else
139 begin
140
141 if(LD_time)
142 begin
143
144 r_hour <= H_in1*10 + H_in2;
145 r_min <= M_in1*10 + M_in2;
146 r_sec <= 0;
147
148 end
149
150 else
151 begin
152
153 r_sec <= r_sec + 1;
154
155 if(r_sec >= 59)
156 begin
157
158 r_min <= r_min + 1;
159 r_sec <= 0;
160
161 if(r_min >= 59)
162 begin
163
164 r_hour <= r_hour + 1;
```

```
r_min <= 0;

if(r_hour >= 24)
begin

r_hour <= 0;

end

end

end

end

end

end

//-----------------------------------------------------------------------
//-------------------------Timer Operation-------------------------------
//-----------------------------------------------------------------------

always @(posedge clk_1s or posedge reset)
begin

if(reset)
begin

indicator <= 0;

end

else
begin

if(LD_timer)
begin

t_timer <= M_in1*600 + M_in2*60;

end

else
begin

if(timer_ON)
begin

t_timer <= t_timer-1;

if(t_timer==0) indicator <= 1;

end

if(STOP_ALtm) indicator <= 0;

end

```

```
224 end
225
226 end
227
228 //-------------------------------------------------------------------------
229 //-------------------------Alarm Operation-------------------------------
230 //-------------------------------------------------------------------------
231
232 always @(posedge clk_1s or posedge reset)
233 begin
234
235 if(reset)
236 begin
237
238 indicator <= 0;
239
240 end
241
242 else
243 begin
244
245 if(LD_alarm)
246 begin
247
248 a_hour1 <= H_in1;
249 a_hour0 <= H_in2;
250 a_min1 <= M_in1;
251 a_min0 <= M_in2;
252 a_sec1 <= 4'b0000;
253 a_sec0 <= 4'b0000;
254
255 end
256
257 else
258 begin
259
260 if({a_hour1,a_hour0,a_min1,a_min0,a_sec1,a_sec0} == {c_hour1,c_hour0,c_min1,c_min0,c_sec1,c_sec0})
261
262 begin
263
264 if(AL_ON) indicator <= 1;
265
266 end
267
268 if(STOP_ALtm) indicator <= 0;
269
270 end
271
272 end
273
274 end
275
276 //-------------------------------------------------------------------------
277 //-------------------------Stopwatch Operation----------------------
278 //-------------------------------------------------------------------------
279
280 always @(posedge clk_1s or posedge reset)
281 begin
282
```

```
283 if(reset)
284 begin
285
286 h_hour <= 0;
287 h_min <= 0;
288 h_sec <= 0;
289
290 end
291
292 else
293 begin
294
295 if(stop_watch)
296 begin
297
298
299 if(start_stop)
300 begin
301
302 h_sec <= h_sec + 1;
303
304 if(r_sec >= 59)
305 begin
306
307 h_min <= h_min + 1;
308 h_sec <= 0;
309
310 if(r_min >= 59)
311 begin
312
313 h_hour <= h_hour + 1;
314 h_min <= 0;
315
316 if(h_hour >= 24)
317 begin
318
319 h_hour <= 0;
320
321 end
322
323 end
324
325 end
326
327 end
328
329 end
330
331 else
332 begin
333
334 h_hour <= 0;
335 h_min <= 0;
336 h_sec <= 0;
337
338 end
339
340 end
341
```

```
342 end
343
344 //------------------------------------------------------------------------
345 //--------------------Output Helper Function--------------------------
346 //------------------------------------------------------------------------
347 function [3:0] mod_10(input [5:0] number);
348 begin
349
350 if(number>=50)
351 begin
352
353 mod_10 = 5;
354
355 end
356
357 else if(number>=40)
358 begin
359
360 mod_10 = 4;
361
362 end
363
364 else if(number>=30)
365 begin
366
367 mod_10 = 3;
368
369 end
370
371 else if(number>=20)
372 begin
373
374 mod_10 = 2;
375
376 end
377 else if(number>=10)
378 begin
379
380 mod_10 = 1;
381
382 end
383
384 else
385 begin
386
387 mod_10 = 0;
388
389 end
390
391 end
392
393 endfunction
394
395 //------------------------------------------------------------------------
396 //--------------------Output Control for Clock--------------------------
397 //------------------------------------------------------------------------
398
399 always @(*)
400 begin
```

```
401
402 if(stop_watch)
403 begin
404
405 if(h_hour>=20)
406 begin
407
408 c_hour1 = 2;
409
410 end
411
412 else
413 begin
414
415 if(h_hour >=10) c_hour1  = 1;
416
417 else c_hour1 = 0;
418
419 end
420
421 c_hour0 = h_hour - 10*c_hour1;
422 c_min1 = mod_10(h_min);
423 c_min0 = h_min - 10*c_min1;
424 c_sec1 = mod_10(h_sec);
425 c_sec0 = h_sec - 10*c_sec1;
426
427 end
428
429 else
430 begin
431
432 if(r_hour>=20)
433 begin
434
435 c_hour1 = 2;
436
437 end
438
439 else
440 begin
441
442 if(r_hour >=10) c_hour1  = 1;
443
444 else c_hour1 = 0;
445
446 end
447
448 c_hour0 = r_hour - 10*c_hour1;
449 c_min1 = mod_10(r_min);
450 c_min0 = r_min - 10*c_min1;
451 c_sec1 = mod_10(r_sec);
452 c_sec0 = r_sec - 10*c_sec1;
453
454 end
455
456 end
457
458 assign H_out1 = c_hour1;
459 assign H_out2 = c_hour0;
```

```
assign M_out1 = c_min1;
assign M_out2 = c_min0;
assign S_out1 = c_sec1;
assign S_out2 = c_sec0;

//-------------------------------------------------------------------------
//-------------------------Clock Display-----------------------------------
//-------------------------------------------------------------------------


segment7_decoder s1(d_6,{{1'b0,1'b0},c_hour1});
segment7_decoder s2(d_5,c_hour0);
segment7_decoder s3(d_4,c_min1);
segment7_decoder s4(d_3,c_min0);
segment7_decoder s5(d_2,c_sec1);
segment7_decoder s6(d_1,c_sec0);



endmodule



//=================================================================================================
/*--------------------------SEVEN SEGMENT DISPLAY---------------------------------------*/
//=================================================================================================

`timescale 1ns / 1ps

module segment7_decoder(seg7,bcd);

input [3:0] bcd;

output reg [6:0] seg7;

always @(bcd)

case(bcd)

0 : seg7 <= 7'b1111110;
1 : seg7 <= 7'b0110000;
2 : seg7 <= 7'b1101101;
3 : seg7 <= 7'b1111001;
4 : seg7 <= 7'b0110011;
5 : seg7 <= 7'b1011011;
6 : seg7 <= 7'b1011111;
7 : seg7 <= 7'b1110000;
8 : seg7 <= 7'b1111111;
9 : seg7 <= 7'b1111011;

default : seg7 <= 7'b0000000;

endcase

endmodule

//=================================================================================================
/*-------------------------TEST BENCH-------------------------------------------------------*/
//=================================================================================================
```

```
519
520 `timescale 1ns / 1ps
521
522 module teest;
523
524 // Inputs
525 reg reset;
526 reg clk;
527 reg [1:0] H_in1;
528 reg [3:0] H_in2;
529 reg [3:0] M_in1;
530 reg [3:0] M_in2;
531 reg LD_time;
532 reg LD_alarm;
533 reg LD_timer;
534 reg timer_ON;
535 reg AL_ON;
536 reg STOP_ALtm;
537 reg stop_watch;
538 reg start_stop;
539
540 // Outputs
541 wire indicator;
542 wire [1:0] H_out1;
543 wire [3:0] H_out2;
544 wire [3:0] M_out1;
545 wire [3:0] M_out2;
546 wire [3:0] S_out1;
547 wire [3:0] S_out2;
548 wire [6:0] d_6;
549 wire [6:0] d_5;
550 wire [6:0] d_4;
551 wire [6:0] d_3;
552 wire [6:0] d_2;
553 wire [6:0] d_1;
554
555
556 // Instantiate the Unit Under Test (UUT)
557
558 digital_clock uut (
559 .reset(reset),
560 .clk(clk),
561 .H_in1(H_in1),
562 .H_in2(H_in2),
563 .M_in1(M_in1),
564 .M_in2(M_in2),
565 .LD_time(LD_time),
566 .LD_alarm(LD_alarm),
567 .LD_timer(LD_timer),
568 .timer_ON(timer_ON),
569 .AL_ON(AL_ON),
570 .STOP_ALtm(STOP_ALtm),
571 .stop_watch(stop_watch),
572 .start_stop(start_stop),
573 .indicator(indicator),
574 .H_out1(H_out1),
575 .H_out2(H_out2),
576 .M_out1(M_out1),
577 .M_out2(M_out2),
```

```
.S_out1(S_out1),
.S_out2(S_out2),
.d_6(d_6),
.d_5(d_5),
.d_4(d_4),
.d_3(d_3),
.d_2(d_2),
.d_1(d_1)

);


initial begin

clk = 0;
forever #50 clk = ~clk;

end

initial begin
// Initialize Inputs

// -------------------------RESET AND LOAD TIME-----------------------

reset = 1;
H_in1 = 1;
H_in2 = 0;
M_in1 = 3;
M_in2 = 0;
LD_time = 1;
LD_alarm = 0;
LD_timer = 0;
timer_ON = 0;
AL_ON = 0;
STOP_ALtm = 0;
stop_watch = 0;
start_stop = 0;

// Wait 100 ns for global reset to finish
#100;
#2


reset = 0;
LD_time = 0;

#100

// ----------LOAD ALARM TIME AND ALARM ON----------------------

LD_alarm = 1;
AL_ON = 1;
H_in1 = 1;
H_in2 = 0;
M_in1 = 4;
M_in2 = 0;

#1000

```

```
LD_alarm = 0;

#1000000

// ---------------TRIGGER STOP ALARM-------------------------

STOP_ALtm = 1;


// ----------------ALARM OFF--------------------------
#10000

STOP_ALtm = 0;
AL_ON = 0;

// ----------LOAD TIMER TIME AND TIMER ON----------------------
#300000

LD_timer = 1;
timer_ON = 1;
H_in1 = 0;
H_in2 = 0;
M_in1 = 0;
M_in2 = 1;


#10000

LD_timer = 0;

#200000

// ---------------TRIGGER STOP TIMER--------------------------

STOP_ALtm = 1;

#10000

// ------------------TIMER OFF--------------------------
STOP_ALtm = 0;
timer_ON = 0;

// -----------------SWITCH DISPLAY TO STOPWATCH -------------------

stop_watch = 1;
#100000

//---------------------START STOPWATCH----------------------------
start_stop = 1;

#200000
//---------------------STOP STOPWATCH----------------------------
start_stop = 0;

#100000
// -----------------SWITCH DISPLAY TO REAL TIME ------------------
stop_watch = 0;

// Add stimulus here
```

```
696
697 end
698
699 endmodule
700
701 //=============================================================================================
702 //=============================================================================================
```

4 | Simulation Results

This section displays a series of simulation results of the digital clock, emphasizing the various features and their functionality with inferences, describing a particular schedule as per the testbench.

REAL-TIME CLOCK

4.1 Reset and Load Time

The reset is made high and time is set to **10:30** (AM)*.

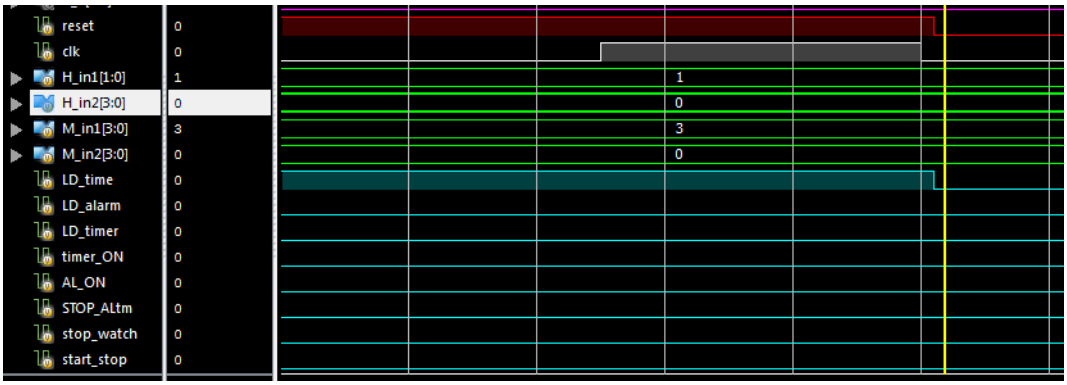


Figure 4.1: Input Perspective

The output is set to the input time and clock operation starts. The seven segment display is also visible.

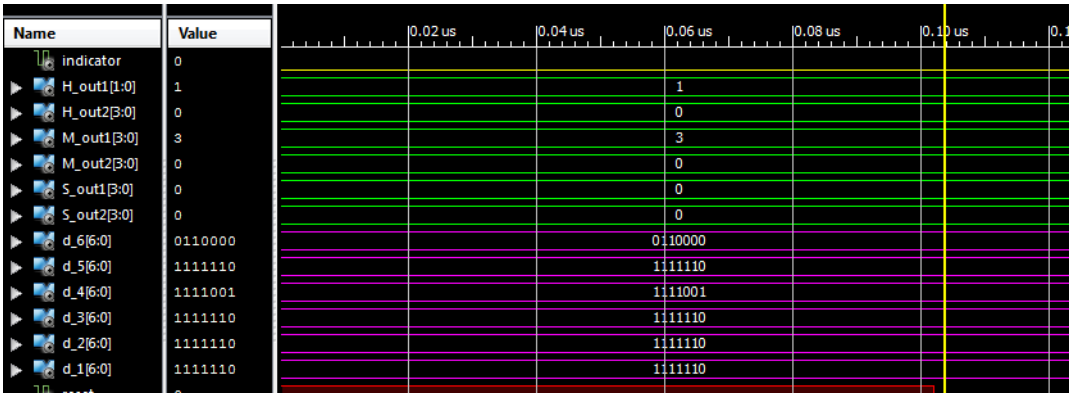


Figure 4.2: Output Perspective

*Please note that this a 24 hr Clock Display

A Note on the Test Bench

For the purpose of simulation we consider an external clock of *10 MHz*, from which we deduce that the internal clock works at *1 MHz*. This implies that one second in real-time is equal to 1 μs (or) one microsecond in the testbench*.

Also note that, during deployment, this external clock can be replaced by a *10 Hz* clock, setting the internal clock to a frequency of *1 Hz*†

4.2 Real-Time Clock Operation

Please note that the real-time clock is working as expected with proper duration as can be seen from the output shown below. Also, the seven segment display is also working as expected.

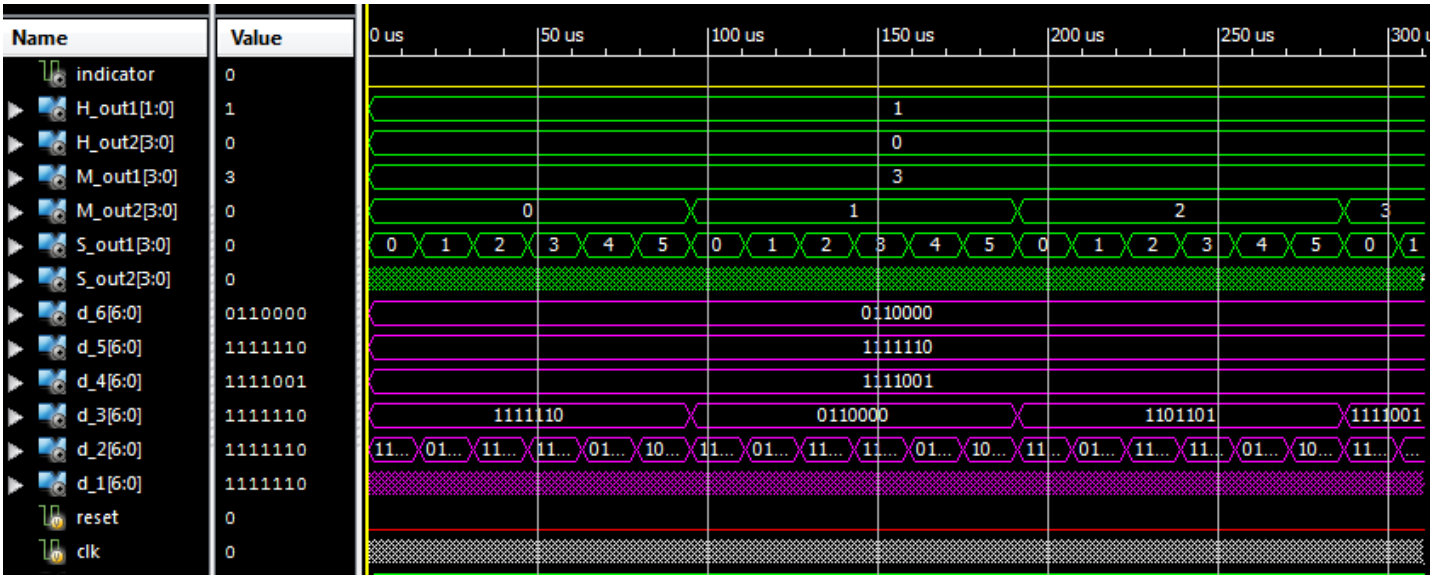


Figure 4.3: Output Perspective

*For simulation purposes

†Thereby satisfying real-time constraints.

****ALARM****

4.3 Loading Alarm Time and Switching ON Alarm

Please note that the Alarm time is set to **10:40** (AM)[†]. Also, the alarm has been switched ON.

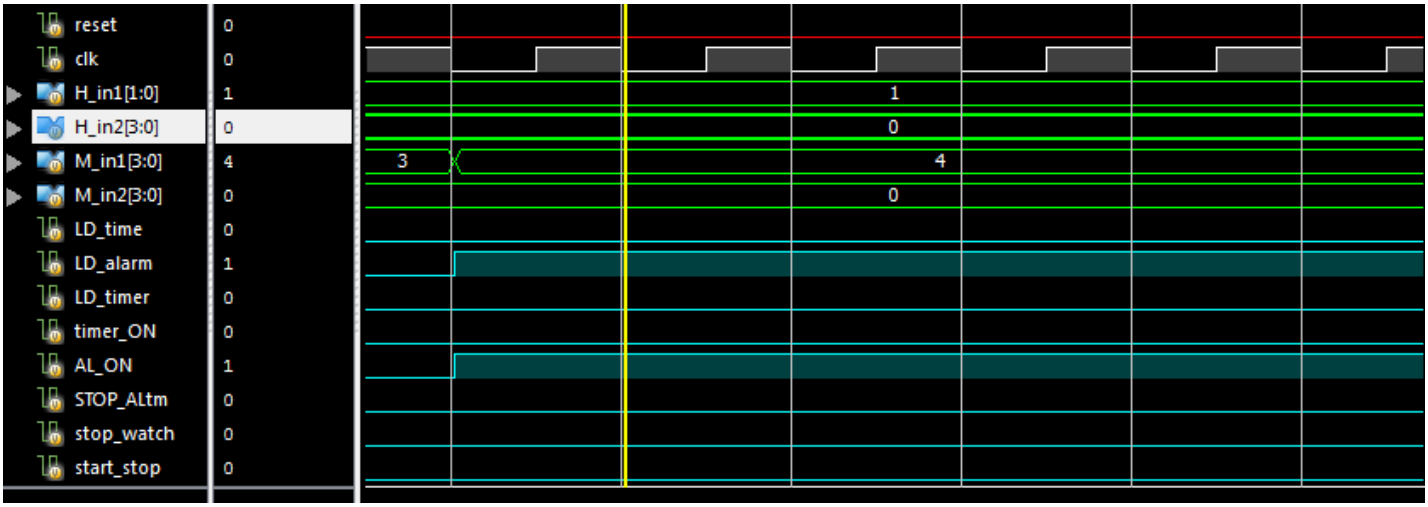


Figure 4.4: Input Perspective

4.4 Stop Loading Alarm

The loading process of the alarm operation is complete. Please note that, in the output perspective the real-time clock operation process continues unhindered.

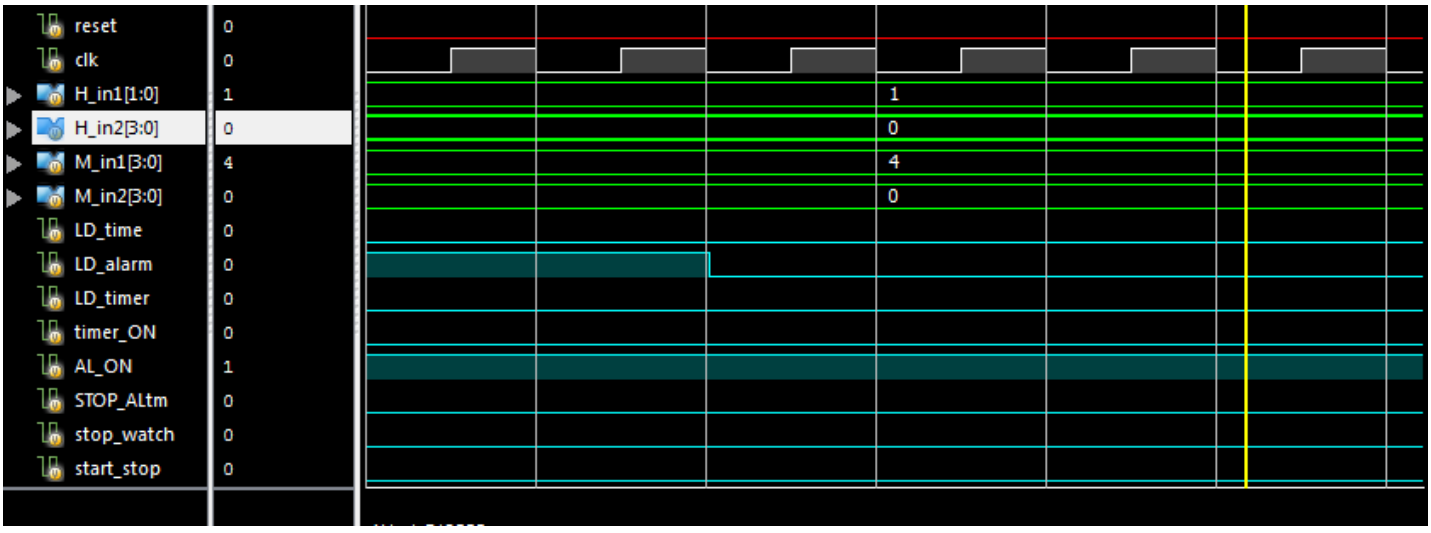


Figure 4.5: Input Perspective

[†]For demonstration Purposes.

4.5 Checking Alarm Functionality

We can clearly see that the output of the indicator goes HIGH as soon as the set time is achieved ***(10:40** AM)*.

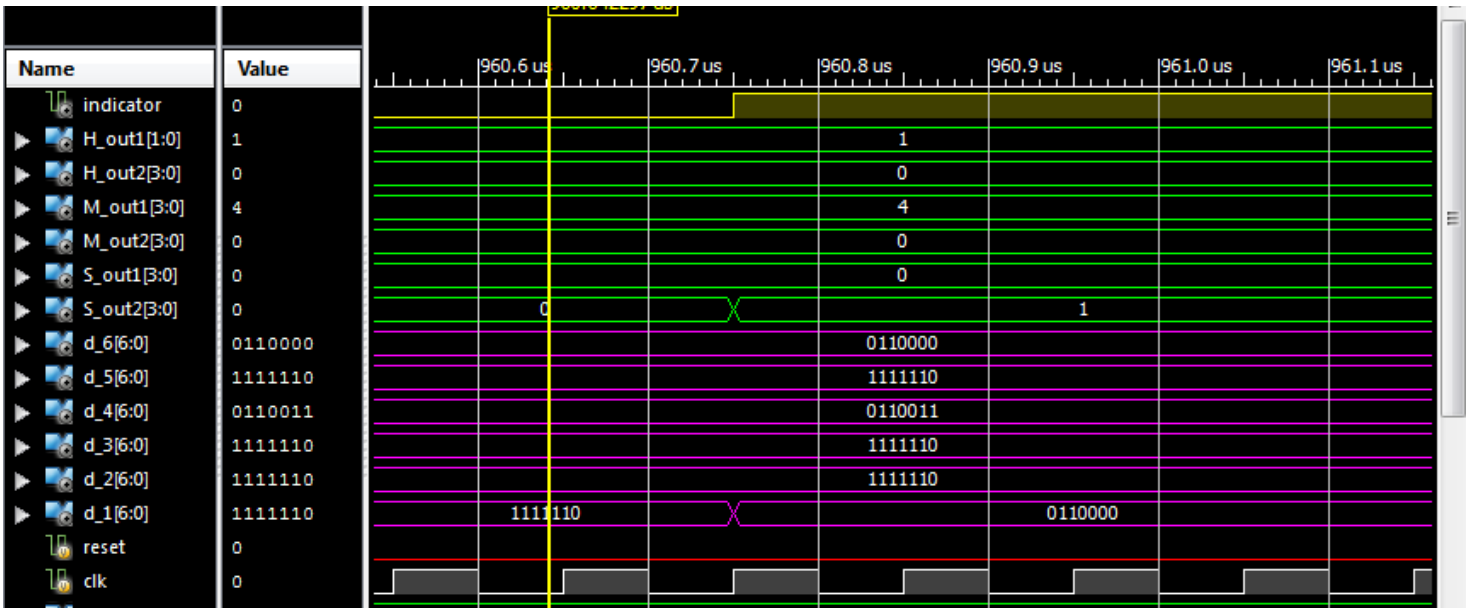


Figure 4.6: Output Perspective

4.6 Triggering “STOP ALARM"

Please note from the following figure that the Stop_ALtm signal is made HIGH after some duration after which the indicator goes HIGH.

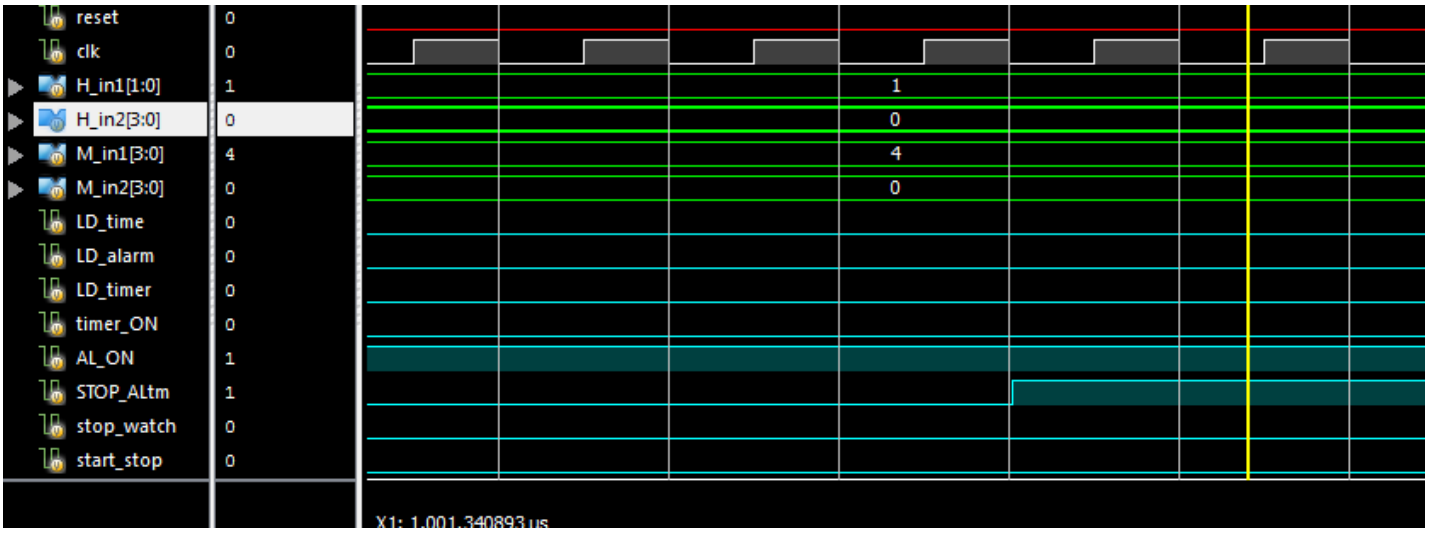


Figure 4.7: Input Perspective

4.7 Checking “STOP ALARM" Functionality

As we can see, as expected the output of the indicator goes LOW as an effect of the previous action.

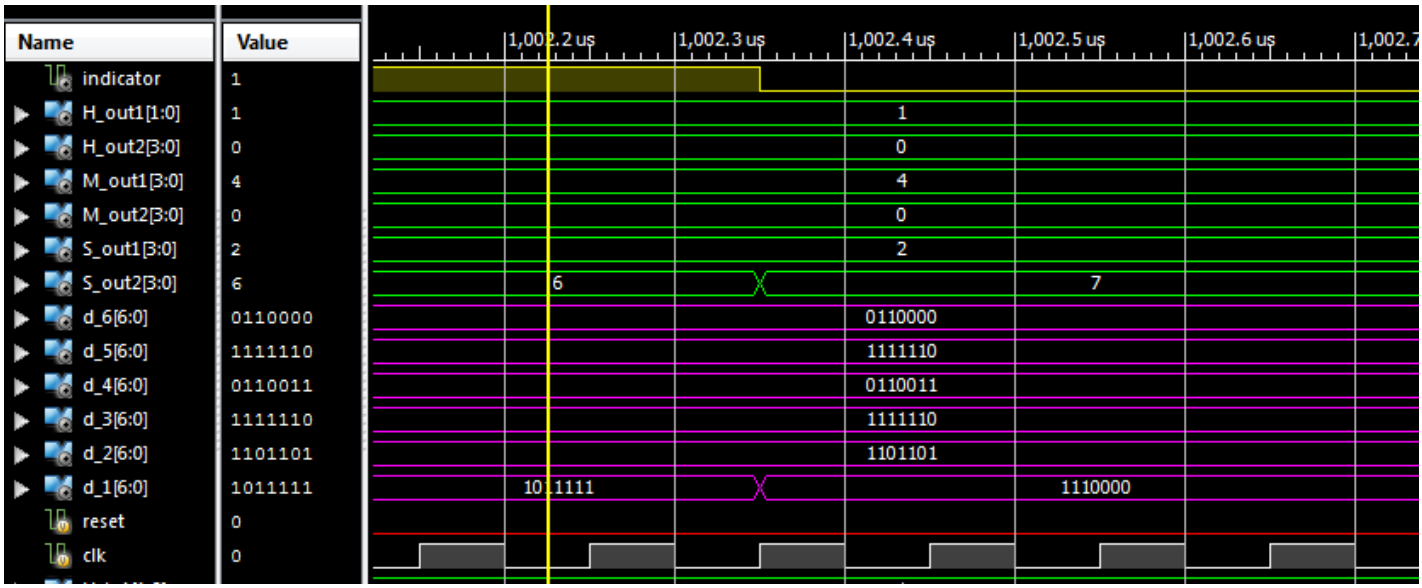


Figure 4.8: Output Perspective

4.8 Switching OFF Alarm

Please note that the Alarm functionality is switched OFF in order to demonstrate the next functionality.

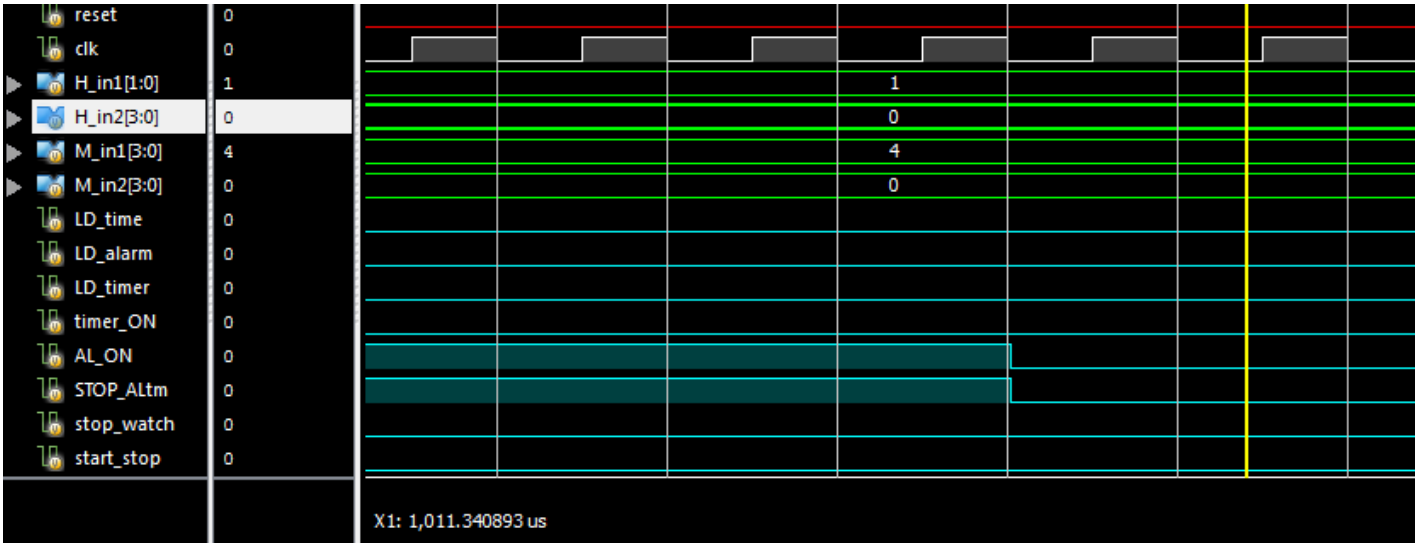


Figure 4.9: Input Perspective

****TIMER****

4.9 Loading Timer Time and Switching to Timer Functionality

Please observe that the timer is switched ON and the timer time is being set to a duration of 1 minute.

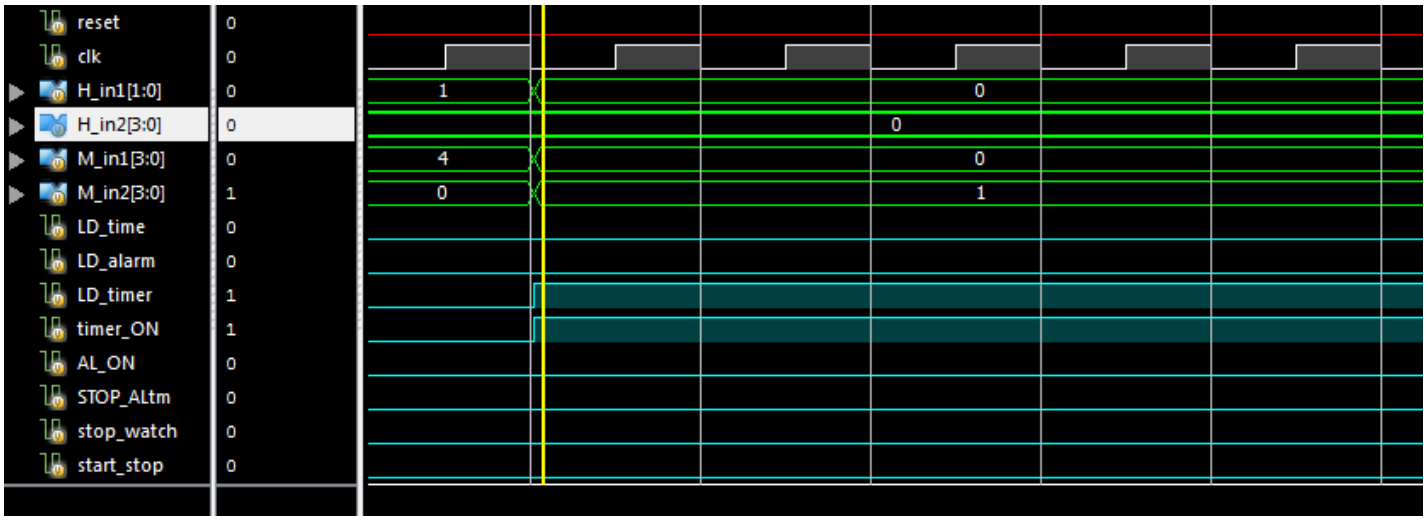


Figure 4.10: Input Perspective

4.10 Stop Loading Timer and Timer STARTS

Please note carefully that the timer has been stopped loading and the timer starts its 1 min (set in the previous step) duration. [‡]

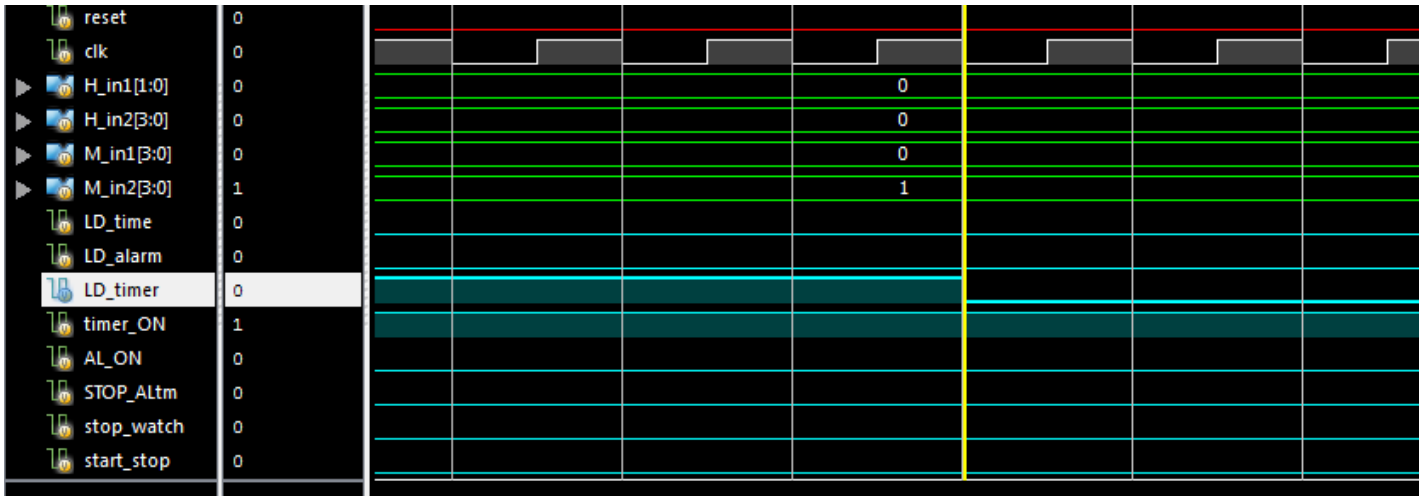


Figure 4.11: Input Perspective

[‡]Note that the Real-Time Clock operation is carried out unhindered.

The timestamp at which timer starts its operation is **10:43:46** (AM).

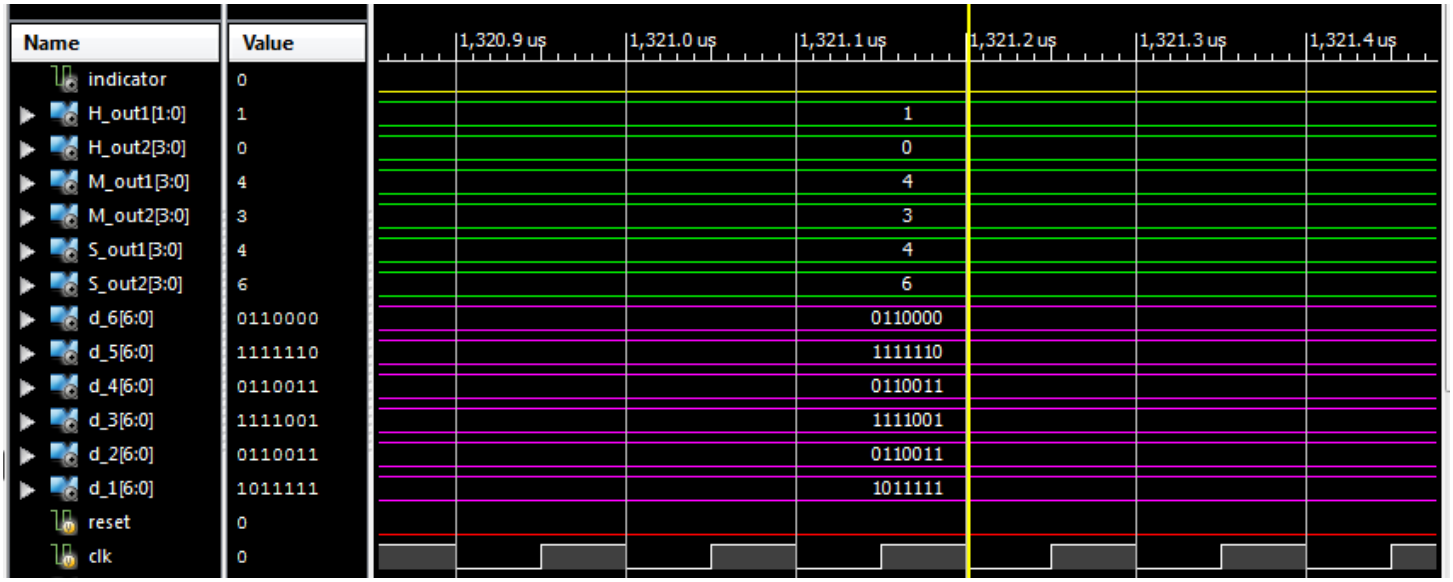


Figure 4.12: Output Perspective

4.11 Checking Timer Functionality

Please note that the output of the indicator goes HIGH as it completes **10:44:46** (AM) *i.e.* exactly 1 min (as set) after **10:43:46** (AM).

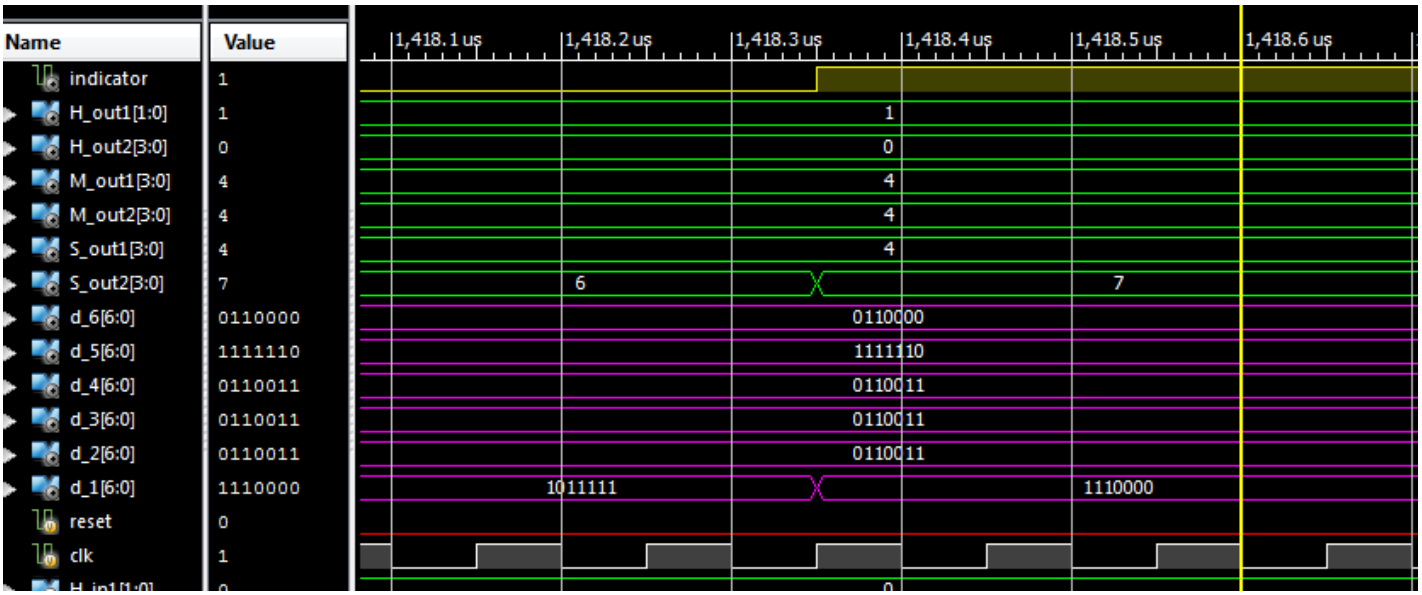


Figure 4.13: Output Perspective

4.12 Triggering “STOP TIMER"

Please note from the following figure that the Stop_ALtm signal is made HIGH after some duration after which the indicator goes HIGH.

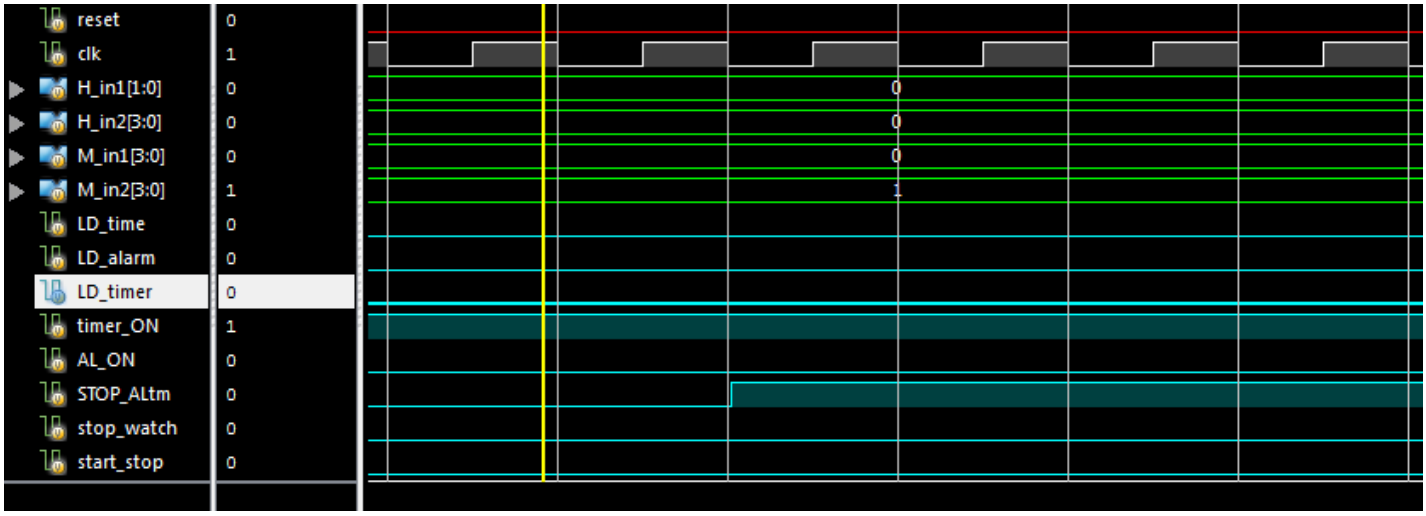


Figure 4.14: Input Perspective

4.13 Checking “STOP TIMER" Functionality

As we can see, as expected the output of the indicator goes LOW as an effect of the previous action.

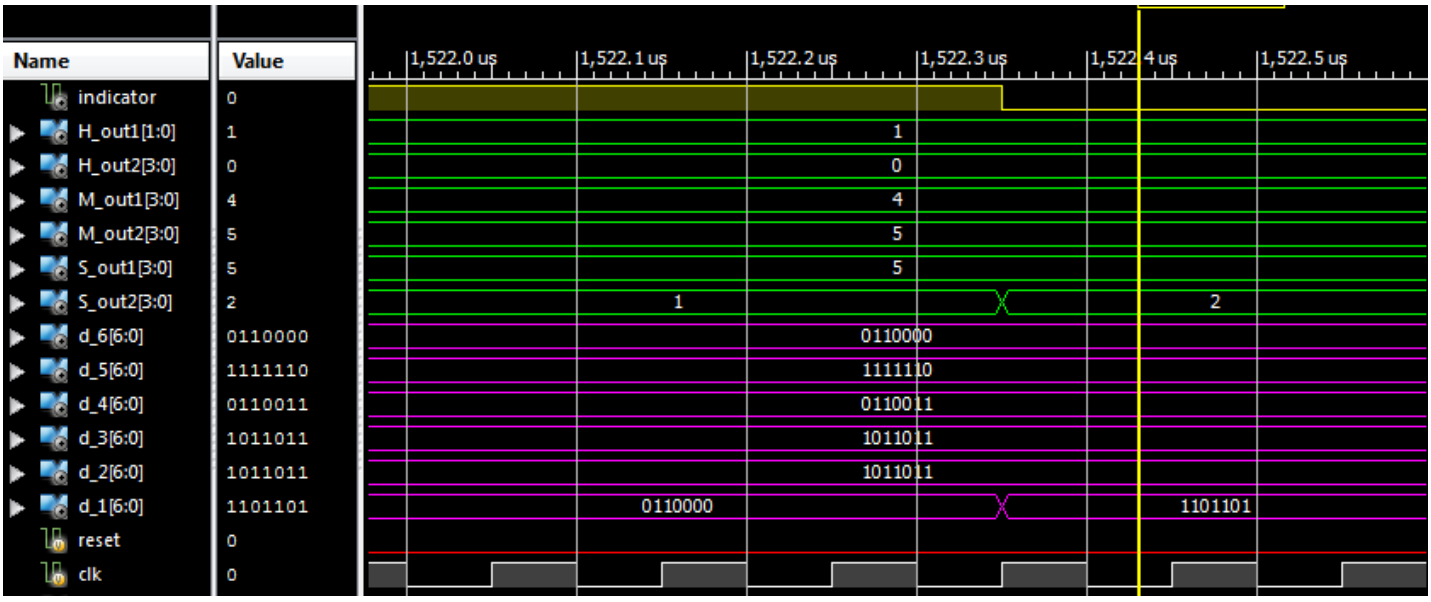


Figure 4.15: Output Perspective

4.14 Switching OFF Timer

Please note that the Timer functionality is switched OFF in order to demonstrate the next functionality.

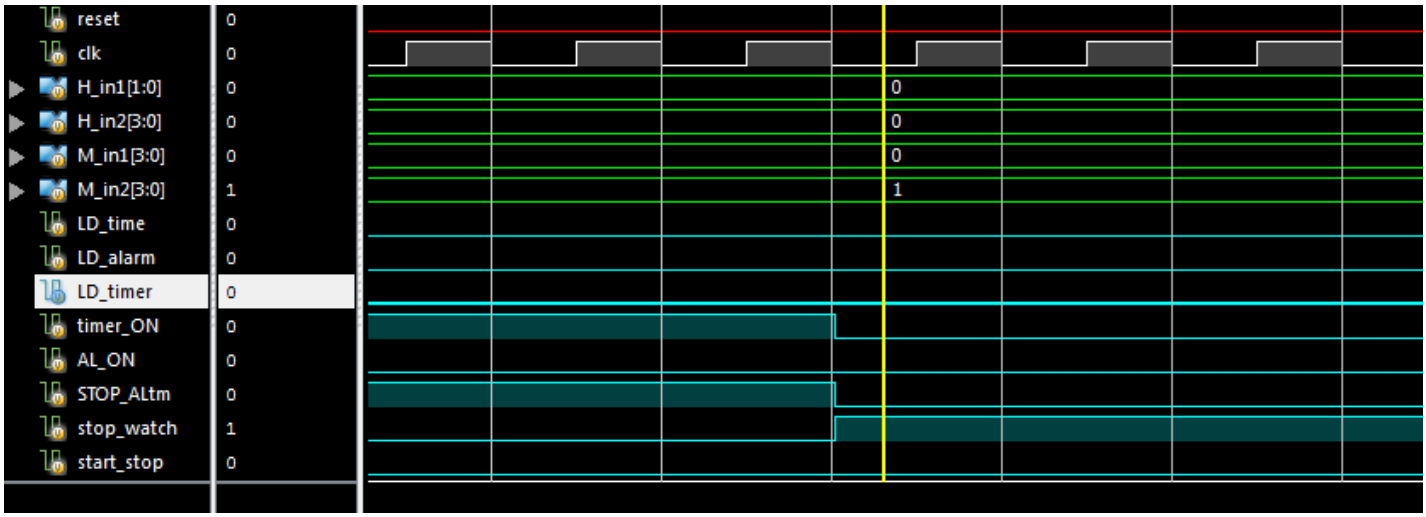


Figure 4.16: Input Perspective

STOPWATCH

4.15 Switching ON Stopwatch

Please note that the *stop_watch* is made HIGH , whose function is switch the display from Real-Time to Stopwatch Display and vice versa. [§]

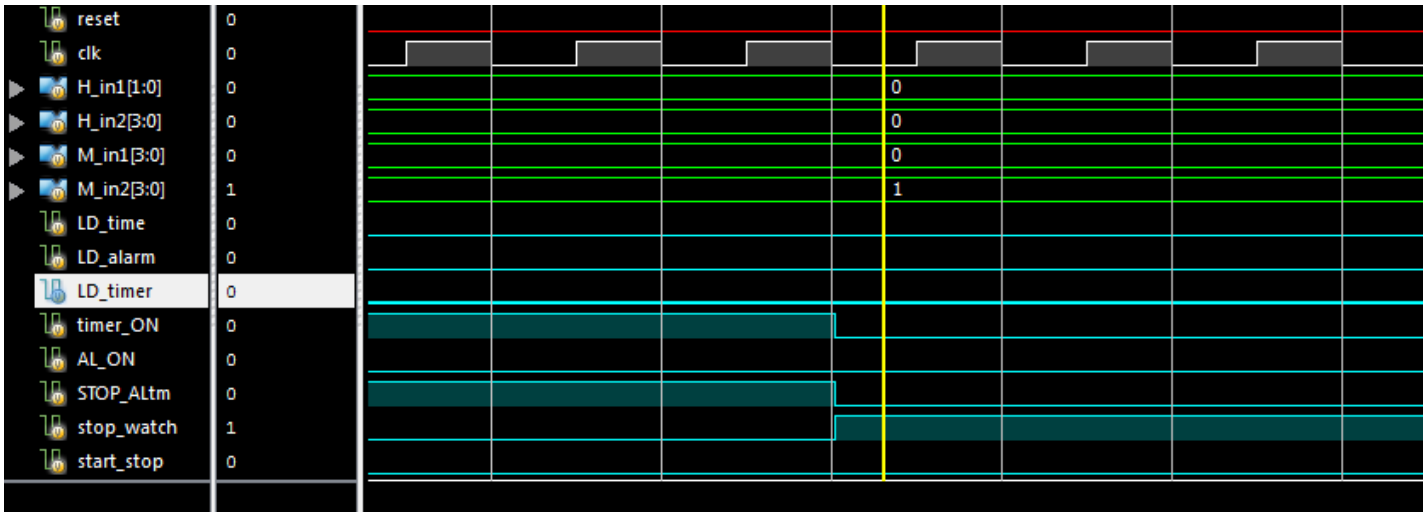


Figure 4.17: Input Perspective

[§]When *stop_watch* is HIGH, StopWatch Display
When *stop_watch* is LOW , Real-Time Display

4.16 Checking Display-Switch (Real-Time –> Stopwatch)

Please note that as expected the display of the clock has been switched from Real-Time Display to Stopwatch Display (which is initialized to all 0's).¶

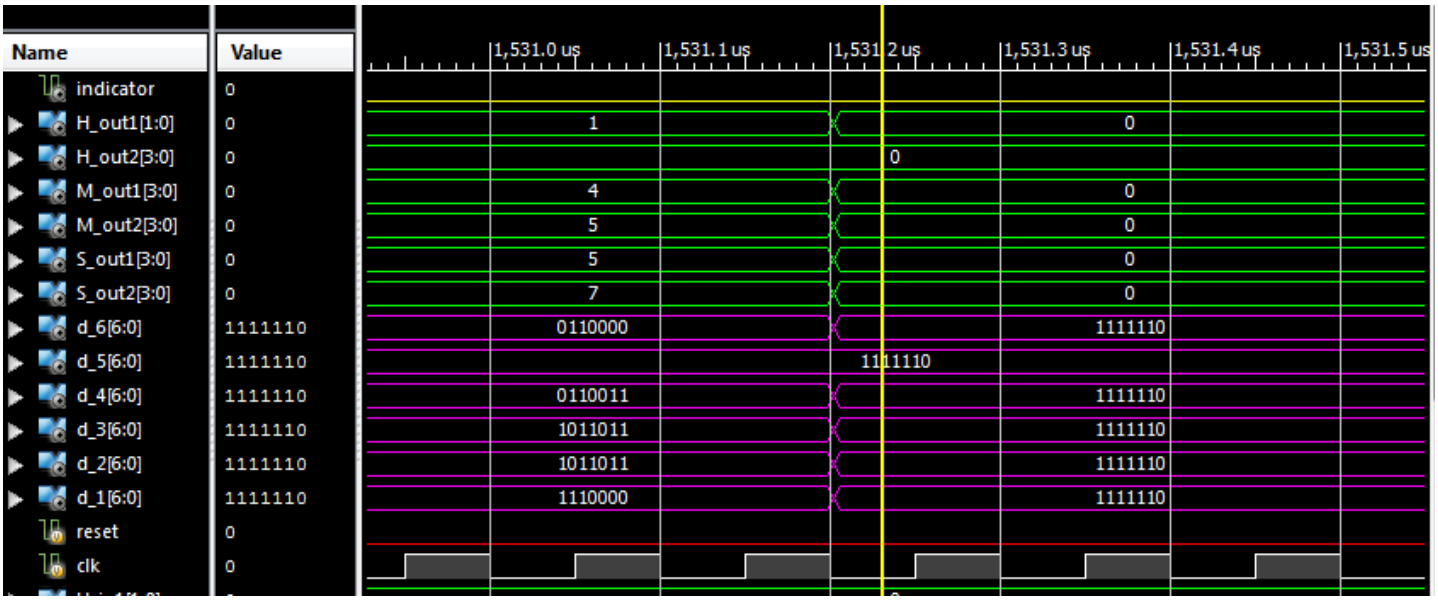


Figure 4.18: Input Perspective

4.17 Starting the Stopwatch

As we can clearly see the *start_stop* function is made HIGH indicating that the stopwatch has started functioning.

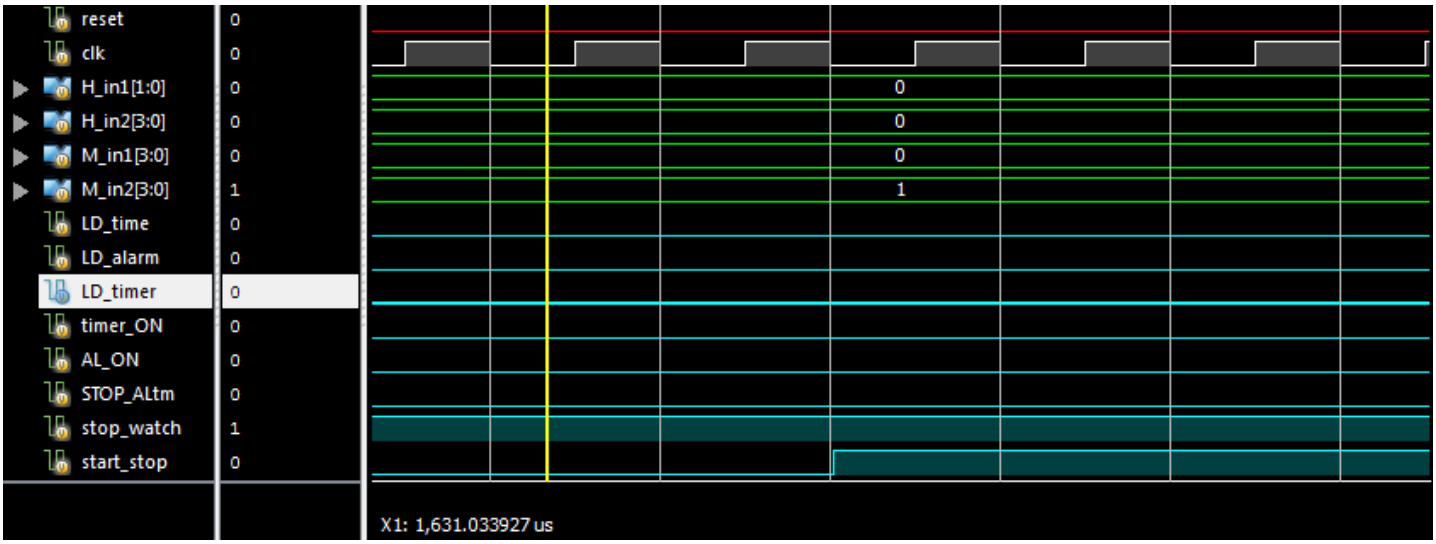


Figure 4.19: Input Perspective

¶**Note that even though the display switches the real-time clock operation is carried out unhindered and when we switch back to real-time display it starts displaying from that point in time, taking in account the spent time during the display-switch**

4.18 Checking the Stopwatch Functionality (START)

As expected the stopwatch has started functioning.

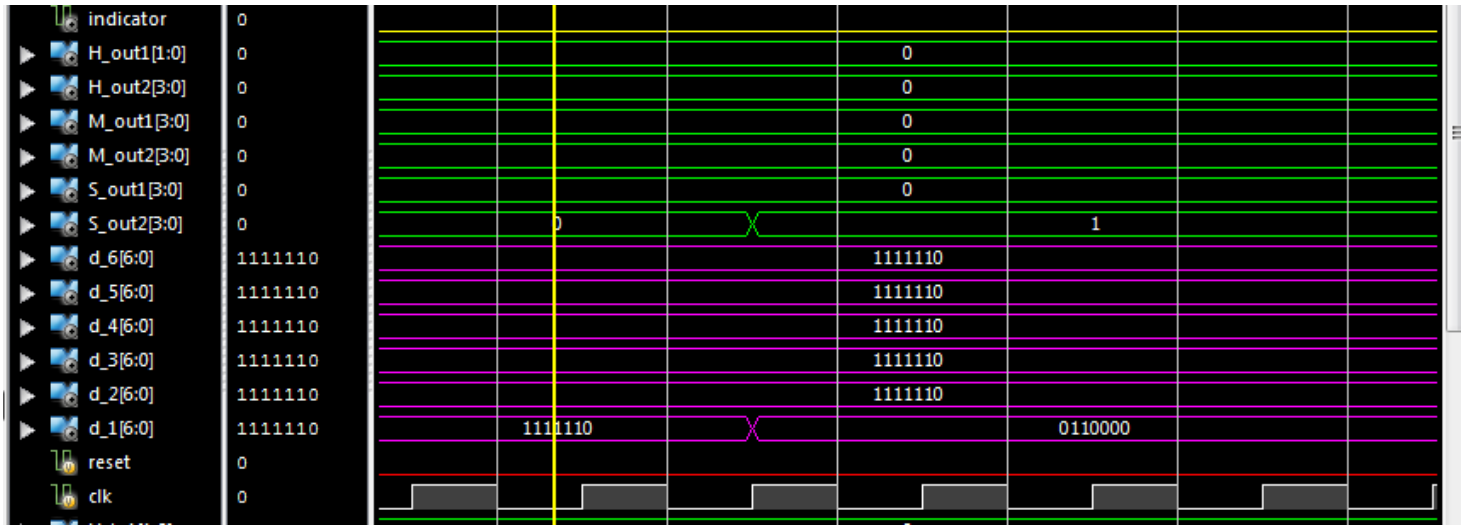


Figure 4.20: Output Perspective

4.19 Stopping the Stopwatch

As we can clearly see the *start_stop* function is made LOW indicating that the stopwatch has to be stopped functioning.

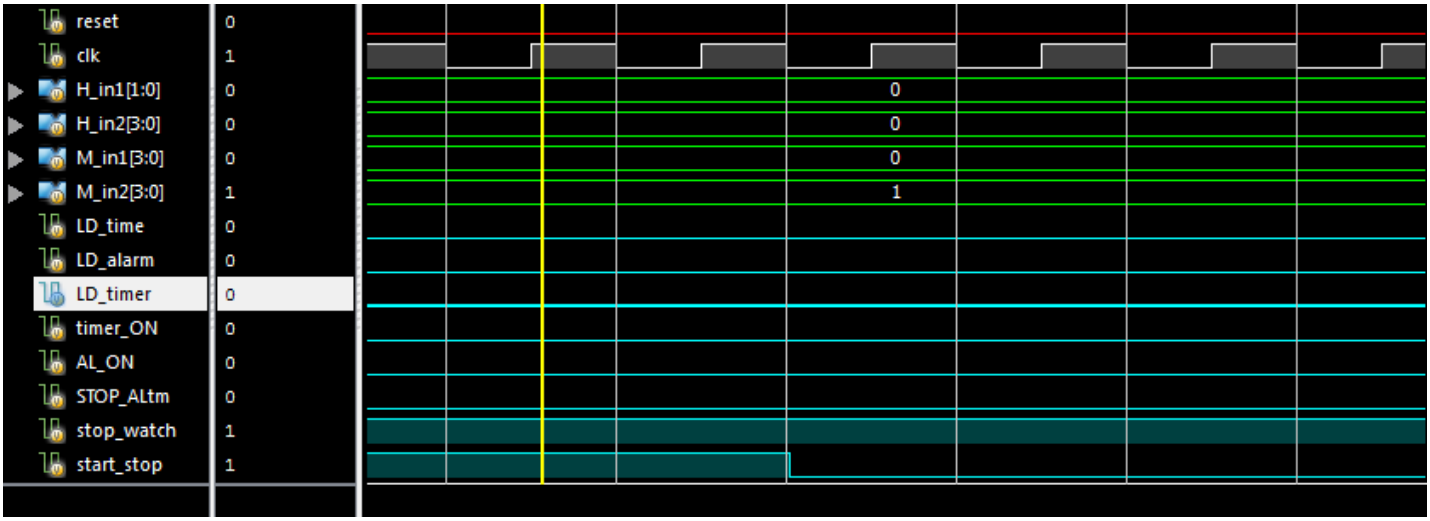


Figure 4.21: Input Perspective

4.20 Checking the Stopwatch Functionality (STOP)

As expected the stopwatch has stopped functioning but the display remains frozen to the stopwatch screen time at which point the stopwatch was stopped.

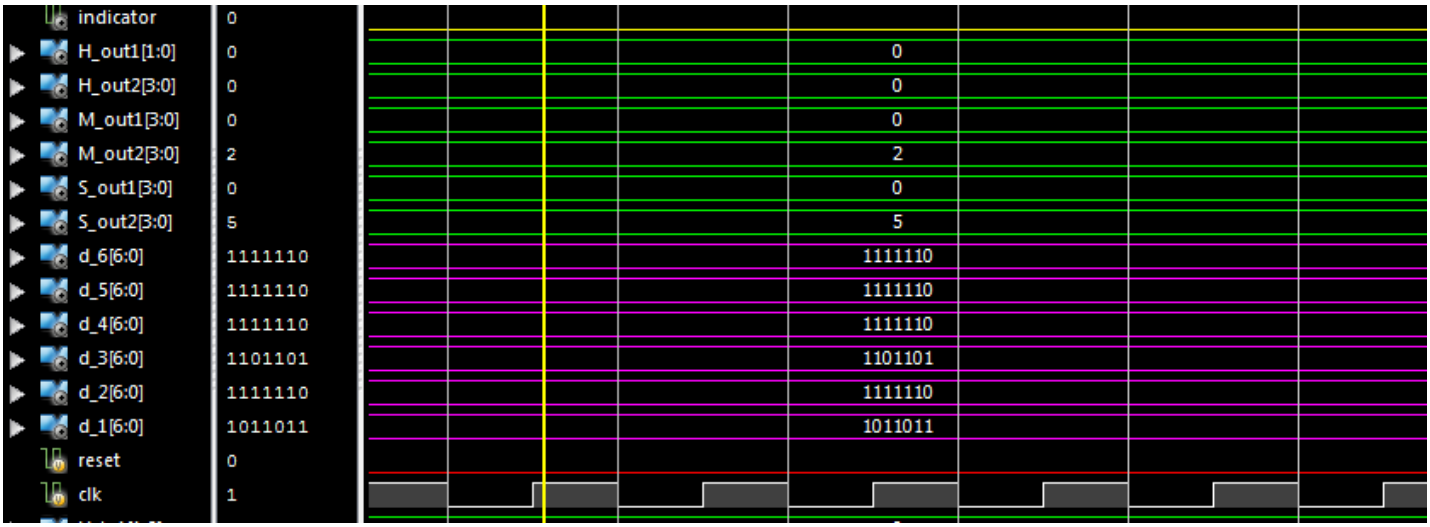


Figure 4.22: Output Perspective

4.21 Switching OFF Stopwatch

Please note that the *stop_watch* is made LOW , whose function is switch the display from Real-Time to Stopwatch Display and vice versa. ‖

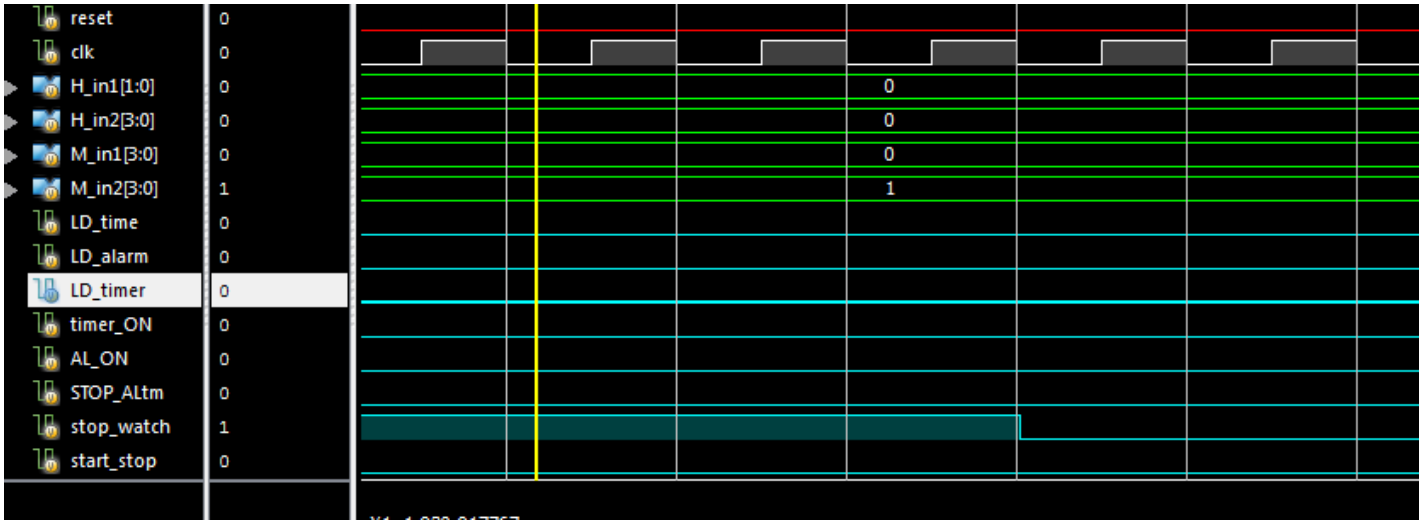


Figure 4.23: Input Perspective

‖When *stop_watch* is HIGH, StopWatch Display
When *stop_watch* is LOW , Real-Time Display

4.22 Checking Display-Switch (Stopwatch –> Real-Time)

Please observe that the display switches from the frozen stopwatch screen to current Real-time clock screen as expected.

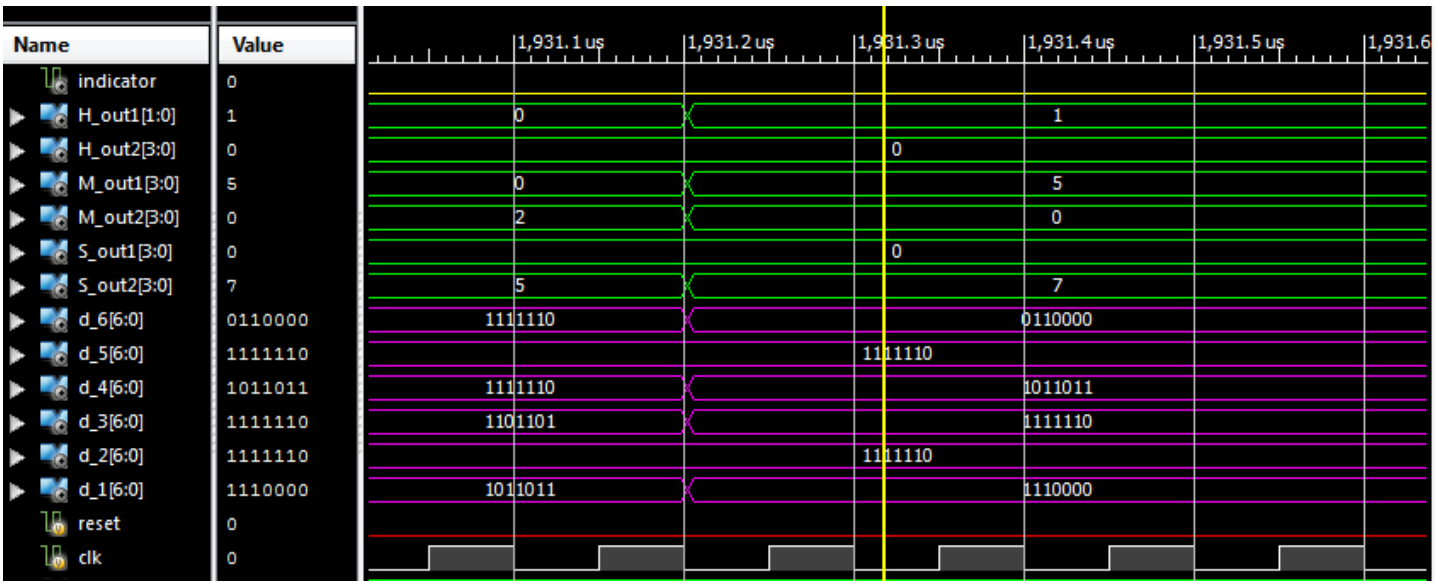


Figure 4.24: Output Perspective

Real-Time Clock

4.23 Back to Real-Time Clock (Functionality Check)

Please note that after a schedule of demonstrating various functionalities, the real-time clock and its display function work as expected.

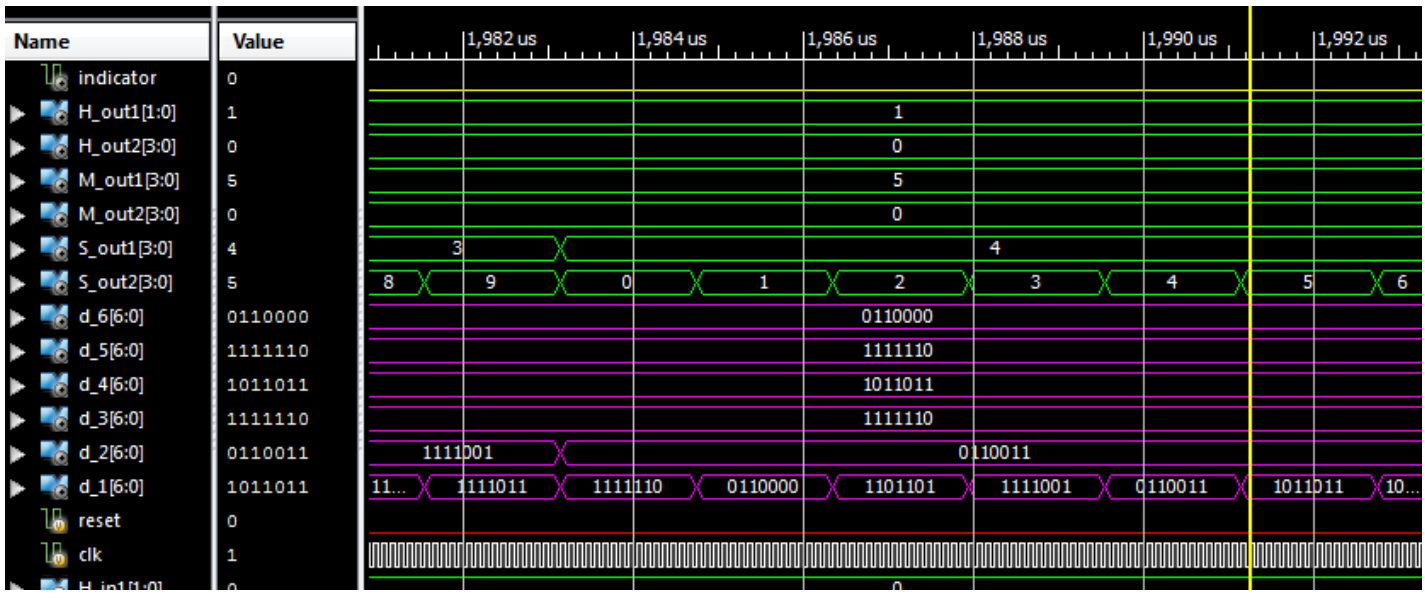


Figure 4.25: Output Perspective

5 | Concluding Remarks

5.1 A note on Implementation

Please note the following remarks regarding implementation :

- The project *DIGITAL CLOCK* can be implemented on an FPGA (Field Programmable Gate Array) for prototyping.
- The inputs to the module can be given via the push-button switches present on the FPGA by including the User Constraints File.
- Please note the redundant use of the clock's display , consisting both of seven segment and BCD output, is just for simulation and demonstration purposes and during implementation the BCD outputs may be discarded.
- Also, note the indicator (For Timer and Alarm) output can be connected to a buzzer or an LED as per the user's convenience.

5.2 Conclusion

The aim of the project ***"DIGITAL CLOCK"*** to realize a simple yet useful real-time clock consisting of the various features namely :

Alarm, Stopwatch, Timer and 24 hr Clock Display was achieved completely using Verilog Hardware Description Language (HDL)

and was also demonstrated with a variety of informative and inferential simulation results.

References

- *https://www.fpga4student.com/2016/11/verilog-code-for-alarm-clock-on-fpga.html*
- *https://nptel.ac.in/courses/106/105/106105165/*
- *https://www.slideshare.net/AbhishekSainkar1/digital-clock-using-verilog*
- *https://tex.stackexchange.com/*