Key Design Features

- Synthesizable, technology independent VHDL IP Core
- Function y = a * b
- Input values as signed or unsigned numbers
- Output values as signed or unsigned numbers
- Configurable data with and pipeline depth
- Supports both LUT-based or hard multiplier blocks
- Includes a classic shift-add multiplier for larger width implementations
- High-speed fully pipelined architecture

Applications

- Fixed-point mathematics
- Fundamental building block in all digital processing functions

Pin-out Description

Pin name	*I/O*	*Description*	*Active state*
clk	in	Synchronous clock	rising edge
en	in	Clock enable	high
a_in [dw - 1:0]	in	Input value #1	data
b_in [dw - 1:0]	in	Input value #2	data
product [dw*2 - 1:0]	out	Output product	data

Generic Parameters

Generic name	*Description*	*Type*	*Valid range*
dw	Input data width	integer	≥ 1
levels	Number of pipeline stages	integer	≥ 1
style	Multiplier style (compiler hint)	string	Altera®: dsp, logic Xilinx®: auto, block, lut pipe_lut, etc.
use_signed	Use signed or unsigned arithmetic	boolean	TRUE/FALSE

Block Diagram

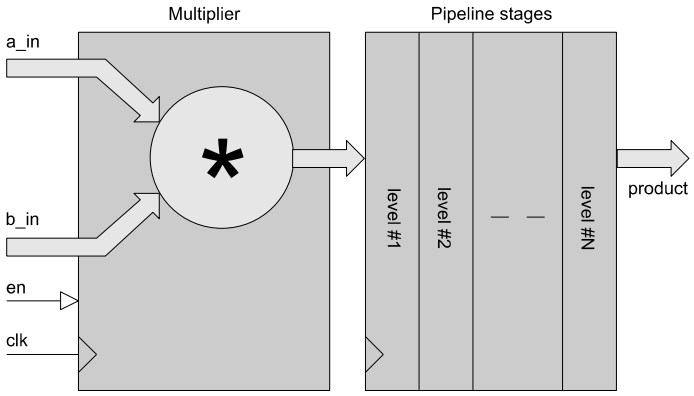

Figure 1: Pipelined multiplier architecture (conceptual model)

General Description

PIPE_MULT (Figure 1) is a general purpose multiplier with a configurable data width and configurable number of pipeline stages. Input values are accepted as either signed or unsigned numbers depending on the generic setting *use_signed*. Likewise, output values are either signed or unsigned depending on the same setting.

The number of pipeline stages may be programmed using the generic parameter *levels*. By changing this value, a multiplier may be generated which trades off latency against maximum attainable clock frequency.

In addition, the pipelined multiplier component also includes a compiler hint generic setting. By modifying this setting, the compiler can be instructed to infer LUT-based or hard multiplier/DSP resources.

Values are sampled on the rising clock-edge of *clk* when *en* is high. The function has a clock-cycle latency which is equal to the number of pipeline levels.

Functional Timing

Figure 2 demonstrates the computation of: a_in * b_in, where a = 0xA92F (-22225 in decimal) and b = 0x712C (28972 in decimal). In this example, the parameters have been set to dw = 16, levels = 3, use_signed = true. The result, 0xD99ED314 (-643902700) has a latency of 3 clock cycles.

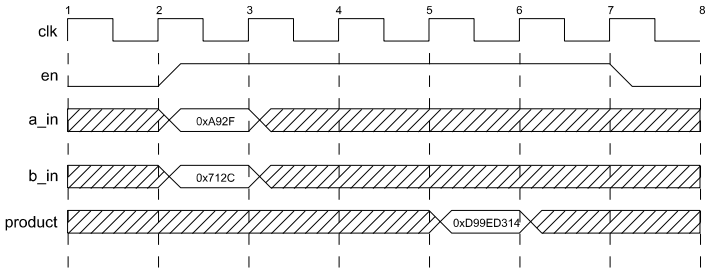

*Figure 2: Calculation of a * b*

Source File Description

All source files are provided as text files coded in VHDL. The following table gives a brief description of each file.

Source file	Description
pipe_mult_reg.vhd	Pipeline register block
pipe_mult_classic.vhd	Classic pipelined multiplier (More suited to large width LUT-based implementations)
pipe_mult_classic_unsigned.vhd	Classic pipelined multiplier (Unsigned version)
pipe_mult.vhd	Top-level block
pipe_mult_bench.vhd	Top-level test bench

Functional Testing

An example VHDL testbench is provided for use in a suitable VHDL simulator. The compilation order of the source code is as follows:

1. pipe_mult_reg.vhd
2. pipe_mult.vhd
3. pipe_mult_bench.vhd

The VHDL testbench instantiates the multiplier component and the user may modify the generic parameters as required. The simulation must be run for at least 2 ms during which time the multiplier will be driven with a randomized sequence input values. The test terminates automatically.

The simulation generates two text files called: *pipe_mult_in.txt* and *pipe_mult_out.txt*. These files respectively contain the input and output data samples captured at the interfaces during the test.

Figure 3 shows the results of the multiplier used to implement the function $f(x) = x^2$. Results are shown for the first 1000 samples.

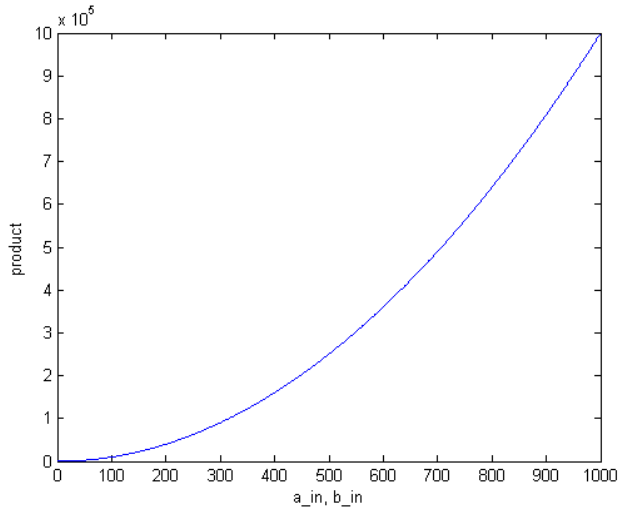

Figure 3: Plot of test results for function: $f(x) = x^2$

Synthesis

The source files required for synthesis and the design hierarchy is shown below:

- pipe_mult.vhd
  - pipe_mult_reg.vhd

The VHDL core is designed to be technology independent. However, as a benchmark, synthesis results have been provided for the Xilinx® Virtex 6 and Spartan 6 FPGA devices. Synthesis results for other FPGAs and technologies can be provided on request.

Synthesis results are shown with the generic parameters set to: dw = 32, levels = 5, style = auto, use_signed = true.

Note that increasing the number of pipeline levels will increase the maximum attainable clock frequency (up to a point) for a given multiplier data width.

Two additional 'Classic' implementations of the pipelined multiplier are also provided with the source code In some instances these may give better results than the standard 'pipe_mult.vhd' component. These files are called: 'pipe_mult_classic.vhd' and 'pipe_mult_classic_unsigned.vhd'.

These versions of the multiplier have a fixed latency of 4 and 3 cycles respectively. They are coded as a series of partial products, shifts and adds and are generally more suited to LUT-based or very wide multiplier implementations.

Resource usage is specified after Place and Route.

VIRTEX 6

Resource type	Quantity used
Slice register	49
Slice LUT	17
Block RAM	0
DSP48	4
Occupied slices	10
Clock frequency (approx)	650 MHz

SPARTAN 6

Resource type	Quantity used
Slice register	49
Slice LUT	22
Block RAM	0
DSP48	4
Occupied slices	9
Clock frequency (approx)	200 MHz

Revision History

Revision	*Change description*	*Date*
1.0	Initial revision	22/07/2008
1.1	Included additional 'Classic' pipelined multiplier implementations	30/10/2010
1.1	Updated synthesis results for Xilinx® 6 series FPGAs	07/06/2012
1.3	Added multiplier 'style' generic setting	17/07/2014

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for Development Software ***category:***

Click to view products by Zipcores ***manufacturer:***

Other Similar products are found below :

SRP004001-01 SW163052 SYSWINEV21 WS01NCTF1E W128E13 SW89CN0-ZCC IP-UART-16550 MPROG-PRO535E AFLCF-08-LX-CE060-R21 WS02-CFSC1-EV3-UP SYSMAC-STUDIO-EIPCPLR 1120270005 SW006021-2H ATATMELSTUDIO 2400573 2702579 2988609 SW006022-DGL 2400303 88970111 DG-ACC-NET-CD 55195101-101 55195101-102 SW1A-W1C MDK-ARM SW006021-2NH B10443 SW006021-1H SW006021-2 SW006022-2 SW006023-2 SW007023 MIKROE-730 MIKROE-2401 MIKROE-499 MIKROE-722 MIKROE-724 MIKROE-726 MIKROE-728 MIKROE-732 MIKROE-734 MIKROE-736 MIKROE-738 MIKROE-744 MIKROE-928 MIKROE-936 1120270002 1120270003 1120275015 NT-ZJCAT1-EV4