**TIC TAC TOE**

**Rules for the Game**

The player who makes all three marks in a vertical/horizontal or diagonal row wins the game, at first. A player plays the Tic Tac Toe Game with a computer (2nd player).

The following 2-bit values will be stored in one of the 9 positions. (All are random bit numbers)

00 – if neither player and computer played in that position [REST STATE]

01 – represents **X**. If player played at that position [PLAYER STATE]

10 – represents **0**. If computer played at that position [COMPUTER STATE]

11-game is finished [OVER STATE]

|  |  |  |
| --- | --- | --- |
| **1** | **2** | **3** |
| **4** | **5** | **6** |
| **7** | **8** | **9** |

The player or computer wins the game if 3 similar X/0 are put in the following rows:

(1,2,3), (4,5,6), (7,8,9) [HORIZONTAL]

(1,4,7), (2,5,8), (3,6,9) [VERTICAL]

(1,5,9), (3,5,7) [DIAGONAL]

**INPUTS:**

>Reset: 1 AFTER THE GAME IS OVER, RESETS THE GAME; 0 WHEN GAME STARTS

>Pp: 1 WHEN THE PLAYER PLAYS; 0 WHEN NOBODY IS PLAYING

>Cp: 1 TO SWITCH  FROM COMPUTER TO REST STATE; 0 TO REMAIN IN COMPUTER STATE

>wrong\_move: 1 WHEN PLAYER/COMPUTER MAKES A WRONG MOVE AND REST STATE IS RESTORED; 0 TO SWITCH STATES BETWEEN PLAYER TO COMPUTER (Cp=1)/ COMPUTER TO PLAYER (Pp=1) WHEN EITHER OF THEM MAKES A WRONG MOVE.

>Winner: 1 WHEN WINNER IS FOUND AND GAME IS RESET AFTER FINISHING; 0 WHEN NO WINNER FOUND

**Modules (Verilog):**

>module Tic\_Tac\_Toe(clock, reset, pp, cp, computer, player, pos1, pos2, pos3, pos4, pos5, pos6, pos7, pos8, pos9, winner);

input clock, reset, pp, cp;

input [3:0] computer , player;

output wire [1:0] pos1,pos2,pos3,pos4,pos5,pos6,pos7,pos8,pos9;

output wire[1:0] winner;

>module pos\_decode(input, en, out\_enable);

input[3:0] input ;

input en;

output wire [15:0] out\_enable;

>module position(clock, reset, wrong\_move, c\_enable, p\_enable, pos1, pos2, pos3, pos4, pos5, pos6, pos7, pos8, pos9);

input clock, reset, wrong\_move;

input [8:0] c\_enable; // computer enable

input [8:0] p\_enable;// player enable

output reg[1:0] pos1, pos2, pos3, pos4, pos5, pos6, pos7, pos8, pos9;

>module finish(pos1, pos2, pos3, pos4, pos5, pos6, pos7, pos8, pos9, filled);

input [1:0] pos1, pos2, pos3, pos4, pos5, pos6, pos7, pos8, pos9;

output filled; //all spaces are filled

>module re\_block (pos1, pos2, pos3, pos4, pos5, pos6, pos7, pos8, pos9, c\_enable, p\_enable, wrong\_move);

input [8:0] c\_enable, p\_enable;

input [1:0] pos1, pos2, pos3, pos4, pos5, pos6, pos7, pos8, pos9;

output wire wrong\_move;