(32-bit)

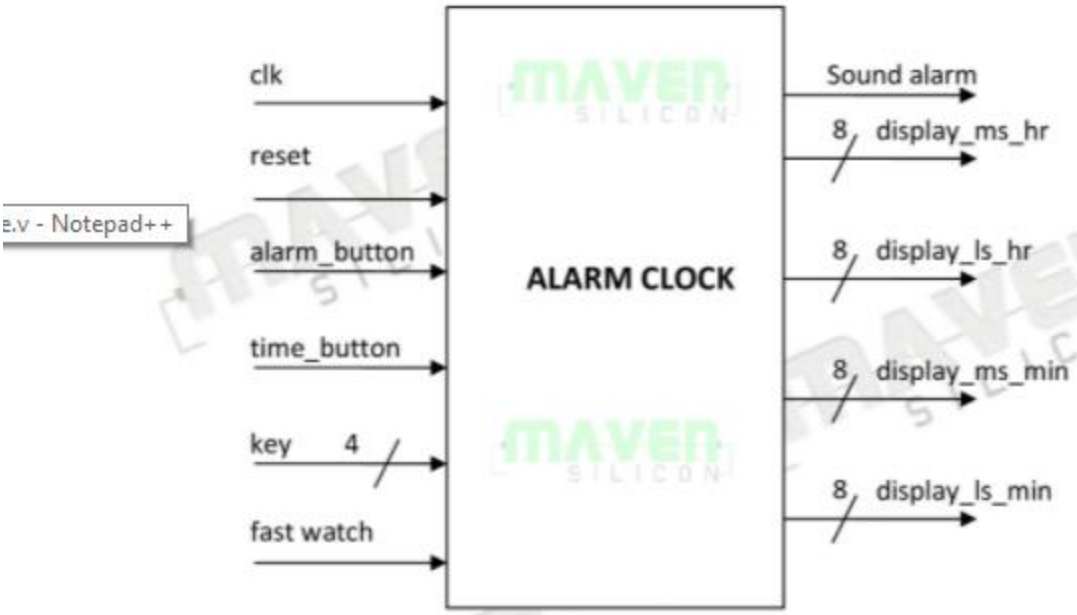
Alarm Clock Project

Table of Contents

Alarm Clock

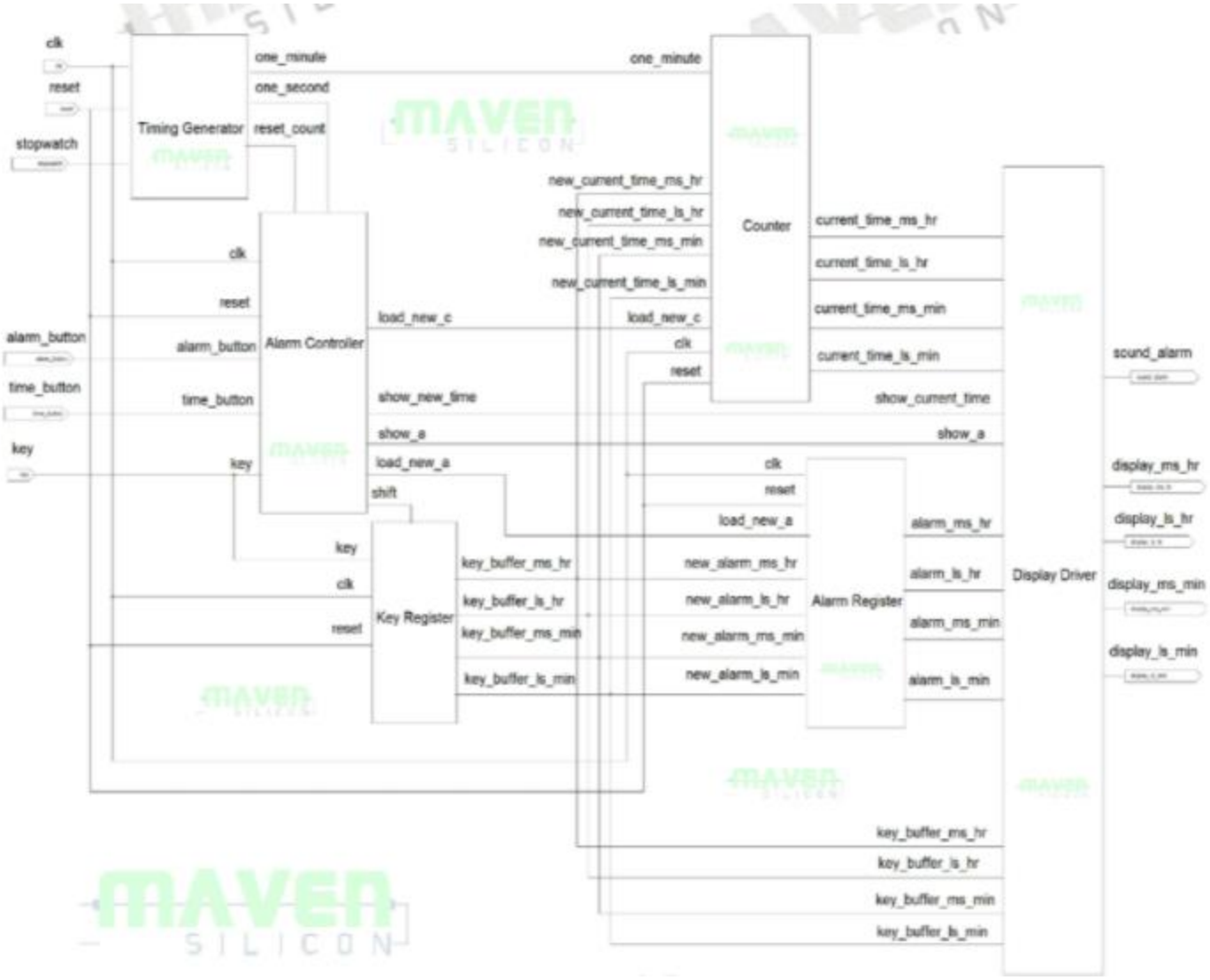
A digital alarm clock displays time in the LCD display format.

Block Diagram

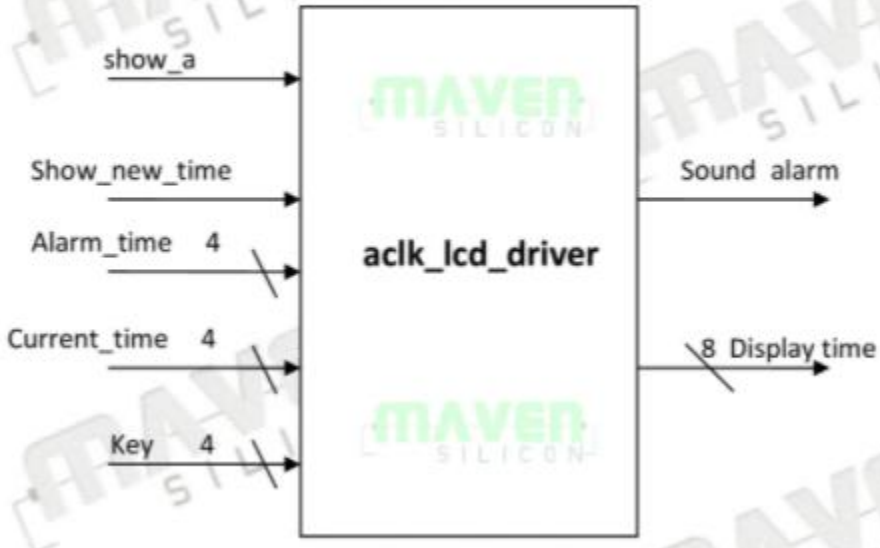


I/O Ports :

- "clk" is a 256 Hz clock.
- "reset" is an asynchronous active high reset.
- "Key" is the four bits key input
- "alarm_button" is an active high control signal for setting the Alarm Time
- "time_button" is an active high control signal for setting the Current Time
- "sound_alarm" is an active high output.
- "Display Time" are the outputs which display the values in LCD format.
- In "fast_watch" mode clock runs faster by using second pulse as minute



ALARM CLOCK: LCD Display Driver



"aclk_lcd_driver" is a combinational logic unit which displays the four bit binary values of the keys in the LCD format and it also generates the alarm sound signal.

Functionality:

"show_a" and "show_new_time" are active high control signals which control the display_time.

- [show_a = 1] AND [show_new_time = 0] => [display_time = alarm_time]
- [show_a = 0] AND [show_new_time = 0] => [display_time = current_time]
- [show_a = 0] AND [show_new_time = 1] => [display_time = key_time]

t) The inputs alarm_time , current_time and key use 4 bits BCD

The output display_time uses 8bits equivalent LCD, as per the table shown below.

BCD Value	LCD Value
0000	8'h30
0001	8'h31
0010	8'h32
0011	8'h33
0100	8'h34
0101	8'h35
0110	8'h36
0111	8'h37
1000	8'h38
1001	8'h39

RTL Design Procedure:

[1] Naming Convention:

[A] Modules and Files

Module Name: aclk_lcd_driver File Name: aclk_lcd_driver.v

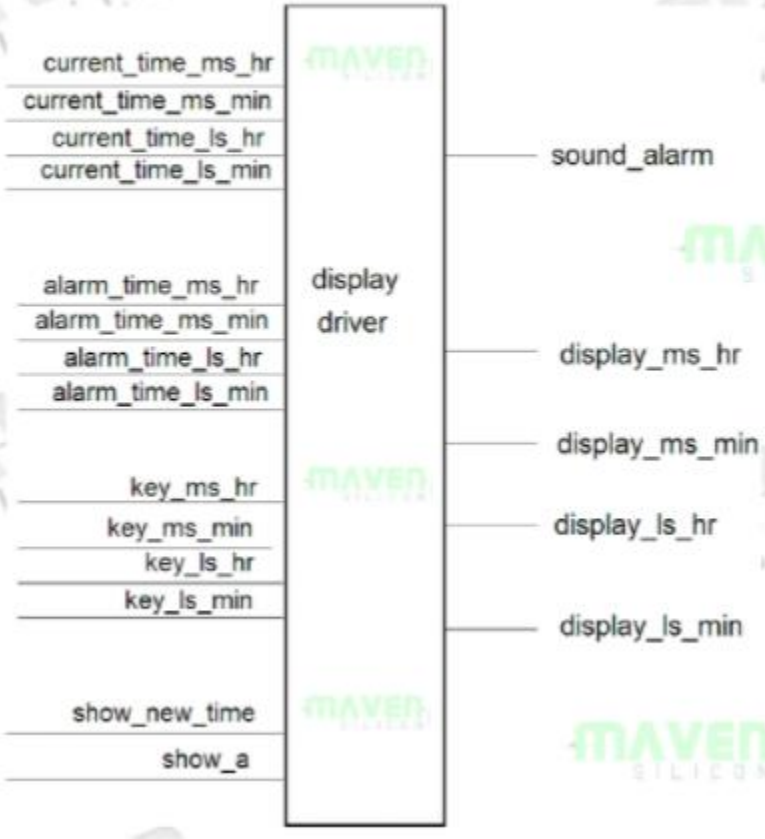
Test bench Name: tb_aclk_lcd_driver File Name: tb_aclk_lcd_driver.v

[B] Name the ports as per the block

[2] Verify the functionality of the RTL

[3] Synthesize the RTL

ALARM CLOCK: LCD Display Unit



"aclk_lcd_display" is a display unit which displays all the digits MS_HR, LS_HR, MS_MIN and LS_MIN in the LCD format as shown below.

MS Hour LS Hour MS Minute LS Minute

RTL Design Procedure:

[1] Naming Convention:

[A] Modules and Files

Module Name: aclk_lcd_display File Name: aclk_lcd_display.v

Test bench Name: tb_aclk_lcd_display FileName: tb_aclk_lcd_display.v

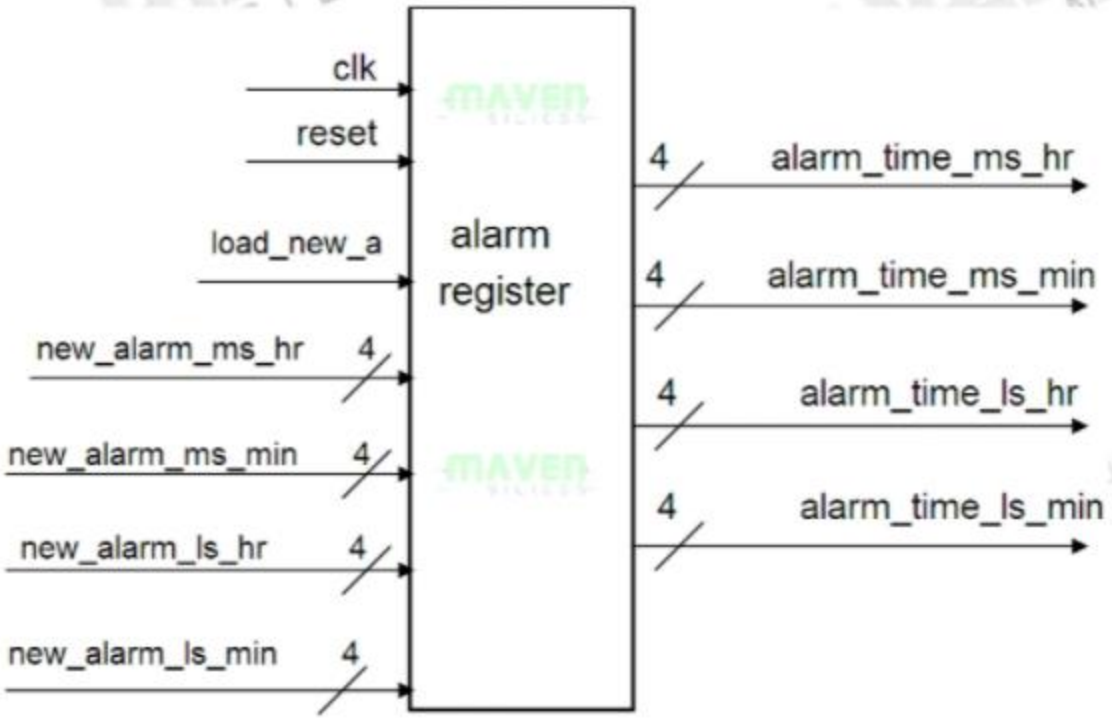
[B] Name the ports as per the block diagram

[2] Instantiate the display driver "aclk_lcd_driver" and create instance for each digit and implement this display unit.

[3] Verify the functionality of the RTL

[3] Synthesize the RTL

ALARM CLOCK: The Alarm Register



Functionality:

This alarm register is a sequential block which stores the values of alarm time values.

- [reset = 1] => All outputs become ZERO
- [load_new_a = 1] AND [reset = 0] => All outputs get the value of all the inputs.
- [load_new_a = 0] AND [reset = 0] => All outputs retain their previous value

RTL Design Procedure:

[1] Naming Convention:

[A] Modules and Files

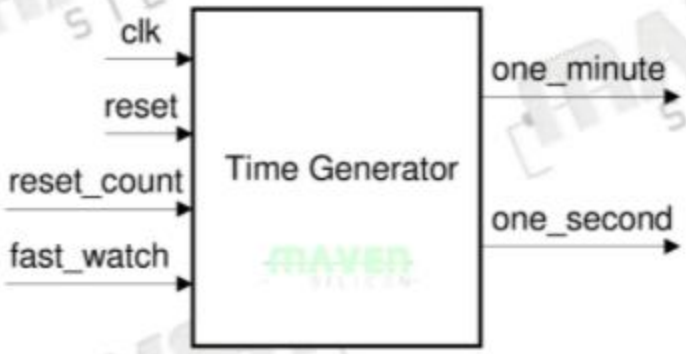
Module Name: aclk_areg — File Name: aclk_areg.v
Test bench Name: tb_aclk_areg — File Name: tb_aclk_areg.v

[B] Name the ports as per the block diagram

[2] Verify the functionality of the RTL

[3] Synthesize the RTL

ALARM CLOCK: Time Generator



Time Generator is a sequential logic block which generates the one_minute pulse for the counter that counts the minutes and hours.

Functionality:

dclk_256 is the divided clock input. Time Generator generates the one_minute and one_second pulses based on the algorithm mentioned below.

For 256Hz clk → one second = 256 cycles
→ one minute = 256*60 = 15360 cycles

$15360_{10} = 11110000000000_2$

- one_second will be active high for one clock period, when the number of clock cycles = 256
- one_minute will be active high for one clock period only when the number of clock cycles = 15360
- [fast_watch=1] → [one_minute = one_second] – Useful only for faster simulation.
- [reset_count = 1] → All outputs reset to ZERO
- [reset = 1] → All outputs reset to ZERO

RTL Design Procedure:

[1] Naming Convention:

[A] Modules and Files

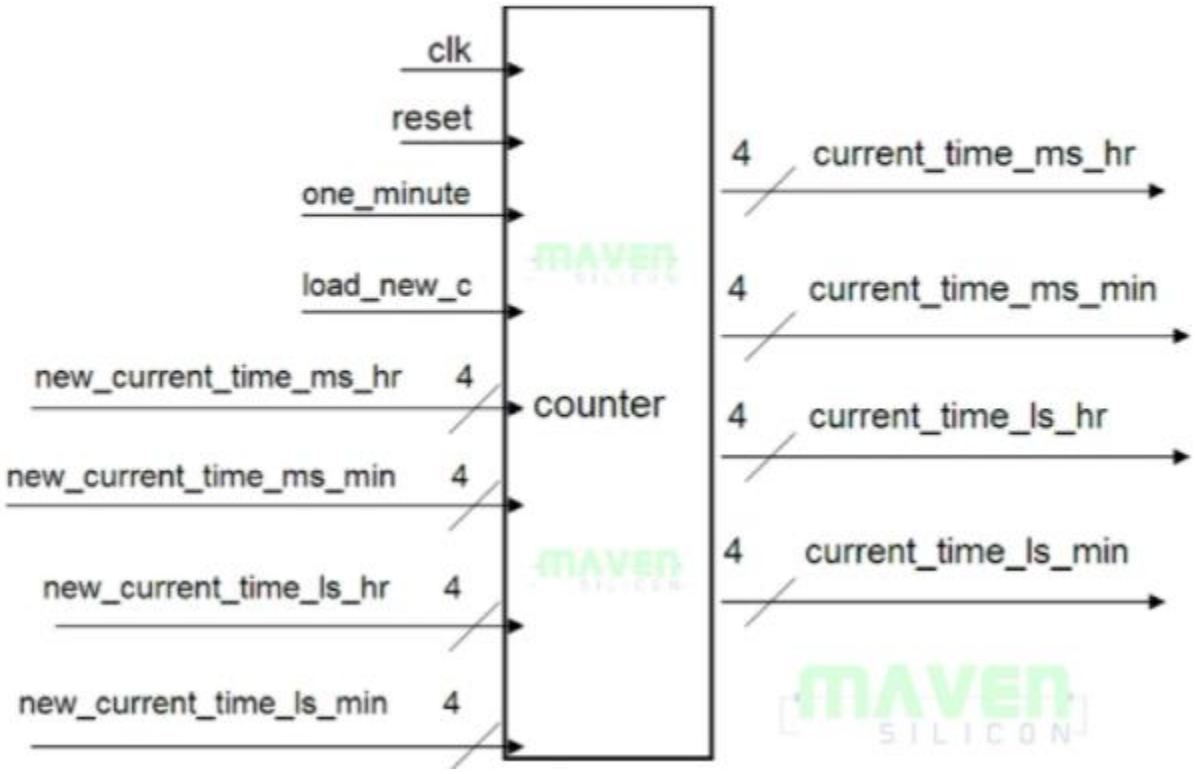
Module Name: aclk_timegen — File Name: aclk_timegen.v
Test bench Name: tb_aclk_timegen — File Name: tb_aclk_timegen.v

[B] Name the ports as per the block diagram

[2] Verify the functionality of the RTL

[3] Synthesize the RTL

ALARM CLOCK: Counter



Counter is a sequential block which counts the minutes and hours based on the one_minute pulse generated by the Time Generator.

Functionality:

- [reset = 1] → All the outputs reset to ZERO
- [load_new_c=1] AND [reset=0] → Counter is loaded with all the inputs new_current_time
- [load_new_c=0] AND [one_minute=0] → Counter retains it's previous values
- load_new_c=0] AND [one_minute=1] → Counter counts based on the counting algorithm

Counting Algorithm :

- [LS_MIN = 9] → [LS_MIN = 0] AND [MS_MIN = MS_MIN + 1]
- [MS_MIN = 5] AND [LS_MIN = 9] →

  [MS_MIN=0, LS_MIN =0] AND [LS_HR = LS_HR+1]
- [LS_HR=9 and MS_MIN= 5 and LS_MIN =9] →

  [MS_MIN=0, LS_MIN=0, LS_HR=0] AND [MS_HR = MS_HR+1]
- [MS_HR=2 and LS_HR =3 and MS_MIN=5 and LS_MIN =9] →

  [MS_MIN=0, LS_MIN=0, LS_HR=0, MS_HR=0]

RTL Design Procedure:

[1] Naming Convention:

[A] Modules and Files

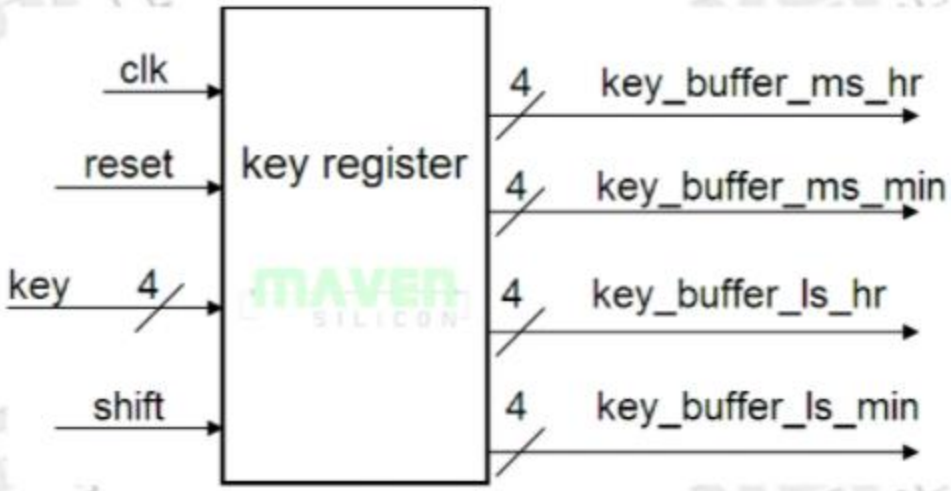
Module Name: aclk_counter	File Name: aclk_counter.v
Test bench Name: tb_aclk_counter	FileName: tb_aclk_counter.v

[B] Name the ports as per the block diagram

[2] Verify the functionality of the RTL

[3] Synthesize the RTL

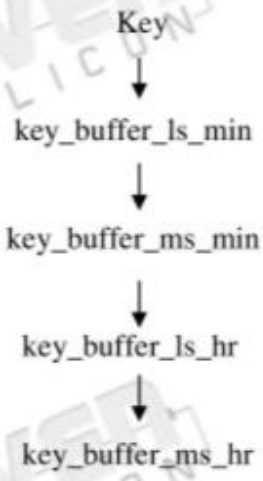
ALARM CLOCK: Key Register



Key Register is sequential logic unit which shifts and stores the key values.

Functionality:

The key value will be assigned to key_buffer_ls_min at the positive edge of clock when the shift signal goes high. The old key values will be shifted to other registers (key_buffer_ms_min, key_buffer_ms_hr, key_buffer_ls_hr) whenever you press new keys.



Current Time: 1 2 : 59
New Time: 0 9 : 43

Display Patterns:

1 2 5 9
2 5 9 **0**
5 9 **0 9**
9 **0 9 4**
0 9 4 3

RTL Design Procedure:

[1] Naming Convention:

[A] Modules and Files

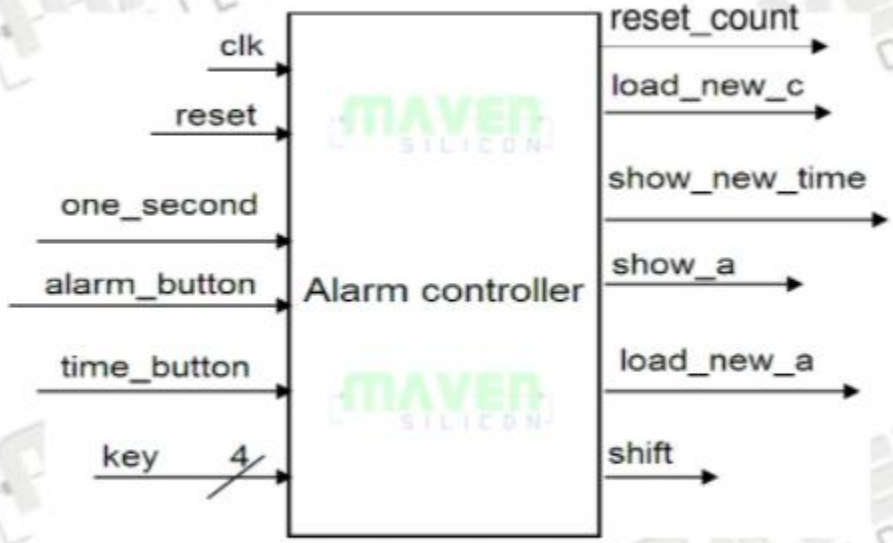
Module Name: aclk_keyreg — File Name: aclk_keyreg.v
Test bench Name: tb_aclk_keyreg — FileName: tb_aclk_keyreg.v

[B] Name the ports as per the block diagram

[2] Verify the functionality of the RTL

[3] Synthesize the RTL

ALARM CLOCK: Controller Unit



This is a controller unit for the Alarm Clock. It generates various control signals which controls the other blocks, Key Reg, Display Driver and Time Generator.

Functionality:

Functionality:

Setting Time Digits :

0 to 9 are the valid keys to set the time value.
10 represents "No Key" – Means, the user is not pressing any keys. [Default value of the keypad output]

One has to press four keys to set all the digits MS_HR, LS_HR, MS_MIN and LS_MIN, by taking maximum 10 seconds for each key. If no keys are pressed within 10 seconds, the new value will be ignored and the display immediately returns to the current time.

The new values will be shifted through the digits MS_HR, LS_HR, MS_MIN and LS_MIN, from right to left, as shown below.

Current Time: 1 2 59
New Time: 0 9 43

Display Patterns :

1 2 5 9
2 5 9 **0**
5 9 **0 9**
9 **0 9 4**
0 9 4 3

Setting new alarm time:

The user sets the alarm time as explained above and presses the alarm button. Now the clock considers the digits set by user for the alarm time. The display will show the alarm time.

Setting new time:

The user sets the alarm time as explained above and presses the time button. Now the clock considers the digits set by user for the current time. The display will show the new time.

Displaying Alarm Time:

Alarm Time is displayed when the user presses the alarm button, without pressing any other keys.

Output Logic:

[State = SHOW_ALARM] => show_a = 1'b1

RTL Design Procedure:

[1] Naming Convention:

[A] Modules and Files

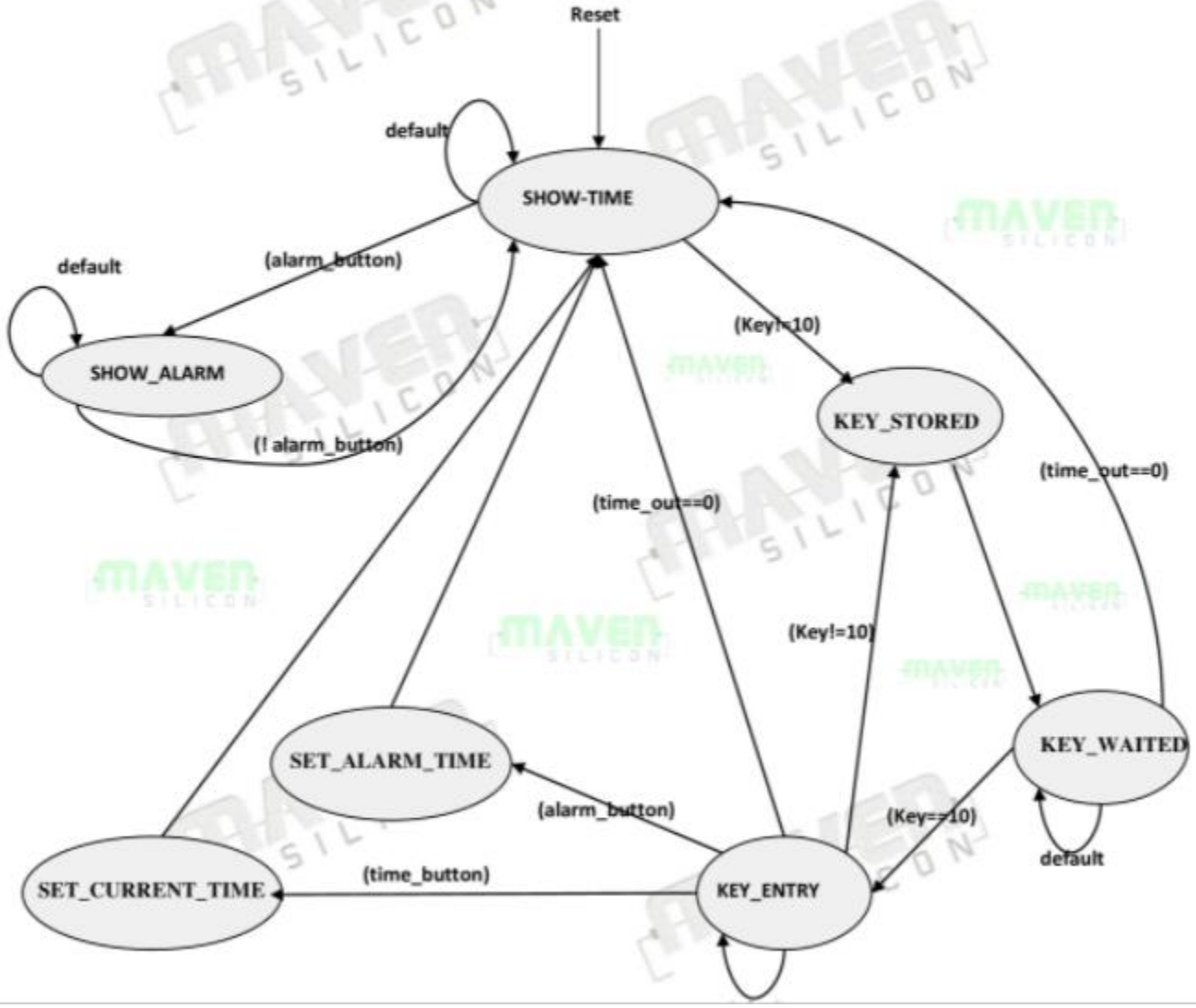
Module Name: aclk_controller — File Name: aclk_controller.v
Test bench Name: tb_aclk_controller — FileName: tb_aclk_controller.v

[B] Name the ports as per the block diagram

[2] Verify the functionality of the RTL

[3] Synthesize the RTL

Controller – FSM



ALARM CLOCK: Top Level RTL Module

RTL Design Procedure:

[1] Naming Convention:

[A] Modules and Files

Module Name: alarm_clk_rtl — File Name: alarm_clk_rtl.v
Test bench Name: tb_alarm_clk_rtl — FileName: tb_alarm_clk_rtl.v

[B] Name the ports as per the block diagram

[2] Instantiate all the lower level modules and implement the top module, as per the architecture shown in page 4.

[3] Write a testbench and verify the functionality of the RTL

[4] Synthesize the RTL