□ Mini Project Report

Title: Traffic Light Controller using Verilog HDL on DE10-Standard FPGA

Course: Digital System Design (DSD)

Tool Used: Intel Quartus Prime

Hardware: DE10-Standard (Cyclone V 5CSXFC6D6F31C6N)

Language: Verilog HDL

Name: Zaahir Ali Shaik

Reg no.24BEC1361

◈ 1. Objective

To design and implement a **Traffic Light Controller** using **Verilog HDL** and demonstrate its working on the **DE10-Standard FPGA** development board.
The system controls three LEDs representing **Red**, **Yellow**, and **Green** signals with realistic time delays between each state.

◈ 2. Components and Tools

Component	Description
DE10-Standard Board	Intel Cyclone V SoC FPGA
Intel Quartus Prime	HDL design & synthesis tool
USB-Blaster	Programming interface
Verilog HDL	Hardware Description Language used
Push Button KEY0	Reset input
LEDR[0 – 2]	Red LEDs used for traffic lights

---

3. Theory

The **Traffic Light Controller** is a **Finite State Machine (FSM)**-based sequential circuit. It cycles through **Red → Green → Yellow → Red**, controlling the LEDs for specific time intervals.

Each state is maintained by counting the FPGA's 50 MHz clock pulses until the desired time elapses.
The design demonstrates:

- **Sequential Logic Design**
- **Timing Control using Counters**
- **FSM Implementation in Verilog**

---

4. Block Diagram

```
   +-------------------------------------+
   |          Traffic Light FSM          |
   |                                     |
CLK |--> [Counter / Timing Logic] --> State Transition
RST |--> [Reset Control]                  |
   |                                     |
   +-------------------> LEDR0-2 Outputs |
```

---

5. Updated Verilog Code (with slower timing)

```
module traffic_light_controller (
    input  wire CLOCK_50,      // 50 MHz clock from DE10-Standard
    input  wire [3:0] KEY,     // push buttons
    output reg  [9:0] LEDR     // red LEDs on board
);

    // Active-low reset from KEY0
    wire reset = ~KEY[0];

    // State encoding
    localparam RED    = 2'b00;
    localparam YELLOW = 2'b01;
    localparam GREEN  = 2'b10;

    reg [1:0] state;
    reg [27:0] counter; // 28 bits to handle large delays

    // Realistic delay times (~3 sec for RED/GREEN, 1.5 sec for YELLOW)
    localparam RED_TIME    = 28'd150000000;
    localparam YELLOW_TIME = 28'd75000000;
    localparam GREEN_TIME  = 28'd150000000;

    // FSM: state transition + timing
    always @(posedge CLOCK_50 or posedge reset) begin
        if (reset) begin
            state   <= RED;
            counter <= RED_TIME;
```

```
        end else begin
            if (counter == 0) begin
                case (state)
                    RED:    begin state <= GREEN;  counter <= GREEN_TIME;
end
                    GREEN:  begin state <= YELLOW; counter <= YELLOW_TIME;
end
                    YELLOW: begin state <= RED;    counter <= RED_TIME;
end
                    default: begin state <= RED;   counter <= RED_TIME;
end
                endcase
            end else begin
                counter <= counter - 1;
            end
        end
    end

    // Output logic
    always @(*) begin
        LEDR = 10'b0;  // all LEDs off by default
        case (state)
            RED:    LEDR[2] = 1'b1;  // top LED (stop)
            YELLOW: LEDR[1] = 1'b1;  // middle LED (wait)
            GREEN:  LEDR[0] = 1'b1;  // bottom LED (go)
        endcase
    end

endmodule
```

---

6. Pin Assignments

Signal	FPGA Pin	Description
CLOCK_50	PIN_AF14	50 MHz Clock
KEY[0]	PIN_AJ4	Reset Button (KEY0, active-low)
LEDR[0]	PIN_AA24	Green LED (logical)
LEDR[1]	PIN_AB23	Yellow LED (logical)
LEDR[2]	PIN_AC23	Red LED (logical)

I/O Standard: 3.3-V LVTTL

---

7. Working Principle

1. When power is applied and reset (KEY0) is released, the FSM starts in the **RED** state.

2. After ~3 seconds, it transitions to **GREEN**, keeping the signal ON for 3 seconds.
3. Next, it switches to **YELLOW** for ~1.5 seconds.
4. The cycle repeats indefinitely.
5. Pressing **KEY0** resets the sequence back to RED.

🔷 8. Output Observation

State	LEDR2	LEDR1	LEDR0	Meaning
RED	ON	OFF	OFF	Stop
GREEN	OFF	OFF	ON	Go
YELLOW	OFF	ON	OFF	Wait

🔷 9. Result

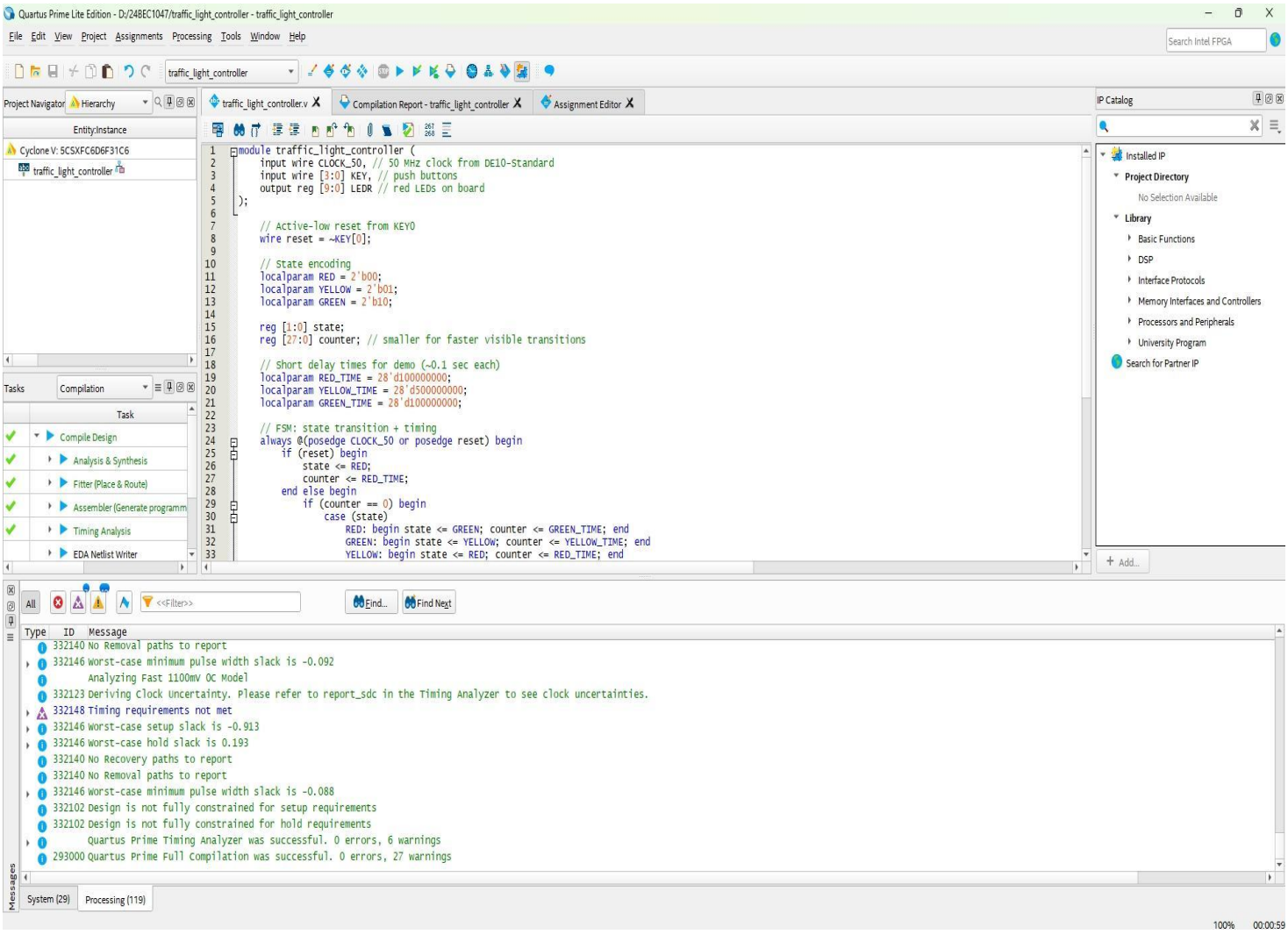
✅ The **Traffic Light Controller** was successfully implemented on the **DE10-Standard FPGA** board.
✅ The LEDs transitioned at realistic time intervals, simulating an actual traffic signal.
✅ Reset functionality worked correctly, returning the system to the RED state.

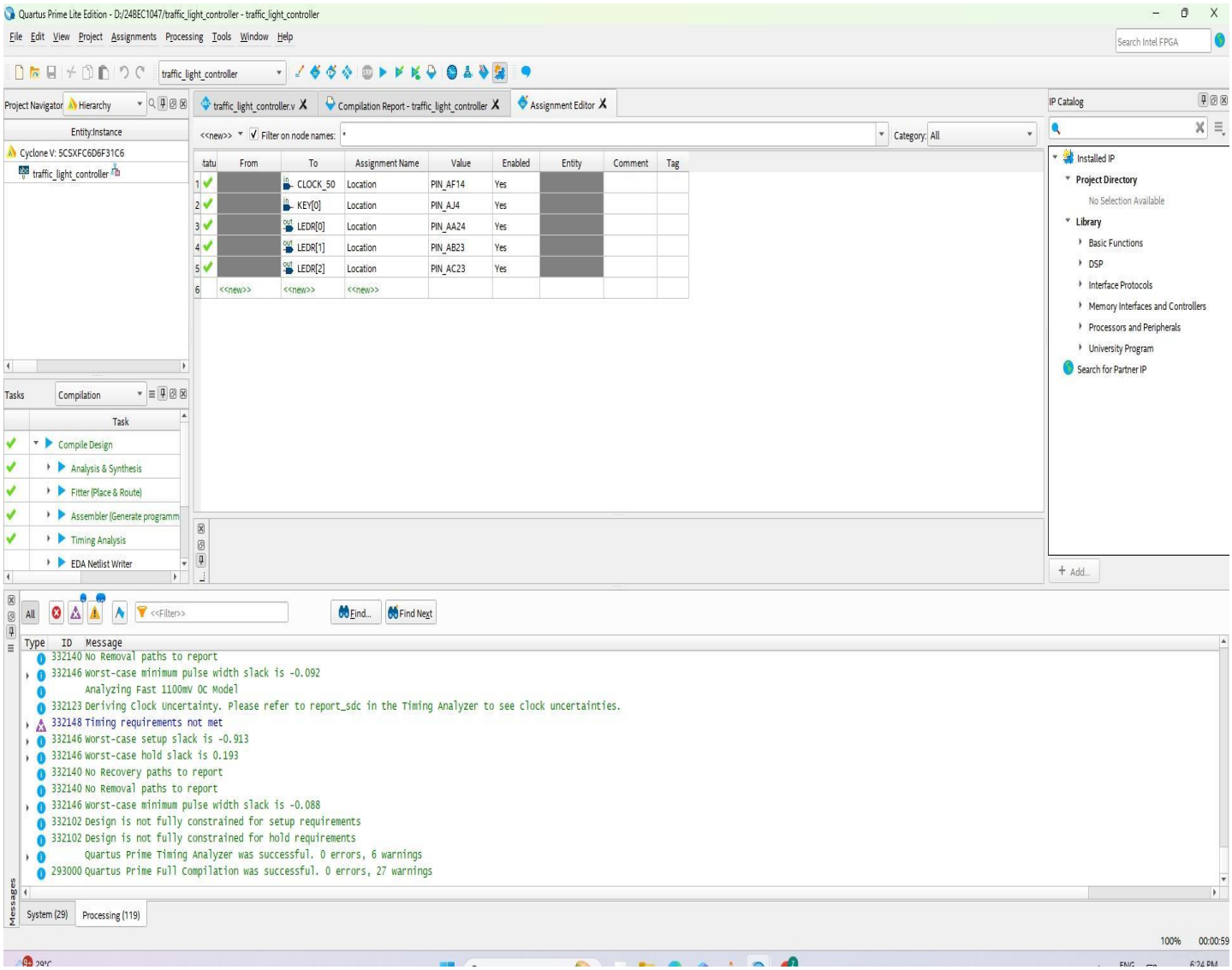
🔷 10. Applications

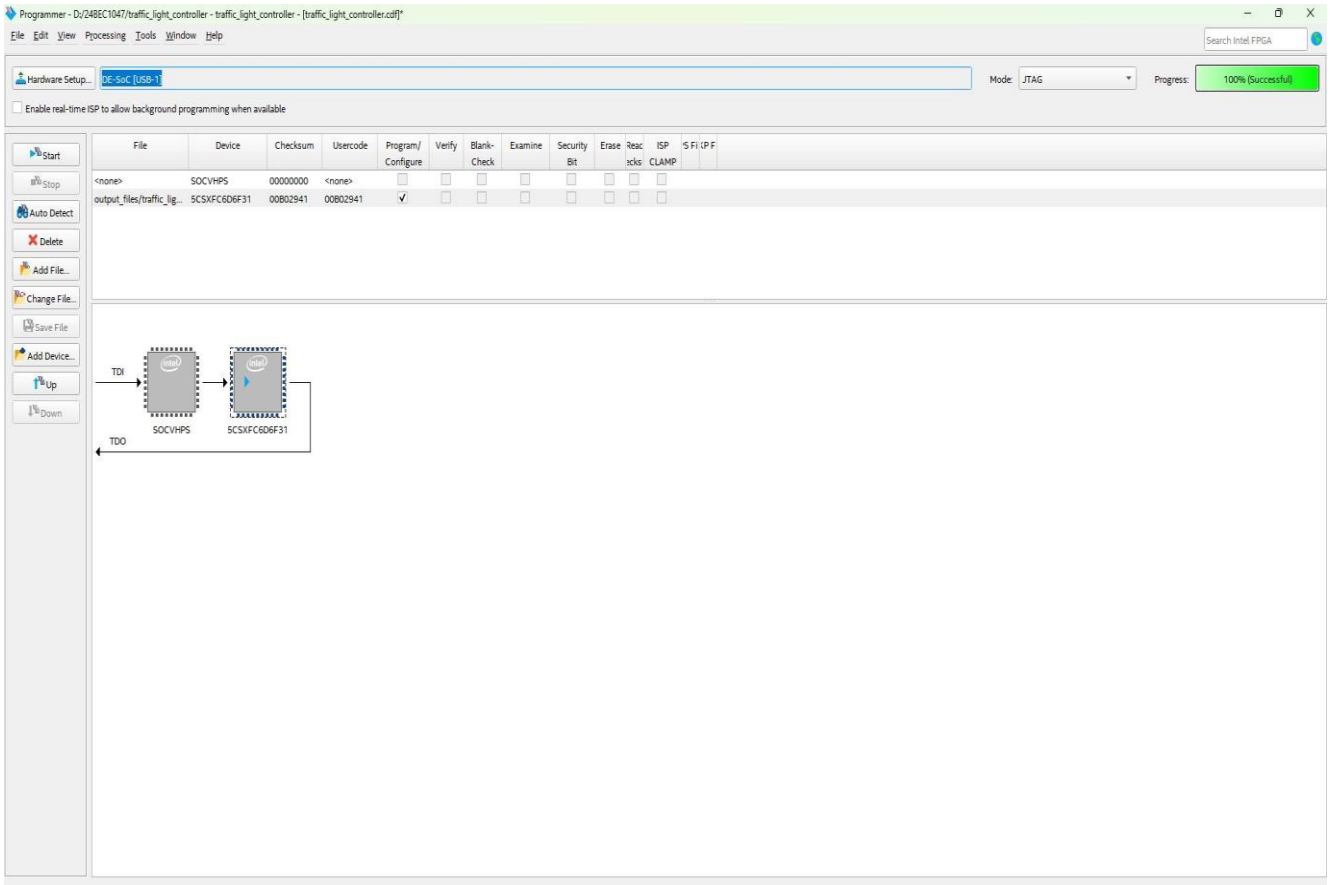
- Traffic-signal simulation and timing analysis
- Basic FSM design training
- Sequential logic demonstration in FPGA lab
- Foundation for more complex embedded control systems

🔷 11. Conclusion

This project demonstrates a real-time **Finite State Machine** using **Verilog HDL** on an FPGA.
The system effectively models a real traffic light, showing how timing counters and state transitions can control hardware outputs.
It reinforces key DSD concepts such as **synchronous design**, **clock division**, and **state-based logic**.


Quartus Prime Lite Edition - D:/24BEC1047/traffic_light_controller - traffic_light_controller
File Edit View Project Assignments Processing Tools Window Help
Search Intel FPGA
traffic_light_controller
Project Navigator Hierarchy
Entity:Instance
Cyclone V: 5CSXFC6D6F31C6
traffic_light_controller
Tasks Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programm
Timing Analysis
EDA Netlist Writer
traffic_light_controller.v
Compilation Report - traffic_light_controller
Assignment Editor
1 module traffic_light_controller (
2 input wire CLOCK_50, // 50 MHz clock from DE10-Standard
3 input wire [3:0] KEY, // push buttons
4 output reg [9:0] LEDR // red LEDs on board
5);
6
7 // Active-low reset from KEY0
8 wire reset = ~KEY[0];
9
10 // State encoding
11 localparam RED = 2'b00;
12 localparam YELLOW = 2'b01;
13 localparam GREEN = 2'b10;
14
15 reg [1:0] state;
16 reg [27:0] counter; // smaller for faster visible transitions
17
18 // Short delay times for demo (~0.1 sec each)
19 localparam RED_TIME = 28'd100000000;
20 localparam YELLOW_TIME = 28'd500000000;
21 localparam GREEN_TIME = 28'd100000000;
22
23 // FSM: state transition + timing
24 always @(posedge CLOCK_50 or posedge reset) begin
25 if (reset) begin
26 state <= RED;
27 counter <= RED_TIME;
28 end else begin
29 if (counter == 0) begin
30 case (state)
31 RED: begin state <= GREEN; counter <= GREEN_TIME; end
32 GREEN: begin state <= YELLOW; counter <= YELLOW_TIME; end
33 YELLOW: begin state <= RED; counter <= RED_TIME; end
IP Catalog
Installed IP
Project Directory
No Selection Available
Library
Basic Functions
DSP
Interface Protocols
Memory Interfaces and Controllers
Processors and Peripherals
University Program
Search for Partner IP
Add...
All
<<Filter>>
Find...
Find Next
Type ID Message
332140 No Removal paths to report
332146 Worst-case minimum pulse width slack is -0.092
Analyzing Fast 1100mV 0C Model
332123 Deriving Clock Uncertainty. Please refer to report_sdc in the Timing Analyzer to see clock uncertainties.
332148 Timing requirements not met
332146 Worst-case setup slack is -0.913
332146 Worst-case hold slack is 0.193
332140 No Recovery paths to report
332140 No Removal paths to report
332146 Worst-case minimum pulse width slack is -0.088
332102 Design is not fully constrained for setup requirements
332102 Design is not fully constrained for hold requirements
Quartus Prime Timing Analyzer was successful. 0 errors, 6 warnings
293000 Quartus Prime Full Compilation was successful. 0 errors, 27 warnings
Messages
System (29)
Processing (119)
100%
00:00:59


Quartus Prime Lite Edition - D:/24BEC1047/traffic_light_controller - traffic_light_controller
File Edit View Project Assignments Processing Tools Window Help
Search Intel FPGA
traffic_light_controller
Project Navigator
Hierarchy
Entity:Instance
Cyclone V: 5CSXFC6D6F31C6
traffic_light_controller
traffic_light_controller.v
Compilation Report - traffic_light_controller
Assignment Editor
<<new>>
Filter on node names:
Category: All
tatu From To Assignment Name Value Enabled Entity Comment Tag
1 CLOCK_50 Location PIN_AF14 Yes
2 KEY[0] Location PIN_AJ4 Yes
3 LEDR[0] Location PIN_AA24 Yes
4 LEDR[1] Location PIN_AB23 Yes
5 LEDR[2] Location PIN_AC23 Yes
6 <<new>> <<new>> <<new>>
IP Catalog
Installed IP
Project Directory
No Selection Available
Library
Basic Functions
DSP
Interface Protocols
Memory Interfaces and Controllers
Processors and Peripherals
University Program
Search for Partner IP
Add...
Tasks
Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programm
Timing Analysis
EDA Netlist Writer
All
<<Filter>>
Find...
Find Next
Type ID Message
332140 No Removal paths to report
332146 Worst-case minimum pulse width slack is -0.092
Analyzing Fast 1100mv 0C Model
332123 Deriving Clock Uncertainty. Please refer to report_sdc in the Timing Analyzer to see clock uncertainties.
332148 Timing requirements not met
332146 Worst-case setup slack is -0.913
332146 Worst-case hold slack is 0.193
332140 No Recovery paths to report
332140 No Removal paths to report
332146 Worst-case minimum pulse width slack is -0.088
332102 Design is not fully constrained for setup requirements
332102 Design is not fully constrained for hold requirements
Quartus Prime Timing Analyzer was successful. 0 errors, 6 warnings
293000 Quartus Prime Full Compilation was successful. 0 errors, 27 warnings
Messages
System (29)
Processing (119)
100%
00:00:59


Programmer - D:/24BEC1047/traffic_light_controller - traffic_light_controller - [traffic_light_controller.cdf]*
File Edit View Processing Tools Window Help
Search Intel FPGA
Hardware Setup...
DE-SoC [USB-1]
Mode: JTAG
Progress:
100% (Successful)
Enable real-time ISP to allow background programming when available
Start
Stop
Auto Detect
Delete
Add File...
Change File...
Save File
Add Device...
Up
Down
File
Device
Checksum
Usercode
Program/ Configure
Verify
Blank-Check
Examine
Security Bit
Erase
ISP CLAMP
<none>
SOCVHPS
00000000
<none>
output_files/traffic_lig...
5CSXFC6D6F31
00B02941
00B02941
TDI
SOCVHPS
5CSXFC6D6F31
TDO