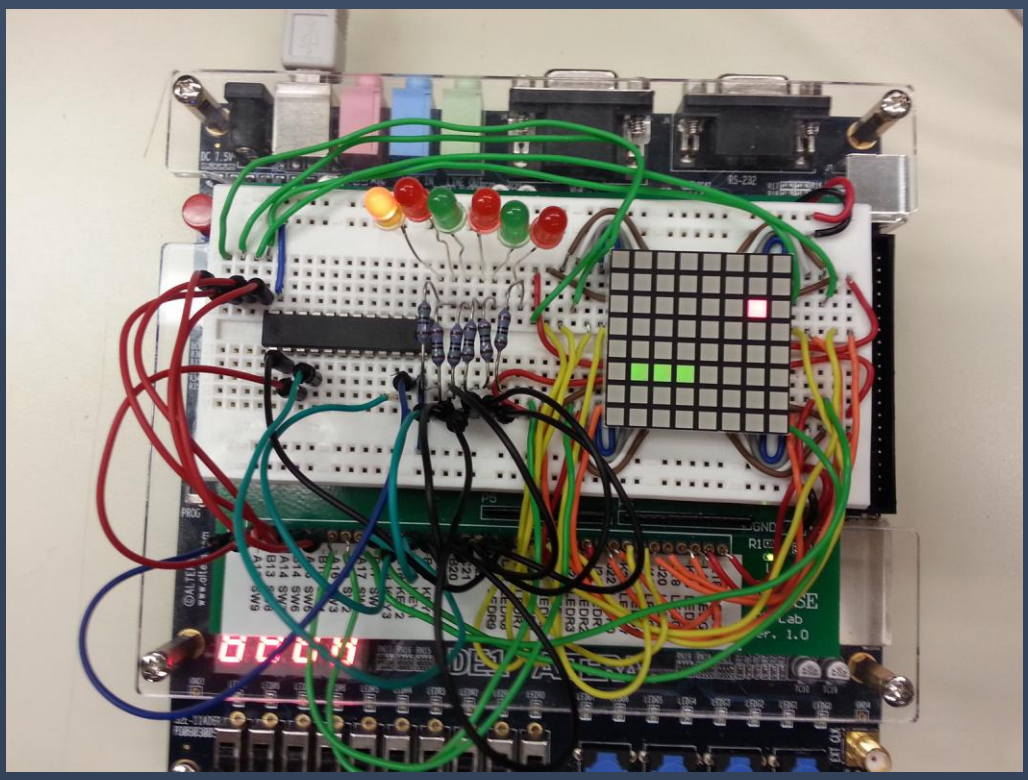
FALL
2012

EE 271 Final Project Lab Report



Snake Game

HAIYU SHI

|ALL MEMBERS CONTRIBUTED AN EQUAL AMOUNT TO ALL ASPECTS OF THE LAB
|APPROXIMATE HOURS SPENT ON EACH SECTION: DESIGN 30, TEST/DEBUG 40, DOCUMENTATION 7
|UNIVERSITY OF WASHINGTON

Table of Contents

ABSTRACT

Our implementation of the game ‘Snake’ was designed to emulate the same main rules of the original game. Additional design requirements were specified in the laboratory specification document. We designed the snake as a finite state machine. We also used a linear finite shift registers to make a random number generator for the position of the food pixel. Our design was implemented on the Altera DE1 board, and 8X8 LED matrix and a GAL22V10 gate array chip. The game design was confirmed to operate correctly through extensive testing in Verilog and on the DE1 board.

INTRODUCTION

The purpose of this lab was to design, develop, and implement a simple game. Our game was based on a finite state machine for the snake controls, counter on a GAL22V10 chip and counters on the DE1 board. We incorporated an 8X8 LED matrix for our display. The design also includes a mechanism for generating random food positions on the LED display matrix as well as shift registers for keeping track of the snake body pixels. We completed the project in small individual stages but tested for proper operation as a whole. One tool that made debugging easier was the high/low probe. It allowed us to debug errors in our LED matrix rows and columns.

DESIGN SPECIFICATION

Our implementation of the game Snake uses most of the rules of the original Snake game. The ‘leader’ pixel for the body continuously moves to one adjacent square at a time in one direction until the user presses the ‘right’ or ‘left’ button. Pressing ‘right’ or ‘left’ will cause the snake to change directions relative the direction it is moving. When the leader collides with the food, the snake adds one additional segment and the food is generated in a random location. The game ends when the snake collides with itself. The Snake needs to be able to turn left or right at any point, and it needs to grow when it 'eats' food. In addition, an 8x8 LED matrix with 128 LED's in a common cathode configuration is to be used for the game display. Additional requirements were set by the EE 271 laboratory 4 specification sheet. The project must be implemented using the DE1 board, must incorporate at least one finite state machine module utilizing at least one GAL 22V10, and must implement at least one finite state machine module for the Cyclone II FPGA using schematic capture.

DESIGN PROCEDURE

The overall top-level design of our game is shown in Figure 0.

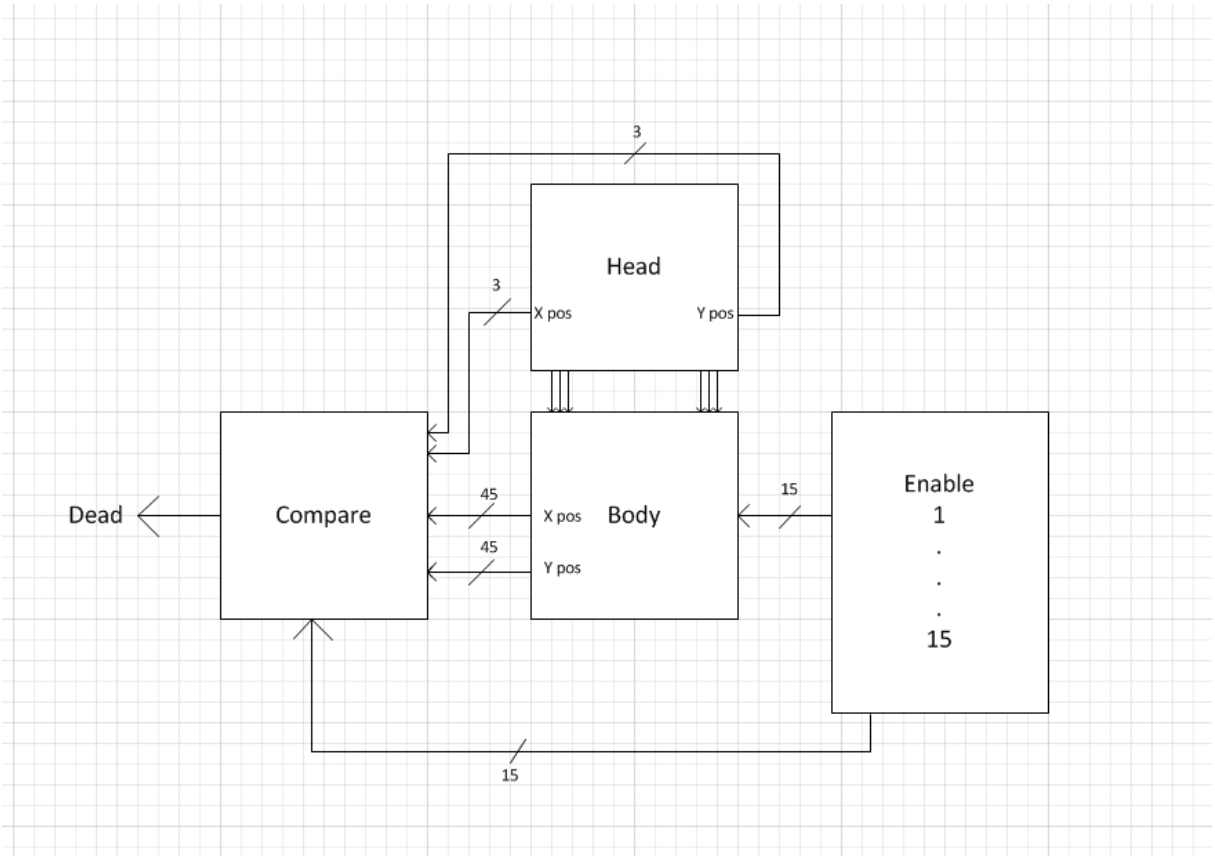


Figure 1
Top-Level Desgin

The foundation for Snake and the first element of the game we built was the Finite State Machine, a simple logic machine which depends on the inputs to reach certain states and then output data, to control the movement of the head of the snake. As shown below in Figure 1, it operates in one of five states, and it remains there unless an input key is pressed. These states then output to 2 up/down counters which act to remember the position of the snake. Since our LED matrix was 8x8, we decided to use a coordinate system based off of cartesian coordinates but with binary. Due to the small space of the matrix however, rather than the edges of the display being walls, the snake instead wraps around to the opposite side. This is implemented by having two counters, one storing x-position and one storing y-position which count up from 000 to 111 and then from 111 it returns back to 000. They both have enables, as well as a flag which indicates whether or not to count up or down and these are output from the finite state machine of the snake.

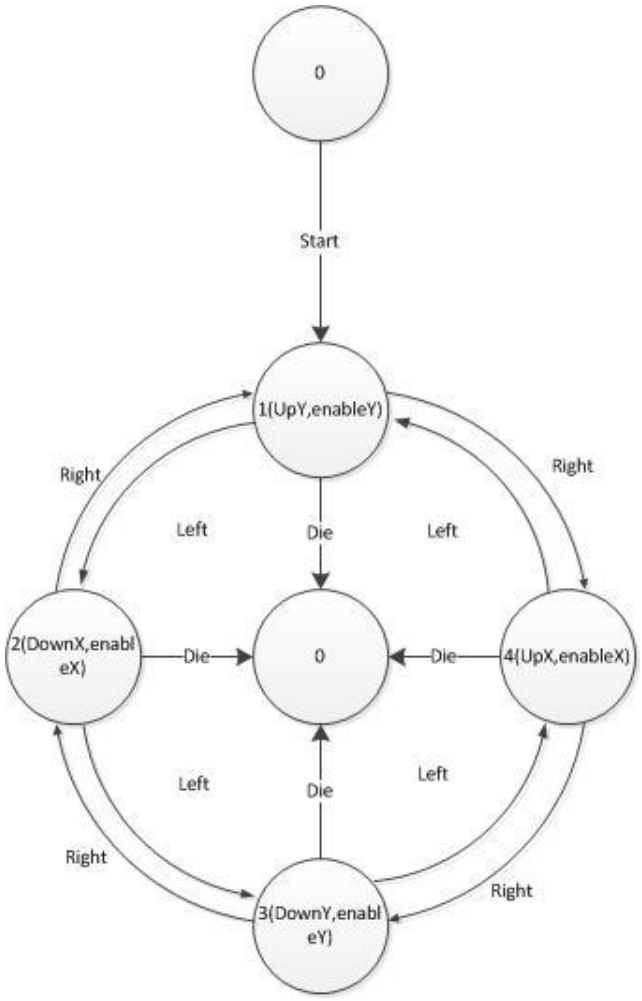
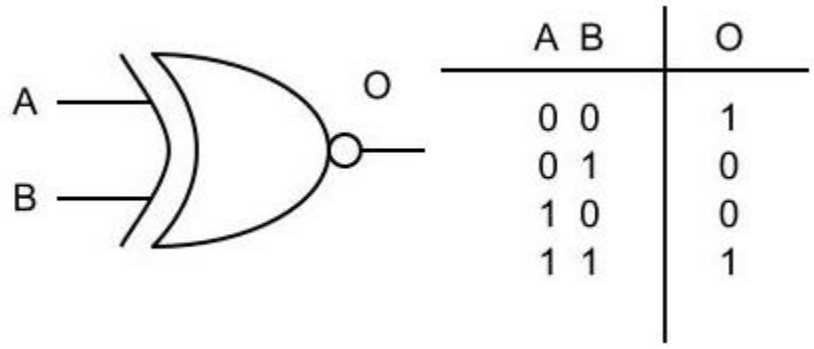


Figure 2
State Diagram of Snake Movement

After that we designed and built the mechanism for displaying food in a random position on the board, so generating a random number of the for a coordinate of the form (100, 010) where x is represented by the first three bits and y is the last three, allowing for 64 total possible combinations. To do this we used a slightly modified LFSR, or a linear feedback shift register run on a very fast clock to generate pseudo random numbers, then when either the start key is pressed or the eat output goes high, a register comprised of D flip flops is loaded with the 6 'random' bits which are held and output to the display until start or eat goes high again. Then, the next crucial step in building the system was the logic required for collisions. For a collision to happen, there must be either the head and food or the head and the body, two separate objects at the same coordinates. This means that every coordinate bit of one matches every corresponding coordinate bit in the order. After a realization that the XNOR gate returns a 0 when the two input

bits are different and a 1 when they match as shown below.



A	B	O
0	0	1
0	1	0
1	0	0
1	1	1

Figure 3
Xnor Gate

Utilizing the power of the XNOR gate, use it in multiple modules as the basis for calculating collision. In one case it's to compare food and the head, and in another it's used to compare the head and the body's segments.

The growth of the body is controlled by cascading registers with individual enables. The first register has 6 flip-flops which act to output a position bit passed in 1 time unit previously, so as a collective the flip flops create a 6 bit signal delay. These registers were then attached with the first register receiving input directly from the head's x and y positions, and then it's output attached to the input of the next and on down the chain. Each register only reflects the input in its output when enable is active.

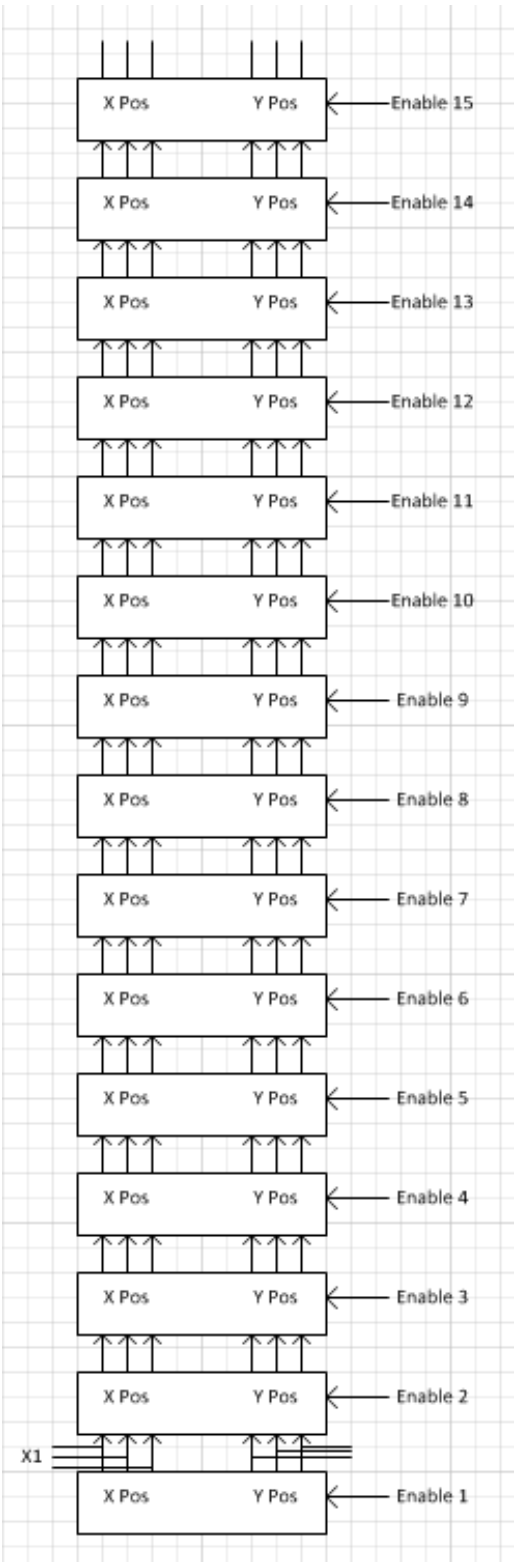


Figure 4
Cascading Parallel Input Parallel Output Registers

Due to the size of the LED display matrix, we set the maximum number of segments the snake

could add at 15 pixels. Then we designed a counter on the GAL chip which was clocked off of the eat signal, so that every time food is eaten, the 4 bit counter increments one, until it reaches 1111, where it holds. This counter then controls the logic which allowed each register to be enabled. The enables also worked to control when a piece of the snake 'exists' or when they should be displayed and possible to collide with itself.

The next major step was getting all the data properly displayed. Referring to the LED matrix schematic shown in Figure 5, in order to light up an LED means that one of the columns of anodes needs to be powered, while the corresponding row of cathodes needs to be grounded.

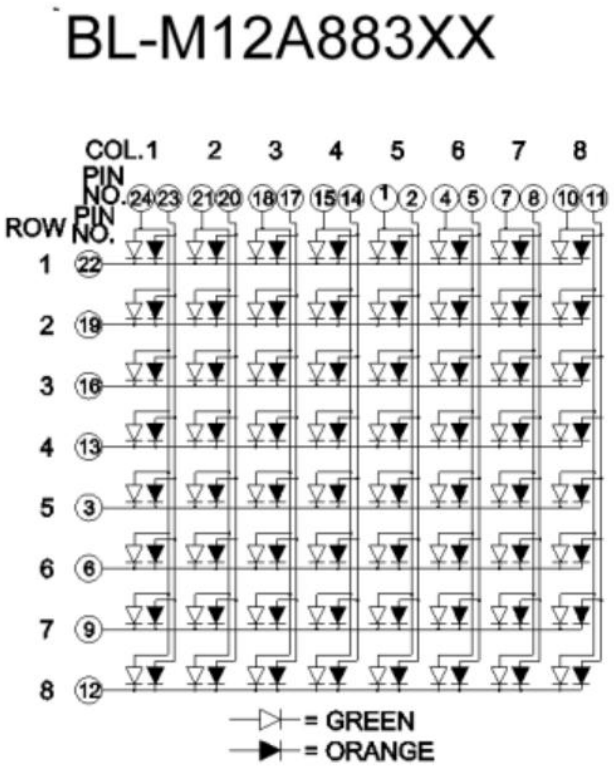


Figure 5
8x8 Bi-color LED Matrix Schematic

All the LED's in one or multiple columns of the matrix can be lit up by grounding all the cathodes and putting a voltage to the anode, while 1 row can be lit up by grounding one cathode and powering all the anodes. This creates a wide possibility of options for displaying, but it's impossible to synchronously light up two LED's in different rows and columns since you're grounding 2 cathodes and powering 2 anodes, thus leading to 4 LED's being lit. So, time has to be manipulated so as to allow the proper displaying. We used an 8 bit shift register loaded with one 0, and seven 1s such that as the shift register passes the values from one flip flop to the next, only one cathode is ever grounded at a time.

Next, we built a counter clocked off the same clock as the shift register with the intent that it

should keep track of which cathode is being grounded. This is just a 3 bit counter, but the beauty of it is that the counter's state exactly matches the x-coordinate, which means that if something needs to be displayed in that row, the anodes can be powered just while the counter's state matches and so the exact desired points can be lit up. This however brings up a separate problem of lighting up multiple LEDS in one column corresponding to a single cathode. We solved it by converting the 3 bit y-position into an 8 bit code where each bit represents an anode, then combining the 8 bit codes for every displayed point with the same x-coordinate (require the same cathode to be grounded). This made it possible for the entire snake game to be displayed by 16 different 8 bit vectors. 8 are for the food position and the red anodes, and contain very little information, while 8 are for the green anodes and the snake's body. These 16 are activated 1 green and 1 red corresponding to the state of the strobe counter and they are constantly cycling through. This selection of vector to be displayed was done by an 8-to-1 mux created with the Quartus schematic capture software.

HARDWARE IMPLEMENTATION

Our hardware implementation is shown in the figures below:

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength
Food1	Output	PIN_E21	5	B5_N0	PIN_E21	3.3-V LV...default)		24mA (default)
Food2	Output	PIN_E22	5	B5_N0	PIN_E22	3.3-V LV...default)		24mA (default)
Food3	Output	PIN_F21	5	B5_N0	PIN_F21	3.3-V LV...default)		24mA (default)
Food4	Output	PIN_F22	5	B5_N0	PIN_F22	3.3-V LV...default)		24mA (default)
Food5	Output	PIN_G21	5	B5_N1	PIN_G21	3.3-V LV...default)		24mA (default)
Food6	Output	PIN_G22	5	B5_N1	PIN_G22	3.3-V LV...default)		24mA (default)
Food7	Output	PIN_J21	5	B5_N1	PIN_J21	3.3-V LV...default)		24mA (default)
Food8	Output	PIN_J22	5	B5_N1	PIN_J22	3.3-V LV...default)		24mA (default)
Xout1	Output	PIN_B17	4	B4_N1	PIN_B17	3.3-V LV...default)		24mA (default)
Xout2	Output	PIN_A17	4	B4_N1	PIN_A17	3.3-V LV...default)		24mA (default)
Xout3	Output	PIN_B16	4	B4_N1	PIN_B16	3.3-V LV...default)		24mA (default)
Xout4	Output	PIN_A16	4	B4_N1	PIN_A16	3.3-V LV...default)		24mA (default)
Xout5	Output	PIN_B15	4	B4_N1	PIN_B15	3.3-V LV...default)		24mA (default)
Xout6	Output	PIN_A15	4	B4_N1	PIN_A15	3.3-V LV...default)		24mA (default)
Xout7	Output	PIN_B14	4	B4_N1	PIN_B14	3.3-V LV...default)		24mA (default)
Xout8	Output	PIN_A14	4	B4_N1	PIN_A14	3.3-V LV...default)		24mA (default)
Y1	Output	PIN_K21	5	B5_N1	PIN_K21	3.3-V LV...default)		24mA (default)
Y2	Output	PIN_K22	5	B5_N1	PIN_K22	3.3-V LV...default)		24mA (default)
Y3	Output	PIN_J19	5	B5_N1	PIN_J19	3.3-V LV...default)		24mA (default)
Y4	Output	PIN_J20	5	B5_N1	PIN_J20	3.3-V LV...default)		24mA (default)
Y5	Output	PIN_J18	5	B5_N1	PIN_J18	3.3-V LV...default)		24mA (default)
Y6	Output	PIN_K20	5	B5_N1	PIN_K20	3.3-V LV...default)		24mA (default)
Y7	Output	PIN_L19	5	B5_N1	PIN_L19	3.3-V LV...default)		24mA (default)
Y8	Output	PIN_L18	5	B5_N1	PIN_L18	3.3-V LV...default)		24mA (default)
clk	Input	PIN_E12	3	B3_N0	PIN_E12	3.3-V LV...default)		24mA (default)
fxa	Output	PIN_B19	4	B4_N0	PIN_B19	3.3-V LV...default)		24mA (default)
fxb	Output	PIN_A20	4	B4_N0	PIN_A20	3.3-V LV...default)		24mA (default)
fxc	Output	PIN_B20	4	B4_N0	PIN_B20	3.3-V LV...default)		24mA (default)
fya	Output	PIN_C21	5	B5_N0	PIN_C21	3.3-V LV...default)		24mA (default)
fyb	Output	PIN_C22	5	B5_N0	PIN_C22	3.3-V LV...default)		24mA (default)
fyc	Output	PIN_D21	5	B5_N0	PIN_D21	3.3-V LV...default)		24mA (default)
galClock	Output	PIN_B13	4	B4_N1	PIN_B13	3.3-V LV...default)		24mA (default)
inL	Input	PIN_T22	6	B6_N0	PIN_T22	3.3-V LV...default)		24mA (default)
inR	Input	PIN_T21	6	B6_N0	PIN_T21	3.3-V LV...default)		24mA (default)
inStart	Input	PIN_R22	6	B6_N0	PIN_R22	3.3-V LV...default)		24mA (default)
rst	Input	PIN_L2	2	B2_N1	PIN_L2	3.3-V LV...default)		24mA (default)
start	Output	PIN_A13	4	B4_N1	PIN_A13	3.3-V LV...default)		24mA (default)

Figure 6
Pin Assignments on DE1 Board

Referring back to Figure 5, we attached the pins Labeled Xout 1-8 off the DE1 board to the cathodes 22, 19, 16, 13, 12, 9, 6, 3 on the LED matrix. The outputs labeled Food go to the red rows 1-8, or pins 23, 20, 17, 14, 2, 5, 8, 11 on the LED matrix and the Y outputs went to the green anodes at 24, 21, 18, 15, 1, 4, 7, 10. GalClock the opposite of the signal eat, which acts to clock the food counter on the negative edge. In addition, we used the keys as inputs and they are active low, meaning we had to invert their signals on the board.

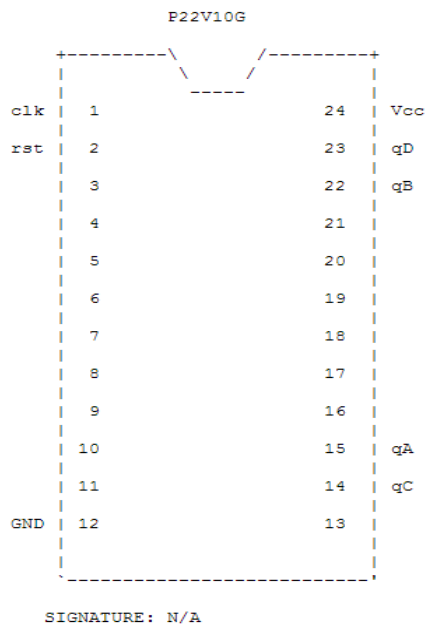


Figure 7
Gal Chip Diagram

TEST PLAN

In this project, our game is comprised of 17 modules. To test each module separately and then debug each one at a time would be inefficient, therefore, we test our design as a combined file. One advantage of testing the entire system is that all of the modules connect to another module, allowing us to reduce the number of test inputs. In our Verilog tester, we only have 5 inputs, which are reset, clock, start, left, and right. This makes the test more realistic, it could also ease the debugging process. After we passed the entire test in the software, we would wire everything up on the DE1 board and actually test it out with our 8X8 LED matrix and play the game.

Based on the rule of the game and our design logic, we list all the test cases we need to fit:

- The start should start the game, the snake should start moving and the food position should appear in a random position in the screen.
- When the user press left or right, the snake should turn left or right in its perspective correctly.
- When the snake hit itself, it should die and stop moving and the body will disappear.
- When the snake eats the food, the food would be respawned in a new random position.
- When the snake eats one food, the body should be extended one more.
- When the snake hit the wall, it should appear from the opposite edge of the screen and keep its original position when it hit the wall.

All of the conditions above need to be met both in the hardware test and in Verilog.
In the Verilog code, we would be using the following test case

```
#stimDelay rst = 0;
            #periodstart=1;
            #period start=0;
 #period R = 1;
            #period R = 0;
            #period L = 1;
            #period L = 0;  // (multiple random R and L combinations)
```

And in the hardware implementation, the test would can simply be run by playing the game and testing the same cases mentioned above. Our expectation for our game would be that it meets all the test cases both in Verilog and in the real hardware implementation of the game.

RESULTS

Verilog testing

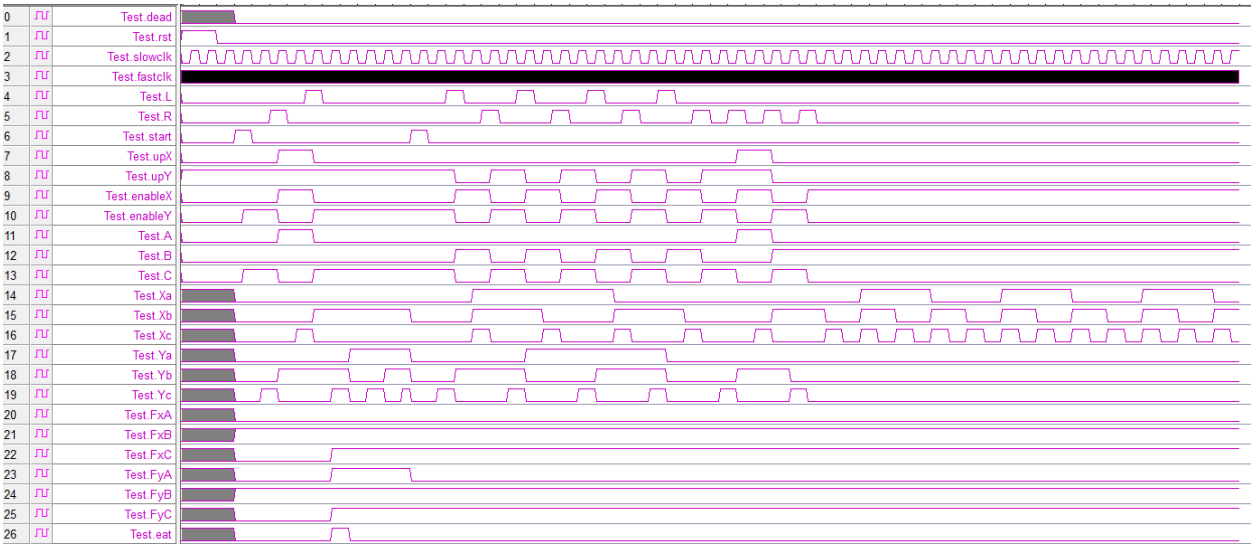


Figure 8
Simulation Waveform for the Whole System Part 1

In the part 1 test, we have almost all the most important key elements for our system, the first one is the dead, since the snake can die only when it hit itself, and our food generator is generating totally random numbers which make it almost impossible to just fit the condition that the snake would eat the food for 4 times and then kill itself, so in our simulation, the dead is always low.

The reset and two different clocks we used for our system, operate as intended and the reset controls the initial conditions of most elements in the system. Most parameters are dependent on either the slowclock or the fastclock to change their states correctly. The other two inputs, L and R, which corresponding to the left and right, operate correctly as the controls for the game. The last input is the start, and it will reset the food position and the state of the snake. These parameters in the waveform are our input for the test case.

The UpX, UpY, enableX, enableY are our states for the snake, so when the snake is in a specific state, it should have the corresponding upx,upY and enableX, enableY.

The way we determine current state of the snake based on the counting machine we used. For this one specifically, we used the A, B, and C as our state count, which A is the most significant bit and C is the LSB. And the state machine should be controlled only the start, left and right, and we can see from the waveform above, that it moves to the right state after we input the left and right.

Following is our head position, and food position, as we indicated in the design procedure, we used 6 bits to represent the position of everything that needs to be displayed in the LED. Variables Xa,Xb,Xc,Ya,Yb,Yc are the position for the head, FxA, FxB, FxC, FyA, FyB, FyC are our food position. The head always starts at 0, and the later position depends on the state of the snake, because it will determine which direction the snake will move. If the snake eats the food, the food position wouldn't change until the snake eats it, and in the waveform we can see it fit this condition since eat only happened once.

Figure 9
Simulation Waveform for the Whole System Part 2

In part 2, we have the food count, which counts up the food the food haven been eaten, and we saw only one which fit our test case, because we only make eat happened once in this test. And once it eat the food, the enable used to control the body is also turned on immediately. After it is turned on, the first body position would be on, and it will take the position passed by the head. We can see this from Xa1, Xb1, Xc1, Ya1, Yb1, and Yc1 in the test.

Figure 10
Simulation Waveform for the Whole System Part 3

In the part 3, we tested all the display logic, using the logic that has been indicated in the designing procedure. The Xouts that have been scanned over by the fast clock and the rest outputs are constantly passing signals based on the position of the snake and its body points. Overall, we went through all the connections between our inputs, variables and outputs, they all fit our expectation and the test case in the test plan, which proved our code is working perfectly as we expected and it match our design logic.

Hardware testing

The hardware testing result is much more visible and understandable, because it would be just playing the game, and try out all test cases we had list in the test plan. And the result was shown in the demotion section; also we took a video for our project for reference of the hardware testing:

http://www.youtube.com/watch?feature=player_embedded&v=hrZ98rFMMqg

SUMMARY

The implementation of the game 'Snake' we designed for this project closely matched the rules of the original Snake game. Additional design requirements were specified in the laboratory specification document. Beginning with a top-level design diagram, we identified which concepts were going to be needed. The first component that we designed was the finite state machine for the snake controls. We used linear finite shift registers to make a random number generator for the position of the food pixel. Counters for the X and Y position of the snake were used to keep track of the position. Clock dividers were used to produce a slow and fast frequency to drive different components and especially the quick scanning of the cathode rows on the LED matrix. Our design was implemented on the Altera DE1 board, and 8X8 LED matrix and a GAL22V10 gate array chip. We confirmed proper game through extensive testing of the specification checks in Verilog and on the DE1 board.

CONCLUSION

Our Snake game design was completed in separate stages and then tested as a complete system. To design the game, we used finite state machines for the snake control, linear finite shift registers for the random food position generator, counters for the X and Y position of the snake, and clock dividers to produce a slow and fast frequency. Our design was implemented on the Altera DE1 board, and 8X8 LED matrix and a GAL22V10 gate array chip By testing all of the modules together in Verilog and also on the DE1 board, we verified that our design operated correctly. One significant lesson our lab group inferred from this project was that debugging using the LEDs and a high/low indicator probe made it much easier to pin point exact problems.

APPENDIX

Verilog Code of Snake with Test Bench

```
module foodDisplay(foodDispY1,foodDispY2, foodDispY3,foodDispY4, foodDispY5,
foodDispY6, foodDispY7, foodDispY8,
 a,b,c, fy1, fy2, fy3, fy4,fy5, fy6, fy7, fy8, fxa, fxb, fxc);

        input a,b,c, fy1, fy2, fy3, fy4,fy5, fy6, fy7, fy8, fxa, fxb, fxc;

        output foodDispY1,foodDispY2, foodDispY3,foodDispY4, foodDispY5,
foodDispY6, foodDispY7, foodDispY8;

        and foodDisp1(foodDispY1, a,b,c,fxa,fxb,fxc, fy1),
        foodDisp2(foodDispY2, a,b,c,fxa,fxb,fxc, fy2),
        foodDisp3(foodDispY3, a,b,c,fxa,fxb,fxc, fy3),
        foodDisp4(foodDispY4, a,b,c,fxa,fxb,fxc, fy4),
        foodDisp5(foodDispY5, a,b,c,fxa,fxb,fxc, fy5),
        foodDisp6(foodDispY6, a,b,c,fxa,fxb,fxc, fy6),
        foodDisp7(foodDispY7, a,b,c,fxa,fxb,fxc, fy7),
        foodDisp8(foodDispY8, a,b,c,fxa,fxb,fxc, fy8);

endmodule

module foodScan(Food1, Food2, Food3, Food4, Food5, Food6, Food7,
Food8,a,b,c,fya, fyb, fyc, fxa, fxb, fxc);

        input a,b,c, fya, fyb, fyc, fxa, fxb, fxc;

        output Food1, Food2, Food3, Food4, Food5, Food6, Food7, Food8;

        convertToSerialY gogo(fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8, fya,
fyb, fyc, 1'b1);

        foodDisplay dispA(A1,A2, A3,A4, A5, A6, A7, A8,~a,~b,~c, fy1, fy2,
fy3, fy4, fy5, fy6, fy7, fy8, ~fxa, ~fxb, ~fxc);
        foodDisplay dispB(B1,B2, B3,B4, B5, B6, B7, B8,~a,~b,c, fy1, fy2, fy3,
fy4, fy5, fy6, fy7, fy8, ~fxa, ~fxb, fxc);
        foodDisplay dispC(C1,C2, C3,C4, C5, C6, C7, C8,~a,b,~c, fy1, fy2, fy3,
fy4, fy5, fy6, fy7, fy8, ~fxa, fxb, ~fxc);
        foodDisplay dispD(D1,D2, D3,D4, D5, D6, D7, D8,~a,b,c, fy1, fy2, fy3,
fy4, fy5, fy6, fy7, fy8, ~fxa, fxb, fxc);
        foodDisplay dispE(E1,E2, E3,E4, E5, E6, E7, E8,a,~b,~c, fy1, fy2, fy3,
fy4, fy5, fy6, fy7, fy8, fxa, ~fxb, ~fxc);
        foodDisplay dispF(F1,F2, F3,F4, F5, F6, F7, F8,a,~b,c, fy1, fy2, fy3,
fy4, fy5, fy6, fy7, fy8, fxa, ~fxb, fxc);
        foodDisplay dispG(G1,G2, G3,G4, G5, G6, G7, G8,a,b,~c, fy1, fy2, fy3,
fy4, fy5, fy6, fy7, fy8, fxa, fxb, ~fxc);
        foodDisplay dispH(H1,H2, H3,H4, H5, H6, H7, H8,a,b,c, fy1, fy2, fy3,
fy4, fy5, fy6, fy7, fy8, fxa, fxb, fxc);

        or foodYscan1(Food1, A1, B1, C1, D1, E1, F1, G1, H1),
                foodYscan2(Food2, A2, B2, C2, D2, E2, F2, G2, H2),
                foodYscan3(Food3, A3, B3, C3, D3, E3, F3, G3, H3),
```

```
                foodYscan4(Food4, A4, B4, C4, D4, E4, F4, G4, H4),
                foodYscan5(Food5, A5, B5, C5, D5, E5, F5, G5, H5),
                foodYscan6(Food6, A6, B6, C6, D6, E6, F6, G6, H6),
                foodYscan7(Food7, A7, B7, C7, D7, E7, F7, G7, H7),
                foodYscan8(Food8, A8, B8, C8, D8, E8, F8, G8, H8);


endmodule

//////////////////////////////////////////////////////////////////////////////
/
//////////////////////////////////////////////////////////////////////////////
/


module condenseDisplay(ay1, ay2, ay3, ay4, ay5, ay6, ay7, ay8,by1, by2, by3,
by4, by5, by6, by7, by8,cy1, cy2, cy3, cy4, cy5, cy6, cy7, cy8,
        dy1, dy2, dy3, dy4, dy5, dy6, dy7, dy8, eey1, eey2, eey3, eey4, eey5,
eey6, eey7, eey8, ffy1, ffy2, ffy3, ffy4, ffy5, ffy6, ffy7, ffy8,
ggy1, ggy2, ggy3, ggy4, ggy5, ggy6, ggy7, ggy8, hhy1, hhy2, hhy3, hhy4, hhy5,
hhy6, hhy7, hhy8, Xa,Xb,Xc,Ya,Yb,Yc,
Xa1,Xb1,Xc1,Ya1,Yb1,Yc1,Xa2,Xb2,Xc2,Ya2,Yb2,Yc2,Xa3,Xb3,Xc3,Ya3,Yb3,Yc3,

        Xa4,Xb4,Xc4,Ya4,Yb4,Yc4,Xa5,Xb5,Xc5,Ya5,Yb5,Yc5,

        Xa6,Xb6,Xc6,Ya6,Yb6,Yc6,Xa7,Xb7,Xc7,Ya7,Yb7,Yc7,

        Xa8,Xb8,Xc8,Ya8,Yb8,Yc8,Xa9,Xb9,Xc9,Ya9,Yb9,Yc9,Xa10,Xb10,Xc10,Ya10,Yb
10,Yc10,

        Xa11,Xb11,Xc11,Ya11,Yb11,Yc11,Xa12,Xb12,Xc12,Ya12,Yb12,Yc12,

        Xa13,Xb13,Xc13,Ya13,Yb13,Yc13,Xa14,Xb14,Xc14,Ya14,Yb14,Yc14,

        Xa15,Xb15,Xc15,Ya15,Yb15,Yc15,enable1,enable2,enable3,enable4,

        enable5,enable6,enable7,enable8,enable9,enable10,enable11,enable12,
                                    enable13,enable14,enable15, dead);

        input Xa,Xb,Xc,Ya,Yb,Yc,
Xa1,Xb1,Xc1,Ya1,Yb1,Yc1,Xa2,Xb2,Xc2,Ya2,Yb2,Yc2,Xa3,Xb3,Xc3,Ya3,Yb3,Yc3,

        Xa4,Xb4,Xc4,Ya4,Yb4,Yc4,Xa5,Xb5,Xc5,Ya5,Yb5,Yc5,

        Xa6,Xb6,Xc6,Ya6,Yb6,Yc6,Xa7,Xb7,Xc7,Ya7,Yb7,Yc7,

        Xa8,Xb8,Xc8,Ya8,Yb8,Yc8,Xa9,Xb9,Xc9,Ya9,Yb9,Yc9,Xa10,Xb10,Xc10,Ya10,Yb
10,Yc10,

        Xa11,Xb11,Xc11,Ya11,Yb11,Yc11,Xa12,Xb12,Xc12,Ya12,Yb12,Yc12,

        Xa13,Xb13,Xc13,Ya13,Yb13,Yc13,Xa14,Xb14,Xc14,Ya14,Yb14,Yc14,

        Xa15,Xb15,Xc15,Ya15,Yb15,Yc15,enable1,enable2,enable3,enable4,

        enable5,enable6,enable7,enable8,enable9,enable10,enable11,enable12,
                                    enable13,enable14,enable15, dead;
```

```
        output ay1, ay2, ay3, ay4, ay5, ay6, ay7, ay8,by1, by2, by3, by4, by5,
by6, by7, by8,cy1, cy2, cy3, cy4, cy5, cy6, cy7, cy8,
        dy1, dy2, dy3, dy4, dy5, dy6, dy7, dy8, eey1, eey2, eey3, eey4, eey5,
eey6, eey7, eey8, ffy1, ffy2, ffy3, ffy4, ffy5, ffy6, ffy7, ffy8,
ggy1, ggy2, ggy3, ggy4, ggy5, ggy6, ggy7, ggy8, hhy1, hhy2, hhy3, hhy4, hhy5,
hhy6, hhy7, hhy8;

        convertToSerialX convertHeadX(ex1, ex2, ex3, ex4, ex5, ex6, ex7, ex8,
Xa, Xb, Xc, ~dead);
        convertToSerialY convertHeadY(ey1, ey2, ey3, ey4, ey5, ey6, ey7, ey8,
Ya, Yb, Yc, ~dead);

        convertToSerialX convertHeadX1(fx1, fx2, fx3, fx4, fx5, fx6, fx7, fx8,
Xa1, Xb1, Xc1, enable1);
        convertToSerialY convertHeadY1(fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8,
Ya1, Yb1, Yc1, enable1);

        convertToSerialX convertHeadX2(gx1, gx2, gx3, gx4, gx5, gx6, gx7, gx8,
Xa2, Xb2, Xc2, enable2);
        convertToSerialY convertHeadY2(gy1, gy2, gy3, gy4, gy5, gy6, gy7, gy8,
Ya2, Yb2, Yc2, enable2);

        convertToSerialX convertHeadX3(hx1, hx2, hx3, hx4, hx5, hx6, hx7, hx8,
Xa3, Xb3, Xc3, enable3);
        convertToSerialY convertHeadY3(hy1, hy2, hy3, hy4, hy5, hy6, hy7, hy8,
Ya3, Yb3, Yc3, enable3);

        convertToSerialX convertHeadX4(ix1, ix2, ix3, ix4, ix5, ix6, ix7, ix8,
Xa4, Xb4, Xc4, enable4);
        convertToSerialY convertHeadY4(iy1, iy2, iy3, iy4, iy5, iy6, iy7, iy8,
Ya4, Yb4, Yc4, enable4);

        convertToSerialX convertHeadX5(jx1, jx2, jx3, jx4, jx5, jx6, jx7, jx8,
Xa5, Xb5, Xc5, enable5);
        convertToSerialY convertHeadY5(jy1, jy2, jy3, jy4, jy5, jy6, jy7, jy8,
Ya5, Yb5, Yc5, enable5);

        convertToSerialX convertHeadX6(kx1, kx2, kx3, kx4, kx5, kx6, kx7, kx8,
Xa6, Xb6, Xc6, enable6);
        convertToSerialY convertHeadY6(ky1, ky2, ky3, ky4, ky5, ky6, ky7, ky8,
Ya6, Yb6, Yc6, enable6);

        convertToSerialX convertHeadX7(lx1, lx2, lx3, lx4, lx5, lx6, lx7, lx8,
Xa7, Xb7, Xc7, enable7);
        convertToSerialY convertHeadY7(ly1, ly2, ly3, ly4, ly5, ly6, ly7, ly8,
Ya7, Yb7, Yc7, enable7);

        convertToSerialX convertHeadX8(mx1, mx2, mx3, mx4, mx5, mx6, mx7, mx8,
Xa8, Xb8, Xc8, enable8);
        convertToSerialY convertHeadY8(my1, my2, my3, my4, my5, my6, my7, my8,
Ya8, Yb8, Yc8, enable8);

        convertToSerialX convertHeadX9(nx1, nx2, nx3, nx4, nx5, nx6, nx7, nx8,
Xa9, Xb9, Xc9, enable9);
        convertToSerialY convertHeadY9(ny1, ny2, ny3, ny4, ny5, ny6, ny7, ny8,
Ya9, Yb9, Yc9, enable9);
```

```
        convertToSerialX convertHeadX10(ox1, ox2, ox3, ox4, ox5, ox6, ox7,
ox8, Xa10, Xb10, Xc10, enable10);
        convertToSerialY convertHeadY10(oy1, oy2, oy3, oy4, oy5, oy6, oy7,
oy8, Ya10, Yb10, Yc10, enable10);

        convertToSerialX convertHeadX11(px1, px2, px3, px4, px5, px6, px7,
px8, Xa11, Xb11, Xc11, enable11);
        convertToSerialY convertHeadY11(py1, py2, py3, py4, py5, py6, py7,
py8, Ya11, Yb11, Yc11, enable11);

        convertToSerialX convertHeadX12(qx1, qx2, qx3, qx4, qx5, qx6, qx7,
qx8, Xa12, Xb12, Xc12, enable12);
        convertToSerialY convertHeadY12(qy1, qy2, qy3, qy4, qy5, qy6, qy7,
qy8, Ya12, Yb12, Yc12, enable12);

        convertToSerialX convertHeadX13(rx1, rx2, rx3, rx4, rx5, rx6, rx7,
rx8, Xa13, Xb13, Xc13, enable13);
        convertToSerialY convertHeadY13(ry1, ry2, ry3, ry4, ry5, ry6, ry7,
ry8, Ya13, Yb13, Yc13, enable13);

        convertToSerialX convertHeadX14(sx1, sx2, sx3, sx4, sx5, sx6, sx7,
sx8, Xa14, Xb14, Xc14, enable14);
        convertToSerialY convertHeadY14(sy1, sy2, sy3, sy4, sy5, sy6, sy7,
sy8, Ya14, Yb14, Yc14, enable14);

        convertToSerialX convertHeadX15(tx1, tx2, tx3, tx4, tx5, tx6, tx7,
tx8, Xa15, Xb15, Xc15, enable15);
        convertToSerialY convertHeadY15(ty1, ty2, ty3, ty4, ty5, ty6, ty7,
ty8, Ya15, Yb15, Yc15, enable15);

        getYCode getYA(ay1, ay2, ay3, ay4, ay5, ay6, ay7,
ay8,ex1,fx1,gx1,hx1,ix1,jx1,kx1,lx1,mx1,nx1,ox1,px1,qx1,rx1,sx1,tx1, ey1,
ey2, ey3, ey4, ey5, ey6, ey7, ey8, fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8,
                gy1, gy2, gy3, gy4, gy5, gy6, gy7, gy8, hy1, hy2, hy3, hy4,
hy5, hy6, hy7, hy8,iy1, iy2, iy3, iy4, iy5, iy6, iy7, iy8,
                jy1, jy2, jy3, jy4, jy5, jy6, jy7, jy8, ky1, ky2, ky3, ky4,
ky5, ky6, ky7, ky8, ly1, ly2, ly3, ly4, ly5, ly6, ly7, ly8,
                my1, my2, my3, my4, my5, my6, my7, my8, ny1, ny2, ny3, ny4,
ny5, ny6, ny7, ny8,oy1, oy2, oy3, oy4, oy5, oy6, oy7, oy8,
                py1, py2, py3, py4, py5, py6, py7, py8,qy1, qy2, qy3, qy4,
qy5, qy6, qy7, qy8,ry1, ry2, ry3, ry4, ry5, ry6, ry7, ry8,
                sy1, sy2, sy3, sy4, sy5, sy6, sy7, sy8, ty1, ty2, ty3, ty4,
ty5, ty6, ty7, ty8);

        getYCode getYB(by1, by2, by3, by4, by5, by6, by7,
by8,ex2,fx2,gx2,hx2,ix2,jx2,kx2,lx2,mx2,nx2,ox2,px2,qx2,rx2,sx2,tx2, ey1,
ey2, ey3, ey4, ey5, ey6, ey7, ey8, fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8,
                gy1, gy2, gy3, gy4, gy5, gy6, gy7, gy8, hy1, hy2, hy3, hy4,
hy5, hy6, hy7, hy8,iy1, iy2, iy3, iy4, iy5, iy6, iy7, iy8,
                jy1, jy2, jy3, jy4, jy5, jy6, jy7, jy8, ky1, ky2, ky3, ky4,
ky5, ky6, ky7, ky8, ly1, ly2, ly3, ly4, ly5, ly6, ly7, ly8,
                my1, my2, my3, my4, my5, my6, my7, my8, ny1, ny2, ny3, ny4,
ny5, ny6, ny7, ny8,oy1, oy2, oy3, oy4, oy5, oy6, oy7, oy8,
                py1, py2, py3, py4, py5, py6, py7, py8,qy1, qy2, qy3, qy4,
qy5, qy6, qy7, qy8,ry1, ry2, ry3, ry4, ry5, ry6, ry7, ry8,
```

```
                sy1, sy2, sy3, sy4, sy5, sy6, sy7, sy8, ty1, ty2, ty3, ty4,
ty5, ty6, ty7, ty8);

        getYCode getYC(cy1, cy2, cy3, cy4, cy5, cy6, cy7,
cy8,ex3,fx3,gx3,hx3,ix3,jx3,kx3,lx3,mx3,nx3,ox3,px3,qx3,rx3,sx3,tx2, ey1,
ey2, ey3, ey4, ey5, ey6, ey7, ey8, fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8,
                gy1, gy2, gy3, gy4, gy5, gy6, gy7, gy8, hy1, hy2, hy3, hy4,
hy5, hy6, hy7, hy8,iy1, iy2, iy3, iy4, iy5, iy6, iy7, iy8,
                jy1, jy2, jy3, jy4, jy5, jy6, jy7, jy8, ky1, ky2, ky3, ky4,
ky5, ky6, ky7, ky8, ly1, ly2, ly3, ly4, ly5, ly6, ly7, ly8,
                my1, my2, my3, my4, my5, my6, my7, my8, ny1, ny2, ny3, ny4,
ny5, ny6, ny7, ny8,oy1, oy2, oy3, oy4, oy5, oy6, oy7, oy8,
                py1, py2, py3, py4, py5, py6, py7, py8,qy1, qy2, qy3, qy4,
qy5, qy6, qy7, qy8,ry1, ry2, ry3, ry4, ry5, ry6, ry7, ry8,
                sy1, sy2, sy3, sy4, sy5, sy6, sy7, sy8, ty1, ty2, ty3, ty4,
ty5, ty6, ty7, ty8);

        getYCode getYD(dy1, dy2, dy3, dy4, dy5, dy6, dy7,
dy8,ex4,fx4,gx4,hx4,ix4,jx4,kx4,lx4,mx4,nx4,ox4,px4,qx4,rx4,sx4,tx4, ey1,
ey2, ey3, ey4, ey5, ey6, ey7, ey8, fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8,
                gy1, gy2, gy3, gy4, gy5, gy6, gy7, gy8, hy1, hy2, hy3, hy4,
hy5, hy6, hy7, hy8,iy1, iy2, iy3, iy4, iy5, iy6, iy7, iy8,
                jy1, jy2, jy3, jy4, jy5, jy6, jy7, jy8, ky1, ky2, ky3, ky4,
ky5, ky6, ky7, ky8, ly1, ly2, ly3, ly4, ly5, ly6, ly7, ly8,
                my1, my2, my3, my4, my5, my6, my7, my8, ny1, ny2, ny3, ny4,
ny5, ny6, ny7, ny8,oy1, oy2, oy3, oy4, oy5, oy6, oy7, oy8,
                py1, py2, py3, py4, py5, py6, py7, py8,qy1, qy2, qy3, qy4,
qy5, qy6, qy7, qy8,ry1, ry2, ry3, ry4, ry5, ry6, ry7, ry8,
                sy1, sy2, sy3, sy4, sy5, sy6, sy7, sy8, ty1, ty2, ty3, ty4,
ty5, ty6, ty7, ty8);

        getYCode getYE(eey1, eey2, eey3, eey4, eey5, eey6, eey7,
eey8,ex5,fx5,gx5,hx5,ix5,jx5,kx5,lx5,mx5,nx5,ox5,px5,qx5,rx5,sx5,tx5, ey1,
ey2, ey3, ey4, ey5, ey6, ey7, ey8, fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8,
                gy1, gy2, gy3, gy4, gy5, gy6, gy7, gy8, hy1, hy2, hy3, hy4,
hy5, hy6, hy7, hy8,iy1, iy2, iy3, iy4, iy5, iy6, iy7, iy8,
                jy1, jy2, jy3, jy4, jy5, jy6, jy7, jy8, ky1, ky2, ky3, ky4,
ky5, ky6, ky7, ky8, ly1, ly2, ly3, ly4, ly5, ly6, ly7, ly8,
                my1, my2, my3, my4, my5, my6, my7, my8, ny1, ny2, ny3, ny4,
ny5, ny6, ny7, ny8,oy1, oy2, oy3, oy4, oy5, oy6, oy7, oy8,
                py1, py2, py3, py4, py5, py6, py7, py8,qy1, qy2, qy3, qy4,
qy5, qy6, qy7, qy8,ry1, ry2, ry3, ry4, ry5, ry6, ry7, ry8,
                sy1, sy2, sy3, sy4, sy5, sy6, sy7, sy8, ty1, ty2, ty3, ty4,
ty5, ty6, ty7, ty8);

        getYCode getYF(ffy1, ffy2, ffy3, ffy4, ffy5, ffy6, ffy7,
ffy8,ex6,fx6,gx6,hx6,ix6,jx6,kx6,lx6,mx6,nx6,ox6,px6,qx6,rx6,sx6,tx6, ey1,
ey2, ey3, ey4, ey5, ey6, ey7, ey8, fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8,
                gy1, gy2, gy3, gy4, gy5, gy6, gy7, gy8, hy1, hy2, hy3, hy4,
hy5, hy6, hy7, hy8,iy1, iy2, iy3, iy4, iy5, iy6, iy7, iy8,
                jy1, jy2, jy3, jy4, jy5, jy6, jy7, jy8, ky1, ky2, ky3, ky4,
ky5, ky6, ky7, ky8, ly1, ly2, ly3, ly4, ly5, ly6, ly7, ly8,
                my1, my2, my3, my4, my5, my6, my7, my8, ny1, ny2, ny3, ny4,
ny5, ny6, ny7, ny8,oy1, oy2, oy3, oy4, oy5, oy6, oy7, oy8,
                py1, py2, py3, py4, py5, py6, py7, py8,qy1, qy2, qy3, qy4,
qy5, qy6, qy7, qy8,ry1, ry2, ry3, ry4, ry5, ry6, ry7, ry8,
```

```
                sy1, sy2, sy3, sy4, sy5, sy6, sy7, sy8, ty1, ty2, ty3, ty4,
ty5, ty6, ty7, ty8);

        getYCode getYG(ggy1, ggy2, ggy3, ggy4, ggy5, ggy6, ggy7,
ggy8,ex7,fx7,gx7,hx7,ix7,jx7,kx7,lx7,mx7,nx7,ox7,px7,qx7,rx7,sx7,tx7, ey1,
ey2, ey3, ey4, ey5, ey6, ey7, ey8, fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8,
                gy1, gy2, gy3, gy4, gy5, gy6, gy7, gy8, hy1, hy2, hy3, hy4,
hy5, hy6, hy7, hy8,iy1, iy2, iy3, iy4, iy5, iy6, iy7, iy8,
                jy1, jy2, jy3, jy4, jy5, jy6, jy7, jy8, ky1, ky2, ky3, ky4,
ky5, ky6, ky7, ky8, ly1, ly2, ly3, ly4, ly5, ly6, ly7, ly8,
                my1, my2, my3, my4, my5, my6, my7, my8, ny1, ny2, ny3, ny4,
ny5, ny6, ny7, ny8,oy1, oy2, oy3, oy4, oy5, oy6, oy7, oy8,
                py1, py2, py3, py4, py5, py6, py7, py8,qy1, qy2, qy3, qy4,
qy5, qy6, qy7, qy8,ry1, ry2, ry3, ry4, ry5, ry6, ry7, ry8,
                sy1, sy2, sy3, sy4, sy5, sy6, sy7, sy8, ty1, ty2, ty3, ty4,
ty5, ty6, ty7, ty8);

        getYCode getYH(hhy1, hhy2, hhy3, hhy4, hhy5, hhy6, hhy7,
hhy8,ex8,fx8,gx8,hx8,ix8,jx8,kx8,lx8,mx8,nx8,ox8,px8,qx8,rx8,sx8,tx8, ey1,
ey2, ey3, ey4, ey5, ey6, ey7, ey8, fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8,
                gy1, gy2, gy3, gy4, gy5, gy6, gy7, gy8, hy1, hy2, hy3, hy4,
hy5, hy6, hy7, hy8,iy1, iy2, iy3, iy4, iy5, iy6, iy7, iy8,
                jy1, jy2, jy3, jy4, jy5, jy6, jy7, jy8, ky1, ky2, ky3, ky4,
ky5, ky6, ky7, ky8, ly1, ly2, ly3, ly4, ly5, ly6, ly7, ly8,
                my1, my2, my3, my4, my5, my6, my7, my8, ny1, ny2, ny3, ny4,
ny5, ny6, ny7, ny8,oy1, oy2, oy3, oy4, oy5, oy6, oy7, oy8,
                py1, py2, py3, py4, py5, py6, py7, py8,qy1, qy2, qy3, qy4,
qy5, qy6, qy7, qy8,ry1, ry2, ry3, ry4, ry5, ry6, ry7, ry8,
                sy1, sy2, sy3, sy4, sy5, sy6, sy7, sy8, ty1, ty2, ty3, ty4,
ty5, ty6, ty7, ty8);


endmodule



module getYCode(y1, y2, y3, y4, y5, y6, y7,
y8,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t, ey1, ey2, ey3, ey4, ey5, ey6, ey7, ey8,
fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8,
                gy1, gy2, gy3, gy4, gy5, gy6, gy7, gy8, hy1, hy2, hy3, hy4,
hy5, hy6, hy7, hy8,iy1, iy2, iy3, iy4, iy5, iy6, iy7, iy8,
                jy1, jy2, jy3, jy4, jy5, jy6, jy7, jy8, ky1, ky2, ky3, ky4,
ky5, ky6, ky7, ky8, ly1, ly2, ly3, ly4, ly5, ly6, ly7, ly8,
                my1, my2, my3, my4, my5, my6, my7, my8, ny1, ny2, ny3, ny4,
ny5, ny6, ny7, ny8,oy1, oy2, oy3, oy4, oy5, oy6, oy7, oy8,
                py1, py2, py3, py4, py5, py6, py7, py8,qy1, qy2, qy3, qy4,
qy5, qy6, qy7, qy8,ry1, ry2, ry3, ry4, ry5, ry6, ry7, ry8,
                sy1, sy2, sy3, sy4, sy5, sy6, sy7, sy8, ty1, ty2, ty3, ty4,
ty5, ty6, ty7, ty8);

        input e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t, ey1, ey2, ey3, ey4, ey5, ey6,
ey7, ey8, fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8,
                gy1, gy2, gy3, gy4, gy5, gy6, gy7, gy8, hy1, hy2, hy3, hy4,
hy5, hy6, hy7, hy8,iy1, iy2, iy3, iy4, iy5, iy6, iy7, iy8,
                jy1, jy2, jy3, jy4, jy5, jy6, jy7, jy8, ky1, ky2, ky3, ky4,
ky5, ky6, ky7, ky8, ly1, ly2, ly3, ly4, ly5, ly6, ly7, ly8,
```

```
                my1, my2, my3, my4, my5, my6, my7, my8, ny1, ny2, ny3, ny4,
ny5, ny6, ny7, ny8,oy1, oy2, oy3, oy4, oy5, oy6, oy7, oy8,
                py1, py2, py3, py4, py5, py6, py7, py8,qy1, qy2, qy3, qy4,
qy5, qy6, qy7, qy8,ry1, ry2, ry3, ry4, ry5, ry6, ry7, ry8,
                sy1, sy2, sy3, sy4, sy5, sy6, sy7, sy8, ty1, ty2, ty3, ty4,
ty5, ty6, ty7, ty8;

        output y1, y2, y3, y4, y5, y6, y7, y8;

                createY wy1(y1,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t, ey1,  fy1,
gy1,  hy1,  iy1,  jy1,  ky1,  ly1,  my1,  ny1,  oy1,  py1,  qy1,  ry1,  sy1,
ty1);
                createY wy2(y2,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t, ey2,  fy2,
gy2,  hy2,  iy2,  jy2,  ky2,  ly2,  my2,  ny2,  oy2,  py2,  qy2,  ry2,  sy2,
ty2);
                createY wy3(y3,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t, ey3,  fy3,
gy3,  hy3,  iy3,  jy3,  ky3,  ly3,  my3,  ny3,  oy3,  py3,  qy3,  ry3,  sy3,
ty3);
                createY wy4(y4,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t, ey4,  fy4,
gy4,  hy4,  iy4,  jy4,  ky4,  ly4,  my4,  ny4,  oy4,  py4,  qy4,  ry4,  sy4,
ty4);
                createY wy5(y5,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t, ey5,  fy5,
gy5,  hy5,  iy5,  jy5,  ky5,  ly5,  my5,  ny5,  oy5,  py5,  qy5,  ry5,  sy5,
ty5);
                createY wy6(y6,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t, ey6,  fy6,
gy6,  hy6,  iy6,  jy6,  ky6,  ly6,  my6,  ny6,  oy6,  py6,  qy6,  ry6,  sy6,
ty6);
                createY wy7(y7,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t, ey7,  fy7,
gy7,  hy7,  iy7,  jy7,  ky7,  ly7,  my7,  ny7,  oy7,  py7,  qy7,  ry7,  sy7,
ty7);
                createY wy8(y8,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t, ey8,  fy8,
gy8,  hy8,  iy8,  jy8,  ky8,  ly8,  my8,  ny8,  oy8,  py8,  qy8,  ry8,  sy8,
ty8);



endmodule

module createY(y,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,ey, fy, gy, hy, iy, jy, ky,
ly, my, ny, oy, py, qy, ry, sy, ty);

        input e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,ey, fy, gy, hy, iy, jy, ky, ly,
my, ny, oy, py, qy, ry, sy, ty;

        output y;

        and e0(qe, e, ey), f0(qf, f, fy), g0(qg,g,gy), h0(qh,h,hy), i0(qi, i,
iy),
                j0(qj, j , jy), k0(qk, k, ky), l0(ql, l, ly), m0(qm, m, my),
n0(qn, n, ny),
                o0(qo, o, oy), p0(qp, p, py), q0(qq, q, qy), r0(qr, r, ry),
s0(qs, s, sy), t0(qt, t, ty);

        or getY(y, qe, qf, qg, qh, qi, qj, qk, ql, qm, qn, qo, qp, qq, qr, qs,
qt);
```

```
endmodule

module convertToSerialY(g1, g2, g3, g4, g5, g6, g7, g8, a, b, c, enable);
        input a, b, c, enable;
        output g1, g2, g3, g4, g5, g6, g7, g8;

        and G1(g1, enable, ~a, ~b, ~c);
        //g2
        and G2(g2, enable, ~a, ~b, c);
        //g3
        and G3(g3, enable, ~a, b, ~c);
        //g4
        and G4(g4, enable, ~a, b, c);
        //g5
        and G5(g5, enable, a, ~b, ~c);
        //g6
        and G6(g6, enable, a, ~b, c);
        //g7
        and G7(g7, enable, a, b, ~c);
        //g8
        and G8(g8, enable, a, b, c);
endmodule

module convertToSerialX(g1, g2, g3, g4, g5, g6, g7, g8, a, b, c, enable);
        input a, b, c, enable;
        output g1, g2, g3, g4, g5, g6, g7, g8;

        and G1(g1, enable, ~a, ~b, ~c);
        //g2
        and G2(g2, enable, ~a, ~b, c);
        //g3
        and G3(g3, enable, ~a, b, ~c);
        //g4
        and G4(g4, enable, ~a, b, c);
        //g5
        and G5(g5, enable, a, ~b, ~c);
        //g6
        and G6(g6, enable, a, ~b, c);
        //g7
        and G7(g7, enable, a, b, ~c);
        //g8
        and G8(g8, enable, a, b, c);
endmodule


module scanner(Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8, ay1, ay2, ay3, ay4, ay5, ay6,
ay7, ay8,by1, by2, by3, by4, by5, by6, by7, by8,cy1, cy2, cy3, cy4, cy5, cy6,
cy7, cy8,
        dy1, dy2, dy3, dy4, dy5, dy6, dy7, dy8, eey1, eey2, eey3, eey4, eey5,
eey6, eey7, eey8, ffy1, ffy2, ffy3, ffy4, ffy5, ffy6, ffy7, ffy8,
ggy1, ggy2, ggy3, ggy4, ggy5, ggy6, ggy7, ggy8, hhy1, hhy2, hhy3, hhy4, hhy5,
hhy6, hhy7, hhy8, clk, rst, A, B, C);


        input  ay1, ay2, ay3, ay4, ay5, ay6, ay7, ay8,by1, by2, by3, by4, by5,
by6, by7, by8,cy1, cy2, cy3, cy4, cy5, cy6, cy7, cy8,
```

```
        dy1, dy2, dy3, dy4, dy5, dy6, dy7, dy8, eey1, eey2, eey3, eey4, eey5,
eey6, eey7, eey8, ffy1, ffy2, ffy3, ffy4, ffy5, ffy6, ffy7, ffy8,
ggy1, ggy2, ggy3, ggy4, ggy5, ggy6, ggy7, ggy8, hhy1, hhy2, hhy3, hhy4, hhy5,
hhy6, hhy7, hhy8, clk, rst, A, B, C;

        output Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8;


        and a0(arnoldy0,ay1, ~A, ~B, ~C), a1(arnoldy1,by1, ~A, ~B, C),
a2(arnoldy2,cy1, ~A, B, ~C), a3(arnoldy3,dy1, ~A, B, C), a4(arnoldy4,eey1, A,
~B, ~C), a5(arnoldy5,ffy1, A, ~B, C),
                        a6(arnoldy6,ggy1, A, B, ~C), a7(arnoldy7,hhy1, A, B,
C);

        and b0(barney0,ay2, ~A, ~B, ~C), b1(barney1,by2, ~A, ~B, C),
b2(barney2,cy2, ~A, B, ~C), b3(barney3,dy2, ~A, B, C), b4(barney4,eey2, A,
~B, ~C), b5(barney5,ffy2, A, ~B, C),
                        b6(barney6,ggy2, A, B, ~C), b7(barney7,hhy2, A, B, C);

        and c0(carry0,ay3, ~A, ~B, ~C), c1(carry1,by3, ~A, ~B, C),
c2(carry2,cy3, ~A, B, ~C), c3(carry3,dy3, ~A, B, C), c4(carry4,eey3, A, ~B,
~C), c5(carry5,ffy3, A, ~B, C),
                        c6(carry6,ggy3, A, B, ~C), c7(carry7, hhy3,A, B, C);

        and d0(davey0,ay4, ~A, ~B, ~C), d1(davey1,by4, ~A, ~B, C), d2(davey2,
cy4,~A, B, ~C), d3(davey3,dy4, ~A, B, C), d4(davey4,eey4, A, ~B, ~C),
d5(davey5,ffy4, A, ~B, C),
                        d6(davey6,ggy4, A, B, ~C), d7(davey7,hhy4, A, B, C);

        and e0(eevee0,ay5, ~A, ~B, ~C), e1(eevee1,by5, ~A, ~B, C),
e2(eevee2,cy5, ~A, B, ~C), e3(eevee3,dy5, ~A, B, C), e4(eevee4,eey5, A, ~B,
~C), e5(eevee5,ffy5, A, ~B, C),
                        e6(eevee6,ggy5, A, B, ~C), e7(eevee7,hhy5, A, B, C);

        and f0(froggy0,ay6, ~A, ~B, ~C), f1(froggy1,by6, ~A, ~B, C),
f2(froggy2,cy6, ~A, B, ~C), f3(froggy3, dy6,~A, B, C), f4(froggy4,eey6, A,
~B, ~C), f5(froggy5,ffy6, A, ~B, C),
                        f6(froggy6,ggy6, A, B, ~C), f7(froggy7, hhy6,A, B, C);

        and g0(gilly0,ay7, ~A, ~B, ~C), g1(gilly1,by7, ~A, ~B, C),
g2(gilly2,cy7, ~A, B, ~C), g3(gilly3,dy7, ~A, B, C), g4(gilly4,eey7, A, ~B,
~C), g5(gilly5, ffy7,A, ~B, C),
                        g6(gilly6,ggy7, A, B, ~C), g7(gilly7, hhy7,A, B, C);

        and h0(harry0,ay8, ~A, ~B, ~C), h1(harry1, by8,~A, ~B, C),
h2(harry2,cy8, ~A, B, ~C), h3(harry3,dy8, ~A, B, C), h4(harry4, eey8,A, ~B,
~C), h5(harry5,ffy8, A, ~B, C),
                        h6(harry6,ggy8, A, B, ~C), h7(harry7,hhy8,A, B, C);

        or doY1(Y1, arnoldy0, arnoldy1, arnoldy2, arnoldy3, arnoldy4,
arnoldy5, arnoldy6, arnoldy7),
        doY2(Y2,  barney0, barney1, barney2, barney3, barney4, barney5,
barney6, barney7),
        doY3(Y3,  carry0, carry1, carry2, carry3, carry4, carry5, carry6,
carry7),
        doY4(Y4,  davey0, davey1, davey2, davey3, davey4, davey5, davey6,
davey7),
```

```
        doY5(Y5,  eevee0, eevee1, eevee2, eevee3, eevee4, eevee5, eevee6,
eevee7),
        doY6(Y6, froggy0, froggy1, froggy2, froggy3, froggy4, froggy5,
froggy6, froggy7),
        doY7(Y7,  gilly0, gilly1, gilly2, gilly3, gilly4, gilly5, gilly6,
gilly7),
        doY8(Y8, harry0, harry1, harry2, harry3, harry4, harry5, harry6,
harry7);



endmodule


module dFforFood(q, qBar, D, clk);
  input D, clk;
  output q, qBar;

  reg q;

  not n1 (qBar, q);

  always@ (posedge clk)
  begin
      q = D;

  end

endmodule

module foodPosG(fxA, fxB, fxC,fyA, fyB, fyC,start,rst,clk,eat);
    input rst,clk,eat,start;
    output fxA, fxB, fxC,fyA, fyB, fyC;

    or or1(enable,start,eat);
  foodPos foodPos1(fxa, fxb, fxc,fya, fyb, fyc,rst,clk);

          dFforFood dF1(fxA, qBarA, fxa, enable);
          dFforFood dF2(fxB, qBarB, fxb, enable);
          dFforFood dF3(fxC, qBarC, fxc, enable);
          dFforFood dF4(fyA, qBarD, fya, enable);
          dFforFood dF5(fyB, qBarE, fyb, enable);
          dFforFood dF6(fyC, qBarF, fyc, enable);

endmodule

module dF(q, qBar, D, clk, rst);
  input D, clk, rst;
  output q, qBar;

  reg q;

  not n1 (qBar, q);

  always@ (posedge rst or posedge clk)
  begin
    if(rst)
```

```
      q = 0;

    else
      q = D;

  end

endmodule

module foodPos(fxA, fxB, fxC,fyA, fyB, fyC,rst,clk);

  input rst,clk;
  output fxA, fxB, fxC,fyA, fyB, fyC;

  and and1(all1s,fxA,fxB,fxC,fyA,fyB);
  xor xor1(fyCxoralll1s,all1s,fyC);
  xnor xnor1(fxAxnorfyC,fxA,fyCxoralll1s);
  assign  T=fxAxnorfyC;

  dF dF1(fxA, qBarA, fyCxoralll1s, clk, rst);
  dF dF2(fxB, qBarB, T, clk, rst);
  dF dF3(fxC, qBarC, fxB, clk, rst);
  dF dF4(fyA, qBarD, fxC, clk, rst);
  dF dF5(fyB, qBarE, fyA, clk, rst);
  dF dF6(fyC, qBarF, fyB, clk, rst);

endmodule



module dFEnabled(q, qBar, D, clk, rst, enable);
  input D, clk, rst, enable;
  output q, qBar;

  reg q;

  not n1 (qBar, q);

  always@ (posedge rst or posedge clk)
  begin
    if(rst)
      q = 0;
       else if(~enable)
               q = q;
    else
      q = D;

  end

endmodule

module eatFood(eat, xA, xB, xC, FxA, FxB, FxC, yA, yB, yC,
                              FyA, FyB, FyC,clk,rst);

        input xA, xB, xC, FxA, FxB, FxC, yA, yB, yC,
                              FyA, FyB, FyC,rst,clk;
```

```
        output eat;

        //compare x positions
        xnor xnor1(xResult1, FxA, xA);
        xnor xnor2(xResult2, FxB, xB);
        xnor xnor3(xResult3, FxC, xC);

        and and1(xOut, xResult1, xResult2, xResult3);

        //compare y postions
        xnor xnor4(yResult1, FyA, yA);
        xnor xnor5(yResult2, FyB, yB);
        xnor xnor6(yResult3, FyC, yC);

        and and2(yOut, yResult1, yResult2, yResult3);
        //output eat '1' True '0' False
        and and3(samepos, xOut, yOut);

  and (almosteat,samepos,qBarEat);

  dF dfforeat(eat,qBarEat,samepos,clk,rst);

endmodule

module up4bit(qA, qB, qC, qD, rst, clk);
        output qA, qB, qC, qD;
        input rst, clk;

        dF Aff(qA, qBarA, inA, ~clk, rst);
        dF Bff(qB, qBarB, inB, ~clk, rst);
        dF Cff(qC, qBarC, inC, ~clk, rst);
        dF Dff(qD, qBarD, inD, ~clk, rst);

        and and0(ABCD, qA, qB, qC, qD);
        //D flipFlop
        or or0(inD, qBarD, ABCD);
        //C flipflop
        xor Cin1(cin1, qC, qD);
        or or1(inC, cin1, ABCD);
        //B flipflop
        and and1(Bin1,qBarB, qC, qD), and2(Bin2, qB, qBarC), and3(Bin3, qB,
qBarD);
        or or2(inB, Bin1, Bin2, ABCD, Bin3);
        //A flipflop
        and andA1(BCD, qB, qC, qD);
        or or3(inA, BCD, qA);

endmodule

module enableLogic(enable1, enable2, enable3, enable4, enable5, enable6,
enable7,
enable8, enable9, enable10, enable11, enable12, enable13, enable14, enable15,
A, B, C, D);

  input  A, B, C, D;
```

```
  output enable1, enable2, enable3, enable4, enable5, enable6, enable7,
enable8, enable9, enable10, enable11, enable12, enable13, enable14, enable15;


//enable1
 or eno1(enable1, A, B, C, D);
//enable2
or eno2(enable2, A, B, C);
//enable3
and ena0(t0, C, D);
or eno3(enable3, t0, A, B);
//enable4
or eno4(enable4, A, B);
//enable5
and ena1(t1, C, B), ena2(t2, B, D);
or eno5(enable5, t1, t2, A);
//enable6
and ena3(t3, B, C);
or eno6(enable6, A, t3);
//enable7
and ena4(t4, B, C, D);
or eno7(enable7, A, t4);
//enable8
assign enable8 = A;
//enable9
or eno9(t5, B, C, D);
and ena5(enable9, t5, A);
//enable10
or eno10(t14, B, C);
and ena10(enable10, A, t14);
//enable11
and ena6(t6, A, B), ena333(t7, A, C, D);
or eno11(enable11, t6, t7);
//enable12
and ena7(enable12, A, B);
//enable13
or eno13(t13, C, D);
and ena13(enable13, A, B, t13);
//enable14
and ena8(enable14, A, B, C);
//enable15
and ena9(enable15, A, B, C, D);

endmodule

module delayFF(q, qBar, D, clk,rst, enable);
  input D, clk, enable,rst;
  output q, qBar;

  reg q;

  not n1 (qBar, q);

  always@ (posedge clk)
  begin
  if(rst)
      q = 0;
```

```
        else if(enable)
      q = D;
    else
      q = q;

  end

endmodule


module positionReg(delayedX0, delayedX1, delayedX2, x0, x1, x2, clk,rst,
enable);
        output delayedX0, delayedX1, delayedX2;
        input x0, x1, x2, clk, rst,enable;

        delayFF delayx0(delayedX0, qBar0, x0, clk,rst, enable);
        delayFF delayx1(delayedX1, qBar1, x1, clk,rst, enable);
        delayFF delayx2(delayedX2, qBar2, x2, clk,rst, enable);

endmodule

module positionCaster(
                                Xa1,Xb1,Xc1,Ya1,Yb1,Yc1,
                                Xa2,Xb2,Xc2,Ya2,Yb2,Yc2,
                                Xa3,Xb3,Xc3,Ya3,Yb3,Yc3,
                                        Xa4,Xb4,Xc4,Ya4,Yb4,Yc4,
                                        Xa5,Xb5,Xc5,Ya5,Yb5,Yc5,
                                        Xa6,Xb6,Xc6,Ya6,Yb6,Yc6,
                                        Xa7,Xb7,Xc7,Ya7,Yb7,Yc7,
                                        Xa8,Xb8,Xc8,Ya8,Yb8,Yc8,
                                        Xa9,Xb9,Xc9,Ya9,Yb9,Yc9,
                                        Xa10,Xb10,Xc10,Ya10,Yb10,Yc10,
                                        Xa11,Xb11,Xc11,Ya11,Yb11,Yc11,
                                        Xa12,Xb12,Xc12,Ya12,Yb12,Yc12,
                                        Xa13,Xb13,Xc13,Ya13,Yb13,Yc13,
                                        Xa14,Xb14,Xc14,Ya14,Yb14,Yc14,
                                        Xa15,Xb15,Xc15,Ya15,Yb15,Yc15,clk,rst,
                Xa,Xb,Xc,Ya,Yb,Yc,

        enable1,enable2,enable3,enable4,enable5,enable6,enable7,enable8,enable
9,enable10,

        enable11,enable12,enable13,enable14,enable15);

   input Xa,Xb,Xc,Ya,Yb,Yc,clk,rst,

enable1,enable2,enable3,enable4,enable5,enable6,enable7,enable8,enable9,enabl
e10,
                enable11,enable12,enable13,enable14,enable15;

   output   Xa1,Xb1,Xc1,Ya1,Yb1,Yc1,
                        Xa2,Xb2,Xc2,Ya2,Yb2,Yc2,
                        Xa3,Xb3,Xc3,Ya3,Yb3,Yc3,
                        Xa4,Xb4,Xc4,Ya4,Yb4,Yc4,
                        Xa5,Xb5,Xc5,Ya5,Yb5,Yc5,
                        Xa6,Xb6,Xc6,Ya6,Yb6,Yc6,
                        Xa7,Xb7,Xc7,Ya7,Yb7,Yc7,
```

```
                        Xa8,Xb8,Xc8,Ya8,Yb8,Yc8,
                        Xa9,Xb9,Xc9,Ya9,Yb9,Yc9,
                        Xa10,Xb10,Xc10,Ya10,Yb10,Yc10,
                        Xa11,Xb11,Xc11,Ya11,Yb11,Yc11,
                Xa12,Xb12,Xc12,Ya12,Yb12,Yc12,
                        Xa13,Xb13,Xc13,Ya13,Yb13,Yc13,
                        Xa14,Xb14,Xc14,Ya14,Yb14,Yc14,
                        Xa15,Xb15,Xc15,Ya15,Yb15,Yc15;

          positionReg
positionForX0(Xa0,Xb0,Xc0,Xa,Xb,Xc,clk,rst,enable1),positionForY0(Ya0,Yb0,Yc0
,Ya,Yb,Yc,clk,rst,enable1);
          positionReg
positionForX1(Xa1,Xb1,Xc1,Xa0,Xb0,Xc0,clk,rst,enable1),positionForY1(Ya1,Yb1,
Yc1,Ya0,Yb0,Yc0,clk,rst,enable1);
          positionReg
positionForX2(Xa2,Xb2,Xc2,Xa1,Xb1,Xc1,clk,rst,enable2),positionForY2(Ya2,Yb2,
Yc2,Ya1,Yb1,Yc1,clk,rst,enable2);
          positionReg
positionForX3(Xa3,Xb3,Xc3,Xa2,Xb2,Xc2,clk,rst,enable3),positionForY3(Ya3,Yb3,
Yc3,Ya2,Yb2,Yc2,clk,rst,enable3);
          positionReg
positionForX4(Xa4,Xb4,Xc4,Xa3,Xb3,Xc3,clk,rst,enable4),positionForY4(Ya4,Yb4,
Yc4,Ya3,Yb3,Yc3,clk,rst,enable4);
          positionReg
positionForX5(Xa5,Xb5,Xc5,Xa4,Xb4,Xc4,clk,rst,enable5),positionForY5(Ya5,Yb5,
Yc5,Ya4,Yb4,Yc4,clk,rst,enable5);
          positionReg
positionForX6(Xa6,Xb6,Xc6,Xa5,Xb5,Xc5,clk,rst,enable6),positionForY6(Ya6,Yb6,
Yc6,Ya5,Yb5,Yc5,clk,rst,enable6);
          positionReg
positionForX7(Xa7,Xb7,Xc7,Xa6,Xb6,Xc6,clk,rst,enable7),positionForY7(Ya7,Yb7,
Yc7,Ya6,Yb6,Yc6,clk,rst,enable7);
          positionReg
positionForX8(Xa8,Xb8,Xc8,Xa7,Xb7,Xc7,clk,rst,enable8),positionForY8(Ya8,Yb8,
Yc8,Ya7,Yb7,Yc7,clk,rst,enable8);
          positionReg
positionForX9(Xa9,Xb9,Xc9,Xa8,Xb8,Xc8,clk,rst,enable9),positionForY9(Ya9,Yb9,
Yc9,Ya8,Yb8,Yc8,clk,rst,enable9);
          positionReg
positionForX10(Xa10,Xb10,Xc10,Xa9,Xb9,Xc9,clk,rst,enable10),positionForY10(Ya
10,Yb10,Yc10,Ya9,Yb9,Yc9,clk,rst,enable10);
          positionReg
positionForX11(Xa11,Xb11,Xc11,Xa10,Xb10,Xc10,clk,rst,enable11),positionForY11
(Ya11,Yb11,Yc11,Ya10,Yb10,Yc10,clk,rst,enable11);
          positionReg
positionForX12(Xa12,Xb12,Xc12,Xa11,Xb11,Xc11,clk,rst,enable12),positionForY12
(Ya12,Yb12,Yc12,Ya11,Yb11,Yc11,clk,rst,enable12);
          positionReg
positionForX13(Xa13,Xb13,Xc13,Xa12,Xb12,Xc12,clk,rst,enable13),positionForY13
(Ya13,Yb13,Yc13,Ya12,Yb12,Yc12,clk,rst,enable13);
          positionReg
positionForX14(Xa14,Xb14,Xc14,Xa13,Xb13,Xc13,clk,rst,enable14),positionForY14
(Ya14,Yb14,Yc14,Ya13,Yb13,Yc13,clk,rst,enable14);
          positionReg
positionForX15(Xa15,Xb15,Xc15,Xa14,Xb14,Xc14,clk,rst,enable15),positionForY15
(Ya15,Yb15,Yc15,Ya14,Yb14,Yc14,clk,rst,enable15);
```

```
endmodule




module
Xnor(XaXa1,XbXb1,XcXc1,YaYa1,YbYb1,YcYc1,Xa,Xb,Xc,Ya,Yb,Yc,Xa1,Xb1,Xc1,Ya1,Yb
1,Yc1);

  input Xa,Xb,Xc,Ya,Yb,Yc,Xa1,Xb1,Xc1,Ya1,Yb1,Yc1;
  output XaXa1,XbXb1,XcXc1,YaYa1,YbYb1,YcYc1;

        xnor xnor1(XaXa1,Xa,Xa1);
        xnor xnor2(XbXb1,Xb,Xb1);
        xnor xnor3(XcXc1,Xc,Xc1);
        xnor xnor4(YaYa1,Ya,Ya1);
        xnor xnor5(YbYb1,Yb,Yb1);
        xnor xnor6(YcYc1,Yc,Yc1);

endmodule

module ifHitSelf(dead,Xa,Xb,Xc,Ya,Yb,Yc,
                                        Xa1,Xb1,Xc1,Ya1,Yb1,Yc1,
                                        Xa2,Xb2,Xc2,Ya2,Yb2,Yc2,
                                        Xa3,Xb3,Xc3,Ya3,Yb3,Yc3,
                Xa4,Xb4,Xc4,Ya4,Yb4,Yc4,
                Xa5,Xb5,Xc5,Ya5,Yb5,Yc5,
                Xa6,Xb6,Xc6,Ya6,Yb6,Yc6,
                Xa7,Xb7,Xc7,Ya7,Yb7,Yc7,
                Xa8,Xb8,Xc8,Ya8,Yb8,Yc8,
                Xa9,Xb9,Xc9,Ya9,Yb9,Yc9,
                Xa10,Xb10,Xc10,Ya10,Yb10,Yc10,
                Xa11,Xb11,Xc11,Ya11,Yb11,Yc11,
                Xa12,Xb12,Xc12,Ya12,Yb12,Yc12,
                Xa13,Xb13,Xc13,Ya13,Yb13,Yc13,
                Xa14,Xb14,Xc14,Ya14,Yb14,Yc14,
                Xa15,Xb15,Xc15,Ya15,Yb15,Yc15,
                enable1,enable2,enable3,enable4,
                enable5,enable6,enable7,enable8,enable9,enable10,

enable11,enable12,enable13,enable14,enable15
                );

  input Xa,Xb,Xc,Ya,Yb,Yc,
        Xa1,Xb1,Xc1,Ya1,Yb1,Yc1,
                Xa2,Xb2,Xc2,Ya2,Yb2,Yc2,
                Xa3,Xb3,Xc3,Ya3,Yb3,Yc3,
                Xa4,Xb4,Xc4,Ya4,Yb4,Yc4,
                Xa5,Xb5,Xc5,Ya5,Yb5,Yc5,
                Xa6,Xb6,Xc6,Ya6,Yb6,Yc6,
                Xa7,Xb7,Xc7,Ya7,Yb7,Yc7,
                Xa8,Xb8,Xc8,Ya8,Yb8,Yc8,
                Xa9,Xb9,Xc9,Ya9,Yb9,Yc9,
                Xa10,Xb10,Xc10,Ya10,Yb10,Yc10,
                Xa11,Xb11,Xc11,Ya11,Yb11,Yc11,
```

```
                Xa12,Xb12,Xc12,Ya12,Yb12,Yc12,
                Xa13,Xb13,Xc13,Ya13,Yb13,Yc13,
                Xa14,Xb14,Xc14,Ya14,Yb14,Yc14,
                Xa15,Xb15,Xc15,Ya15,Yb15,Yc15,
      enable1,enable2,enable3,enable4,
      enable5,enable6,enable7,enable8,enable9,enable10,
                enable11,enable12,enable13,enable14,enable15;

  output dead;

  Xnor
Xnor1(XaXa1,XbXb1,XcXc1,YaYa1,YbYb1,YcYc1,Xa,Xb,Xc,Ya,Yb,Yc,Xa1,Xb1,Xc1,Ya1,Y
b1,Yc1);
  Xnor
Xnor2(XaXa2,XbXb2,XcXc2,YaYa2,YbYb2,YcYc2,Xa,Xb,Xc,Ya,Yb,Yc,Xa2,Xb2,Xc2,Ya2,Y
b2,Yc2);
  Xnor
Xnor3(XaXa3,XbXb3,XcXc3,YaYa3,YbYb3,YcYc3,Xa,Xb,Xc,Ya,Yb,Yc,Xa3,Xb3,Xc3,Ya3,Y
b3,Yc3);
  Xnor
Xnor4(XaXa4,XbXb4,XcXc4,YaYa4,YbYb4,YcYc4,Xa,Xb,Xc,Ya,Yb,Yc,Xa4,Xb4,Xc4,Ya4,Y
b4,Yc4);
  Xnor
Xnor5(XaXa5,XbXb5,XcXc5,YaYa5,YbYb5,YcYc5,Xa,Xb,Xc,Ya,Yb,Yc,Xa5,Xb5,Xc5,Ya5,Y
b5,Yc5);
  Xnor
Xnor6(XaXa6,XbXb6,XcXc6,YaYa6,YbYb6,YcYc6,Xa,Xb,Xc,Ya,Yb,Yc,Xa6,Xb6,Xc6,Ya6,Y
b6,Yc6);
  Xnor
Xnor7(XaXa7,XbXb7,XcXc7,YaYa7,YbYb7,YcYc7,Xa,Xb,Xc,Ya,Yb,Yc,Xa7,Xb7,Xc7,Ya7,Y
b7,Yc7);
  Xnor
Xnor8(XaXa8,XbXb8,XcXc8,YaYa8,YbYb8,YcYc8,Xa,Xb,Xc,Ya,Yb,Yc,Xa8,Xb8,Xc8,Ya8,Y
b8,Yc8);
  Xnor
Xnor9(XaXa9,XbXb9,XcXc9,YaYa9,YbYb9,YcYc9,Xa,Xb,Xc,Ya,Yb,Yc,Xa9,Xb9,Xc9,Ya9,Y
b9,Yc9);
  Xnor
Xnor10(XaXa10,XbXb10,XcXc10,YaYa10,YbYb10,YcYc10,Xa,Xb,Xc,Ya,Yb,Yc,Xa10,Xb10,
Xc10,Ya10,Yb10,Yc10);
  Xnor
Xnor11(XaXa11,XbXb11,XcXc11,YaYa11,YbYb11,YcYc11,Xa,Xb,Xc,Ya,Yb,Yc,Xa11,Xb11,
Xc11,Ya11,Yb11,Yc11);
  Xnor
Xnor12(XaXa12,XbXb12,XcXc12,YaYa12,YbYb12,YcYc12,Xa,Xb,Xc,Ya,Yb,Yc,Xa12,Xb12,
Xc12,Ya12,Yb12,Yc12);
  Xnor
Xnor13(XaXa13,XbXb13,XcXc13,YaYa13,YbYb13,YcYc13,Xa,Xb,Xc,Ya,Yb,Yc,Xa13,Xb13,
Xc13,Ya13,Yb13,Yc13);
  Xnor
Xnor14(XaXa14,XbXb14,XcXc14,YaYa14,YbYb14,YcYc14,Xa,Xb,Xc,Ya,Yb,Yc,Xa14,Xb14,
Xc14,Ya14,Yb14,Yc14);
  Xnor
Xnor15(XaXa15,XbXb15,XcXc15,YaYa15,YbYb15,YcYc15,Xa,Xb,Xc,Ya,Yb,Yc,Xa15,Xb15,
Xc15,Ya15,Yb15,Yc15);

  and and1(hita1,XaXa1,XbXb1,XcXc1,YaYa1,YbYb1,YcYc1,enable1);
```

```
  and and2(hita2,XaXa2,XbXb2,XcXc2,YaYa2,YbYb2,YcYc2,enable2);
  and and3(hita3,XaXa3,XbXb3,XcXc3,YaYa3,YbYb3,YcYc3,enable3);
  and and4(hita4,XaXa4,XbXb4,XcXc4,YaYa4,YbYb4,YcYc4,enable4);
  and and5(hita5,XaXa5,XbXb5,XcXc5,YaYa5,YbYb5,YcYc5,enable5);
  and and6(hita6,XaXa6,XbXb6,XcXc6,YaYa6,YbYb6,YcYc6,enable6);
  and and7(hita7,XaXa7,XbXb7,XcXc7,YaYa7,YbYb7,YcYc7,enable7);
  and and8(hita8,XaXa8,XbXb8,XcXc8,YaYa8,YbYb8,YcYc8,enable8);
  and and9(hita9,XaXa9,XbXb9,XcXc9,YaYa9,YbYb9,YcYc9,enable9);
  and and10(hita10,XaXa10,XbXb10,XcXc10,YaYa10,YbYb10,YcYc10,enable10);
  and and11(hita11,XaXa11,XbXb11,XcXc11,YaYa11,YbYb11,YcYc11,enable11);
  and and12(hita12,XaXa12,XbXb12,XcXc12,YaYa12,YbYb12,YcYc12,enable12);
  and and13(hita13,XaXa13,XbXb13,XcXc13,YaYa13,YbYb13,YcYc13,enable13);
  and and14(hita14,XaXa14,XbXb14,XcXc14,YaYa14,YbYb14,YcYc14,enable14);
  and and15(hita15,XaXa15,XbXb15,XcXc15,YaYa15,YbYb15,YcYc15,enable15);

  or
gameEnd(dead,hita1,hita2,hita3,hita4,hita5,hita6,hita7,hita8,hita9,hita10,hit
a11,hita12,hita13,hita14,hita15);

endmodule



module Snake(upX,upY,enableX,enableY,qA, qB, qC,L,R,Die,start, clk, rst);
  input L,R,Die,start, clk, rst;
  output upX,upY,enableX,enableY,qA, qB, qC;


  dF A(qA, qBarA, snakeA, clk, rst);
  dF B(qB, qBarB, snakeB, clk, rst);
  dF C(qC, qBarC, snakeC, clk, rst);

  //A flip flop
  and andforA1(orA1, qBarA,qBarB, qC, R,~Die),
         andforA2(orA2, qBarA,qB, qC, L,~Die),
      andforA3(orA3,qA,qBarB,qBarC,~L,~R,~Die);
  or  orforA(snakeA,orA1,orA2,orA3);

  //B flip flop
  and andforB1(orB1, qBarA,qBarB, qC, L,~Die),
         andforB2(orB2, qBarA,qB, qBarC, L,~Die),
         andforB3(orB3, qBarA,qB, qC, R,~Die),
   andforB4(orB4,qA,qBarB, qBarC, R,~Die),
   andforB5(orB5, qBarA,qB,qBarC, ~L, ~R, ~Die),
   andforB6(orB6, qBarA,qB,qC,~Die);

  or orforB(snakeB, orB1, orB2,orB3,orB4, orB5, orB6);

  //C flip flop
 and andforC1(orC1, qBarA,qBarB, qBarC, start),
         andforC2(orC2, qBarA,qB, qBarC, R),
         andforC3(orC3, qBarA,qB, qBarC, L),
         andforC4(orC4, qA,qBarB, qBarC, L),
         andforC5(orC5, qA,qBarB, qBarC, R),
      andforC6(orC6, qBarA,qBarB,qC, ~L, ~R, ~Die),
      andforC7(orC7, qBarA,qB,qC, ~L, ~R, ~Die);
  or orforC(snakeC,orC1,orC2,orC3,orC4,orC5,orC6,orC7);
```

```
  //enableX
  and enXa(enX1, qBarA, qB, qBarC), enXa0(enX2, qA, qBarB, qBarC);
  or enableTheX(enableX, enX1, enX2);
  //enableY
  and enYa(enY1, qBarA, qBarB, qC), enYa0(enY2, qBarA, qB, qC);
  or enableTheY(enableY, enY1, enY2);
  //UpX
  assign upX = qA;
  //UpY
  assign upY= qBarB;

endmodule


module upDown(qA, qB, qC, rst, clk,Up,enable);
        output qA, qB, qC;
        input rst, clk, Up, enable;

        dFEnabled Aff(qA, qBarA, inA, clk, rst, enable);
        dFEnabled Bff(qB, qBarB, inB, clk, rst, enable);
        dFEnabled Cff(qC, qBarC, inC, clk, rst, enable);

        //Aff
  and inA0(inUpA1,qBarA, qB, qC), inA1(inUpA2, qA, qBarC), inA2(inUpA3, qA,
qBarB);
  or inAUp(UpInA, inUpA1, inUpA2, inUpA3);
  and anA(DownAt0, qA, qB), an0A(DownAt1, qA, qC), an1A(DownAt2, qBarA,
qBarB, qBarC);
        or oA(DownInA, DownAt0, DownAt1, DownAt2);
  and an2A(CountUp,UpInA,Up), an3A(CountDown, DownInA, ~Up);
  or goA(inA, CountUp, CountDown);
        //Bff
        xor intoB(DownInB, qBarB, qC);
  xor intoBUp(UpInB, qB, qC);
  and inB0(BUp, UpInB, Up), inB1(BDown, DownInB, ~Up);
  or goB(inB, BUp, BDown);
        //Cff
        assign inC = qBarC;

endmodule

module dFShift(q, qBar, D, loader, clk, rst, load);
  input D, clk, rst, loader, load;
  output q, qBar;

  reg q;

  not n1 (qBar, q);

  always@ (posedge rst or posedge clk)
  begin
    if(rst)
      q = 1;

      else if(load)
        q = loader;
```

```
    else
      q = D;

  end

endmodule


module shiftReg(qA, qB, qC,qD, qE, qF, qG, qH, clk, rst, load);

        input clk, rst, load;
        output qA, qB, qC, qD, qE, qF, qG, qH;

        dFShift shiftA(qA, qBarA, qH, 1'b0, clk, rst, load);
        dFShift shiftB(qB, qBarB, qA, 1'b1, clk, rst, load);
        dFShift shiftC(qC, qBarC, qB, 1'b1, clk, rst, load);
        dFShift shiftD(qD, qBarD, qC, 1'b1, clk, rst, load);
        dFShift shiftE(qE, qBarE, qD, 1'b1, clk, rst, load);
        dFShift shiftF(qF, qBarF, qE, 1'b1, clk, rst, load);
        dFShift shiftG(qG, qBarG, qF, 1'b1, clk, rst, load);
        dFShift shiftH(qH, qBarH, qG, 1'b1, clk, rst, load);

endmodule

module Test;

        reg    rst,slowclk, fastclk, L, R, start;
  parameter period = 10;
  parameter stimDelay = 20;

  shiftReg shiftershifter(Xout1, Xout2, Xout3, Xout4, Xout5, Xout6, Xout7,
Xout8, fastclk, dead, start);

  Snake trial(upX,upY,enableX,enableY,A,B,C,L, R, dead, start, slowclk, rst);

  upDown countX(Xa, Xb, Xc, start, slowclk ,upX,enableX);
        upDown countY(Ya, Yb, Yc, start, slowclk,upY,enableY);

  foodPosG foodPosG1(FxA, FxB, FxC, FyA, FyB, FyC,start,rst,fastclk,eat);

  eatFood eatfood(eat, Xa,Xb,Xc, FxA, FxB, FxC, Ya,Yb,Yc,
                                  FyA, FyB, FyC,slowclk,start);

        up4bit up4bit1(qA, qB, qC, qD, start,eat);

  enableLogic work(enable1, enable2, enable3, enable4, enable5, enable6,
enable7,
                   enable8, enable9, enable10, enable11, enable12, enable13,
enable14, enable15, qA, qB, qC, qD);

        positionCaster mypositionCaster(
                              Xa1,Xb1,Xc1,Ya1,Yb1,Yc1,
                              Xa2,Xb2,Xc2,Ya2,Yb2,Yc2,
                              Xa3,Xb3,Xc3,Ya3,Yb3,Yc3,
                                      Xa4,Xb4,Xc4,Ya4,Yb4,Yc4,
                                      Xa5,Xb5,Xc5,Ya5,Yb5,Yc5,
```

```
                                        Xa6,Xb6,Xc6,Ya6,Yb6,Yc6,
                                        Xa7,Xb7,Xc7,Ya7,Yb7,Yc7,
                                        Xa8,Xb8,Xc8,Ya8,Yb8,Yc8,
                                        Xa9,Xb9,Xc9,Ya9,Yb9,Yc9,
                                        Xa10,Xb10,Xc10,Ya10,Yb10,Yc10,
                                        Xa11,Xb11,Xc11,Ya11,Yb11,Yc11,
                                        Xa12,Xb12,Xc12,Ya12,Yb12,Yc12,
                                        Xa13,Xb13,Xc13,Ya13,Yb13,Yc13,
                                        Xa14,Xb14,Xc14,Ya14,Yb14,Yc14,
                                    Xa15,Xb15,Xc15,Ya15,Yb15,Yc15,slowclk,start,
                Xa,Xb,Xc,Ya,Yb,Yc,

        enable1,enable2,enable3,enable4,enable5,enable6,enable7,enable8,enable
9,enable10,

        enable11,enable12,enable13,enable14,enable15);

        ifHitSelf ifHitSelf1(dead,Xa,Xb,Xc,Ya,Yb,Yc,
                                              Xa1,Xb1,Xc1,Ya1,Yb1,Yc1,
                                              Xa2,Xb2,Xc2,Ya2,Yb2,Yc2,
                                              Xa3,Xb3,Xc3,Ya3,Yb3,Yc3,
                        Xa4,Xb4,Xc4,Ya4,Yb4,Yc4,
                        Xa5,Xb5,Xc5,Ya5,Yb5,Yc5,
                        Xa6,Xb6,Xc6,Ya6,Yb6,Yc6,
                        Xa7,Xb7,Xc7,Ya7,Yb7,Yc7,
                        Xa8,Xb8,Xc8,Ya8,Yb8,Yc8,
                        Xa9,Xb9,Xc9,Ya9,Yb9,Yc9,
                        Xa10,Xb10,Xc10,Ya10,Yb10,Yc10,
                        Xa11,Xb11,Xc11,Ya11,Yb11,Yc11,
                        Xa12,Xb12,Xc12,Ya12,Yb12,Yc12,
                        Xa13,Xb13,Xc13,Ya13,Yb13,Yc13,
                        Xa14,Xb14,Xc14,Ya14,Yb14,Yc14,

Xa15,Xb15,Xc15,Ya15,Yb15,Yc15,enable1,enable2,enable3,enable4,enable5,enable6
,enable7,enable8,enable9,enable10,

enable11,enable12,enable13,enable14,enable15);



   foodScan scanFood(Food1, Food2, Food3, Food4, Food5, Food6, Food7, Food8,
a, b, c, FyA, FyB, FyC, FxA, FxB, FxC);



        upDown up(a, b, c, rst, fastclk, 1'b1, 1'b1);


  scanner scan(Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8, ay1, ay2, ay3, ay4, ay5, ay6,
ay7, ay8,by1, by2, by3, by4, by5, by6, by7, by8,cy1, cy2, cy3, cy4, cy5, cy6,
cy7, cy8,
        dy1, dy2, dy3, dy4, dy5, dy6, dy7, dy8, eey1, eey2, eey3, eey4, eey5,
eey6, eey7, eey8, ffy1, ffy2, ffy3, ffy4, ffy5, ffy6, ffy7, ffy8,
  ggy1, ggy2, ggy3, ggy4, ggy5, ggy6, ggy7, ggy8, hhy1, hhy2, hhy3, hhy4,
hhy5, hhy6, hhy7, hhy8, fastclk, rst, a, b, c);
```

```
  condenseDisplay condenseDisplay1(ay1, ay2, ay3, ay4, ay5, ay6, ay7,
ay8,by1, by2, by3, by4, by5, by6, by7, by8,cy1, cy2, cy3, cy4, cy5, cy6, cy7,
cy8,
        dy1, dy2, dy3, dy4, dy5, dy6, dy7, dy8, eey1, eey2, eey3, eey4, eey5,
eey6, eey7, eey8, ffy1, ffy2, ffy3, ffy4, ffy5, ffy6, ffy7, ffy8,
  ggy1, ggy2, ggy3, ggy4, ggy5, ggy6, ggy7, ggy8, hhy1, hhy2, hhy3, hhy4,
hhy5, hhy6, hhy7, hhy8, Xa,Xb,Xc,Ya,Yb,Yc,
Xa1,Xb1,Xc1,Ya1,Yb1,Yc1,Xa2,Xb2,Xc2,Ya2,Yb2,Yc2,Xa3,Xb3,Xc3,Ya3,Yb3,Yc3,

        Xa4,Xb4,Xc4,Ya4,Yb4,Yc4,Xa5,Xb5,Xc5,Ya5,Yb5,Yc5,

        Xa6,Xb6,Xc6,Ya6,Yb6,Yc6,Xa7,Xb7,Xc7,Ya7,Yb7,Yc7,

        Xa8,Xb8,Xc8,Ya8,Yb8,Yc8,Xa9,Xb9,Xc9,Ya9,Yb9,Yc9,Xa10,Xb10,Xc10,Ya10,Yb
10,Yc10,

        Xa11,Xb11,Xc11,Ya11,Yb11,Yc11,Xa12,Xb12,Xc12,Ya12,Yb12,Yc12,

        Xa13,Xb13,Xc13,Ya13,Yb13,Yc13,Xa14,Xb14,Xc14,Ya14,Yb14,Yc14,

        Xa15,Xb15,Xc15,Ya15,Yb15,Yc15,enable1,enable2,enable3,enable4,

        enable5,enable6,enable7,enable8,enable9,enable10,enable11,enable12,
                                        enable13,enable14,enable15, dead);


        initial


                begin
                        rst=1;
                        fastclk=0;
    slowclk = 0;
     L = 0;
     R = 0;
     start = 0;
                end

        always #(0.1) fastclk=~fastclk;
  always #(0.5 * period) slowclk = ~slowclk;


  always @(posedge slowclk);

        initial
                begin
       // $monitor("clk=%b Xa=%b Xb=%b Xc=%b  Ya=%b Yb=%b Yc=%b Xa1=%b Xb1=%b
Xc1=%b  Ya1=%b Yb1=%b Yc1=%b "
         //             , clk,Xa,Xb,Xc,Ya,Yb,Yc,Xa1,Xb1,Xc1,Ya1,Yb1,Yc1,
$time);
                        begin

              //All left
                #stimDelay rst = 0;
                #period
```

```
                start=1;
                #period
                start=0;
                #period R = 1;
                #period R = 0;

                #period L = 1;
                #period L = 0;


                #(stimDelay * 2);

                //All right
              //  #stimDelay rst = 1;
               // #stimDelay rst = 0;

                #period start=1;
                #period start=0;
                #period L = 1;
                #period L = 0;

                #period R = 1;
                #period R = 0;

                #period L = 1;
                #period L = 0;

                #period R = 1;
                #period R = 0;
               // #(2*stimDelay) food=1;
              //  #period food=0;
                 #(stimDelay * 2);


                 //Game end
                // #stimDelay rst = 1;
                 //#stimDelay rst =
              end

            #(20*period);
            $finish;
          end


endmodule
```

Quartus Code of Snake

```
module SnakeTop3(galClock, start, fxa, fxb, fxc,fya, fyb, fyc, Xout1, Xout2,
Xout3, Xout4, Xout5, Xout6, Xout7, Xout8,
                                        Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8,Food1,
Food2, Food3, Food4, Food5, Food6, Food7, Food8,
                                                inL, inR, clk, rst, inStart);



        input inL, inR, clk, rst, inStart;

        //Outputs for scanning, body, food
        output galClock, start,fxa, fxb, fxc,fya, fyb, fyc,Xout1, Xout2,
Xout3, Xout4, Xout5, Xout6, Xout7, Xout8, Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8,
Food1, Food2, Food3, Food4, Food5, Food6, Food7, Food8;

        not n0(L, inL), n1(R, inR), n2(start, inStart), n3(galClock, eat);

        shiftReg shiftershifter(Xout1, Xout2, Xout3, Xout4, Xout5, Xout6,
Xout7, Xout8, fastclk, rst,start);


        dF enaD1(dispEna1, nena1, enable1, slowclk, rst);
        dF enaD2(dispEna2, nena2, enable2, slowclk, rst);
        dF enaD3(dispEna3, nena3, enable3, slowclk, rst);
        dF enaD4(dispEna4, nena4, enable4, slowclk, rst);
        dF enaD5(dispEna5, nena5, enable5, slowclk, rst);
        dF enaD6(dispEna6, nena6, enable6, slowclk, rst);
        dF enaD7(dispEna7, nena7, enable7, slowclk, rst);
        dF enaD8(dispEna8, nena8, enable8, slowclk, rst);
        dF enaD9(dispEna9, nena9, enable9, slowclk, rst);
        dF enaD10(dispEna10, nena10, enable10, slowclk, rst);
        dF enaD11(dispEna11, nena11, enable11, slowclk, rst);
        dF enaD12(dispEna12, nena12, enable12, slowclk, rst);
        dF enaD13(dispEna13, nena13, enable13, slowclk, rst);
        dF enaD14(dispEna14, nena14, enable14, slowclk, rst);
        dF enaD15(dispEna15, nena15, enable15, slowclk, rst);


        clkdivider pleasework(slowclk, fastclk, clk, rst);


        Snake trial(qqA, qqB, qqC, upX, upY, enableX, enableY, L, R, dead,
start, slowclk, rst);

    upDown countX(Xa, Xb, Xc, start, slowclk, upX, enableX);
        upDown countY(Ya, Yb, Yc, start, slowclk, upY, enableY);

    foodPosG foodPosG1(fxa, fxb, fxc,fya, fyb, fyc, FxA, FxB, FxC, FyA, FyB,
FyC, start, eat, slowclk, eat);

    eatFood eatfood(eat, Xa, Xb, Xc, FxA, FxB, FxC,
                                                Ya, Yb, Yc, FyA, FyB, FyC,
slowclk, start);
```

```
        up4bit up4bit1(qA, qB, qC, qD, start,eat);

        enableLogic work(outEnable1, outEnable2, outEnable3, outEnable4,
outEnable5, outEnable6, outEnable7,
                                                outEnable8, outEnable9,
outEnable10, outEnable11, outEnable12, outEnable13, outEnable14, outEnable15,
qA, qB, qC, qD);

        buf e0(enable1, outEnable1), e1(enable2, outEnable2), e2(enable3,
outEnable3), e3(enable4, outEnable4), e4(enable5, outEnable5),
        e5(enable6, outEnable6), e6(enable7, outEnable7), e7(enable8,
outEnable8), e8(enable9, outEnable9), e9(enable10, outEnable10),
        e10(enable11, outEnable11), e11(enable12, outEnable12), e12(enable13,
outEnable13), e13(enable14, outEnable14), e14(enable15, outEnable15);

        buf buf0(Xa1, outXa1), buf1(Xb1, outXb1), buf2(Xc1, outXc1), buf3(Ya1,
outYa1), buf4(Yb1, outYb1), buf5(Yc1, outYc1),
       buf6(Xa2, outXa2), buf7(Xb2, outXb2), buf8(Xc2, outXc2), buf9(Ya2,
outYa2), buf10(Yb2, outYb2), buf11(Yc2, outYc2),
                buf12(Xa3, outXa3), buf13(Xb3, outXb3), buf14(Xc3, outXc3),
buf15(Ya3, outYa3), buf16(Yb3, outYb3), buf17(Yc3, outYc3),
                buf18(Xa4, outXa4), buf19(Xb4, outXb4), buf20(Xc4, outXc4),
buf21(Ya4, outYa4), buf22(Yb4, outYb4), buf23(Yc4, outYc4),
                buf24(Xa5, outXa5), buf25(Xb5, outXb5), buf26(Xc5, outXc5),
buf27(Ya5, outYa5), buf28(Yb5, outYb5), buf29(Yc5, outYc5),
                buf30(Xa6, outXa6), buf31(Xb6, outXb6), buf32(Xc6, outXc6),
buf33(Ya6, outYa6), buf34(Yb6, outYb6), buf35(Yc6, outYc6),
                buf36(Xa7, outXa7), buf37(Xb7, outXb7), buf38(Xc7, outXc7),
buf39(Ya7, outYa7), buf40(Yb7, outYb7), buf41(Yc7, outYc7),
                buf42(Xa8, outXa8), buf43(Xb8, outXb8), buf44(Xc8, outXc8),
buf45(Ya8, outYa8), buf46(Yb8, outYb8), buf47(Yc8, outYc8),
                buf48(Xa9, outXa9), buf49(Xb9, outXb9), buf50(Xc9, outXc9),
buf51(Ya9, outYa9), buf52(Yb9, outYb9), buf53(Yc9, outYc9),
                buf54(Xa10, outXa10), buf55(Xb10, outXb10), buf56(Xc10,
outXc10), buf57(Ya10, outYa10), buf58(Yb10, outYb10), buf59(Yc10, outYc10),
                buf60(Xa11, outXa11), buf61(Xb11, outXb11), buf62(Xc11,
outXc11), buf63(Ya11, outYa11), buf64(Yb11, outYb11), buf65(Yc11, outYc11),
                buf66(Xa12, outXa12), buf67(Xb12, outXb12), bug68(Xc12,
outXc12), buf69(Ya12, outYa12), buf70(Yb12, outYb12), buf71(Yc12, outYc12),
                buf72(Xa13, outXa13), buf73(Xb13, outXb13), buf74(Xc13,
outXc13), buf75(Ya13, outYa13), buf76(Yb13, outYb13), buf77(Yc13, outYc13),
                buf78(Xa14, outXa14), buf79(Xb14, outXb14), buf80(Xc14,
outXc14), buf81(Ya14, outYa14), buf82(Yb14, outYb14), buf83(Yc14, outYc14),
                buf84(Xa15, outXa15), buf85(Xb15, outXb15), buf86(Xc15,
outXc15), buf87(Ya15, outYa15), buf88(Yb15, outYb15), buf89(Yc15, outYc15);


 \8_to_1 col1(ay1, by1, ccy1, dy1, eey1, ffy1, ggy1, hhy1,c, b,
a,Y1,output21);
 //8_to_1
col2(pin_name4,pin_name5,pin_name6,pin_name7,pin_name8,pin_name9,pin_name10,p
in_name11,q2,q1,q0,Food1,output2);
 \8_to_1 col3(ay2, by2, ccy2, dy2, eey2, ffy2, ggy2, hhy2,c, b,
a,Y2,output22);
```

```
 //8_to_1
col4(pin_name4,pin_name5,pin_name6,pin_name7,pin_name8,pin_name9,pin_name10,p
in_name11,q2,q1,q0,Food2,output2);
 \8_to_1 col45(ay3, by3, ccy3, dy3, eey3, ffy3, ggy3, hhy3, c, b, a, Y3,
workDamnit);
 //8_to_1
col6(pin_name4,pin_name5,pin_name6,pin_name7,pin_name8,pin_name9,pin_name10,p
in_name11,q2,q1,q0,Food3,output2);
 \8_to_1 col7(ay4, by4, ccy4, dy4, eey4, ffy4, ggy4, hhy4,c, b,
a,Y4,output24);
 //8_to_1
col8(pin_name4,pin_name5,pin_name6,pin_name7,pin_name8,pin_name9,pin_name10,p
in_name11,q2,q1,q0,Food4,output2);
 \8_to_1 col9(ay5, by5, ccy5, dy5, eey5, ffy5, ggy5, hhy5, c, b,
a,Y5,output25);
 //8_to_1
col10(pin_name4,pin_name5,pin_name6,pin_name7,pin_name8,pin_name9,pin_name10,
pin_name11,q2,q1,q0,Food5,output2);
 \8_to_1 col11(ay6, by6, ccy6, dy6, eey6, ffy6, ggy6, hhy6,c, b,
a,Y6,output26);
 //8_to_1
col12(pin_name4,pin_name5,pin_name6,pin_name7,pin_name8,pin_name9,pin_name10,
pin_name11,q2,q1,q0,Food6,output2);
 \8_to_1 col13(ay7, by7, ccy7, dy7, eey7, ffy7, ggy7, hhy7 ,c, b,
a,Y7,output27);
 //8_to_1
col14(pin_name4,pin_name5,pin_name6,pin_name7,pin_name8,pin_name9,pin_name10,
pin_name11,q2,q1,q0,Food7,output2);
 \8_to_1 col15(ay8, by8, ccy8, dy8, eey8, ffy8, ggy8, hhy8,c, b,
a,Y8,output28);
 //8_to_1
col16(pin_name4,pin_name5,pin_name6,pin_name7,pin_name8,pin_name9,pin_name10,
pin_name11,q2,q1,q0,Food8,output2);

        positionCaster mypositionCaster(
                                outXa1,outXb1,outXc1,outYa1,outYb1,outYc1,
                                outXa2,outXb2,outXc2,outYa2,outYb2,outYc2,
                                outXa3,outXb3,outXc3,outYa3,outYb3,outYc3,

        outXa4,outXb4,outXc4,outYa4,outYb4,outYc4,

        outXa5,outXb5,outXc5,outYa5,outYb5,outYc5,

        outXa6,outXb6,outXc6,outYa6,outYb6,outYc6,

        outXa7,outXb7,outXc7,outYa7,outYb7,outYc7,

        outXa8,outXb8,outXc8,outYa8,outYb8,outYc8,

        outXa9,outXb9,outXc9,outYa9,outYb9,outYc9,

        outXa10,outXb10,outXc10,outYa10,outYb10,outYc10,

        outXa11,outXb11,outXc11,outYa11,outYb11,outYc11,

        outXa12,outXb12,outXc12,outYa12,outYb12,outYc12,
```

```
        outXa13,outXb13,outXc13,outYa13,outYb13,outYc13,

        outXa14,outXb14,outXc14,outYa14,outYb14,outYc14,

outXa15,outXb15,outXc15,outYa15,outYb15,outYc15,
                                                slowclk,start,
Xa,Xb,Xc,Ya,Yb,Yc,

        enable1,enable2,enable3,enable4,enable5,enable6,enable7,enable8,enable
9,enable10,

        enable11,enable12,enable13,enable14,enable15);

        ifHitSelf ifHitSelf1(dead,Xa,Xb,Xc,Ya,Yb,Yc,
                                        Xa1,Xb1,Xc1,Ya1,Yb1,Yc1,
                                        Xa2,Xb2,Xc2,Ya2,Yb2,Yc2,
                                        Xa3,Xb3,Xc3,Ya3,Yb3,Yc3,
                                                        Xa4,Xb4,Xc4,Ya4,Yb4,Yc4,
                                                        Xa5,Xb5,Xc5,Ya5,Yb5,Yc5,
                                                        Xa6,Xb6,Xc6,Ya6,Yb6,Yc6,
                                                        Xa7,Xb7,Xc7,Ya7,Yb7,Yc7,
                                                        Xa8,Xb8,Xc8,Ya8,Yb8,Yc8,
                                                        Xa9,Xb9,Xc9,Ya9,Yb9,Yc9,

        Xa10,Xb10,Xc10,Ya10,Yb10,Yc10,

        Xa11,Xb11,Xc11,Ya11,Yb11,Yc11,

        Xa12,Xb12,Xc12,Ya12,Yb12,Yc12,

        Xa13,Xb13,Xc13,Ya13,Yb13,Yc13,

        Xa14,Xb14,Xc14,Ya14,Yb14,Yc14,

        Xa15,Xb15,Xc15,Ya15,Yb15,Yc15,

        enable1,enable2,enable3,enable4,enable5,enable6,enable7,enable8,

        enable9,enable10,enable11,enable12,enable13,enable14,enable15);



  foodScan scanFood(Food1, Food2, Food3, Food4, Food5, Food6, Food7, Food8,
a, b, c, FyA, FyB, FyC, FxA, FxB, FxC);



  upDown up(a, b, c, rst, fastclk, 1, 1);


//  scanner scan(Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8,
//                              ay1, ay2, ay3, ay4, ay5, ay6, ay7, ay8,
//                                      by1, by2, by3, by4, by5, by6, by7, by8,
//                                              cy1, cy2, cy3, cy4, cy5, cy6,
cy7, cy8,
```

```
//                                                    dy1, dy2, dy3, dy4, dy5,
dy6, dy7, dy8,
//                                                           eey1, eey2,
eey3, eey4, eey5, eey6, eey7, eey8,
//                                                                  ffy1,
ffy2, ffy3, ffy4, ffy5, ffy6, ffy7, ffy8,
//
        ggy1, ggy2, ggy3, ggy4, ggy5, ggy6, ggy7, ggy8,
//                                                                                hhy1,
//
//


  condenseDisplay condenseDisplay1(  ay1,  ay2,  ay3,  ay4,  ay5,  ay6,  ay7,
ay8,




        Xa,Xb,Xc,Ya,Yb,Yc,

        Xa1,Xb1,Xc1,Ya1,Yb1,Yc1,

        Xa2,Xb2,Xc2,Ya2,Yb2,Yc2,

        Xa3,Xb3,Xc3,Ya3,Yb3,Yc3,

        Xa4,Xb4,Xc4,Ya4,Yb4,Yc4,

        Xa5,Xb5,Xc5,Ya5,Yb5,Yc5,

        Xa6,Xb6,Xc6,Ya6,Yb6,Yc6,

        Xa7,Xb7,Xc7,Ya7,Yb7,Yc7,

        Xa8,Xb8,Xc8,Ya8,Yb8,Yc8,

        Xa9,Xb9,Xc9,Ya9,Yb9,Yc9,

        Xa10,Xb10,Xc10,Ya10,Yb10,Yc10,

        Xa11,Xb11,Xc11,Ya11,Yb11,Yc11,

        Xa12,Xb12,Xc12,Ya12,Yb12,Yc12,

        Xa13,Xb13,Xc13,Ya13,Yb13,Yc13,

        Xa14,Xb14,Xc14,Ya14,Yb14,Yc14,

        Xa15,Xb15,Xc15,Ya15,Yb15,Yc15,

        dispEna1,dispEna2,dispEna3,dispEna4,dispEna5,dispEna6,dispEna7,dispEna
8,
```

```
        dispEna9,dispEna10,dispEna11,dispEna12, dispEna13,dispEna14,dispEna15,
dead);

endmodule


module \8_to_1 (
        pin_name4,
        pin_name5,
        pin_name6,
        pin_name7,
        pin_name8,
        pin_name9,
        pin_name10,
        pin_name11,
        q2,
        q1,
        q0,
        output1,
        output2
);


input wire      pin_name4;
input wire      pin_name5;
input wire      pin_name6;
input wire      pin_name7;
input wire      pin_name8;
input wire      pin_name9;
input wire      pin_name10;
input wire      pin_name11;
input wire      q2;
input wire      q1;
input wire      q0;
output wire     output1;
output wire     output2;

wire    SYNTHESIZED_WIRE_0;

assign  SYNTHESIZED_WIRE_0 = 0;



\81mux  b2v_inst(
        .C(q0),
        .D1(pin_name5),
        .D0(pin_name4),
        .D5(pin_name9),
        .D4(pin_name8),
        .D2(pin_name6),
        .D3(pin_name7),
        .GN(SYNTHESIZED_WIRE_0),
        .D7(pin_name11),
        .D6(pin_name10),
        .A(q2),
```

```
        .B(q1),
        .Y(output1),
        .WN(output2));



endmodule


module foodDisplay(foodDispY1,foodDispY2, foodDispY3,foodDispY4, foodDispY5,
foodDispY6, foodDispY7, foodDispY8,
 a,b,c, fy1, fy2, fy3, fy4,fy5, fy6, fy7, fy8, fxa, fxb, fxc);

        input a,b,c, fy1, fy2, fy3, fy4,fy5, fy6, fy7, fy8, fxa, fxb, fxc;

        output foodDispY1,foodDispY2, foodDispY3,foodDispY4, foodDispY5,
foodDispY6, foodDispY7, foodDispY8;



        xnor checkSameA(samePosA, a, fxa), checkSameB(samePosB, b, fxb),
checkSameC(samePosC, c, fxc);

        and foodDisp1(foodDispY1, samePosA,samePosB, samePosC, fy1),
        foodDisp2(foodDispY2, samePosA,samePosB, samePosC, fy2),
        foodDisp3(foodDispY3, samePosA,samePosB, samePosC,fy3),
        foodDisp4(foodDispY4, samePosA,samePosB, samePosC, fy4),
        foodDisp5(foodDispY5, samePosA,samePosB, samePosC, fy5),
        foodDisp6(foodDispY6, samePosA,samePosB, samePosC,fy6),
        foodDisp7(foodDispY7,samePosA,samePosB, samePosC, fy7),
        foodDisp8(foodDispY8, samePosA,samePosB, samePosC, fy8);

        /*and foodDisp1(foodDispY1, a,b,c,fxa,fxb,fxc, fy1),
        foodDisp2(foodDispY2, a,b,c,fxa,fxb,fxc, fy2),
        foodDisp3(foodDispY3, a,b,c,fxa,fxb,fxc, fy3),
        foodDisp4(foodDispY4, a,b,c,fxa,fxb,fxc, fy4),
        foodDisp5(foodDispY5, a,b,c,fxa,fxb,fxc, fy5),
        foodDisp6(foodDispY6, a,b,c,fxa,fxb,fxc, fy6),
        foodDisp7(foodDispY7, a,b,c,fxa,fxb,fxc, fy7),
        foodDisp8(foodDispY8, a,b,c,fxa,fxb,fxc, fy8);*/

endmodule

module foodScan(Food1, Food2, Food3, Food4, Food5, Food6, Food7,
Food8,a,b,c,fya, fyb, fyc, fxa, fxb, fxc);

        input a,b,c, fya, fyb, fyc, fxa, fxb, fxc;

        output Food1, Food2, Food3, Food4, Food5, Food6, Food7, Food8;

        convertToSerialY gogo(fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8, fya,
fyb, fyc, 1);


        foodDisplay dispA(Food1,Food2, Food3,Food4, Food5, Food6, Food7,
Food8,a,b,c, fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8, fxa, fxb, fxc);
```

```
        /*or foodYscan1(Food1, A1, B1, C1, D1, E1, F1, G1, H1),
                foodYscan2(Food2, A2, B2, C2, D2, E2, F2, G2, H2),
                foodYscan3(Food3, A3, B3, C3, D3, E3, F3, G3, H3),
                foodYscan4(Food4, A4, B4, C4, D4, E4, F4, G4, H4),
                foodYscan5(Food5, A5, B5, C5, D5, E5, F5, G5, H5),
                foodYscan6(Food6, A6, B6, C6, D6, E6, F6, G6, H6),
                foodYscan7(Food7, A7, B7, C7, D7, E7, F7, G7, H7),
                foodYscan8(Food8, A8, B8, C8, D8, E8, F8, G8, H8);*/

        //when abc == fxafxbfxc, fyafybfyc should be powered


        /*
 \8_to_1 col1(ay1, by1, cy1, dy1, eey1, ffy1, ggy1, hhy1,a, b,
c,Food1,outputF21);
 //8_to_1
col2(pin_name4,pin_name5,pin_name6,pin_name7,pin_name8,pin_name9,pin_name10,p
in_name11,q2,q1,q0,Food1,output2);
 \8_to_1 col3(ay2, by2, cy2, dy2, eey2, ffy2, ggy2, hhy2,a, b,
c,Food2,outputF22);
 //8_to_1
col4(pin_name4,pin_name5,pin_name6,pin_name7,pin_name8,pin_name9,pin_name10,p
in_name11,q2,q1,q0,Food2,output2);
 \8_to_1 col5(ay3, by3, cy3, fy3, eey3, ffy3, ggy3,
hhy3,a,b,c,Food3,outputF23);
 //8_to_1
col6(pin_name4,pin_name5,pin_name6,pin_name7,pin_name8,pin_name9,pin_name10,p
in_name11,q2,q1,q0,Food3,output2);
 \8_to_1 col7(ay4, by4, cy4, dy4, eey4, ffy4, ggy4, hhy4,a, b,
c,Food4,outputF24);
 //8_to_1
col8(pin_name4,pin_name5,pin_name6,pin_name7,pin_name8,pin_name9,pin_name10,p
in_name11,q2,q1,q0,Food4,output2);
 \8_to_1 col9(ay5, by5, cy5, dy5, eey5, ffy5, ggy5, hhy5, a, b,
c,Food5,outputF25);
 //8_to_1
col10(pin_name4,pin_name5,pin_name6,pin_name7,pin_name8,pin_name9,pin_name10,
pin_name11,q2,q1,q0,Food5,output2);
 \8_to_1 col11(ay6, by6, cy6, dy6, eey6, ffy6, ggy6, hhy6,a, b,
c,Food6,outputF26);
 //8_to_1
col12(pin_name4,pin_name5,pin_name6,pin_name7,pin_name8,pin_name9,pin_name10,
pin_name11,q2,q1,q0,Food6,output2);
 \8_to_1 col13(ay7, by7, cy7, dy7, eey5, ffy6, ggy7, hhy7 ,a, b,
c,Food7,outputF27);
 //8_to_1
col14(pin_name4,pin_name5,pin_name6,pin_name7,pin_name8,pin_name9,pin_name10,
pin_name11,q2,q1,q0,Food7,output2);
 \8_to_1 col15(ay8, by8, cy8, dy8, eey8, ffy8, ggy8, hhy8,a, b,
c,Food8,outputF28);
 //8_to_1
col16(pin_name4,pin_name5,pin_name6,pin_name7,pin_name8,pin_name9,pin_name10,
pin_name11,q2,q1,q0,Food8,output2);
        */

endmodule
```

```
////////////////////////////////////////////////////////////////////////////
/
////////////////////////////////////////////////////////////////////////////
/


module condenseDisplay(ay1, ay2, ay3, ay4, ay5, ay6, ay7, ay8,
                                                    by1, by2, by3, by4,
by5, by6, by7, by8,
                                                    ccy1,  ccy2,  ccy3,
ccy4, ccy5,  ccy6,  ccy7,  ccy8,
                                                    dy1, dy2, dy3, dy4,
dy5, dy6, dy7, dy8,
                                                    eey1, eey2, eey3,
eey4, eey5, eey6, eey7, eey8,
                                                    ffy1, ffy2, ffy3,
ffy4, ffy5, ffy6, ffy7, ffy8,
                                                    ggy1, ggy2, ggy3,
ggy4, ggy5, ggy6, ggy7, ggy8,
                                                    hhy1, hhy2, hhy3,
hhy4, hhy5, hhy6, hhy7, hhy8,

                                                   Xa,Xb,Xc,Ya,Yb,Yc,

                                        Xa1,Xb1,Xc1,Ya1,Yb1,Yc1,
                                        Xa2,Xb2,Xc2,Ya2,Yb2,Yc2,
                                        Xa3,Xb3,Xc3,Ya3,Yb3,Yc3,
                                        Xa4,Xb4,Xc4,Ya4,Yb4,Yc4,
                                        Xa5,Xb5,Xc5,Ya5,Yb5,Yc5,
                                        Xa6,Xb6,Xc6,Ya6,Yb6,Yc6,
                                        Xa7,Xb7,Xc7,Ya7,Yb7,Yc7,
                                        Xa8,Xb8,Xc8,Ya8,Yb8,Yc8,
                                        Xa9,Xb9,Xc9,Ya9,Yb9,Yc9,
                                        Xa10,Xb10,Xc10,Ya10,Yb10,Yc10,
                                        Xa11,Xb11,Xc11,Ya11,Yb11,Yc11,
                                        Xa12,Xb12,Xc12,Ya12,Yb12,Yc12,
                                        Xa13,Xb13,Xc13,Ya13,Yb13,Yc13,
                                        Xa14,Xb14,Xc14,Ya14,Yb14,Yc14,
                                        Xa15,Xb15,Xc15,Ya15,Yb15,Yc15,

                                        enable1,enable2,enable3,enable4,
                                        enable5,enable6,enable7,enable8,
                                        enable9,enable10,enable11,enable12,
                                        enable13,enable14,enable15, dead);

        input Xa,Xb,Xc,Ya,Yb,Yc,
        Xa1,Xb1,Xc1,Ya1,Yb1,Yc1,
        Xa2,Xb2,Xc2,Ya2,Yb2,Yc2,
        Xa3,Xb3,Xc3,Ya3,Yb3,Yc3,
        Xa4,Xb4,Xc4,Ya4,Yb4,Yc4,
        Xa5,Xb5,Xc5,Ya5,Yb5,Yc5,
        Xa6,Xb6,Xc6,Ya6,Yb6,Yc6,
        Xa7,Xb7,Xc7,Ya7,Yb7,Yc7,
        Xa8,Xb8,Xc8,Ya8,Yb8,Yc8,
        Xa9,Xb9,Xc9,Ya9,Yb9,Yc9,
        Xa10,Xb10,Xc10,Ya10,Yb10,Yc10,
```

```
        Xa11,Xb11,Xc11,Ya11,Yb11,Yc11,
        Xa12,Xb12,Xc12,Ya12,Yb12,Yc12,
        Xa13,Xb13,Xc13,Ya13,Yb13,Yc13,
        Xa14,Xb14,Xc14,Ya14,Yb14,Yc14,
        Xa15,Xb15,Xc15,Ya15,Yb15,Yc15,

                                        enable1,enable2,enable3,enable4,

        enable5,enable6,enable7,enable8,enable9,enable10,enable11,enable12,
                                        enable13,enable14,enable15, dead;

        output ay1, ay2, ay3, ay4, ay5, ay6, ay7, ay8,by1, by2, by3, by4, by5,
by6, by7, by8,ccy1,  ccy2,  ccy3,  ccy4, ccy5,  ccy6,  ccy7,  ccy8,
        dy1, dy2, dy3, dy4, dy5, dy6, dy7, dy8, eey1, eey2, eey3, eey4, eey5,
eey6, eey7, eey8, ffy1, ffy2, ffy3, ffy4, ffy5, ffy6, ffy7, ffy8,
ggy1, ggy2, ggy3, ggy4, ggy5, ggy6, ggy7, ggy8, hhy1, hhy2, hhy3, hhy4, hhy5,
hhy6, hhy7, hhy8;

        convertToSerialX convertHeadX(ex1, ex2, ex3, ex4, ex5, ex6, ex7, ex8,
Xa, Xb, Xc, ~dead);
        convertToSerialY convertHeadY(ey1, ey2, ey3, ey4, ey5, ey6, ey7, ey8,
Ya, Yb, Yc, ~dead);

        convertToSerialX convertHeadX1(fx1, fx2, fx3, fx4, fx5, fx6, fx7, fx8,
Xa1, Xb1, Xc1, enable1);
        convertToSerialY convertHeadY1(fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8,
Ya1, Yb1, Yc1, enable1);

        convertToSerialX convertHeadX2(gx1, gx2, gx3, gx4, gx5, gx6, gx7, gx8,
Xa2, Xb2, Xc2, enable2);
        convertToSerialY convertHeadY2(gy1, gy2, gy3, gy4, gy5, gy6, gy7, gy8,
Ya2, Yb2, Yc2, enable2);

        convertToSerialX convertHeadX3(hx1, hx2, hx3, hx4, hx5, hx6, hx7, hx8,
Xa3, Xb3, Xc3, enable3);
        convertToSerialY convertHeadY3(hy1, hy2, hy3, hy4, hy5, hy6, hy7, hy8,
Ya3, Yb3, Yc3, enable3);

        convertToSerialX convertHeadX4(ix1, ix2, ix3, ix4, ix5, ix6, ix7, ix8,
Xa4, Xb4, Xc4, enable4);
        convertToSerialY convertHeadY4(iy1, iy2, iy3, iy4, iy5, iy6, iy7, iy8,
Ya4, Yb4, Yc4, enable4);

        convertToSerialX convertHeadX5(jx1, jx2, jx3, jx4, jx5, jx6, jx7, jx8,
Xa5, Xb5, Xc5, enable5);
        convertToSerialY convertHeadY5(jy1, jy2, jy3, jy4, jy5, jy6, jy7, jy8,
Ya5, Yb5, Yc5, enable5);

        convertToSerialX convertHeadX6(kx1, kx2, kx3, kx4, kx5, kx6, kx7, kx8,
Xa6, Xb6, Xc6, enable6);
        convertToSerialY convertHeadY6(ky1, ky2, ky3, ky4, ky5, ky6, ky7, ky8,
Ya6, Yb6, Yc6, enable6);

        convertToSerialX convertHeadX7(lx1, lx2, lx3, lx4, lx5, lx6, lx7, lx8,
Xa7, Xb7, Xc7, enable7);
        convertToSerialY convertHeadY7(ly1, ly2, ly3, ly4, ly5, ly6, ly7, ly8,
Ya7, Yb7, Yc7, enable7);
```

```
        convertToSerialX convertHeadX8(mx1, mx2, mx3, mx4, mx5, mx6, mx7, mx8,
Xa8, Xb8, Xc8, enable8);
        convertToSerialY convertHeadY8(my1, my2, my3, my4, my5, my6, my7, my8,
Ya8, Yb8, Yc8, enable8);

        convertToSerialX convertHeadX9(nx1, nx2, nx3, nx4, nx5, nx6, nx7, nx8,
Xa9, Xb9, Xc9, enable9);
        convertToSerialY convertHeadY9(ny1, ny2, ny3, ny4, ny5, ny6, ny7, ny8,
Ya9, Yb9, Yc9, enable9);

        convertToSerialX convertHeadX10(ox1, ox2, ox3, ox4, ox5, ox6, ox7,
ox8, Xa10, Xb10, Xc10, enable10);
        convertToSerialY convertHeadY10(oy1, oy2, oy3, oy4, oy5, oy6, oy7,
oy8, Ya10, Yb10, Yc10, enable10);

        convertToSerialX convertHeadX11(px1, px2, px3, px4, px5, px6, px7,
px8, Xa11, Xb11, Xc11, enable11);
        convertToSerialY convertHeadY11(py1, py2, py3, py4, py5, py6, py7,
py8, Ya11, Yb11, Yc11, enable11);

        convertToSerialX convertHeadX12(qx1, qx2, qx3, qx4, qx5, qx6, qx7,
qx8, Xa12, Xb12, Xc12, enable12);
        convertToSerialY convertHeadY12(qy1, qy2, qy3, qy4, qy5, qy6, qy7,
qy8, Ya12, Yb12, Yc12, enable12);

        convertToSerialX convertHeadX13(rx1, rx2, rx3, rx4, rx5, rx6, rx7,
rx8, Xa13, Xb13, Xc13, enable13);
        convertToSerialY convertHeadY13(ry1, ry2, ry3, ry4, ry5, ry6, ry7,
ry8, Ya13, Yb13, Yc13, enable13);

        convertToSerialX convertHeadX14(sx1, sx2, sx3, sx4, sx5, sx6, sx7,
sx8, Xa14, Xb14, Xc14, enable14);
        convertToSerialY convertHeadY14(sy1, sy2, sy3, sy4, sy5, sy6, sy7,
sy8, Ya14, Yb14, Yc14, enable14);

        convertToSerialX convertHeadX15(tx1, tx2, tx3, tx4, tx5, tx6, tx7,
tx8, Xa15, Xb15, Xc15, enable15);
        convertToSerialY convertHeadY15(ty1, ty2, ty3, ty4, ty5, ty6, ty7,
ty8, Ya15, Yb15, Yc15, enable15);

        getYCode getYA(ay1, ay2, ay3, ay4, ay5, ay6, ay7, ay8,
        ex1,fx1,gx1,hx1,ix1,jx1,kx1,lx1,mx1,nx1,ox1,px1,qx1,rx1,sx1,tx1,
           ey1, ey2, ey3, ey4, ey5, ey6, ey7, ey8,
                fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8,
                gy1, gy2, gy3, gy4, gy5, gy6, gy7, gy8,
                hy1, hy2, hy3, hy4, hy5, hy6, hy7, hy8,
                iy1, iy2, iy3, iy4, iy5, iy6, iy7, iy8,
                jy1, jy2, jy3, jy4, jy5, jy6, jy7, jy8,
                ky1, ky2, ky3, ky4, ky5, ky6, ky7, ky8,
                ly1, ly2, ly3, ly4, ly5, ly6, ly7, ly8,
                my1, my2, my3, my4, my5, my6, my7, my8,
                ny1, ny2, ny3, ny4, ny5, ny6, ny7, ny8,
                oy1, oy2, oy3, oy4, oy5, oy6, oy7, oy8,
                py1, py2, py3, py4, py5, py6, py7, py8,
                qy1, qy2, qy3, qy4, qy5, qy6, qy7, qy8,
                ry1, ry2, ry3, ry4, ry5, ry6, ry7, ry8,
```

```
        sy1, sy2, sy3, sy4, sy5, sy6, sy7, sy8,
        ty1, ty2, ty3, ty4, ty5, ty6, ty7, ty8);

getYCode getYB(by1, by2, by3, by4, by5, by6, by7, by8,
ex2,fx2,gx2,hx2,ix2,jx2,kx2,lx2,mx2,nx2,ox2,px2,qx2,rx2,sx2,tx2,
   ey1, ey2, ey3, ey4, ey5, ey6, ey7, ey8,
        fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8,
        gy1, gy2, gy3, gy4, gy5, gy6, gy7, gy8,
        hy1, hy2, hy3, hy4, hy5, hy6, hy7, hy8,
        iy1, iy2, iy3, iy4, iy5, iy6, iy7, iy8,
        jy1, jy2, jy3, jy4, jy5, jy6, jy7, jy8,
        ky1, ky2, ky3, ky4, ky5, ky6, ky7, ky8,
        ly1, ly2, ly3, ly4, ly5, ly6, ly7, ly8,
        my1, my2, my3, my4, my5, my6, my7, my8,
        ny1, ny2, ny3, ny4, ny5, ny6, ny7, ny8,
        oy1, oy2, oy3, oy4, oy5, oy6, oy7, oy8,
        py1, py2, py3, py4, py5, py6, py7, py8,
        qy1, qy2, qy3, qy4, qy5, qy6, qy7, qy8,
        ry1, ry2, ry3, ry4, ry5, ry6, ry7, ry8,
        sy1, sy2, sy3, sy4, sy5, sy6, sy7, sy8,
        ty1, ty2, ty3, ty4, ty5, ty6, ty7, ty8);

getYCode getYC(ccy1,  ccy2,  ccy3,  ccy4, ccy5,  ccy6,  ccy7,  ccy8,
   ex3,fx3,gx3,hx3,ix3,jx3,kx3,lx3,mx3,nx3,ox3,px3,qx3,rx3,sx3,tx3,
   ey1, ey2, ey3, ey4, ey5, ey6, ey7, ey8,
        fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8,
        gy1, gy2, gy3, gy4, gy5, gy6, gy7, gy8,
        hy1, hy2, hy3, hy4, hy5, hy6, hy7, hy8,
        iy1, iy2, iy3, iy4, iy5, iy6, iy7, iy8,
        jy1, jy2, jy3, jy4, jy5, jy6, jy7, jy8,
        ky1, ky2, ky3, ky4, ky5, ky6, ky7, ky8,
        ly1, ly2, ly3, ly4, ly5, ly6, ly7, ly8,
        my1, my2, my3, my4, my5, my6, my7, my8,
        ny1, ny2, ny3, ny4, ny5, ny6, ny7, ny8,
        oy1, oy2, oy3, oy4, oy5, oy6, oy7, oy8,
        py1, py2, py3, py4, py5, py6, py7, py8,
        qy1, qy2, qy3, qy4, qy5, qy6, qy7, qy8,
        ry1, ry2, ry3, ry4, ry5, ry6, ry7, ry8,
        sy1, sy2, sy3, sy4, sy5, sy6, sy7, sy8,
        ty1, ty2, ty3, ty4, ty5, ty6, ty7, ty8);

getYCode getYD(dy1, dy2, dy3, dy4, dy5, dy6, dy7, dy8,
   ex4,fx4,gx4,hx4,ix4,jx4,kx4,lx4,mx4,nx4,ox4,px4,qx4,rx4,sx4,tx4,
        ey1, ey2, ey3, ey4, ey5, ey6, ey7, ey8,
        fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8,
        gy1, gy2, gy3, gy4, gy5, gy6, gy7, gy8,
        hy1, hy2, hy3, hy4, hy5, hy6, hy7, hy8,
        iy1, iy2, iy3, iy4, iy5, iy6, iy7, iy8,
        jy1, jy2, jy3, jy4, jy5, jy6, jy7, jy8,
        ky1, ky2, ky3, ky4, ky5, ky6, ky7, ky8,
        ly1, ly2, ly3, ly4, ly5, ly6, ly7, ly8,
        my1, my2, my3, my4, my5, my6, my7, my8,
        ny1, ny2, ny3, ny4, ny5, ny6, ny7, ny8,
        oy1, oy2, oy3, oy4, oy5, oy6, oy7, oy8,
        py1, py2, py3, py4, py5, py6, py7, py8,
        qy1, qy2, qy3, qy4, qy5, qy6, qy7, qy8,
        ry1, ry2, ry3, ry4, ry5, ry6, ry7, ry8,
```

```
        sy1, sy2, sy3, sy4, sy5, sy6, sy7, sy8,
        ty1, ty2, ty3, ty4, ty5, ty6, ty7, ty8);

getYCode getYE(eey1, eey2, eey3, eey4, eey5, eey6, eey7, eey8,
ex5,fx5,gx5,hx5,ix5,jx5,kx5,lx5,mx5,nx5,ox5,px5,qx5,rx5,sx5,tx5,
   ey1, ey2, ey3, ey4, ey5, ey6, ey7, ey8,
   fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8,
        gy1, gy2, gy3, gy4, gy5, gy6, gy7, gy8,
        hy1, hy2, hy3, hy4, hy5, hy6, hy7, hy8,
        iy1, iy2, iy3, iy4, iy5, iy6, iy7, iy8,
        jy1, jy2, jy3, jy4, jy5, jy6, jy7, jy8,
        ky1, ky2, ky3, ky4, ky5, ky6, ky7, ky8,
        ly1, ly2, ly3, ly4, ly5, ly6, ly7, ly8,
        my1, my2, my3, my4, my5, my6, my7, my8,
        ny1, ny2, ny3, ny4, ny5, ny6, ny7, ny8,
        oy1, oy2, oy3, oy4, oy5, oy6, oy7, oy8,
        py1, py2, py3, py4, py5, py6, py7, py8,
        qy1, qy2, qy3, qy4, qy5, qy6, qy7, qy8,
        ry1, ry2, ry3, ry4, ry5, ry6, ry7, ry8,
        sy1, sy2, sy3, sy4, sy5, sy6, sy7, sy8,
        ty1, ty2, ty3, ty4, ty5, ty6, ty7, ty8);

getYCode getYF(ffy1, ffy2, ffy3, ffy4, ffy5, ffy6, ffy7, ffy8,
ex6,fx6,gx6,hx6,ix6,jx6,kx6,lx6,mx6,nx6,ox6,px6,qx6,rx6,sx6,tx6,
        ey1, ey2, ey3, ey4, ey5, ey6, ey7, ey8,
        fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8,
        gy1, gy2, gy3, gy4, gy5, gy6, gy7, gy8,
        hy1, hy2, hy3, hy4, hy5, hy6, hy7, hy8,
        iy1, iy2, iy3, iy4, iy5, iy6, iy7, iy8,
        jy1, jy2, jy3, jy4, jy5, jy6, jy7, jy8,
        ky1, ky2, ky3, ky4, ky5, ky6, ky7, ky8,
        ly1, ly2, ly3, ly4, ly5, ly6, ly7, ly8,
        my1, my2, my3, my4, my5, my6, my7, my8,
        ny1, ny2, ny3, ny4, ny5, ny6, ny7, ny8,
        oy1, oy2, oy3, oy4, oy5, oy6, oy7, oy8,
        py1, py2, py3, py4, py5, py6, py7, py8,
        qy1, qy2, qy3, qy4, qy5, qy6, qy7, qy8,
        ry1, ry2, ry3, ry4, ry5, ry6, ry7, ry8,
        sy1, sy2, sy3, sy4, sy5, sy6, sy7, sy8,
        ty1, ty2, ty3, ty4, ty5, ty6, ty7, ty8);

getYCode getYG(ggy1, ggy2, ggy3, ggy4, ggy5, ggy6, ggy7, ggy8,
ex7,fx7,gx7,hx7,ix7,jx7,kx7,lx7,mx7,nx7,ox7,px7,qx7,rx7,sx7,tx7,
        ey1, ey2, ey3, ey4, ey5, ey6, ey7, ey8,
        fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8,
        gy1, gy2, gy3, gy4, gy5, gy6, gy7, gy8,
        hy1, hy2, hy3, hy4, hy5, hy6, hy7, hy8,
        iy1, iy2, iy3, iy4, iy5, iy6, iy7, iy8,
        jy1, jy2, jy3, jy4, jy5, jy6, jy7, jy8,
        ky1, ky2, ky3, ky4, ky5, ky6, ky7, ky8,
        ly1, ly2, ly3, ly4, ly5, ly6, ly7, ly8,
        my1, my2, my3, my4, my5, my6, my7, my8,
        ny1, ny2, ny3, ny4, ny5, ny6, ny7, ny8,
        oy1, oy2, oy3, oy4, oy5, oy6, oy7, oy8,
        py1, py2, py3, py4, py5, py6, py7, py8,
        qy1, qy2, qy3, qy4, qy5, qy6, qy7, qy8,
        ry1, ry2, ry3, ry4, ry5, ry6, ry7, ry8,
```

```
                sy1, sy2, sy3, sy4, sy5, sy6, sy7, sy8,
                ty1, ty2, ty3, ty4, ty5, ty6, ty7, ty8);

        getYCode getYH(hhy1, hhy2, hhy3, hhy4, hhy5, hhy6, hhy7, hhy8,
        ex8,fx8,gx8,hx8,ix8,jx8,kx8,lx8,mx8,nx8,ox8,px8,qx8,rx8,sx8,tx8,
           ey1, ey2, ey3, ey4, ey5, ey6, ey7, ey8,
                fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8,
                gy1, gy2, gy3, gy4, gy5, gy6, gy7, gy8,
                hy1, hy2, hy3, hy4, hy5, hy6, hy7, hy8,
                iy1, iy2, iy3, iy4, iy5, iy6, iy7, iy8,
                jy1, jy2, jy3, jy4, jy5, jy6, jy7, jy8,
                ky1, ky2, ky3, ky4, ky5, ky6, ky7, ky8,
                ly1, ly2, ly3, ly4, ly5, ly6, ly7, ly8,
                my1, my2, my3, my4, my5, my6, my7, my8,
                ny1, ny2, ny3, ny4, ny5, ny6, ny7, ny8,
                oy1, oy2, oy3, oy4, oy5, oy6, oy7, oy8,
                py1, py2, py3, py4, py5, py6, py7, py8,
                qy1, qy2, qy3, qy4, qy5, qy6, qy7, qy8,
                ry1, ry2, ry3, ry4, ry5, ry6, ry7, ry8,
                sy1, sy2, sy3, sy4, sy5, sy6, sy7, sy8,
                ty1, ty2, ty3, ty4, ty5, ty6, ty7, ty8);


endmodule



module getYCode(y1, y2, y3, y4, y5, y6, y7,
y8,ine,inf,ing,inh,ini,inj,ink,inl,inm,inn,ino,inp,inq,inr,ins,intt,
                ey1, ey2, ey3, ey4, ey5, ey6, ey7, ey8, fy1, fy2, fy3, fy4,
fy5, fy6, fy7, fy8,
                gy1, gy2, gy3, gy4, gy5, gy6, gy7, gy8, hy1, hy2, hy3, hy4,
hy5, hy6, hy7, hy8, iy1, iy2, iy3, iy4, iy5, iy6, iy7, iy8,
                jy1, jy2, jy3, jy4, jy5, jy6, jy7, jy8, ky1, ky2, ky3, ky4,
ky5, ky6, ky7, ky8, ly1, ly2, ly3, ly4, ly5, ly6, ly7, ly8,
                my1, my2, my3, my4, my5, my6, my7, my8, ny1, ny2, ny3, ny4,
ny5, ny6, ny7, ny8, oy1, oy2, oy3, oy4, oy5, oy6, oy7, oy8,
                py1, py2, py3, py4, py5, py6, py7, py8, qy1, qy2, qy3, qy4,
qy5, qy6, qy7, qy8, ry1, ry2, ry3, ry4, ry5, ry6, ry7, ry8,
                sy1, sy2, sy3, sy4, sy5, sy6, sy7, sy8, ty1, ty2, ty3, ty4,
ty5, ty6, ty7, ty8);

        input
ine,inf,ing,inh,ini,inj,ink,inl,inm,inn,ino,inp,inq,inr,ins,intt, ey1, ey2,
ey3, ey4, ey5, ey6, ey7, ey8, fy1, fy2, fy3, fy4, fy5, fy6, fy7, fy8,
                gy1, gy2, gy3, gy4, gy5, gy6, gy7, gy8, hy1, hy2, hy3, hy4,
hy5, hy6, hy7, hy8, iy1, iy2, iy3, iy4, iy5, iy6, iy7, iy8,
                jy1, jy2, jy3, jy4, jy5, jy6, jy7, jy8, ky1, ky2, ky3, ky4,
ky5, ky6, ky7, ky8, ly1, ly2, ly3, ly4, ly5, ly6, ly7, ly8,
                my1, my2, my3, my4, my5, my6, my7, my8, ny1, ny2, ny3, ny4,
ny5, ny6, ny7, ny8, oy1, oy2, oy3, oy4, oy5, oy6, oy7, oy8,
                py1, py2, py3, py4, py5, py6, py7, py8, qy1, qy2, qy3, qy4,
qy5, qy6, qy7, qy8, ry1, ry2, ry3, ry4, ry5, ry6, ry7, ry8,
                sy1, sy2, sy3, sy4, sy5, sy6, sy7, sy8, ty1, ty2, ty3, ty4,
ty5, ty6, ty7, ty8;

        output y1, y2, y3, y4, y5, y6, y7, y8;
```

```
                buf b0(e, ine), b1(f, inf), b2(g, ing), b3(h, inh), b4(i,
ini), b5(j, inj), b6(k, ink), b7(l, inl), b8(m, inm),
                b9(n, inn), b10(o, ino), b11(p, inp), b12(q, inq), b13(r,
inr), b14(s, ins), b15(t, intt);

                createY wy1(y1,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t, ey1,  fy1,
gy1,  hy1,  iy1,  jy1,  ky1,  ly1,  my1,  ny1,  oy1,  py1,  qy1,  ry1,  sy1,
ty1);
                createY wy2(y2,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t, ey2,  fy2,
gy2,  hy2,  iy2,  jy2,  ky2,  ly2,  my2,  ny2,  oy2,  py2,  qy2,  ry2,  sy2,
ty2);
                createY wy3(y3,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t, ey3,  fy3,
gy3,  hy3,  iy3,  jy3,  ky3,  ly3,  my3,  ny3,  oy3,  py3,  qy3,  ry3,  sy3,
ty3);
                createY wy4(y4,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t, ey4,  fy4,
gy4,  hy4,  iy4,  jy4,  ky4,  ly4,  my4,  ny4,  oy4,  py4,  qy4,  ry4,  sy4,
ty4);
                createY wy5(y5,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t, ey5,  fy5,
gy5,  hy5,  iy5,  jy5,  ky5,  ly5,  my5,  ny5,  oy5,  py5,  qy5,  ry5,  sy5,
ty5);
                createY wy6(y6,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t, ey6,  fy6,
gy6,  hy6,  iy6,  jy6,  ky6,  ly6,  my6,  ny6,  oy6,  py6,  qy6,  ry6,  sy6,
ty6);
                createY wy7(y7,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t, ey7,  fy7,
gy7,  hy7,  iy7,  jy7,  ky7,  ly7,  my7,  ny7,  oy7,  py7,  qy7,  ry7,  sy7,
ty7);
                createY wy8(y8,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t, ey8,  fy8,
gy8,  hy8,  iy8,  jy8,  ky8,  ly8,  my8,  ny8,  oy8,  py8,  qy8,  ry8,  sy8,
ty8);



endmodule

module createY(y,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,ey, fy, gy, hy, iy, jy, ky,
ly, my, ny, oy, py, qy, ry, sy, ty);

        input e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,ey, fy, gy, hy, iy, jy, ky, ly,
my, ny, oy, py, qy, ry, sy, ty;

        output y;

        and e0(qe, e, ey), f0(qf, f, fy), g0(qg,g,gy), h0(qh,h,hy), i0(qi, i,
iy),
                j0(qj, j , jy), k0(qk, k, ky), l0(ql, l, ly), m0(qm, m, my),
n0(qn, n, ny),
                o0(qo, o, oy), p0(qp, p, py), q0(qq, q, qy), r0(qr, r, ry),
s0(qs, s, sy), t0(qt, t, ty);

        or getY(y, qe, qf, qg, qh, qi, qj, qk, ql, qm, qn, qo, qp, qq, qr, qs,
qt);


endmodule

module convertToSerialY(g1, g2, g3, g4, g5, g6, g7, g8, a, b, c, enable);
```

```
        input a, b, c, enable;
        output g1, g2, g3, g4, g5, g6, g7, g8;

        and aG1(g1, enable, ~a, ~b, ~c);
        //g2
        and aG2(g2, enable, ~a, ~b, c);
        //g3
        and aG3(g3, enable, ~a, b, ~c);
        //g4
        and aG4(g4, enable, ~a, b, c);
        //g5
        and aG5(g5, enable, a, ~b, ~c);
        //g6
        and aG6(g6, enable, a, ~b, c);
        //g7
        and aG7(g7, enable, a, b, ~c);
        //g8
        and aG8(g8, enable, a, b, c);
endmodule

module convertToSerialX(g1, g2, g3, g4, g5, g6, g7, g8, a, b, c, enable);
        input a, b, c, enable;
        output g1, g2, g3, g4, g5, g6, g7, g8;

        and aG1(g1, enable, ~a, ~b, ~c);
        //g2
        and aG2(g2, enable, ~a, ~b, c);
        //g3
        and aG3(g3, enable, ~a, b, ~c);
        //g4
        and aG4(g4, enable, ~a, b, c);
        //g5
        and aG5(g5, enable, a, ~b, ~c);
        //g6
        and aG6(g6, enable, a, ~b, c);
        //g7
        and aG7(g7, enable, a, b, ~c);
        //g8
        and aG8(g8, enable, a, b, c);
endmodule


module scanner(Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8, ay1, ay2, ay3, ay4, ay5, ay6,
ay7, ay8,by1, by2, by3, by4, by5, by6, by7, by8,cy1, cy2, cy3, cy4, cy5, cy6,
cy7, cy8,
        dy1, dy2, dy3, dy4, dy5, dy6, dy7, dy8, eey1, eey2, eey3, eey4, eey5,
eey6, eey7, eey8, ffy1, ffy2, ffy3, ffy4, ffy5, ffy6, ffy7, ffy8,
ggy1, ggy2, ggy3, ggy4, ggy5, ggy6, ggy7, ggy8, hhy1, hhy2, hhy3, hhy4, hhy5,
hhy6, hhy7, hhy8, clk, rst, A, B, C);


        input  ay1, ay2, ay3, ay4, ay5, ay6, ay7, ay8,by1, by2, by3, by4, by5,
by6, by7, by8,cy1, cy2, cy3, cy4, cy5, cy6, cy7, cy8,
        dy1, dy2, dy3, dy4, dy5, dy6, dy7, dy8, eey1, eey2, eey3, eey4, eey5,
eey6, eey7, eey8, ffy1, ffy2, ffy3, ffy4, ffy5, ffy6, ffy7, ffy8,
ggy1, ggy2, ggy3, ggy4, ggy5, ggy6, ggy7, ggy8, hhy1, hhy2, hhy3, hhy4, hhy5,
hhy6, hhy7, hhy8, clk, rst, A, B, C;
```

```
        output Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8;


        and a0(arnoldy0,ay1, ~A, ~B, ~C), a1(arnoldy1,by1, ~A, ~B, C),
a2(arnoldy2,cy1, ~A, B, ~C), a3(arnoldy3,dy1, ~A, B, C), a4(arnoldy4,eey1, A,
~B, ~C), a5(arnoldy5,ffy1, A, ~B, C),
                        a6(arnoldy6,ggy1, A, B, ~C), a7(arnoldy7,hhy1, A, B,
C);

        and b0(barney0,ay2, ~A, ~B, ~C), b1(barney1,by2, ~A, ~B, C),
b2(barney2,cy2, ~A, B, ~C), b3(barney3,dy2, ~A, B, C), b4(barney4,eey2, A,
~B, ~C), b5(barney5,ffy2, A, ~B, C),
                        b6(barney6,ggy2, A, B, ~C), b7(barney7,hhy2, A, B, C);

        and c0(carry0,ay3, ~A, ~B, ~C), c1(carry1,by3, ~A, ~B, C),
c2(carry2,cy3, ~A, B, ~C), c3(carry3,dy3, ~A, B, C), c4(carry4,eey3, A, ~B,
~C), c5(carry5,ffy3, A, ~B, C),
                        c6(carry6,ggy3, A, B, ~C), c7(carry7, hhy3,A, B, C);

        and d0(davey0,ay4, ~A, ~B, ~C), d1(davey1,by4, ~A, ~B, C), d2(davey2,
cy4,~A, B, ~C), d3(davey3,dy4, ~A, B, C), d4(davey4,eey4, A, ~B, ~C),
d5(davey5,ffy4, A, ~B, C),
                        d6(davey6,ggy4, A, B, ~C), d7(davey7,hhy4, A, B, C);

        and e0(eevee0,ay5, ~A, ~B, ~C), e1(eevee1,by5, ~A, ~B, C),
e2(eevee2,cy5, ~A, B, ~C), e3(eevee3,dy5, ~A, B, C), e4(eevee4,eey5, A, ~B,
~C), e5(eevee5,ffy5, A, ~B, C),
                        e6(eevee6,ggy5, A, B, ~C), e7(eevee7,hhy5, A, B, C);

        and f0(froggy0,ay6, ~A, ~B, ~C), f1(froggy1,by6, ~A, ~B, C),
f2(froggy2,cy6, ~A, B, ~C), f3(froggy3, dy6,~A, B, C), f4(froggy4,eey6, A,
~B, ~C), f5(froggy5,ffy6, A, ~B, C),
                        f6(froggy6,ggy6, A, B, ~C), f7(froggy7, hhy6,A, B, C);

        and g0(gilly0,ay7, ~A, ~B, ~C), g1(gilly1,by7, ~A, ~B, C),
g2(gilly2,cy7, ~A, B, ~C), g3(gilly3,dy7, ~A, B, C), g4(gilly4,eey7, A, ~B,
~C), g5(gilly5, ffy7,A, ~B, C),
                        g6(gilly6,ggy7, A, B, ~C), g7(gilly7, hhy7,A, B, C);

        and h0(harry0,ay8, ~A, ~B, ~C), h1(harry1, by8,~A, ~B, C),
h2(harry2,cy8, ~A, B, ~C), h3(harry3,dy8, ~A, B, C), h4(harry4, eey8,A, ~B,
~C), h5(harry5,ffy8, A, ~B, C),
                        h6(harry6,ggy8, A, B, ~C), h7(harry7,hhy8,A, B, C);

        nor doY1(Y1, arnoldy0, arnoldy1, arnoldy2, arnoldy3, arnoldy4,
arnoldy5, arnoldy6, arnoldy7),
        doY2(Y2,  barney0, barney1, barney2, barney3, barney4, barney5,
barney6, barney7),
        doY3(Y3,  carry0, carry1, carry2, carry3, carry4, carry5, carry6,
carry7),
        doY4(Y4,  davey0, davey1, davey2, davey3, davey4, davey5, davey6,
davey7),
        doY5(Y5,  eevee0, eevee1, eevee2, eevee3, eevee4, eevee5, eevee6,
eevee7),
        doY6(Y6, froggy0, froggy1, froggy2, froggy3, froggy4, froggy5,
froggy6, froggy7),
```

```
        doY7(Y7,  gilly0, gilly1, gilly2, gilly3, gilly4, gilly5, gilly6,
gilly7),
        doY8(Y8, harry0, harry1, harry2, harry3, harry4, harry5, harry6,
harry7);

endmodule


module dFforFood(q, qBar, D, clk);
  input D, clk;
  output q, qBar;

  reg q;

  not n1 (qBar, q);

  always@ (posedge clk)
  begin
      q = D;

  end

endmodule

module foodPosG(fxa, fxb, fxc,fya, fyb, fyc, fxA, fxB, fxC,fyA, fyB,
fyC,start,rst,clk,eat);
    input rst,clk,eat,start;
    output fxA, fxB, fxC,fyA, fyB, fyC, fxa, fxb, fxc,fya, fyb, fyc;

    or or1(enable,start,eat);
  foodPos foodPos1(fxa, fxb, fxc,fya, fyb, fyc,rst,clk);

         dFforFood dF1(fxA, qBarA, fxa, enable);
         dFforFood dF2(fxB, qBarB, fxb, enable);
         dFforFood dF3(fxC, qBarC, fxc, enable);
         dFforFood dF4(fyA, qBarD, fya, enable);
         dFforFood dF5(fyB, qBarE, fyb, enable);
         dFforFood dF6(fyC, qBarF, fyc, enable);

endmodule



module foodPos(fxA, fxB, fxC,fyA, fyB, fyC,rst,clk);

  input rst,clk;
  output fxA, fxB, fxC,fyA, fyB, fyC;

  and and1(all1s,fxA,fxB,fxC,fyA,fyB);
  xor xor1(fyCxoralll1s,all1s,fyC);
  xnor xnor1(fxAxnorfyC,fxA,fyCxoralll1s);


  dF dF1(fxA, qBarA, fyCxoralll1s, clk, rst);
  dF dF2(fxB, qBarB, fxAxnorfyC, clk, rst);
  dF dF3(fxC, qBarC, fxB, clk, rst);
  dF dF4(fyA, qBarD, fxC, clk, rst);
```

```
  dF dF5(fyB, qBarE, fyA, clk, rst);
  dF dF6(fyC, qBarF, fyB, clk, rst);

endmodule



module dFEnabled(q, qBar, D, clk, rst, enable);
  input D, clk, rst, enable;
  output q, qBar;

  reg q;

  not n1 (qBar, q);

  always@ (posedge rst or posedge clk)
  begin
    if(rst)
      q = 0;
       else if(~enable)
              q = q;
    else
      q = D;

  end

endmodule

module eatFood(eat, xA, xB, xC, FxA, FxB, FxC, yA, yB, yC,
                              FyA, FyB, FyC,clk,rst);

        input xA, xB, xC, FxA, FxB, FxC, yA, yB, yC,
                              FyA, FyB, FyC,rst,clk;

        output eat;

        //compare x positions
        xnor xnor1(xResult1, FxA, xA);
        xnor xnor2(xResult2, FxB, xB);
        xnor xnor3(xResult3, FxC, xC);

        and and1(xOut, xResult1, xResult2, xResult3);

        //compare y postions
        xnor xnor4(yResult1, FyA, yA);
        xnor xnor5(yResult2, FyB, yB);
        xnor xnor6(yResult3, FyC, yC);

        and and2(yOut, yResult1, yResult2, yResult3);
        //output eat '1' True '0' False
        and and3(samepos, xOut, yOut);

  and (almosteat,samepos,qBarEat);

  dF dfforeat(eat,qBarEat,almosteat,clk,rst);

endmodule
```

```
module up4bit(qA, qB, qC, qD, rst, clk);
        output qA, qB, qC, qD;
        input rst, clk;

        dF Aff(qA, qBarA, inA, ~clk, rst);
        dF Bff(qB, qBarB, inB, ~clk, rst);
        dF Cff(qC, qBarC, inC, ~clk, rst);
        dF Dff(qD, qBarD, inD, ~clk, rst);

        and and0(ABCD, qA, qB, qC, qD);
        //D flipFlop
        or or0(inD, qBarD, ABCD);
        //C flipflop
        xor Cin1(cin1, qC, qD);
        or or1(inC, cin1, ABCD);
        //B flipflop
        and and1(Bin1,qBarB, qC, qD), and2(Bin2, qB, qBarC), and3(Bin3, qB,
qBarD);
        or or2(inB, Bin1, Bin2, ABCD, Bin3);
        //A flipflop
        and andA1(BCD, qB, qC, qD);
        or or3(inA, BCD, qA);

endmodule

module enableLogic(enable1, enable2, enable3, enable4, enable5, enable6,
enable7,
enable8, enable9, enable10, enable11, enable12, enable13, enable14, enable15,
A, B, C, D);

  input  A, B, C, D;


  output enable1, enable2, enable3, enable4, enable5, enable6, enable7,
enable8, enable9, enable10, enable11, enable12, enable13, enable14, enable15;


//enable1
 or eno1(enable1, A, B, C, D);
//enable2
or eno2(enable2, A, B, C);
//enable3
and ena0(t0, C, D);
or eno3(enable3, t0, A, B);
//enable4
or eno4(enable4, A, B);
//enable5
and ena1(t1, C, B), ena2(t2, B, D);
or eno5(enable5, t1, t2, A);
//enable6
and ena3(t3, B, C);
or eno6(enable6, A, t3);
//enable7
and ena4(t4, B, C, D);
or eno7(enable7, A, t4);
//enable8
```

```
assign enable8 = A;
//enable9
or eno9(t5, B, C, D);
and ena5(enable9, t5, A);
//enable10
or eno10(t14, B, C);
and ena10(enable10, A, t14);
//enable11
and ena6(t6, A, B), ena333(t7, A, C, D);
or eno11(enable11, t6, t7);
//enable12
and ena7(enable12, A, B);
//enable13
or eno13(t13, C, D);
and ena13(enable13, A, B, t13);
//enable14
and ena8(enable14, A, B, C);
//enable15
and ena9(enable15, A, B, C, D);

endmodule

module delayFF(q, qBar, D, clk,rst, enable);
  input D, clk, enable,rst;
  output q, qBar;

  reg q;

  not n1 (qBar, q);

  always@ (posedge clk)
  begin
  if(rst)
      q = 0;
       else if(enable)
      q = D;
    else
      q = q;

  end

endmodule


module positionReg(delayedX0, delayedX1, delayedX2, x0, x1, x2, clk,rst,
enable);
        output delayedX0, delayedX1, delayedX2;
        input x0, x1, x2, clk, rst,enable;

        delayFF delayx0(delayedX0, qBar0, x0, clk,rst, enable);
        delayFF delayx1(delayedX1, qBar1, x1, clk,rst, enable);
        delayFF delayx2(delayedX2, qBar2, x2, clk,rst, enable);

endmodule

module positionCaster(
                              Xa1,Xb1,Xc1,Ya1,Yb1,Yc1,
```

```
                                Xa2,Xb2,Xc2,Ya2,Yb2,Yc2,
                                Xa3,Xb3,Xc3,Ya3,Yb3,Yc3,
                                        Xa4,Xb4,Xc4,Ya4,Yb4,Yc4,
                                        Xa5,Xb5,Xc5,Ya5,Yb5,Yc5,
                                        Xa6,Xb6,Xc6,Ya6,Yb6,Yc6,
                                        Xa7,Xb7,Xc7,Ya7,Yb7,Yc7,
                                        Xa8,Xb8,Xc8,Ya8,Yb8,Yc8,
                                        Xa9,Xb9,Xc9,Ya9,Yb9,Yc9,
                                        Xa10,Xb10,Xc10,Ya10,Yb10,Yc10,
                                        Xa11,Xb11,Xc11,Ya11,Yb11,Yc11,
                                        Xa12,Xb12,Xc12,Ya12,Yb12,Yc12,
                                        Xa13,Xb13,Xc13,Ya13,Yb13,Yc13,
                                        Xa14,Xb14,Xc14,Ya14,Yb14,Yc14,

        Xa15,Xb15,Xc15,Ya15,Yb15,Yc15,inclk,inrst,
                                                        Xa,Xb,Xc,Ya,Yb,Yc,

        enable1,enable2,enable3,enable4,enable5,enable6,enable7,enable8,enable
9,enable10,

        enable11,enable12,enable13,enable14,enable15);

   input Xa,Xb,Xc,Ya,Yb,Yc,inclk,inrst,

enable1,enable2,enable3,enable4,enable5,enable6,enable7,enable8,enable9,enabl
e10,
                enable11,enable12,enable13,enable14,enable15;

   output   Xa1,Xb1,Xc1,Ya1,Yb1,Yc1,
                        Xa2,Xb2,Xc2,Ya2,Yb2,Yc2,
                        Xa3,Xb3,Xc3,Ya3,Yb3,Yc3,
                        Xa4,Xb4,Xc4,Ya4,Yb4,Yc4,
                        Xa5,Xb5,Xc5,Ya5,Yb5,Yc5,
                        Xa6,Xb6,Xc6,Ya6,Yb6,Yc6,
                        Xa7,Xb7,Xc7,Ya7,Yb7,Yc7,
                        Xa8,Xb8,Xc8,Ya8,Yb8,Yc8,
                        Xa9,Xb9,Xc9,Ya9,Yb9,Yc9,
                        Xa10,Xb10,Xc10,Ya10,Yb10,Yc10,
                        Xa11,Xb11,Xc11,Ya11,Yb11,Yc11,
               Xa12,Xb12,Xc12,Ya12,Yb12,Yc12,
                        Xa13,Xb13,Xc13,Ya13,Yb13,Yc13,
                        Xa14,Xb14,Xc14,Ya14,Yb14,Yc14,
                        Xa15,Xb15,Xc15,Ya15,Yb15,Yc15;

                buf b1(clk, inclk), b2(rst, inrst);

         positionReg
positionForX1(Xa1,Xb1,Xc1,Xa,Xb,Xc,clk,rst,enable1),positionForY1(Ya1,Yb1,Yc1
,Ya,Yb,Yc,clk,rst,enable1);
         positionReg
positionForX2(Xa2,Xb2,Xc2,Xa1,Xb1,Xc1,clk,rst,enable2),positionForY2(Ya2,Yb2,
Yc2,Ya1,Yb1,Yc1,clk,rst,enable2);
         positionReg
positionForX3(Xa3,Xb3,Xc3,Xa2,Xb2,Xc2,clk,rst,enable3),positionForY3(Ya3,Yb3,
Yc3,Ya2,Yb2,Yc2,clk,rst,enable3);
```

```
         positionReg
positionForX4(Xa4,Xb4,Xc4,Xa3,Xb3,Xc3,clk,rst,enable4),positionForY4(Ya4,Yb4,
Yc4,Ya3,Yb3,Yc3,clk,rst,enable4);
         positionReg
positionForX5(Xa5,Xb5,Xc5,Xa4,Xb4,Xc4,clk,rst,enable5),positionForY5(Ya5,Yb5,
Yc5,Ya4,Yb4,Yc4,clk,rst,enable5);
         positionReg
positionForX6(Xa6,Xb6,Xc6,Xa5,Xb5,Xc5,clk,rst,enable6),positionForY6(Ya6,Yb6,
Yc6,Ya5,Yb5,Yc5,clk,rst,enable6);
         positionReg
positionForX7(Xa7,Xb7,Xc7,Xa6,Xb6,Xc6,clk,rst,enable7),positionForY7(Ya7,Yb7,
Yc7,Ya6,Yb6,Yc6,clk,rst,enable7);
         positionReg
positionForX8(Xa8,Xb8,Xc8,Xa7,Xb7,Xc7,clk,rst,enable8),positionForY8(Ya8,Yb8,
Yc8,Ya7,Yb7,Yc7,clk,rst,enable8);
         positionReg
positionForX9(Xa9,Xb9,Xc9,Xa8,Xb8,Xc8,clk,rst,enable9),positionForY9(Ya9,Yb9,
Yc9,Ya8,Yb8,Yc8,clk,rst,enable9);
         positionReg
positionForX10(Xa10,Xb10,Xc10,Xa9,Xb9,Xc9,clk,rst,enable10),positionForY10(Ya
10,Yb10,Yc10,Ya9,Yb9,Yc9,clk,rst,enable10);
         positionReg
positionForX11(Xa11,Xb11,Xc11,Xa10,Xb10,Xc10,clk,rst,enable11),positionForY11
(Ya11,Yb11,Yc11,Ya10,Yb10,Yc10,clk,rst,enable11);
         positionReg
positionForX12(Xa12,Xb12,Xc12,Xa11,Xb11,Xc11,clk,rst,enable12),positionForY12
(Ya12,Yb12,Yc12,Ya11,Yb11,Yc11,clk,rst,enable12);
         positionReg
positionForX13(Xa13,Xb13,Xc13,Xa12,Xb12,Xc12,clk,rst,enable13),positionForY13
(Ya13,Yb13,Yc13,Ya12,Yb12,Yc12,clk,rst,enable13);
         positionReg
positionForX14(Xa14,Xb14,Xc14,Xa13,Xb13,Xc13,clk,rst,enable14),positionForY14
(Ya14,Yb14,Yc14,Ya13,Yb13,Yc13,clk,rst,enable14);
         positionReg
positionForX15(Xa15,Xb15,Xc15,Xa14,Xb14,Xc14,clk,rst,enable15),positionForY15
(Ya15,Yb15,Yc15,Ya14,Yb14,Yc14,clk,rst,enable15);

endmodule




module
aXnor(XaXa1,XbXb1,XcXc1,YaYa1,YbYb1,YcYc1,Xa,Xb,Xc,Ya,Yb,Yc,Xa1,Xb1,Xc1,Ya1,Y
b1,Yc1);

  input Xa,Xb,Xc,Ya,Yb,Yc,Xa1,Xb1,Xc1,Ya1,Yb1,Yc1;
  output XaXa1,XbXb1,XcXc1,YaYa1,YbYb1,YcYc1;

        xnor xnor1(XaXa1,Xa,Xa1);
        xnor xnor2(XbXb1,Xb,Xb1);
        xnor xnor3(XcXc1,Xc,Xc1);
        xnor xnor4(YaYa1,Ya,Ya1);
        xnor xnor5(YbYb1,Yb,Yb1);
        xnor xnor6(YcYc1,Yc,Yc1);
```

```
endmodule

module ifHitSelf(dead,inXa,inXb,inXc,inYa,inYb,inYc,
                                        Xa1,Xb1,Xc1,Ya1,Yb1,Yc1,
                                        Xa2,Xb2,Xc2,Ya2,Yb2,Yc2,
                                        Xa3,Xb3,Xc3,Ya3,Yb3,Yc3,
                 Xa4,Xb4,Xc4,Ya4,Yb4,Yc4,
                 Xa5,Xb5,Xc5,Ya5,Yb5,Yc5,
                 Xa6,Xb6,Xc6,Ya6,Yb6,Yc6,
                 Xa7,Xb7,Xc7,Ya7,Yb7,Yc7,
                 Xa8,Xb8,Xc8,Ya8,Yb8,Yc8,
                 Xa9,Xb9,Xc9,Ya9,Yb9,Yc9,
                 Xa10,Xb10,Xc10,Ya10,Yb10,Yc10,
                 Xa11,Xb11,Xc11,Ya11,Yb11,Yc11,
                 Xa12,Xb12,Xc12,Ya12,Yb12,Yc12,
                 Xa13,Xb13,Xc13,Ya13,Yb13,Yc13,
                 Xa14,Xb14,Xc14,Ya14,Yb14,Yc14,
                 Xa15,Xb15,Xc15,Ya15,Yb15,Yc15,
                 enable1,enable2,enable3,enable4,
                 enable5,enable6,enable7,enable8,enable9,enable10,

enable11,enable12,enable13,enable14,enable15
                 );

  input inXa,inXb,inXc,inYa,inYb,inYc,
        Xa1,Xb1,Xc1,Ya1,Yb1,Yc1,
                Xa2,Xb2,Xc2,Ya2,Yb2,Yc2,
                Xa3,Xb3,Xc3,Ya3,Yb3,Yc3,
                Xa4,Xb4,Xc4,Ya4,Yb4,Yc4,
                Xa5,Xb5,Xc5,Ya5,Yb5,Yc5,
                Xa6,Xb6,Xc6,Ya6,Yb6,Yc6,
                Xa7,Xb7,Xc7,Ya7,Yb7,Yc7,
                Xa8,Xb8,Xc8,Ya8,Yb8,Yc8,
                Xa9,Xb9,Xc9,Ya9,Yb9,Yc9,
                Xa10,Xb10,Xc10,Ya10,Yb10,Yc10,
                Xa11,Xb11,Xc11,Ya11,Yb11,Yc11,
                Xa12,Xb12,Xc12,Ya12,Yb12,Yc12,
                Xa13,Xb13,Xc13,Ya13,Yb13,Yc13,
                Xa14,Xb14,Xc14,Ya14,Yb14,Yc14,
                Xa15,Xb15,Xc15,Ya15,Yb15,Yc15,
      enable1,enable2,enable3,enable4,
      enable5,enable6,enable7,enable8,enable9,enable10,
                enable11,enable12,enable13,enable14,enable15;

  output dead;

  buf b0(Xa, inXa), b1(Xb, inXb), b2(Xc, inXc), b3(Ya, inYa), b4(Yb, inYb),
b5(Yc, inYc);

  aXnor
Xnor1(XaXa1,XbXb1,XcXc1,YaYa1,YbYb1,YcYc1,Xa,Xb,Xc,Ya,Yb,Yc,Xa1,Xb1,Xc1,Ya1,Y
b1,Yc1);
  aXnor
Xnor2(XaXa2,XbXb2,XcXc2,YaYa2,YbYb2,YcYc2,Xa,Xb,Xc,Ya,Yb,Yc,Xa2,Xb2,Xc2,Ya2,Y
b2,Yc2);
```

```
  aXnor
Xnor3(XaXa3,XbXb3,XcXc3,YaYa3,YbYb3,YcYc3,Xa,Xb,Xc,Ya,Yb,Yc,Xa3,Xb3,Xc3,Ya3,Y
b3,Yc3);
  aXnor
Xnor4(XaXa4,XbXb4,XcXc4,YaYa4,YbYb4,YcYc4,Xa,Xb,Xc,Ya,Yb,Yc,Xa4,Xb4,Xc4,Ya4,Y
b4,Yc4);
  aXnor
Xnor5(XaXa5,XbXb5,XcXc5,YaYa5,YbYb5,YcYc5,Xa,Xb,Xc,Ya,Yb,Yc,Xa5,Xb5,Xc5,Ya5,Y
b5,Yc5);
  aXnor
Xnor6(XaXa6,XbXb6,XcXc6,YaYa6,YbYb6,YcYc6,Xa,Xb,Xc,Ya,Yb,Yc,Xa6,Xb6,Xc6,Ya6,Y
b6,Yc6);
  aXnor
Xnor7(XaXa7,XbXb7,XcXc7,YaYa7,YbYb7,YcYc7,Xa,Xb,Xc,Ya,Yb,Yc,Xa7,Xb7,Xc7,Ya7,Y
b7,Yc7);
  aXnor
Xnor8(XaXa8,XbXb8,XcXc8,YaYa8,YbYb8,YcYc8,Xa,Xb,Xc,Ya,Yb,Yc,Xa8,Xb8,Xc8,Ya8,Y
b8,Yc8);
  aXnor
Xnor9(XaXa9,XbXb9,XcXc9,YaYa9,YbYb9,YcYc9,Xa,Xb,Xc,Ya,Yb,Yc,Xa9,Xb9,Xc9,Ya9,Y
b9,Yc9);
  aXnor
Xnor10(XaXa10,XbXb10,XcXc10,YaYa10,YbYb10,YcYc10,Xa,Xb,Xc,Ya,Yb,Yc,Xa10,Xb10,
Xc10,Ya10,Yb10,Yc10);
  aXnor
Xnor11(XaXa11,XbXb11,XcXc11,YaYa11,YbYb11,YcYc11,Xa,Xb,Xc,Ya,Yb,Yc,Xa11,Xb11,
Xc11,Ya11,Yb11,Yc11);
  aXnor
Xnor12(XaXa12,XbXb12,XcXc12,YaYa12,YbYb12,YcYc12,Xa,Xb,Xc,Ya,Yb,Yc,Xa12,Xb12,
Xc12,Ya12,Yb12,Yc12);
  aXnor
Xnor13(XaXa13,XbXb13,XcXc13,YaYa13,YbYb13,YcYc13,Xa,Xb,Xc,Ya,Yb,Yc,Xa13,Xb13,
Xc13,Ya13,Yb13,Yc13);
  aXnor
Xnor14(XaXa14,XbXb14,XcXc14,YaYa14,YbYb14,YcYc14,Xa,Xb,Xc,Ya,Yb,Yc,Xa14,Xb14,
Xc14,Ya14,Yb14,Yc14);
  aXnor
Xnor15(XaXa15,XbXb15,XcXc15,YaYa15,YbYb15,YcYc15,Xa,Xb,Xc,Ya,Yb,Yc,Xa15,Xb15,
Xc15,Ya15,Yb15,Yc15);

  and and1(hita1,XaXa1,XbXb1,XcXc1,YaYa1,YbYb1,YcYc1,enable1);
  and and2(hita2,XaXa2,XbXb2,XcXc2,YaYa2,YbYb2,YcYc2,enable2);
  and and3(hita3,XaXa3,XbXb3,XcXc3,YaYa3,YbYb3,YcYc3,enable3);
  and and4(hita4,XaXa4,XbXb4,XcXc4,YaYa4,YbYb4,YcYc4,enable4);
  and and5(hita5,XaXa5,XbXb5,XcXc5,YaYa5,YbYb5,YcYc5,enable5);
  and and6(hita6,XaXa6,XbXb6,XcXc6,YaYa6,YbYb6,YcYc6,enable6);
  and and7(hita7,XaXa7,XbXb7,XcXc7,YaYa7,YbYb7,YcYc7,enable7);
  and and8(hita8,XaXa8,XbXb8,XcXc8,YaYa8,YbYb8,YcYc8,enable8);
  and and9(hita9,XaXa9,XbXb9,XcXc9,YaYa9,YbYb9,YcYc9,enable9);
  and and10(hita10,XaXa10,XbXb10,XcXc10,YaYa10,YbYb10,YcYc10,enable10);
  and and11(hita11,XaXa11,XbXb11,XcXc11,YaYa11,YbYb11,YcYc11,enable11);
  and and12(hita12,XaXa12,XbXb12,XcXc12,YaYa12,YbYb12,YcYc12,enable12);
  and and13(hita13,XaXa13,XbXb13,XcXc13,YaYa13,YbYb13,YcYc13,enable13);
  and and14(hita14,XaXa14,XbXb14,XcXc14,YaYa14,YbYb14,YcYc14,enable14);
  and and15(hita15,XaXa15,XbXb15,XcXc15,YaYa15,YbYb15,YcYc15,enable15);
```

```
  or
gameEnd(dead,hita1,hita2,hita3,hita4,hita5,hita6,hita7,hita8,hita9,hita10,hit
a11,hita12,hita13,hita14,hita15);

endmodule



module Snake(qA, qB, qC, upX,upY,enableX,enableY,L,R,Die,start, clk, rst);
  input L,R,Die,start, clk, rst;
  output qA, qB, qC, upX,upY,enableX,enableY;


  dF A(qA, qBarA, snakeA, clk, rst);
  dF B(qB, qBarB, snakeB, clk, rst);
  dF C(qC, qBarC, snakeC, clk, rst);

  //A flip flop
  and andforA1(orA1, qBarA,qBarB, qC, R,~Die),
          andforA2(orA2, qBarA,qB, qC, L,~Die),
     andforA3(orA3,qA,qBarB,qBarC,~L,~R,~Die);
  or  orforA(snakeA,orA1,orA2,orA3);

  //B flip flop
  and andforB1(orB1, qBarA,qBarB, qC, L,~Die),
          andforB2(orB2, qBarA,qB, qBarC, L,~Die),
          andforB3(orB3, qBarA,qB, qC, R,~Die),
   andforB4(orB4,qA,qBarB, qBarC, R,~Die),
   andforB5(orB5, qBarA,qB,qBarC, ~L, ~R, ~Die),
   andforB6(orB6, qBarA,qB,qC,~L, ~R, ~Die);

  or orforB(snakeB, orB1, orB2,orB3,orB4, orB5, orB6);

  //C flip flop
 and andforC1(orC1, qBarA,qBarB, qBarC, start),
          andforC2(orC2, qBarA,qB, qBarC, R),
          andforC3(orC3, qBarA,qB, qBarC, L),
          andforC4(orC4, qA,qBarB, qBarC, L),
          andforC5(orC5, qA,qBarB, qBarC, R),
     andforC6(orC6, qBarA,qBarB,qC, ~L, ~R, ~Die),
     andforC7(orC7, qBarA,qB,qC, ~L, ~R, ~Die);
  or orforC(snakeC,orC1,orC2,orC3,orC4,orC5,orC6,orC7);

  //enableX
  and enXa(enX1, qBarA, qB, qBarC), enXa0(enX2, qA, qBarB, qBarC);
  or enableTheX(enableX, enX1, enX2);
  //enableY
  and enYa(enY1, qBarA, qBarB, qC), enYa0(enY2, qBarA, qB, qC);
  or enableTheY(enableY, enY1, enY2);
  //UpX
  assign upX = qA;
  //UpY
  assign upY= qBarB;

endmodule

module clkdivider(slowclk, fastclk,clk,rst);
```

```
  output fastclk, slowclk;
  input clk;
  input rst;

  reg [24:0]  tBase;


  always@(posedge clk) tBase <= tBase + 1;

  dF FF1(fastclk, n1q0, tBase[1],clk, rst);//12MHz
  dF FF0(slowclk, nq0, tBase[21],clk, rst);//5.96Hz


endmodule

module dF(q, qBar, D, clk, rst);
  input D, clk, rst;
  output q, qBar;

  reg q;

  not n1 (qBar, q);

  always@ (posedge rst or posedge clk)
  begin
    if(rst)
      q = 0;

    else
      q = D;

  end

endmodule

module dFShift(q, qBar, D, loader, clk, rst, load);
  input D, clk, rst, loader, load;
  output q, qBar;

  reg q;

  not n1 (qBar, q);

  always@ (posedge clk)
  begin
    if(rst)
      q = 1;

        else if(load)
          q = loader;

    else
      q = D;

  end
```

```
endmodule


module shiftReg(qA, qB, qC,qD, qE, qF, qG, qH, clk, rst, load);

        input clk, rst, load;
        output qA, qB, qC, qD, qE, qF, qG, qH;

        dFShift shiftA(qA, qBarA, qH, 0, clk, rst, load);
        dFShift shiftB(qB, qBarB, qA, 1, clk, rst, load);
        dFShift shiftC(qC, qBarC, qB, 1, clk, rst, load);
        dFShift shiftD(qD, qBarD, qC, 1, clk, rst, load);
        dFShift shiftE(qE, qBarE, qD, 1, clk, rst, load);
        dFShift shiftF(qF, qBarF, qE, 1, clk, rst, load);
        dFShift shiftG(qG, qBarG, qF, 1, clk, rst, load);
        dFShift shiftH(qH, qBarH, qG, 1, clk, rst, load);

endmodule



module upDown(qA, qB, qC, rst, clk,Up,enable);

        output qA, qB, qC;
        input rst, clk, Up, enable;

        dFEnabled Aff(qA, qBarA, inA, clk, rst, enable);
        dFEnabled Bff(qB, qBarB, inB, clk, rst, enable);
        dFEnabled Cff(qC, qBarC, inC, clk, rst, enable);

        //Aff
  and inA0(inUpA1,qBarA, qB, qC), inA1(inUpA2, qA, qBarC), inA2(inUpA3, qA,
qBarB);
  or inAUp(UpInA, inUpA1, inUpA2, inUpA3);
  and anA(DownAt0, qA, qB), an0A(DownAt1, qA, qC), an1A(DownAt2, qBarA,
qBarB, qBarC);
        or oA(DownInA, DownAt0, DownAt1, DownAt2);
  and an2A(CountUp,UpInA,Up), an3A(CountDown, DownInA, ~Up);
  or goA(inA, CountUp, CountDown);
        //Bff
        xor intoB(DownInB, qBarB, qC);
  xor intoBUp(UpInB, qB, qC);
  and inB0(BUp, UpInB, Up), inB1(BDown, DownInB, ~Up);
  or goB(inB, BUp, BDown);
        //Cff
        assign inC = qBarC;

endmodule
```

Verilog Code of food counter in the Gal chip

```
module up4bit(qA, qB, qC, qD, rst, clk);
```

```
        output qA, qB, qC, qD;
        input rst, clk;


        dF Aff(qA, qBarA, inA, clk, rst);
        dF Bff(qB, qBarB, inB, clk, rst);
        dF Cff(qC, qBarC, inC, clk, rst);
        dF Dff(qD, qBarD, inD, clk, rst);

        and and0(ABCD, qA, qB, qC, qD);
        //D flipFlop
        or or0(inD, qBarD, ABCD);
        //C flipflop
        xor Cin1(cin1, qC, qD);
        or or1(inC, cin1, ABCD);
        //B flipflop
        and and1(Bin1,qBarB, qC, qD), and2(Bin2, qB, qBarC), and3(Bin3, qB, qBarD);
        or or2(inB, Bin1, Bin2, ABCD, Bin3);
        //A flipflop
        and andA1(BCD, qB, qC, qD);
        or or3(inA, BCD, qA);

endmodule

module dF(q, qBar, D, clk, rst);
  input D, clk, rst;
  output q, qBar;

  reg q;

  not n1 (qBar, q);

  always@ (posedge rst or posedge clk)
  begin
    if(rst)
      q = 0;

    else
      q = D;

  end

endmodule
```