**DIGITAL CLOCK ALARM USING VERILOG**

**Aim:** Verilog code for simple alarm with 24-hour format in which we can set time set alarm and atop alarm.

**Source code:**

**module aclock (**

**input reset, // Active high reset**

**input clk, // 10Hz clock input**

**input [1:0] H\_in1, // Most significant hour digit input**

**input [3:0] H\_in0, // Least significant hour digit input**

**input [3:0] M\_in1, // Most significant minute digit input**

**input [3:0] M\_in0, // Least significant minute digit input**

**input LD\_time, // Load time signal**

**input LD\_alarm, // Load alarm signal**

**input STOP\_al, // Stop alarm signal**

**input AL\_ON, // Alarm on signal**

**output reg Alarm, // Alarm output**

**output [1:0] H\_out1, // Most significant hour digit output**

**output [3:0] H\_out0, // Least significant hour digit output**

**output [3:0] M\_out1, // Most significant minute digit output**

**output [3:0] M\_out0, // Least significant minute digit output**

**output [3:0] S\_out1, // Most significant second digit output**

**output [3:0] S\_out0 // Least significant second digit output**

**);**

**reg [3:0] clk\_div;**

**reg clk\_1s; // 1s clock signal**

**reg [5:0] hour, minute, second; // Clock time**

**reg [5:0] alarm\_hour, alarm\_minute; // Alarm time**

**// Generate 1s clock from 10Hz clock**

**always @(posedge clk or posedge reset) begin**

**if (reset) begin**

**clk\_div <= 0;**

**clk\_1s <= 0;**

**end else begin**

**clk\_div <= clk\_div + 1;**

**if (clk\_div == 5) begin**

**clk\_1s <= ~clk\_1s;**

**clk\_div <= 0;**

**end**

**end**

**end**

**always @(posedge clk\_1s or posedge reset) begin**

**if (reset) begin**

**hour <= H\_in1 \* 10 + H\_in0;**

**minute <= M\_in1 \* 10 + M\_in0;**

**second <= 0;**

**alarm\_hour <= 0;**

**alarm\_minute <= 0;**

**Alarm <= 0;**

**end else begin**

**if (LD\_time) begin**

**hour <= H\_in1 \* 10 + H\_in0;**

**minute <= M\_in1 \* 10 + M\_in0;**

**second <= 0;**

**end else if (LD\_alarm) begin**

**alarm\_hour <= H\_in1 \* 10 + H\_in0;**

**alarm\_minute <= M\_in1 \* 10 + M\_in0;**

**end else begin**

**second <= second + 1;**

**if (second >= 59) begin**

**second <= 0;**

**minute <= minute + 1;**

**if (minute >= 59) begin**

**minute <= 0;**

**hour <= hour + 1;**

**if (hour >= 24) hour <= 0;**

**end**

**end**

**end**

**end**

**end**

**always @(posedge clk\_1s or posedge reset) begin**

**if (reset) begin**

**Alarm <= 0;**

**end else begin**

**if (AL\_ON && (hour == alarm\_hour) && (minute == alarm\_minute) && (second == 0)) begin**

**Alarm <= 1;**

**end**

**if (STOP\_al) begin**

**Alarm <= 0;**

**end**

**end**

**end**

**assign H\_out1 = hour / 10;**

**assign H\_out0 = hour % 10;**

**assign M\_out1 = minute / 10;**

**assign M\_out0 = minute % 10;**

**assign S\_out1 = second / 10;**

**assign S\_out0 = second % 10;**

**endmodule**

**Test bench:**

**`timescale 1ns / 1ps**

**module aclock\_tb;**

**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;**

**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 begin**

**#50000000 clk = ~clk; // Toggle every 50ms => 10Hz**

**end**

**end**

**initial begin**

**reset = 1;**

**H\_in1 = 2'b00;**

**H\_in0 = 4'b0000;**

**M\_in1 = 4'b0000;**

**M\_in0 = 4'b0000;**

**LD\_time = 0;**

**LD\_alarm = 0;**

**STOP\_al = 0;**

**AL\_ON = 0;**

**#100;**

**reset = 0;**

**H\_in1 = 2'b01;**

**H\_in0 = 4'b0010;**

**M\_in1 = 4'b0011;**

**M\_in0 = 4'b0100;**

**LD\_time = 1;**

**#100;**

**LD\_time = 0;**

**#5000;**

**H\_in1 = 2'b01;**

**H\_in0 = 4'b0010;**

**M\_in1 = 4'b0011;**

**M\_in0 = 4'b0101;**

**LD\_alarm = 1;**

**#100;**

**LD\_alarm = 0;**

**AL\_ON = 1;**

**#10000;**

**STOP\_al = 1;**

**#100;**

**STOP\_al = 0;**

**// Finish simulation**

**#2000;**

**$finish;**

**end**

**endmodule**

**Arduino:**

**#include <LiquidCrystal.h>**

**// Pin Definitions**

**const int button1Pin = 8; // Mode switch button**

**const int button2Pin = 9; // Increment button**

**const int button3Pin = 6; // Select hours or minutes**

**const int alarmButtonPin = 7; // Alarm toggle button**

**const int buzzerPin = 10; // Buzzer**

**// LCD Pin Definitions**

**LiquidCrystal lcd(12, 11, 5, 4, 3, 2);**

**// Time and Alarm Variables**

**unsigned long previousMillis = 0;**

**const unsigned long interval = 1000; // 1 second**

**unsigned long currentMillis = 0;**

**int mode = 0; // 0: Clock, 1: Set Time, 2: Set Alarm**

**int hours = 0, minutes = 0, seconds = 0; // Current time**

**int alarmHours = 0, alarmMinutes = 0; // Alarm time**

**bool alarmOn = false; // Alarm state**

**bool adjustHours = true; // Toggle between hours/minutes**

**// Button State Variables**

**bool button1Pressed = false;**

**bool button2Pressed = false;**

**bool button3Pressed = false;**

**bool alarmButtonPressed = false;**

**// Setup**

**void setup() {**

**lcd.begin(16, 2); // Initialize a 16x2 LCD**

**pinMode(button1Pin, INPUT\_PULLUP);**

**pinMode(button2Pin, INPUT\_PULLUP);**

**pinMode(button3Pin, INPUT\_PULLUP);**

**pinMode(alarmButtonPin, INPUT\_PULLUP);**

**pinMode(buzzerPin, OUTPUT);**

**lcd.print("Initializing...");**

**delay(2000);**

**lcd.clear();**

**}**

**// Loop**

**void loop() {**

**currentMillis = millis();**

**// Handle Buttons**

**handleButtons();**

**// Update Clock**

**if (currentMillis - previousMillis >= interval) {**

**previousMillis = currentMillis;**

**if (mode == 0) { // Only update time in Clock mode**

**seconds++;**

**if (seconds >= 60) {**

**seconds = 0;**

**minutes++;**

**if (minutes >= 60) {**

**minutes = 0;**

**hours = (hours + 1) % 24;**

**}**

**}**

**displayTime();**

**}**

**}**

**// Modes: Set Time or Alarm**

**if (mode == 1) displaySetTime();**

**if (mode == 2) displaySetAlarm();**

**// Check Alarm**

**if (alarmOn) checkAlarm();**

**}**

**// Handle Buttons**

**void handleButtons() {**

**// Mode switch button**

**if (!digitalRead(button1Pin)) { // Button 1 pressed**

**if (!button1Pressed) {**

**button1Pressed = true;**

**mode = (mode + 1) % 3; // Cycle through modes: Clock -> Set Time -> Set Alarm**

**lcd.clear();**

**}**

**} else {**

**button1Pressed = false;**

**}**

**// Increment button**

**if (!digitalRead(button2Pin)) { // Button 2 pressed**

**if (!button2Pressed) {**

**button2Pressed = true;**

**if (mode == 1) incrementTime();**

**else if (mode == 2) incrementAlarm();**

**}**

**} else {**

**button2Pressed = false;**

**}**

**// Toggle hours/minutes adjustment**

**if (!digitalRead(button3Pin)) { // Button 3 pressed**

**if (!button3Pressed) {**

**button3Pressed = true;**

**adjustHours = !adjustHours; // Toggle between hours and minutes**

**lcd.clear();**

**}**

**} else {**

**button3Pressed = false;**

**}**

**// Alarm toggle button**

**if (!digitalRead(alarmButtonPin)) { // Alarm toggle button pressed**

**if (!alarmButtonPressed) {**

**alarmButtonPressed = true;**

**alarmOn = !alarmOn; // Toggle alarm state**

**lcd.clear();**

**displayTime(); // Refresh clock display with alarm state**

**}**

**} else {**

**alarmButtonPressed = false;**

**}**

**}**

**// Increment Time**

**void incrementTime() {**

**seconds = 0; // Reset seconds when setting time**

**if (adjustHours) {**

**hours = (hours + 1) % 24;**

**} else {**

**minutes = (minutes + 1) % 60;**

**}**

**}**

**// Increment Alarm**

**void incrementAlarm() {**

**if (adjustHours) {**

**alarmHours = (alarmHours + 1) % 24;**

**} else {**

**alarmMinutes = (alarmMinutes + 1) % 60;**

**}**

**}**

**// Display Current Time**

**void displayTime() {**

**lcd.setCursor(0, 0);**

**lcd.print("Time: ");**

**lcd.print(hours < 10 ? "0" : "");**

**lcd.print(hours);**

**lcd.print(":");**

**lcd.print(minutes < 10 ? "0" : "");**

**lcd.print(minutes);**

**lcd.print(":");**

**lcd.print(seconds < 10 ? "0" : "");**

**lcd.print(seconds);**

**lcd.setCursor(0, 1);**

**lcd.print("Alarm: ");**

**lcd.print(alarmOn ? "ON " : "OFF");**

**lcd.print(" ");**

**lcd.print(alarmHours < 10 ? "0" : "");**

**lcd.print(alarmHours);**

**lcd.print(":");**

**lcd.print(alarmMinutes < 10 ? "0" : "");**

**lcd.print(alarmMinutes);**

**}**

**// Display Set Time**

**void displaySetTime() {**

**lcd.setCursor(0, 0);**

**lcd.print("Set Time: ");**

**lcd.setCursor(0, 1);**

**if (adjustHours) {**

**lcd.print("Hours: ");**

**lcd.print(hours < 10 ? "0" : "");**

**lcd.print(hours);**

**} else {**

**lcd.print("Minutes: ");**

**lcd.print(minutes < 10 ? "0" : "");**

**lcd.print(minutes);**

**}**

**}**

**// Display Set Alarm**

**void displaySetAlarm() {**

**lcd.setCursor(0, 0);**

**lcd.print("Set Alarm: ");**

**lcd.setCursor(0, 1);**

**if (adjustHours) {**

**lcd.print("Hours: ");**

**lcd.print(alarmHours < 10 ? "0" : "");**

**lcd.print(alarmHours);**

**} else {**

**lcd.print("Minutes: ");**

**lcd.print(alarmMinutes < 10 ? "0" : "");**

**lcd.print(alarmMinutes);**

**}**

**}**

**// Check Alarm**

**void checkAlarm() {**

**if (hours == alarmHours && minutes == alarmMinutes && seconds == 0) {**

**digitalWrite(buzzerPin, HIGH); // Turn on buzzer**

**delay(1000); // Beep for 1 second**

**digitalWrite(buzzerPin, LOW); // Turn off buzzer**

**}**

**}**