Amar Vignesh S
EVD17I003
SOPC Lab Project Report
May 31st, 2020

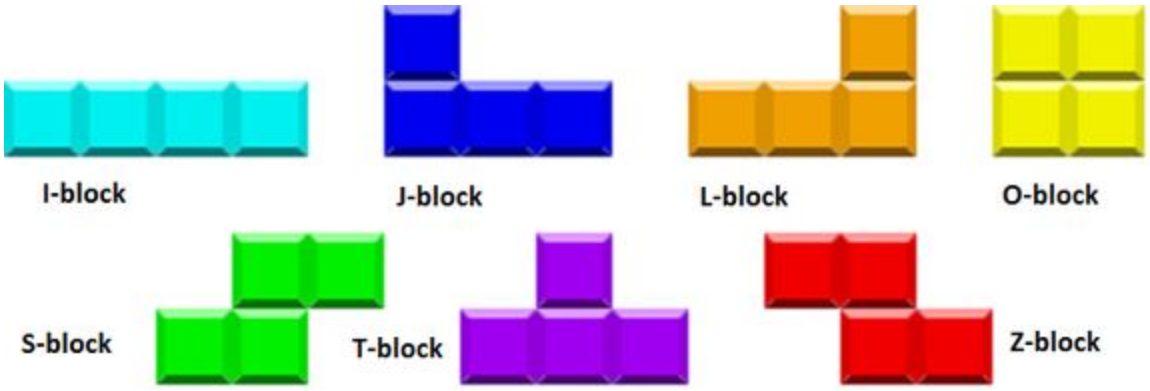
Final Project : Tetris implementation using Verilog

Introduction:

Tetris is a retro game where the objective for the player is to move blocks of different shapes to form as many lines as possible which then gets cleared. If the player arranges the block in such a way that the blocks touch the roof the game the game is over. Tetris was one of my favourite games as a toddler and recreating it using verilog sparked interest within me.

Game Rules:

As mentioned above the main objective of Tetris is to clear as many lines as possible. To achieve this players are offered with the freedom to move and rotate random pieces of blocks of different shapes. These blocks should be arranged in such a way that they form horizontal lines which then gets cleared. Each Tetrimino is composed of 4 square which are connected to each other in different ways .



19	152	153	154	155	156	157	158	159
18	144	145	146	147	148	149	150	151
17	136	137	138	139	140	141	142	143
16	128	129	130	131	132	133	134	135
15	120	121	122	123	124	125	126	127
14	112	113	114	115	116	117	118	119
13	104	105	106	107	108	109	110	111
12	96	97	98	99	100	101	102	103
11	88	89	90	91	92	93	94	95
10	80	81	82	83	84	85	86	87
9	72	73	74	75	76	77	78	79
8	64	65	66	67	68	69	70	71
7	56	57	58	59	60	61	62	63
6	48	49	50	51	52	53	54	55
5	40	41	42	43	44	45	46	47
4	32	33	34	35	36	37	38	39
3	24	25	26	27	28	29	30	31
2	16	17	18	19	20	21	22	23
1	8	9	10	11	12	13	14	15
0	0	1	2	3	4	5	6	7
	0	1	2	3	4	5	6	7

The grid shown above is a visual representation of 160 size grid and blocks 153,154,145,146 represent square Tetrimino and blocks 152,153,154,155 represent horizontal bar Tetrimino

Assumptions:

- The original Tetris game has 7 Tetriminos but in the case for this project I have used only 2 Tetriminos namely Bar and Square.
- The grid size is 160 with 20 rows and 8 columns
- The point of reference for block generation is block 154 , every calculation for block generation , rotation and movement is based on this block
- The states have been defined in the following way
  INITIAL = 8'b0000_0001,
  GENERATE_PIECE = 8'b0000_0010,
  ROTATE_PIECE = 8'b0000_0100,
  COLLISION = 8'b0000_1000,//bottom collision with other blocks
  LOSE = 8'b0001_0000,//at the top collision
  CLEAR_ROW = 8'b0010_0000,
  UNKNOWN = 8'bxxxx_xxxx;

  With outputs relating to states in the following way
  assign { O_Lose, O_Collision, O_Rotate, O_Generate, O_Initial} = state[4:0] ;//lose is 1 bit I is LSB

- The block type is referenced with a 3 bit number
  SQUARE = 3'b000,
  BAR = 3'b001,
  S = 3'b010,
  Z = 3'b011,
  L = 3'b100,
  J = 3'b101,
  T = 3'b110;

- Maximum Loop size = 25'd1 (extreme case 24+1)

Implementation Procedure :

There are two files one is for backend code and other is testbench file , due to the lack of fpga the game cannot be played by the user rather predefined inputs and their corresponding outputs could be verified with the help of testbench file and gtkwave.

Basic Outline of the code would be , A massive grid of 160 blocks is present representing the tetris grid with 20 rows and 8 columns. In each element of the grid value “1” represents presence of a block and “0” represents absence of block/ empty space.

There are five main states in this file: **Initial**, **Generate Block**, **Move** and **Rotate**, **Collision**, and **Lose**.

The initial state just is there for after the game loads and transitions to generate block when the game is started.

The generate block state would pull up a random number and generate a random block from the potential options, in this case a bar and a square. If the incoming block would be blocked by the current grid you would lose, just like in the real game. From here it moves to the rotate and move state.

The rotate and move state allows the player to adjust where the block will land. It is also here where the player is allowed to rotate the brick. After a certain loop condition the state would be changed to the collision state.

In the collision state the current brick is moved down. If this makes it hit another brick its location will be saved and a new brick would be generated. It is also in this state where it checks if a row has been filled. It checks the row and up to four rows to be deleted as this is the max case when a vertical bar is put into place.The rows are deleted by pushing the grid values down by the number of rows it got full and emptying the top rows accordingly.

From here it can either go to generate piece or rotate. If it hits a place it needs to stop (another brick or the bottom) it will go to generate piece. It is does not hit anything it will go to rotate and the cycle will continue

Results and Discussions :

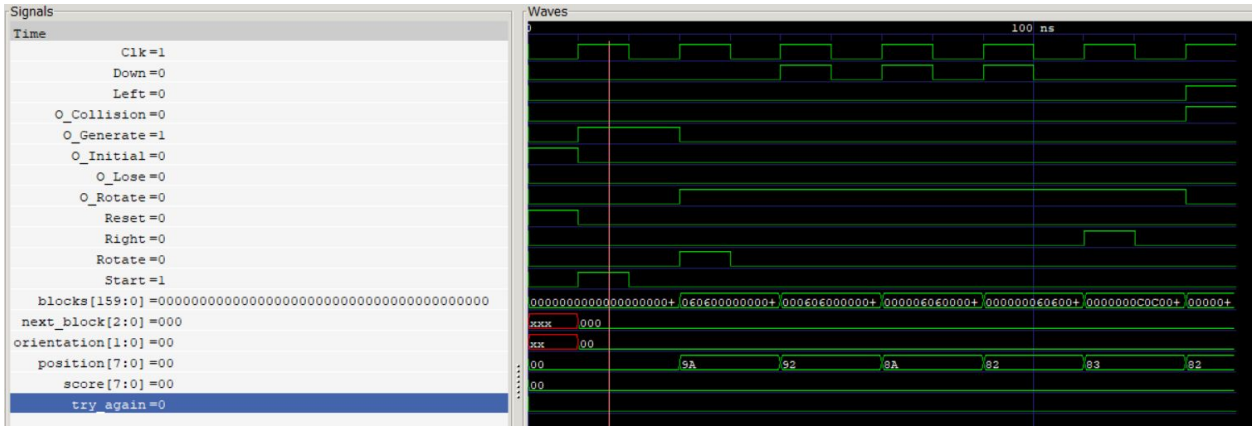
This project doesn't have submodules
It has a backend file and a testbench for pre-defined simulation

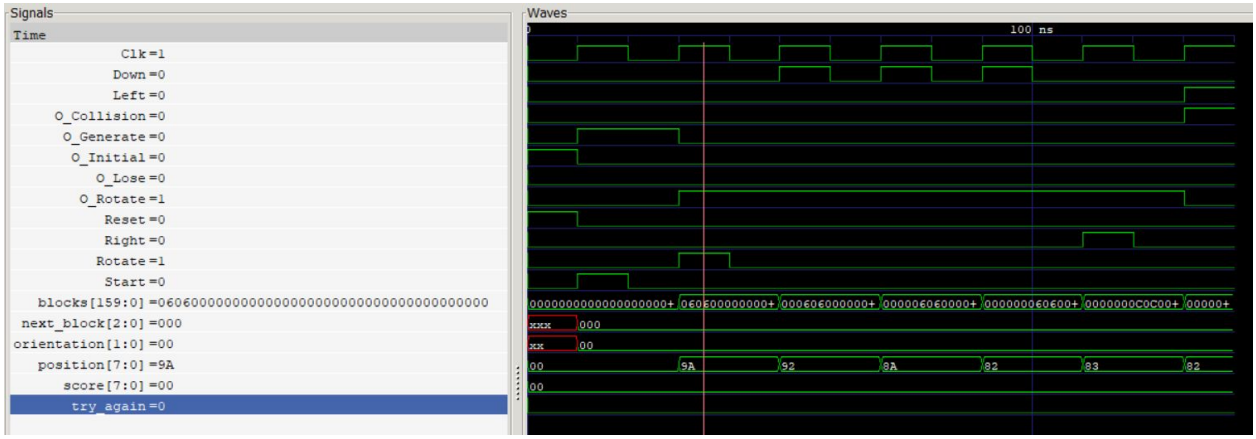
As this is a pre-defined input simulation , inputting multiple blocks , playing the game and clearing the blocks is very hard to write in testbench as it would be very long and confusing

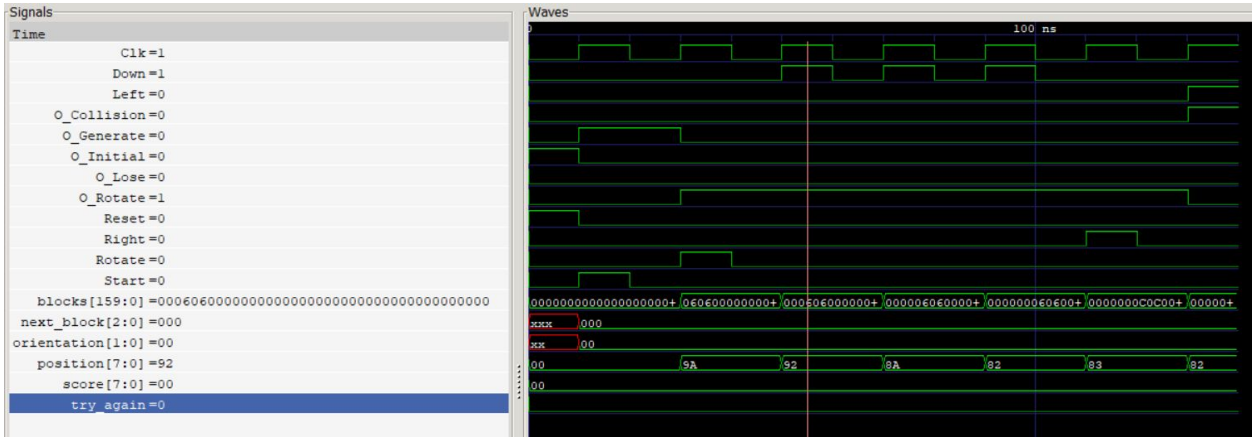
So to clear up a bit I have demonstrated the core functions such as Generating block , Reseting , Lateral movements and rotation

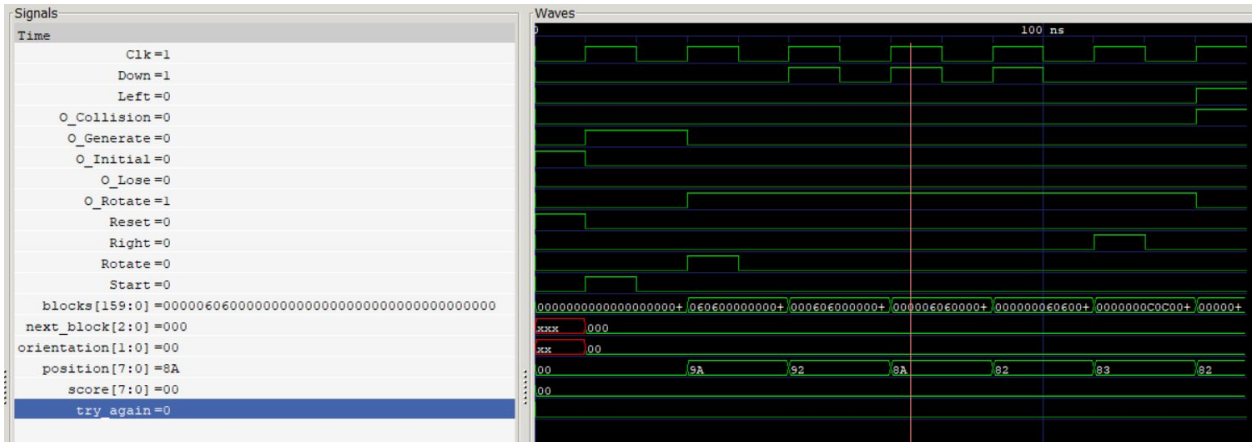
Reset, Clk, Start, try_again, Left, Right ,Down, Rotate are the Inputs
O_Collision, O_Generate , O_Lose , O_Rotate , O_Initial are the Outputs

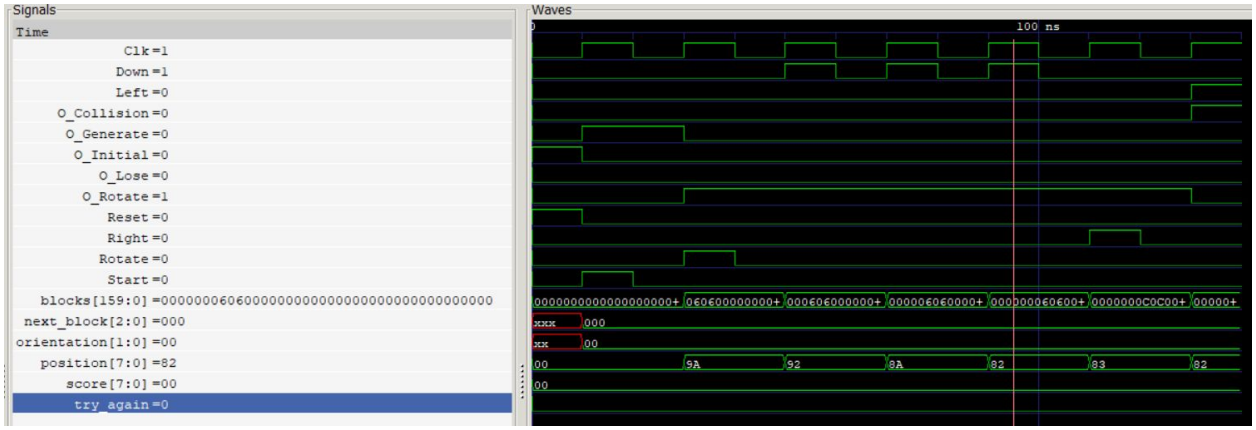
1. Reset has been pressed
2. Start has been pressed and game has started
3. Bar_0 has been generated and rotated
4. Down button has been pressed
5. Down button has been pressed again
6. Down button has been pressed again
7. Right button has been pressed
8. Left button has been pressed (Note the blocks value it reverts back to previous value)

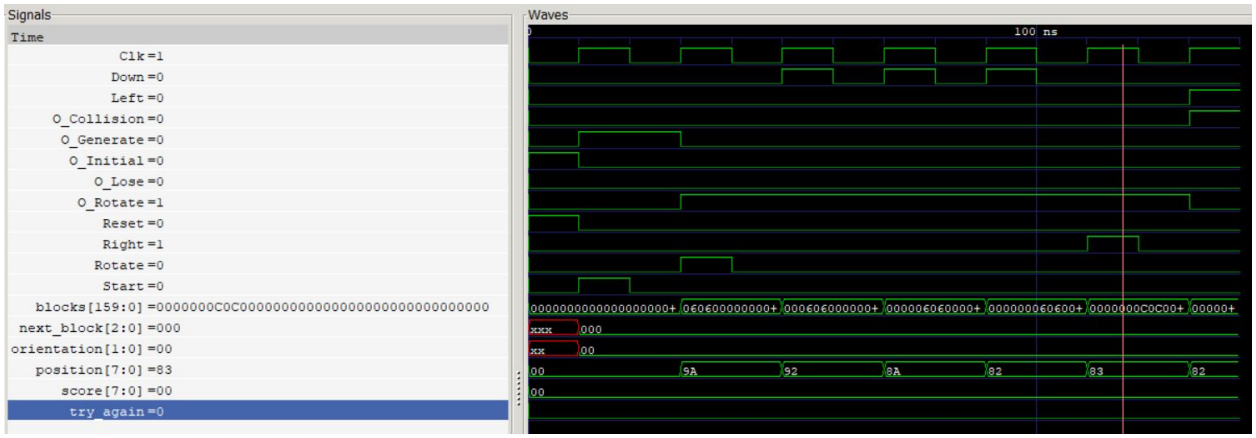

Signals
Time
Clk =1
Down =0
Left =0
O_Collision =0
O_Generate =1
O_Initial =0
O_Lose =0
O_Rotate =0
Reset =0
Right =0
Rotate =0
Start =1
blocks[159:0] =00000000000000000000000000000000000000000000
next_block[2:0] =000
orientation[1:0] =00
position[7:0] =00
score[7:0] =00
try_again =0
Waves
100 ns
00000000000000000000+ 0606000000000+ 0000606000000+ 0000006060000+ 0000000606000+ 0000000C0C000+ 00000+
xxx 000
xx 00
00 9A 92 8A 82 83 82
00

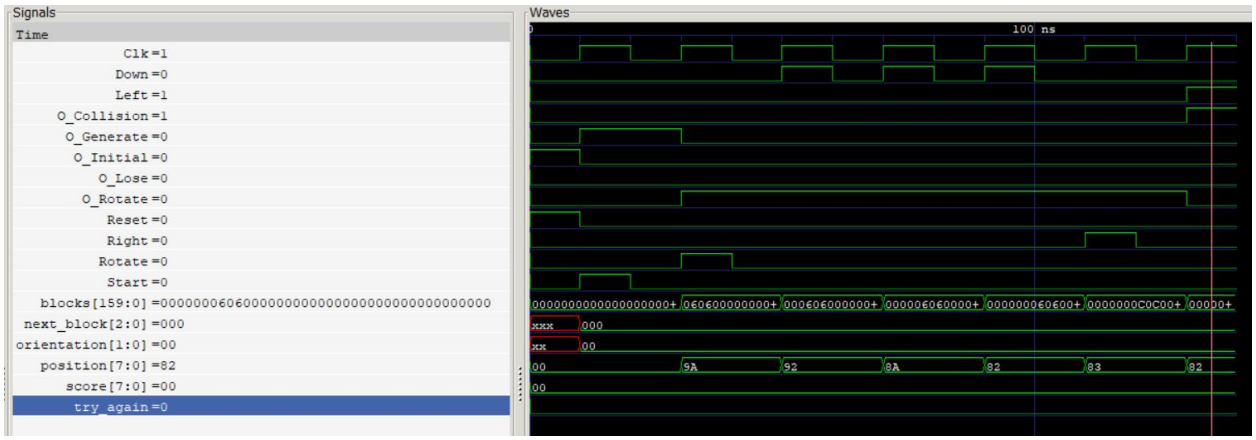

Signals
Time
Clk =1
Down =0
Left =0
O_Collision =0
O_Generate =0
O_Initial =0
O_Lose =0
O_Rotate =1
Reset =0
Right =0
Rotate =1
Start =0
blocks[159:0] =06060000000000000000000000000000000000000000
next_block[2:0] =000
orientation[1:0] =00
position[7:0] =9A
score[7:0] =00
try_again =0
Waves
100 ns
00000000000000000000+ 0606000000000+ 0000606000000+ 0000006060000+ 0000000606000+ 0000000C0C000+ 00000+
xxx 000
xx 00
00 9A 92 8A 82 83 82
00


Signals
Time
Clk =1
Down =1
Left =0
O_Collision =0
O_Generate =0
O_Initial =0
O_Lose =0
O_Rotate =1
Reset =0
Right =0
Rotate =0
Start =0
blocks[159:0] =00060600000000000000000000000000000000000000
next_block[2:0] =000
orientation[1:0] =00
position[7:0] =92
score[7:0] =00
try_again =0
Waves
100 ns
00000000000000000000+ 0606000000000+ 0000606000000+ 0000006060000+ 0000000606000+ 0000000C0C000+ 00000+
xxx 000
xx 00
00 9A 92 8A 82 83 82
00


Signals
Time
Clk =1
Down =1
Left =0
O_Collision =0
O_Generate =0
O_Initial =0
O_Lose =0
O_Rotate =1
Reset =0
Right =0
Rotate =0
Start =0
blocks[159:0] =000000606000000000000000000000000000000000
next_block[2:0] =000
orientation[1:0] =00
position[7:0] =8A
score[7:0] =00
try_again =0
Waves
100 ns
000000000000000000+
0606000000000+
0000606000000+
0000006060000+
0000000606000+
0000000C0C000+
00000+
000
00
9A
92
8A
82
83
82


Signals
Time
Clk =1
Down =1
Left =0
O_Collision =0
O_Generate =0
O_Initial =0
O_Lose =0
O_Rotate =1
Reset =0
Right =0
Rotate =0
Start =0
blocks[159:0] =000000060600000000000000000000000000000000
next_block[2:0] =000
orientation[1:0] =00
position[7:0] =82
score[7:0] =00
try_again =0
Waves
100 ns
000000000000000000+
0606000000000+
0000606000000+
0000006060000+
0000000606000+
0000000C0C000+
00000+
000
00
9A
92
8A
82
83
82


Signals
Time
Clk =1
Down =0
Left =0
O_Collision =0
O_Generate =0
O_Initial =0
O_Lose =0
O_Rotate =1
Reset =0
Right =1
Rotate =0
Start =0
blocks[159:0] =0000000C0C000000000000000000000000000000
next_block[2:0] =000
orientation[1:0] =00
position[7:0] =83
score[7:0] =00
try_again =0
Waves
100 ns
000000000000000000+
0606000000000+
0000606000000+
0000006060000+
0000000606000+
0000000C0C000+
00000+
000
00
9A
92
8A
82
83
82



Appendix :

tetrisf.v - Backend of the game
tetris1_test1.v - testbench