```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/02/2020 09:13:50 PM
// Design Name:
// Module Name: state_machine
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module state_machine(

    );
endmodule
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 02/10/2020 02:16:43 AM
// Design Name:
// Module Name: StateMachine
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
```

```
module state_machine(
    input start, clk, crash, twoSecs,
    output started, blinkCar, resetScore, starting, resetTimer, loadRoad,
countScore, blinkScore, resetLED
    );

    wire SPAWN, STARTING, STARTED, STOPPED;
    wire Next_SPAWN, Next_STARTING, Next_STARTED, Next_STOPPED;
    wire [6:0]PS;
    wire [6:0]NS;

    assign SPAWN = PS[0];
    assign STARTING = PS[1];
    assign STARTED = PS[2];
    assign STOPPED = PS[3];

    assign NS[0] = Next_SPAWN;
    assign NS[1] = Next_STARTING;
    assign NS[2] = Next_STARTED;
    assign NS[3] = Next_STOPPED;

     assign Next_SPAWN = SPAWN & ~start;
     assign Next_STARTING = (((SPAWN | STOPPED) & start) | (STARTING & ~twoSecs));
     assign Next_STARTED = (STARTING & twoSecs) | (STARTED & ~crash);
     assign Next_STOPPED = (STARTED & crash) | (STOPPED & ~start);

     assign blinkCar = STOPPED | STARTING;
     assign blinkScore = STOPPED;
     assign loadRoad = (SPAWN & ~start) | (STOPPED & start);
     assign resetTimer = ((SPAWN|STOPPED) & start) | (STARTING & twoSecs);
     assign started = STARTED;
     assign starting = STARTING;
     assign resetScore = SPAWN | STARTING;
     assign countScore = STARTED;
     assign resetLED = STARTING & twoSecs;

//    assign Next_WAITING = ( (WAITING | LtR | LtR_R | RtL | RtL_L)  & (L & R) );
//    assign Next_LtR = ( (LtR | WAITING | LtR_C) & (~L & R) );
//    assign Next_LtR_C = ( (LtR_C | LtR | LtR_R) & (~L & ~R) );
//    assign Next_LtR_R = ( (LtR_R | LtR_C) & (L & ~R));
//    assign Next_RtL = ( (RtL | WAITING | RtL_C) & (L & ~R) );
//    assign Next_RtL_C = ( (RtL_C | RtL | RtL_L) & (~L & ~R) );
//    assign Next_RtL_L = ( (RtL_L | RtL_C) & (~L & R));
```

```
//      assign LeftFromRight = (LtR_R & L & R);
//      assign LeftFromLeft = (RtL_L & L & R);
//      assign DisplayTimer = ~WAITING;
//      assign ResetTimer = WAITING;



    FDRE #(.INIT(1'b1)) Q0_FF (.C(clk),.CE(1'b1),.D(NS[0]),.Q(PS[0]));
    FDRE #(.INIT(1'b0)) Q6_FF[3:1]
(.C({3{clk}}),.CE({3{1'b1}}),.D(NS[3:1]),.Q(PS[3:1]));
endmodule
```