```
`timescale 1ns / 1ps
`include "SoundGen.v"
`include "debouncer.v"
`include "DigitalClock.v"

module AlarmTimer(
    input clk1Hz,
    input alarmEnable,
    input reset,
    input [5:0] alarmSecond,
    input [5:0] alarmMinute,
    input [5:0] clockSecond,
    input [5:0] clockMinute,
    output reg soundEnable
);

reg [5:0] counter;

assign startSound = ((alarmMinute == clockMinute) && (alarmSecond == clockSecond))?
1:0;

always @(posedge clk1Hz, posedge startSound, posedge reset)
begin
    if (counter > 58 || reset)
        begin
            counter = 0;
            soundEnable = 0;
        end
    else if (startSound && !soundEnable && alarmEnable)
        soundEnable = 1;
    else if (soundEnable)
        counter = counter+1;
end

endmodule

module SwitchingTimeDisplayDriver4Digit
(
    input clk400Hz,
    input reset,
    input [5:0] second,
    input [5:0] minute,
    output [1:16] CAN
);
    wire CA,CB,CC,CD,CE,CF,CG,DP;
    wire [7:0] AN;                    // Seven segment display enable

    wire [1:0] select;
    wire [ 1:28] minSec7SegCode;

    counter2bit  c2b(!reset,clk400Hz,1,select);
    dec2to4      dec(select,1,AN[7:4]);
    sixBitBinaryToTwoDisplayDecoder secdec(second, minSec7SegCode[
8:14],minSec7SegCode[ 1: 7]);
```

```
53      sixBitBinaryToTwoDisplayDecoder mindec(minute,
    minSec7SegCode[22:28],minSec7SegCode[15:21]);
54      sevenBits4x1Mux mux(select, minSec7SegCode,{CA,CB,CC,CD,CE,CF,CG});
55      assign AN[3:0] = 4'hF;              // disable upper 4 7-seg display
56      assign DP = 1;                      // disable decimal point
57      assign CAN = {CA,CB,CC,CD,CE,CF,CG,DP,AN[7:0]};
58
59  endmodule
60
61  module CANrot
62  (
63      input [1:16] CAN0,  //{CA:CG,DP,AN}
64      input [1:16] CAN1,
65      input clk400Hz,
66      output [1:16] Q
67  );
68
69  reg [2:0] counter = 0;
70
71  always @(posedge clk400Hz)
72      counter = counter + 1;
73
74  assign Q = (counter[2])? CAN1 : CAN0 ;
75
76  endmodule
77
78  module T_Flip_Flop(T, Clk, Q, QBar);
79      input T;
80      input Clk;
81      output Q;
82      output QBar;
83    reg Q;
84    always@(posedge Clk)
85        begin
86          if (T)
87            Q<=~Q;
88              else
89            Q<=Q;
90        end
91      assign QBar = ~Q;
92  endmodule
93
94
95  module AlarmClock(
96      input BTNC, BTNL, BTNR, BTNU, BTND,
97      input CLK100MHZ,
98      input [15:0] SW,
99      output CA,CB,CC,CD,CE,CF,CG,DP,
100     output [7:0] AN,
101     output AUD_PWM, AUD_SD,
102     output [0:0] LED
103 );
104     assign reset = BTNR;
105     assign load  = BTNL;
106     assign alarmTimeSetBTN = BTND;
```

```
107     assign stop  = BTNC;
108     assign alarmEnableBTN = BTNU;
109 
110     reg [5:0] alarmSecond = 0;
111     reg [5:0] alarmMinute = 0;
112     wire [5:0] clockSecond;
113     wire [5:0] clockMinute;
114     wire [1:16] clockCAN; //{CA:CG,DP,AN}
115     wire [1:16] alarmCAN; //{CA:CG,DP,AN}
116 
117     clockGen     cgn(CLK100MHZ, reset, clk400Hz, clk1Hz);
118     DigitalClock
    dck(CLK100MHZ,load,reset,SW,clockCAN[1],clockCAN[2],clockCAN[3],clockCAN[4],clockCAN[5
    ,clockCAN[6],clockCAN[7],clockCAN[8],clockCAN[9:16],clockSecond[5:0],clockMinute[5:0])
119     SoundGen sgn(CLK100MHZ, reset, soundEnable, AUD_PWM, AUD_SD);
120     AlarmTimer atm(clk1Hz, alarmEnable, |{stop,reset}, alarmSecond[5:0],
    alarmMinute[5:0], clockSecond[5:0], clockMinute[5:0], soundEnable);
121 
122     debouncer db1(CLK100MHZ, alarmEnableBTN, alarmEnableDB);
123     T_Flip_Flop tff(alarmEnableDB, alarmEnableDB, alarmEnable, dummy);
124 
125     always @(posedge CLK100MHZ)
126     begin
127         if (alarmTimeSetBTN)
128         begin
129             alarmMinute = SW[15:8];
130             alarmSecond = SW[7:0];
131         end
132     end
133 
134     assign LED[0] = alarmEnable;
135 
136     SwitchingTimeDisplayDriver4Digit std(clk400Hz, reset, alarmSecond[5:0],
    alarmMinute[5:0], alarmCAN[1:16]);
137     CANrot crot(clockCAN[1:16], alarmCAN[1:16], clk400Hz,
    {CA,CB,CC,CD,CE,CF,CG,DP,AN[7:0]});
138 
139 endmodule
140 
141 
142 
```