

CORDIC_ALGO. vhd

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

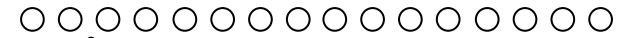
ENTITY testbench IS
END testbench;

ARCHITECTURE behavior OF testbench IS
signal x, y, z : Real;
BEGIN
  tb : PROCESS
    -- variable x, y, z : Real;
    variable delta, alpha : Real;
    variable x0, y0 : Real;
    variable theta : Real;
    variable xnext, ynext, znext : Real;
  BEGIN
    theta := 0.7854; -- pi/4 , 45degree
    x0 := 0.6073;
    y0 := 0.0;
    x <= x0; y <= y0; z <= 0.0;
    for I in 0 to 13 loop
      wait for 10 ns;
      -- alpha, delta
      case I is
        when 0 => alpha := 0.7854; delta := 1.0;
        when 1 => alpha := 0.4636; delta := 0.5;
        when 2 => alpha := 0.2450; delta := 0.25;
        when 3 => alpha := 0.1244; delta := 0.125;
        when 4 => alpha := 0.0624; delta := 0.0625;
        when 5 => alpha := 0.0312; delta := 0.03125;
        when 6 => alpha := 0.0156; delta := 0.015625;
        when 7 => alpha := 0.0078; delta := 0.0078125;
        when 8 => alpha := 0.0039; delta := 0.00390625;
        when 9 => alpha := 0.0020; delta := 0.001953125;
        when 10 => alpha := 0.00097656; delta := 0.00097560;
        when 11 => alpha := 0.00048828; delta := 0.00048828;
        when 12 => alpha := 0.00024414; delta := 0.00024414;
        when others => alpha := 0.00012207; delta := 0.00012207;
      end case;
      -- z > theta HIKAKU
      if z > theta then
        delta := - delta; alpha := - alpha;
      end if;
      -- CORDIC COMPUTATION
      xnext := x - delta * y;
      ynext := y + delta * x;
      znext := z + alpha;
      --
      x <= xnext; y <= ynext; z <= znext;
    end loop;
    wait; -- will wait forever
  END PROCESS tb;
END;

```

CORDIC 回路 タイミングダイアグラム ver.2

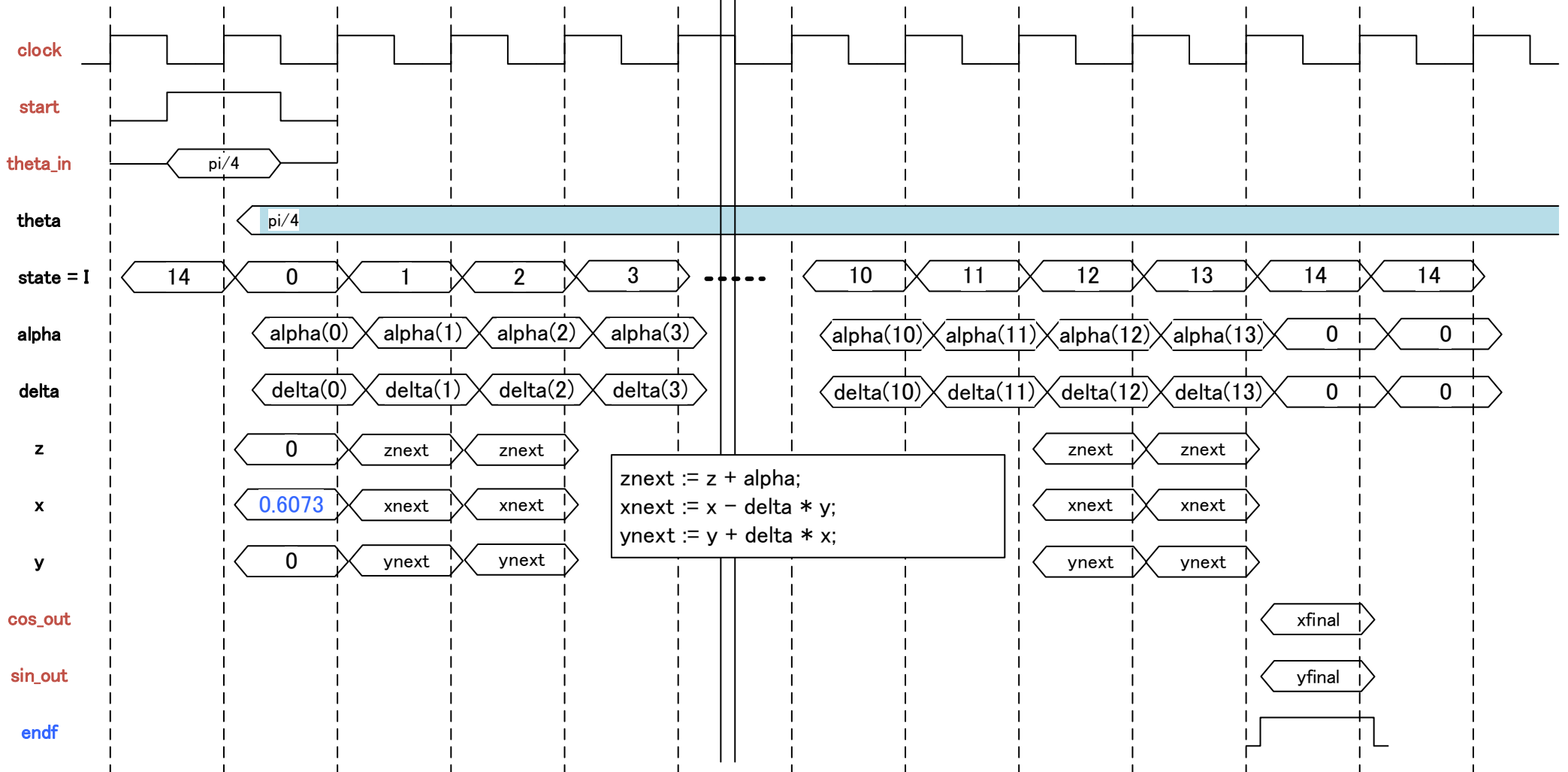
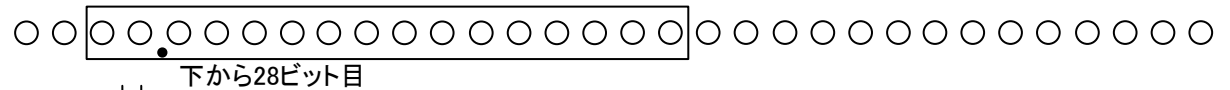
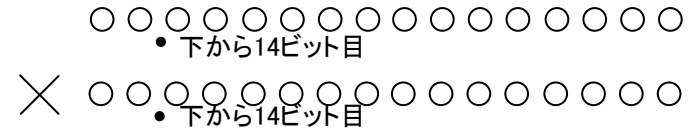
16ビット固定小数点
2の補数表現(符号付数)



↑ 回路中には小数点を示すものではなく、単なる16ビットの数であるが、この場所に小数点があると思いながら設計する

信号名	入出力	ビット幅	説明
clock	IN	1	クロック入力
reset	IN	1	'0' でリセット
start	IN	1	演算開始を示す
theta_in	IN	16	θ 入力
cos_out	OUT	16	cos出力
sin_out	OUT	16	sin出力
endf	OUT	1	出力イネーブル

16ビットどうしを乗算すると、32ビットの数となり、小数点は以下の位置となる。したがって、乗算結果の16ビットは以下の部分をとりなくてはならない。



CORDIC 回路 回路アーキテクチャ図 ver.2

