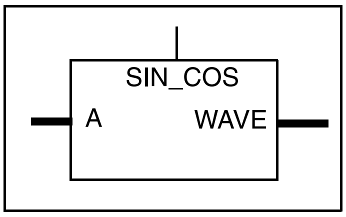
DW_sincos

Sine and Cosine

Version, STAR and Download Information: IP Directory

Features and Benefits

- ■Parameterized word length
- ■sine or cosine output by controlling `SIN_COS` port.
- ■DesignWare Datapath generator is employed for better timing and area.



Description

DW_ sincos is a fixed point sine and cosine unit that calculates `WAVE` = sin(πA) or cos(πA) where `A` and `WAVE` are fixed point value by controlling the `SIN_COS` pin. DW_sincos has two additional parameters: *arch* is for the implementation between the area-optimized and the speed-optimized, and *err_range* allows users to choose the error range between 1 ulp error and 2 ulp error.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
A	*A_width* bits	Input	Input data (radian)
SIN_COS	1 bit	Input	Function select: 0 = sine function 1 = cosine function
WAVE	*WAVE_width* bits	Output	Sine or cosine value

Table 1-2 Parameter Description

Parameter	Values	Description
A_width	2 to 34 bits	Word length of `A`
WAVE_width	2 to 35 bits	Word length of `WAVE`
arch	0 or 1 Default: 0	Select the implementation 0: area-optimized implementation 1: speed-optimized implementation
err_range	1 or 2 Default: 1	Select the error range 1 : |true value - calculated| < 1 ulp 2 : |true value - calculated| < 2 ulp

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

Table 1-4 Simulation Models

Model	Function
DW02.DW_SINCOS_CFG_SIM	Design unit name for VHDL simulation
dw/dw02/src/DW_sincos_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_sincos.v	Verilog simulation model source code

Functional Description

The input angle `A` is treated as a binary fixed point number which is converted to radians when multiplied by π. When `A` is interpreted as unsigned, the input angle `A` is a binary subdivision of the range $0 \leq A < 2$. When `A` is interpreted as signed (two's complement), the range is $-1 \leq A < 1$.

Table 1-5 Operations of DW_sincos

`sin_cos` **input**	`a` **input**	`wave` **output**
0	A (unsigned or two's complement)	sin(πa)
1	A (unsigned or two's complement)	cos(πa)

Input Data Format

The input value a is interpreted with the MSB as the only bit to the left of the decimal point. For example, if *A_width* = 6, the input value 111000 is interpreted as 1.11000.

The sine (or cosine) value calculated by DW_sincos is the same whether `A` is interpreted is an unsigned number or as a signed (two's complement) number. For example, consider the case where `SIN_COS` = 0 and the input value `A` is 1.10000. Interpreted as an unsigned number, the decimal equivalent of `A` is 3/2, the angle is 3π/2, and the sine value is -1 (decimal).

When 1.10000 is interpreted as a signed number, the MSB becomes the sign bit (1 in this example, indicating a negative number) and .10000 is the two's complement of .10000. The decimal equivalent is (-1/2). DW_sincos calculates the sine of (-π/2), which is also -1 (decimal).

Similarly, when `SIN_COS` = 1, DW_sincos calculates the cosine. If the input value (`A` = 1.10000) is interpreted as an unsigned number, DW_sincos calculates the cosine of 3π/2, which equals 0. If the input value (`A` = 1.10000) is interpreted as a two's complement number, DW_sincos calculates the cosine of (-π/2), which also equals 0.

Output Data Format

The output value WAVE is of the form SX.XXXX, where S is the sign bit and X.XXXX is the two's complement of the binary value of the sine or cosine. For example, the sine value 1 (decimal) is represented as 01.0000 (when *WAVE_width* = 6), and the sine value -1 is represented as 11.0000.

Table 6 shows the bit assignments for the input and output pins. Table 7 shows input and output values for an example implementation where *A_width* = 6, *WAVE_width* = 6.

Table 1-6 Input and Output Data Format

Pin	Range (binary)	Range (decimal)
A (unsigned)	1.111…111 (< 2) 1.111…110 … 1.000…000 (= 1) … 0.000…001 0.000…000 (= 0)	$0 \le A < 2$
A (two's complement)	0.111…111 (< 1) 0.111…110 … 0.000…000 (= 0) 1.111…111 (< 0) … 1.000…001 1.000…000 (= -1)	$-1 \le A < 1$
WAVE (two's complement)	01.000…000 (= 1) 00.111…111 … 00.000…000 (= 0) 11.111…111 (< 0) … 11.000…001 11.000…000 (= -1)	$-1 \le WAVE \le 1$

Table 1-7 Sample Data Value for *A_width* = 6 and *WAVE_width* = 6

Value of A	SIN_COS	Function	Value of WAVE
1.00000 (unsigned)	0	sin(π)	00.0000
1.10000 (unsigned)	0	sin(3π/2)	11.0000
1.00000 (two's complement)	1	cos(-π)	11.0000
0.10000 (two's complement)	0	sin(π/2)	01.0000

Alternative Implementation of Sine and Cosine with DW_lp_multifunc

The sine and cosine operation can also be implemented by DW_lp_multifunc component, which evaluates the value of sine and cosine with 1 ulp error bound. There will be 1 ulp difference between the value from DW_lp_multifunc and the value from DW_sincos. Performance and area of the synthesis results are different between the DW_sincos and sine and cosine implementation of the DW_lp_multifunc, depending on synthesis constraints, library cells and synthesis environments. By comparing performance and area between the sine and cosine implementation of DW_lp_multifunc and DW_sincos component, the DW_lp_multifunc provides more choices for the better synthesis results. Below is an example of the Verilog description for the sine and cosine of the DW_lp_multifunc. For more detailed information, see the DW_lp_multifunc datasheet.

```
DW_lp_multifunc #(op_width, 24) U1 (
     .A(A),
     .FUNC(FUNC),
     .Z(Z),
     .STATUS(STATUS)
);
```

Related Topics

- Datapath – Floating Point Overview
- DesignWare Building Block IP Documentation Overview

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DWpackages.all;
use DWARE.DW_Foundation_comp_arith.all;

entity DW_sincos_inst is
      generic (
        inst_A_width : INTEGER := 24;
        inst_WAVE_width : INTEGER := 25;
        inst_arch : INTEGER := 0;
        inst_err_range : INTEGER := 1
        );
      port (
        inst_A : in std_logic_vector(inst_A_width-1 downto 0);
        inst_SIN_COS : in std_logic;
        WAVE_inst : out std_logic_vector(inst_WAVE_width-1 downto 0)
        );
    end DW_sincos_inst;


architecture inst of DW_sincos_inst is

begin

    -- Instance of DW_sincos
    U1 : DW_sincos
    generic map (
          A_width => inst_A_width,
          WAVE_width => inst_WAVE_width,
          arch => inst_arch,
          err_range => inst_err_range
          )
    port map (
          A => inst_A,
          SIN_COS => inst_SIN_COS,
          WAVE => WAVE_inst
          );


end inst;
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_sincos_inst( inst_A, inst_SIN_COS, WAVE_inst );

parameter inst_A_width = 24;
parameter inst_WAVE_width = 25;
parameter inst_arch = 0;
parameter inst_err_range = 1;


input [inst_A_width-1 : 0] inst_A;
input inst_SIN_COS;
output [inst_WAVE_width-1 : 0] WAVE_inst;

    // Instance of DW_sincos
    DW_sincos #(inst_A_width, inst_WAVE_width, inst_arch, inst_err_range) U1 (
                .A(inst_A),
                .SIN_COS(inst_SIN_COS),
                .WAVE(WAVE_inst) );

endmodule
```