///TIME\_GENERATOR

module aclk\_timegen(clk,reset,reset\_count,fast\_watch,one\_second,one\_minute);

input clk,reset,reset\_count,fast\_watch;

output reg one\_second,one\_minute;

reg [13:0]count;

reg one\_minutereg;

always@(posedge clk or posedge reset)

begin

if(reset)

begin

count<=14'b0;

one\_minutereg<=1'b0;

end

else if(reset\_count)

begin

count<=14'b0;

one\_minutereg<=1'b0;

end

else if(count[13:0]==14'd15359)

begin

count<=14'b0;

one\_minutereg=1'b1;

end

else

begin

count<=count+1'b1;

one\_minutereg=1'b0;

end

end

always@(posedge clk or posedge reset)

begin

if(reset)

begin

one\_second<=1'b0;

end

else if(reset\_count)

begin

one\_second<=1'b0;

end

else if(count[7:0]==8'd255)

begin

one\_second<=1'b1;

end

else

begin

one\_second<=1'b0;

end

end

always@(\*)

begin

if(fast\_watch)

one\_minute<=one\_second;

else

one\_minute<=one\_minutereg;

end

endmodule

//TIME\_GENERATOR\_TESTBENCH

`timescale 1ms/1ns

module tb\_aclk\_timegen;

reg clk,reset,reset\_count,fast\_watch;

wire one\_second,one\_minute;

integer k,l;

aclk\_timegen DUT(clk,reset,reset\_count,fast\_watch,one\_second,one\_minute);

always

begin

#1.953125 clk=1'b0;

#1.953125 clk=1'b1;

end

task rst\_all;

begin

@(negedge clk) reset=1;

@(negedge clk) reset=0;

end

endtask

task rst\_count;

begin

@(negedge clk) reset\_count=1;

@(negedge clk) reset\_count=0;

end

endtask

task fas\_watch(input a);

begin

fast\_watch=a;

end

endtask

initial

begin

rst\_all;

fas\_watch(1'b1);

for(k=0;k<120;k=k+1)

begin

if(k==60)

fas\_watch(1'b0);

for(l=0;l<256;l=l+1)

#3.906250;

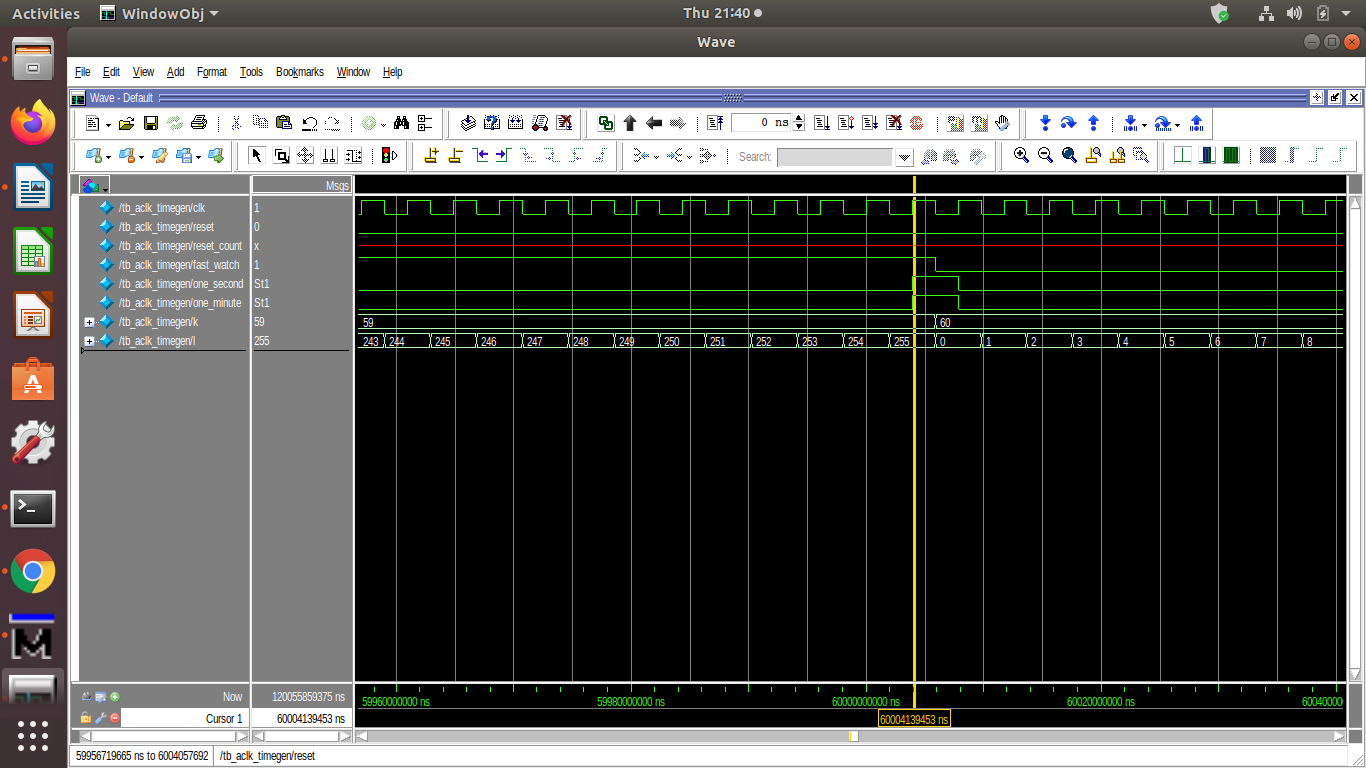
end

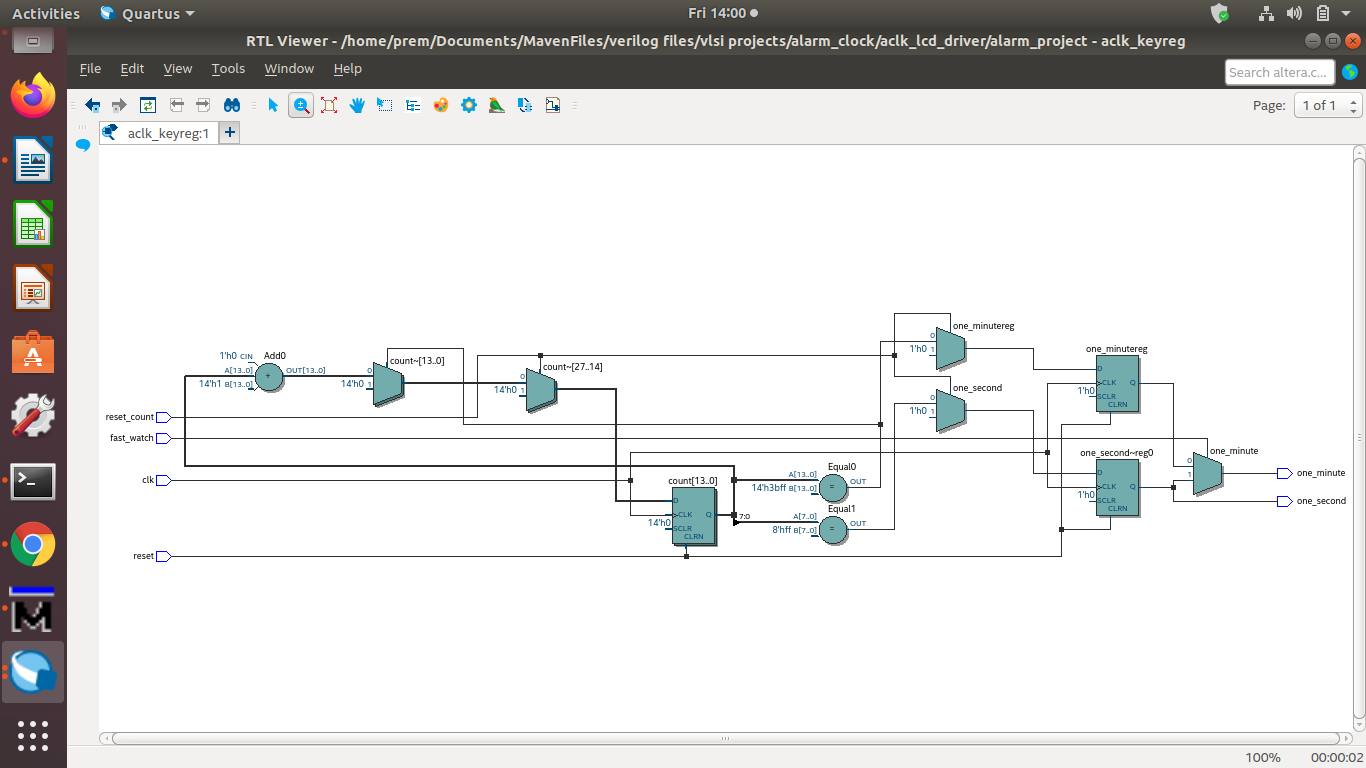
#50 $finish;

end

endmodule

TIME\_GENERATOR\_SIMULATION



TIME\_GENERATOR\_SYNTHESIS

//CONTROLLER

module aclk\_controller(clk,reset,one\_second,alarm\_button,time\_button,key,

reset\_count,load\_new\_c,show\_new\_time,show\_a,load\_new\_a,shift);

input clk,reset,one\_second,alarm\_button,time\_button;

input [3:0]key;

output reset\_count,load\_new\_c,show\_new\_time,show\_a,load\_new\_a;

output wire shift;

reg [6:0]state,nextstate;

reg [3:0] count1,count2;

wire time\_out;

parameter SHOW\_TIME=7'd1,KEY\_STORED=7'd2,KEY\_WAITED=7'd4,KEY\_ENTRY=7'd8,

SET\_ALARM\_TIME=7'd16,SET\_CURRENT\_TIME=7'd32,SHOW\_ALARM=7'd64;

//sequential logic

always@(posedge clk or posedge reset)

begin

if(reset) state<=SHOW\_TIME;

else state<=nextstate;

end

//next state decoding logic

always@( alarm\_button or time\_button or key or state or time\_out)

begin

case(state)

SHOW\_TIME : begin

if(alarm\_button) nextstate<=SHOW\_ALARM;

else if(key!=4'd10) nextstate<=KEY\_STORED;

else nextstate<=SHOW\_TIME;

end

KEY\_STORED : nextstate=KEY\_WAITED;

KEY\_WAITED : begin

if(key==4'd10) nextstate=KEY\_ENTRY;

else if(time\_out==0) nextstate=SHOW\_TIME;

else nextstate=KEY\_WAITED;

end

KEY\_ENTRY : begin

if(alarm\_button) nextstate=SET\_ALARM\_TIME;

else if(time\_button) nextstate=SET\_CURRENT\_TIME;

else if(time\_out==0) nextstate=SHOW\_TIME;

else if(key!=4'd10) nextstate=KEY\_STORED;

else nextstate=KEY\_ENTRY;

end

SET\_ALARM\_TIME : nextstate=SHOW\_TIME;

SET\_CURRENT\_TIME: nextstate=SHOW\_TIME;

SHOW\_ALARM : begin

if(!alarm\_button) nextstate=SHOW\_TIME;

else nextstate=SHOW\_ALARM;

end

default : nextstate=SHOW\_TIME;

endcase

end

//output logic

assign reset\_count=(state==SET\_CURRENT\_TIME)?1:0;

assign load\_new\_c=(state==SET\_CURRENT\_TIME)?1:0;

assign show\_new\_time=(state==KEY\_ENTRY || state==KEY\_STORED || state==KEY\_WAITED)?1:0;

assign show\_a=(state==SHOW\_ALARM)?1:0;

assign load\_new\_a=(state==SET\_ALARM\_TIME)?1:0;

assign shift=(state==KEY\_STORED)?1:0;

//timers

always@(posedge clk or posedge reset)

begin

if(reset)

count1<=4'd0;

else if(!(state==KEY\_WAITED))

count1<=4'd0;

else if(count1==9)

count1<=4'd0;

else if(one\_second)

count1=count1+1'b1;

end

always@(posedge clk or posedge reset)

begin

if(reset)

count2<=4'd0;

else if(!(state==KEY\_ENTRY))

count2<=4'd0;

else if(count2==9)

count2<=4'd0;

else if(one\_second)

count2=count2+1'b1;

end

assign time\_out=((count1==9) || (count2==9))?0:1;

endmodule

//CONTROLLER\_TESTBENCH

module tb\_aclk\_controller;

reg clk,reset,one\_second,alarm\_button,time\_button;

reg [3:0]key;

wire reset\_count,load\_new\_c,show\_new\_time,show\_a,load\_new\_a,shift;

aclk\_controller control(clk,reset,one\_second,alarm\_button,time\_button,key,

reset\_count,load\_new\_c,show\_new\_time,show\_a,load\_new\_a,shift);

always

begin

#5 clk=1'b0;

#5 clk=1'b1;

end

task rst;

begin

@(negedge clk)reset=1;

@(negedge clk) reset=0;

end

endtask

task stimulus(input a,t,s);

begin

alarm\_button=a;

time\_button=t;

@(negedge clk) one\_second=s;

@(negedge clk) one\_second=1'b0;

end

endtask

initial

begin

rst;

#10 key=4'h0;

#10 key=4'hA;

#20 key=4'h9;

#10 key=4'hA;

#20 key=4'h5;

#10 key=4'hA;

#20 key=4'h3;

#10 key=4'hA;

#10 stimulus(1'b1,1'b0,1'b0);

#40 stimulus(1'b0,1'b1,1'b0);

#10 key=4'h0;

#10 key=4'hA;

#20 key=4'h9;

#10 key=4'hA;

#20 key=4'h5;

#10 key=4'hA;

#20 key=4'h3;

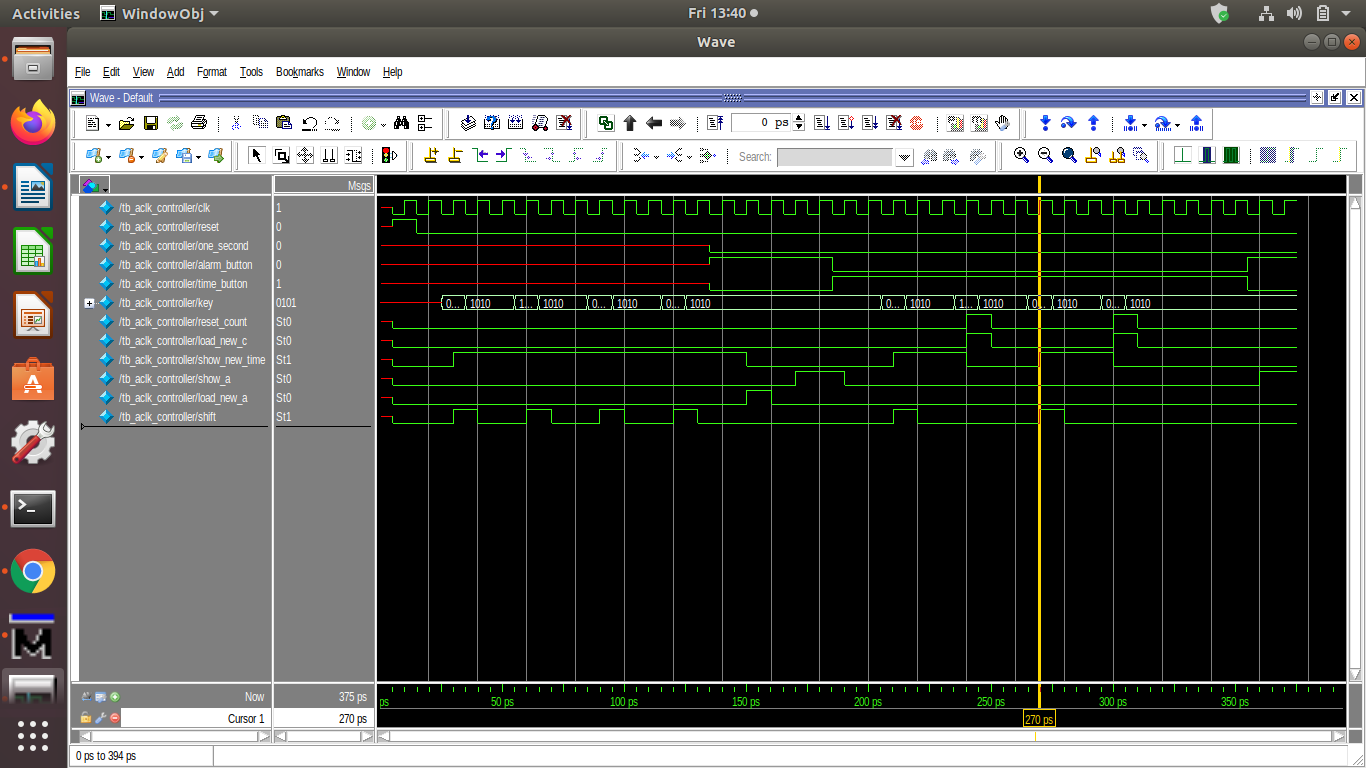
#10 key=4'hA;

#50 stimulus(1'b1,1'b0,1'b0);

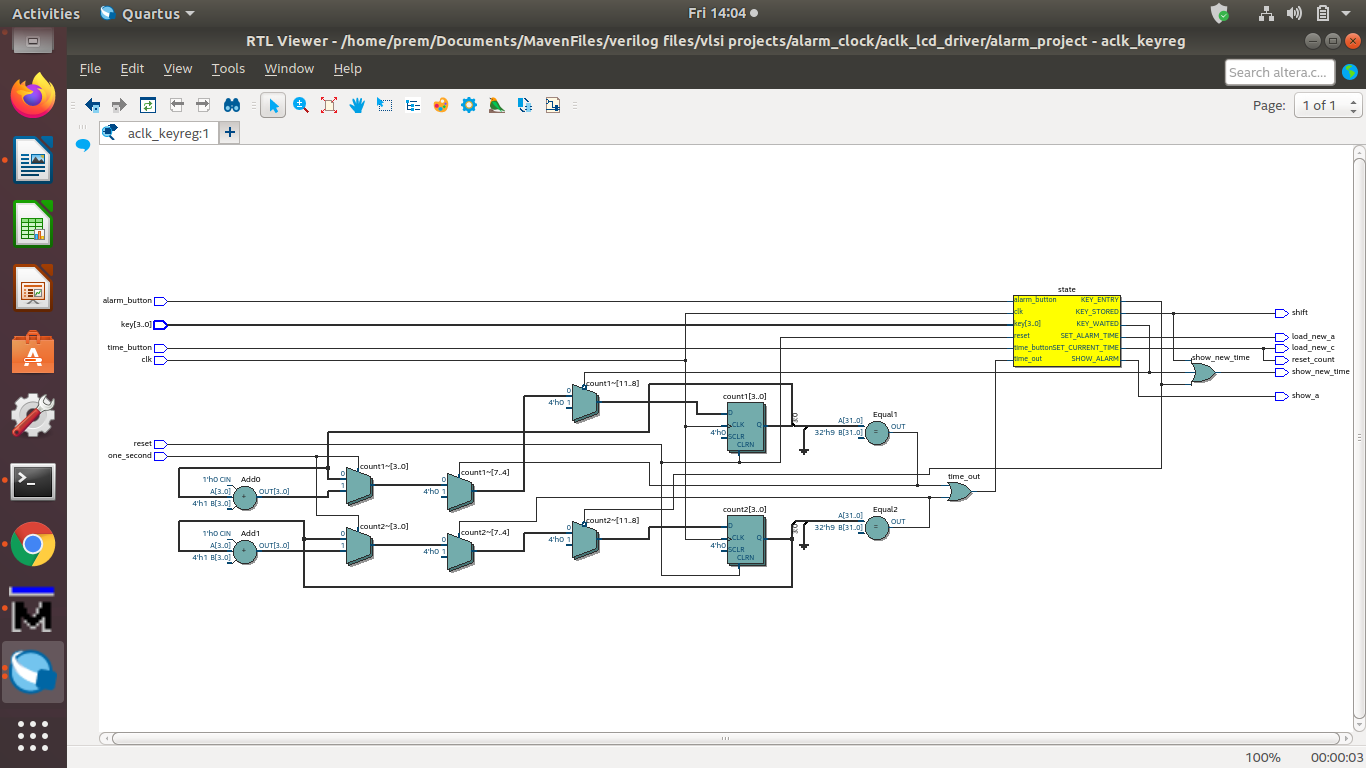
#10 $finish;

end

endmodule

CONTROLLER\_SIMULATION

CONTROLLER\_SYNTHESIS



//KEY\_REGISTER

module aclk\_keyreg(clk,reset,key,shift,

key\_buffer\_ms\_hr,key\_buffer\_ms\_min,key\_buffer\_ls\_hr,key\_buffer\_ls\_min);

input clk,reset,shift;

input[3:0] key;

output reg[3:0] key\_buffer\_ms\_hr,key\_buffer\_ms\_min,key\_buffer\_ls\_hr,key\_buffer\_ls\_min;

always@(posedge clk or posedge reset)

begin

if(reset)

begin

key\_buffer\_ms\_hr<=0;

key\_buffer\_ms\_min<=0;

key\_buffer\_ls\_hr<=0;

key\_buffer\_ls\_min<=0;

end

else

begin

if(shift)

begin

key\_buffer\_ms\_hr<=key\_buffer\_ls\_hr;

key\_buffer\_ls\_hr<=key\_buffer\_ms\_min;

key\_buffer\_ms\_min<=key\_buffer\_ls\_min;

key\_buffer\_ls\_min<=key;

end

end

end

endmodule

//KEY\_REGISTER\_TESTBENCH

module tb\_aclk\_keyreg;

reg clk,reset,shift;

reg[3:0] key;

wire[3:0] key\_buffer\_ms\_hr,key\_buffer\_ms\_min,key\_buffer\_ls\_hr,key\_buffer\_ls\_min;

aclk\_keyreg DUT(clk,reset,key,shift,key\_buffer\_ms\_hr,key\_buffer\_ms\_min,

key\_buffer\_ls\_hr,key\_buffer\_ls\_min);

always

begin

#5 clk=1'b0;

#5 clk=1'b1;

end

task rst;

begin

@(negedge clk) reset=1;

@(negedge clk) reset=0;

end

endtask

task \_shift;

begin

@(negedge clk) shift=1;

@(negedge clk) shift=0;

end

endtask

initial

begin

rst;

key=4'd0;

\_shift;

key=4'd9;

\_shift;

key=4'd5;

\_shift;

key=4'd7;

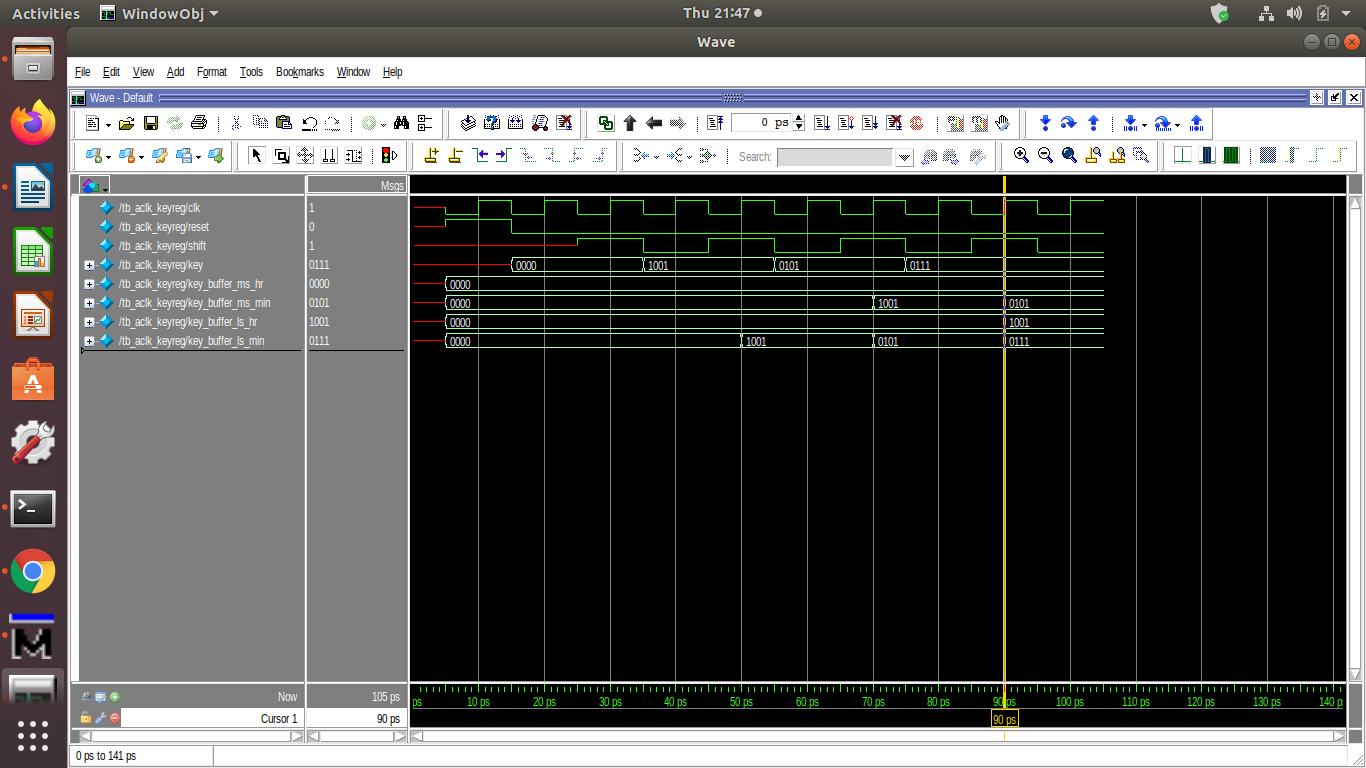
\_shift;

#10 $finish;

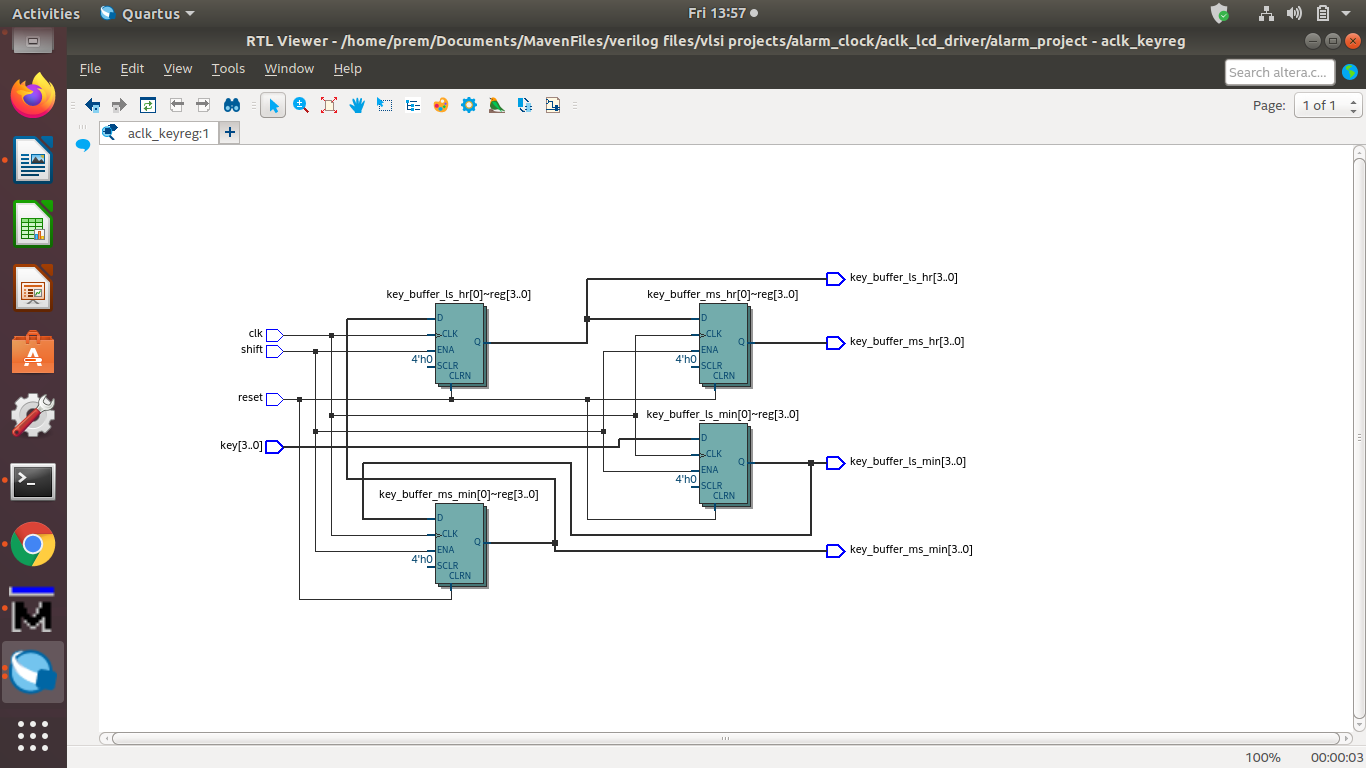
end

endmodule

KEY\_REGISTER\_SIMULATION



KEY\_REG\_SYNTHESIS



//ALARM\_REGISTER

module aclk\_areg(input clk,reset,load\_new\_a,

input[3:0]new\_alarm\_ms\_hr,new\_alarm\_ms\_min,

new\_alarm\_ls\_hr,new\_alarm\_ls\_min,

output reg[3:0]alarm\_time\_ms\_hr,alarm\_time\_ms\_min,

alarm\_time\_ls\_hr,alarm\_time\_ls\_min);

always@(posedge clk or posedge reset or posedge load\_new\_a)

begin

if(reset)

begin

alarm\_time\_ms\_hr<=0;alarm\_time\_ms\_min<=0;

alarm\_time\_ls\_hr<=0;alarm\_time\_ls\_min<=0;

end

else if(load\_new\_a)

begin

alarm\_time\_ms\_hr<=new\_alarm\_ms\_hr;alarm\_time\_ms\_min<=new\_alarm\_ms\_min;

alarm\_time\_ls\_hr<=new\_alarm\_ls\_hr;alarm\_time\_ls\_min<=new\_alarm\_ls\_min;

end

end

endmodule

//ALARM\_REG\_TESTBENCH

module tb\_aclk\_areg;

reg clk,rst,load;

reg [3:0]n\_a\_ms\_hr,n\_a\_ms\_min,n\_a\_ls\_hr,n\_a\_ls\_min;

wire [3:0]a\_t\_ms\_hr,a\_t\_ms\_min,a\_t\_ls\_hr,a\_t\_ls\_min;

aclk\_areg DUT(.clk(clk),.reset(rst),.load\_new\_a(load),

.new\_alarm\_ms\_hr(n\_a\_ms\_hr),.new\_alarm\_ms\_min(n\_a\_ms\_min),

.new\_alarm\_ls\_hr(n\_a\_ls\_hr),.new\_alarm\_ls\_min(n\_a\_ls\_min),

.alarm\_time\_ms\_hr(a\_t\_ms\_hr),.alarm\_time\_ms\_min(a\_t\_ms\_min),

.alarm\_time\_ls\_hr(a\_t\_ls\_hr),.alarm\_time\_ls\_min(a\_t\_ls\_min));

task \_rst;

begin

@(negedge clk)

rst=1'b1;

@(negedge clk)

rst=1'b0;

end

endtask

task \_load(input[3:0]ms\_hr,ms\_min,ls\_hr,ls\_min);

begin

@(negedge clk)

load=1'b1;

n\_a\_ms\_hr=ms\_hr;n\_a\_ms\_min=ms\_min;

n\_a\_ls\_hr=ls\_hr;n\_a\_ls\_min=ls\_min;

@(negedge clk)

load=1'b0;

end

endtask

always

begin

#5 clk=1'b0;

#5 clk=1'b1;

end

initial

begin

\_rst;

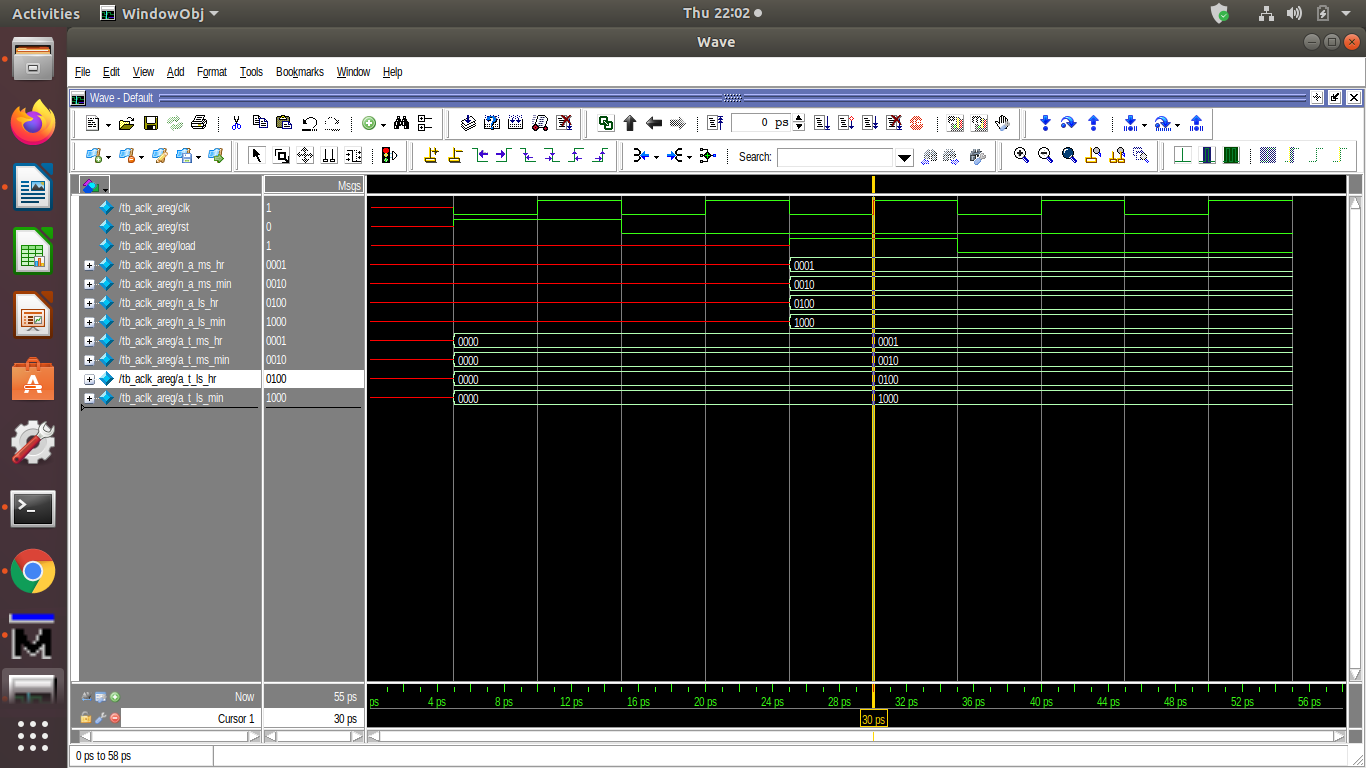
\_load(1,2,4,8);

#20 $finish;

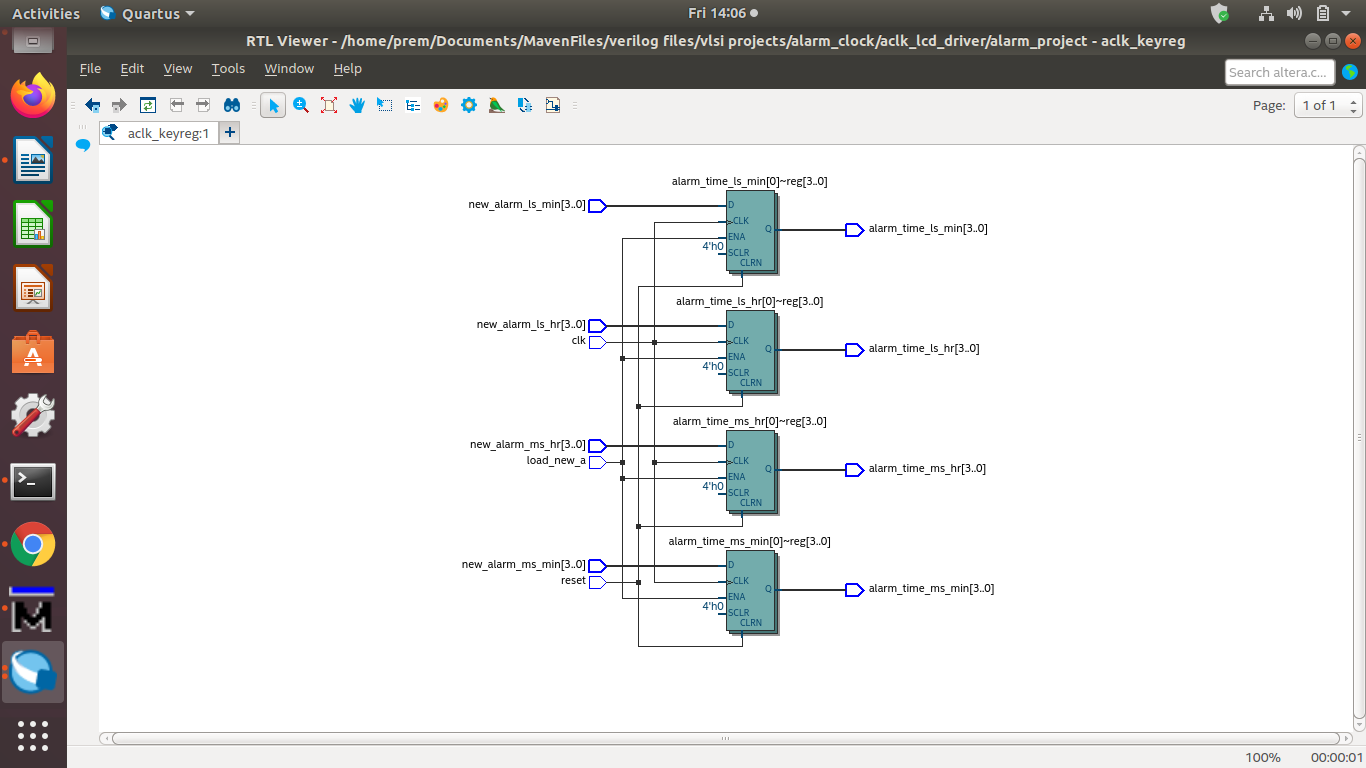
end

endmodule

ALARM\_REGISTER\_SIMULATION



ALARM\_REGISTER\_SYNTHESIS



//COUNTER

module aclk\_counter(clk,reset,one\_minute,load\_new\_c,

new\_current\_ms\_hr,new\_current\_ms\_min,new\_current\_ls\_hr,new\_current\_ls\_min,

current\_time\_ms\_hr,current\_time\_ms\_min,current\_time\_ls\_hr,current\_time\_ls\_min);

input clk,reset,one\_minute,load\_new\_c;

input[3:0] new\_current\_ms\_hr,new\_current\_ms\_min,new\_current\_ls\_hr,new\_current\_ls\_min;

output reg[3:0] current\_time\_ms\_hr,current\_time\_ms\_min,current\_time\_ls\_hr,current\_time\_ls\_min;

always@(posedge clk or posedge reset )

begin

if(reset)

begin

current\_time\_ms\_hr<=4'd0;current\_time\_ms\_min<=4'd0;

current\_time\_ls\_hr<=4'd0;current\_time\_ls\_min<=4'd0;

end

else if(load\_new\_c)

begin

current\_time\_ms\_hr<=new\_current\_ms\_hr;current\_time\_ms\_min<=new\_current\_ms\_min;

current\_time\_ls\_hr<=new\_current\_ls\_hr;current\_time\_ls\_min<=new\_current\_ls\_min;

end

else

begin

if(one\_minute)

begin

if(current\_time\_ms\_hr==4'd2 && current\_time\_ls\_hr==4'd3 && current\_time\_ms\_min==4'd5 && current\_time\_ls\_min==4'd9)

begin

current\_time\_ms\_hr<=4'd0;current\_time\_ls\_hr<=4'd0;

current\_time\_ms\_min<=4'd0;current\_time\_ls\_min<=4'd0;

end

else if(current\_time\_ls\_hr==4'd9 && current\_time\_ms\_min==4'd5 && current\_time\_ls\_min==4'd9)

begin

current\_time\_ms\_hr<=current\_time\_ms\_hr+1'd1;current\_time\_ls\_hr<=4'd0;

current\_time\_ms\_min<=4'd0;current\_time\_ls\_min<=4'd0;

end

else if(current\_time\_ms\_min==4'd5 && current\_time\_ls\_min==4'd9)

begin

current\_time\_ls\_hr<=current\_time\_ls\_hr+1'd1;

current\_time\_ms\_min<=4'd0;current\_time\_ls\_min<=4'd0;

end

else if(current\_time\_ls\_min==4'd9)

begin

current\_time\_ms\_min<=current\_time\_ms\_min+1'd1;current\_time\_ls\_min<=4'd0;

end

else

current\_time\_ls\_min<=current\_time\_ls\_min+1'd1;

end

end

end

endmodule

//COUNTER\_TESTBENCH

module tb\_aclk\_counter;

reg clk,reset,one\_minute,load\_new\_c;

reg[3:0] new\_current\_ms\_hr,new\_current\_ms\_min,new\_current\_ls\_hr,new\_current\_ls\_min;

wire[3:0] current\_time\_ms\_hr,current\_time\_ms\_min,current\_time\_ls\_hr,current\_time\_ls\_min;

integer i;

aclk\_counter DUT(clk,reset,one\_minute,load\_new\_c,

new\_current\_ms\_hr,new\_current\_ms\_min,new\_current\_ls\_hr,new\_current\_ls\_min,

current\_time\_ms\_hr,current\_time\_ms\_min,current\_time\_ls\_hr,current\_time\_ls\_min);

always

begin

#5 clk=1'b0;

#5 clk=1'b1;

end

task rst;

begin

@(negedge clk) reset=1;

@(negedge clk) reset=0;

end

endtask

task load(input [3:0]ms\_hr,ls\_hr,ms\_min,ls\_min);

begin

@(negedge clk) load\_new\_c=1;

new\_current\_ms\_hr<=ms\_hr;new\_current\_ms\_min<=ms\_min;

new\_current\_ls\_hr<=ls\_hr;new\_current\_ls\_min<=ls\_min;

@(negedge clk) load\_new\_c=0;

end

endtask

task minute;

begin

@(negedge clk) one\_minute=1;

@(negedge clk) one\_minute=0;

end

endtask

initial

begin

rst;

for(i=0;i<1440;i=i+1)

minute;

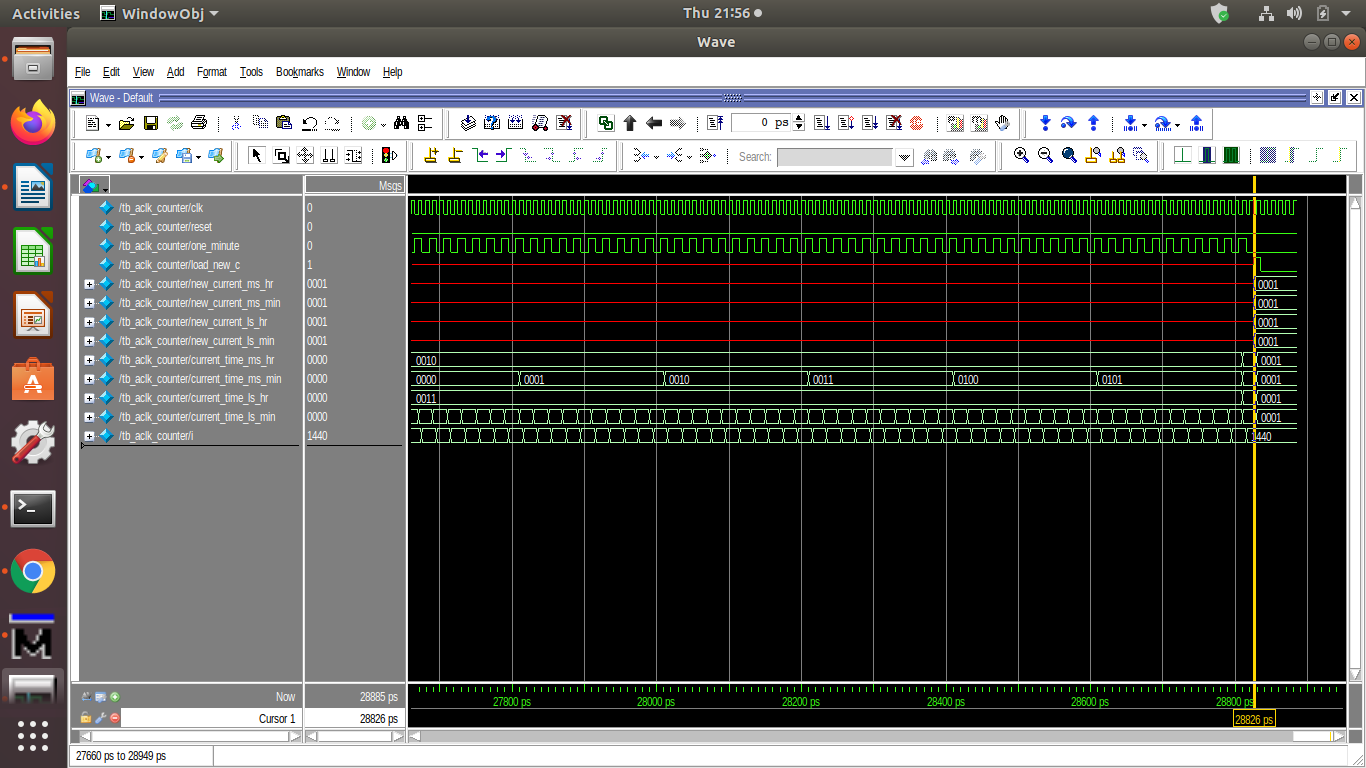
load(1,1,1,1);

#50 $finish;

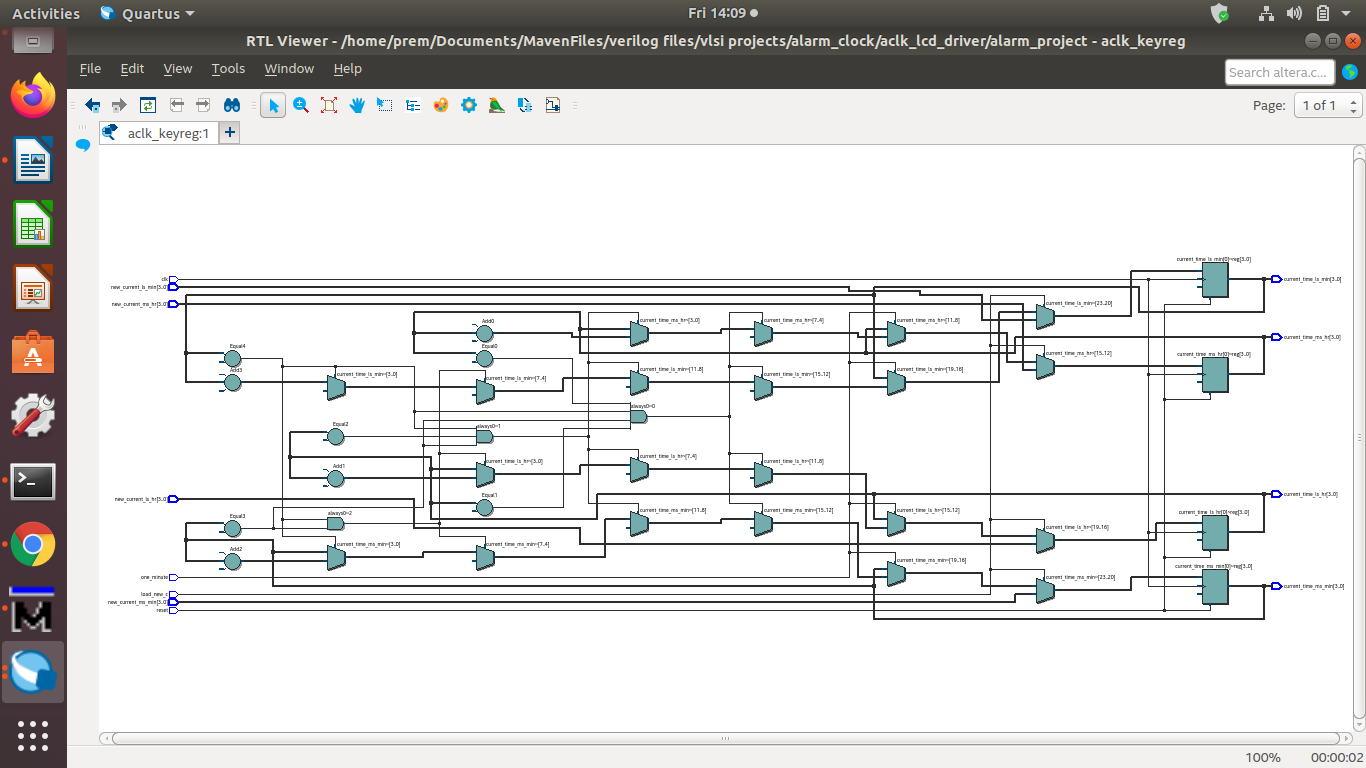
end

endmodule

COUNTER\_SIMULATION



COUNTER\_SYNTHESIS



//LCD\_DRIVER\_MODULE

module aclk\_lcd\_driver(show\_a,show\_new\_time,alarm\_time,current\_time,key,

display\_time,sound\_alarm);

input show\_a,show\_new\_time;

input [3:0]alarm\_time,current\_time,key;

output reg[7:0]display\_time;

output reg sound\_alarm;

function [7:0]\_display\_time(input[3:0] in\_time);

parameter ZERO=4'b0000,ONE=4'b0001,TWO=4'b0010,THREE=4'b0011,FOUR=4'b0100,

FIVE=4'b0101,SIX=4'b0110,SEVEN=4'b0111,EIGHT=4'b1000,NINE=4'b1001;

begin

case(in\_time)

ZERO : \_display\_time=8'h30;

ONE : \_display\_time=8'h31;

TWO : \_display\_time=8'h32;

THREE : \_display\_time=8'h33;

FOUR : \_display\_time=8'h34;

FIVE : \_display\_time=8'h35;

SIX : \_display\_time=8'h36;

SEVEN : \_display\_time=8'h37;

EIGHT : \_display\_time=8'h38;

NINE : \_display\_time=8'h39;

default: \_display\_time=8'h3A;

endcase

end

endfunction

always@(\*)

begin

if(!show\_a && !show\_new\_time)

display\_time=\_display\_time(current\_time);

else if(!show\_a && show\_new\_time)

display\_time=\_display\_time(key);

else if(show\_a && !show\_new\_time)

display\_time=\_display\_time(alarm\_time);

else

display\_time=\_display\_time(current\_time);

if(alarm\_time==current\_time)

sound\_alarm=1'b1;

else

sound\_alarm=1'b0;

end

endmodule

//LCD\_DRIVER\_TESTBENCH

module tb\_aclk\_lcd\_driver;

reg show\_a,show\_new\_time;

reg [3:0]alarm\_time,current\_time,key;

wire [7:0]display\_time;

wire sound\_alarm;

aclk\_lcd\_driver dut(.show\_a(show\_a),.show\_new\_time(show\_new\_time),

.alarm\_time(alarm\_time),.current\_time(current\_time),.key(key),

.display\_time(display\_time),.sound\_alarm(sound\_alarm));

initial

begin

show\_a=0;show\_new\_time=0;

current\_time=4'b0010;

alarm\_time=4'b0011;

key=4'b0100;

#10 show\_a=0;show\_new\_time=1;

current\_time=4'b0010;

alarm\_time=4'b0011;

key=4'b0100;

#10 show\_a=1;show\_new\_time=0;

current\_time=4'b0010;

alarm\_time=4'b0011;

key=4'b0100;

#10 show\_a=1;show\_new\_time=1;

current\_time=4'b0010;

alarm\_time=4'b0011;

key=4'b0100;

#10 show\_a=1;show\_new\_time=1;

current\_time=4'b0010;

alarm\_time=4'b0010;

key=4'b0100;

#10 show\_a=1;show\_new\_time=0;

current\_time=4'b0110;

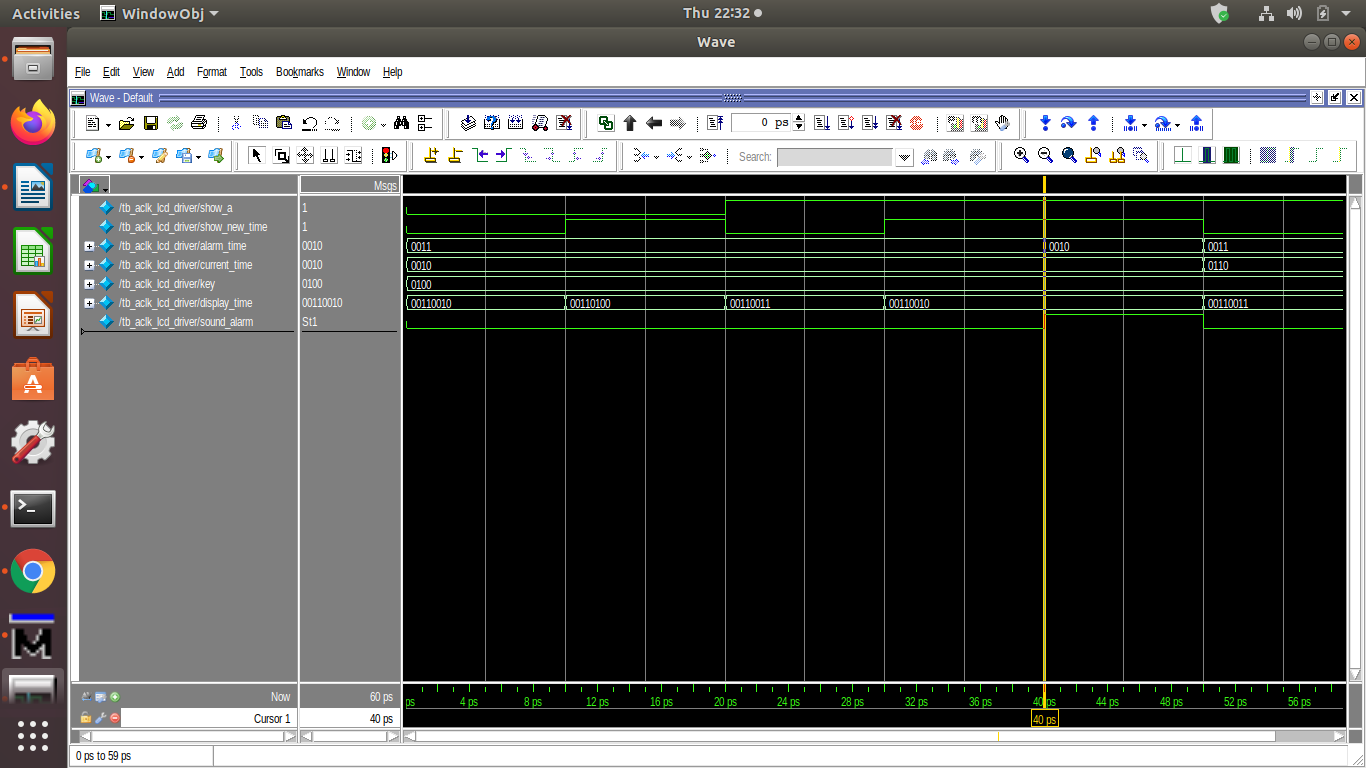
alarm\_time=4'b0011;

key=4'b0100;

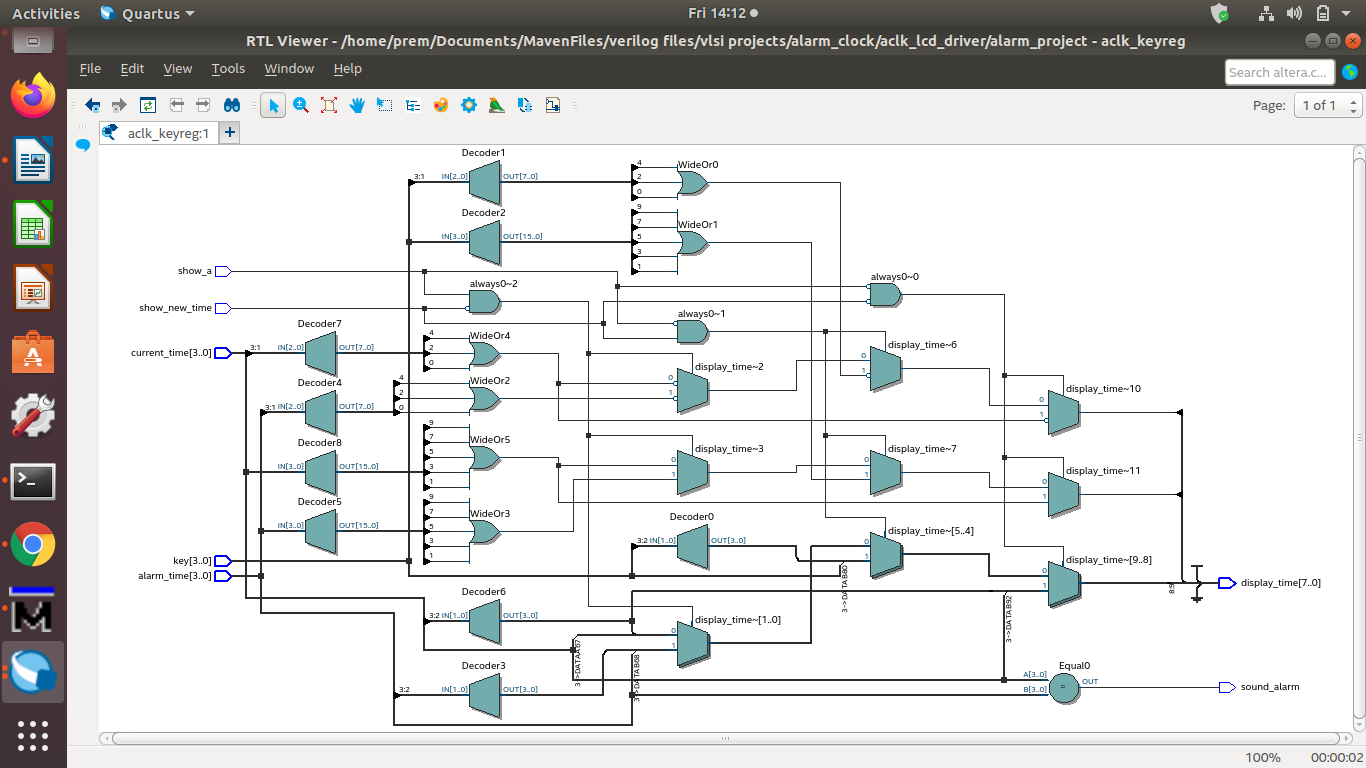
#10 $finish;

end

endmodule

LCD\_DRIVER\_SIMULATION

LCD\_DRIVER\_SYNTHESIS



//LCD\_DRIVER\_UNIT

module aclk\_lcd\_driver\_unit(show\_a,show\_new\_time,

alarm\_time\_ms\_hr,current\_time\_ms\_hr,key\_ms\_hr,

alarm\_time\_ms\_min,current\_time\_ms\_min,key\_ms\_min,

alarm\_time\_ls\_hr,current\_time\_ls\_hr,key\_ls\_hr,

alarm\_time\_ls\_min,current\_time\_ls\_min,key\_ls\_min,

display\_time\_ms\_hr,display\_time\_ms\_min,

display\_time\_ls\_hr,display\_time\_ls\_min,

sound\_alarm);

input show\_a,show\_new\_time;

input [3:0] alarm\_time\_ms\_hr,current\_time\_ms\_hr,key\_ms\_hr;

input [3:0] alarm\_time\_ms\_min,current\_time\_ms\_min,key\_ms\_min;

input [3:0] alarm\_time\_ls\_hr,current\_time\_ls\_hr,key\_ls\_hr;

input [3:0] alarm\_time\_ls\_min,current\_time\_ls\_min,key\_ls\_min;

output [7:0]display\_time\_ms\_hr,display\_time\_ms\_min;

output [7:0]display\_time\_ls\_hr,display\_time\_ls\_min;

output wand sound\_alarm;

wire sound\_alarm1,sound\_alarm2,sound\_alarm3,sound\_alarm4;

aclk\_lcd\_driver LCD\_DR1(show\_a,show\_new\_time,

alarm\_time\_ms\_hr,current\_time\_ms\_hr,key\_ms\_hr,

display\_time\_ms\_hr,sound\_alarm1);

aclk\_lcd\_driver LCD\_DR2(show\_a,show\_new\_time,

alarm\_time\_ms\_min,current\_time\_ms\_min,key\_ms\_min,

display\_time\_ms\_min,sound\_alarm2);

aclk\_lcd\_driver LCD\_DR3(show\_a,show\_new\_time,

alarm\_time\_ls\_hr,current\_time\_ls\_hr,key\_ls\_hr,

display\_time\_ls\_hr,sound\_alarm3);

aclk\_lcd\_driver LCD\_DR4(show\_a,show\_new\_time,

alarm\_time\_ls\_min,current\_time\_ls\_min,key\_ls\_min,

display\_time\_ls\_min,sound\_alarm4);

assign sound\_alarm=sound\_alarm1 && sound\_alarm2 && sound\_alarm3 && sound\_alarm4;

endmodule

//LCD\_DRIVER\_UNIT\_TESTBENCH

module tb\_aclk\_lcd\_driver\_unit;

reg show\_a,show\_new\_time;

reg[3:0] a\_t\_ms\_hr,c\_t\_ms\_hr,k\_ms\_hr;

reg[3:0] a\_t\_ms\_min,c\_t\_ms\_min,k\_ms\_min;

reg[3:0] a\_t\_ls\_hr,c\_t\_ls\_hr,k\_ls\_hr;

reg[3:0] a\_t\_ls\_min,c\_t\_ls\_min,k\_ls\_min;

wire[7:0] disp\_t\_ms\_hr,disp\_t\_ms\_min;

wire[7:0] disp\_t\_ls\_hr,disp\_t\_ls\_min;

wire sound\_alarm;

aclk\_lcd\_driver\_unit DUT(.show\_a(show\_a),.show\_new\_time(show\_new\_time),

.alarm\_time\_ms\_hr(a\_t\_ms\_hr),.current\_time\_ms\_hr(c\_t\_ms\_hr),.key\_ms\_hr(k\_ms\_hr),

.alarm\_time\_ms\_min(a\_t\_ms\_min),.current\_time\_ms\_min(c\_t\_ms\_min),.key\_ms\_min(k\_ms\_min),

.alarm\_time\_ls\_hr(a\_t\_ls\_hr),.current\_time\_ls\_hr(c\_t\_ls\_hr),.key\_ls\_hr(k\_ls\_hr),

.alarm\_time\_ls\_min(a\_t\_ls\_min),.current\_time\_ls\_min(c\_t\_ls\_min),.key\_ls\_min(k\_ls\_min),

.display\_time\_ms\_hr(disp\_t\_ms\_hr),.display\_time\_ms\_min(disp\_t\_ms\_min),

.display\_time\_ls\_hr(disp\_t\_ls\_hr),.display\_time\_ls\_min(disp\_t\_ls\_min),

.sound\_alarm(sound\_alarm));

task stimulus\_alarm(input[3:0]i,j,k,l);

begin

a\_t\_ms\_hr=i;

a\_t\_ms\_min=k;

a\_t\_ls\_hr=j;

a\_t\_ls\_min=l;

end

endtask

task stimulus\_current(input[3:0]i,j,k,l);

begin

c\_t\_ms\_hr=i;

c\_t\_ms\_min=k;

c\_t\_ls\_hr=j;

c\_t\_ls\_min=l;

end

endtask

task stimulus\_key(input[3:0]i,j,k,l);

begin

k\_ms\_hr=i;

k\_ms\_min=k;

k\_ls\_hr=j;

k\_ls\_min=l;

end

endtask

task delay;

#10;

endtask

task sel\_time(input[1:0]i);

begin

show\_a=i[0];show\_new\_time=i[1];

end

endtask

initial

begin

//initialize

sel\_time(0);

stimulus\_current(0,0,0,0);

stimulus\_alarm(0,0,0,0);

stimulus\_key(0,0,0,0);

delay;

//current time

sel\_time(0);

stimulus\_current(0,9,0,0);

stimulus\_alarm(0,0,8,0);

stimulus\_key(0,0,4,0);

delay;

//alarm time

sel\_time(1);

delay;

//key time

sel\_time(2);

delay;

//testing alarm sound

stimulus\_current(0,0,1,0);

stimulus\_alarm(0,0,1,0);

delay;

//testing alarm sound

stimulus\_current(0,1,1,0);

stimulus\_alarm(0,0,1,0);

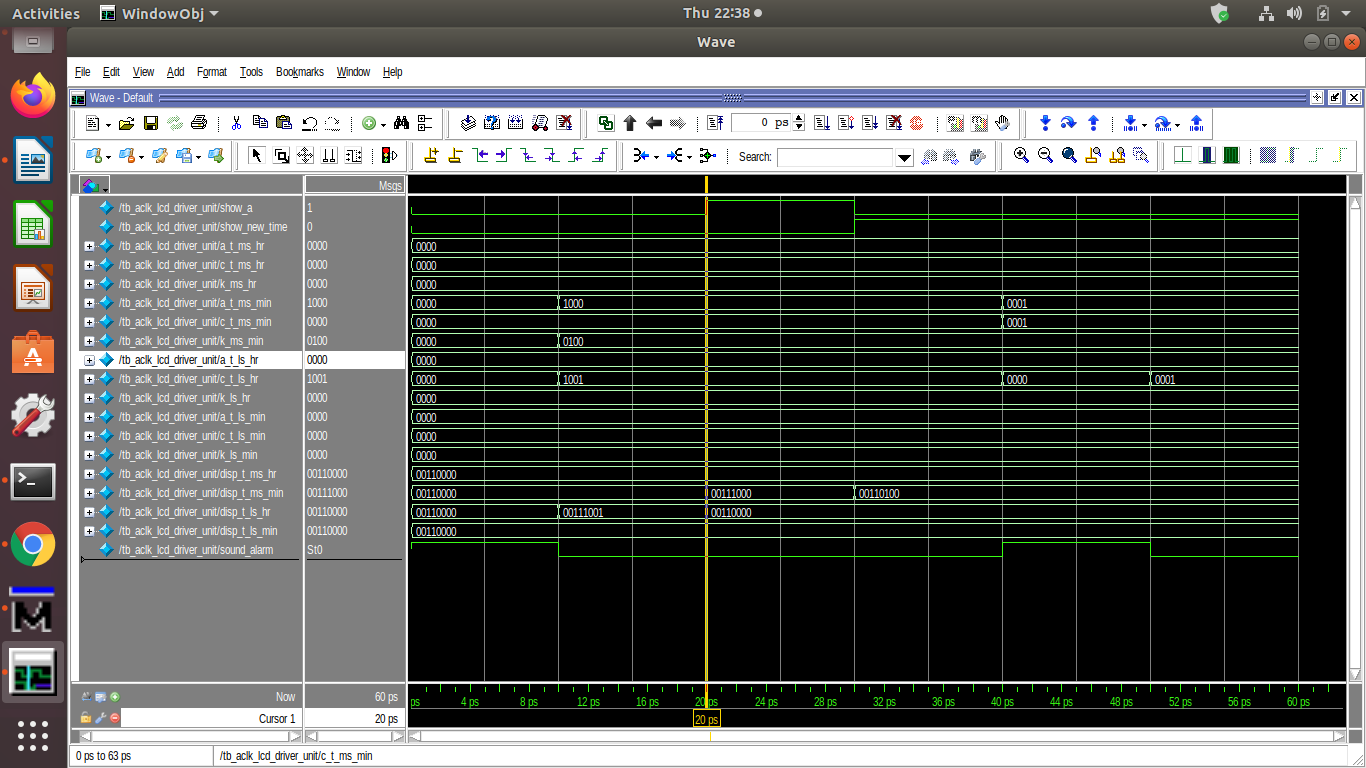
delay;

$finish;

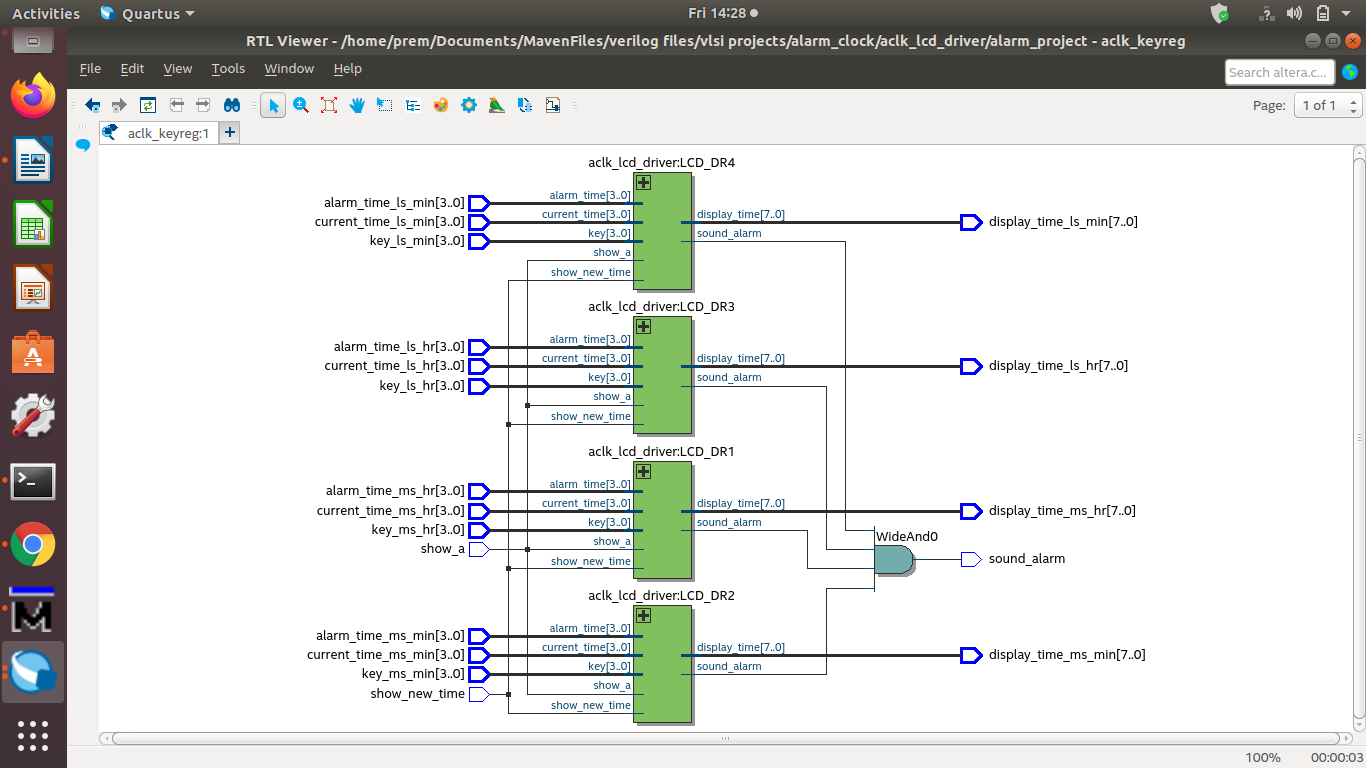
end

endmodule

LCD\_DRIVER\_UNIT\_SIMULATION



LCD\_DRIVER\_UNIT\_SYNTHESIS



//ALARM\_TOP

module alarm\_clk\_rtl(clk,reset,alarm\_button,time\_button,key,fast\_watch,

sound\_alarm,

display\_ms\_hr,display\_ls\_hr,display\_ms\_min,display\_ls\_min);

input clk,reset,alarm\_button,time\_button,fast\_watch;

input [3:0]key;

output reg sound\_alarm;

output reg [7:0]display\_ms\_hr,display\_ls\_hr,display\_ms\_min,display\_ls\_min;

wire sound\_alarm1;

wire show\_a,show\_new\_time;

wire [3:0] alarm\_time\_ms\_hr,current\_time\_ms\_hr,key\_ms\_hr;

wire [3:0] alarm\_time\_ms\_min,current\_time\_ms\_min,key\_ms\_min;

wire [3:0] alarm\_time\_ls\_hr,current\_time\_ls\_hr,key\_ls\_hr;

wire [3:0] alarm\_time\_ls\_min,current\_time\_ls\_min,key\_ls\_min;

wire [7:0]display\_time\_ms\_hr,display\_time\_ms\_min;

wire [7:0]display\_time\_ls\_hr,display\_time\_ls\_min;

wire load\_new\_a;

wire new\_alarm\_ms\_hr,new\_alarm\_ms\_min;

wire new\_alarm\_ls\_hr,new\_alarm\_ls\_min;

wire one\_minute,reset\_count;

wire[3:0] new\_current\_ms\_hr,new\_current\_ms\_min,new\_current\_ls\_hr,new\_current\_ls\_min;

wire load\_new\_c;

wire[3:0] key\_buffer\_ms\_hr,key\_buffer\_ms\_min,key\_buffer\_ls\_hr,key\_buffer\_ls\_min;

wire shift;

aclk\_lcd\_driver\_unit LCD\_DRIVER(show\_a,show\_new\_time,

alarm\_time\_ms\_hr,current\_time\_ms\_hr,key\_ms\_hr,

alarm\_time\_ms\_min,current\_time\_ms\_min,key\_ms\_min,

alarm\_time\_ls\_hr,current\_time\_ls\_hr,key\_ls\_hr,

alarm\_time\_ls\_min,current\_time\_ls\_min,key\_ls\_min,

display\_time\_ms\_hr,display\_time\_ms\_min,

display\_time\_ls\_hr,display\_time\_ls\_min,

sound\_alarm1);

aclk\_areg ALARM\_REG(clk,reset,load\_new\_a,

new\_alarm\_ms\_hr,new\_alarm\_ms\_min,

new\_alarm\_ls\_hr,new\_alarm\_ls\_min,

alarm\_time\_ms\_hr,alarm\_time\_ms\_min,

alarm\_time\_ls\_hr,alarm\_time\_ls\_min);

aclk\_timegen TIME\_GEN(clk,reset,reset\_count,fast\_watch,one\_second,one\_minute);

aclk\_counter COUNTER(clk,reset,one\_minute,load\_new\_c,

new\_current\_ms\_hr,new\_current\_ms\_min,new\_current\_ls\_hr,new\_current\_ls\_min,

current\_time\_ms\_hr,current\_time\_ms\_min,current\_time\_ls\_hr,current\_time\_ls\_min);

aclk\_keyreg KEY\_REG(clk,reset,key,shift,

key\_buffer\_ms\_hr,key\_buffer\_ms\_min,key\_buffer\_ls\_hr,key\_buffer\_ls\_min);

aclk\_controller CONTROLLER(clk,reset,one\_second,alarm\_button,time\_button,key,

reset\_count,load\_new\_c,show\_new\_time,show\_a,load\_new\_a,shift);

always@(\*)

begin

if(reset)

begin

sound\_alarm<=0;

display\_ms\_hr<=0;display\_ls\_hr<=0;

display\_ms\_min<=0;display\_ls\_min<=0;

end

else

begin

sound\_alarm<=sound\_alarm1;

display\_ms\_hr<=display\_time\_ms\_hr;display\_ls\_hr<=display\_time\_ls\_hr;

display\_ms\_min<=display\_time\_ms\_min;display\_ls\_min<=display\_time\_ls\_min;

end

end

assign one\_minute=(fast\_watch)?one\_second:one\_minute;

endmodule

//ALARM\_TOP\_TESTBENCH

`timescale 1ms/1ns

module tb\_alarm\_clk;

reg clk,reset,alarm\_button,time\_button,fast\_watch;

reg [3:0]key;

wire sound\_alarm;

wire [7:0]display\_ms\_hr,display\_ls\_hr,display\_ms\_min,display\_ls\_min;

alarm\_clk\_rtl DUT(clk,reset,alarm\_button,time\_button,key,fast\_watch,sound\_alarm,

display\_ms\_hr,display\_ls\_hr,display\_ms\_min,display\_ls\_min);

integer l,k;

always

begin

#1.953125 clk=1'b0;

#1.953125 clk=1'b1;

end

task rst;

begin

reset=1;

@(negedge clk) reset=0;

end

endtask

task \_key(input [3:0]a);

begin

@(negedge clk);

key=a;

@(negedge clk) ;

end

endtask

task \_ta\_button(input t,a);

begin

@(negedge clk) time\_button=t;

alarm\_button=a;

@(negedge clk) time\_button=1'b0;

alarm\_button=1'b0;

end

endtask

initial

begin

rst;

\_key(4'd0);

\_key(4'd10);

\_key(4'd9);

\_key(4'd10);

\_key(4'd5);

\_key(4'd10);

\_key(4'd3);

\_key(4'd10);

\_ta\_button(1'b1,1'b0);

\_key(4'd0);

\_key(4'd10);

\_key(4'd9);

\_key(4'd10);

\_key(4'd5);

\_key(4'd10);

\_key(4'd5);

\_key(4'd10);

\_ta\_button(1'b0,1'b1);

fast\_watch=1'b1;

for(k=0;k<3;k=k+1)

begin

for(l=0;l<256;l=l+1)

#3.906250;

end

\_key(4'd0);

\_key(4'd10);

\_key(4'd9);

\_key(4'd10);

\_key(4'd5);

\_key(4'd10);

\_key(4'd3);

\_key(4'd10);

\_ta\_button(1'b1,1'b0);

fast\_watch=1'b0;

for(k=0;k<122;k=k+1)

begin

for(l=0;l<256;l=l+1)

#3.906250;

end

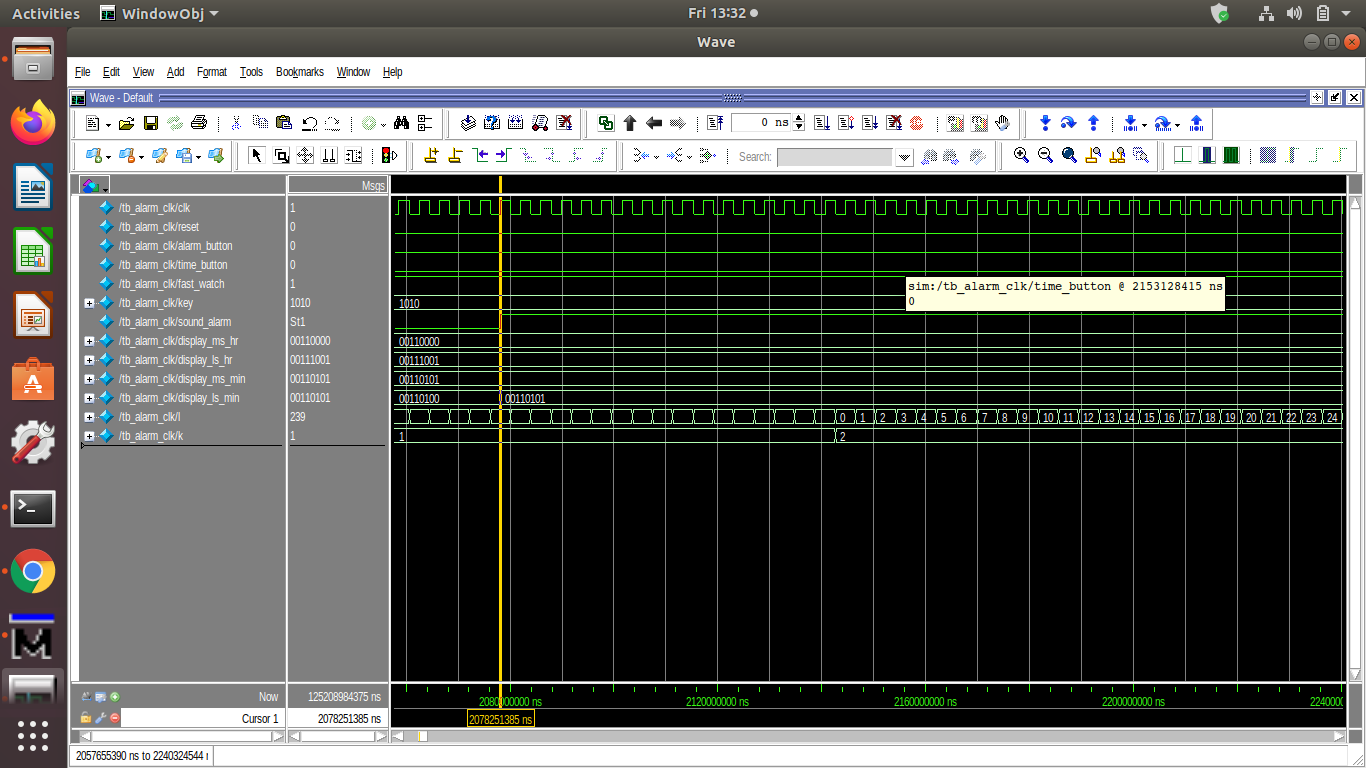
$finish;

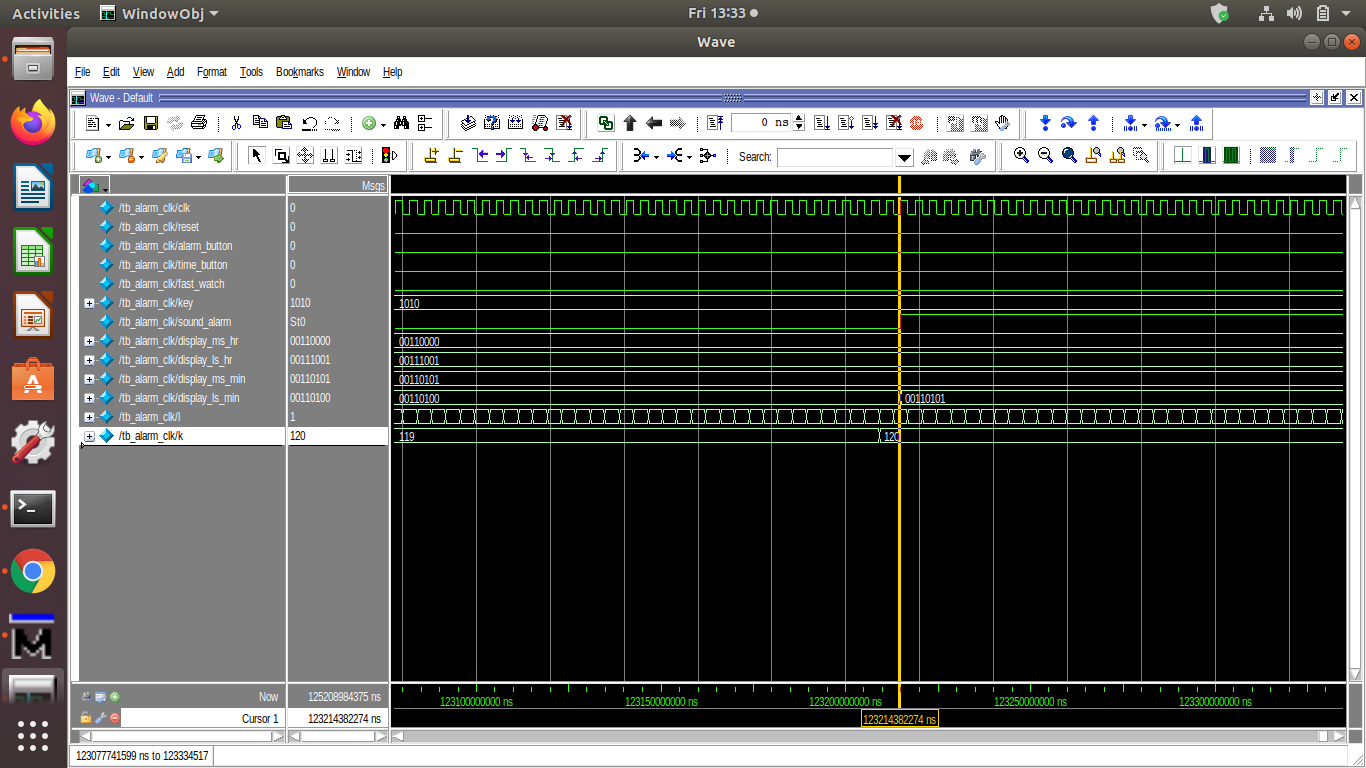
end

endmodule

ALARM\_TOP\_SIMULATIONS

(fast\_watch=1)



(fast\_watch=0)

ALARM\_TOP\_SYNTHESIS

