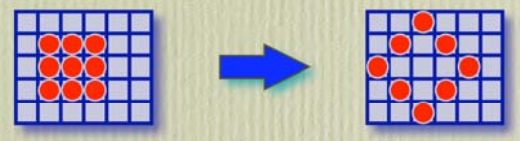
A hardwired Game of life

Eduardo Sanchez

Game of life

- Cellular automaton invented in 1970 by John Conway
  - Birth of a cell

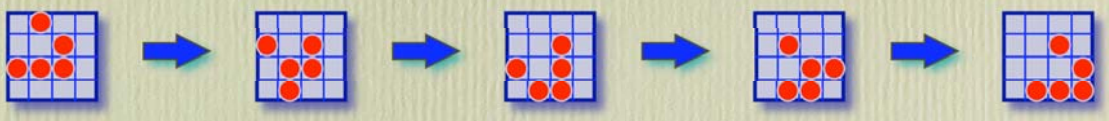
  - Three neighbors
  - Death of a cell



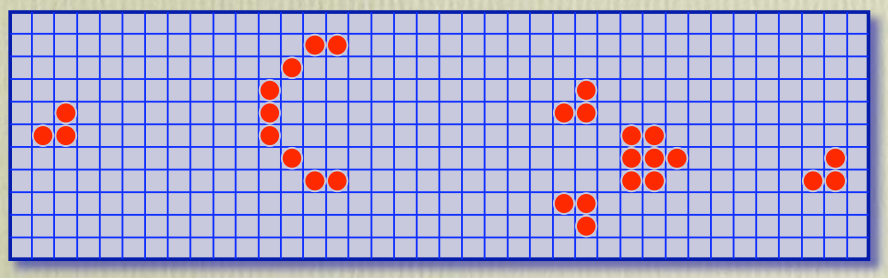
  - More than 3 neighbors
  - Less than 2 neighbors
  - Survival of a cell

  - Two or three neighbors

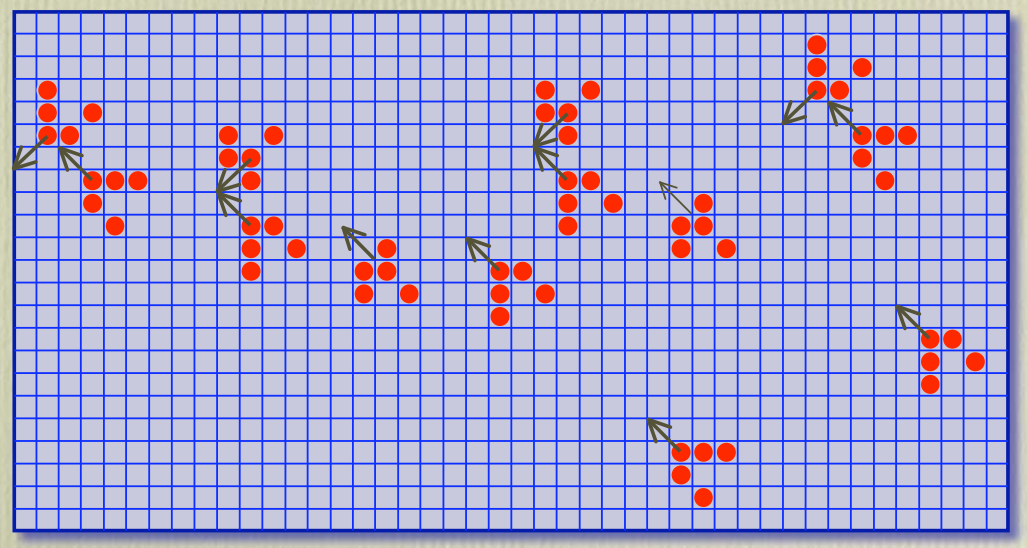
- A glider:



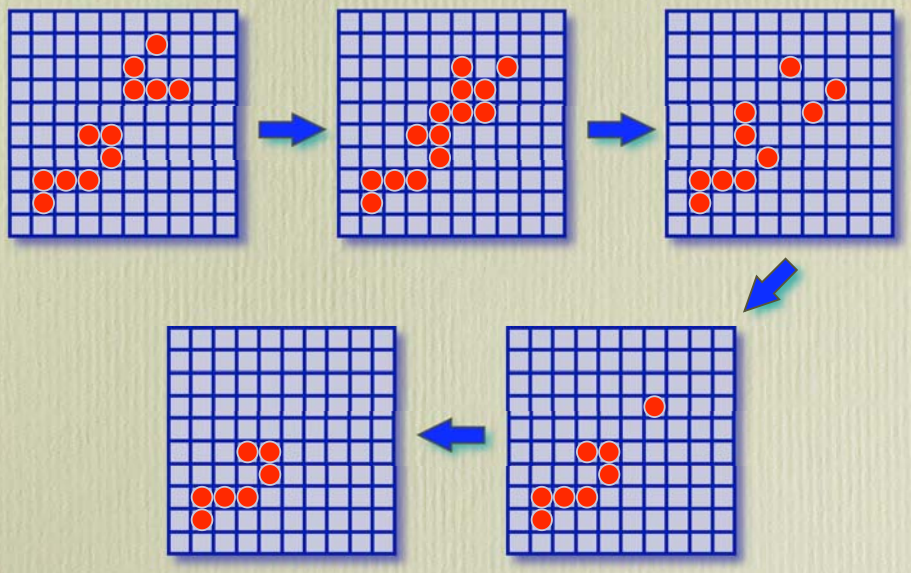
- A glider gun:



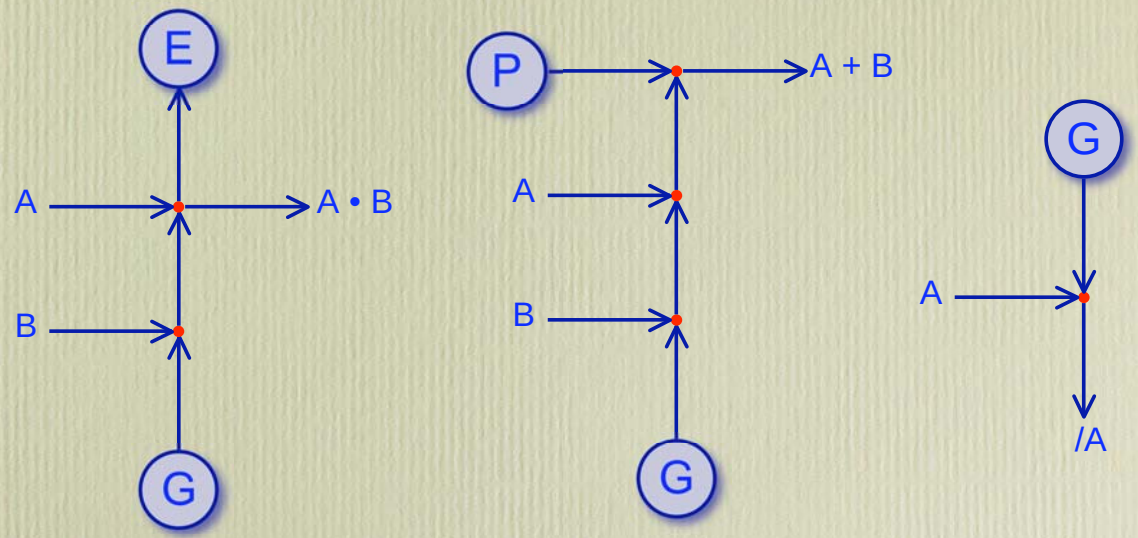
- A glider gun generator:



- A glider's eater:



- Logic gates:



```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

package vgapack is

  constant width          : integer := 8;
  constant maxWidthBit    : integer := width-1;
  constant maxColAddrBit  : integer := 2;   -- width = 2**(maxColAddrBit+1)
  constant height         : integer := 8;
  constant maxHeightBit   : integer := height-1;
  constant maxRowAddrBit  : integer := 2;   -- height = 2**(maxRowAddrBit+1)
  constant cellSize       : integer := 512/width;
  constant maxCellSizeBit : integer := 5;   -- size = 2**(maxCellSizeBit+1)

  type lineType is array(0 to maxWidthBit) of std_logic;
  type boardType is array(0 to maxHeightBit) of lineType;
```



```
component lifeCell
   port(clk, reset, firstValue     : in std_logic;
        nw, n, ne, e, se, s, sw, w : in std_logic;
        state                      : out std_logic);
 end component;

 component lifeRow
   port (clk, reset   : in std_logic;
         initialValue : in lineType;
         northIn      : in lineType;
         northOut     : out lineType;
         southIn      : in lineType;
         southOut     : out lineType;
         eastOut      : out std_logic;
         westOut      : out std_logic);
 end component;
```

```
 component lifeBoard
    port (clk, reset   : in std_logic;
          initialValue : in boardType;
          rowAddress   : in std_logic_vector(maxRowAddrBit downto 0);
          colAddress   : in std_logic_vector(maxColAddrBit downto 0);
          outCell      : out std_logic);
  end component;

  component vga
    port(clk, reset            : in std_logic;
         red, green, blue      : in std_logic;
         r, g, b, hsync, vsync : out std_logic;
         row                   : out std_logic_vector(8 downto 0);
         column                : out std_logic_vector(9 downto 0));
  end component;

end vgapack;
```



```
  library ieee;
  use ieee.std_logic_1164.all;
  use ieee.std_logic_arith.all;
  use ieee.std_logic_unsigned.all;

  entity vga is
    port(clk, reset            : in std_logic;
         red, green, blue      : in std_logic;
         r, g, b, hsync, vsync : out std_logic;
         row                   : out std_logic_vector(8 downto 0);
         column                : out std_logic_vector(9 downto 0));
  end vga;

  architecture synt of vga is

    signal videoon, videov, videoh : std_logic;
    signal hcount, vcount          : std_logic_vector(9 downto 0);
```

```
begin

hcounter: process (clk, reset)
          begin
          if reset='1'
            then hcount <= (others => '0');
            else if (clk'event and clk='1')
                   then if hcount=799
                          then hcount <= (others => '0');
                          else hcount <= hcount + 1;
                        end if;
                 end if;
          end if;
          end process;

          process (hcount)
          begin
          videoh <= '1';
          column <= hcount;
          if hcount>639
            then videoh <= '0';
                 column <= (others => '0');
          end if;
          end process;
```

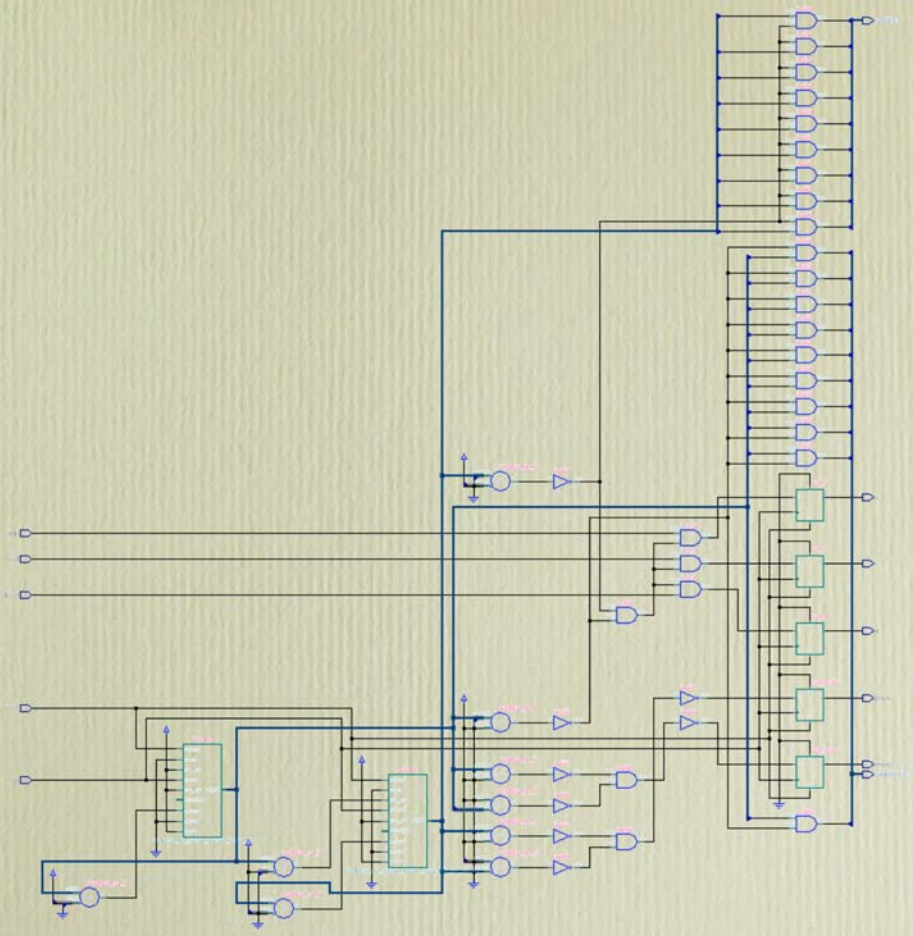


```
vcounter: process (clk, reset)
          begin
          if reset='1'
            then vcount <= (others => '0');
            else if (clk'event and clk='1')
                   then if hcount=699
                          then if vcount=524
                                 then vcount <= (others => '0');
                                 else vcount <= vcount + 1;
                               end if;
                        end if;
                 end if;
          end if;
          end process;

          process (vcount)
          begin
          videov <= '1';
          row <= vcount(8 downto 0);
          if vcount>479
            then videov <= '0';
                 row <= (others => '0');
          end if;
          end process;
```

```
sync: process (clk, reset)
      begin
      if reset='1'
        then hsync <= '0';
             vsync <= '0';
        else if (clk'event and clk='1')
               then if (hcount<=755 and hcount>=659)
                      then hsync <= '0';
                      else hsync <= '1';
                    end if;
                    if (vcount<=494 and vcount>=493)
                      then vsync <= '0';
                      else vsync <= '1';
                    end if;
             end if;
      end if;
      end process;

videoon <= videoh and videov;
```

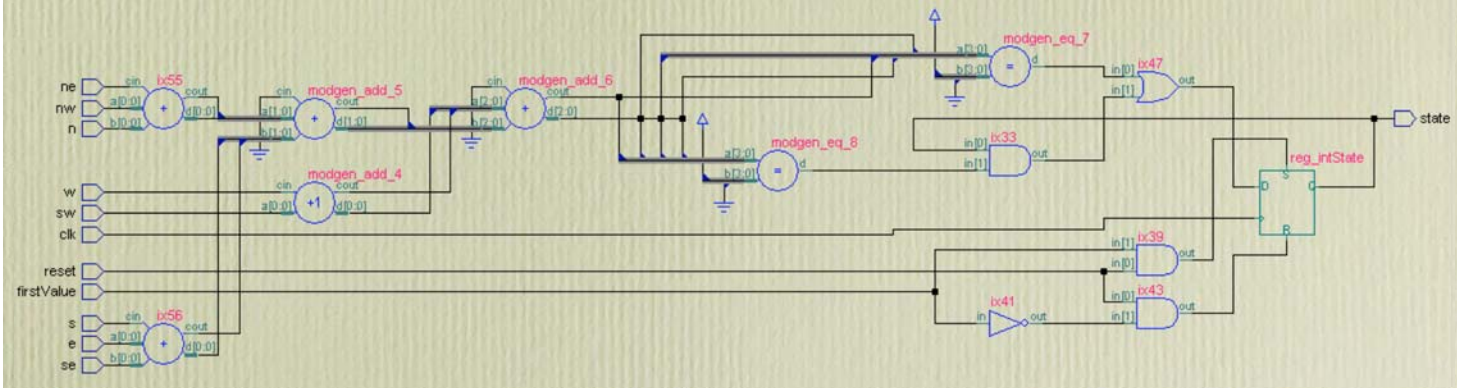


```
     colors: process (clk, reset)
             begin
             if reset='1'
               then r <= '0';
                    g <= '0';
                    b <= '0';
               elsif (clk'event and clk='1')
                 then r <= red and videoon;
                      g <= green and videoon;
                      b <= blue and videoon;
             end if;
             end process;

     end synt;
```




```
library ieee;
use  ieee.std_logic_1164.all;
use  ieee.std_logic_unsigned.all;

entity lifeCell is
  port(clk, reset, firstValue     : in std_logic;
       nw, n, ne, e, se, s, sw, w : in std_logic;
       state                      : out std_logic);
end lifeCell;

architecture synt of lifeCell is

  signal temp1      : std_logic_vector(1 downto 0);
  signal temp2      : std_logic_vector(1 downto 0);
  signal temp3      : std_logic_vector(1 downto 0);
  signal temp4      : std_logic_vector(2 downto 0);
  signal sum        : std_logic_vector(3 downto 0);
  signal intState   : std_logic;
  signal nextState  : std_logic;

begin

temp1 <= ('0' & nw) + ('0' & n) + ('0' & ne);
temp2 <= ('0' & e) + ('0' & se) + ('0' & s);
temp3 <= ('0' & sw) + ('0' & w);
temp4 <= ('0' & temp1) + ('0' & temp2);
sum <= ("00" & temp3) + ('0' & temp4);
```

```
process (intState, sum)
begin
nextState <= '0';
if sum="0011"
  then nextState <= '1';
  elsif (intState='1' and sum="0010")
    then nextState <= '1';
end if;
end process;

process (clk, reset, firstValue)
begin
if reset='1'
  then intState <= firstValue;
  elsif (clk'event and clk='1')
    then intState <= nextState;
end if;
end process;

state <= intState;

end synt;
```

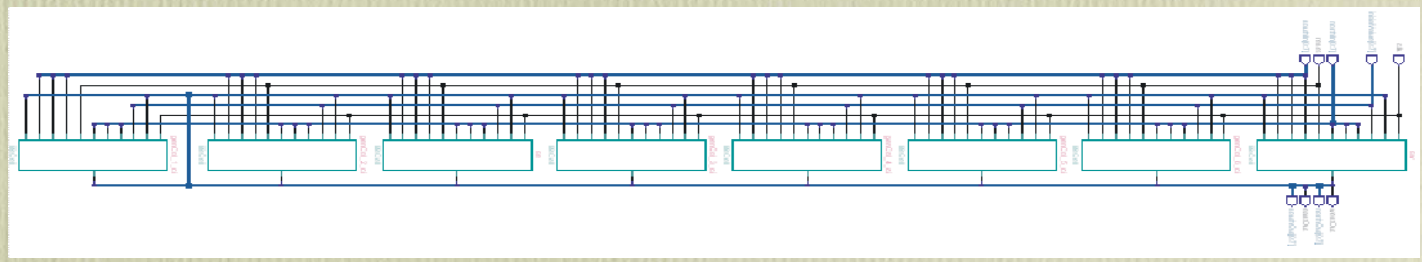



```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

use work.vgapack.all;

entity lifeRow is
  port (clk, reset   : in std_logic;
        initialValue : in lineType;
        northIn      : in lineType;
        northOut     : out lineType;
        southIn      : in lineType;
        southOut     : out lineType;
        eastOut      : out std_logic;
        westOut      : out std_logic);
end lifeRow;

architecture synt of lifeRow is

  signal state : lineType;

begin
northOut <= state;
southOut <= state;
westOut <= state(0);
eastOut <= state(maxWidthBit);
```



```
cw: lifeCell port map (clk => clk,
                       reset => reset,
                       firstValue => initialValue(0),
                       nw => northIn(maxWidthBit),
                       n => northIn(0),
                       ne => northIn(1),
                       e => state(1),
                       se => southIn(1),
                       s => southIn(0),
                       sw => southIn(maxWidthBit),
                       w => state(maxWidthBit),
                       state => state(0));
genCol: for i in 1 to maxWidthBit-1 generate
          ci: lifeCell port map (clk => clk,
                                 reset => reset,
                                 firstValue => initialValue(i),
                                 nw => northIn(i-1),
                                 n => northIn(i),
                                 ne => northIn(i+1),
                                 e => state(i+1),
                                 se => southIn(i+1),
                                 s => southIn(i),
                                 sw => southIn(i-1),
                                 w => state(i-1),
                                 state => state(i));
        end generate;
```

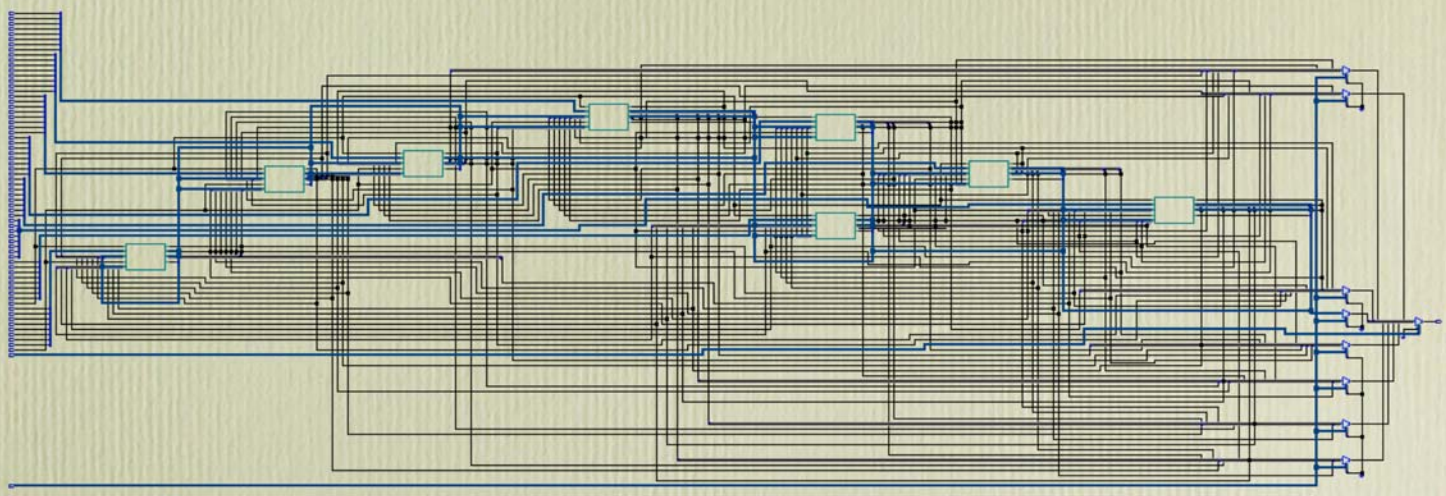
```
ce: lifeCell port map (clk => clk,
                       reset => reset,
                       firstValue => initialValue(maxWidthBit),
                       nw => northIn(maxWidthBit-1),
                       n => northIn(maxWidthBit),
                       ne => northIn(0),
                       e => state(0),
                       se => southIn(0),
                       s => southIn(maxWidthBit),
                       sw => southIn(maxWidthBit-1),
                       w => state(maxWidthBit-1),
                       state => state(maxWidthBit));

end synt;
```

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use work.vgapack.all;

entity lifeBoard is
  port (clk, reset   : in std_logic;
        initialValue : in boardType;
        rowAddress   : in std_logic_vector(maxRowAddrBit downto 0);
        colAddress   : in std_logic_vector(maxColAddrBit downto 0);
        outCell      : out std_logic);
end lifeBoard;

architecture synt of lifeBoard is

  signal board : boardType;

begin
l0: lifeRow port map (clk => clk,
                      reset => reset,
                      initialValue => initialValue(0),
                      northIn => board(maxHeightBit),
                      northOut => board(0),
                      southIn => board(1),
                      southOut => board(0),
                      eastOut => board(0)(maxWidthBit),
                      westOut => board(0)(0));
```



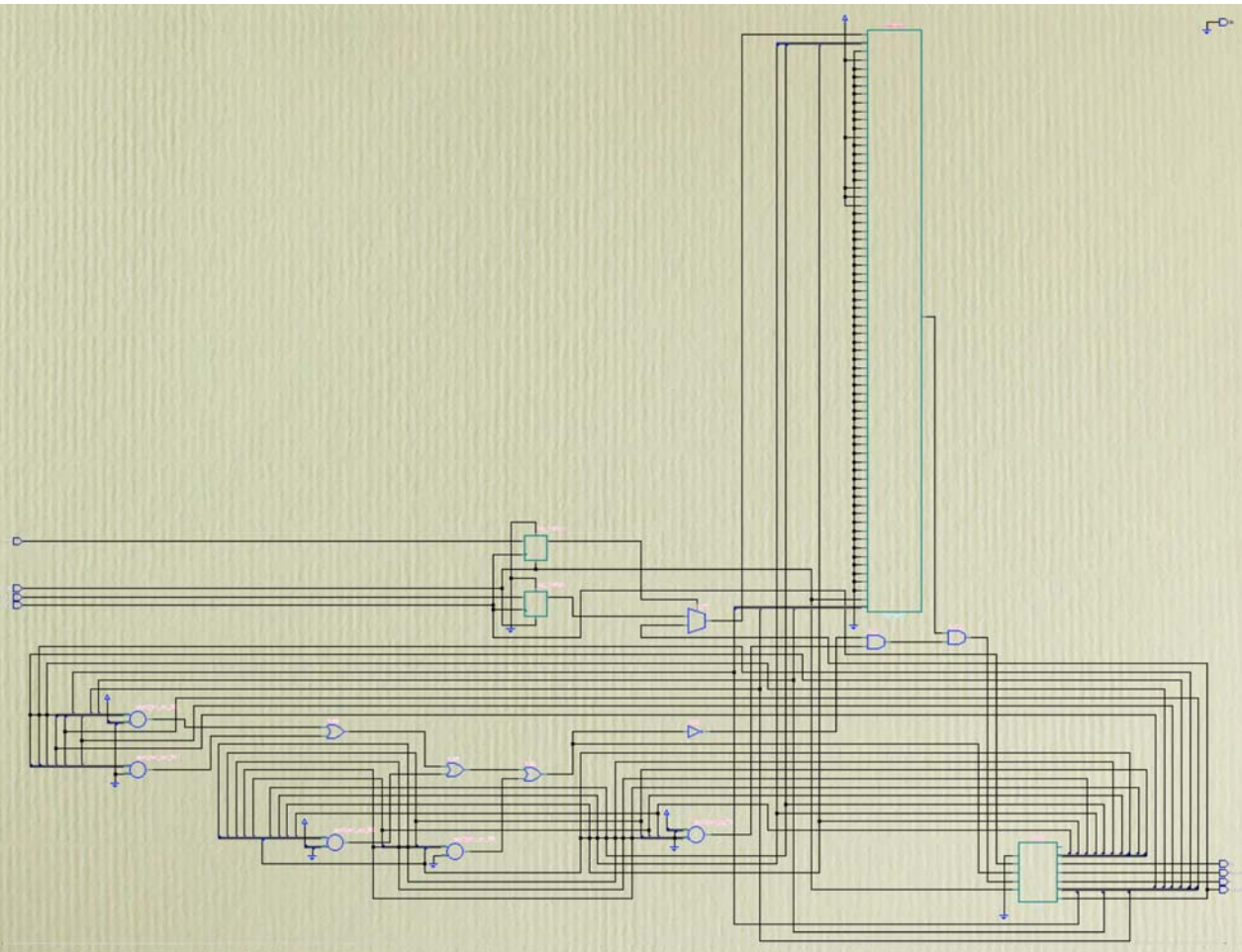
```
genLine: for i in 1 to maxHeightBit-1 generate
           li: lifeRow port map (clk => clk,
                                 reset => reset,
                                 initialValue => initialValue(i),
                                 northIn => board(i-1),
                                 northOut => board(i),
                                 southIn => board(i+1),
                                 southOut => board(i),
                                 eastOut => board(i)(maxWidthBit),
                                 westOut => board(i)(0));
         end generate;
lmax: lifeRow port map (clk => clk,
                        reset => reset,
                        initialValue => initialValue(maxHeightBit),
                        northIn => board(maxHeightBit-1),
                        northOut => board(maxHeightBit),
                        southIn => board(0),
                        southOut => board(maxHeightBit),
                        eastOut => board(maxHeightBit)(maxWidthBit),
                        westOut => board(maxHeightBit)(0));

outCell <= board(conv_integer(rowAddress))(conv_integer(colAddress));

end synt;
```




```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

use work.vgapack.all;

entity GameOfLife is
  port (clk, reset, run, step : in std_logic;
        r, g, b, hsync, vsync : out std_logic);
end GameOfLife;

architecture synt of GameOfLife is

  signal cellColor  : std_logic;
  signal intRed     : std_logic;
  signal intGreen   : std_logic;
  signal rowAddress : std_logic_vector(8 downto 0);
  signal colAddress : std_logic_vector(9 downto 0);
  signal lifeClk    : std_logic;
  signal intVsync   : std_logic;
  signal intRun     : std_logic;
  signal intStep    : std_logic;
  signal limit      : std_logic;
  signal greenLine  : std_logic;
  signal pattern    : boardType;
```

```
begin

pattern(0) <= "01000000";
pattern(1) <= "00100000";
pattern(2) <= "11100000";
pattern(3) <= "00000000";
pattern(4) <= "00000000";
pattern(5) <= "00000000";
pattern(6) <= "00000000";
pattern(7) <= "00000000";

process (clk, reset)
begin
if reset='1'
  then intRun <= '0';
       intStep <= '0';
  elsif (clk'event and clk='1')
    then intRun <= run;
         intStep <= step;
end if;
end process;
```



```
uVga: vga port map (clk => clk,
                    reset => reset,
                    red => intRed,
                    green => intGreen,
                    blue => '0',
                    r => r,
                    g => g,
                    b => b,
                    hsync => hsync,
                    vsync => intVsync,
                    row => rowAddress,
                    column => colAddress);

vsync <= intVsync;
```

```
lifeClk <= intVsync when intRun='1'
             else intStep;

uBoard: lifeBoard port map (clk => lifeClk,
                            reset => reset,
                            initialValue => pattern,
                            rowAddress => rowAddress(8 downto 8-maxRowAddrBit),
                            colAddress => colAddress(8 downto 8-maxColAddrBit),
                            outCell => cellColor);

limit <= '1' when colAddress>=512 else '0';
greenLine <= '1' when (rowAddress=479 or rowAddress(maxCellSizeBit downto 0)=0
                       or colAddress=639 or colAddress(maxCellSizeBit downto
0)=0)
                  else '0';
intRed <= cellColor when (limit='0' and greenLine='0') else '0';
intGreen <= '1' when greenLine='1' else '0';

end synt;
```