Graduate Final Project

Racing Game Controlled by Ultrasonic Sensors

EE253

Liyuan Wang, Chang Lu

1. Introduction

This report is about our graduate final project in course EE253: embedded system in University of California, Santa Barbara. Our final project is to design a racing game displayed on LCD screen, controlled by buttons and ultrasonic sensors, processed by MicroBlaze microprocessor core on Xilinx FPGA DDR4 board. The LCD screen can display user interface for the game and other game contents. And the ultrasonic sensors are used to control the sideway motion speed of the game character (A unique car driven by ?). Buttons on FPGA board are used to select options at Menus of the game. The main game concept is to let users control the unique car to dodge other cars and collect as much coins as they can on the road. Once they hit other cars or drift out of the roadside, the game is over.

1.1 Goals

- Implement ultrasonic sensors on FPGA DDR4 including writing a driver.
- Implement other peripherals like LCD screen, buttons, timers and encoders.
- Design and draw the start menu and the gaming graphic.
- Design and implement game mechanics.
- Design and draw all items graphs like car, road and coin.
- The final version of the game can show the records and be successfully run on LCD and played by using ultrasonic sensors.

2. Methodology

a. Overview of design tasks

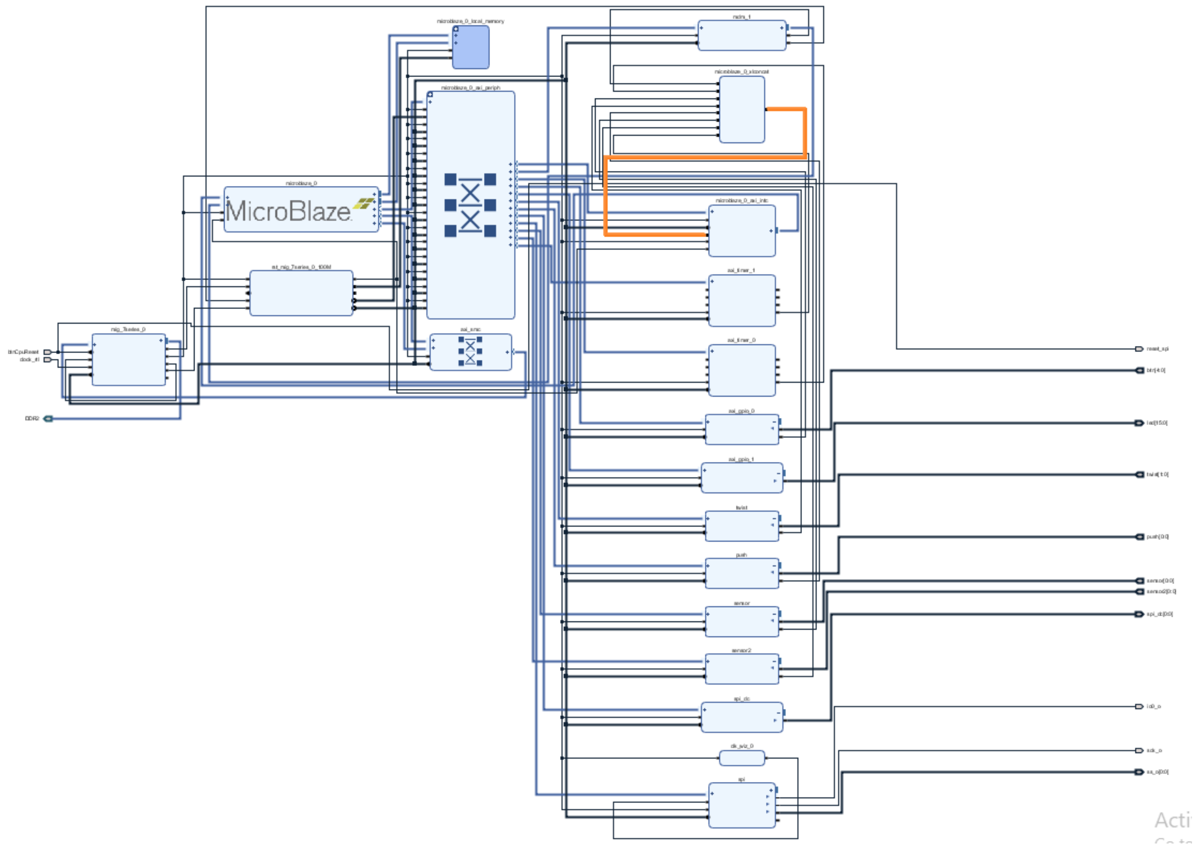


Figure 1 Overview of Block Diagrams

To support this game running, we firstly design the block diagrams that we need in Vivado to build our system environment. The picture above shows the whole view of all blocks for the system we used. Among these blocks, most of them were built in previous labs. We only did a few modifications to implement extra peripherals.

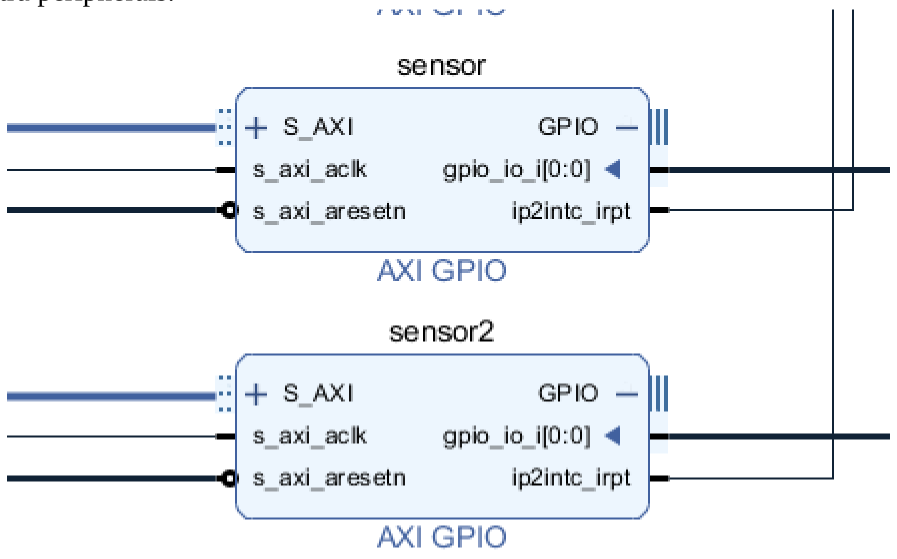


Figure 2 GPIO blocks for ultrasonic sensors

The sensor blocks are added to take in pulse width outputs of the ultrasonic sensors.

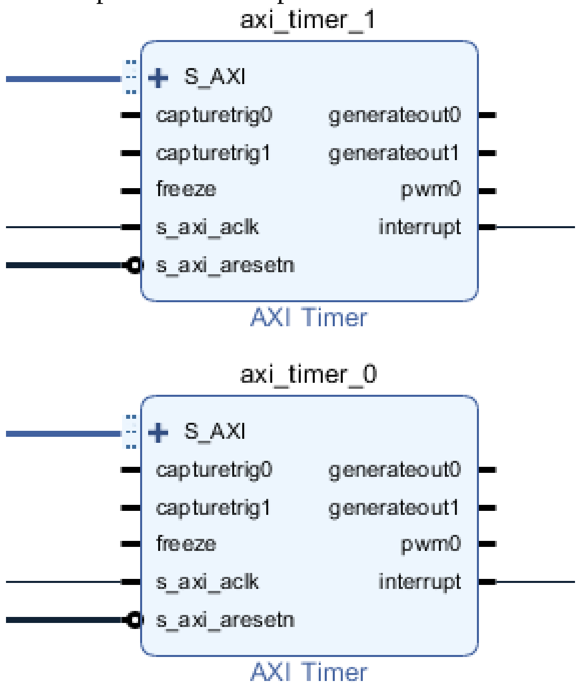
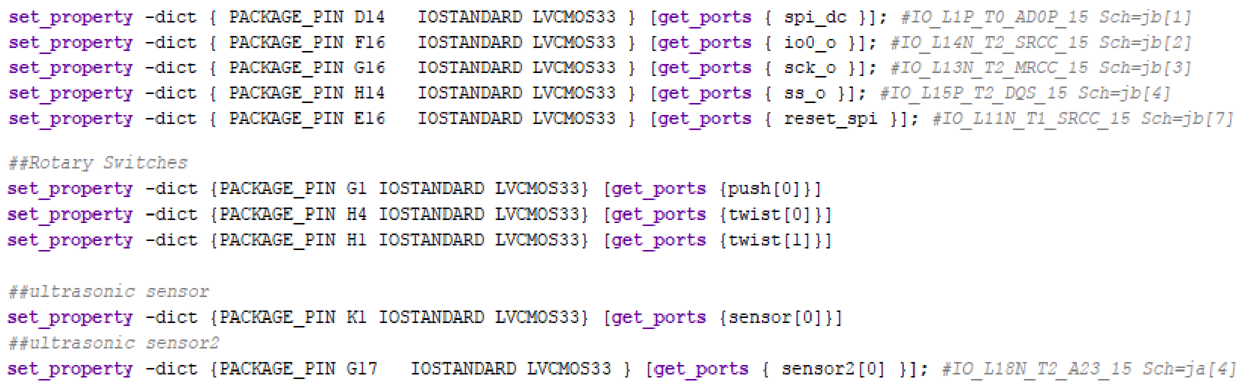


Figure 3 Timer blocks

We also added one extra timer for this project because one Axi_timer block only contains two individual counters and we need 3 to 4 counters in this project.



```
set_property -dict { PACKAGE_PIN D14   IOSTANDARD LVCMOS33 } [get_ports { spi_dc }]; #IO_L1P_T0_AD0P_15 Sch=jb[1]
set_property -dict { PACKAGE_PIN F16   IOSTANDARD LVCMOS33 } [get_ports { io0_o }]; #IO_L14N_T2_SRCC_15 Sch=jb[2]
set_property -dict { PACKAGE_PIN G16   IOSTANDARD LVCMOS33 } [get_ports { sck_o }]; #IO_L13N_T2_MRCC_15 Sch=jb[3]
set_property -dict { PACKAGE_PIN H14   IOSTANDARD LVCMOS33 } [get_ports { ss_o }]; #IO_L15P_T2_DQS_15 Sch=jb[4]
set_property -dict { PACKAGE_PIN E16   IOSTANDARD LVCMOS33 } [get_ports { reset_spi }]; #IO_L11N_T1_SRCC_15 Sch=jb[7]

##Rotary Switches
set_property -dict {PACKAGE_PIN G1 IOSTANDARD LVCMOS33} [get_ports {push[0]}]
set_property -dict {PACKAGE_PIN H4 IOSTANDARD LVCMOS33} [get_ports {twist[0]}]
set_property -dict {PACKAGE_PIN H1 IOSTANDARD LVCMOS33} [get_ports {twist[1]}]

##ultrasonic sensor
set_property -dict {PACKAGE_PIN K1 IOSTANDARD LVCMOS33} [get_ports {sensor[0]}]
##ultrasonic sensor2
set_property -dict {PACKAGE_PIN G17   IOSTANDARD LVCMOS33 } [get_ports { sensor2[0] }]; #IO_L18N_T2_A23_15 Sch=ja[4]
```

Figure 4 Ports Assignment for Peripherals

Since we have assigned our LCD output to be at port JB and rotary encoder to be at port JD, we need assigned our two ultrasonic sensors at ports JA and JC. Once we finish building our block diagrams, we generated bitstream and exported the hardware to SDK. Then we could start our code design.

For code design, we split the work into 3 major tasks. The first is to implement drivers for all peripherals and interrupt controller. The second task would be design game menu page and record page and all item graphs including characters, roads and coins. Also in the second task, color choices might not be aesthetic at first place, but they would be easy to change later in the progress. The final part is to implement the game construability and game mechanics including button explanation.

b. Assumptions required for design analysis and procedures

The ultrasonic sensors are matched with the descriptions on their datasheets. The accuracy of pulse width measurement can be 1 uS/mm. The measurement cycle is 100 mS. The range of pulse width output is 300uS for 300-mm to 5000uS for 5000-mm. Pulse width output is +/- 1% of the serial data sent. The sensors operate on voltages from 2.5V - 5.5V DC.
The interrupts priority would not affect the game play or the game control. For examples, during game play, the two ultrasonic sensors are generating interrupts to count the time for distance measurements in a really short time period (from 300 uS to 5000 uS).
The LCD screen is capable to support multiple drawing tasks at the same time. Foe example, the coin drawing is a continue task because coin is moving all the time and we don’t want it to be lagged by other drawing tasks.

c. Observations from the design tasks

At very first part of this project, we chose to add one peripheral at a time and do related coding for that peripheral. This workflow was really not efficient because every time we changed the diagram block, the vivado need do re-synthesize and re-generate the bitstream and this process often took half an hour to finish. So once we spotted this problem, we added all IP blocks at once and made sure the hardware got everything this project needs. Then we focused on coding for the rest of the time.
During the time that we wrote codes for this project, we encountered the BRAM capacity exceeded issue. After we searched this issue on internet and tried a few methods, we found out the way we implemented our code was a major reason for that issue. The reason is that we put a lot of if-else statements and nearly all codes into the main method which had a great cost of the BRAM resources. After we re-write our

codes in to several methods and reduced some redundant code and if-else statement, the issue has never occurred again.

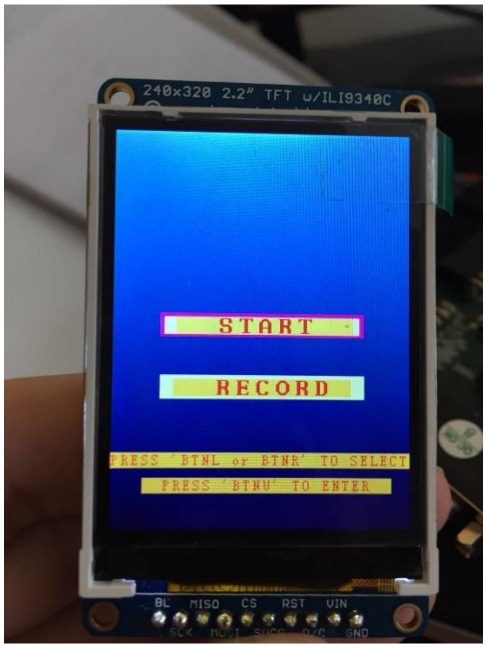
d. Plan for design testing

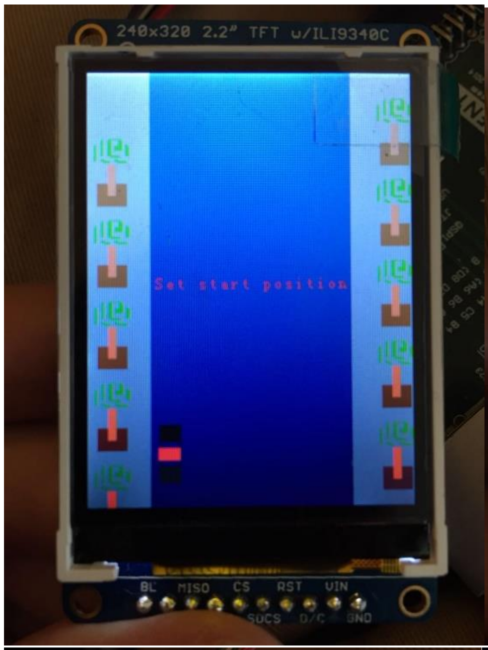
Our plan for testing our new peripherals which are two ultrasonic sensors was to use a multimeter to test the analog voltage output of those sensors that were powered by our FPGA board. If the readings are matched with their datasheets, then we could proof that they were working properly.

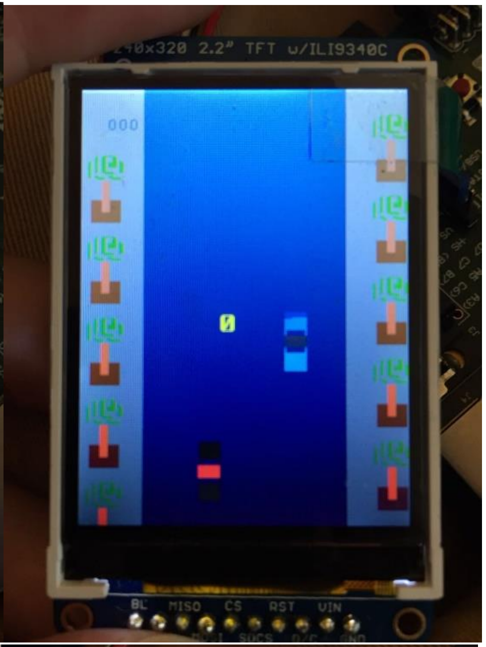
The plan for testing our game is to set specific value for game objects like the coordinates of speeds of our cars and coins. Then we could just run the game to see if it went like how we designed. Also for control of the main character, we firstly tested it’s movement by using rotary encoder rather than ultrasonic sensors to see the real movement on the screen. Then we used the sensors and our hands to control it so that we could feel the smoothness of car speed control. For item graphs and colors, we tested them directly on the LCD screen to see if they have the shapes or colors we want.

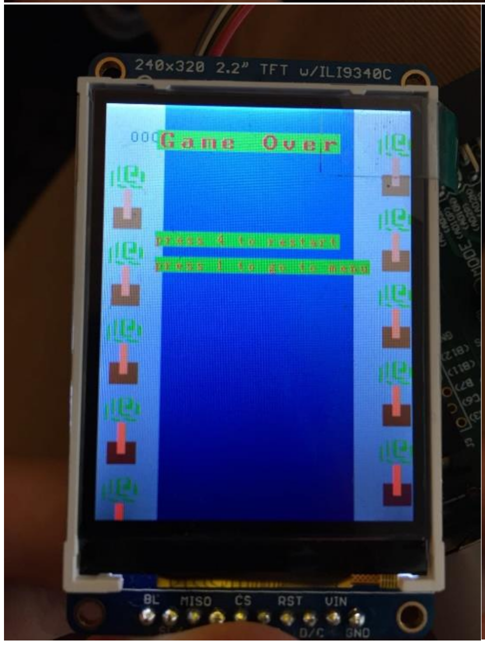
3. Results

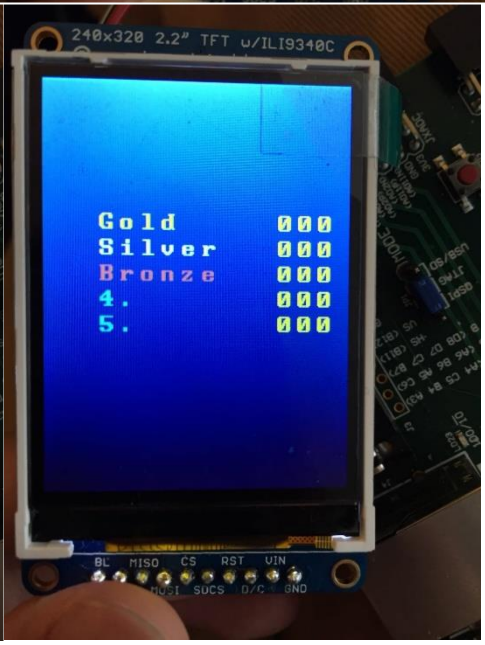
3.1 Metric Results











The expected result of the project is a balance control system controlled by two sonar sensors. In the experiment, we successfully implement the system with a LCD and two sonar sensors. We design a game based on the system. The game is a car racing game where the player controls the moving direction of the car with two sensors, avoiding the crush with other cars on the road and get coins on the road. The more coin the car gets the more score the player will get. To implement the game will also use the buttons on the board to control the user interface, user can check ranking records and start the game on the menu page. To play the game the player has to put his hands above both of the sensors, the moving speed of the car is decided by the relative distance of two hands to the sensors. The player has to keep the car on the road if the car reach the road side, the game is failed. The top five scores are recorded. The location of every coin and other car on the road is generated randomly.

3.2 limitation of the design

There are several limitations of the design: the first limitation is that the player has to put hands 30 centimeters above the sensor to control the game since the sensor will generate a pulse whose width represent the distance of the object reflects the ultrasonic wave, the scale factor of the width is $1*10^{-6}$ s per mm. The least width of the pulse is $30*10^{-6}$ s which means that object which locate less than 30cm above the sensor will also generate a pulse with $30*10^{-6}$ s. To calculate the relative distance of two hands, both hands should locate more than 30cm above the sensor, otherwise the relative distance is always zero. The detect range of the sensor is up to 5 meters above, but in practice, we can only reach less than 1 meter above with our hands. So we cannot fully use the sensor.

The second limitation of the design is that then difficulty of the game is fixed. In our expectation, the difficulty of the game can be modified depends on the score the player got. The higher score the player gets, the more difficult the game will be. We planned to add more obstacles on the road to increase difficulty of the game, but we find the LCD will be vague if there are too many pixels updated once, therefore, the make the display smooth, we only put one obstacle on the screen. And the difficulty of the game is thus fixed.

The third limitation of the design is that the ranking records can only record the score since the game is launched on the board. As long as the board is turned off, the records will lose. Because we have no sim card on the board, so we can not store data on the board when the board is out of power.

3.3 Design Test Result

To test the crush of the car controlled by the player and other cars on the road, we set the location of the other car as the same location of the car controlled by the player to check whether the bump will cause the game fail. Also, to check the record function, we set the location of the coin same as the location of the car and we speed up the coin to minimize the time for the score to reach more than 10. Because in the test we found some case where the record printed on the screen appears as signs other than number. Finally, we found that this is because we did not add the offset of the number to the char, we just transfer the number to a char, therefore the display is incorrect. To find an appropriate scale factor between the speed of the car and relative distance from the sensor. We modify the coefficient of the code and try to control the car, if the factor is too large, a slightly moving will cause the car moving fast on the screen and is difficult to stop the car before it bump in to the road side. On the contrary, if the factor is small, it is difficult to avoid other cars on the road. Finally, we find a median which is 0.2 pixel per second per cm. if the relative distance is 5 cm, the car will move at the speed of 1 pixel per second. We found it is easier to control the car.

3.4 Road Blocks

We planned to use timer interrupt to time the senor pulse, but we found it take too much time to handle interrupt that the code will miss many pulse generated by the sensor and therefore there will be a delay between the action of the player and the move of the car. To solve the problem, we turn to use the hardware timer to count for the pulse. In this way we can control the car frequently because it doesn't take time to handle the interrupt any more. Because we have two sensors and they generate pulse at different time, so we need two different timers to feedback the pulse width. The next step is draw the car on the screen, the position of the car is decided by the pulse width of the sensor, we update the position periodically by add a variable to the x position of the car. The variable equals to the speed of the car times the periodic time.

The second road block is that the LCD we used in the lab needs too much time to refresh itself, so we cannot add too many moving items on the screed, otherwise the screen will be vague, for example if we make the tree on the background moving at the opposite direction of the car, the car movement will become discontinuous because when the screen begin to update the tree on the screen, the position of the car may also be changed, but the screen can only clear and redraw one item once, so the position of the car will not change, and when the position of the car changed again the LCD can update the car on the screen, but since there is a position not updated, the car will seems like teleporting from one position to another position. We have to reduce the moving item on the screen and reduce the size of moving item to avoid this problem. So, we cannot modify the difficulty of the game by adding more barrier car on the road.
Another problem we met during coding is that the code size is too large that the BRAM of the board is full, and we can not add any other code. We planned to increase the BRAM of the board in the hardware design, but to increase the BRAM size we must add a BRAM block in the hardware design, and it may cause some other change in the code, so we decide to modify our code to reduce the code size. We made some functions for the code which we used several times and call the function instead of use a large chunk of code repeatedly, therefore the code size is reduced. And by modifying the code into functions the code is easier to read and change.
4.5 Issue and error

The first issue is that when we use timer interrupt to time the pulse width, since the granularity of the sensor is $1*10^{-6}$ s, if the timer interrupts the code every $1*10^{-6}$ s, it appears like the code will be locked in the timer interrupt handler because every time the interrupt is handled, the next interrupt is coming. And if we increase the granularity of the timer, the sensitivity of the sensor will decrease, the player has to rise his hands Exaggeratedly to move the car on the screen.
The second issue we met is when we add only one sensor in the code, the sensor can feedback the pulse width correctly, but when we add another sensor to the code, two sensors feedback the same pulse width, we find that it is because we use only one timer to count for two different sensor, and the feedback are the same, so we add another sensor for the second sensor to make two sensors work separately, so we can get two feedback from two sensor.

4. Result

4.1 Metric Results

The expected result of the project is a balance control system controlled by two sonar sensors. In the experiment, we successfully implement the system with a LCD and two sonar sensors. We design a game based on the system. The game is a car racing game where the player controls the moving direction of the car with two sensors, avoiding the crush with other cars on the road and get coins on the road. The more coin the car gets the more score the player will get. To implement the game will also use the buttons on the board to control the user interface, user can check ranking records and start the game on the menu page. To play the game the player has to put his hands above both of the sensors, the moving speed of the car is decided by the relative distance of two hands to the sensors. The player has to keep the car on the road if the car reach the road side, the game is failed. The top five scores are recorded. The location of every coin and other car on the road is generated randomly.

4.2 limitation of the design

There are several limitations of the design: the first limitation is that the player has to put hands 30 centimeters above the sensor to control the game since the sensor will generate a pulse whose width represent the distance of the object reflects the ultrasonic wave, the scale factor of the width is 1*10-6 s per mm. The least width of the pulse is 30*10-6 s which means that object which locate less than 30cm above the sensor will also generate a pulse with 30*10-6 s. To calculate the relative distance of two hands, both hands should locate more than 30cm above the sensor, otherwise the relative distance is always zero. The detect range of the sensor is up to 5 meters above, but in practice, we can only reach less than 1 meter above with our hands. So we cannot fully use the sensor.

The second limitation of the design is that then difficulty of the game is fixed. In our expectation, the difficulty of the game can be modified depends on the score the player got. The higher score the player gets, the more difficult the game will be. We planned to add more obstacles on the road to increase difficulty of the game, but we find the LCD will be vague if there are too many pixels updated once, therefore, the make the display smooth, we only put one obstacle on the screen. And the difficulty of the game is thus fixed.

The third limitation of the design is that the ranking records can only record the score since the game is launched on the board. As long as the board is turned off, the records will lose. Because we have no sim card on the board, so we can not store data on the board when the board is out of power.

4.3 Design Test Result

To test the crush of the car controlled by the player and other cars on the road, we set the location of the other car as the same location of the car controlled by the player to check whether the bump will cause the game fail. Also, to check the record function, we set the location of the coin same as the location of the car and we speed up the coin to minimize the time for the score to reach more than 10. Because in the test we found some case where the record printed on the screen appears as signs other than number. Finally, we found that this is because we did not add the offset of the number to the char, we just transfer the number to a char, therefore the display is incorrect. To find an appropriate scale factor between the speed of the car and relative distance from the sensor. We modify the coefficient of the code and try to control the car, if the factor is too large, a slightly moving will cause the car moving fast on the screen and is difficult to stop the car before it bump in to the road side. On the contrary, if the factor is small, it is difficult to avoid other cars on the road. Finally, we find a median which is 0.2 pixel per second per cm. if the relative distance is 5 cm, the car will move at the speed of 1 pixel per second. We found it is easier to control the car.

4.4 Road Blocks

We planned to use timer interrupt to time the senor pulse, but we found it take too much time to handle interrupt that the code will miss many pulse generated by the sensor and therefore there will be a delay between the action of the player and the move of the car. To solve the problem, we turn to use the hardware timer to count for the pulse. In this way we can control the car frequently because it doesn't take time to handle the interrupt any more. Because we have two sensors and they generate pulse at different time, so we need two different timers to feedback the pulse width. The next step is draw the car on the screen, the position of the car is decided by the pulse width of the sensor, we update the position periodically by add a variable to the x position of the car. The variable equals to the speed of the car times the periodic time.

The second road block is that the LCD we used in the lab needs too much time to refresh itself, so we cannot add too many moving items on the screed, otherwise the screen will be vague, for example if we make the tree on the background moving at the opposite direction of the car, the car movement will become discontinuous because when the screen begin to update the tree on the screen, the position of the car may also be changed, but the screen can only clear and redraw one item once, so the position of the car will not change, and when the position of the car changed again the LCD can update the car on the screen, but since there is a position not updated, the car will seems like teleporting from one position to another position. We have to reduce the moving item on the screen and reduce the size of moving item to avoid this problem. So, we cannot modify the difficulty of the game by adding more barrier car on the road.

Another problem we met during coding is that the code size is too large that the BRAM of the board is full, and we can not add any other code. We planned to increase the BRAM of the board in the hardware design, but to increase the BRAM size we must add a BRAM block in the hardware design, and it may cause some other change in the code, so we decide to modify our code to reduce the code size. We made some functions for the code which we used several times and call the function instead of use a large chunk of code repeatedly, therefore the code size is reduced. And by modifying the code into functions the code is easier to read and change.

4.5 Issue and error

The first issue is that when we use timer interrupt to time the pulse width, since the granularity of the sensor is 1*10-6 s, if the timer interrupts the code every 1*10-6 s, it appears like the code will be locked in the timer interrupt handler because every time the interrupt is handled, the next interrupt is coming. And if we increase the granularity of the timer, the sensitivity of the sensor will decrease, the player has to rise his hands Exaggeratedly to move the car on the screen.

The second issue we met is when we add only one sensor in the code, the sensor can feedback the pulse width correctly, but when we add another sensor to the code, two sensors feedback the same pulse width, we find that it is because we use only one timer to count for two different sensor, and the feedback are the same, so we add another sensor for the second sensor to make two sensors work separately, so we can get two feedback from two sensor.