DIGISM_ROUND_2-SOLUTION

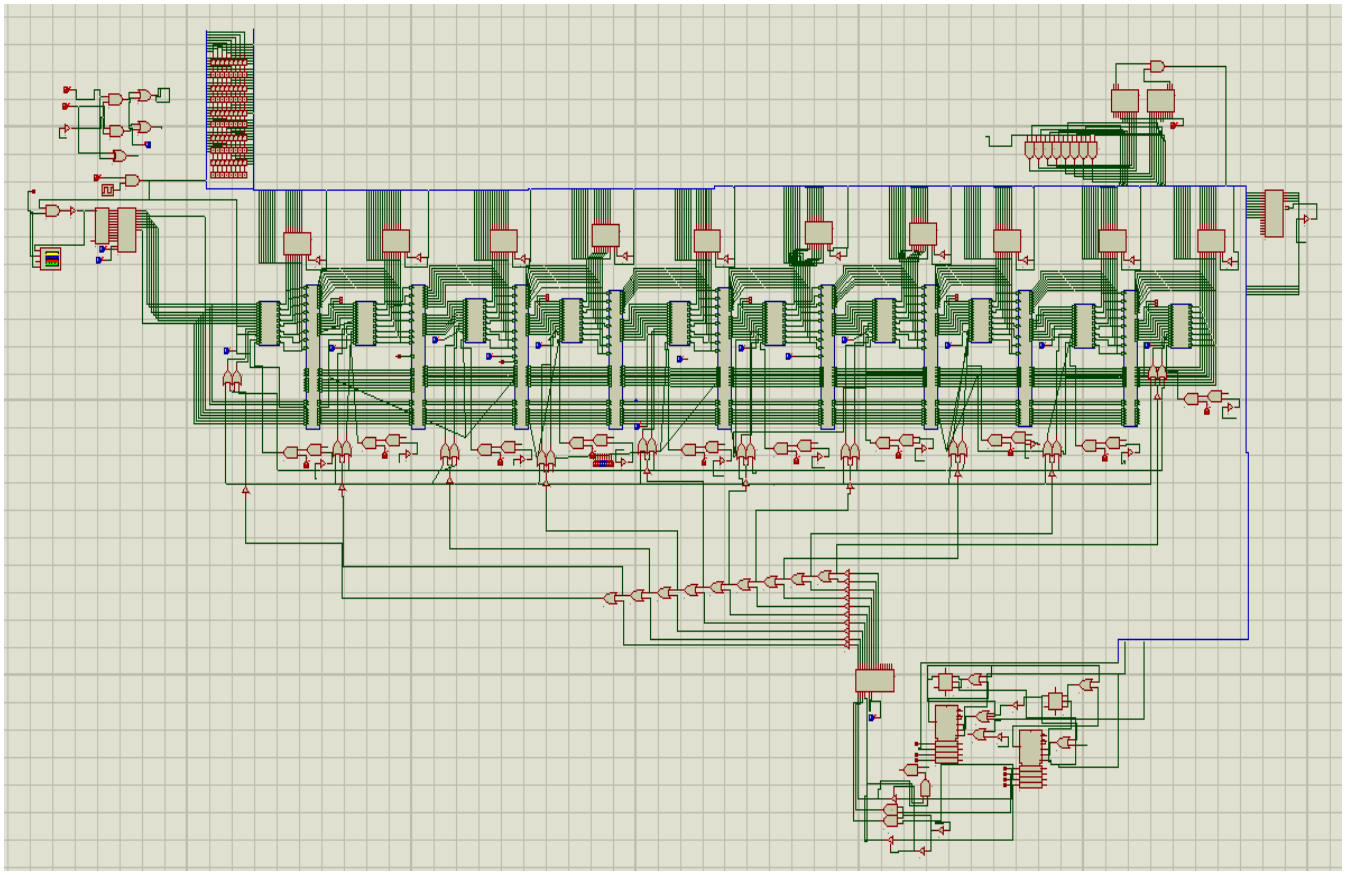
A BRIEF OVERVIEW

TETTRIS

-BRICKS AND BYTES

Team Name:555 Timers

1. R.Lokesh Krishna
2. Sirusala Niranth Sai



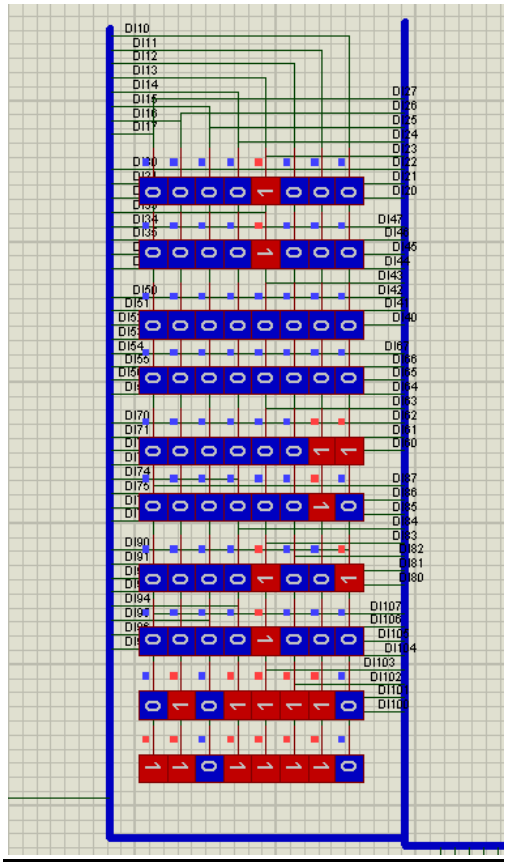
Aim:

To develop a simplified version of the classic Tetris game with the minimalistic components possible, with basic gates and sequential circuit components.

Key Aspects of the Project:

1. There is no use of any microcontroller or a microprocessor used.
2. Also no full-fledged CPU architecture was mimicked rather a crude version of bi-counter system with a predefined flow of control and interrupt signals was used.

Our Approach:



We broke down the overall process as two:

1. The part were user interacts ie until a piece entering reaches the end of the stack.
2. The second part is traversing the stack for rows that are full and removing them.

Both these process require separate iterators/counters to keep track of them and also interrupt signals. In brief

Process No.	Process undertaken	Counter Active	Interrupt Caused	No. of counts	No. of clock cycle to complete
1.	Bringing one piece from top to bottom as well as updating user actions	Niranth_Counter(N_C)	User_PRESS	10	10<
2.	Traverses the full screen and removes fully filled registers aka rows in the game	Lokesh_Counter(L_C)	Full_row_detected	10	10<

<u>Our Overall design:</u>

1. A display Matrix
2. Data flow Pipeline (10 - 8 bit shit register + custom Combinatinol circuit for collision detection)
3. Bi Counter Program Controller(2 counters,2 flip-flops and gates)
4. Control Updater (3-8-deCoder,gates)
5. Rom + Counter (2732,12 bit counter for controlling the rom)
6. User Input Block (toggles and gates)

Flow of control:

Pseudo code of the process 1:

1. Start
2. Load a layer from ROM
3. Check for User press. //user interrupt ,thus here counter is
   // freezed until user action is updated
4. if User press == Left
   Shift bits to left of ith register
   else if User press == Right
   Shift bits to right ith register
5. Check Vertical Collision and update register values accordingly
6. Increment Niranth_Counter
7. If Niranth_Counter = 10
   Go to process 2
   else
   Go to step 1

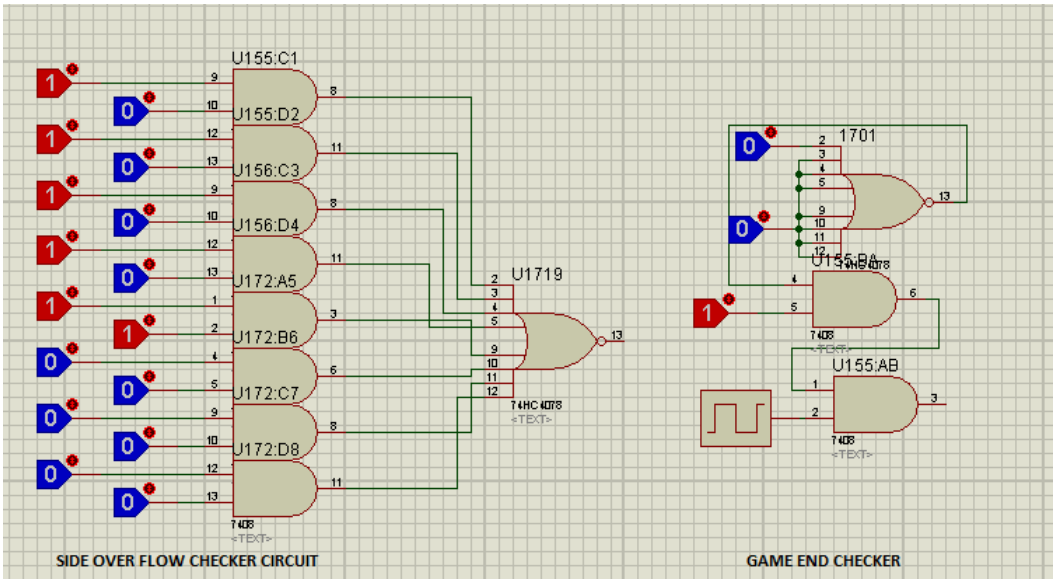
Pseudo code of the process 2:

1. Start
2. freeze the Register
3. Check if register 'j' is full //interrupt..thus here counter is if User
4. If register [j] == full
   1. Flush all bits
5. Check Vertical Collision and update register values accordingly
6. Increment Lokesh_Counter
7. If Lokesh_Counter = 10
   i. Go to process 1
   else
   ii. Go to step 1

Limitations and innovations:

- The Horizontal collision is not detected as the register have no control over every individual bit and a custom design makes it bulky and costly.

- The above problem also causes the bricks to break, for which we have formulated a abstract idea as well which was implemented and not incorporated(video available).
- The Left and Right Over flow can be detected and the circuit has as well been developed but not added given the time constraint.
- The Game Over condition is true when there is at least a single brick in the top most layer of the stack. Again, the circuit implementing this has been implemented and not added to the main circuit



- We have as well written a python script to create a required binary File for the ROM with the details of the input bricks were the static frames are encoded as decimals.