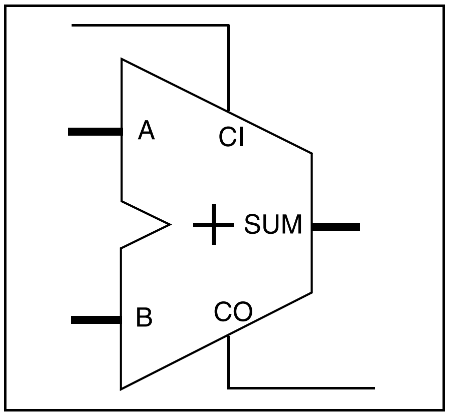
DW01_add

Adder

Version, STAR and Download Information: IP Directory

Features and Benefits

- Parameterized word length
- Carry-in and carry-out signals



Description

DW01_add adds two operands `A` and `B` with a carry-in `CI` to produce the output `SUM` with a carry-out `CO`.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
A	*width* bit(s)	Input	Input data
B	*width* bit(s)	Input	Input data
CI	1 bit	Input	Carry-in
SUM	*width* bit(s)	Output	Sum of (A + B + CI)
CO	1 bit	Output	Carry-out

Table 1-2 Parameter Description

Parameter	Values	Description
width	≥1	Word length of A, B, and SUM

Table 1-3 Synthesis Implementations[a]

Implementation	Function	License Feature Required
rpl	Ripple-carry synthesis model	none
cla	Carry-look-ahead synthesis model	none
pparch	Delay-optimized flexible parallel-prefix	DesignWare

Table 1-3 Synthesis Implementations[a] (Continued)

Implementation	Function	License Feature Required
apparch	Area-optimized flexible architecture that can be optimized for area, for speed, or for area, speed	DesignWare

a. During synthesis, Design Compiler will select the appropriate architecture for your constraints. However, you may force Design Compiler to use any architectures described in this table. For more, see *DesignWare Building Block IP User Guide*

Table 1-4 Simulation Models

Model	Function
DW01.DW01_ADD_CFG_SIM	Design unit name for VHDL simulation
dw/dw01/src/DW01_add_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW01_add.v	Verilog simulation model source code

Refer to Application Note AN 98-001 for detailed information regarding inferring carry-in and carry-out bits.

Related Topics

- Math – Arithmetic Overview
- DesignWare Building Block IP Documentation Overview

HDL Usage Through Operator Inferencing - VHDL

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

entity  DW01_add_oper is
  generic(wordlength: integer := 8);
  port(in1, in2 : in STD_LOGIC_VECTOR(wordlength-1 downto 0);
       sum      : out STD_LOGIC_VECTOR(wordlength-1 downto 0));
end DW01_add_oper;

architecture oper of DW01_add_oper is
  signal in1_signed, in2_signed, sum_signed: SIGNED(wordlength-1 downto 0);
begin
  in1_signed <= SIGNED(in1);
  in2_signed <= SIGNED(in2);
  -- infer the "+" addition operator
  sum_signed <= in1_signed + in2_signed;
  sum <= STD_LOGIC_VECTOR(sum_signed);
end oper;
```

HDL Usage Through Operator Inferencing - Verilog

```
module DW01_add_oper(in1,in2,sum);
  parameter wordlength = 8;

  input [wordlength-1:0] in1,in2;
  output [wordlength-1:0] sum;

  assign  sum = in1 + in2;
endmodule
```

HDL Usage Through Component Instantiation - VHDL

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

entity DW01_add_inst is
  generic ( inst_width : NATURAL := 8 );
  port ( inst_A    : in std_logic_vector(inst_width-1 downto 0);
         inst_B    : in std_logic_vector(inst_width-1 downto 0);
         inst_CI   : in std_logic;
         SUM_inst : out std_logic_vector(inst_width-1 downto 0);
         CO_inst   : out std_logic );
end DW01_add_inst;

architecture inst of DW01_add_inst is
begin

  -- Instance of DW01_add
  U1 : DW01_add
  generic map ( width => inst_width )
  port map ( A => inst_A, B => inst_B, CI => inst_CI,
             SUM => SUM_inst, CO => CO_inst );
end inst;

-- pragma translate_off
configuration DW01_add_inst_cfg_inst of DW01_add_inst is
  for inst
  end for; -- inst
end DW01_add_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW01_add_inst( inst_A, inst_B, inst_CI, SUM_inst, CO_inst );

  parameter width = 8;

  input [width-1 : 0] inst_A;
  input [width-1 : 0] inst_B;
  input inst_CI;
  output [width-1 : 0] SUM_inst;
  output CO_inst;

  // Instance of DW01_add
  DW01_add #(width)
    U1 (.A(inst_A), .B(inst_B), .CI(inst_CI), .SUM(SUM_inst), .CO(CO_inst) );

endmodule
```

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043

www.synopsys.com