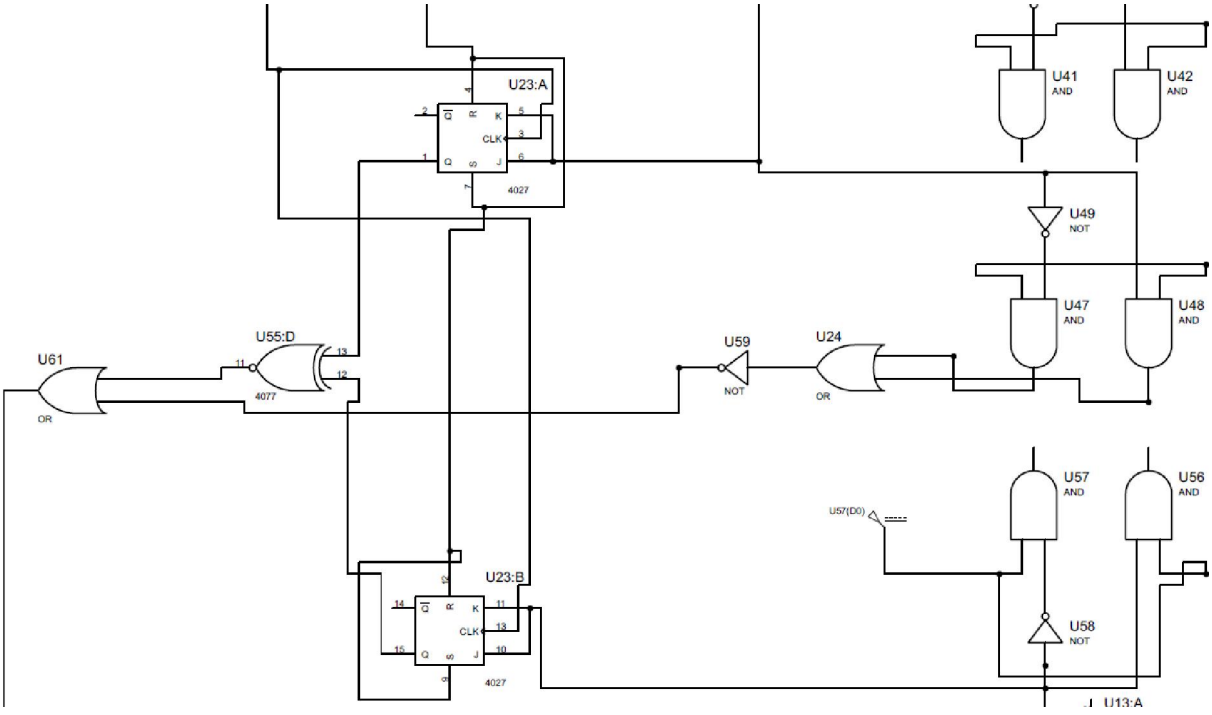
Snatch or Lose – Duos

Final Review

Design of the remaining elements based upon the Idea:

Scoring Logic:

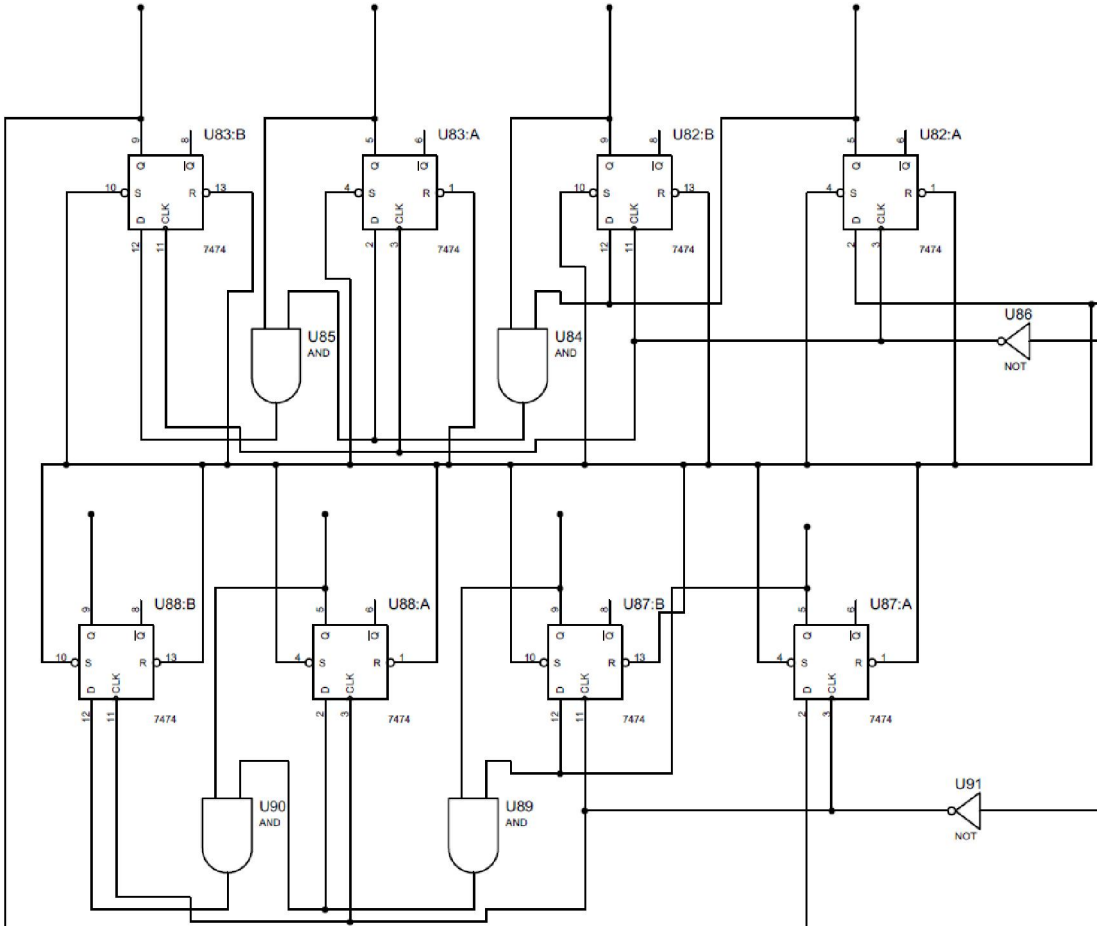
As mentioned in the Review 1, we were going to use simple AND logic for scoring mechanism of the game but it turned out to be wrong. We needed clock triggered scoring logic in order to check for the score or death only at the falling edge of the clock. Moreover the scoring logic had to be inactive until the first appearance of the falling object in the last row. Considering all this, design of the scoring logic circuit was done.



Counter for Display:

To display the score, we used two counters from 0 to 9 in order to display the score from 00 to 99 on two 7 segment displays.

This had simple sequential logic and it was designed by making the state table for both the counters.



Reset button:

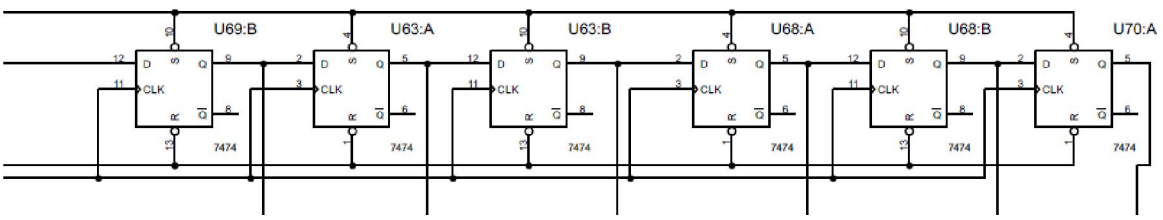
We realised we needed a reset button in case of death in the game. When the scoring logic fails, death occurs and game stops since the scoring logic triggers the stoppage of the game clock. After that, all the flip flops in almost all the elements had to be reset to zero or some initial value. For this, we used a reset button. One input of this button was Vcc and the other end was connected to all the set or preset of the flip flops which had to change from 0 to 1. So normally it passed 0 when the button was pressed it passed 1, thus, fulfilling our requirement.

Game control buttons:

The game control buttons were connected to the clock of two JK flip flops (one for each). These JK's had their J and K shorted and given Vcc, so on applying clock they changed the input value for the demultiplexer of the platform LED's, thus, changing the glow position.

**A last minute problem:

One thing that we missed while designing was that since we used demultiplexers for the falling objects, even when they had 0 as input the LED's to their respective output glowed from the start of the game. The idea of the game is that initially objects occur in the first row and gradually fall down. To solve this, we used another 6 bit shift register ("Controller"). The input of this controller was Vcc and all the flip flops were initially set to zero. The outputs of the flip flops in sequence were connected to the AND gate's of the demultiplexers of the falling objects of row 2 to row 7. Therefore, initially only the first row glowed and with increment of the clock consecutive rows were 'activated'.

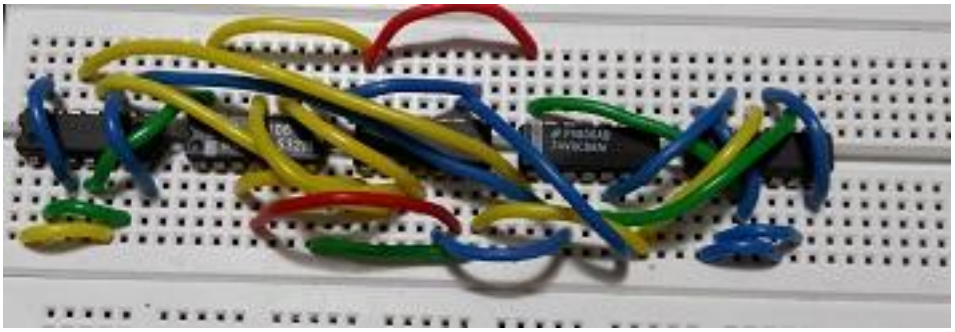


Output board:

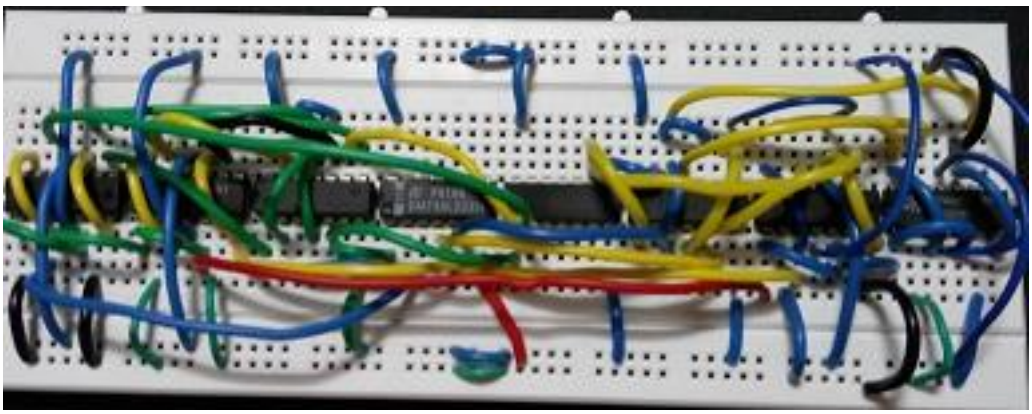
The output board is a strip board on which we soldered 32 LED's, 28 Red for falling objects and 4 Green for platform. Also the game buttons and Reset button was soldered along with the two 7 segment displays to display the score.

Implementation of the design:

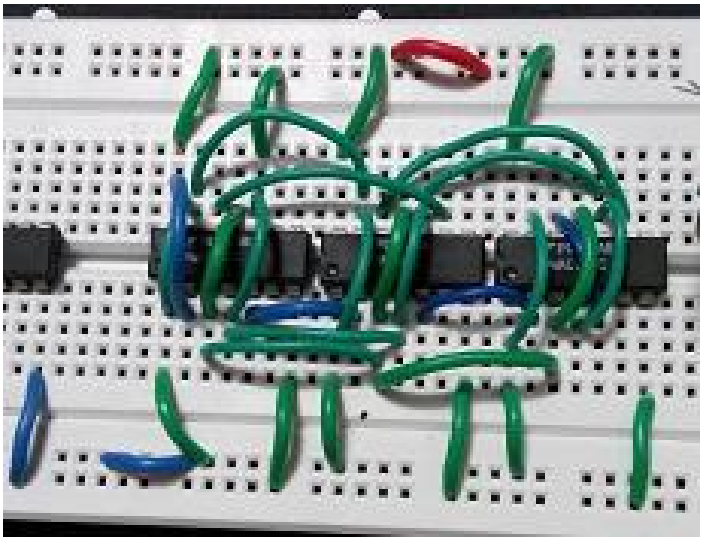
Scoring Logic:



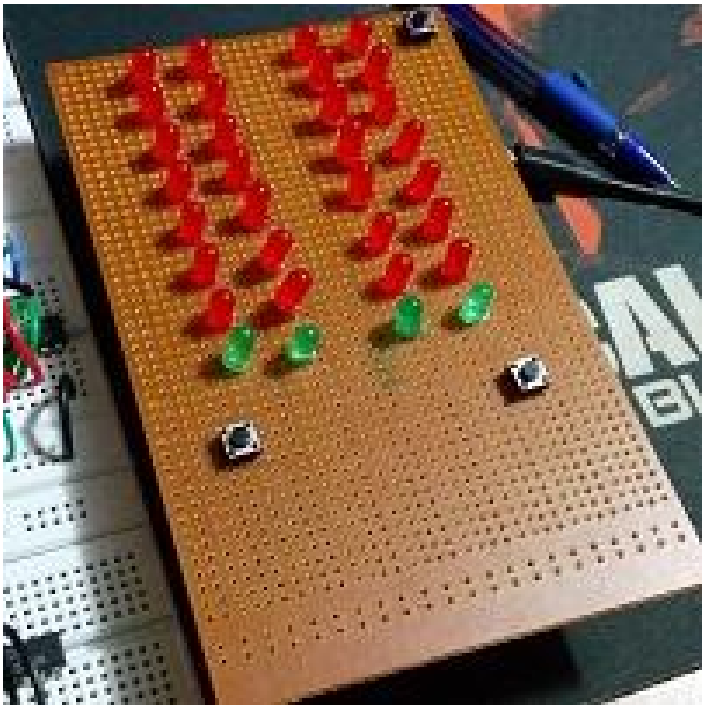
Counter for Display:



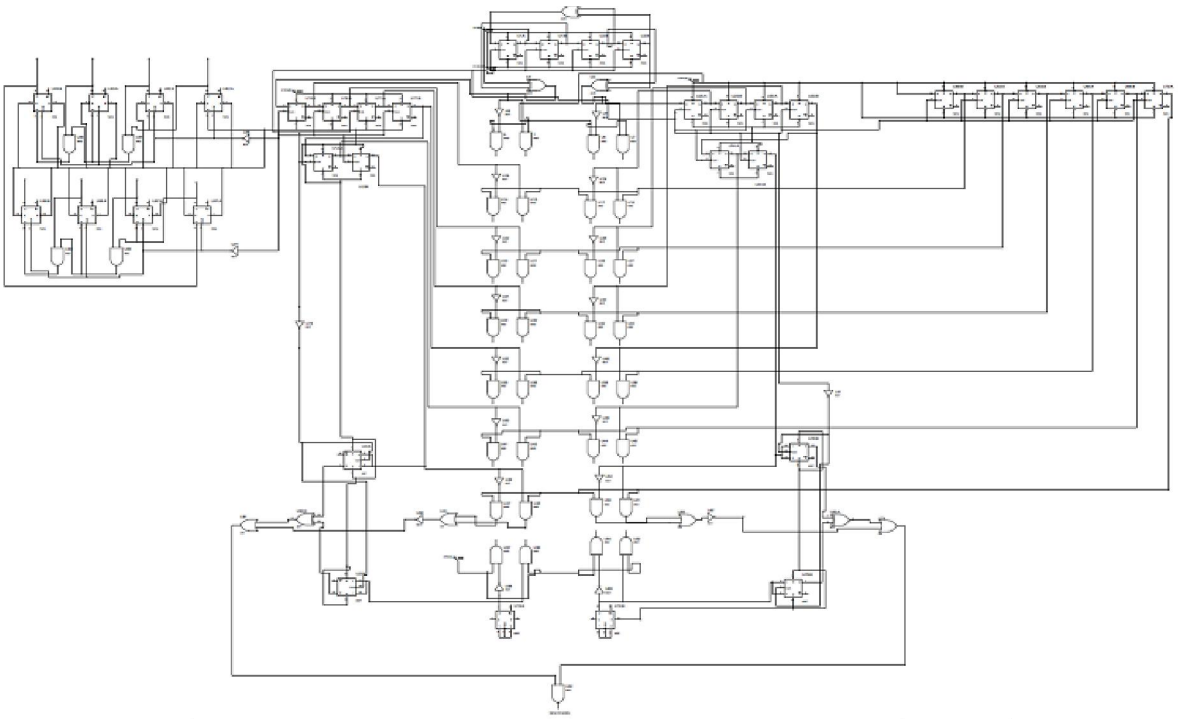
Controller:



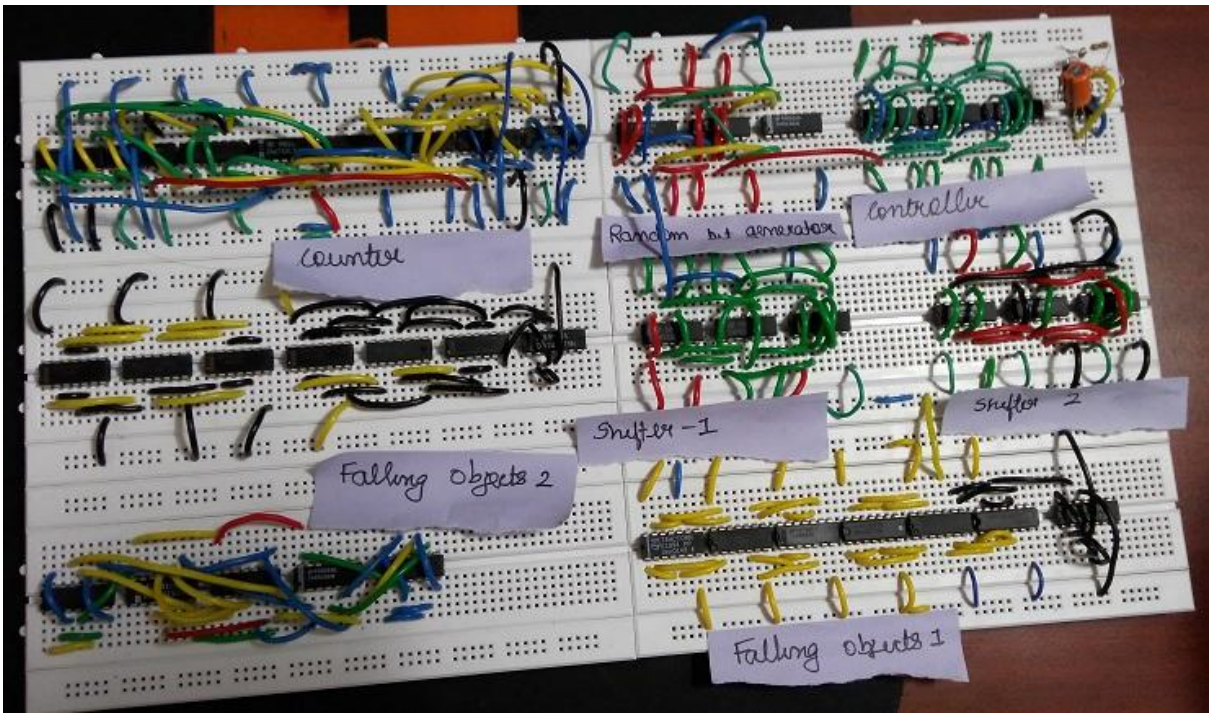
Output Board:



Full Circuit Diagram:



Full Circuit Implementation:



Along with all this, we completed the whole VHDL coding for the circuit.

VHDL Code:

```
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;

library work;
use work.snatch_lose3_pkg.ALL;



entity snatch_lose1 is
        port (

                sw_0 : in std_logic;
```

```
        IO1 : in std_logic;
        Led_0 : out std_logic;
        Led_1 : out std_logic;
        Led_2 : out std_logic;
        Led_3 : out std_logic;
        IO2 : out std_logic;
        IO3 : out std_logic;
        IO4 : out std_logic;
        IO5 : out std_logic;
        IO6 : out std_logic;
        IO7 : out std_logic;
        IO8 : out std_logic;
        IO9 : out std_logic;
        IO10 : out std_logic;
        IO11 : out std_logic;
        IO12 : out std_logic;
        IO13 : out std_logic;
        IO14 : out std_logic;
        IO15 : out std_logic;
        IO16 : out std_logic;
        IO17 : out std_logic;
        IO18 : out std_logic;
        IO19 : out std_logic;
        IO20 : out std_logic;
        IO21 : out std_logic;
        IO22 : out std_logic;
        IO23 : out std_logic;
        IO24 : out std_logic;
        IO25 : out std_logic;
        IO26 : out std_logic;
        IO27 : out std_logic;
        IO28 : out std_logic;
        IO29 : out std_logic;
        clk : in std_logic
);
```

```
end snatch_lose1;



architecture behavioral of snatch_lose1 is



        component AND2_NI
                port (
  B : in STD_LOGIC := 'X';
  A : in STD_LOGIC := 'X';
  Y : out STD_LOGIC := 'U'
 );
        end component;

        component AUTO_IBUF
                port(
                I : in std_logic;
                O : out std_logic
        );
        end component;

        component AUTO_OBUF
                port(
                I : in std_logic;
                O : out std_logic
        );
        end component;

        component BUF_INV_NI
                port (
  A : in STD_LOGIC := 'X';
  Y : out STD_LOGIC := 'U'
 );
        end component;
```

```
		component DIGITAL_HIGH_NI
			Port (
 Y : out  STD_LOGIC:='1'
 );
		end component;

		component FF_D_PCLR_CO_NI
			port (
  D: in STD_LOGIC;
  PR: in STD_LOGIC;
  CLK : in STD_LOGIC;
  CLR : in STD_LOGIC;
  Q : out STD_LOGIC;
  Qneg : out STD_LOGIC
);
		end component;

		component FF_JK_PSCLR_CO_NI
			port (
  J: in STD_LOGIC;
  K: in STD_LOGIC;
  CLK : in STD_LOGIC;
  PR: in STD_LOGIC;
  CLR: in STD_LOGIC;
  Q : out STD_LOGIC;
  Qneg : out STD_LOGIC
);
		end component;

		component OR2_NI
			port (
   B : in STD_LOGIC := 'X';
   A : in STD_LOGIC := 'X';
   Y : out STD_LOGIC := 'U'
  );
```

```
        end component;

        component XNOR2_NI
                port (
  B : in STD_LOGIC := 'X';
  A : in STD_LOGIC := 'X';
  Y : out STD_LOGIC := 'U'
);
        end component;

        component XOR2_NI
                port (
  B : in STD_LOGIC := 'X';
  A : in STD_LOGIC := 'X';
  Y : out STD_LOGIC := 'U'
);
        end component;
        signal \1\ : std_logic;
        signal \81\ : std_logic;
        signal \31\ : std_logic;
        signal \7\ : std_logic;
        signal \3\ : std_logic;
        signal \2\ : std_logic;
        signal \55\ : std_logic;
        signal \57\ : std_logic;
        signal \58\ : std_logic;
        signal \59\ : std_logic;
        signal \60\ : std_logic;
        signal \61\ : std_logic;
        signal \62\ : std_logic;
        signal \63\ : std_logic;
        signal \64\ : std_logic;
        signal \65\ : std_logic;
        signal \66\ : std_logic;
        signal \67\ : std_logic;
```

```
signal \68\ : std_logic;
signal \69\ : std_logic;
signal \70\ : std_logic;
signal \71\ : std_logic;
signal \72\ : std_logic;
signal \73\ : std_logic;
signal \74\ : std_logic;
signal \75\ : std_logic;
signal \76\ : std_logic;
signal \77\ : std_logic;
signal \78\ : std_logic;
signal \80\ : std_logic;
signal \79\ : std_logic;
signal \82\ : std_logic;
signal \83\ : std_logic;
signal \84\ : std_logic;
signal \85\ : std_logic;
signal \86\ : std_logic;
signal \87\ : std_logic;
signal \89\ : std_logic;
signal \4\ : std_logic;
signal \90\ : std_logic;
signal \6\ : std_logic;
signal \5\ : std_logic;
signal \8\ : std_logic;
signal \9\ : std_logic;
signal \10\ : std_logic;
signal \11\ : std_logic;
signal \12\ : std_logic;
signal \14\ : std_logic;
signal \15\ : std_logic;
signal \13\ : std_logic;
signal \16\ : std_logic;
signal \24\ : std_logic;
signal \17\ : std_logic;
```

```
signal \18\ : std_logic;
signal \19\ : std_logic;
signal \20\ : std_logic;
signal \21\ : std_logic;
signal \22\ : std_logic;
signal \23\ : std_logic;
signal \30\ : std_logic;
signal \25\ : std_logic;
signal \26\ : std_logic;
signal \27\ : std_logic;
signal \28\ : std_logic;
signal \29\ : std_logic;
signal \32\ : std_logic;
signal \33\ : std_logic;
signal \34\ : std_logic;
signal \35\ : std_logic;
signal \36\ : std_logic;
signal \37\ : std_logic;
signal \38\ : std_logic;
signal \39\ : std_logic;
signal \40\ : std_logic;
signal \41\ : std_logic;
signal \42\ : std_logic;
signal \43\ : std_logic;
signal \44\ : std_logic;
signal \45\ : std_logic;
signal \46\ : std_logic;
signal \47\ : std_logic;
signal \48\ : std_logic;
signal \49\ : std_logic;
signal \50\ : std_logic;
signal \51\ : std_logic;
signal \52\ : std_logic;
signal \53\ : std_logic;
signal \54\ : std_logic;
```

```
        signal \56\ : std_logic;
        signal \113\ : std_logic;
        signal \112\ : std_logic;
        signal \111\ : std_logic;
        signal \110\ : std_logic;
        signal \109\ : std_logic;
        signal \108\ : std_logic;
        signal \107\ : std_logic;
        signal \106\ : std_logic;
        signal \105\ : std_logic;
        signal \104\ : std_logic;
        signal \103\ : std_logic;
        signal \102\ : std_logic;
        signal \101\ : std_logic;
        signal \100\ : std_logic;
        signal \99\ : std_logic;
        signal \98\ : std_logic;
        signal \97\ : std_logic;
        signal \96\ : std_logic;
        signal \95\ : std_logic;
        signal \94\ : std_logic;
        signal \93\ : std_logic;
        signal \92\ : std_logic;
        signal \91\ : std_logic;
        signal \88\ : std_logic;
        signal \117\ : std_logic;
        signal \116\ : std_logic;
        signal \115\ : std_logic;
        signal \114\ : std_logic;
begin
        sw_0_AUTOBUF : AUTO_IBUF
                port map( I => sw_0, O => \79\ );
        IO1_AUTOBUF : AUTO_IBUF
                port map( I => IO1, O => \90\ );
        Led_0_AUTOBUF : AUTO_OBUF
```

```
            port map( I => \88\, O => Led_0 );
Led_1_AUTOBUF : AUTO_OBUF
            port map( I => \91\, O => Led_1 );
Led_2_AUTOBUF : AUTO_OBUF
            port map( I => \92\, O => Led_2 );
Led_3_AUTOBUF : AUTO_OBUF
            port map( I => \93\, O => Led_3 );
IO2_AUTOBUF : AUTO_OBUF
            port map( I => \94\, O => IO2 );
IO3_AUTOBUF : AUTO_OBUF
            port map( I => \95\, O => IO3 );
IO4_AUTOBUF : AUTO_OBUF
            port map( I => \96\, O => IO4 );
IO5_AUTOBUF : AUTO_OBUF
            port map( I => \97\, O => IO5 );
IO6_AUTOBUF : AUTO_OBUF
            port map( I => \98\, O => IO6 );
IO7_AUTOBUF : AUTO_OBUF
            port map( I => \99\, O => IO7 );
IO8_AUTOBUF : AUTO_OBUF
            port map( I => \100\, O => IO8 );
IO9_AUTOBUF : AUTO_OBUF
            port map( I => \101\, O => IO9 );
IO10_AUTOBUF : AUTO_OBUF
            port map( I => \102\, O => IO10 );
IO11_AUTOBUF : AUTO_OBUF
            port map( I => \103\, O => IO11 );
IO12_AUTOBUF : AUTO_OBUF
            port map( I => \104\, O => IO12 );
IO13_AUTOBUF : AUTO_OBUF
            port map( I => \105\, O => IO13 );
IO14_AUTOBUF : AUTO_OBUF
            port map( I => \106\, O => IO14 );
IO15_AUTOBUF : AUTO_OBUF
            port map( I => \107\, O => IO15 );
```

```
IO16_AUTOBUF : AUTO_OBUF
        port map( I => \108\, O => IO16 );
IO17_AUTOBUF : AUTO_OBUF
        port map( I => \109\, O => IO17 );
IO18_AUTOBUF : AUTO_OBUF
        port map( I => \110\, O => IO18 );
IO19_AUTOBUF : AUTO_OBUF
        port map( I => \111\, O => IO19 );
IO20_AUTOBUF : AUTO_OBUF
        port map( I => \112\, O => IO20 );
IO21_AUTOBUF : AUTO_OBUF
        port map( I => \113\, O => IO21 );
IO22_AUTOBUF : AUTO_OBUF
        port map( I => \48\, O => IO22 );
IO23_AUTOBUF : AUTO_OBUF
        port map( I => \49\, O => IO23 );
IO24_AUTOBUF : AUTO_OBUF
        port map( I => \50\, O => IO24 );
IO25_AUTOBUF : AUTO_OBUF
        port map( I => \51\, O => IO25 );
IO26_AUTOBUF : AUTO_OBUF
        port map( I => \114\, O => IO26 );
IO27_AUTOBUF : AUTO_OBUF
        port map( I => \115\, O => IO27 );
IO28_AUTOBUF : AUTO_OBUF
        port map( I => \116\, O => IO28 );
IO29_AUTOBUF : AUTO_OBUF
        port map( I => \117\, O => IO29 );
U1 : FF_JK_PSCLR_CO_NI
        port map( J => \7\, Q => \2\, Qneg => open, K => \7\, CLR => \6\, CLK => \9\, PR => \6\ );
U2 : FF_JK_PSCLR_CO_NI
        port map( J => \8\, Q => \14\, Qneg => open, K => \8\, CLR => \6\, CLK => \9\, PR => \6\ );
U3 : FF_JK_PSCLR_CO_NI
        port map( J => \4\, Q => \15\, Qneg => open, K => \4\, CLR => \6\, CLK => \9\, PR => \6\ );
U4 : FF_JK_PSCLR_CO_NI
```

```
        port map( J => \6\, Q => \4\, Qneg => open, K => \6\, CLR => \6\, CLK => \9\, PR => \6\ );
U5 : AND2_NI
        port map( A => \8\, B => \14\, Y => \7\ );
U6 : AND2_NI
        port map( A => \4\, B => \15\, Y => \8\ );
U7 : FF_JK_PSCLR_CO_NI
        port map( J => \3\, Q => \1\, Qneg => open, K => \3\, CLR => \6\, CLK => \13\, PR => \6\ );
U8 : FF_JK_PSCLR_CO_NI
        port map( J => \11\, Q => \10\, Qneg => open, K => \11\, CLR => \6\, CLK => \13\, PR => \6\ );
U9 : FF_JK_PSCLR_CO_NI
        port map( J => \12\, Q => \5\, Qneg => open, K => \12\, CLR => \6\, CLK => \13\, PR => \6\ );
U10 : FF_JK_PSCLR_CO_NI
        port map( J => \2\, Q => \12\, Qneg => open, K => \2\, CLR => \6\, CLK => \13\, PR => \6\ );
U11 : BUF_INV_NI
        port map( Y => \9\, A => \16\ );
U12 : AND2_NI
        port map( A => \11\, B => \10\, Y => \3\ );
U13 : AND2_NI
        port map( A => \12\, B => \5\, Y => \11\ );
U14 : BUF_INV_NI
        port map( Y => \13\, A => \16\ );
U15 : FF_D_PCLR_CO_NI
        port map( D => \89\, Q => \18\, Qneg => open, CLR => \6\, CLK => \16\, PR => \86\ );
U16 : FF_D_PCLR_CO_NI
        port map( D => \18\, Q => \19\, Qneg => open, CLR => \6\, CLK => \16\, PR => \86\ );
U17 : FF_D_PCLR_CO_NI
        port map( D => \19\, Q => \20\, Qneg => open, CLR => \6\, CLK => \16\, PR => \86\ );
U18 : FF_D_PCLR_CO_NI
        port map( D => \20\, Q => \21\, Qneg => open, CLR => \6\, CLK => \16\, PR => \86\ );
U19 : FF_D_PCLR_CO_NI
        port map( D => \21\, Q => \22\, Qneg => open, CLR => \23\, CLK => \16\, PR => open );
U20 : FF_D_PCLR_CO_NI
        port map( D => \22\, Q => \56\, Qneg => open, CLR => \23\, CLK => \16\, PR => \23\ );
U21 : BUF_INV_NI
        port map( Y => \24\, A => \16\ );
```

```
U22 : FF_JK_PSCLR_CO_NI
        port map( J => \56\, Q => \25\, Qneg => open, K => \56\, CLR => \23\, CLK => \24\, PR => \23\ );
U23 : XNOR2_NI
        port map( A => \27\, B => \25\, Y => \26\ );
U24 : OR2_NI
        port map( A => \46\, B => \26\, Y => \32\ );
U25 : FF_JK_PSCLR_CO_NI
        port map( J => \28\, Q => \27\, Qneg => open, K => \28\, CLR => \23\, CLK => \24\, PR => \23\ );
U26 : FF_JK_PSCLR_CO_NI
        port map( J => \29\, Q => \28\, Qneg => open, K => \29\, CLR => \29\, CLK => \79\, PR => \29\ );
U27 : AND2_NI
        port map( A => \37\, B => \30\, Y => \114\ );
U28 : BUF_INV_NI
        port map( Y => \30\, A => \28\ );
U29 : AND2_NI
        port map( A => \28\, B => \37\, Y => \115\ );
U30 : AND2_NI
        port map( A => \33\, B => \32\, Y => open );
U31 : FF_JK_PSCLR_CO_NI
        port map( J => \34\, Q => \35\, Qneg => open, K => \34\, CLR => \38\, CLK => \90\, PR => \34\ );
U32 : BUF_INV_NI
        port map( Y => \36\, A => \35\ );
U33 : AND2_NI
        port map( A => \37\, B => \36\, Y => \116\ );
U34 : AND2_NI
        port map( A => \35\, B => \37\, Y => \117\ );
U35 : FF_JK_PSCLR_CO_NI
        port map( J => \35\, Q => \39\, Qneg => open, K => \35\, CLR => \38\, CLK => \44\, PR => \38\ );
U36 : XNOR2_NI
        port map( A => \43\, B => \39\, Y => \40\ );
U37 : OR2_NI
        port map( A => \40\, B => \41\, Y => \33\ );
U38 : BUF_INV_NI
        port map( Y => \41\, A => \42\ );
U39 : OR2_NI
```

```
        port map( A => \51\, B => \50\, Y => \42\ );
U40 : FF_JK_PSCLR_CO_NI
        port map( J => \45\, Q => \43\, Qneg => open, K => \45\, CLR => \38\, CLK => \44\, PR => \38\ );
U41 : BUF_INV_NI
        port map( Y => \44\, A => \62\ );
U42 : BUF_INV_NI
        port map( Y => \46\, A => \47\ );
U43 : OR2_NI
        port map( A => \49\, B => \48\, Y => \47\ );
U44 : AND2_NI
        port map( A => \52\, B => \53\, Y => \48\ );
U45 : AND2_NI
        port map( A => \53\, B => \56\, Y => \49\ );
U46 : AND2_NI
        port map( A => \54\, B => \53\, Y => \50\ );
U47 : AND2_NI
        port map( A => \53\, B => \45\, Y => \51\ );
U48 : BUF_INV_NI
        port map( Y => \52\, A => \56\ );
U49 : BUF_INV_NI
        port map( Y => \54\, A => \45\ );
U50 : AND2_NI
        port map( A => \65\, B => \60\, Y => \110\ );
U51 : AND2_NI
        port map( A => \60\, B => \22\, Y => \111\ );
U52 : AND2_NI
        port map( A => \67\, B => \60\, Y => \112\ );
U53 : AND2_NI
        port map( A => \60\, B => \76\, Y => \113\ );
U54 : BUF_INV_NI
        port map( Y => \65\, A => \22\ );
U55 : BUF_INV_NI
        port map( Y => \67\, A => \76\ );
U56 : AND2_NI
        port map( A => \64\, B => \59\, Y => \106\ );
```

```
U57 : AND2_NI
        port map( A => \59\, B => \21\, Y => \107\ );
U58 : AND2_NI
        port map( A => \66\, B => \59\, Y => \108\ );
U59 : AND2_NI
        port map( A => \59\, B => \78\, Y => \109\ );
U60 : BUF_INV_NI
        port map( Y => \64\, A => \21\ );
U61 : BUF_INV_NI
        port map( Y => \66\, A => \78\ );
U62 : AND2_NI
        port map( A => \69\, B => \58\, Y => \102\ );
U63 : AND2_NI
        port map( A => \58\, B => \20\, Y => \103\ );
U64 : AND2_NI
        port map( A => \68\, B => \58\, Y => \104\ );
U65 : AND2_NI
        port map( A => \58\, B => \75\, Y => \105\ );
U66 : BUF_INV_NI
        port map( Y => \69\, A => \20\ );
U67 : BUF_INV_NI
        port map( Y => \68\, A => \75\ );
U68 : AND2_NI
        port map( A => \70\, B => \57\, Y => \98\ );
U69 : AND2_NI
        port map( A => \57\, B => \19\, Y => \99\ );
U70 : AND2_NI
        port map( A => \71\, B => \57\, Y => \100\ );
U71 : AND2_NI
        port map( A => \57\, B => \74\, Y => \101\ );
U72 : BUF_INV_NI
        port map( Y => \70\, A => \19\ );
U73 : BUF_INV_NI
        port map( Y => \71\, A => \74\ );
U74 : AND2_NI
```

```
            port map( A => \72\, B => \55\, Y => \94\ );
U75 : AND2_NI
            port map( A => \55\, B => \18\, Y => \95\ );
U76 : AND2_NI
            port map( A => \73\, B => \55\, Y => \96\ );
U77 : AND2_NI
            port map( A => \55\, B => \63\, Y => \97\ );
U78 : BUF_INV_NI
            port map( Y => \72\, A => \18\ );
U79 : BUF_INV_NI
            port map( Y => \73\, A => \63\ );
U80 : AND2_NI
            port map( A => \82\, B => \83\, Y => \88\ );
U81 : AND2_NI
            port map( A => \83\, B => \89\, Y => \91\ );
U82 : AND2_NI
            port map( A => \84\, B => \83\, Y => \92\ );
U83 : AND2_NI
            port map( A => \83\, B => \87\, Y => \93\ );
U84 : BUF_INV_NI
            port map( Y => \82\, A => \89\ );
U85 : BUF_INV_NI
            port map( Y => \84\, A => \87\ );
U86 : FF_D_PCLR_CO_NI
            port map( D => \55\, Q => \57\, Qneg => open, CLR => \61\, CLK => \62\, PR => \61\ );
U87 : FF_D_PCLR_CO_NI
            port map( D => \57\, Q => \58\, Qneg => open, CLR => \61\, CLK => \62\, PR => \61\ );
U88 : FF_D_PCLR_CO_NI
            port map( D => \58\, Q => \59\, Qneg => open, CLR => \61\, CLK => \62\, PR => \61\ );
U89 : FF_D_PCLR_CO_NI
            port map( D => \59\, Q => \60\, Qneg => open, CLR => \61\, CLK => \62\, PR => \61\ );
U90 : FF_D_PCLR_CO_NI
            port map( D => \60\, Q => \53\, Qneg => open, CLR => \61\, CLK => \62\, PR => \61\ );
U91 : FF_D_PCLR_CO_NI
            port map( D => \61\, Q => \55\, Qneg => open, CLR => \61\, CLK => \62\, PR => \61\ );
```

```
U92 : FF_D_PCLR_CO_NI
        port map( D => \63\, Q => \74\, Qneg => open, CLR => \61\, CLK => \62\, PR => \61\ );
U93 : FF_D_PCLR_CO_NI
        port map( D => \74\, Q => \75\, Qneg => open, CLR => \61\, CLK => \62\, PR => \61\ );
U94 : FF_D_PCLR_CO_NI
        port map( D => \75\, Q => \78\, Qneg => open, CLR => \61\, CLK => \62\, PR => \61\ );
U95 : FF_D_PCLR_CO_NI
        port map( D => \87\, Q => \63\, Qneg => open, CLR => \61\, CLK => \62\, PR => \61\ );
U96 : FF_D_PCLR_CO_NI
        port map( D => \78\, Q => \76\, Qneg => open, CLR => \77\, CLK => \62\, PR => \80\ );
U97 : FF_D_PCLR_CO_NI
        port map( D => \76\, Q => \45\, Qneg => open, CLR => \77\, CLK => \62\, PR => \80\ );
U98 : FF_D_PCLR_CO_NI
        port map( D => \87\, Q => \81\, Qneg => open, CLR => \86\, CLK => \62\, PR => \86\ );
U99 : FF_D_PCLR_CO_NI
        port map( D => \81\, Q => \31\, Qneg => open, CLR => \86\, CLK => \62\, PR => \86\ );
U100 : FF_D_PCLR_CO_NI
        port map( D => \31\, Q => \85\, Qneg => open, CLR => \86\, CLK => \62\, PR => \86\ );
U101 : FF_D_PCLR_CO_NI
        port map( D => \85\, Q => \17\, Qneg => open, CLR => \86\, CLK => \62\, PR => \86\ );
U102 : XOR2_NI
        port map( A => \85\, B => \17\, Y => \87\ );
U103 : XOR2_NI
        port map( A => \81\, B => \31\, Y => \89\ );
U104 : DIGITAL_HIGH_NI
        port map( Y => \29\ );
U105 : DIGITAL_HIGH_NI
        port map( Y => \34\ );
U107 : DIGITAL_HIGH_NI
        port map( Y => \37\ );
U108 : DIGITAL_HIGH_NI
        port map( Y => \86\ );
U109 : DIGITAL_HIGH_NI
        port map( Y => open );
U110 : DIGITAL_HIGH_NI
```

```
                port map( Y => \61\ );

        U111 : DIGITAL_HIGH_NI

                port map( Y => \6\ );

end behavioral;
```

Group Members:

- *Harshil Soni – 15BCE1223*
- *Priju Gopi – 15BCE1308*
- *Muktak Pandya – 15BCE1263*
- *Abhishek Ghangas – 15BCE1178*