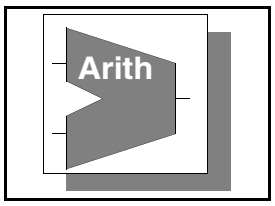
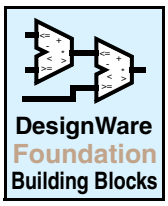




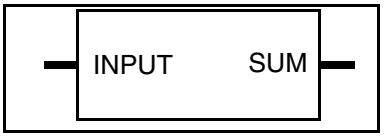
DW02_sum

Vector Adder

Version, STAR and Download Information: IP Directory

Features and Benefits

- Parameterized number of inputs
- Parameterized word length
- Multiple synthesis implementations
- Inferable using a function call



Description

DW02_sum performs a summation of a set of words (INPUT) into a vector result.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
INPUT	*num_inputs* × *input_width* bit(s)	Input	Concatenated input data
SUM	*input_width* bit(s)	Output	Sum

Table 1-2 Parameter Description

Parameter	Values	Description
num_inputs	≥ 1	Number of inputs
input_width	≥ 1	Word length of inputs and sum

Table 1-3 Synthesis Implementations[a]

Implementation Name	Function	License Feature Required
pparch	Delay-optimized flexible parallel-prefix	DesignWare
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 Obsolete Synthesis Implementations[a]

Implementation	Function	Replacement Implementation
csa	Carry-save array synthesis model	pparch
wallace	Wallace-tree synthesis model	pparch
rpl	Ripple-carry synthesis model	apparch

a. DC versions and DesignWare EST releases linked to DC versions prior to 2007.03 will still include these implementations.

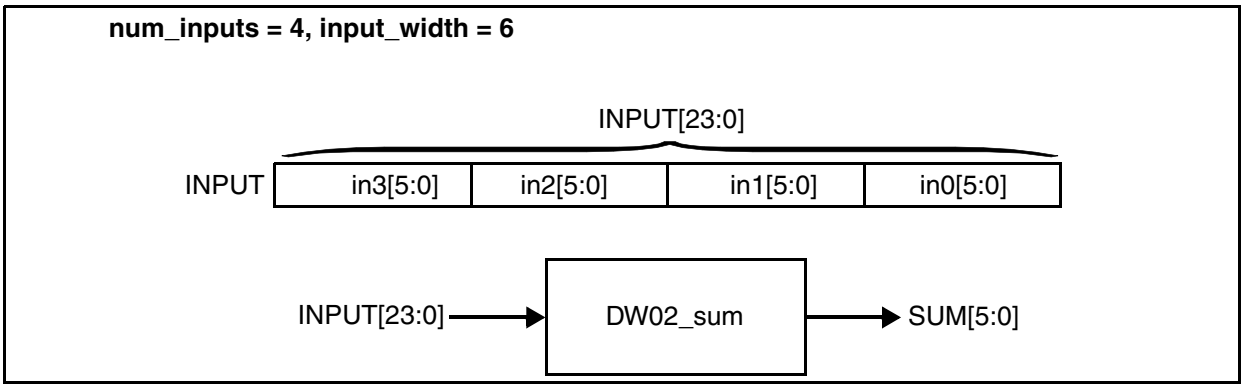
Table 1-5 Simulation Models

Model	Function
DW02.DW02_SUM_CFG_SIM	Design unit name for VHDL simulation
dw/dw02/src/DW02_sum_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW02_sum.v	Verilog simulation model source code

Table 1-6 Functional Description

INPUT	SUM
INPUT	INPUT[*input_width*–1:0] + INPUT[(2∗*input_width)*–1:*input_width*] + INPUT[(3∗*input_width)*–1:2∗*input_width*] + ... + INPUT[(*num_inputs*∗*input_width)*–1:(*num_inputs*–1)∗*input_width*]

Figure 1-1 Functional Operation



The equation for the SUM is as follows:

$$\mathrm{SUM}(m\text{-}1{:}0) = \sum_{j=0}^{N-1} a[(j+1)\times m\text{-}1{:}j\times m]$$

where:

N= *num_inputs*
m = *input_width*

The set of words to be summed must be concatenated into a single word with a length of *num_inputs* × *input_width*. This single word is connected to the INPUT pin. Internally, DW02_sum disassembles the individual words from INPUT and then performs the summation.

To preserve the complete precision of sum values, you must extend the input values before using DW02_sum. For example, when summing three 8-bit unsigned values, the maximum value on each input results in a sum value of (255 × 3 = 765), which requires 10 bits to be fully represented. The number of bits of extension required to maintain full precision is `ceiling(log2(num_inputs))`. When using unsigned values, extend with zeros. When using signed values, sign-extend the input values.

Performance

For addition operations involving three or more operands, (for example, Z=A+B+C), the DW02_sum Vector Adder provides significantly better synthesis results than multiple instances of the DW01_add two-input Adder.

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

Related Topics

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

HDL Usage Through Function Inferencing - VHDL

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

```
library IEEE, DWARE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use DWARE.DW_foundation_arith.all;

entity DW02_sum_func is
  generic(func_num_inputs, func_input_width: INTEGER :=8);
  port(func_INPUT   : in std_logic_vector(func_num_inputs *
                                          func_input_width-1 downto 0);
       SUM_func_UNS : out std_logic_vector(func_input_width-1 downto 0);
       SUM_func_TC  : out std_logic_vector(func_input_width-1 downto 0) );
end DW02_sum_func;

architecture func of DW02_sum_func is
begin
  SUM_func_UNS <= std_logic_vector(DWF_sum(UNSIGNED(func_INPUT),
                                           func_num_inputs));
  SUM_func_TC  <= std_logic_vector(DWF_sum(SIGNED(func_INPUT),
                                           func_num_inputs));
end func;
```

HDL Usage Through Function Inferencing - Verilog

```
module DW02_sum_func(SUM_func,func_INPUT);
  parameter func_num_inputs = 8;
  parameter func_input_width = 8;

  // Passes the widths to the vector adder function
  parameter num_inputs = func_num_inputs;
  parameter input_width = func_input_width;

  // Please add search_path = search_path + {synopsys_root + "/dw/sim_ver"}
  // to your .synopsys_dc.setup file (for synthesis) and add
  // +incdir+$SYNOPSYS/dw/sim_ver+ to your verilog simulator command line
  // (for simulation).
  `include "DW02_sum_function.inc"

  input [func_input_width*func_num_inputs-1:0] func_INPUT;

  output [func_input_width*func_num_inputs-1:0] SUM_func;

  wire [func_input_width*func_num_inputs-1:0] SUM_func;

  // infer DW02_sum
  assign SUM_func = DWF_sum(func_INPUT);
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 DW02_sum_inst is
  generic ( inst_num_inputs  : NATURAL := 8;
            inst_input_width : NATURAL := 8 );
  port ( inst_INPUT : in std_logic_vector(inst_num_inputs*
                                          inst_input_width-1 downto 0);
         SUM_inst   : out std_logic_vector(inst_input_width-1 downto 0) );
end DW02_sum_inst;

architecture inst of DW02_sum_inst is
begin
  -- Instance of DW02_sum
  U1 : DW02_sum
    generic map ( num_inputs => inst_num_inputs,
                  input_width => inst_input_width )
  port map ( INPUT => inst_INPUT,   SUM => SUM_inst );
end inst;

-- pragma translate_off
configuration DW02_sum_inst_cfg_inst of DW02_sum_inst is
  for inst
  end for; -- inst
end DW02_sum_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW02_sum_inst( inst_INPUT, SUM_inst );

  parameter num_inputs = 8;
  parameter input_width = 8;

  input [num_inputs*input_width-1 : 0] inst_INPUT;
  output [input_width-1 : 0] SUM_inst;

  // Instance of DW02_sum
  DW02_sum #(num_inputs,  input_width)
    U1 ( .INPUT(inst_INPUT),  .SUM(SUM_inst) );

endmodule
```

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.

All other product or company names may be trademarks of their respective owners.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

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

www.synopsys.com