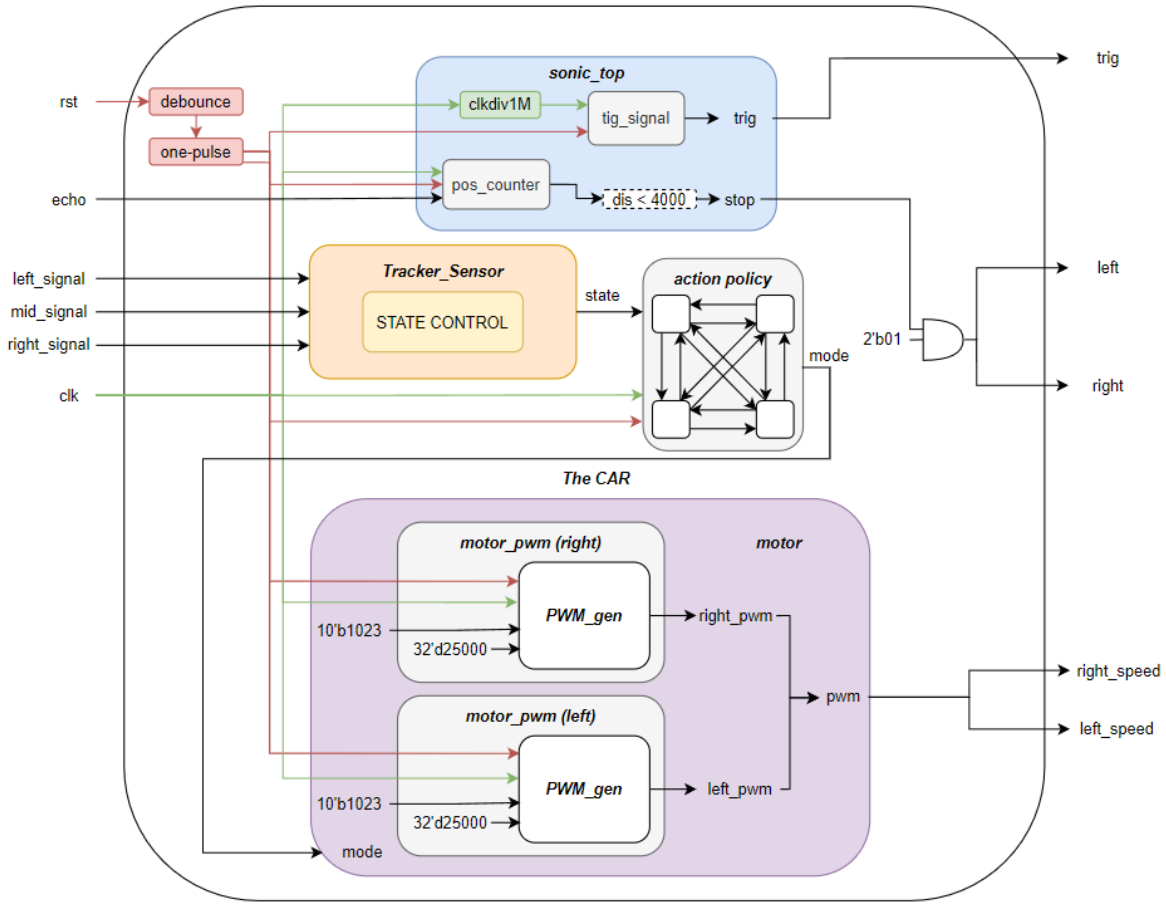
LAB 6 REPORT

(FPGA) The CAR

Team 37:

1. 徐美妮 Mary Madeline Nicole 109006205
2. 林之耀 Kevin Richardson Halim 109006277

I. Block diagrams and state transition diagrams



1.1. The CAR block diagram

II. Design Process

The concept of the CAR module is to formulate an action policy based on the signal from the tracker sensor pointing on the ground, while dealing with the situation where an obstacle within 40 cm is detected by the sonic sensor. In the initial design of this module, we considered adjusting the motor speed and steering at the same time for different conditions, but after testing we found out that good results could be obtained when we adjust the motor steering.

The top module is mainly responsible for concatenating each submodule and processing the state transition required for action. In total of 3 modules, which are ***motor***, ***sonic_top***, and ***tracker_sensor***. Each module I/O will be processed by the **Top** module to pull off required actions. The ***sonic_top*** and ***tracker_sensor*** modules will provide the input of this design and the ***motor*** will act as the output and responsible to rotate the wheels which are placed at the rear car.

The ***sonic_top*** where ***stop*** will turn to 1 when the distance between the ultrasonic sensor and an obstacle, and ***left_signal***, ***middle_signal***, ***right_signal*** will sense and give 1 when a light senses and these sensors placed underneath the front car and these signals will determine a ***state***. The ***motor*** will act as the output and responsible to rotate the wheels which are placed at the rear car, the module will take a ***mode*** as an input and it's determined by ***state*** and previous ***mode***.

Details about ***tracker_sensor***. The ***tracker_sensor***'s output ***state*** will change depending on the signals or both the signals and it's previous ***state***. The ***state*** has 8 states. **OUT** acts as an error, where unknown behavior occurs. **MID** acts where all signals are turned on. **L** and **R** where **L** as left and **R** as right consist of **1**, **2**, **3** to determine how much the sensor has outbound to the left or right.

{***left_signal***, ***middle_signal***, ***right_signal***}	***state***	***next_state***
{1, 1, 1}	x	MID
{1, 1, 0}	x	L1
{1, 0, 0}	x	L2
{0, 0, 0}	L1 or L2 or L3	L3
{0, 1, 1}	x	R1
{0, 0, 1}	x	R2
{0, 0, 0}	R1 or R2 or R3	R3

2.1. The CAR state diagram

Details about ***sonic_top***. The ***sonic_top's*** output is ***stop*** and it's assigned depending on the distance is calculated by the time where ***trig*** is triggered and when the ***echo*** is detected. The interval time between those will be calculated and give a ***dis*** and its unit is in 1/10 millisecond. Therefore, the ***stop*** is assigned to be 1 when ***dis*** is less than 4000.

The ***motor*** module would be responsible as an output by ***mode*** which is processed by ***next_mode*** inside the ***Top*** module and depends on the ***state*** and the previous ***mode***. The mode consists of 7 states. **HARD_TURN_L** to trigger the right wheel motor at full speed and left motor to stop, vice versa with **HARD_TURN_R**. **MAINTAIN** to maintain the angle but at the maximum speed. **TURN_L** to increase the right motor at full speed and decrease the left speed, vice versa with **TURN_R**. **MIDDLE** to set both motors at full speed, and **STOP** puts all motors to stop.

The state default or reset is used for the car to only go straight forward, because basically when you are about to start (when you start moving), you will press reset. If there is an obstacle within 40 cm in front of the car, It will stop, which means staying stationary on the road. The left or right will sense the road boundary. Which means that if the tracker sensor senses a black road at the left side, it needs to steer right and vice versa. So we stop the left wheel and lose the right to keep moving forward. Using all the modules with each responsible to its machine which is binded with a jumper from the Pmods.

III. Design Test

We tested the success of our program by testing it directly on the FPGA. For the CAR, we test it using 16 leds on the FPGA and bind it with a variable for the sake of observing how it will behave. Because we program our FPGA so that it will light up when it detects certain conditions, like turning left a little, turning more left, or sharp turn left. By this, we can test and see whether or not we programmed our CAR accordingly.

After some repeated debugging by tweaking some statements and conditions and a bunch of testing to trigger specific sensors, at the end our module works as expected.

IV. What we have learned from Lab 6 FPGA

- ➔ Program FPGA to an external device and external power source.
- ➔ Using an ultrasonic sensor to detect an object in front of it and its distance.
- ➔ Using a 3-tracker sensor to detect bright and dark stuff and implement it as a line scanner to the CAR module.
- ➔ Using motors by controlling each motor's speed with PWM and a signal to change direction of its spin.
- ➔ Design a verilog program to process sensor signals to pull off some complex action that depended on a specific external environment.
- ➔ Setting up multiple jumper cables and its positions to send signals or send some power.
- ➔ Handling external problems, outside the programmed FPGA, real physical problems that intervene us to the goals.

V. List of contributions

Mary Madeline Nicole 徐美妮 109006205 & Kevin Richardson Halim 林之耀 109006277

- Draw state diagram and block diagram
- Module The CAR
- Make report

VI. Problems we faced

- The car keeps moving in a straight line even after it's out of the track.
- The noise produced by an external error (the imperfectness of a sensor).
- Predict, design, and program FPGA to behave based on the environment where it's on.