Problem Statement

For computer vision, it is sometimes necessary to calculate the length of a vector. Hardware implementation of such a calculation will enable significant performance improvement over a software approach.

We note that the length of the vector from the origin to the point \((x,y)\) is defined by the Pythagorean theorem: 
\[ h = \sqrt{x^2 + y^2} \]

Hardware Block Diagram

Here we are computing the sum of squares and then using a lookup table to compute the square root of the sum.

Controller State machine

- Start
- Select Y, store result
- Select X
- Add Squares
- Get SQRT
entity VectorLength is
  port(XIn: in integer :=0;
       YIn: in integer:=0;
       Distance: out integer:=0);
--
end VectorLength;

architecture behavior of VectorLength is

  signal sel:bit :='0';   --Initialize variables
  signal isq: integer:=0;
  signal ysq: integer:=0;
  signal renable: bit:='0';
  signal sqsum: integer:=0;

  TYPE LU_TABLE is array(0 to 199) of real;

  --Implement the LUT for square roots less than 14.12
  CONSTANT SQRT: LU_TABLE := (1.0,1.414213562,1.732050808,
                              2.0,2.23606777,2.449489743,2.645751311,2.828427125,...
                              14.0,14.03566885,14.07124728,14.073598,14.14213562);

  begin

  My_mux: process (sel,renable)           --Here is the register storage
    Begin                                --and squaring operation
      if sel='0' then
        isq <= xin *xin;
      elsif sel = '1' then
        isq <= Yin *Yin;
      end if;

      if Renable='1' then
        ysq <= isq;
      elsif Renable='0' then
        ysq <= 0;
      end if;

    end process my_mux;

  my_controller : process               --Here is the state machine
    constant interval: time :=1 ns;
    begin
      sel <= '0';
      renable <= '0';
      wait for interval;
      sel <='1';
      renable <= '1';
      wait for interval;
      sqsum<=YSq +Isq;
      Distance <= integer(sqrt(sqsum));

    end process my_controller ;

  end architecture behavior;
Results

As we can see from the timing diagram, the circuit computes the distance of a vector (to the nearest integer).

<table>
<thead>
<tr>
<th>/vec</th>
<th>X</th>
<th>Y</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>11</td>
<td>7</td>
<td>9</td>
</tr>
</tbody>
</table>

The first two lines are the X and Y inputs, and the third line is the distance to the origin.

Discussion

This implementation contains tradeoffs. A lookup table is fast for computing the square root, but takes up more memory as the size of the image is increased. We also see that there is some initial glitching in the circuit, which may be able to be refined.