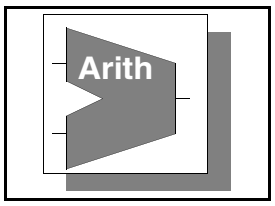
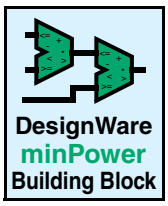


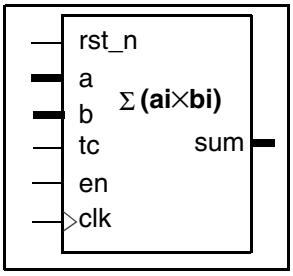


DW_prod_sum_pipe

Stallable Pipelined Generalized Sum of Products

Version, STAR and Download Information: IP Directory

Features and Benefits



- Parameterized word length
- Unsigned and signed (two's complement) data operation
- Parameterized number of pipeline stages
- Parameterized stall mode (stallable or non-stallable)
- Parameterized reset mode (no reset, asynchronous or synchronous reset)
- Automatic pipeline retiming

Provides minPower benefits with the DesignWare-LP license.

Description

DW_prod_sum_pipe is a universal stallable pipelined generalized sum of products generator.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
clk	1 bit	Input	Input clock
rst_n	1 bit	Input	Reset, active-low (not used if parameter *rst_mode*=0)
en	1 bit	Input	Register enable, active high (used only if parameter *stall_mode* = 1) 0 = stall 1 = enable register
tc	1 bit	Input	Two's complement control 0 = unsigned 1 = signed
a	*a_width* × *num_inputs* bit(s)	Input	Concatenated input data vector
b	*b_width* × *num_inputs* bit(s)	Input	Concatenated input data vector
sum	*sum_width* bit(s)	Output	Pipelined data summation

Table 1-2 Parameter Description

Parameter	Values	Description
a_width	≥ 1 Default: None	Word length of `a`
b_width	≥ 1 Default: None	Word length of `b`
num_inputs	≥1 Default: 2	Number of inputs
num_stages	≥ 2 Default: 2	Number of pipeline stages
stall_mode	0 or 1 Default: 1	Stall mode 0 = non-stallable 1 = stallable
rst_mode	0 to 2 Default: 1	Reset mode 0 = no reset 1 = asynchronous reset 2 = synchronous reset
sum_width	≥ 48 Default: None	Word length of sum
op_iso_mode	0 to 4 Default: 0	Operand Isolation Mode (controls datapath gating for minPower flow) If *stall_mode* is 0, this parameter is ignored and no isolation is applied. 0 = DC variable (DW_lp_op_iso_mode) can control the op iso mode 1 = 'none' 2 = 'and' 3 = 'or' 4 = preferred isolation style: 'and'

Table 1-3 Synthesis Implementations

Implementation Name	Implementation	License Feature Required
str[a]	Pipelined str synthesis model	DesignWare

a. Either pparch or apparch implementation is selected based on the constraints of the design.

Table 1-4 Simulation Models

Model	Function
DW02.DW_PROD_SUM_PIPE_CFG_SIM	Design unit name for VHDL simulation

Table 1-4 Simulation Models (Continued)

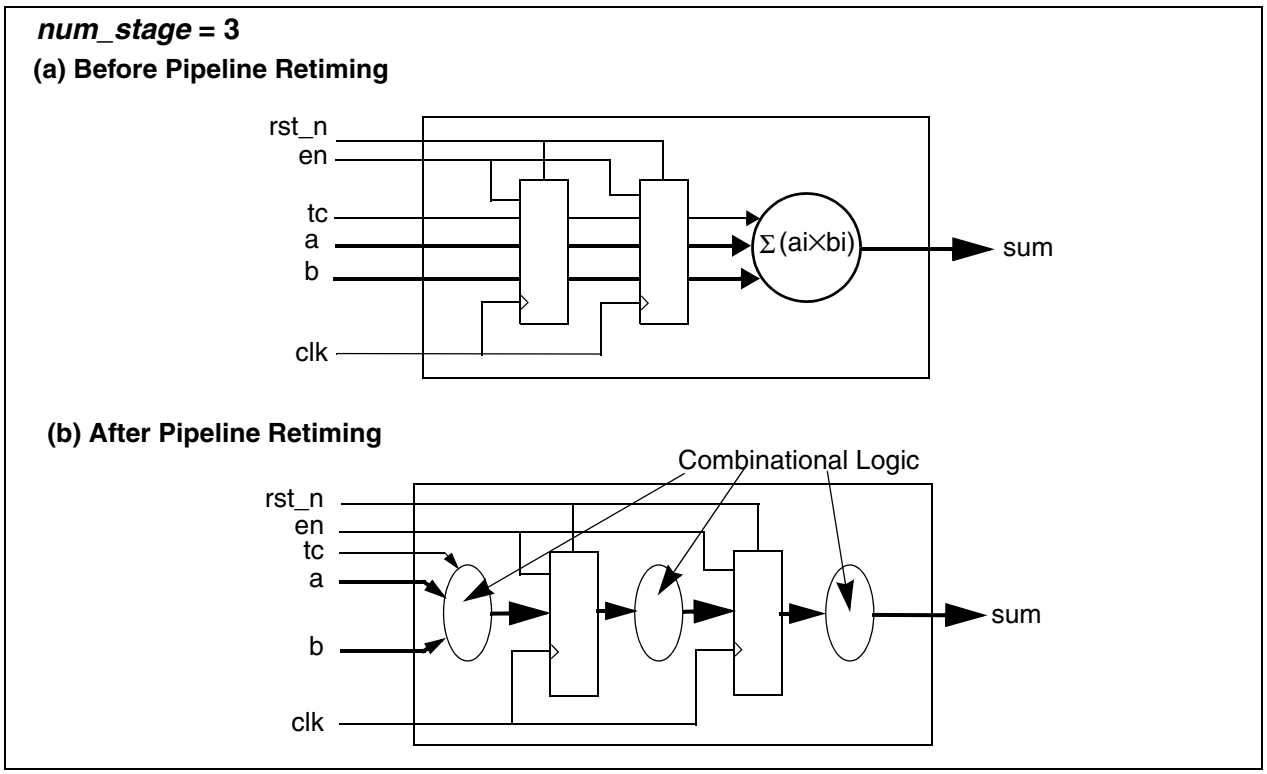
Model	Function
dw/dw02/src/DW_prod_sum_pipe_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_prod_sum_pipe.v	Verilog simulation model source code

DW_prod_sum_pipe computes the generalized sum of products of input vectors `a` and `b` with a latency of `num_stages`–1 clock cycles. The vectors `a` and `b` are formed by concatenation of corresponding inputs to be multiplied and summed. The input pin `tc` determines whether the input and output data is interpreted as unsigned (`tc`=0) or signed (`tc`=1) numbers.

Automatic pipeline retiming ensures optimal placement of pipeline registers within the generalized sum of products generator to achieve maximum throughput. The pipeline can be stalled by setting the load enable signal `en` to a low (when *stall_mode*=1). The pipeline registers can either have no reset (*rst_mode*=0) or an asynchronous (*rst_mode*=1) or synchronous reset (*rst_mode*=2) connected to the reset signal `rst_n`.

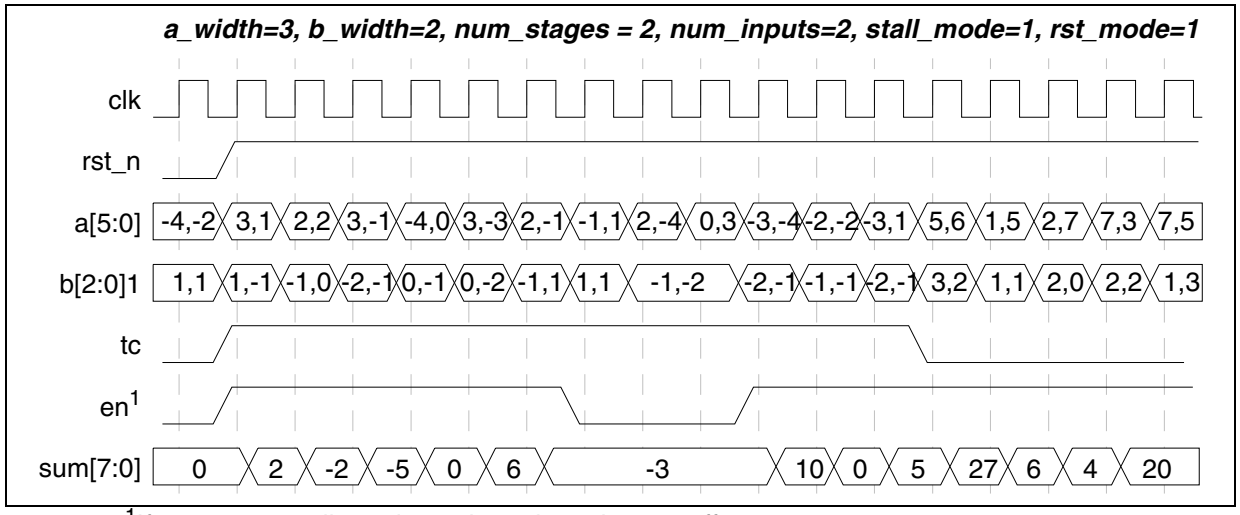
Figure 1-1 shows the block diagram of a 3-stage DW_prod_sum_pipe component.

Figure 1-1 Pipeline Retiming



Timing Waveform

Figure 1-2 Waveform 1



[1]If parameter stall_mode=0, then pin en has no effect.

Figure 1-3 Waveform 2

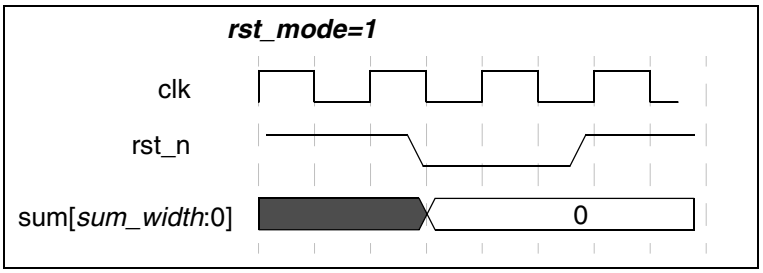
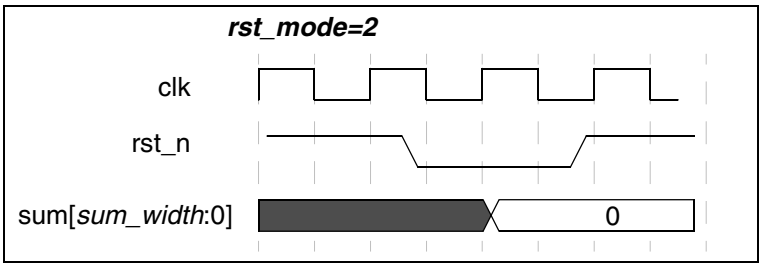


Figure 1-4 Waveform 3



Related Topics

- Math – Arithmetic 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_prod_sum_pipe_inst is
  generic (inst_a_width    : POSITIVE := 2;  inst_b_width     : POSITIVE := 2;
           inst_num_inputs : POSITIVE := 2;  inst_sum_width   : POSITIVE := 4;
           inst_num_stages : POSITIVE := 2;  inst_stall_mode  : NATURAL := 1;
           inst_rst_mode   : NATURAL := 1;   inst_op_iso_mode : NATURAL := 0 );
  port (inst_clk   : in std_logic;
        inst_rst_n : in std_logic;
        inst_en    : in std_logic;
        inst_tc    : in std_logic;
        inst_a     : in std_logic_vector(inst_a_width*inst_num_inputs-1 downto 0);
        inst_b     : in std_logic_vector(inst_b_width*inst_num_inputs-1 downto 0);
        sum_inst   : out std_logic_vector(inst_sum_width-1 downto 0) );
end DW_prod_sum_pipe_inst;

architecture inst of DW_prod_sum_pipe_inst is
begin
  -- Instance of DW_prod_sum_pipe
  U1 : DW_prod_sum_pipe
    generic map (a_width => inst_a_width,   b_width => inst_b_width,
                 num_inputs => inst_num_inputs,  sum_width => inst_sum_width,
                 num_stages => inst_num_stages,   stall_mode => inst_stall_mode,
                 rst_mode => inst_rst_mode,  op_iso_mode => inst_op_iso_mode  )
    port map (clk => inst_clk,   rst_n => inst_rst_n,
              en => inst_en,   tc => inst_tc,   a => inst_a,   b => inst_b,
              sum => sum_inst  );
end inst;

-- Configuration for use with VSS simulator
-- pragma translate_off
configuration DW_prod_sum_pipe_inst_cfg_inst of DW_prod_sum_pipe_inst is
  for inst
  end for; -- inst
end DW_prod_sum_pipe_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_prod_sum_pipe_inst( inst_clk, inst_rst_n, inst_en, inst_tc,
                              inst_a, inst_b, sum_inst );

  parameter inst_a_width = 2;
  parameter inst_b_width = 2;
  parameter inst_num_inputs = 2;
  parameter inst_sum_width = 4;
  parameter inst_num_stages = 2;
  parameter inst_stall_mode = 1;
  parameter inst_rst_mode = 1;
  parameter inst_op_iso_mode = 0;

  input inst_clk;
  input inst_rst_n;
  input inst_en;
  input inst_tc;
  input [inst_a_width*inst_num_inputs-1 : 0] inst_a;
  input [inst_b_width*inst_num_inputs-1 : 0] inst_b;
  output [inst_sum_width-1 : 0] sum_inst;

  // Instance of DW_prod_sum_pipe
  DW_prod_sum_pipe #(inst_a_width, inst_b_width, inst_num_inputs,
                     inst_sum_width, inst_num_stages, inst_stall_mode,
                     inst_rst_mode, inst_op_iso_mode)
    U1 (.clk(inst_clk),    .rst_n(inst_rst_n),
        .en(inst_en),      .tc(inst_tc),
        .a(inst_a),        .b(inst_b),
        .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