Clock Generation with Alarm using Verilog

1. Objective

The objective of this mini project is to design and implement a digital clock with an alarm feature using Verilog HDL.

2. Software Detail

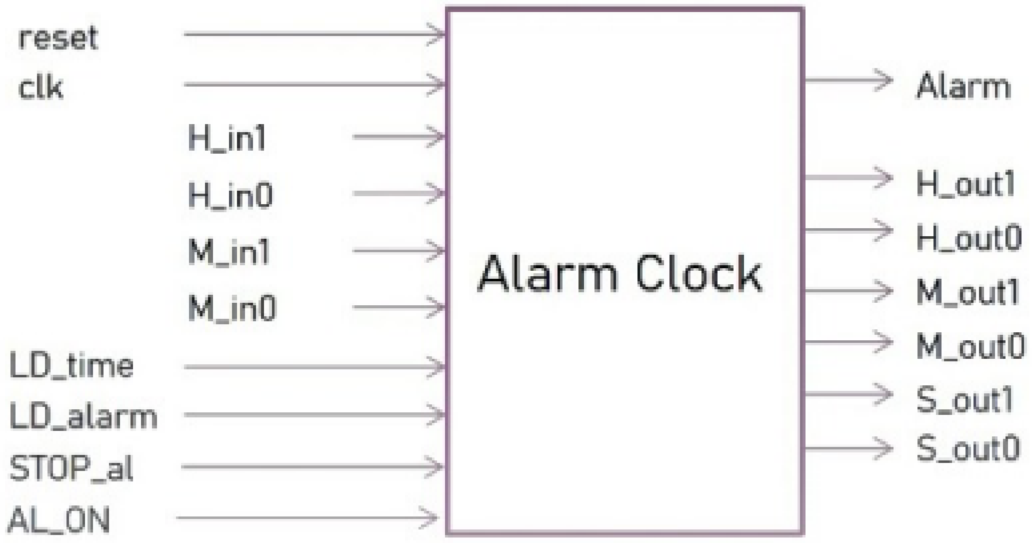
Equipment's:
Computer with Xilinx
Software: Xilinx
Synthesis tool and Simulation tool: Xilinx Vivado

3. Abstract /Introduction

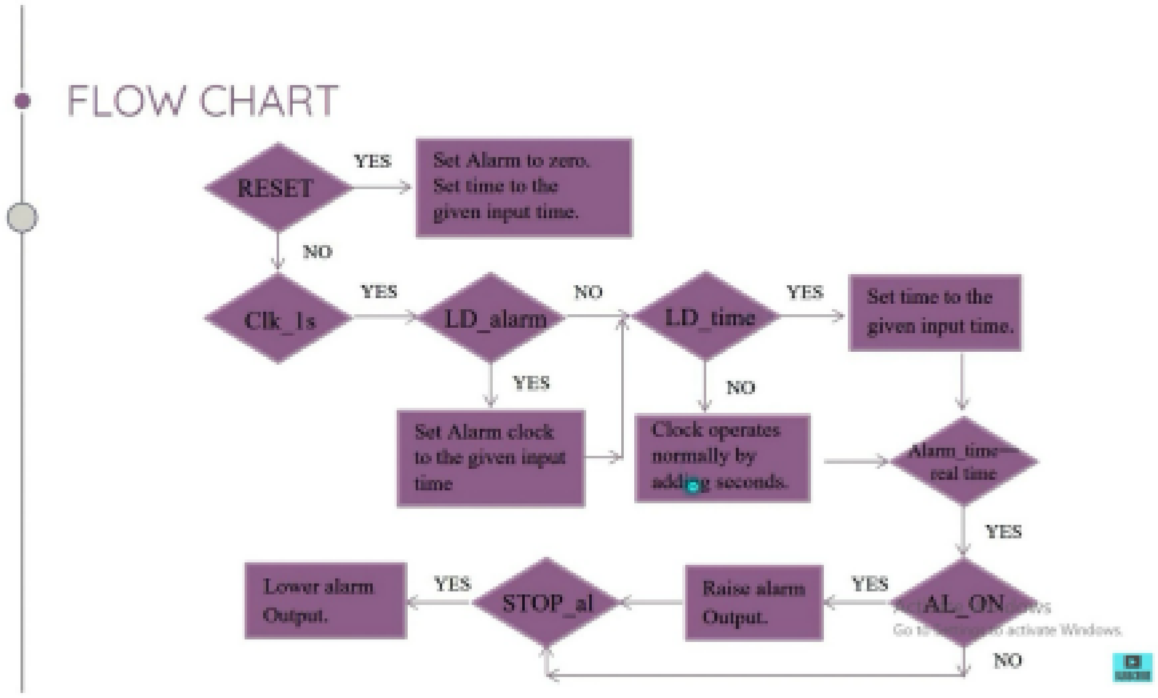
The project involves the implementation of a digital clock with an alarm function using Verilog HDL. The clock includes components such as registers for storing time values, counters for tracking seconds, and logic for comparing the current time with the set alarm time. The clock operates by incrementing time every second and displays the time on a seven-segment display. Users can set the time and alarm time through input signals, and the alarm triggers when the set alarm time matches the current time. The features include:

1. Clock generation
2. Initializing clock time to a particular value
3. Setting time for alarm.
4. Enabling and disabling alarm
5. Stopping alarm

4. Block Diagram



5. Flowchart


FLOW CHART
RESET
YES
Set Alarm to zero. Set time to the given input time.
NO
Clk_1s
YES
LD_alarm
NO
LD_time
YES
Set time to the given input time.
YES
Set Alarm clock to the given input time
NO
Clock operates normally by adding seconds.
Alarm_time== real time
YES
AL_ON
YES
Raise alarm Output.
STOP_al
YES
Lower alarm Output.
NO

6. WORKING

The clock operates by incrementing time every second based on a 1 Hz clock signal. Time values for hours, minutes, and seconds are stored in registers and updated accordingly.
Users can set the time and alarm time using input signals. When the LD_time signal is activated, the clock updates the time registers with the user-defined values. Similarly, when LD_alarm is activated, the alarm registers are updated. The clock continuously compares the current time with the alarm time. When they match and the AL_ON signal is activated, the alarm is triggered.
The clock can be stopped by activating the STOP_al signal**.**

7. VERILOG CODE

```
module Aclock(
input reset,
input clk,
input [1:0] H_in1,
input [3:0] H_in0,
input [3:0] M_in1,
input [3:0] M_in0,
input LD_time,
input LD_alarm,
input STOP_al,
input AL_ON,
output reg Alarm,
output [1:0] H_out1,
output [3:0] H_out0,
output [3:0] M_out1,
output [3:0] M_out0,
output [3:0] S_out1,
output [3:0] S_out0);

reg clk_1s;
reg [3:0] tmp_1s;
reg [5:0] tmp_hour, tmp_minute, tmp_second;
reg [1:0] c_hour1,a_hour1;
reg [3:0] c_hour0,a_hour0;
reg [3:0] c_min1,a_min1;
reg [3:0] c_min0,a_min0;
reg [3:0] c_sec1,a_sec1;
reg [3:0] c_sec0,a_sec0;

function [3:0] mod_10;
 input [5:0] number;
 begin
 mod_10 = (number >=50) ? 5 : ((number >= 40)? 4 :((number >= 30)? 3 :((number
    >= 20)? 2 :((number >= 10)? 1 :0))));
 end
endfunction

always @(posedge clk_1s or posedge reset )
 begin
 if(reset) begin
 a_hour1 <= 2'b00;
 a_hour0 <= 4'b0000;
 a_min1 <= 4'b0000;
 a_min0 <= 4'b0000;
 a_sec1 <= 4'b0000;
 a_sec0 <= 4'b0000;
 tmp_hour <= H_in1*10 + H_in0;
 tmp_minute <= M_in1*10 + M_in0;
 tmp_second <= 0;
 end
 else begin
 if(LD_alarm) begin
 a_hour1 <= H_in1;
 a_hour0 <= H_in0;
 a_min1 <= M_in1;
 a_min0 <= M_in0;
```

```
a_sec1 <= 4'b0000;
a_sec0 <= 4'b0000;
end
if(LD_time) begin
tmp_hour <= H_in1*10 + H_in0;
tmp_minute <= M_in1*10 + M_in0;
tmp_second <= 0;
end
else begin
tmp_second <= tmp_second + 1;
if(tmp_second >=59) begin
tmp_minute <= tmp_minute + 1;
tmp_second <= 0;
if(tmp_minute >=59) begin
tmp_minute <= 0;
tmp_hour <= tmp_hour + 1;
if(tmp_hour >= 24) begin
tmp_hour <= 0;
end
end
end

end
end
end

always @(posedge clk or posedge reset)
begin
if(reset)
begin
tmp_1s <= 0;
clk_1s <= 0;
end
else begin
tmp_1s <= tmp_1s + 1;
```

```
 if(tmp_1s <= 5)
 clk_1s <= 0;
 else if (tmp_1s >= 10) begin
 clk_1s <= 1;
 tmp_1s <= 1;
 end
 else
 clk_1s <= 1;
 end
 end
 always @(*) begin

 if(tmp_hour>=20) begin
 c_hour1 = 2;
 end
 else begin
 if(tmp_hour >=10)
 c_hour1 = 1;
 else
 c_hour1 = 0;
 end
 c_hour0 = tmp_hour - c_hour1*10;
 c_min1 = mod_10(tmp_minute);
 c_min0 = tmp_minute - c_min1*10;
 c_sec1 = mod_10(tmp_second);
 c_sec0 = tmp_second - c_sec1*10;
 end


always @(posedge clk_1s or posedge reset)
begin
 if(reset)
 Alarm <=0;
 else begin
 if({a_hour1,a_hour0,a_min1,a_min0}=={c_hour1,c_hour0,c_min1,c_min0})
```

```
begin
if(AL_ON) Alarm <= 1;
end
if(STOP_al) Alarm <=0;

end
end
assign H_out1 = c_hour1;
assign H_out0 = c_hour0;
assign M_out1 = c_min1;
assign M_out0 = c_min0;
assign S_out1 = c_sec1;
assign S_out0 =

    c_sec0; endmodule
```

TEST BENCH CODE:

```
module Testbench;
reg reset;
 reg clk;
 reg [1:0] H_in1;
 reg [3:0] H_in0;
 reg [3:0] M_in1;
 reg [3:0] M_in0;
 reg LD_time;
 reg LD_alarm;
 reg STOP_al;
 reg AL_ON;

 // Outputs
 wire Alarm;
 wire [1:0] H_out1;
 wire [3:0] H_out0;
 wire [3:0] M_out1;
 wire [3:0] M_out0;
wire [3:0] S_out1;
wire [3:0] S_out0;


Aclock uut (
.reset(reset),
.clk(clk),
.H_in1(H_in1),
.H_in0(H_in0),
.M_in1(M_in1),
.M_in0(M_in0),
.LD_time(LD_time),
.LD_alarm(LD_alarm),
.STOP_al(STOP_al),
.AL_ON(AL_ON),
.Alarm(Alarm),
.H_out1(H_out1),
.H_out0(H_out0),
.M_out1(M_out1),
.M_out0(M_out0),
.S_out1(S_out1),
.S_out0(S_out0)
);

initial begin
clk = 0;
forever #50000000 clk = ~clk;
end
initial begin
// Initialize Inputs
reset = 1;
H_in1 = 1;
H_in0 = 0;
M_in1 = 1;
M_in0 = 9;
LD_time = 0;
LD_alarm = 0;
STOP_al = 0;
AL_ON = 0;

#1000000000;
```

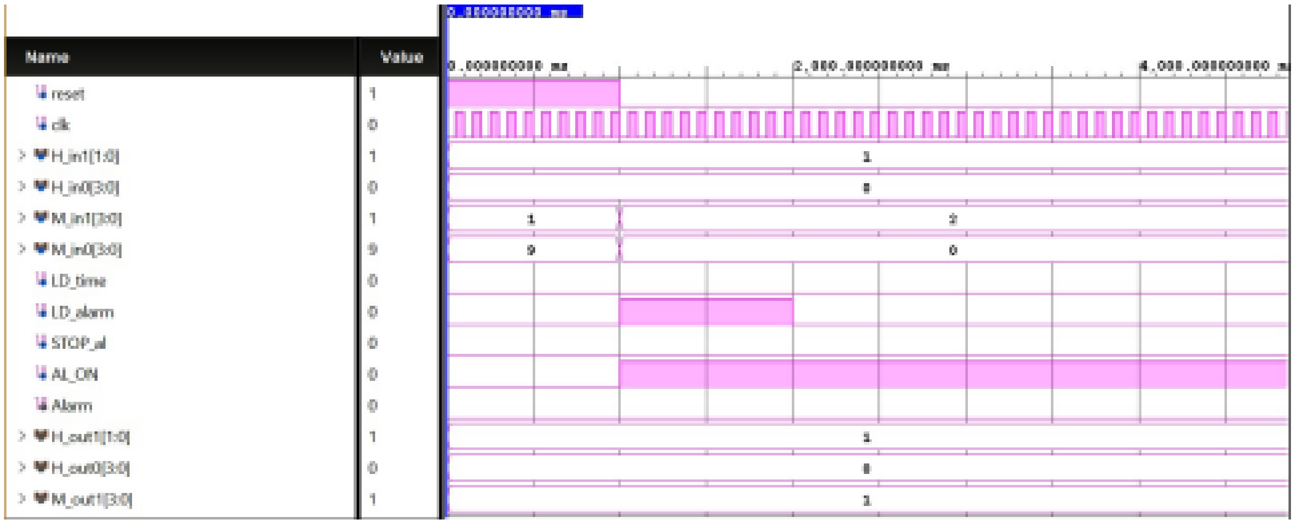
```
 reset = 0;
 H_in1 = 1;
 H_in0 = 0;
 M_in1 = 2;
 M_in0 = 0;
 LD_time = 0;
 LD_alarm = 1;
 STOP_al = 0;
 AL_ON = 1;


#1000000000
 ; reset = 0;
 H_in1 = 1;
 H_in0 = 0;
 M_in1 = 2;
 M_in0 = 0;
 LD_time = 0;
 LD_alarm =
 0; STOP_al =
 0; AL_ON =
 1;

wait(Alarm);
#1000000000
;
#1000000000
;
#1000000000
;
#1000000000
;
#1000000000
;
#1000000000
; STOP_al =
1;


 end
endmodule
```

8. OUTPUT:



9. Result

Upon synthesizing and simulating the Verilog code using suitable software tools, the digital clock with an alarm feature should demonstrate functional behavior. Users can expect to observe the clock accurately displaying the current time and responding appropriately to user inputs for setting the time and alarm. Additionally, the alarm should trigger as expected when the set alarm time matches the current time, provided the alarm function is enabled. Verification through simulation and synthesis helps ensure the correctness and reliability of the clock's operation before potential deployment on hardware platforms.

10. Conclusion

The digital clock with an alarm feature is successfully implemented using Verilog HDL. The project demonstrates the application of sequential and combinational logic in digital design. Further improvements could include additional features such as a snooze function, a user-friendly interface, or integration with external peripherals.