CS207 Project Report

A Real Car

Student Name and Student ID

冯泽欣 12110104

林俊杰 12111511

丁健乐 12111517

1 Development Schedule

1.1 Selection of Project-------A Real Car Project

1.2 Contributions

Member	Contributions	Ratio
冯泽欣	1. Design and analysis of the project 2. Code implementation of most module 3. Code implementation of automatic driving module 4. Debug and on board testing 5. Final video recording	1/3
林俊杰	1. Design and analysis of the project 2. Drawing of state diagram 3. Design of VGA module and implementation of code 4. Report writing 5. Final video recording	1/3
丁健乐	1. Design and analysis of the project 2. Design and analysis of manual driving and automatic driving module 3. Report writing 4. Final video recording	1/3

1.3 Weekly Schedule

Time	Work
Start-12.16	Design and analysis of the project
12.16-12.31	Code implementation of the project
12.31-1.8	Report writing and final video recording

2 User Documentation

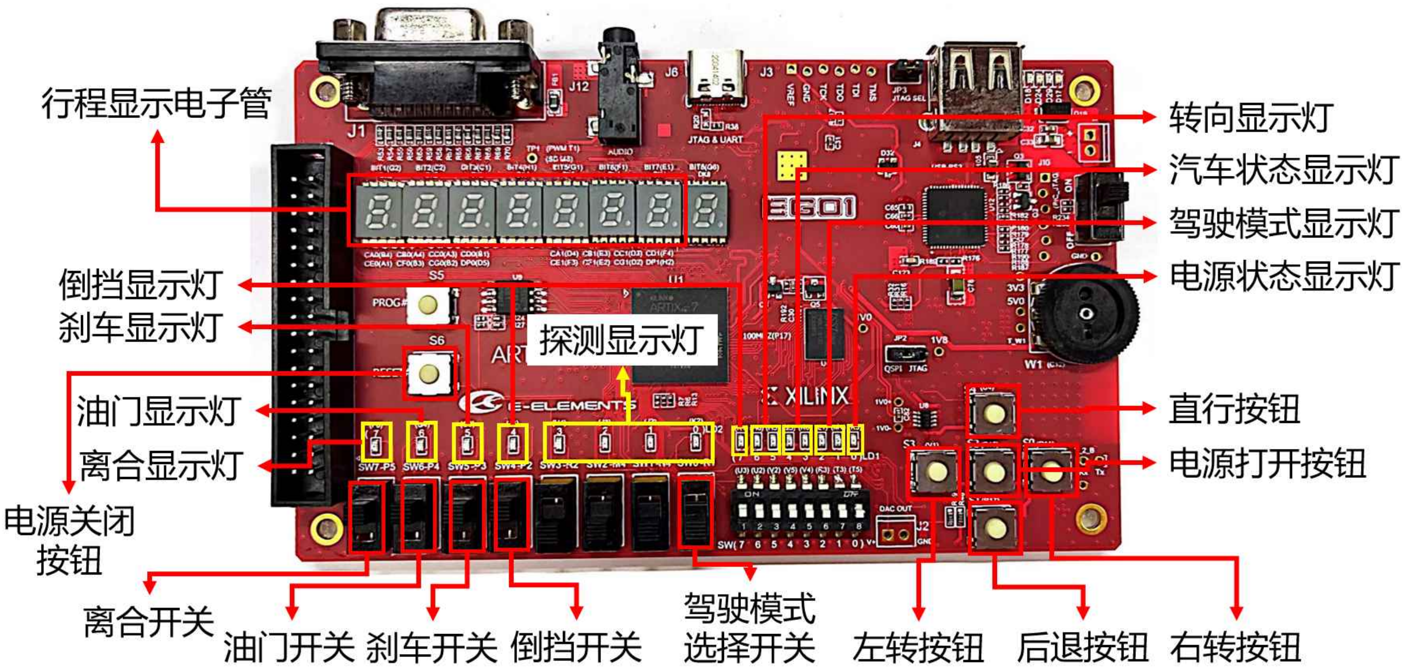


Figure 1 Design of EGO1 development board

2.1 Brief Introduction

This is a car containing 3 functions, which are manual driving, semi-auto driving and auto-driving.

You can press buttons and switches to drive or switch driving modes. Also, the corresponding lights will go on and off with the running state of the car.

2.2 Buttons and Switches

In this system, there are six buttons and four switches in total to be used.

- S0: Turn right button, press and hold on to let the car turn right
- S1: Move backward button(only used in semi-auto mode), press to let the car go backward

- S2: Power-on button, press and hold on for one second to enable the car's engine
- S3: Turn left button, press and hold on to let the car turn left
- S4: Move forward button(only used in semi-auto mode), press to let the car go forward
- S6: Power-off button, press to disable car's engine
- SW0: Mode-changing switch, turn and off once to change mode(manual, semi-auto and auto)
- SW4: Reverse gear switch, when it is not switched, go ahead, else, go backward
- SW5: Break switch, when it is switched, receives break signal
- SW6: Throttle switch, when it is switched, receives throttle signal
- SW7: Clutch switch, when it is switched, receives clutch signal

2.3 Lights

In this system, there are seven kinds of lights that represent different meanings.

- DN0-K1~DN0-K4, DN1-K1~DN1-K4: Travel display tube，which represents the distance that the car has travelled.
- D0: Clutch display light, which displays whether the clutch switch is on.
- D1: Throttle display light, which displays whether the throttle switch is on.
- D2: Break display light, which displays whether the break switch is on.
- D3&D8: Reverse gear display light, which displays whether the car runs forward or backward. (Because the connection of D3 lights sometimes fails, so we bind D8 extra)
- D4&D5&D6&D7: Detection display lamp. If there's a wall in front of the car, the D4 will be on. If there's a wall behind the car, the D5 will be on. If there's a wall on the car's left, the D6 will be on. If there's a wall on the car's right, the D7 will be on.
- D9&D10: Turning display light, which displays whether the car is turning, D9 on for turning left, D10 on for turning right, both on or both off for no turning.
- D11&D12: Car mode display light, which displays the driving mode of the car, D11 on and D12 off for manual driving, D11 off and D12 on for semi-auto driving, D11 on and D12 on for auto driving
- D13&D14: Car driving state display light, which displays the driving state of the car: In manual driving, D13 on and D14 off for non-starting state, D13 off and D14 on for starting state, D13 on and D14 on for moving state; In semi-auto driving, D13 off and D14 off for wait-for-command state, D13 on and D14 on for Moving state; In auto driving, since the car will find the way out itself, D13&D14 will always be off.
- D15:Power display light, which displays whether the car power is on.

2.4 Working Process

In this system, we first press and hold on the power-on button(S2) for one second to enable the car's engine.

2.4.1 Manual Driving

Once the engine is enabled, the D9, D10, D11, D13, D15 lights will be on, which means the power is on. The default driving mode is manual driving, default driving state is non-starting state. And at any time of manual driving, you can turn on and off the mode-changing switch(SW0) once to change the manual driving mode to semi-auto driving mode.

- **Non-starting state**

The default driving state is non-starting state.

In this state, you can only turn on the clutch switch(SW7) and throttle switch(SW6) one by one or at the same time to change driving state to starting state.

Here, you should pay attention that if you turn on the throttle switch(SW6) only, the car will return power off state, and all the lights will be off.

- **Starting state**

If you have already changed into starting state, the D11, D14, D15 lights will be on.

In this state, if you turn on the clutch switch(SW7), you can turn on and off the reverse gear switch(SW4) to change the moving direction of the car. If you turn on the switch(SW4), the reverse gear display light(D3&D8) will be on, and the car's moving direction is backward and if you turn off the switch(SW4), the reverse gear display light(D3&D8) will be off, and the car's moving direction is forward.

If you only turn the throttle switch(SW6) on, and turn the clutch switch(SW7) off, the car will change into the moving state and move forward/backward depending on the moving direction(D3&D8).

You can turn the turning left button(S0) and the turning right button(S3) at any time to let the car turn left or turn right, and if the car is turning left, the D9 light will be on, and if the cat is turning right, the D10 light will be on.

The brake switch(SW5) has higher priority than the throttle switch(SW6). If you turn on the brake switch(SW5), the driving state will change to non-starting state immediately.

- **Moving state**

If you have already changed into moving state, the D11, D13, D14, D15 lights will be on.

In this state, the car is moving according to the reverse gear display light(D3&D8). And the throttle switch(SW6) will always be turned on.

If you turn off the throttle switch(SW6) or turn on the clutch switch(SW7), the car will turn into the non-starting state.

If you turn on the brake switch(SW4), the car will turn into the non-starting state.

You can turn the turning left button(S0) and the turning right button(S3) at any time to let the car turn left or turn right, and if the car is turning left, the D9 light will be on, and if the cat is turning right, the D10 light will be on.

Here you should pay attention if you turn on the reverse gear switch(SW4) directly without turning on the clutch switch(SW7), the car will return power off state, and all the lights will be off.

If the car knocks on the wall while driving, it will go back to the starting point immediately.

2.4.2 Semi-auto Driving

If you get into semi-auto driving mode, the D12, D13, D14, D15 lights will be on, and the default state is forward state. And at any time of semi-auto driving, you can turn on and off the mode-changing button(SW0) once to change the semi-auto driving mode into auto driving mode.

- **Forward state**

In this state, the car will keep moving forward. The D12, D13, D14, D15 lights will be on. Once the car meets an intersection, the car will stop and turn the state into wait-for-command state. If the car faces a dead end, the car will turn for 180° and move forward until it meets a cross.

- **Wait-for-command state**

If the car is in wait-for-command state, the D12, D15 lights will be on. The car will stop at the intersection.

In this state, if you press the turn right button(S0), the car will turn into the turn right state.

In this state, if you press the turn left button(S3), the car will turn into the turn left state.

In this state, if you press the move forward button(S4), the car will turn into the forward state.

In this state, if you press the move backward button(S1), the car will turn into the forward state.

- **Turn right state**

In this state, the turning right display light(D10) will be on and the car will turn right and once the car has turned 90°, the car will stop turning and change into the forward state.

- **Turn left state**

In this state, the turning left display light(D9) will be on and the car will turn left and once the car has turned 90°, the car will stop turning and change into the forward state.

2.4.3 Auto Driving

And at any time of auto driving, you can turn on and off the mode-changing button(SW0) once to change the auto driving mode into manual driving mode.

In this mode, the car will drive on its own and find the exit on its own!

3 Design of the System

3.1 State Diagram

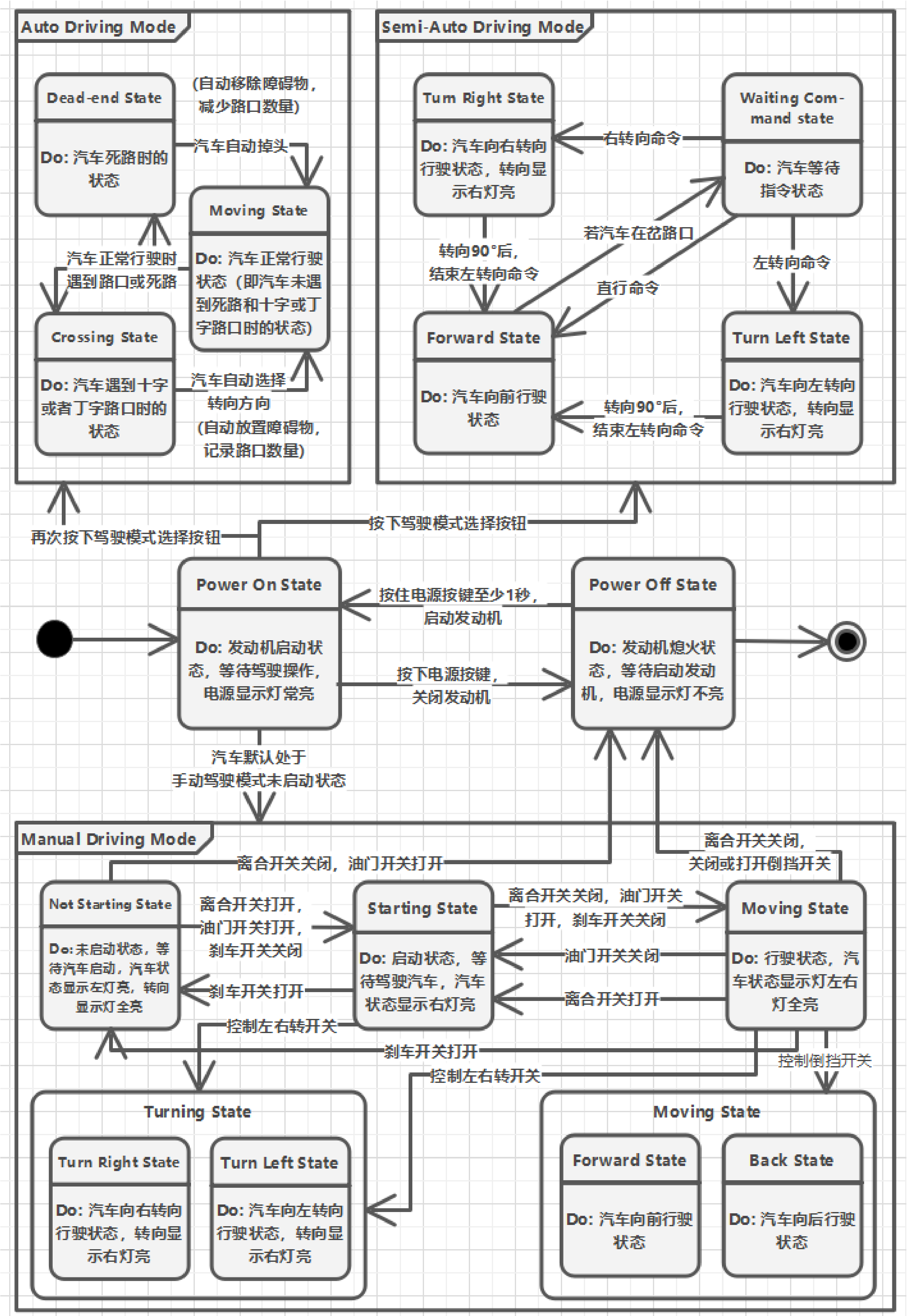

Auto Driving Mode
Dead-end State
Do: 汽车死路时的状态
(自动移除障碍物，减少路口数量)
汽车自动掉头
Moving State
Do: 汽车正常行驶状态（即汽车未遇到死路和十字或丁字路口时的状态）
汽车正常行驶时遇到路口或死路
Crossing State
Do: 汽车遇到十字或者丁字路口时的状态
汽车自动选择转向方向
(自动放置障碍物，记录路口数量)
Semi-Auto Driving Mode
Turn Right State
Do: 汽车向右转向行驶状态，转向显示右灯亮
右转向命令
Waiting Com-mand state
Do: 汽车等待指令状态
若汽车在岔路口
直行命令
转向90°后，结束左转向命令
左转向命令
Forward State
Do: 汽车向前行驶状态
Turn Left State
Do: 汽车向左转向行驶状态，转向显示右灯亮
转向90°后，结束左转向命令
再次按下驾驶模式选择按钮
按下驾驶模式选择按钮
Power On State
Do: 发动机启动状态，等待驾驶操作，电源显示灯常亮
按住电源按键至少1秒，启动发动机
按下电源按键，关闭发动机
Power Off State
Do: 发动机熄火状态，等待启动发动机，电源显示灯不亮
汽车默认处于手动驾驶模式未启动状态
Manual Driving Mode
离合开关关闭，油门开关打开
离合开关关闭，关闭或打开倒挡开关
Not Starting State
Do:未启动状态，等待汽车启动，汽车状态显示左灯亮，转向显示灯全亮
离合开关打开，油门开关打开，刹车开关关闭
Starting State
Do: 启动状态，等待驾驶汽车，汽车状态显示右灯亮
离合开关关闭，油门开关打开，刹车开关关闭
Moving State
Do: 行驶状态，汽车状态显示灯左右灯全亮
油门开关关闭
刹车开关打开
离合开关打开
控制左右转开关
刹车开关打开
控制倒挡开关
控制左右转开关
Turning State
Turn Right State
Do: 汽车向右转向行驶状态，转向显示右灯亮
Turn Left State
Do: 汽车向左转向行驶状态，转向显示右灯亮
Moving State
Forward State
Do: 汽车向前行驶状态
Back State
Do: 汽车向后行驶状态

Figure 2 State diagram

3.2.1 Description

Our state diagram is drawn using the application "亿图图示". The content of the state diagram introduces the working mechanism of the system module and the introduction of the car state.

3.2 Architecture Diagram of the Circuit

3.2.1 Architecture Diagram

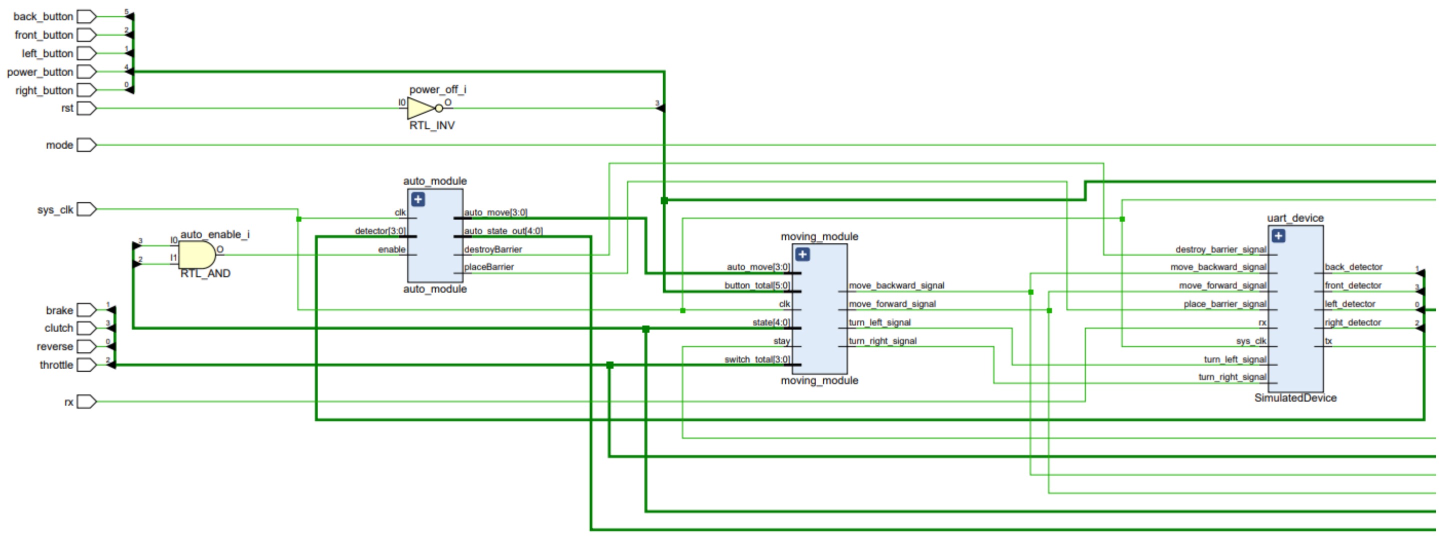


Figure 3 The left part of the architecture diagram

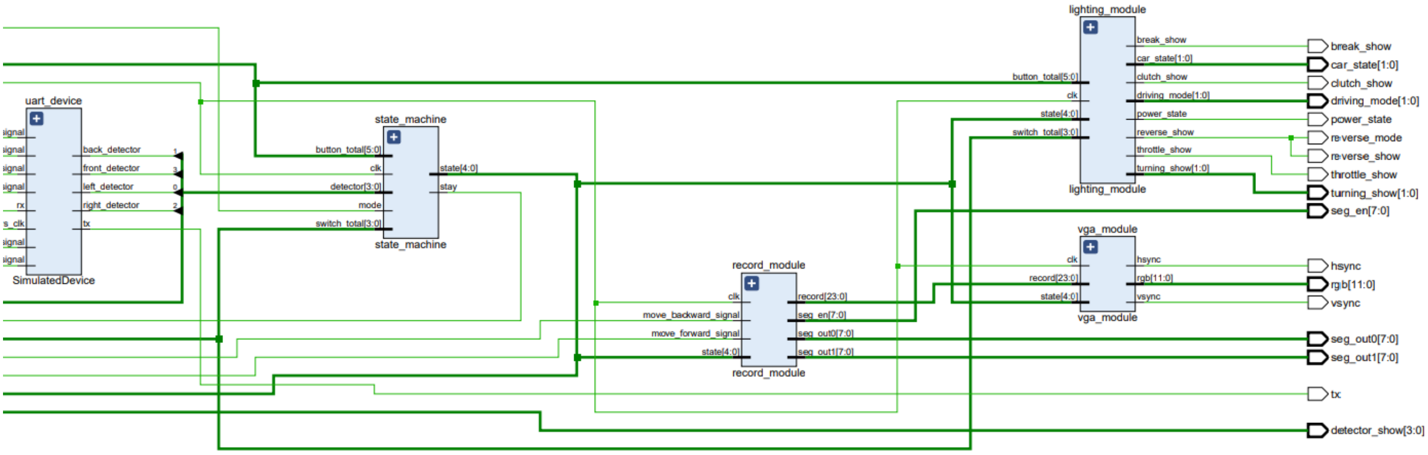


Figure 4 The right part of the architecture diagram

Declare： Since the size of the architecture diagram is too large, we divided it into two parts in order to show it clearly. Figure 3 is the left part of the architecture diagram, and Figure 4 is the right part of the architecture diagram. Many of the architecture diagrams reported below use the same method in order to make their presentation clear. If teacher you can't see clearly, please enlarge it. We're really sorry.

3.2.2 Description of the relationship between the top-level module and each submodule and each sub-module

In our design, the top module ***car*** uses 7 submodules: ***state_machine***, ***auto_module, moving_module, lighting_module, record_module, vga_module*** and ***SimulatedDevice***, where ***state_machine*** realizes the function of switching the state of the car, ***auto_module*** realizes the function of controlling the automatic driving of the car, ***moving_module*** realizes the function of controlling the car driving movement, ***lighting_module*** realizes the function of controlling the display of lights, ***record_module*** realizes the function of recording and displaying the mileage of the car, ***vga_module*** and realizes the function of controlling the display of VGA content.

- ***state_machine***

The module ***state_machine*** sues the clk of top module, a mode signal and switch_total, button_total, detector signals. Its output state is sent to modules ***moving_module***, ***lighting_module***, ***record_module*** and ***vga_module***, cool is sent to module ***moving_module***.

- ***auto_module***

The module ***auto_module*** sues the clk of top module, an auto_enable signal and detector signals. Its outputs placeBarrier, destroyBarrier and auto_state_out are sent to the top module and auto_move is sent to module ***moving_module***.

- ***moving_module***

The module ***moving_module*** sues the clk, switch_total and button_total signals of top module, state and cool signals of moule ***state_machine***, and auto_move signal of module ***auto_module***. Its outputs move_forward_signal, move_backward_signal, turn_left_signal and turn_right_signal are sent to the top module.

- ***lighting_module***

The module ***lighting_module*** sues the clk of top module and state, switch_total, button_total signals. Its outputs power_state, driving_mode, car_state, clutch_show, throttle_show, break_show, reverse_show, turning_show are sent to the top module.

- ***record_module***

The module ***record_module*** sues the clk of top module, state signal of module ***state_machine*** and move_forward_signal, move_backward_signal signals. Its output record is sent to module ***vga_module***, and outputs seg_en, seg_out0 and seg_out1 are sent to the top module.

- ***vga_module***

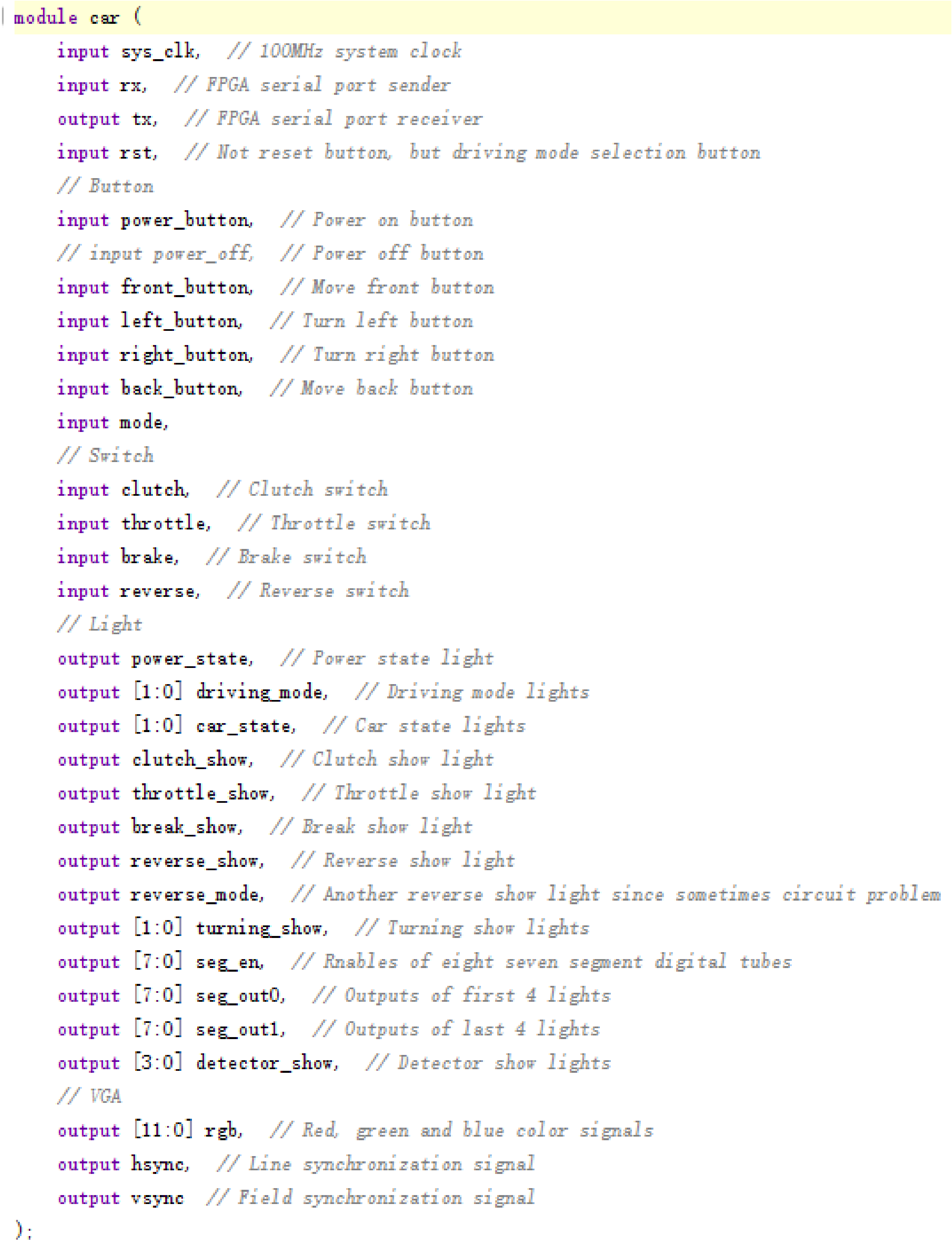
The module ***vga_module*** sues the clk of top module, state signal of module ***state_machine*** and record signals of ***record_module***. Its outputs rgb, hsync, vsync are sent to the top module.

3.3 Function, input and output, types and design architecture of the top module and each sub module

3.3.1 Top Module

- ***car***

This module is the top module and the main module of the whole system design.



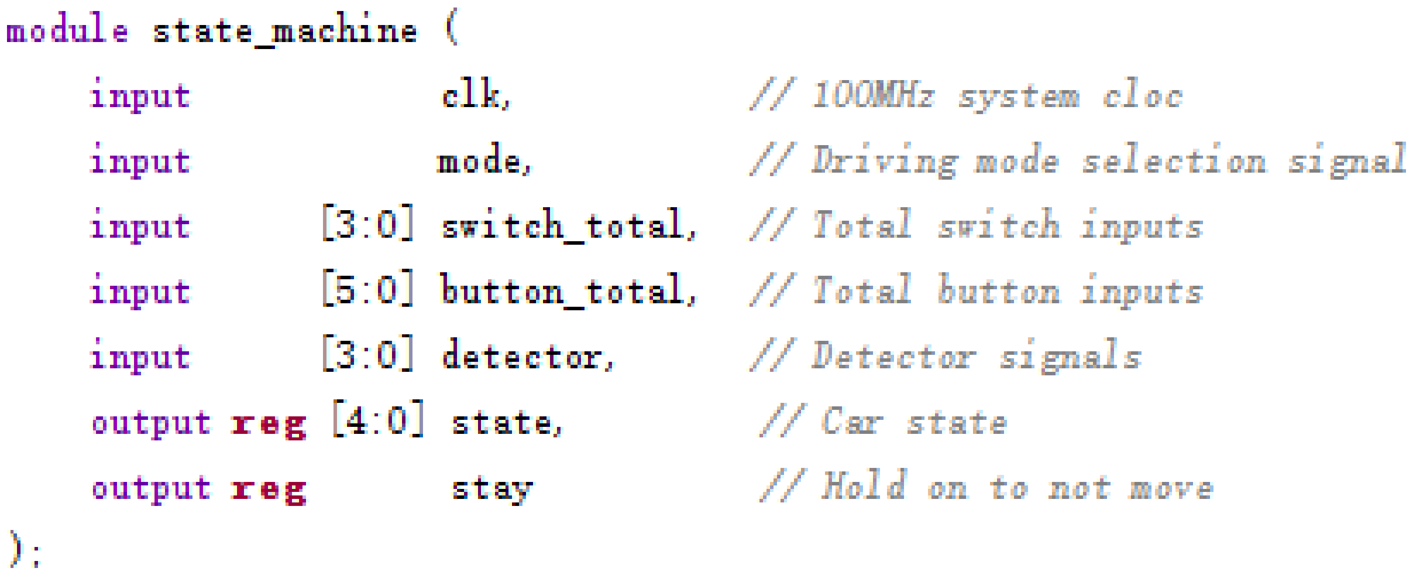
```
module car (
    input sys_clk,  // 100MHz system clock
    input rx,  // FPGA serial port sender
    output tx,  // FPGA serial port receiver
    input rst,  // Not reset button, but driving mode selection button
    // Button
    input power_button,  // Power on button
    // input power_off,  // Power off button
    input front_button,  // Move front button
    input left_button,  // Turn left button
    input right_button,  // Turn right button
    input back_button,  // Move back button
    input mode,
    // Switch
    input clutch,  // Clutch switch
    input throttle,  // Throttle switch
    input brake,  // Brake switch
    input reverse,  // Reverse switch
    // Light
    output power_state,  // Power state light
    output [1:0] driving_mode,  // Driving mode lights
    output [1:0] car_state,  // Car state lights
    output clutch_show,  // Clutch show light
    output throttle_show,  // Throttle show light
    output break_show,  // Break show light
    output reverse_show,  // Reverse show light
    output reverse_mode,  // Another reverse show light since sometimes circuit problem
    output [1:0] turning_show,  // Turning show lights
    output [7:0] seg_en,  // Rnables of eight seven segment digital tubes
    output [7:0] seg_out0,  // Outputs of first 4 lights
    output [7:0] seg_out1,  // Outputs of last 4 lights
    output [3:0] detector_show,  // Detector show lights
    // VGA
    output [11:0] rgb,  // Red, green and blue color signals
    output hsync,  // Line synchronization signal
    output vsync  // Field synchronization signal
);
```

Figure 5 I/O port specifications of module ***car***

3.3.2 Submodules

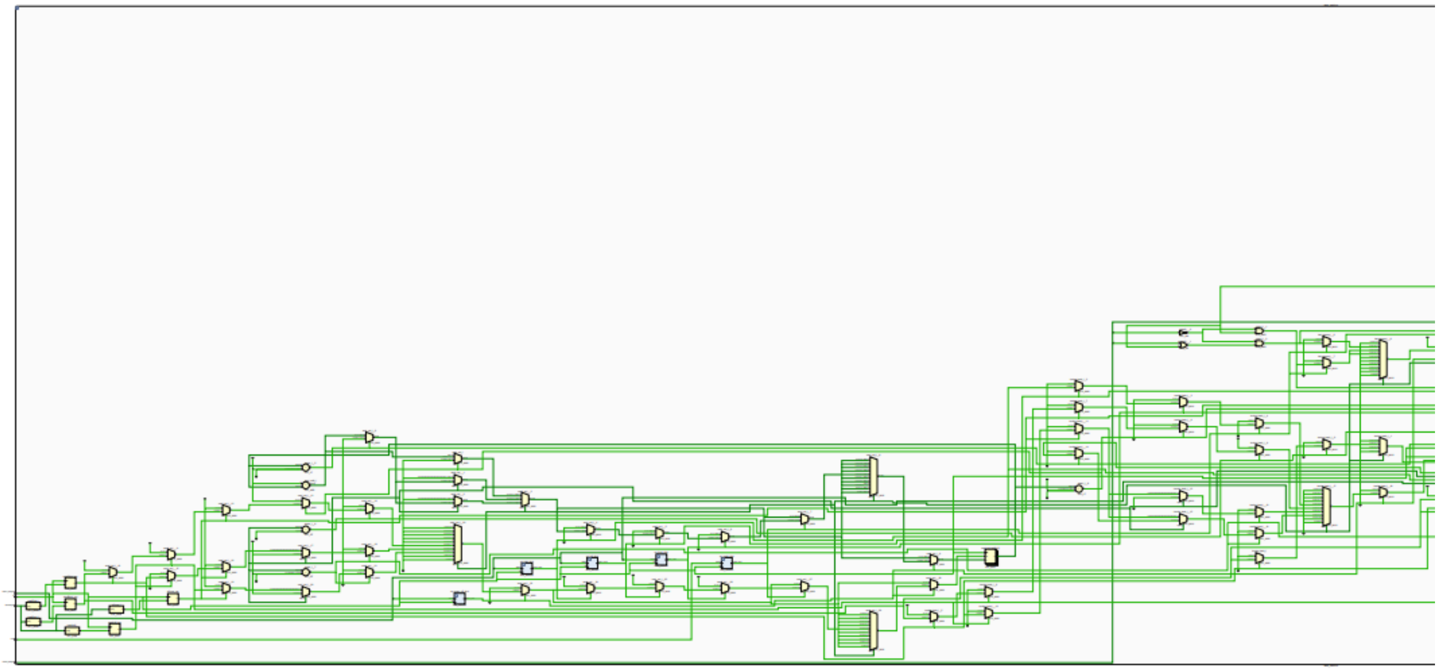
- ***state_machine***

The module is the state machine module, which realizes the function of switching the state of the car.



```
module state_machine (
    input              clk,          // 100MHz system cloc
    input              mode,         // Driving mode selection signal
    input        [3:0] switch_total, // Total switch inputs
    input        [5:0] button_total, // Total button inputs
    input        [3:0] detector,     // Detector signals
    output reg   [4:0] state,        // Car state
    output reg         stay          // Hold on to not move
);
```

Figure 6 I/O port specifications of module ***state_machine***



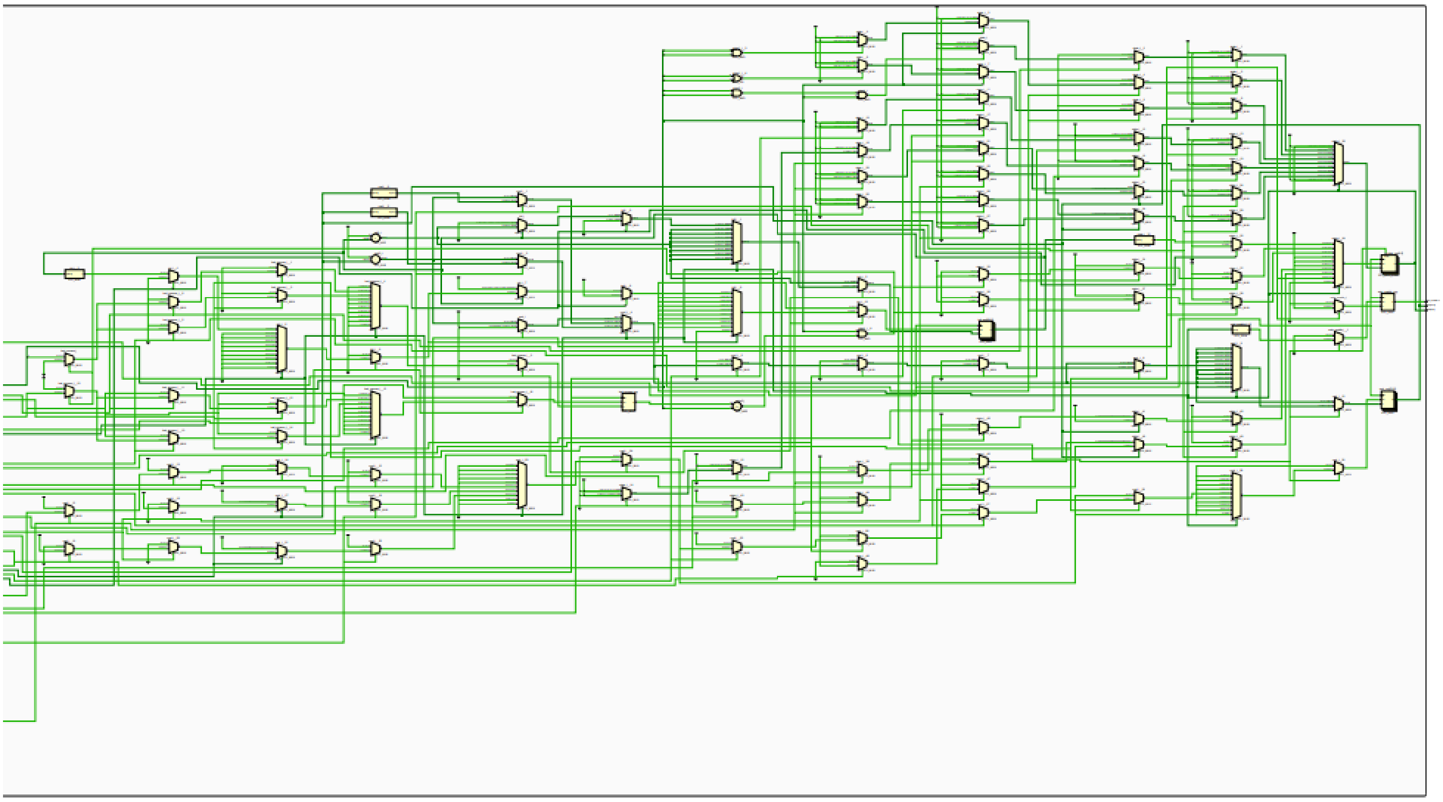
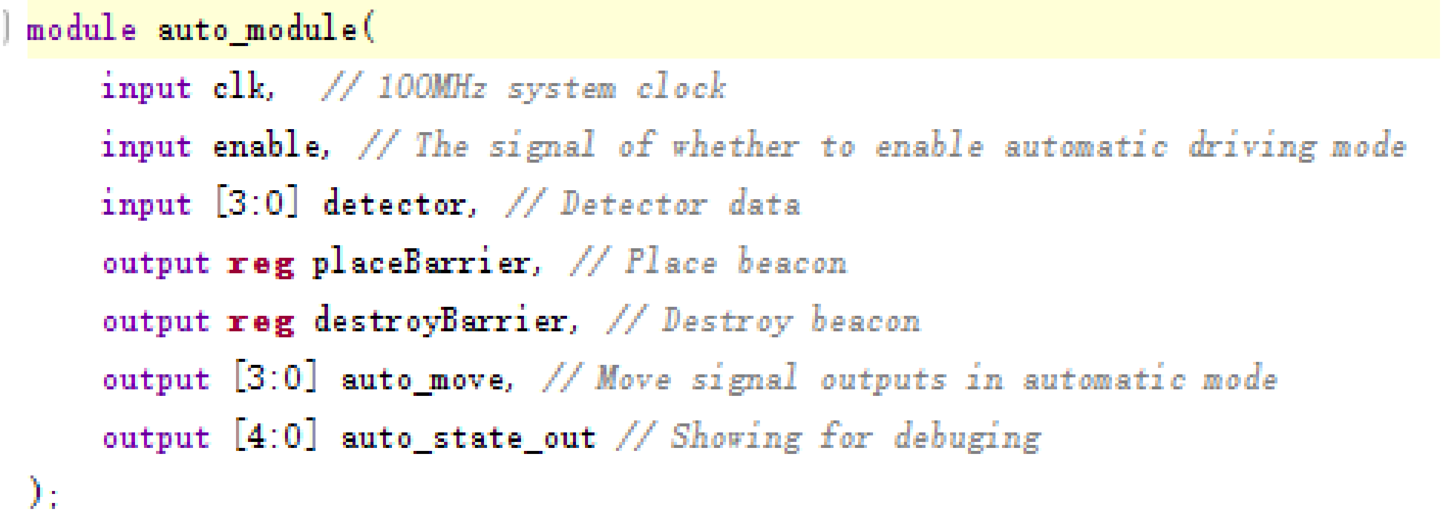


Figure 7 Architecture diagram of module ***state_machine***

- ***auto_module***

The module is an automatic driving module, which realizes the function of controlling the automatic driving of the car.



```
module auto_module(
    input clk,   // 100MHz system clock
    input enable, // The signal of whether to enable automatic driving mode
    input [3:0] detector, // Detector data
    output reg placeBarrier, // Place beacon
    output reg destroyBarrier, // Destroy beacon
    output [3:0] auto_move, // Move signal outputs in automatic mode
    output [4:0] auto_state_out // Showing for debuging
);
```

Figure 8 I/O port specifications of module ***auto_module***

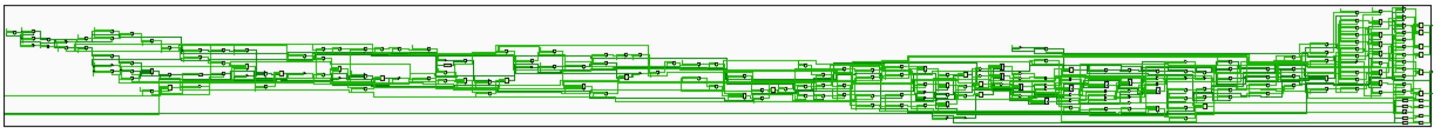
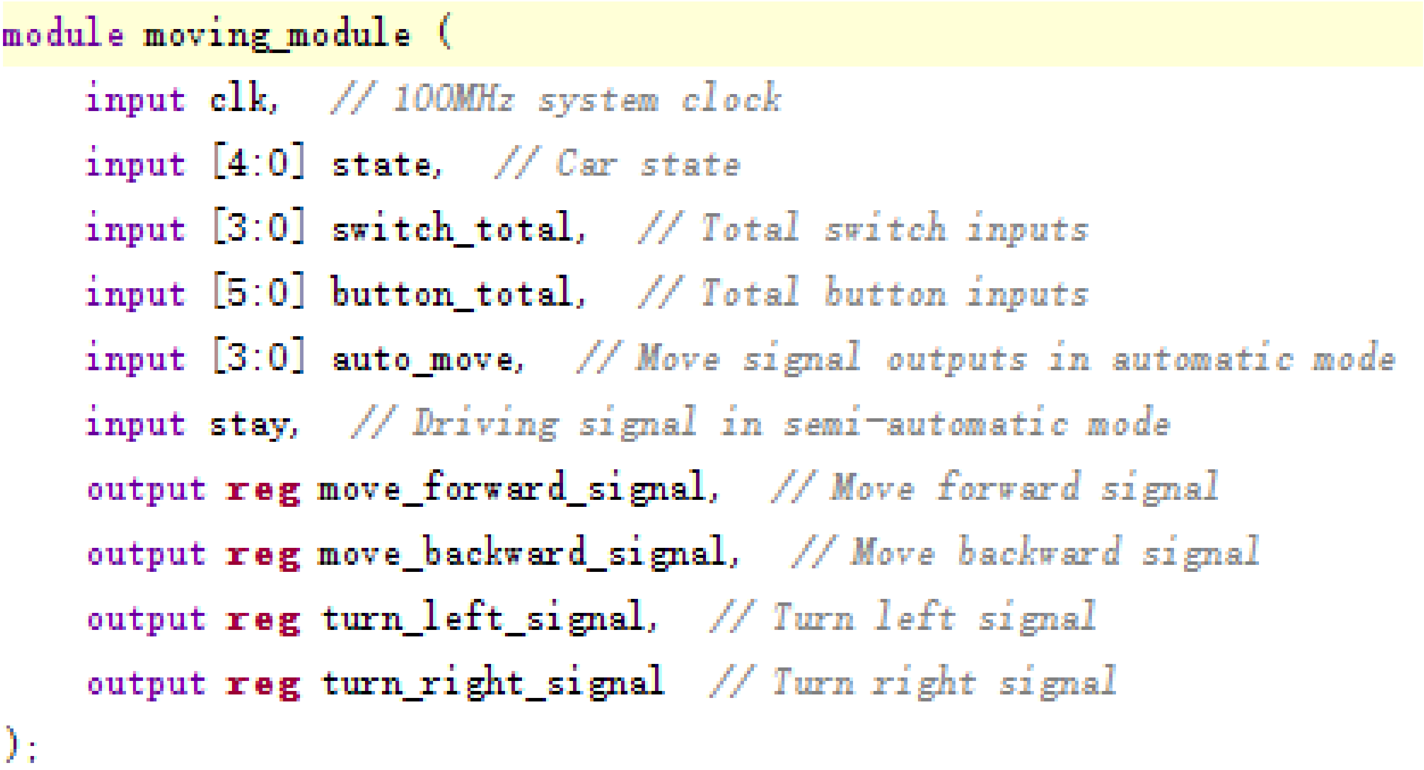


Figure 9 Architecture diagram of module ***auto_module***

- ***moving_module***

The module is the moving module, which realizes the function of controlling the car driving movement.



```
module moving_module (
    input clk,   // 100MHz system clock
    input [4:0] state,   // Car state
    input [3:0] switch_total,   // Total switch inputs
    input [5:0] button_total,   // Total button inputs
    input [3:0] auto_move,   // Move signal outputs in automatic mode
    input stay,   // Driving signal in semi-automatic mode
    output reg move_forward_signal,   // Move forward signal
    output reg move_backward_signal,   // Move backward signal
    output reg turn_left_signal,   // Turn left signal
    output reg turn_right_signal   // Turn right signal
);
```

Figure 10 I/O port specifications of module ***moving_module***

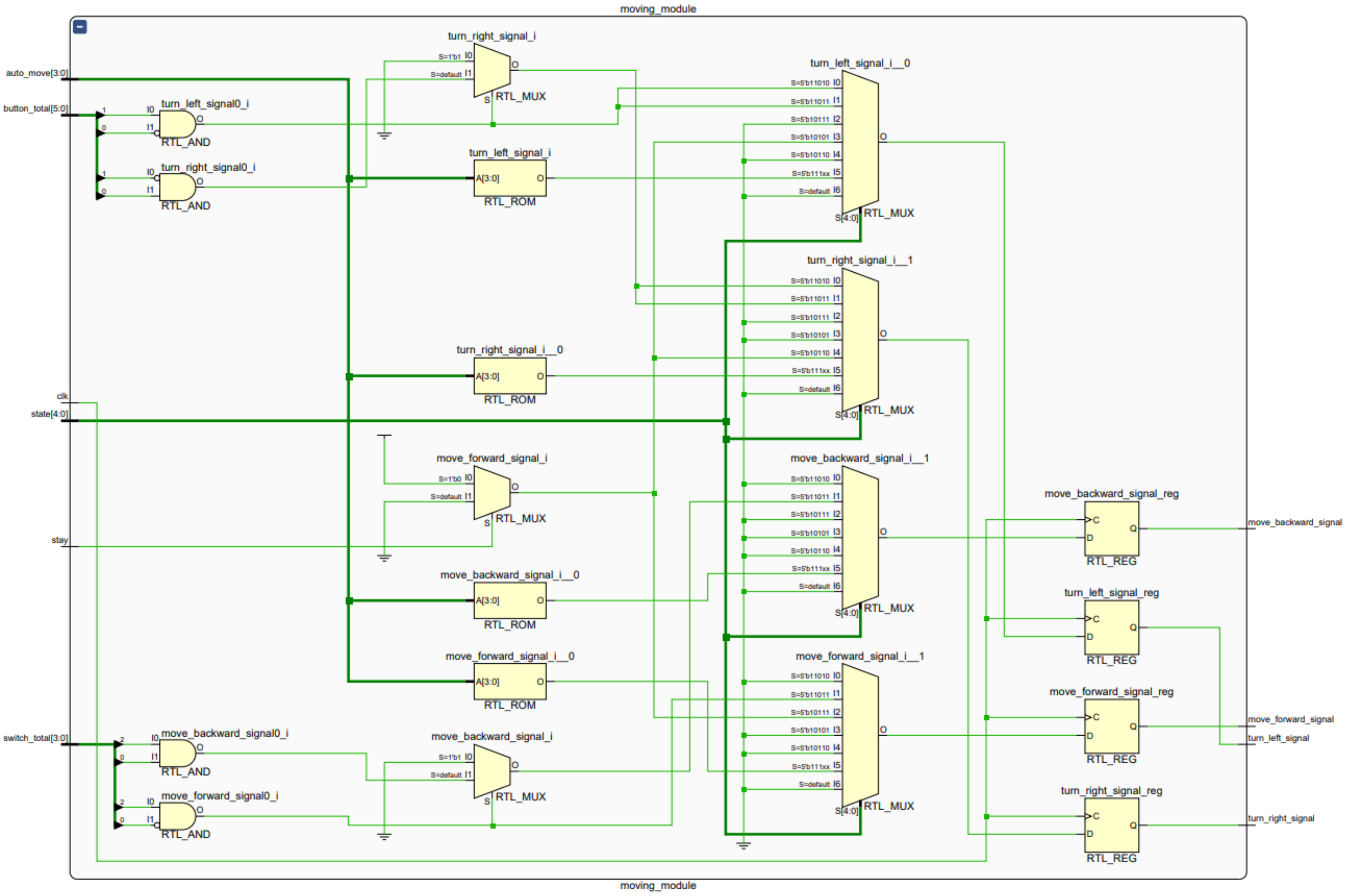
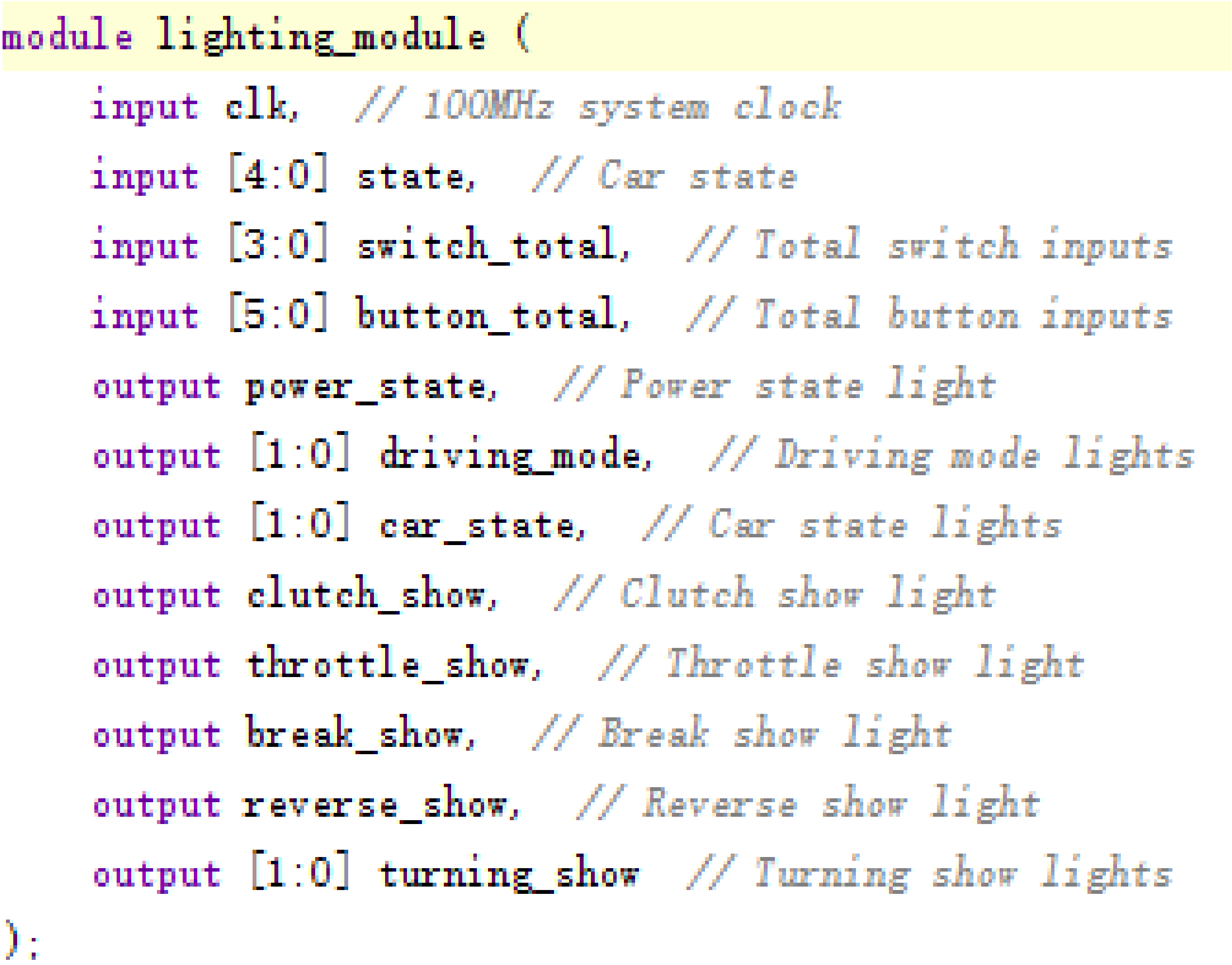


Figure 11 Architecture diagram of module ***moving_module***

- ***lighting_module***

This module is the display light module, which realizes the function of controlling the display of lights.



```
module lighting_module (
    input clk, // 100MHz system clock
    input [4:0] state, // Car state
    input [3:0] switch_total, // Total switch inputs
    input [5:0] button_total, // Total button inputs
    output power_state, // Power state light
    output [1:0] driving_mode, // Driving mode lights
    output [1:0] car_state, // Car state lights
    output clutch_show, // Clutch show light
    output throttle_show, // Throttle show light
    output break_show, // Break show light
    output reverse_show, // Reverse show light
    output [1:0] turning_show // Turning show lights
);
```

Figure 12 I/O port specifications of module ***lighting_module***

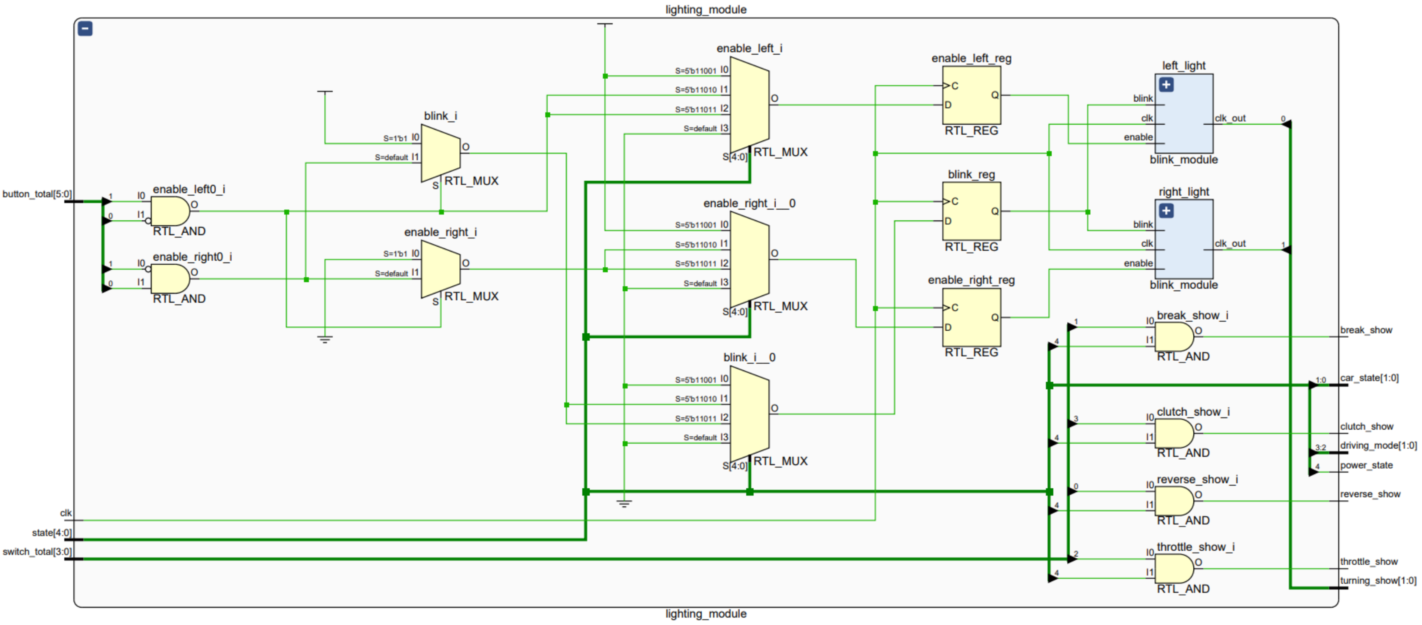
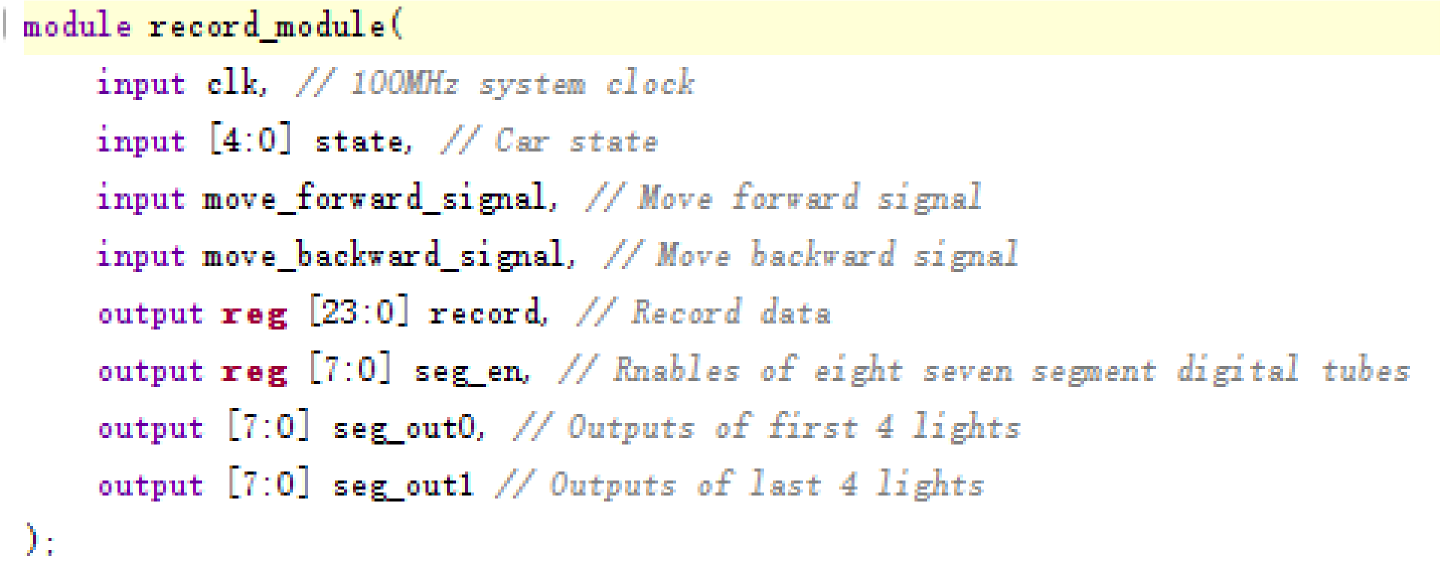


Figure 13 Architecture diagram of module ***lighting_module***

- ***record_module***

The module is mileage recording module, which realizes the function of recording and displaying the mileage of the car.



```
module record_module(
    input clk, // 100MHz system clock
    input [4:0] state, // Car state
    input move_forward_signal, // Move forward signal
    input move_backward_signal, // Move backward signal
    output reg [23:0] record, // Record data
    output reg [7:0] seg_en, // Rnables of eight seven segment digital tubes
    output [7:0] seg_out0, // Outputs of first 4 lights
    output [7:0] seg_out1 // Outputs of last 4 lights
);
```

Figure 14 I/O port specifications of module ***record_module***

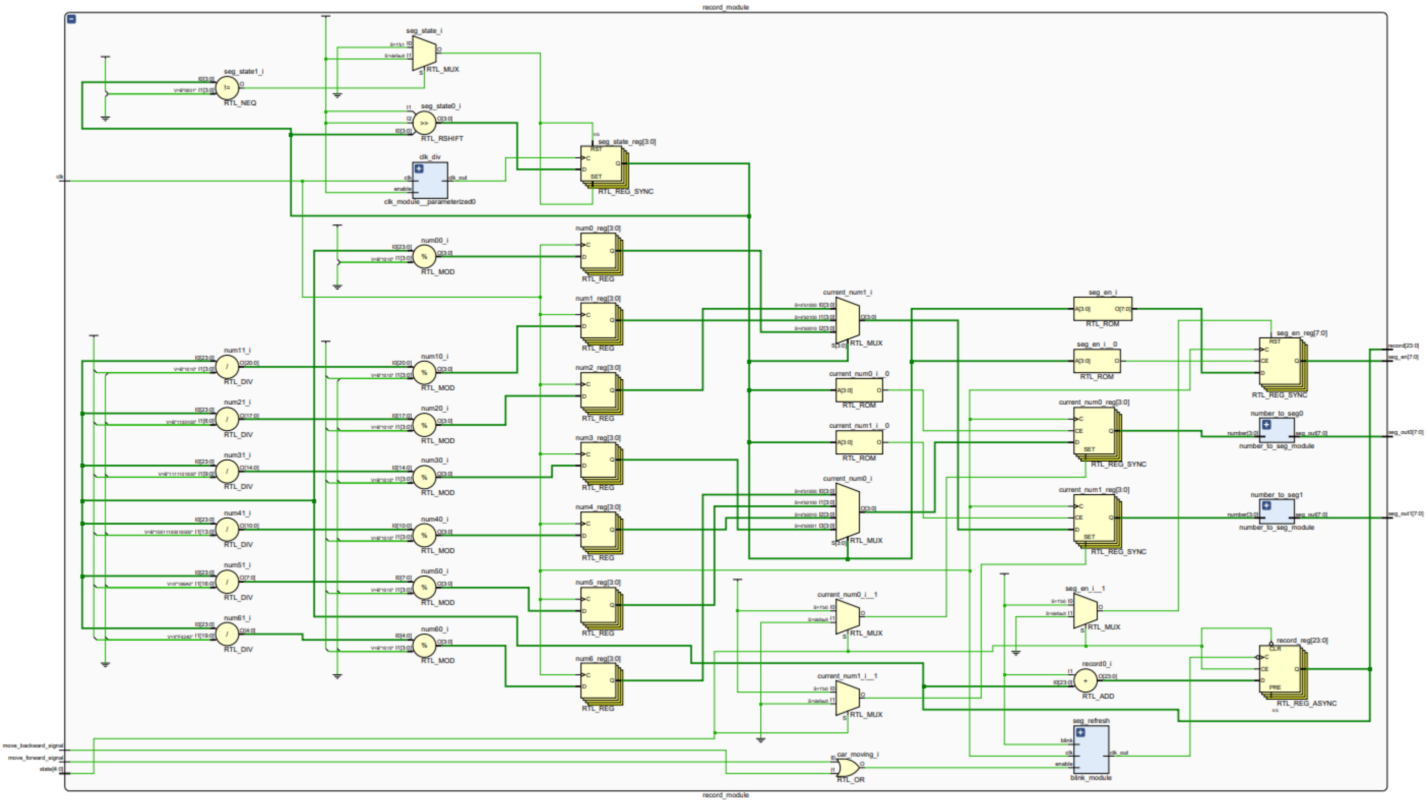
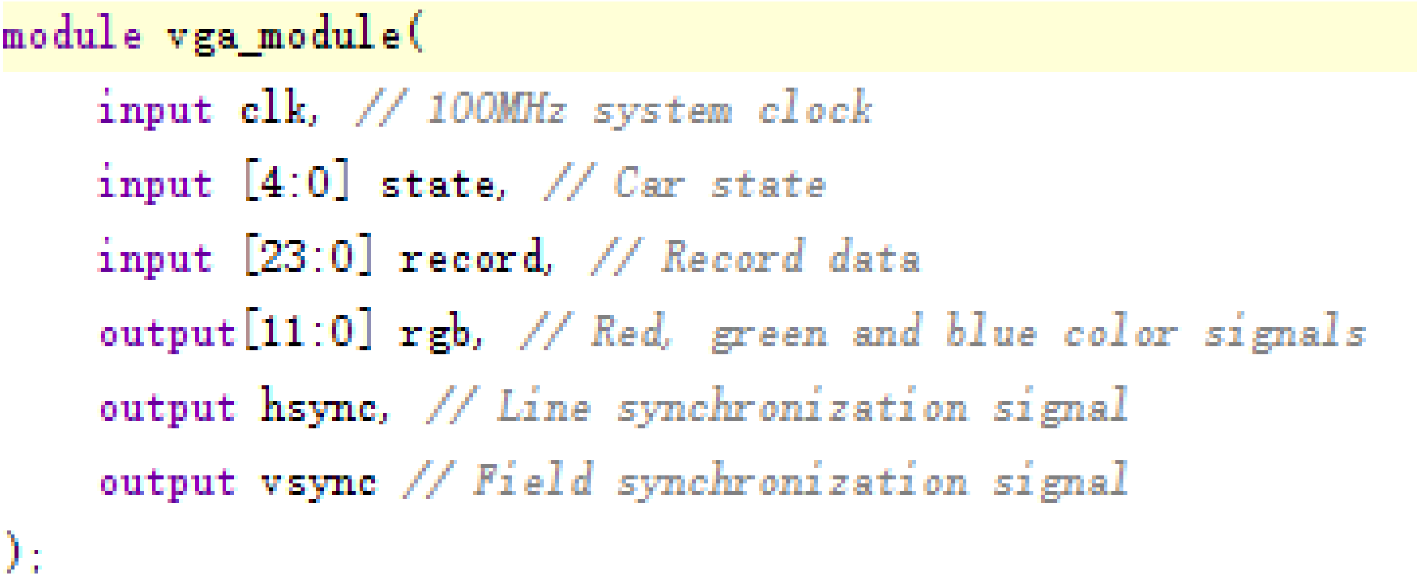


Figure 15 Architecture diagram of module ***record_module***

- ***vga_module***

This module is a VGA module, which realizes the function of controlling the display of VGA content.



```
module vga_module(
    input clk, // 100MHz system clock
    input [4:0] state, // Car state
    input [23:0] record, // Record data
    output[11:0] rgb, // Red, green and blue color signals
    output hsync, // Line synchronization signal
    output vsync // Field synchronization signal
);
```

Figure 16 I/O port specifications of module ***vga_module***

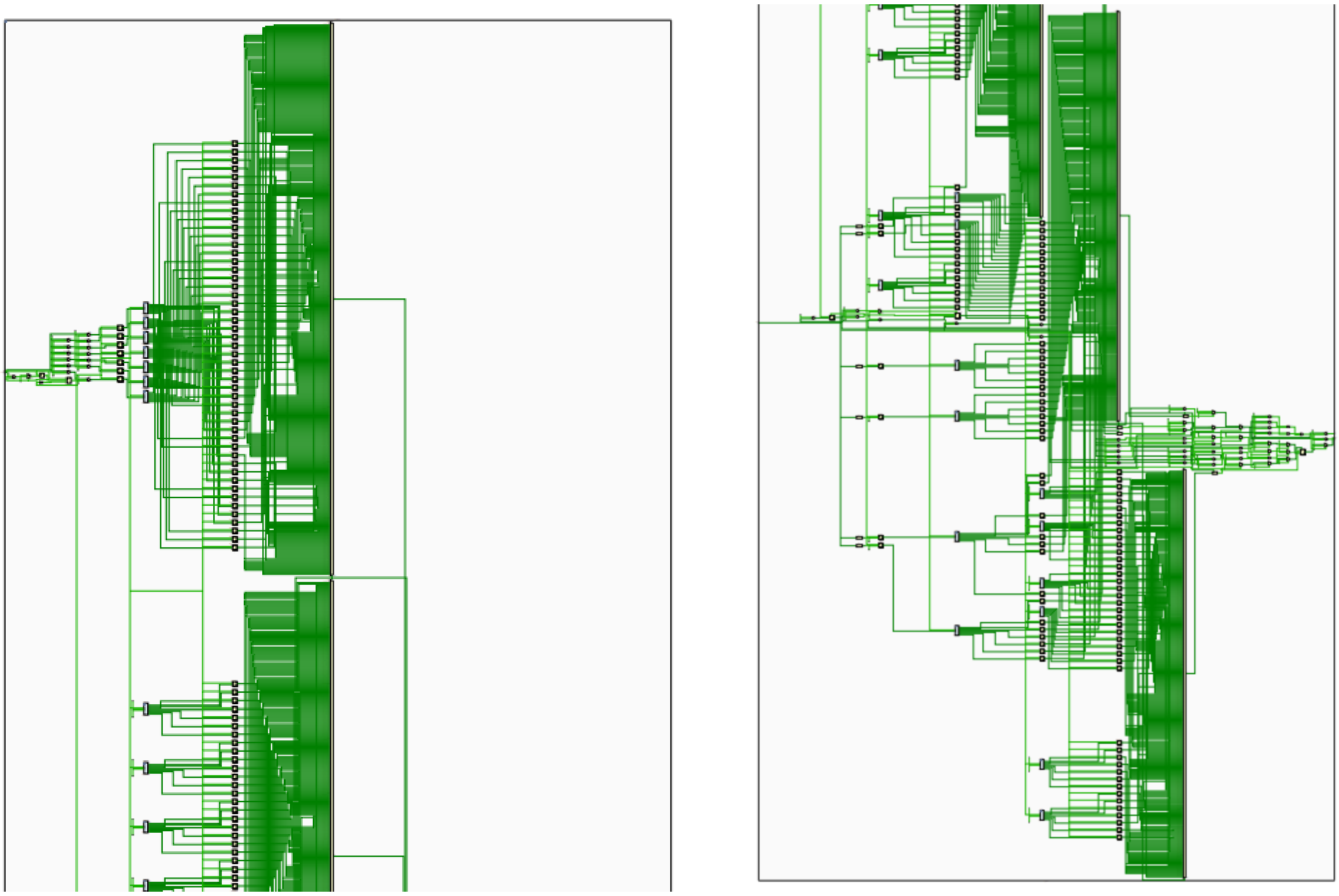
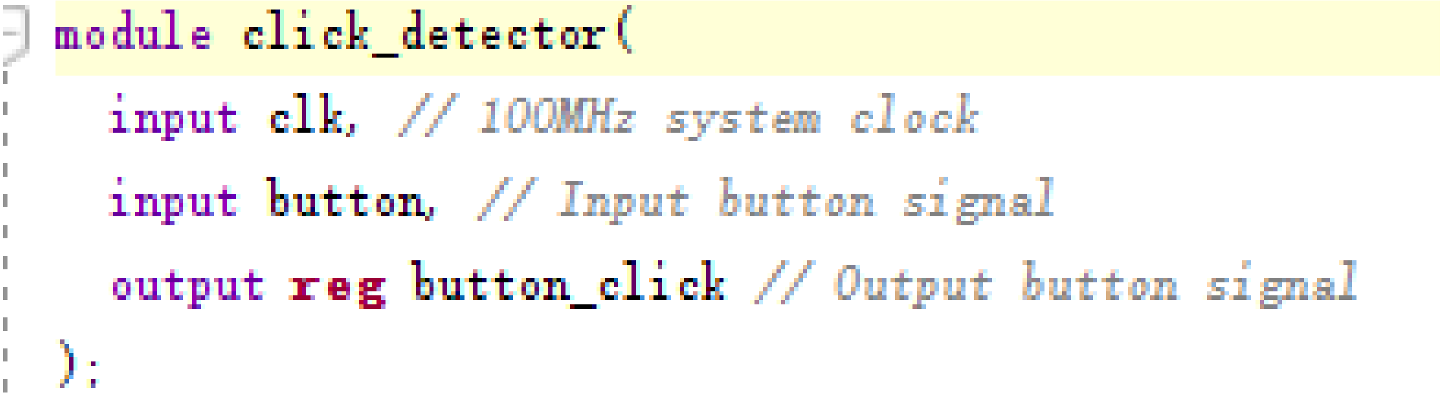


Figure 17 Architecture diagram of *module* ***vga_module***

3.3.3 Other Small Modules

- **click_detector**

This module is the click detection module, which realizes the function of the detection of the long press of buttons, that is, the long press detection of the power on button for 1s.



```
module click_detector(
    input clk, // 100MHz system clock
    input button, // Input button signal
    output reg button_click // Output button signal
);
```

Figure 18 I/O port specifications of module **click_detector**

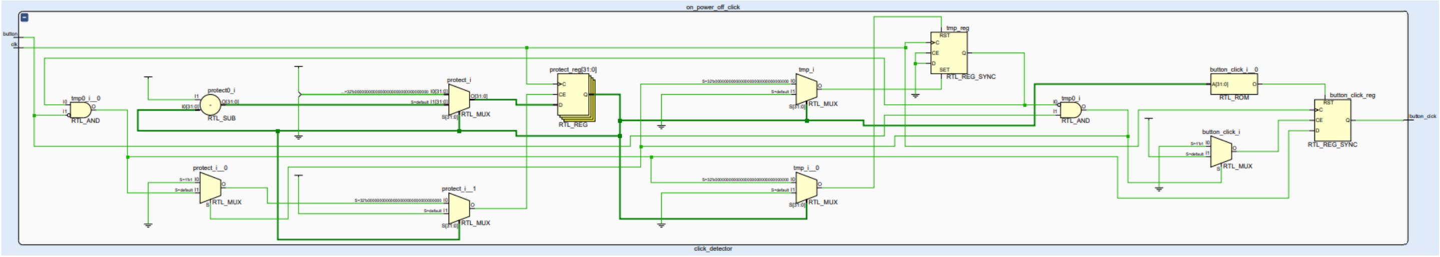
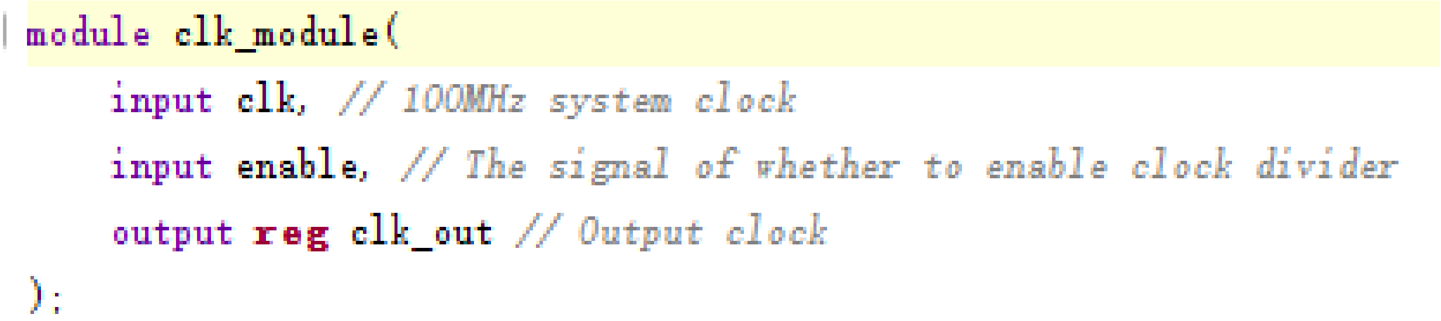


Figure 19 Architecture diagram of module **click_detector**

- **clk_module**

This module is a clock frequency division module, which realizes the function of clock frequency division.



```
module clk_module(
    input clk, // 100MHz system clock
    input enable, // The signal of whether to enable clock divider
    output reg clk_out // Output clock
);
```

Figure 20 I/O port specifications of module **clk_module**

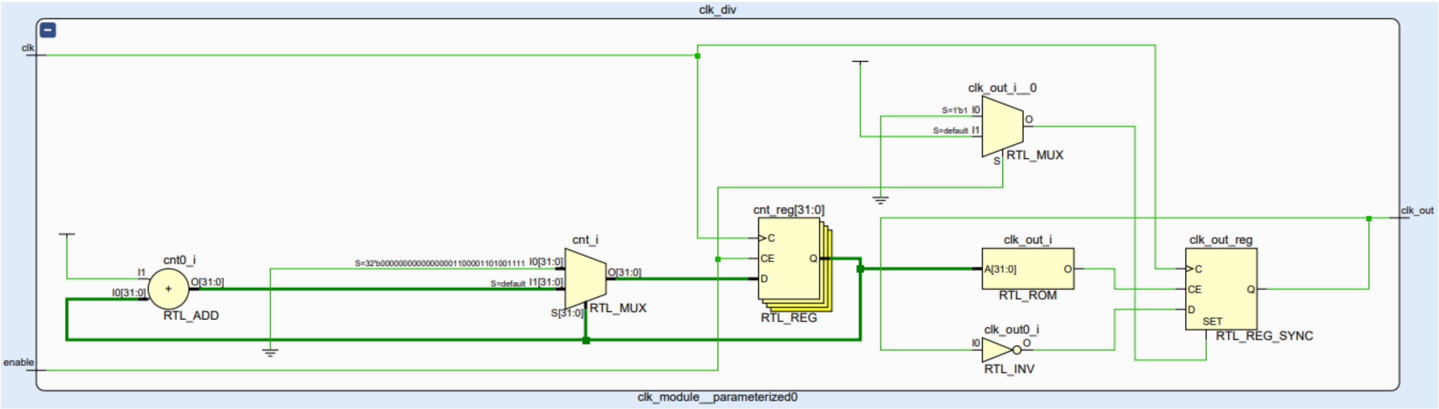
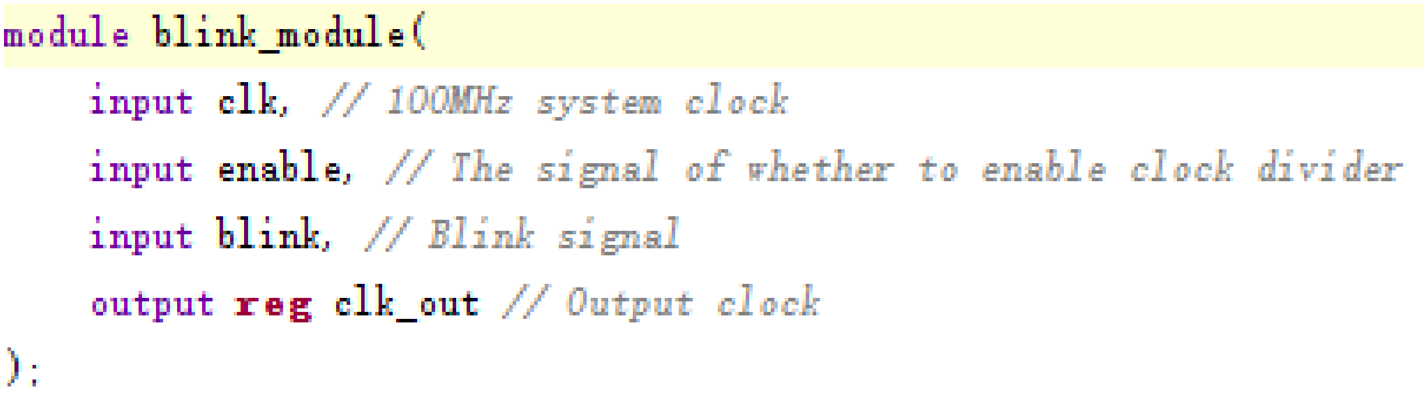


Figure 21 Architecture diagram of module **clk_module**

- **blink_module**

The module is a clock frequency division module, which realizes the function of controlling the flashing of the display light.



```
module blink_module(
    input clk, // 100MHz system clock
    input enable, // The signal of whether to enable clock divider
    input blink, // Blink signal
    output reg clk_out // Output clock
);
```

Figure 22 I/O port specifications of module **blink_module**

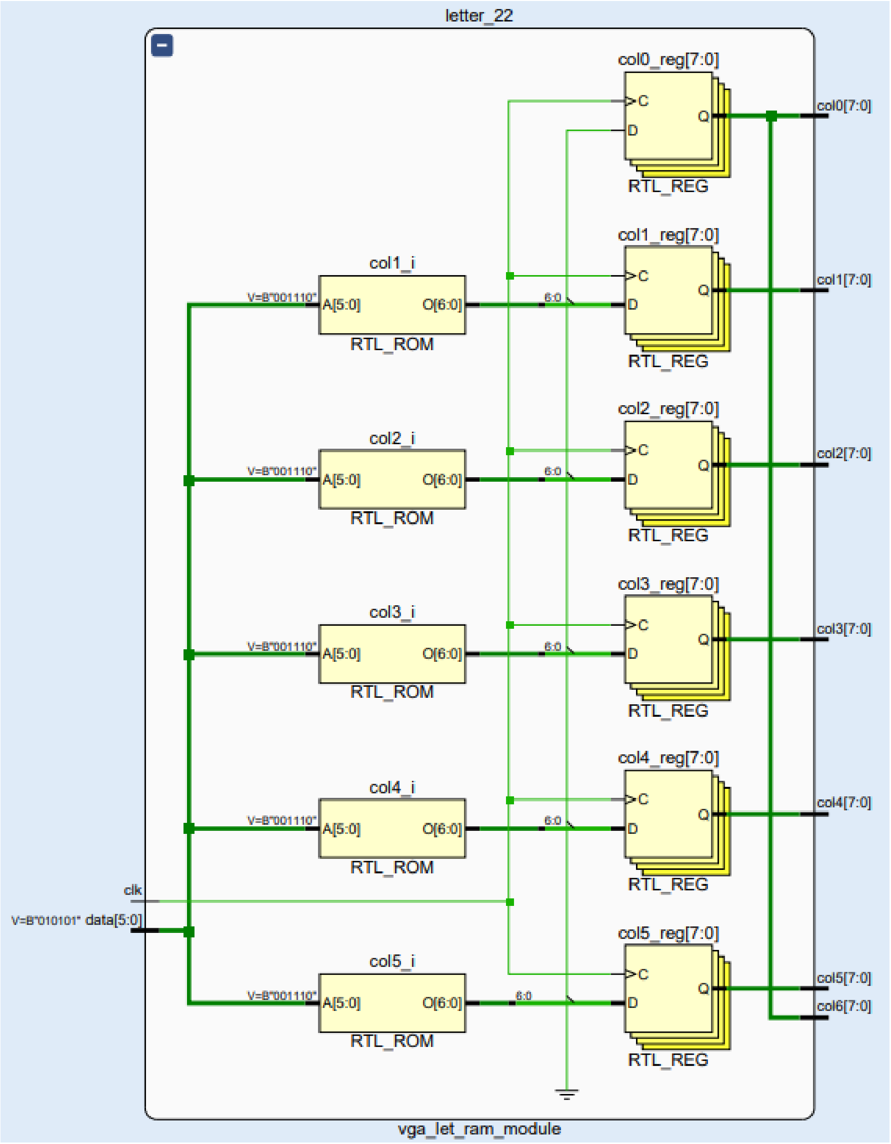
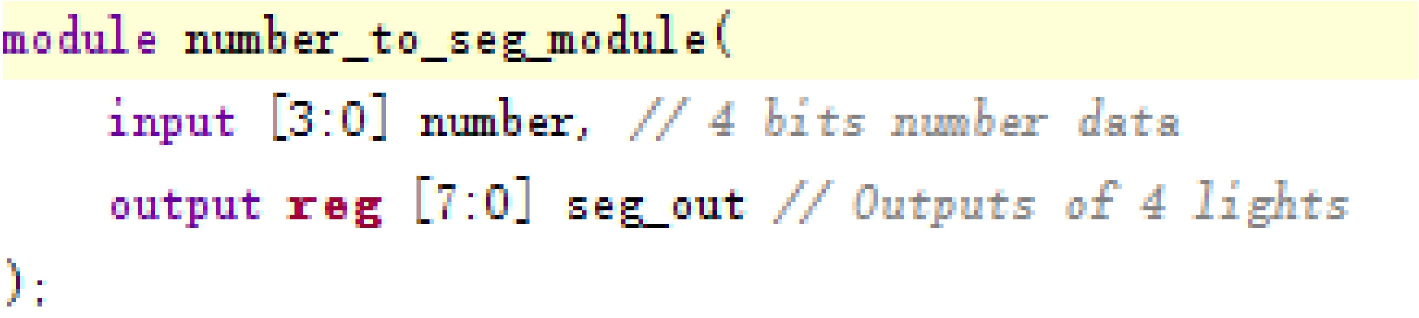


Figure 23 Architecture diagram of module **blink_module**

- **number_to_seg_module**

The module is the number to seven segment data module, which realizes the function of transformation of the number to seven segment data.



```
module number_to_seg_module(
    input [3:0] number, // 4 bits number data
    output reg [7:0] seg_out // Outputs of 4 lights
);
```

Figure 24 I/O port specifications of module **number_to_seg_module**

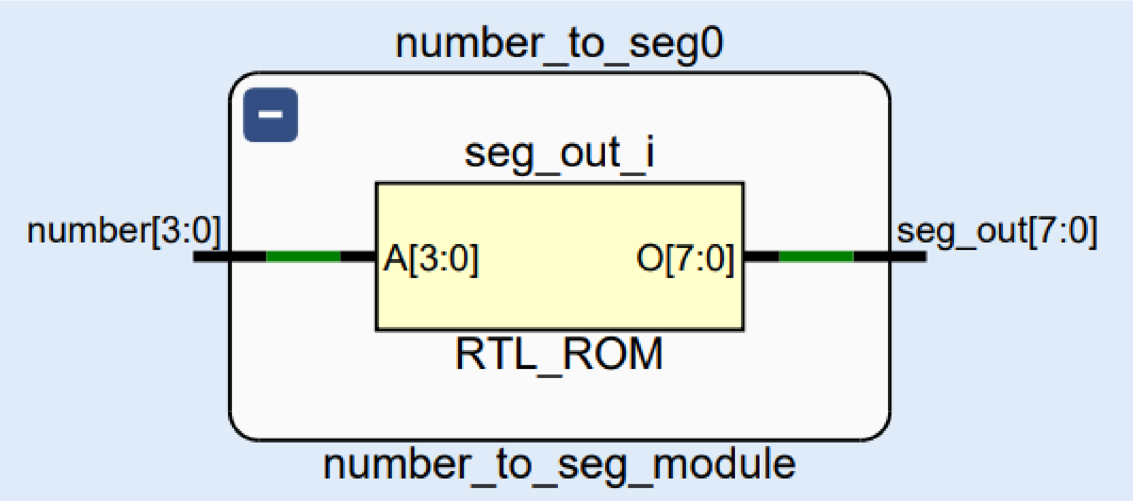
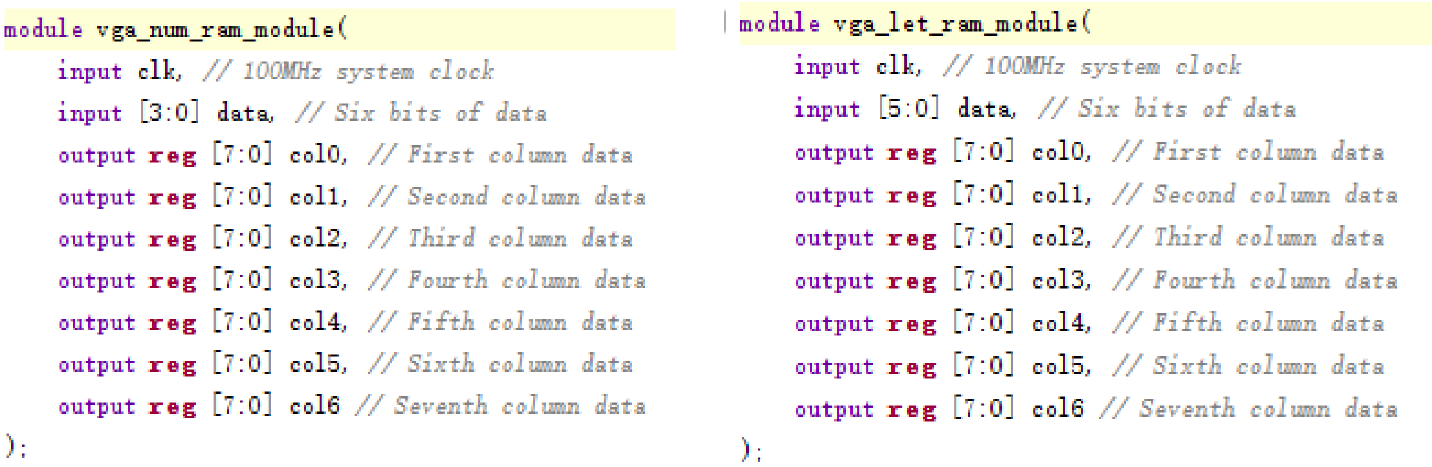


Figure 25 Architecture diagram of module **number_to_seg_module**

- **vga_let_ram_module** and **vga_num_ram_module**

The module is the 4 bits data to columns data module, which realizes the function of transformation of the 4 bits data to columns data.



```
module vga_num_ram_module(
    input clk, // 100MHz system clock
    input [3:0] data, // Six bits of data
    output reg [7:0] col0, // First column data
    output reg [7:0] col1, // Second column data
    output reg [7:0] col2, // Third column data
    output reg [7:0] col3, // Fourth column data
    output reg [7:0] col4, // Fifth column data
    output reg [7:0] col5, // Sixth column data
    output reg [7:0] col6 // Seventh column data
);
```

```
module vga_let_ram_module(
    input clk, // 100MHz system clock
    input [5:0] data, // Six bits of data
    output reg [7:0] col0, // First column data
    output reg [7:0] col1, // Second column data
    output reg [7:0] col2, // Third column data
    output reg [7:0] col3, // Fourth column data
    output reg [7:0] col4, // Fifth column data
    output reg [7:0] col5, // Sixth column data
    output reg [7:0] col6 // Seventh column data
);
```

Figure 26 I/O port specifications of module **vga_let_ram_module** and **vga_num_ram_module**

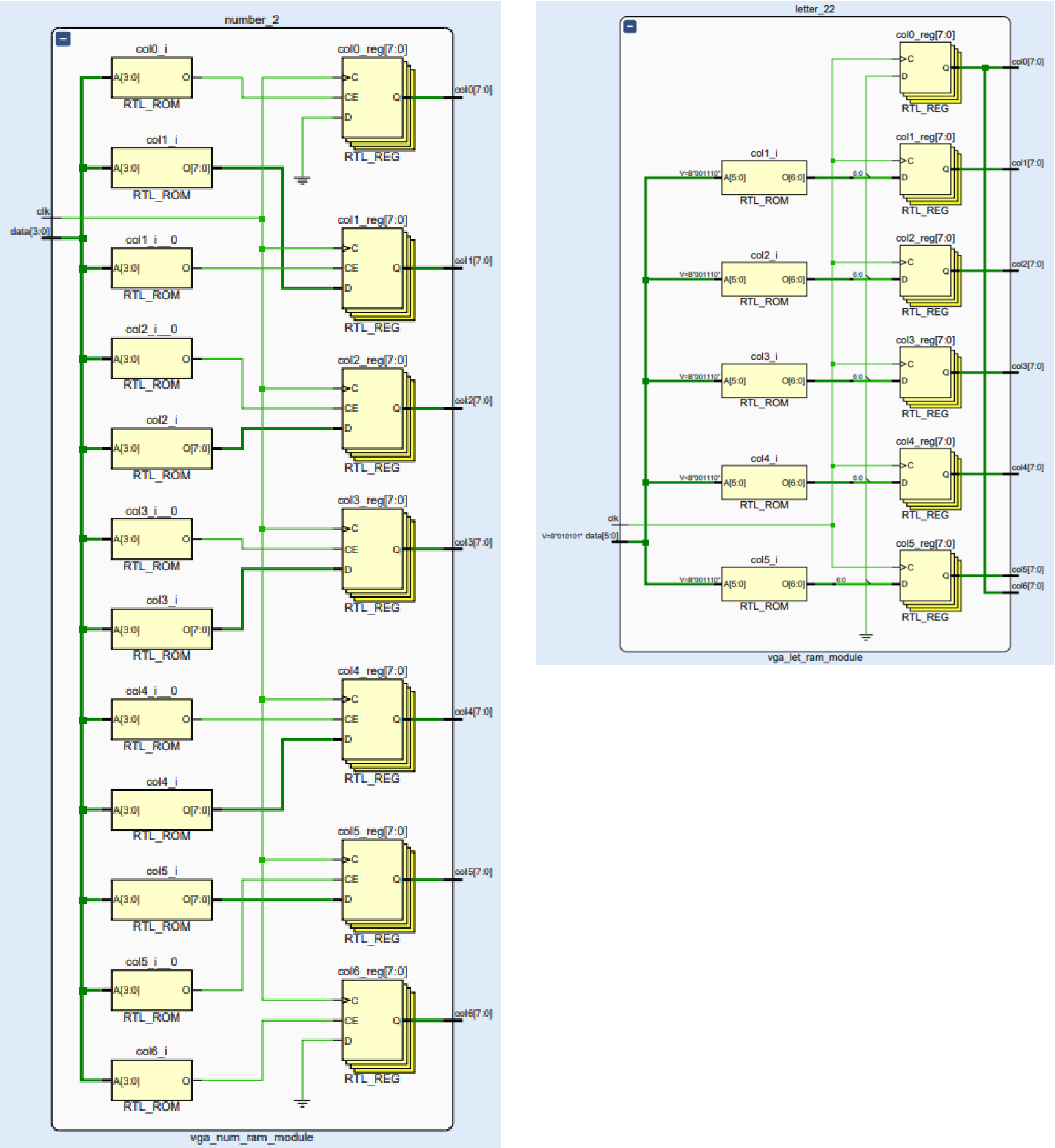
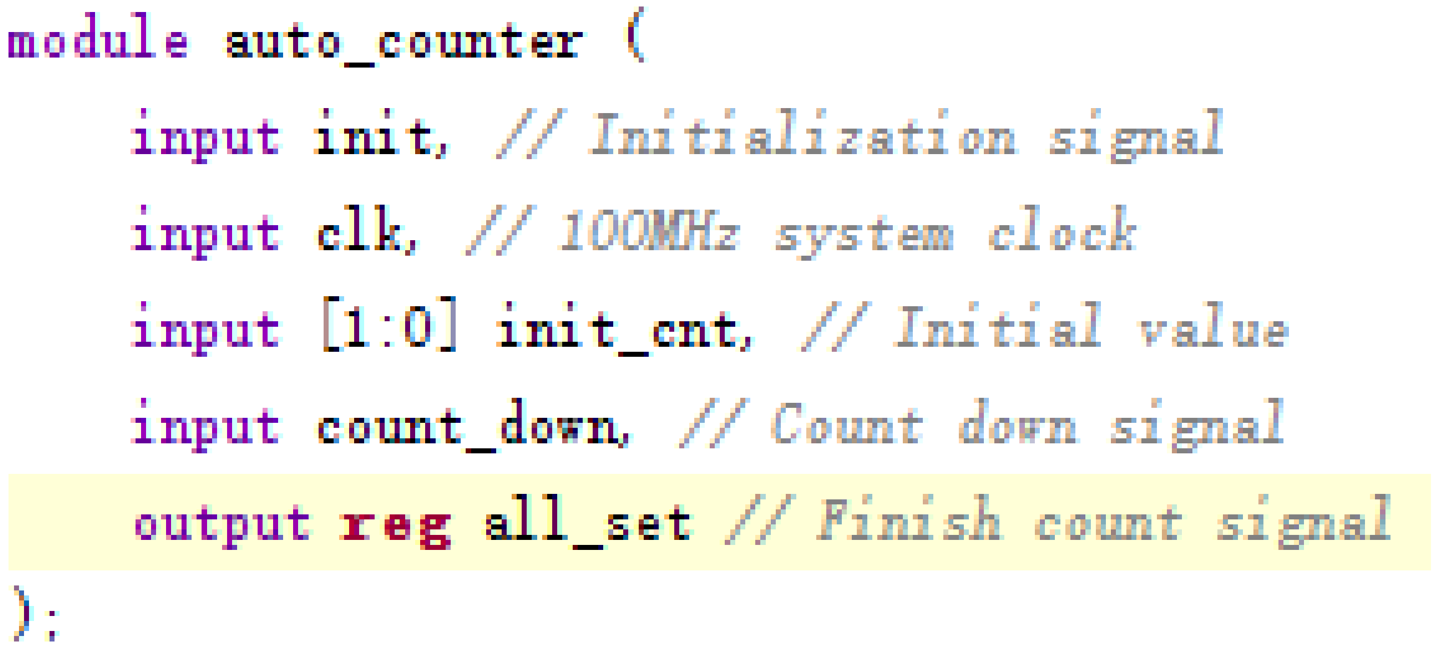


Figure 27 Architecture diagram of module **vga_let_ram_module** and **vga_num_ram_module**

- **auto_counter**

The module is the crossing counter module , which realizes the function of the recording and processing of the number of th e crossing on automatic driving mode.



```
module auto_counter (
    input init, // Initialization signal
    input clk, // 100MHz system clock
    input [1:0] init_cnt, // Initial value
    input count_down, // Count down signal
    output reg all_set // Finish count signal
);
```

Figure 28 I/O port specifications of module **auto_counter**

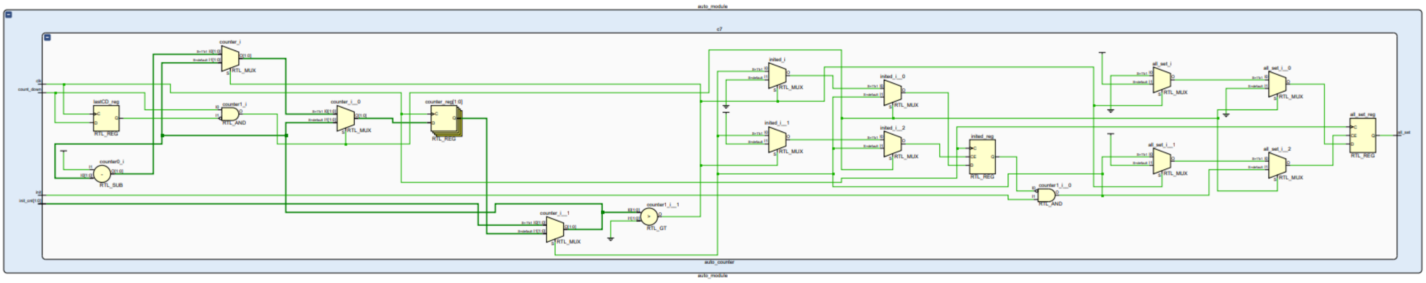


Figure 29 Architecture diagram of module **auto_counter**

4 Summary of Experience

4.1 Group member 冯泽欣

4.1.1 Personal Perspective of Difficulties

- The implementation of basic features is not really hard. If you fully understand the state machine, that will be super fast and you will have a better understanding of state machine.
- The Auto part is kind of difficult because we don't have much information of the map, choosing the solution algorithm is really hard. And the implementation of DFS by verilog is hard too.

4.2.2 Knowledge/Experience Gained

- Difference between blocking assignments and non-blocking assignments
- How to implement VGA (hsync, vsync)
- Removing joggle for push button

- Writing vrilog fluently
- Better Understanding of programming differences between software and hardware language
- How to use state machines to solve problems.
- How to use the knowledge theorems we learned in class to implement features.

4.3.3 Some Possible Improvements

- Better auto algorithm
- Make code more clear
- Find simpler ways to solve problems or implement features. So we can have less cost on extra gates

4.2 Group member 林俊杰

4.2.1 Personal Perspective of Difficulties

- Understanding and analysis of the overall project system design.
- Understanding and writing hardware description language verilog.
- The drawing of our project state diagram.
- The learning and understanding of VGA display design principle and its code implementation.

4.2.2 Knowledge/Experience Gained

- The knowledge learning of combinatorial logic and sequential logic
- Deeper understanding of the drawing of the state diagram.
- Understanding of the concept of the state diagram and drawing precautions, as well as more knowledge about the state diagram, such as the study of the UML diagram, the important role of the state diagram in system design and the representation of the state diagram drawing.
- The understanding of principles in the VGA module.
- The understanding of in-line synchronization timing and field synchronization timing in the VGA display process.
- Verilog language coding design of driving circuit.
- Learning of report writing for the system design project.

4.2.3 What can be improved

If I had more time, I would spend more time designing the VGA module to improve and beautify the VGA display, not only the simple display of car status and mileage, but also the design of a more beautiful VGA display and more in line with practical applications. Certainly, I'll seriously take the time to consolidate the knowledge about combinatorial logic, sequential logic and learn more about state diagrams.

4.3 Group member 丁健乐

4.3.1 Difficulties Encountered

While writing the state machine of manual driving, it appeared red X form when I was doing simulation, and I found that it needed to use caseX instead of case when writing the state machine.

Also, when we were writing the auto driving module, we originally wanted to use dfs to do this, but when the sample map came out, we found that using dfs would take a long time, so we changed our method.

4.3.2 Knowledge/Experience acquired

During this project:

- Firstly, I learned how to do a project in its entirety, including designing, code implementation and report writing.
- Secondly, I learnt the difference between one-stage state machine and two-stage state machine and I learnt how to write these two kinds of state machine.
- Thirdly, I learnt the detailed working process of different modules and a whole project.
- Last but not least, I learned how to use verilog to implement different modules and combine modules into a whole project.

4.3.3 Some Possible Improvements

- Optimize the code structure to make the resulting image more concise.
- Assist team members to optimize the code of the VGA module and make its final display more neat and concise.

Thank you for reading!