

CORDIC

-- **** STUDENT: 64000225	2
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	2
-- **** STUDENT: 64190088	5
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	5
-- **** STUDENT: 64200163	8
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	8
-- **** STUDENT: 64200296	11
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	11
-- **** STUDENT: 64200385	14
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	14
-- **** STUDENT: 64210132	18
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	18
-- **** STUDENT: 64210290	21
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	21
-- **** STUDENT: 64210382	26
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	26
-- **** STUDENT: 64210384	29
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	29
-- **** STUDENT: 64210445	32
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	32
-- **** STUDENT: 64210455	35
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	35
-- **** STUDENT: 64210457	38
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	38
-- **** PREDLOGA VAJE	41
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	41

```

-- *****
-- **** STUDENT: 64000225
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****

LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use WORK.cordic_pkg.all;

ENTITY cordic IS
  GENERIC(
    WIDTH : integer := 32
  );
  PORT(
    clk, nRST, start : IN std_logic;
    angle : IN std_logic_vector ( WIDTH - 1 DOWNT0 0 );      -- angle( radians )!!!
    sin , cos : OUT std_logic_vector ( WIDTH - 1 DOWNT0 0 );
    done : OUT std_logic
  );
END cordic;

ARCHITECTURE rtl OF cordic IS

  SIGNAL      count : std_logic_vector( sizeof( WIDTH - 1 ) - 1 DOWNT0 0 );      -- if width=32, count goes from 0
to 31
  SIGNAL      x_load, y_load, z_load,          x_reg, y_reg, z_reg,          x_result, y_result, z_result,
  cordic_coefficient,          shifted_y_reg, shifted_x_reg : std_logic_vector( WIDTH-1 DOWNT0 0 ) := (
others => '0' );
  SIGNAL      init, load, add_nsub, nadd_sub, rco, regs_load : std_logic;

  constant    x0 : std_logic_vector( WIDTH - 1 downto 0 ) := Conv2fixedPt ( inverseK, WIDTH );      --
1/1.64676025812107
  constant    y0 : std_logic_vector( WIDTH - 1 downto 0 ) := ( others => '0' );

BEGIN

  -- COS_X_REG

```

```

cos_x_reg: data_reg
    generic map ( width => width )
    port map ( clk => clk, load => regs_load, X => x_load, Q => x_reg );
-- BUS MUX X
x_load <= x0 when init = '1' else x_result;

-- SIN_Y_REG
sin_y_reg: data_reg
    generic map ( width => width )
    port map ( clk => clk, load => regs_load, X => y_load, Q => y_reg );
-- BUS MUX Y
y_load <= y0 when init = '1' else y_result;

-- ANGLE_Z_REG
angle_z_reg: data_reg
    generic map ( width => width )
    port map ( clk => clk, load => regs_load, X => z_load, Q => z_reg );
-- BUS MUX Z
z_load <= angle when init = '1' else z_result;

-- BARREL_SHIFT_Y
barrel_shift_y: barrel_shifter_sra
    generic map ( width => width )
    port map ( input => y_reg, output => shifted_y_reg, n => count );

-- BARREL_SHIFT_X
barrel_shift_x: barrel_shifter_sra
    generic map ( width => width )
    port map ( input => x_reg, output => shifted_x_reg, n => count );

add_nsub    <= z_reg( width - 1 );
nadd_sub    <= not add_nsub;
regs_load   <= init or load;

-- CORDIC_ROM
rom: cordic_rom
    generic map ( width => width, size => width )
    port map ( addr => count, dout => cordic_coefficient );

addsub_x : addsub

```

```

        generic map ( width => width )
        port map ( a => x_reg, b => shifted_y_reg, s => x_result, add_nsub => add_nsub );

addsub_y : addsub
    generic map ( width => width )
    port map ( a => y_reg, b => shifted_x_reg, s => y_result, add_nsub => nadd_sub );

addsub_z : addsub
    generic map ( width => width )
    port map ( a => z_reg, b => cordic_coefficient, s => z_result, add_nsub => add_nsub );

-- UPCOUNTER
upcounter: up_counter
    generic map ( modulo => width )
    port map ( clk => clk, nRST => nRST, count_enable => load, count => count, rco => rco );

cordicfsm: cordic_fsm
    port map (
        clk => clk, nRST => nRST, start => start, rco => rco,
        init => init, load => load, done
=> done
    );

    cos    <= x_reg;
    sin    <= y_reg;

END rtl;

```

```

-- *****
-- **** STUDENT: 64190088
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use WORK.cordic_pkg.all;

ENTITY cordic IS
  GENERIC(
    WIDTH : integer := 32
  );
  PORT(
    clk, nRST, start : IN std_logic;
    angle : IN std_logic_vector ( WIDTH - 1 DOWNT0 0 );      -- angle( radians )!!!
    sin , cos : OUT std_logic_vector ( WIDTH - 1 DOWNT0 0 );
    done : OUT std_logic
  );
END cordic;

ARCHITECTURE rtl OF cordic IS

  SIGNAL      count : std_logic_vector( sizeof( WIDTH - 1 ) - 1 DOWNT0 0 );      -- if width=32, count goes from 0
to 31
  SIGNAL      x_load, y_load, z_load,          x_reg, y_reg, z_reg,          x_result, y_result, z_result,
  cordic_coefficient,          shifted_y_reg, shifted_x_reg : std_logic_vector( WIDTH-1 DOWNT0 0 ) := (
others => '0' );
  SIGNAL      init, load, add_nsub, nadd_sub, rco, regs_load : std_logic;

  constant    x0 : std_logic_vector( WIDTH - 1 downto 0 ) := Conv2fixedPt ( inverseK, WIDTH );      --
1/1.64676025812107
  constant    y0 : std_logic_vector( WIDTH - 1 downto 0 ) := ( others => '0' );

BEGIN

  COS_X_REG: data_reg
  generic map ( width => width )

```

```

port map ( clk => clk, load => regs_load, X => x_load, Q => x_reg );

SIN_Y_REG: data_reg
generic map ( width => width )
port map ( clk => clk, load => regs_load, X => y_load, Q => y_reg );

ANGLE_Z_REG: data_reg
generic map ( width => width )
port map ( clk => clk, load => regs_load, X => z_load, Q => z_reg );

-- 3x BUS MUX:
x_load <= x0 when init = '1' else x_result;
y_load <= y0 when init = '1' else y_result;
z_load <= angle when init = '1' else z_result;

BARREL_SHIFT_X: barrel_shifter_sra
generic map ( width => width )
port map ( input => x_reg, output => shifted_x_reg, n => count );

BARREL_SHIFT_Y: barrel_shifter_sra
generic map ( width => width )
port map ( input => y_reg, output => shifted_y_reg, n => count );

ROM: cordic_rom
generic map ( width => width, size => width )
port map ( addr => count, dout => cordic_coefficient );

X_ADDSUB : addsub
generic map ( width => width )
port map ( a => x_reg, b => shifted_y_reg, s => x_result, add_nsub => add_nsub );

Y_ADDSUB : addsub
generic map ( width => width )
port map ( a => y_reg, b => shifted_x_reg, s => y_result, add_nsub => nadd_sub );

Z_ADDSUB : addsub
generic map ( width => width )
port map ( a => z_reg, b => cordic_coefficient, s => z_result, add_nsub => add_nsub );

UPCOUNTER: up_counter

```

```
generic map ( modulo => width )
```

```
port map ( clk => clk, nRST => nRST, count_enable => load, count => count, rco => rco );
```

```
CORDICFSM: cordic_fsm
```

```
port map ( clk => clk, nRST => nRST, start => start, rco => rco, init => init, load => load, done => done );
```

```
add_nsub      <= z_reg( width - 1 );
```

```
nadd_sub      <= not add_nsub;
```

```
regs_load     <= init or load;
```

```
cos           <= x_reg;
```

```
sin           <= y_reg;
```

```
END rtl;
```

```

-- *****
-- **** STUDENT: 64200163
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use WORK.cordic_pkg.all;

entity cordic is
  generic(
    WIDTH : integer := 32
  );
  port(
    clk, nRST, start : in std_logic;
    angle : in std_logic_vector( WIDTH - 1 downto 0 );
    sin, cos : out std_logic_vector( WIDTH - 1 downto 0 );
    done : out std_logic
  );
end cordic;

architecture rtl of cordic is
  signal      count : std_logic_vector( sizeof( WIDTH - 1 ) - 1 downto 0 );
  signal      x_load, y_load, z_load,          x_reg, y_reg, z_reg,          x_result, y_result, z_result,
  cordic_coefficient,          shifted_y_reg, shifted_x_reg : std_logic_vector( WIDTH - 1 downto 0 ) := (
others => '0' );
  signal      init, load, add_nsub, nadd_sub, rco, regs_load : std_logic;
  constant    x0 : std_logic_vector( WIDTH - 1 downto 0 ) := Conv2fixedPt( inverseK, WIDTH );
  constant    y0 : std_logic_vector( WIDTH - 1 downto 0 ) := ( others => '0' );
begin
  x_load <= x0 when init = '1' else x_result;
  cos_x_reg : data_reg
    generic map( width => width )
    port map( clk => clk,          load => regs_load,          X => x_load,          Q => x_reg
    );
  barrel_shift_y : barrel_shifter_sra
    generic map( width => width )
    port map( input => y_reg,          output => shifted_y_reg,          n => count

```

```

    );
    addsub_x : addsub
        generic map( width => width )
        port map( a => x_reg,          b => shifted_y_reg,          s => x_result,          add_nsub =>
add_nsub
        );

    y_load <= y0 when init = '1' else y_result;
    sin_y_reg : data_reg
        generic map( width => width )
        port map( clk => clk,          load => regs_load,          X => y_load,          Q => y_reg
        );
    barrel_shift_x : barrel_shifter_sra
        generic map( width => width )
        port map( input => x_reg,          output => shifted_x_reg,          n => count
        );
    addsub_y : addsub
        generic map( width => width )
        port map( a => y_reg,          b => shifted_x_reg,          s => y_result,          add_nsub =>
nadd_sub
        );

    z_load <= angle when init = '1' else z_result;
    angle_z_reg : data_reg
        generic map( width => width )
        port map( clk => clk,          load => regs_load,          X => z_load,          Q => z_reg
        );
    addsub_z : addsub
        generic map( width => width )
        port map( a => z_reg,          b => cordic_coefficient,          s => z_result,          add_nsub =>
add_nsub
        );
    rom : cordic_rom
        generic map( width => width,          size => width )
        port map( addr => count,          dout => cordic_coefficient
        );

    add_nsub <= z_reg( width - 1 );
    nadd_sub <= not add_nsub;
    regs_load <= init or load;

```

```

    upcounter : up_counter
        generic map( modulo => width )
        port map( clk => clk,          nRST => nRST,          count_enable => load,          count =>
count,          rco => rco
        );

    cordicfsm : cordic_fsm
        port map( clk => clk,          nRST => nRST,          start => start,          rco => rco,
init => init,          load => load,          done => done
        );

    cos    <= x_reg;
    sin    <= y_reg;
end rtl;

```

```

-- *****
-- **** STUDENT: 64200296
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use WORK.cordic_pkg.all;

ENTITY cordic IS
  GENERIC(
    WIDTH : integer := 32
  );
  PORT(
    clk, nRST, start : IN std_logic;
    angle : IN std_logic_vector ( WIDTH - 1 DOWNT0 0 );      -- angle( radians )!!!
    sin , cos : OUT std_logic_vector ( WIDTH - 1 DOWNT0 0 );
    done : OUT std_logic
  );
END cordic;

ARCHITECTURE rtl OF cordic IS

  SIGNAL      count : std_logic_vector( sizeof( WIDTH - 1 ) - 1 DOWNT0 0 );      -- if width=32, count goes from 0
to 31
  SIGNAL      x_load, y_load, z_load,          x_reg, y_reg, z_reg,          x_result, y_result, z_result,
  cordic_coefficient,      shifted_y_reg, shifted_x_reg : std_logic_vector( WIDTH-1 DOWNT0 0 ) := (
others => '0' );
  SIGNAL      init, load, add_nsub, nadd_sub, rco, regs_load : std_logic;

  constant    x0 : std_logic_vector( WIDTH - 1 downto 0 ) := Conv2fixedPt ( inverseK, WIDTH );      --
1/1.64676025812107
  constant    y0 : std_logic_vector( WIDTH - 1 downto 0 ) := ( others => '0' );

BEGIN
  counter: up_counter
    generic map ( modulo => width )
    port map ( clk => clk, nRST => nRST, count_enable => load, count => count, rco => rco );

```

```

rom: cordic_rom
    generic map ( width => width, size => width )
    port map ( addr => count, dout => cordic_coefficient );

cos_x: data_reg
    generic map ( width => width )
    port map ( clk => clk, load => regs_load, X => x_load, Q => x_reg );

sin_y: data_reg
    generic map ( width => width )
    port map ( clk => clk, load => regs_load, X => y_load, Q => y_reg );

angle_z: data_reg
    generic map ( width => width )
    port map ( clk => clk, load => regs_load, X => z_load, Q => z_reg );

shift_y: barrel_shifter_sra
    generic map ( width => width )
    port map ( input => y_reg, output => shifted_y_reg, n => count );

shift_x: barrel_shifter_sra
    generic map ( width => width )
    port map ( input => x_reg, output => shifted_x_reg, n => count );

sub_x : addsub
    generic map ( width => width )
    port map ( a => x_reg, b => shifted_y_reg, s => x_result, add_nsub => add_nsub );

sub_y : addsub
    generic map ( width => width )
    port map ( a => y_reg, b => shifted_x_reg, s => y_result, add_nsub => nadd_sub );

sub_z : addsub
    generic map ( width => width )
    port map ( a => z_reg, b => cordic_coefficient, s => z_result, add_nsub => add_nsub );

cordicfsm: cordic_fsm
    port map (

```

```

        clk => clk, nRST => nRST, start => start, rco => rco,
        init => init, load => load, done
=> done
    );

    x_load <= x0 when init = '1' else x_result;
    y_load <= y0 when init = '1' else y_result;
    z_load <= angle when init = '1' else z_result;
    add_nsub    <= z_reg( width - 1 );
    nadd_sub    <= not add_nsub;
    regs_load   <= init or load;
    cos    <= x_reg;
    sin    <= y_reg;
END rtl;

```

```

-- *****
-- **** STUDENT: 64200385
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use WORK.cordic_pkg.all;

ENTITY cordic IS
  GENERIC(
    WIDTH : integer := 32
  );
  PORT(
    clk, nRST, start : IN std_logic;
    angle : IN std_logic_vector ( WIDTH - 1 DOWNT0 0 );    -- angle( radians )!!!
    sin , cos : OUT std_logic_vector ( WIDTH - 1 DOWNT0 0 );
    done : OUT std_logic
  );
END cordic;

ARCHITECTURE rtl OF cordic IS

  SIGNAL      count : std_logic_vector( sizeof( WIDTH - 1 ) - 1 DOWNT0 0 );
  SIGNAL      x_load, y_load, z_load,          x_reg, y_reg, z_reg,          x_result, y_result, z_result,
  cordic_coefficient,          shifted_y_reg, shifted_x_reg : std_logic_vector( WIDTH-1 DOWNT0 0 ) := (
others => '0' );
  SIGNAL      init, load, add_nsub, nadd_sub, rco, regs_load : std_logic;

  constant    x0 : std_logic_vector( WIDTH - 1 downto 0 ) := Conv2fixedPt ( inverseK, WIDTH );
  constant    y0 : std_logic_vector( WIDTH - 1 downto 0 ) := ( others => '0' );

BEGIN

  x_load      <= x0 when init = '1' else x_result;

  y_load      <= y0 when init = '1' else y_result;

```

```
z_load      <= angle when init = '1' else z_result;
```

```
U0: data_reg  
generic map( width => width )  
port map ( clk => clk,  
load => regs_load,  
X => x_load,  
Q => x_reg );
```

```
U1: data_reg  
generic map( width => width )  
port map ( clk => clk,  
load => regs_load,  
X => y_load,  
Q => y_reg );
```

```
U2: data_reg  
generic map( width => width )  
port map ( clk => clk,  
load => regs_load,  
X => z_load,  
Q => z_reg );
```

```
U3: barrel_shifter_sra  
generic map( width => width )  
port map ( input => y_reg,  
output => shifted_y_reg,  
n => count );
```

```
U4: barrel_shifter_sra  
generic map( width => width )  
port map ( input => x_reg,  
output => shifted_x_reg,  
n => count );
```

```
U5: cordic_rom  
generic map ( width => width )  
port map ( addr => count,  
dout => cordic_coefficient );
```

```

U6: addsub
generic map ( width => width )
port map ( A => x_reg,
B => shifted_y_reg,
S => x_result,
add_nsub => add_nsub );

U7: addsub
generic map ( width => width )
port map ( A => y_reg,
B => shifted_x_reg,
S => y_result,
add_nsub => nadd_sub );

U8: addsub
generic map ( width => width )
port map ( A => z_reg,
B => cordic_coefficient,
S => z_result,
add_nsub => add_nsub );

add_nsub    <= z_reg( width - 1 );
nadd_sub    <= not add_nsub;
regs_load   <= init or load;

U9: up_counter
generic map ( modulo => width )
port map ( clk => clk,
nRST => nRST,
count_enable => load,
count => count,
rco => rco );

U10: cordic_fsm
port map ( clk => clk,
nRST => nRST,
start => start,
rco => rco,
init => init,
load => load,

```

```
done => done );
```

```
cos  <= x_reg;
```

```
sin  <= y_reg;
```

```
END rtl;
```

```

-- *****
-- **** STUDENT: 64210132
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use WORK.cordic_pkg.all;

ENTITY cordic IS
  GENERIC(
    WIDTH : integer := 32
  );
  PORT(
    clk, nRST, start : IN std_logic;
    angle : IN std_logic_vector ( WIDTH - 1 DOWNT0 0 );      -- angle( radians )!!!
    sin , cos : OUT std_logic_vector ( WIDTH - 1 DOWNT0 0 );
    done : OUT std_logic
  );
END cordic;

ARCHITECTURE rtl OF cordic IS

  SIGNAL      count : std_logic_vector( sizeof( WIDTH - 1 ) - 1 DOWNT0 0 );      -- if width=32, count goes from 0
to 31
  SIGNAL      x_load, y_load, z_load,          x_reg, y_reg, z_reg,          x_result, y_result, z_result,
cordic_coefficient,          shifted_y_reg, shifted_x_reg : std_logic_vector( WIDTH-1 DOWNT0 0 ) := (
others => '0' );
  SIGNAL      init, load, add_nsub, nadd_sub, rco, regs_load : std_logic;

  constant    x0 : std_logic_vector( WIDTH - 1 downto 0 ) := Conv2fixedPt ( inverseK, WIDTH );      --
1/1.64676025812107
  constant    y0 : std_logic_vector( WIDTH - 1 downto 0 ) := ( others => '0' );

BEGIN

  x_load <= x_result when init = '0' else x0;

```

```

y_load <= y_result when init = '0' else y0;

z_load <= z_result when init = '0' else angle;

COS_X_REG: data_reg
    generic map ( width => width )
    port map ( clk, regs_load, x_load, x_reg );

SIN_Y_REG: data_reg
    generic map ( width => width )
    port map ( clk, regs_load, y_load, y_reg );

ANGLE_Z_REG: data_reg
    generic map ( width => width )
    port map ( clk, regs_load, z_load, z_reg );

BARREL_SHIFT_Y: barrel_shifter_sra
    generic map ( width => width )
    port map ( y_reg, shifted_y_reg, count );

BARREL_SHIFT_X: barrel_shifter_sra
    generic map ( width => width )
    port map ( x_reg, shifted_x_reg, count );

CORDICROM: cordic_rom
    generic map ( width => width, size => width )
    port map ( count, cordic_coefficient );

ADDSUB_1 : addsub
    generic map ( width => width )
    port map ( x_reg, shifted_y_reg, x_result, add_nsub );

ADDSUB_2 : addsub
    generic map ( width => width )
    port map ( y_reg, shifted_x_reg, y_result, nadd_sub );

ADDSUB_3 : addsub
    generic map ( width => width )
    port map ( z_reg, cordic_coefficient, z_result, add_nsub );

```

```
UPCOUNTER: up_counter
    generic map ( modulo => width )
    port map ( clk, nRST, load, count, rco );

CORDICFSM: cordic_fsm
    port map ( clk, nRST, start, rco, init, load, done );
```

```
add_nsub    <= z_reg( width - 1 );
nadd_sub    <= not add_nsub;
regs_load   <= init or load;
```

```
cos    <= x_reg;
sin    <= y_reg;
```

```
END rtl;
```

```

-- *****
-- **** STUDENT: 64210290
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;
USE WORK.cordic_pkg.all;

ENTITY cordic IS
  GENERIC(
    WIDTH : INTEGER := 32
  );
  PORT(
    clk, nRST, start : IN  STD_LOGIC;
    angle            : IN  STD_LOGIC_VECTOR ( WIDTH - 1 DOWNTO 0 );    -- angle( radians )!!!
    sin, cos         : OUT STD_LOGIC_VECTOR ( WIDTH - 1 DOWNTO 0 );
    done             : OUT STD_LOGIC
  );
END cordic;

ARCHITECTURE rtl OF cordic IS

  component CORDIC_FSM is
    port(
      clk, nRST, start, rco      : in  std_logic;
      init, load, done           : out std_logic
    );
  end component;

  component data_reg is
    generic(
      WIDTH : integer
    );
    port(
      clk, load : in std_logic;
      X         : in  std_logic_vector ( WIDTH-1 downto 0 ); -- Load value ( Loads X when Load = '1' )
      Q         : out std_logic_vector ( WIDTH-1 downto 0 )  -- output value
    );
  end component;

```

```

    );
end component;

component barrel_shifter_sra is
    generic(
        width : integer;
        size  : integer
    );
    port(
        input : in      std_logic_vector ( WIDTH-1 downto 0 );
        output : out std_logic_vector ( WIDTH-1 downto 0 );
        n : in      std_logic_vector ( sizeof( size - 1 ) - 1 downto 0 )
    );
end component;

component CORDIC_ROM is
    generic(
        WIDTH : integer;
        SIZE  : integer
    );
    port(
        addr  : in  std_logic_vector ( sizeof( SIZE - 1 ) - 1 downto 0 );
        dout  : out std_logic_vector ( WIDTH - 1 downto 0 )
    );
end component;

component addsub is
    generic(
        WIDTH : integer
    );
    port(
        A , B      : in  std_logic_vector ( WIDTH-1 downto 0 );
        S          : out std_logic_vector ( WIDTH-1 downto 0 );
        add_nsub   : in  std_logic      -- add_nsub = '0'->subtraction, '1'->addition
    );
end component;

component up_counter is
    generic(
        MODULO : integer

```

```

    );
    port( clk, nRST, count_enable    : in std_logic;
          count : out std_logic_vector( sizeof( MODULO - 1 ) - 1 downto 0 );
          rco    : out std_logic
    );
end component;

signal      count : std_logic_vector( sizeof( WIDTH - 1 ) - 1 downto 0 );    -- if width=32, count goes from 0
to 31
signal      x_load, y_load, z_load,          x_reg, y_reg, z_reg,          x_result, y_result, z_result,
cordic_coefficient,          shifted_y_reg, shifted_x_reg : std_logic_vector( WIDTH-1 downto 0 ) := (
others => '0' );
signal      init, load, add_nsub, nadd_sub, rco, regs_load : std_logic;

constant    x0 : std_logic_vector( WIDTH - 1 downto 0 ) := Conv2fixedPt ( inverseK, WIDTH );    --
1/1.64676025812107
constant    y0 : std_logic_vector( WIDTH - 1 downto 0 ) := ( others => '0' );

BEGIN

    -- FSM
    fsm : CORDIC_FSM
    port map(
=> init,          clk => clk,          nRST => nRST,          start => start,          rco => rco,          init
          load => load,          done => done
    );

    -- MULTIPLEXERS
    x_load <= x0 when init='1' else x_result;
    y_load <= y0 when init='1' else y_result;
    z_load <= angle when init='1' else z_result;

    -- REGISTERS
    cos_x_reg : data_reg
    generic map(
        WIDTH => WIDTH
    )
    port map(
        clk => clk,          load => REGS_LOAD,          X => x_load,          Q => x_reg
    );

```

```

sin_y_reg : data_reg
    generic map(
        WIDTH => WIDTH
    )
    port map(
        clk => clk,          load => REGS_LOAD,          X => y_load,          Q => y_reg
    );
angle_z_reg : data_reg
    generic map(
        WIDTH => WIDTH
    )
    port map(
        clk => clk,          load => REGS_LOAD,          X => z_load,          Q => z_reg
    );
cos    <= x_reg;
sin    <= y_reg;

REGS_LOAD    <= init or load;
add_nsub     <= z_reg( z_reg'high );
nadd_sub     <= not add_nsub;

-- BARREL SHIFTERS
barrel_shift_y : barrel_shifter_sra
    generic map(
        width => width,          size => width
    )
    port map(
        input => y_reg,          output => shifted_y_reg,          n => count
    );
barrel_shift_x : barrel_shifter_sra
    generic map(
        width => width,          size => width
    )
    port map(
        input => x_reg,          output => shifted_x_reg,          n => count
    );

-- ROM
rom : CORDIC_ROM
    generic map(

```

```

        WIDTH => width,          SIZE => width
    )
    port map(
        addr => count,          dout => cordic_coefficient
    );

-- ADDSUB
cos_x_addsub : addsub
    generic map(
        WIDTH => width
    )
    port map(
        A => x_reg,          B => shifted_y_reg,          S => x_result,          add_nsub => add_nsub
    );
sin_y_addsub : addsub
    generic map(
        WIDTH => width
    )
    port map(
        A => y_reg,          B => shifted_x_reg,          S => y_result,          add_nsub => nadd_sub
    );
angle_z_addsub : addsub
    generic map(
        WIDTH => width
    )
    port map(
        A => z_reg,          B => cordic_coefficient,          S => z_result,          add_nsub => add_nsub
    );

-- UPCOUNTER
upcounter : up_counter
    generic map(
        MODULO => width
    )
    port map(
        clk => clk,          nRST => nRST,          count_enable => load,          count => count,
rco => rco
    );

END rtl;

```

```

-- *****
-- **** STUDENT: 64210382
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use WORK.cordic_pkg.all;

ENTITY cordic IS
  GENERIC(
    WIDTH : integer := 32
  );
  PORT(
    clk, nRST, start : IN std_logic;
    angle : IN std_logic_vector ( WIDTH - 1 DOWNT0 0 );      -- angle( radians )!!!
    sin , cos : OUT std_logic_vector ( WIDTH - 1 DOWNT0 0 );
    done : OUT std_logic
  );
END cordic;

ARCHITECTURE rtl OF cordic IS

  SIGNAL      count : std_logic_vector( sizeof( WIDTH - 1 ) - 1 DOWNT0 0 );      -- if width=32, count goes from 0
to 31
  SIGNAL      x_load, y_load, z_load,          x_reg, y_reg, z_reg,          x_result, y_result, z_result,
  cordic_coefficient,          shifted_y_reg, shifted_x_reg : std_logic_vector( WIDTH-1 DOWNT0 0 ) := (
others => '0' );
  SIGNAL      init, load, add_nsub, nadd_sub, rco, regs_load : std_logic;

  constant    x0 : std_logic_vector( WIDTH - 1 downto 0 ) := Conv2fixedPt ( inverseK, WIDTH );      --
1/1.64676025812107
  constant    y0 : std_logic_vector( WIDTH - 1 downto 0 ) := ( others => '0' );

BEGIN
cos_x_reg: data_reg
  generic map ( width => width )
  port map ( clk => clk, load => regs_load, X => x_load, Q => x_reg );

```

```

x_load <= x0 when init = '1' else x_result;

sin_y_reg: data_reg
    generic map ( width => width )
    port map ( clk => clk, load => regs_load, X => y_load, Q => y_reg );

y_load <= y0 when init = '1' else y_result;

angle_z_reg: data_reg
    generic map ( width => width )
    port map ( clk => clk, load => regs_load, X => z_load, Q => z_reg );

z_load <= angle when init = '1' else z_result;

barrel_shift_y: barrel_shifter_sra
    generic map ( width => width )
    port map ( input => y_reg, output => shifted_y_reg, n => count );

barrel_shift_x: barrel_shifter_sra
    generic map ( width => width )
    port map ( input => x_reg, output => shifted_x_reg, n => count );

add_nsub    <= z_reg( width - 1 );
nadd_sub    <= not add_nsub;
regs_load   <= init or load;

rom: cordic_rom
    generic map ( width => width, size => width )
    port map ( addr => count, dout => cordic_coefficient );

addsub_x : addsub
    generic map ( width => width )
    port map ( a => x_reg, b => shifted_y_reg, s => x_result, add_nsub => add_nsub );

addsub_y : addsub
    generic map ( width => width )
    port map ( a => y_reg, b => shifted_x_reg, s => y_result, add_nsub => nadd_sub );

addsub_z : addsub

```

```

        generic map ( width => width )
        port map ( a => z_reg, b => cordic_coefficient, s => z_result, add_nsub => add_nsub );

upcounter: up_counter
    generic map ( modulo => width )
    port map ( clk => clk, nRST => nRST, count_enable => load, count => count, rco => rco );

cordicfsm: cordic_fsm
    port map (
        clk => clk, nRST => nRST, start => start, rco => rco,
        init => init, load => load, done
=> done
    );

    cos    <= x_reg;
    sin    <= y_reg;
END rtl;

```

```

-- *****
-- **** STUDENT: 64210384
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use WORK.cordic_pkg.all;

ENTITY cordic IS
  GENERIC(
    WIDTH : integer := 32
  );
  PORT(
    clk, nRST, start : IN std_logic;
    angle : IN std_logic_vector ( WIDTH - 1 DOWNT0 0 );      -- angle( radians )
    sin , cos : OUT std_logic_vector ( WIDTH - 1 DOWNT0 0 );
    done : OUT std_logic
  );
END cordic;

ARCHITECTURE rtl OF cordic IS

  SIGNAL      count : std_logic_vector( sizeof( WIDTH - 1 ) - 1 DOWNT0 0 );      -- if width=32, count goes from 0
to 31
  SIGNAL      x_load, y_load, z_load,          x_reg, y_reg, z_reg,          x_result, y_result, z_result,
  cordic_coefficient,          shifted_y_reg, shifted_x_reg : std_logic_vector( WIDTH-1 DOWNT0 0 ) := (
others => '0' );
  SIGNAL      init, load, add_nsub, nadd_sub, rco, regs_load : std_logic;

  constant    x0 : std_logic_vector( WIDTH - 1 downto 0 ) := Conv2fixedPt ( inverseK, WIDTH );      --
1/1.64676025812107
  constant    y0 : std_logic_vector( WIDTH - 1 downto 0 ) := ( others => '0' );

BEGIN
  -- COS_X_REG
  cos_x_reg: data_reg
    generic map ( width => width )

```

```

        port map ( clk => clk, load => regs_load, X => x_load, Q => x_reg );
-- BUS MUX X
x_load <= x0 when init = '1' else x_result;

-- SIN_Y_REG
sin_y_reg: data_reg
    generic map ( width => width )
    port map ( clk => clk, load => regs_load, X => y_load, Q => y_reg );
-- BUS MUX Y
y_load <= y0 when init = '1' else y_result;

-- ANGLE_Z_REG
angle_z_reg: data_reg
    generic map ( width => width )
    port map ( clk => clk, load => regs_load, X => z_load, Q => z_reg );
-- BUS MUX Z
z_load <= angle when init = '1' else z_result;

-- BARREL_SHIFT_Y
barrel_shift_y: barrel_shifter_sra
    generic map ( width => width )
    port map ( input => y_reg, output => shifted_y_reg, n => count );

-- BARREL_SHIFT_X
barrel_shift_x: barrel_shifter_sra
    generic map ( width => width )
    port map ( input => x_reg, output => shifted_x_reg, n => count );

add_nsub    <= z_reg( width - 1 );
nadd_sub    <= not add_nsub;
regs_load   <= init or load;

-- CORDIC_ROM
rom: cordic_rom
    generic map ( width => width, size => width )
    port map ( addr => count, dout => cordic_coefficient );

addsub_x : addsub
    generic map ( width => width )
    port map ( a => x_reg, b => shifted_y_reg, s => x_result, add_nsub => add_nsub );

```

```

addsub_y : addsub
    generic map ( width => width )
    port map ( a => y_reg, b => shifted_x_reg, s => y_result, add_nsub => nadd_sub );

addsub_z : addsub
    generic map ( width => width )
    port map ( a => z_reg, b => cordic_coefficient, s => z_result, add_nsub => add_nsub );

-- UPCOUNTER
upcounter: up_counter
    generic map ( modulo => width )
    port map ( clk => clk, nRST => nRST, count_enable => load, count => count, rco => rco );

cordicfsm: cordic_fsm
    port map (
        clk => clk, nRST => nRST, start => start, rco => rco,
        init => init, load => load, done
=> done
    );
    cos    <= x_reg;
    sin    <= y_reg;
END rtl;

```

```

-- *****
-- **** STUDENT: 64210445
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use WORK.cordic_pkg.all;

ENTITY cordic IS
  GENERIC(
    WIDTH : integer := 32
  );
  PORT(
    clk, nRST, start : IN std_logic;
    angle : IN std_logic_vector ( WIDTH - 1 DOWNT0 0 );    -- angle( radians )!!!
    sin , cos : OUT std_logic_vector ( WIDTH - 1 DOWNT0 0 );
    done : OUT std_logic
  );
END cordic;

ARCHITECTURE rtl OF cordic IS

  SIGNAL      count : std_logic_vector( sizeof( WIDTH - 1 ) - 1 DOWNT0 0 );    -- if width=32, count goes from 0
to 31
  SIGNAL      x_load, y_load, z_load,          x_reg, y_reg, z_reg,          x_result, y_result, z_result,
  cordic_coefficient,          shifted_y_reg, shifted_x_reg : std_logic_vector( WIDTH-1 DOWNT0 0 ) := (
others => '0' );
  SIGNAL      init, load, add_nsub, nadd_sub, rco, regs_load : std_logic;

  constant    x0 : std_logic_vector( WIDTH - 1 downto 0 ) := Conv2fixedPt ( inverseK, WIDTH );    --
1/1.64676025812107
  constant    y0 : std_logic_vector( WIDTH - 1 downto 0 ) := ( others => '0' );

BEGIN

  cos_x_reg: data_reg
    generic map ( width => width )

```

```

        port map ( clk => clk, load => regs_load, X => x_load, Q => x_reg );
x_load <= x0 when init = '1' else x_result;

sin_y_reg: data_reg
    generic map ( width => width )
    port map ( clk => clk, load => regs_load, X => y_load, Q => y_reg );
y_load <= y0 when init = '1' else y_result;

angle_z_reg: data_reg
    generic map ( width => width )
    port map ( clk => clk, load => regs_load, X => z_load, Q => z_reg );
z_load <= angle when init = '1' else z_result;

barrel_shift_y: barrel_shifter_sra
    generic map ( width => width )
    port map ( input => y_reg, output => shifted_y_reg, n => count );

barrel_shift_x: barrel_shifter_sra
    generic map ( width => width )
    port map ( input => x_reg, output => shifted_x_reg, n => count );

add_nsub    <= z_reg( width - 1 );
nadd_sub    <= not add_nsub;
regs_load   <= init or load;

rom: cordic_rom
    generic map ( width => width, size => width )
    port map ( addr => count, dout => cordic_coefficient );

addsub_x : addsub
    generic map ( width => width )
    port map ( a => x_reg, b => shifted_y_reg, s => x_result, add_nsub => add_nsub );

addsub_y : addsub
    generic map ( width => width )
    port map ( a => y_reg, b => shifted_x_reg, s => y_result, add_nsub => nadd_sub );

addsub_z : addsub
    generic map ( width => width )
    port map ( a => z_reg, b => cordic_coefficient, s => z_result, add_nsub => add_nsub );

```

```

upcounter: up_counter
    generic map ( modulo => width )
    port map ( clk => clk, nRST => nRST, count_enable => load, count => count, rco => rco );

cordicfsm: cordic_fsm
    port map ( clk => clk, nRST => nRST, start => start, rco => rco,          init => init, load => load, done
=> done );

    cos    <= x_reg;
    sin    <= y_reg;

END rtl;

```

```

-- *****
-- **** STUDENT: 64210455
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use WORK.cordic_pkg.all;

ENTITY cordic IS
  GENERIC(
    WIDTH : integer := 32
  );
  PORT(
    clk, nRST, start : IN std_logic;
    angle : IN std_logic_vector ( WIDTH - 1 DOWNT0 0 );    -- angle( radians )!!!
    sin , cos : OUT std_logic_vector ( WIDTH - 1 DOWNT0 0 );
    done : OUT std_logic
  );
END cordic;

ARCHITECTURE rtl OF cordic IS

  SIGNAL      count : std_logic_vector( sizeof( WIDTH - 1 ) - 1 DOWNT0 0 );    -- if width=32, counts from 0 to
31
  SIGNAL      x_load, y_load, z_load,          x_reg, y_reg, z_reg,          x_result, y_result, z_result,
  cordic_coefficient,          shifted_y_reg, shifted_x_reg : std_logic_vector( WIDTH-1 DOWNT0 0 ) := (
others => '0' );
  SIGNAL      init, load, add_nsub, nadd_sub, rco, regs_load : std_logic;

  constant    x0 : std_logic_vector( WIDTH - 1 downto 0 ) := Conv2fixedPt ( inverseK, WIDTH );
  constant    y0 : std_logic_vector( WIDTH - 1 downto 0 ) := ( others => '0' );

BEGIN
x_load <= x0 when init = '1' else x_result;
y_load <= y0 when init = '1' else y_result;
z_load <= angle when init = '1' else z_result;

```

```

cos_x_reg: data_reg
    generic map ( width => width )
    port map ( clk => clk, load => regs_load, X => x_load, Q => x_reg );

sin_y_reg: data_reg
    generic map ( width => width )
    port map ( clk => clk, load => regs_load, X => y_load, Q => y_reg );

angle_z_reg: data_reg
    generic map ( width => width )
    port map ( clk => clk, load => regs_load, X => z_load, Q => z_reg );

barrel_shift_y: barrel_shifter_sra
    generic map ( width => width )
    port map ( input => y_reg, output => shifted_y_reg, n => count );

barrel_shift_x: barrel_shifter_sra
    generic map ( width => width )
    port map ( input => x_reg, output => shifted_x_reg, n => count );

add_nsub    <= z_reg( width - 1 );
nadd_sub    <= not add_nsub;
regs_load   <= init or load;

rom: cordic_rom
    generic map ( width => width, size => width )
    port map ( addr => count, dout => cordic_coefficient );

addsub_x : addsub
    generic map ( width => width )
    port map ( a => x_reg, b => shifted_y_reg, s => x_result, add_nsub => add_nsub );

addsub_y : addsub
    generic map ( width => width )
    port map ( a => y_reg, b => shifted_x_reg, s => y_result, add_nsub => nadd_sub );

addsub_z : addsub
    generic map ( width => width )
    port map ( a => z_reg, b => cordic_coefficient, s => z_result, add_nsub => add_nsub );

```

```

upcounter: up_counter
    generic map ( modulo => width )
    port map ( clk => clk, nRST => nRST, count_enable => load, count => count, rco => rco );

cordicfsm: cordic_fsm
    port map (
        clk => clk, nRST => nRST, start => start, rco => rco,
        init => init, load => load, done
=> done
    );
    cos    <= x_reg;
    sin    <= y_reg;
END rtl;

```

```

-- *****
-- **** STUDENT: 64210457
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use WORK.cordic_pkg.all;

ENTITY cordic IS
  GENERIC(
    WIDTH : integer := 32
  );
  PORT(
    clk, nRST, start : IN std_logic;
    angle : IN std_logic_vector ( WIDTH - 1 DOWNT0 0 );      -- angle( radians )!!!
    sin , cos : OUT std_logic_vector ( WIDTH - 1 DOWNT0 0 );
    done : OUT std_logic
  );
END cordic;

ARCHITECTURE rtl OF cordic IS

  SIGNAL      count : std_logic_vector( sizeof( WIDTH - 1 ) - 1 DOWNT0 0 );      -- if width=32, count goes from 0
to 31
  SIGNAL      x_load, y_load, z_load,          x_reg, y_reg, z_reg,          x_result, y_result, z_result,
  cordic_coefficient,          shifted_y_reg, shifted_x_reg : std_logic_vector( WIDTH-1 DOWNT0 0 ) := (
others => '0' );
  SIGNAL      init, load, add_nsub, nadd_sub, rco, regs_load : std_logic;

  constant    x0 : std_logic_vector( WIDTH - 1 downto 0 ) := Conv2fixedPt ( inverseK, WIDTH );      --
1/1.64676025812107
  constant    y0 : std_logic_vector( WIDTH - 1 downto 0 ) := ( others => '0' );

BEGIN
  -- COS_X_REG
  cos_x_reg: data_reg
    generic map ( width => width )

```

```

        port map ( clk => clk, load => regs_load, X => x_load, Q => x_reg );
-- BUS MUX X
x_load <= x0 when init = '1' else x_result;

-- SIN_Y_REG
sin_y_reg: data_reg
    generic map ( width => width )
    port map ( clk => clk, load => regs_load, X => y_load, Q => y_reg );
-- BUS MUX Y
y_load <= y0 when init = '1' else y_result;

-- ANGLE_Z_REG
angle_z_reg: data_reg
    generic map ( width => width )
    port map ( clk => clk, load => regs_load, X => z_load, Q => z_reg );
-- BUS MUX Z
z_load <= angle when init = '1' else z_result;

-- BARREL_SHIFT_Y
barrel_shift_y: barrel_shifter_sra
    generic map ( width => width )
    port map ( input => y_reg, output => shifted_y_reg, n => count );

-- BARREL_SHIFT_X
barrel_shift_x: barrel_shifter_sra
    generic map ( width => width )
    port map ( input => x_reg, output => shifted_x_reg, n => count );

add_nsub    <= z_reg( width - 1 );
nadd_sub    <= not add_nsub;
regs_load   <= init or load;

-- CORDIC_ROM
rom: cordic_rom
    generic map ( width => width, size => width )
    port map ( addr => count, dout => cordic_coefficient );

addsub_x : addsub
    generic map ( width => width )
    port map ( a => x_reg, b => shifted_y_reg, s => x_result, add_nsub => add_nsub );

```

```

addsub_y : addsub
    generic map ( width => width )
    port map ( a => y_reg, b => shifted_x_reg, s => y_result, add_nsub => nadd_sub );

addsub_z : addsub
    generic map ( width => width )
    port map ( a => z_reg, b => cordic_coefficient, s => z_result, add_nsub => add_nsub );

-- UPCOUNTER
upcounter: up_counter
    generic map ( modulo => width )
    port map ( clk => clk, nRST => nRST, count_enable => load, count => count, rco => rco );

cordicfsm: cordic_fsm
    port map (
        clk => clk, nRST => nRST, start => start, rco => rco,
        init => init, load => load, done
=> done
    );

    cos    <= x_reg;
    sin    <= y_reg;
END rtl;

```

```

-- *****
-- **** PREDLOGA VAJE
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use WORK.cordic_pkg.all;

ENTITY cordic IS
  GENERIC(
    WIDTH : integer := 32
  );
  PORT(
    clk, nRST, start : IN std_logic;
    angle : IN std_logic_vector ( WIDTH - 1 DOWNT0 0 );      -- angle( radians )!!!
    sin , cos : OUT std_logic_vector ( WIDTH - 1 DOWNT0 0 );
    done : OUT std_logic
  );
END cordic;

ARCHITECTURE rtl OF cordic IS

  SIGNAL      count : std_logic_vector( sizeof( WIDTH - 1 ) - 1 DOWNT0 0 );      -- if width=32, count goes from 0
to 31
  SIGNAL      x_load, y_load, z_load,          x_reg, y_reg, z_reg,          x_result, y_result, z_result,
  cordic_coefficient,          shifted_y_reg, shifted_x_reg : std_logic_vector( WIDTH-1 DOWNT0 0 ) := (
others => '0' );
  SIGNAL      init, load, add_nsub, nadd_sub, rco, regs_load : std_logic;

  constant    x0 : std_logic_vector( WIDTH - 1 downto 0 ) := Conv2fixedPt ( inverseK, WIDTH );      --
1/1.64676025812107
  constant    y0 : std_logic_vector( WIDTH - 1 downto 0 ) := ( others => '0' );

BEGIN

  regs_load    <= init or load;      -- register load control signal      ( when '1' it loads the register )

```

```

-- cos( X ) register
x_load <= x0 when ( init = '1' ) else x_result;      -- initialization bus multiplexer for x register
COS_X_REG : data_reg GENERIC MAP ( WIDTH => WIDTH )
PORT MAP ( clk => clk, load => regs_load, X => x_load, Q => x_reg );

-- sin( Y ) register
y_load <= y0 when ( init = '1' ) else y_result;      -- initialization bus multiplexer for y register
SIN_Y_REG : data_reg GENERIC MAP ( WIDTH => WIDTH )
PORT MAP ( clk => clk, load => regs_load, X => y_load, Q => y_reg );

-- Z register
z_load <= angle when ( init = '1' ) else z_result;   -- initialization bus multiplexer for z register
ANGLE_Z_REG : data_reg GENERIC MAP ( WIDTH => WIDTH )
PORT MAP ( clk => clk, load => regs_load, X => z_load, Q => z_reg );

-- x( i+1 ) = K( i )*[ x( i ) - y( i )*d( i )*2^( -i ) ]
-- y( i+1 ) = K( i )*[ y( i ) + y( i )*d( i )*2^( -i ) ]
-- z( i+1 ) = z( i ) - d( i )*arctg( 2^( -i ) )
-- if ( z < 0 ) d( i ) = -1 else d( i ) = 1

-- add_nsub          x( i+1 )          y( i+1 )          z( i+1 )
--          0          subtraction          addition          subtraction
--          1          addition          subtraction          addition
add_nsub <= z_reg( WIDTH - 1 );      -- sign bit of z
nadd_sub <= NOT( add_nsub );

CORDIC_COUNTER: up_counter GENERIC MAP( MODULO => WIDTH )
PORT MAP( clk => clk, nRST => nRST, count_enable => load, count
=> count, rco => rco );

X_addsub : addsub GENERIC MAP ( WIDTH => WIDTH )
PORT MAP ( x_reg, shifted_y_reg, x_result, add_nsub );

Y_addsub : addsub GENERIC MAP ( WIDTH => WIDTH )
PORT MAP ( y_reg, shifted_x_reg, y_result, nadd_sub );

Z_addsub : addsub GENERIC MAP ( WIDTH => WIDTH )
PORT MAP ( z_reg, cordic_coefficient, z_result, add_nsub );

```

```

LUT          : CORDIC_ROM
              GENERIC MAP (      WIDTH => WIDTH,          SIZE => WIDTH )
              PORT MAP ( count, cordic_coefficient );

BARREL_SHIFT_X : barrel_shifter_sra
              GENERIC MAP (      WIDTH => WIDTH,          SIZE => WIDTH )
              PORT MAP ( x_reg, shifted_x_reg, count );

BARREL_SHIFT_Y : barrel_shifter_sra
              GENERIC MAP (      WIDTH => WIDTH,          SIZE => WIDTH )
              PORT MAP ( y_reg, shifted_y_reg, count );

FSM          : CORDIC_FSM PORT MAP (   clk => clk, nRST => nRST, start => start,
init, load => load, done => done );

              rco => rco, init =>

sin    <= y_reg;
cos    <= x_reg;

END rtl;

```

