1. Timer

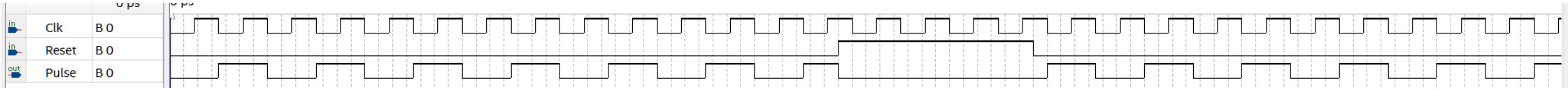
This timer can produce pulses every 2 cycles, and has a high asynchronous reset.

Code:

timer.vhd | RTMD_Controller_test.vhd

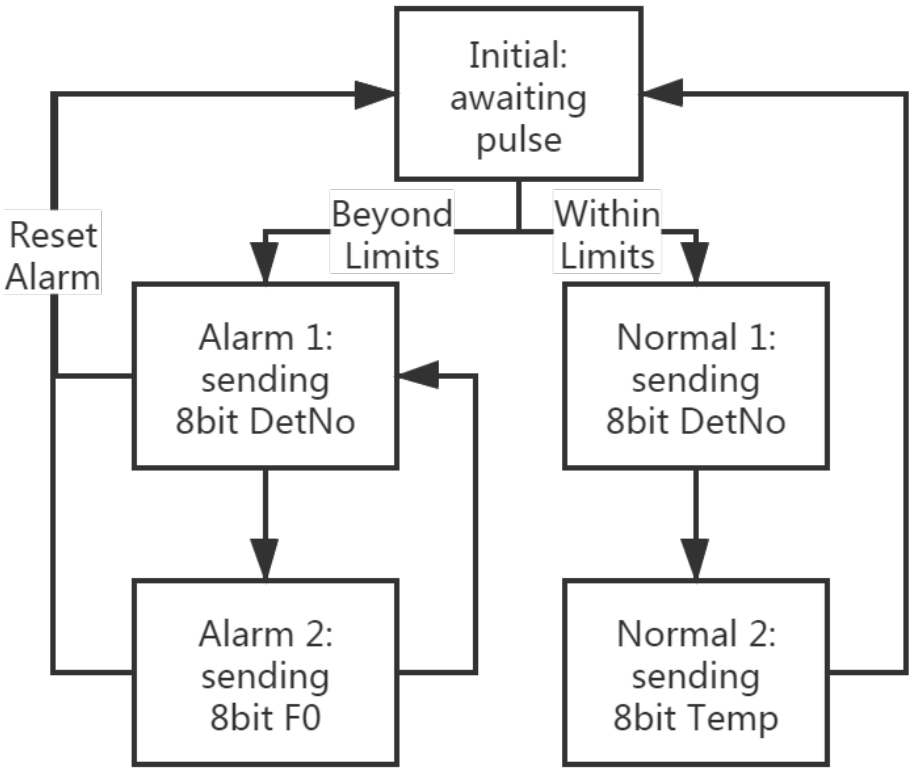
```
1  --student name: ZHIJIE LAN
2  --student number: 201990309
3  --timer module
4
5  library ieee ;
6  use ieee . std_logic_1164 . all ;
7
8  ENTITY timer IS
9     PORT ( Clk , Reset : in std_logic ;
10           Pulse: out std_logic);
11 END timer;
12
13 ARCHITECTURE design_timer OF timer IS
14 signal pul : std_logic := '0';
15 begin
16    process(Clk)
17    begin
18
19       if Reset = '1' then
20          Pul <= '0';
21       elsif (clk'event and clk = '0') then
22          pul <= not pul;
23       end if;
24    end process;
25 pulse <= pul;
26 END design_timer;
27
```

Test:



2. RTMD_Controller

ASM diagram

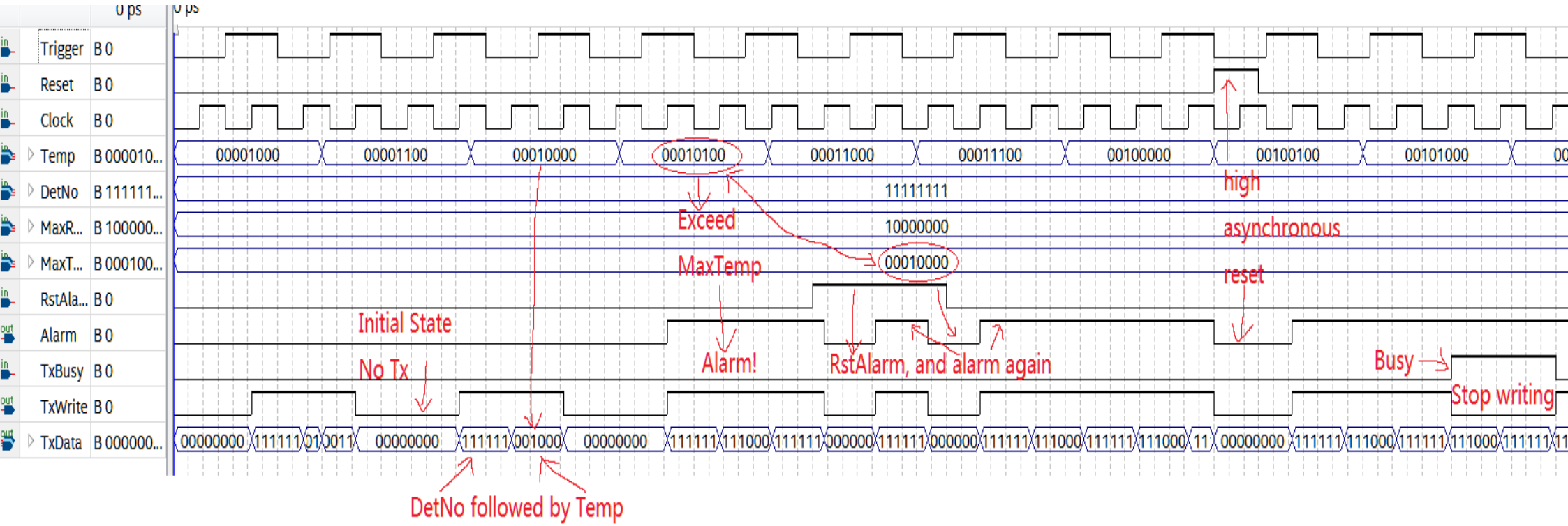


Code:

```
--student name: ZHIJIE LAN
--student number: 201990309
--RTMD_Controller module

library ieee;
use ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY RTMD_Controller IS
   PORT( Trigger, Reset, Clock : IN STD_LOGIC;
         Temp, DetNo, MaxRate, MaxTemp : STD_LOGIC_VECTOR (7 DOWNTO 0);
         RstAlarm, TxBusy : IN STD_LOGIC;
         TxWrite, Alarm : OUT STD_LOGIC;
         TxData : OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
END RTMD_Controller;

ARCHITECTURE design_RTMD OF RTMD_Controller IS

-- Declare temperature rate
shared variable TempRate: integer :=0;
-- Declare a state type
type state_type is (Initial, Normal_1, Normal_2, Alarm_1, Alarm_2);
-- Declare current and next state signals .
signal next_state, current_state: state_type := Initial ;

BEGIN
-- Process to next state
   process ( Clock, Trigger, RstAlarm)
   begin
      if Reset = '1' then -- high asynchronous reset
         Current_state <= Initial;
      elsif (Clock'event and Clock='1') then
            current_state <= next_state ;
      end if ;
   end process ;

```

```
37
38  -- record temp per second (every 32768 cycle) to determine Temp rate
39     process ( Clock)
40     variable cy, currentTemp, lastTemp  : integer := 0;
41     begin
42        if (Clock'event and Clock='1') then
43              cy := cy + 1;
44
45              if (cy = 32768) then -- record temp every 32768 cycle
46
47                    lastTemp := currentTemp;
48                    currentTemp := to_integer(unsigned(Temp));
49                    TempRate := currentTemp - lastTemp;
50                    cy:=0;
51
52              end if;
53        end if;
54     end process ;
55
56  -- determine the next state
57  process ( current_state, Trigger, RstAlarm)
58  begin
59     case current_state is
60        when Initial =>
61           if (Trigger = '1') then
62              if (to_integer(unsigned(Temp)) <= to_integer(unsigned(MaxTemp))
63                 and TempRate <= to_integer(unsigned(MaxRate))) then
64                 -- both temp and rate are within limits
65                 next_state <= Normal_1;
66              else -- beyond limits
67                 next_state <= Alarm_1;
68              end if;
69           else -- if no pulse, stay Initial
70              next_state <= Initial;
71           end if ;
72
73        -- When normal directly go to the next state
74        when Normal_1 =>
75              next_state <= Normal_2;
76        when Normal_2 =>
77              next_state <= Initial;
78
79        -- loop in alarm_1 and alarm_2, until RstAlarm
80        when Alarm_1 =>
81              if (RstAlarm = '1') then
82                 next_state <= Initial ;
83              else
84                 next_state <= Alarm_2;
85              end if;
86        when Alarm_2 =>
87           if (RstAlarm = '1') then
88              next_state <= Initial ;
89           else
90              next_state <= Alarm_1 ;
91           end if ;
92     end case ;
93  end process ;
94
95  -- Conditional assignments for outputs:
96  TxWrite <= '0' when ((TxBusy = '1')
97                    or (Current_state = Initial) ) else '1';
98
99  Alarm <= '1' when (( current_state = Alarm_1 )
100                   or ( current_state = Alarm_2 ) ) else '0';
101
102                   -- In Alarm_1 and Normal_1, send detect No.
103                   -- In Normal_2, send temperature
104 TxData <= DetNo when (( current_state = Alarm_1 )
105                   or ( current_state = Normal_1 ) ) else
106           Temp  when ( current_state = Normal_2 ) else
107           "11110000" when (current_state = Alarm_2) else
108           "00000000";-- In Alarm_2, send F0 which is 11110000
109
110 END design_RTMD ;
```

Test:


0 ps
Trigger B 0
Reset B 0
Clock B 0
Temp B 000010...
DetNo B 111111...
MaxR... B 100000...
MaxT... B 000100...
RstAla... B 0
Alarm B 0
TxBusy B 0
TxWrite B 0
TxData B 000000...
00001000
00001100
00010000
00010100
00011000
00011100
00100000
00100100
00101000
11111111
10000000
00010000
Exceed
MaxTemp
high
asynchronous
reset
Initial State
No Tx
Alarm!
RstAlarm, and alarm again
Busy
Stop writing
DetNo followed by Temp

3. TxSystemRTL

   As the code provided by professor.

4. Complete RTMD system

   Code:

RTMD_system.vhd | RTMD_Controller.vhd | timer

```
--student name: ZHIJIE LAN
--student number: 201990309
--RTMD_Controller module

 library ieee;
 use ieee.std_logic_1164.all;

entity RTMD_system is
   port (CLK, RST, ResetAlarm, V5 : in std_logic;
         Temp, DetNo, MaxRate, MaxTemp : STD_LOGIC_VECTOR (7 DOWNTO 0);
         TxData, Alarm: out std_logic);
end RTMD_system;


ARCHITECTURE design_system OF RTMD_system is

component timer is -- timer module
   PORT ( Clk , Reset : in std_logic ;
          Pulse: out std_logic);
end component ;

component RTMD_Controller is -- controller module
   PORT( Trigger, Reset, Clock : IN STD_LOGIC;
         Temp, DetNo, MaxRate, MaxTemp : STD_LOGIC_VECTOR (7 DOWNTO 0);
         RstAlarm, TxBusy : IN STD_LOGIC;
         TxWrite, Alarm : OUT STD_LOGIC;
         TxData : OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
end component ;

component TxSystemRTL is -- TxSystemRTL module
   PORT( clock, reset,odd_noteven, write_l : IN STD_LOGIC;
         Datain : IN std_logic_vector (7 downto 0);
         busy, TxData : OUT STD_LOGIC
         );
end component ;

-- internal signal, prodeced by each module
signal Pulse_Sig, Busy_sig, Write_sig : std_logic;
signal Data_sig : std_logic_vector(7 downto 0);

begin

--
T0: timer port map (CLK, RST, Pulse_sig);

C0: RTMD_Controller port map (Pulse_sig, RST, CLK,
                              Temp, DetNo, MaxRate, MaxTemp,
                              ResetAlarm, Busy_sig,
                              Write_sig, Alarm,
                              Data_sig);

R0: TxSystemRTL port map (CLK, RST, V5, Write_sig,
                          Data_sig, Busy_sig, TxData);

end design_system;
```