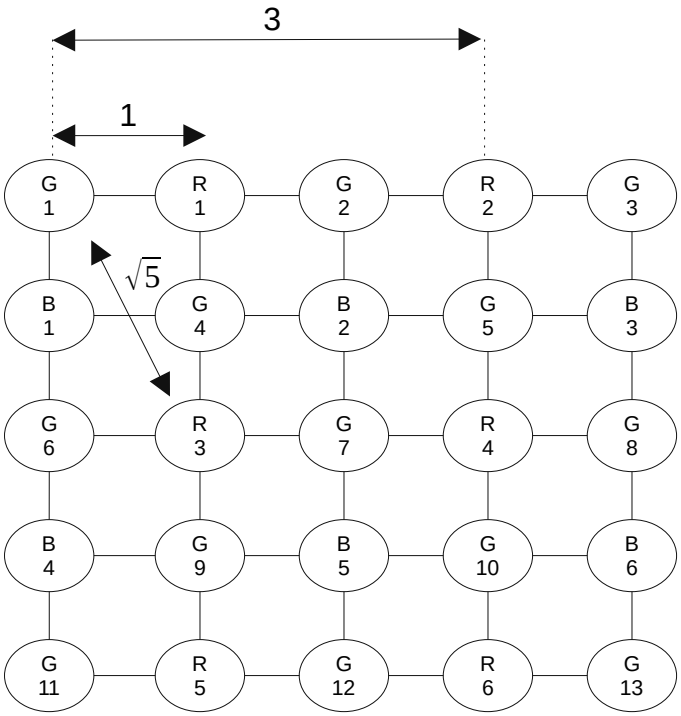


$$\frac{1}{1+\sqrt{5}+3}+\frac{\sqrt{5}}{1+\sqrt{5}+3}+\frac{3}{1+\sqrt{5}+3}=\frac{1+\sqrt{5}+3}{1+\sqrt{5}+3}=1$$

0.48107 0.35857 0.16035

$$\overline{R}=a_1\cdot\overline{R_1}+a_2\cdot\overline{R_2}+a_3\cdot\overline{R_3}$$

$$R^*=\overline{a_1}\cdot R_1+\overline{a_2}\cdot R_2+\overline{a_3}\cdot R_3$$

R * reads as R dual

$$R=0.48107\cdot\left(r_7^1\cdot 2^7+\ldots+r_0^1\cdot 2^0\right)+0.35857\cdot\left(r_7^3\cdot 2^7+\ldots+r_0^3\cdot 2^0\right)+0.16035\cdot\left(r_7^2\cdot 2^7+\ldots+r_0^2\cdot 2^0\right)$$

$$\overline{R}=0.48107\cdot\begin{pmatrix}r_7^1\\r_6^1\\r_5^1\\r_5^1\\r_4^1\\r_3^1\\r_2^1\\r_1^1\\r_0^1\end{pmatrix}+0.35857\cdot\begin{pmatrix}r_7^3\\r_6^3\\r_5^3\\r_5^3\\r_4^3\\r_3^3\\r_2^3\\r_1^3\\r_0^3\end{pmatrix}+0.16035\cdot\begin{pmatrix}r_7^2\\r_6^2\\r_5^2\\r_5^2\\r_4^2\\r_3^2\\r_2^2\\r_1^2\\r_0^2\end{pmatrix}$$

$$R^*=r_7^1r_6^1r_5^1r_4^1r_3^1r_2^1r_1^1r_0^1r_7^3r_6^3r_5^3r_4^3r_3^3r_2^3r_1^3r_0^3r_7^2r_6^2r_5^2r_4^2r_3^2r_2^2r_1^2r_0^2 \quad \text{call it} \quad R_{BIG}$$

Ex1 Px0, three surrounding red pixels

$$R_1=128_{10}=10000000_2$$
$$R_2=53_{10}=00110101_2$$
$$R_3=127_{10}=01111111_2$$

$$R_{BIG}=R_1R_3R_2=\underline{10000000}\underline{01111111}\underline{00110101},$$

24bit

$$R_{IDEAL}=0.48\cdot 128+0.36\cdot 127+0.16\cdot 53=115.64=01110011_2$$

$$R=10000100_2=132$$ not good approx.

■

Ex2 Px0, three surrounding red pixels

$R_1 = 128_{10} = 10000000_2$

$R_2 = 53_{10} = 00110101_2$

$R_3 = 127_{10} = 01111111_2$

$$R_{IDEAL} = 0.48 \cdot 128 + 0.36 \cdot 127 + 0.16 \cdot 53 = 115.64 = 01110011_2$$

Let $R_{12} = \overline{(R_1 >>_1 2) \oplus (R_2 <<_1 1)} = 01110100_2$,

where $>>_1$ and $<<_1$ - logical shift with '1' fill

$\oplus$ - XNOR boolean operator

- better approx. ■

Let $R = r_7 r_6 r_5 r_4 r_3 r_2 r_1 r_0$

Define Я as: $Я = r_4 r_5 r_6 r_7 r_0 r_1 r_2 r_3$ And define reversal operator Я as: $Я(R) = Я$

Ex3 From Ex2 $R_{12} = 01110100_2$

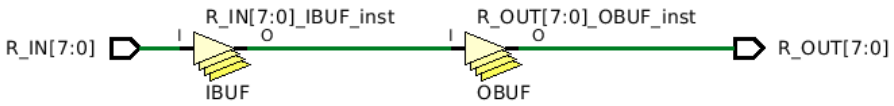
Let $Я_{12} = 11100010_2$

$\overline{Я_{12} \oplus (R_3 <<_1 1)} = 11100010_2$

$Я(\overline{Я_{12} \oplus (R_3 <<_1 1)}) = 01110100_2 = R_{12}$ - identity operator ■

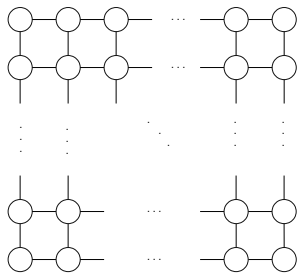
reverser_8b.vhd

```
entity reverser_8b is
  Port ( R_IN : in std_logic_vector(7 downto 0);
    R_OUT : out std_logic_vector(7 downto 0));
end reverser_8b;

architecture Dataflow of reverser_8b is

begin
   R_OUT(7) <= R_IN(4);
   R_OUT(6) <= R_IN(5);
   R_OUT(5) <= R_IN(6);
   R_OUT(4) <= R_IN(7);
   R_OUT(3) <= R_IN(0);
   R_OUT(2) <= R_IN(1);
   R_OUT(1) <= R_IN(2);
   R_OUT(0) <= R_IN(3);
end Dataflow;
```



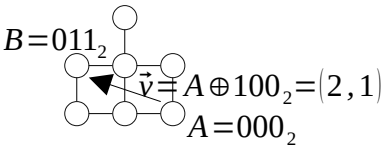
Recall $R^* = r_7^1 r_6^1 r_5^1 r_4^1 r_3^1 r_2^1 r_1^1 r_0^1 r_7^3 r_6^3 r_5^3 r_4^3 r_3^3 r_2^3 r_1^3 r_0^3 r_7^2 r_6^2 r_5^2 r_4^2 r_3^2 r_2^2 r_1^2 r_0^2$ call it R_{BIG}

Therefore R_{BIG} is a finite space and dual space R^* can be viewed as a rectangle or a plane consisting of 2^{24} elements in it arranged in a lattice pattern.



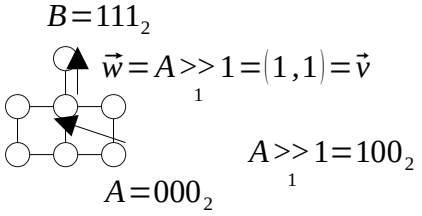
Thus we can define our binary operations such as AND,OR, etc as two different things:
1) Means of traversing the grid by flipping some of the bits
2) vectors
We choose one boolean operator to measure similarity and go from there. Let it be XNOR. Other operations will have to do with adjusting weights as we arrived here from the weighted sum and the end goal is weighted average.

Ex4

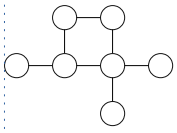


■

Ex5



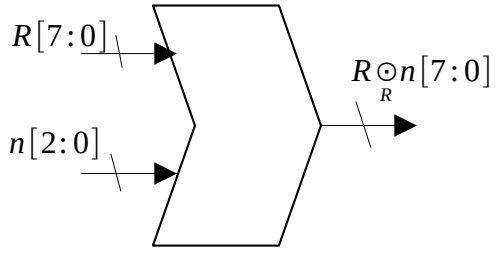
Therefore $\vec{v}, \vec{w}$ are orthogonal (hint from linear algebra) and these two operations (right shift with '1' fill and XNOR) span the entire lattice. ■



Can try different geometries/origin also. Actual notion of distance in this space may depend on the amount of such binary operations required to arrive from origin to any other point. Remember that we are looking for a point in this grid, that surely does exist, that is the closest approximation to the weighted average. Also remember once we contrived this space to be linear and dual by assuming 0.48 0.36 and 0.16 real numbers are independent (orthogonal) thus their products are equal to 0, all normal algebra went out the window.

Let $R = r_7 r_6 r_5 r_4 r_3 r_2 r_1 r_0$

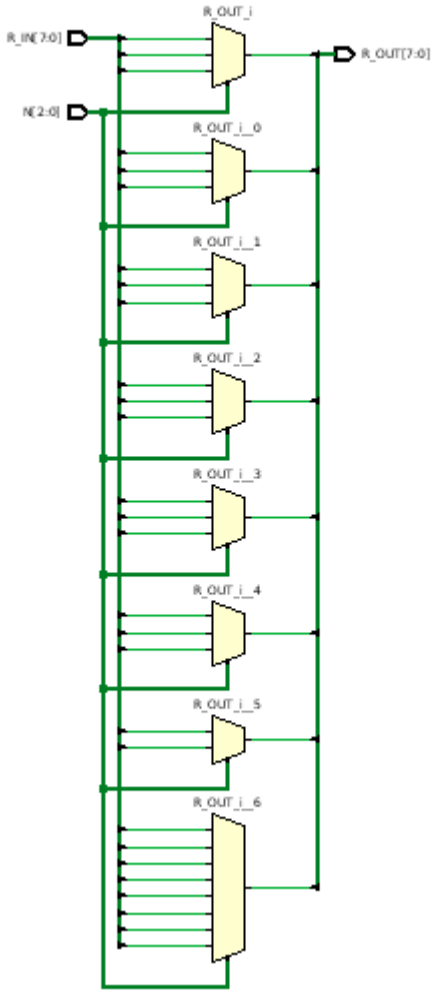
Define RPUSH operator as:



where n – control signal choosing which bit to push to the top position

rpush_8b.vhd

```
entity rpushn_8b is
  Port ( R_IN : in std_logic_vector(7 downto 0);
    N : in std_logic_vector(2 downto 0);
    R_OUT : out std_logic_vector(7 downto 0));
end rpushn_8b;

architecture Dataflow of rpushn_8b is

begin
     with N select
        R_OUT(7) <= R_IN(0) when "000",
          R_IN(1) when "001",
          R_IN(2) when "010",
          R_IN(3) when "011",
          R_IN(4) when "100",
          R_IN(5) when "101",
          R_IN(6) when "110",
          R_IN(7) when others;
      with N select
        R_OUT(6) <= R_IN(6) when "111",
          R_IN(7) when others;
      with N select
        R_OUT(5) <= R_IN(5) when "111",
          R_IN(5) when "110",
          R_IN(6) when others;
      with N select
        R_OUT(4) <= R_IN(4) when "111",
          R_IN(4) when "101",
          R_IN(5) when others;
      with N select
        R_OUT(3) <= R_IN(3) when "111",
          R_IN(3) when "100",
          R_IN(4) when others;
      with N select
        R_OUT(2) <= R_IN(2) when "111",
          R_IN(2) when "011",
          R_IN(3) when others;
      with N select
        R_OUT(1) <= R_IN(1) when "111",
          R_IN(1) when "010",
          R_IN(2) when others;
      with N select
        R_OUT(0) <= R_IN(0) when "111",
          R_IN(0) when "001",
          R_IN(1) when others;
end Dataflow;
```

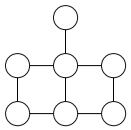


Ex6

Let $R = r_7 r_6 r_5 r_4 r_3 r_2 r_1 r_0$

$R \underset{R}{\odot} 3 = r_3 r_7 r_6 r_5 r_4 r_2 r_1 r_0$ ■

Ex7



$\vec{u} = A \underset{R}{\odot} 0 = \vec{0}$

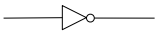
$A = 000_2$

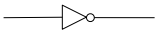
RPUSH operator clearly doesn't do anything for the origin ■

Bilinear: $f: V \times V \rightarrow U$

Basis can also be formed based on a notion of:
1) Sum of flips
2) Sum of shifts

It can be proven that a flip operation such as XNOR contains 0 shifts and a shift operation such as $\underset{1}{>>}$ contains 0 flips. Henceforth operators with this quality constitute a basis.

Square root $\sqrt{\ }$ = inverse = 

Square ^{2} = inverse = 

$|\vec{x}| = id(\vec{x}) = \vec{x}$ - each vector is a measure itself

=> $\vec{x} \cdot \vec{x} = inverse(\vec{x})$ - dot product as we expected.

Let $R_1 = r_7^1 r_6^1 r_5^1 r_4^1 r_3^1 r_2^1 r_1^1 r_0^1$, $R_2 = r_7^2 r_6^2 r_5^2 r_4^2 r_3^2 r_2^2 r_1^2 r_0^2$, $R_3 = r_7^3 r_6^3 r_5^3 r_4^3 r_3^3 r_2^3 r_1^3 r_0^3$

Define Diode as Bilinear function $f: R^* \times R^* \rightarrow A$

$$Diode(R_1, R_2) = \begin{matrix} r_7^1 r_6^1 r_1^1 r_0^1 \\ r_5^1 r_4^1 r_3^1 r_2^1 \\ r_5^2 r_4^2 r_3^2 r_2^2 \\ r_7^2 r_6^2 r_1^2 r_0^2 \end{matrix}$$

Define triode as Trilinear function: $f: R^* \times R^* \times R^* \rightarrow T$

$$Triode(R_1, R_2, R_3) = \begin{matrix} r_7^1 r_6^1 r_3^3 r_4^3 r_1^1 r_0^1 \\ r_5^1 r_4^1 r_2^3 r_5^3 r_3^1 r_2^1 \\ r_5^2 r_4^2 r_1^3 r_6^3 r_3^2 r_2^2 \\ r_7^2 r_6^2 r_0^3 r_7^3 r_1^2 r_0^2 \end{matrix} \quad \text{or} \quad Triode(R_1, R_2, R_3) = \begin{matrix} r_7^1 r_6^1 r_3^3 r_4^3 r_7^3 r_0^3 r_1^1 r_0^1 \\ r_5^1 r_4^1 r_2^3 r_5^3 r_6^3 r_1^3 r_3^1 r_2^1 \\ r_5^2 r_4^2 r_1^3 r_6^3 r_5^3 r_2^3 r_3^2 r_2^2 \\ r_7^2 r_6^2 r_0^3 r_7^3 r_4^3 r_3^3 r_1^2 r_0^2 \end{matrix}$$

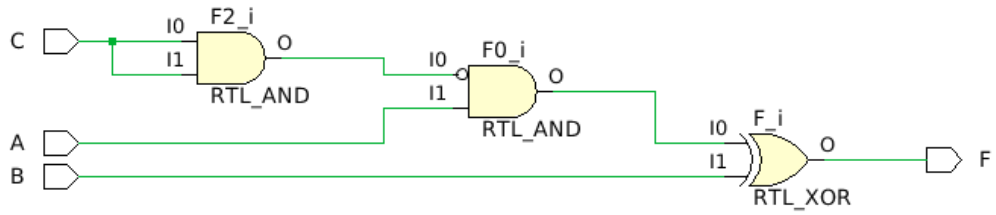
compute sum of flips and sum of shifts.

Cross-product = origin

$dim = \chi = A \oplus B$,if C = '0'
$= \overline{A \oplus B}$,else

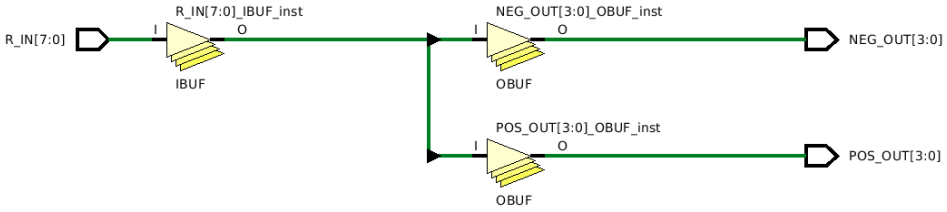
dim_1b.vhd

```
entity dim_1b is
  Port ( A : in std_logic;
    B : in std_logic;
    C : in std_logic;
    F: out std_logic);
end dim_1b;

architecture Dataflow of dim_1b is

begin
  F <= ((C NAND C) AND A) XOR B;

end Dataflow;
```



diode_8b.vhd

```
entity diode_8b is
  Port ( R_IN : in std_logic_vector(7 downto 0);
  POS_OUT : out std_logic_vector(3 downto 0);
  NEG_OUT : out std_logic_vector(3 downto 0));
end diode_8b;

architecture Dataflow of diode_8b is

begin
  POS_OUT(3) <= R_IN(7);
  POS_OUT(2) <= R_IN(6);
  POS_OUT(1) <= R_IN(1);
  POS_OUT(0) <= R_IN(0);
  NEG_OUT(3) <= R_IN(5);
  NEG_OUT(2) <= R_IN(4);
  NEG_OUT(1) <= R_IN(3);
  NEG_OUT(0) <= R_IN(2);

end Dataflow;
```



triode_8b.vhd

```
entity triode_8b is
  Port ( R1_IN : in std_logic_vector(7 downto 0);
  R2_IN : in std_logic_vector(7 downto 0);
  R3_IN : in std_logic_vector(7 downto 0);
  T1_OUT : out std_logic_vector(7 downto 0);
  T2_OUT : out std_logic_vector(7 downto 0);
  T3_OUT : out std_logic_vector(7 downto 0);
  T4_OUT : out std_logic_vector(7 downto 0));
end triode_8b;

architecture Dataflow of triode_8b is

begin
   T1_OUT(7) <= R1_IN(7);
   T1_OUT(6) <= R1_IN(6);
   T1_OUT(5) <= R3_IN(3);
   T1_OUT(4) <= R3_IN(4);
   T1_OUT(3) <= R3_IN(7);
   T1_OUT(2) <= R3_IN(0);
   T1_OUT(1) <= R1_IN(1);
   T1_OUT(0) <= R1_IN(0);

    T2_OUT(7) <= R1_IN(5);
    T2_OUT(6) <= R1_IN(4);
    T2_OUT(5) <= R3_IN(2);
    T2_OUT(4) <= R3_IN(5);
    T2_OUT(3) <= R3_IN(6);
    T2_OUT(2) <= R3_IN(1);
    T2_OUT(1) <= R1_IN(3);
    T2_OUT(0) <= R1_IN(2);

    T3_OUT(7) <= R2_IN(5);
    T3_OUT(6) <= R2_IN(4);
    T3_OUT(5) <= R3_IN(1);
    T3_OUT(4) <= R3_IN(6);
    T3_OUT(3) <= R3_IN(5);
    T3_OUT(2) <= R3_IN(2);
    T3_OUT(1) <= R2_IN(3);
    T3_OUT(0) <= R2_IN(2);

    T4_OUT(7) <= R2_IN(7);
    T4_OUT(6) <= R2_IN(6);
    T4_OUT(5) <= R3_IN(0);
    T4_OUT(4) <= R3_IN(7);
    T4_OUT(3) <= R3_IN(4);
    T4_OUT(2) <= R3_IN(3);
    T4_OUT(1) <= R2_IN(1);
    T4_OUT(0) <= R2_IN(0);

end Dataflow;
```