Key Design Features

- Synthesizable, technology independent VHDL Core
- Function f(x) = √x
- Input and output values as unsigned integers
- Configurable data width
- Unsigned N-bit input generates a result with N/2 integer bits and N/2 fraction bits
- Precision to within $1/2^{(N/2)}$
- High-speed fully pipelined architecture with configurable number of register stages for area/speed trade off
- One output result per clock-cycle (i.e. pipelined operation)
- Capable of clock speeds of 250MHz+ on even basic FPGA platforms

Applications

- Basic building block in digital processing functions
- Square root of integers and fixed-point numbers[1]
- Magnitude calculation of complex signals in digital communications systems

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	Unsigned input number	data
sqrt [dw -1:0]	out	Unsigned output square root	data

Generic Parameters

Generic name	***Description***	***Type***	***Valid range***
dw	Input data width	integer	≥ 2 (dw = even)
reg_stages	Number of pipeline register stages	integer	≥ 1 ≤ dw

1 For fixed-point numbers then inputs must be pre-scaled by a power of 2 and the result wire-shifted. E.g. the computation √0.2 could be done as √51 with a shift right of 4

Block Diagram

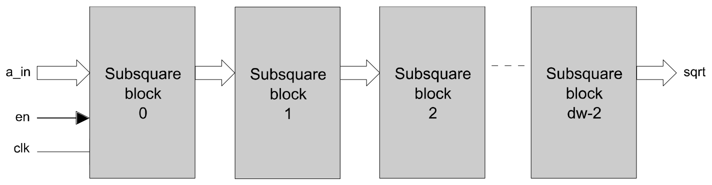

Figure 1: Pipelined Square-root Architecture

General Description

PIPE_SQRT (Figure 1) is a pipelined square-root with configurable data width. The design is fully scalable and modular permitting the user to specify large bit widths without compromising maximum attainable clock speed.

The function accepts input values as unsigned integers whose width is specified by the parameter *dw*. The output result also contains *dw* bits and is separated into an integer part and fractional part each of *dw*/2 bits wide.

Figure 2 below shows an example calculation using 8-bit arithmetic. The input √00111101 (√61) would result in an output of: 0111.1101 – i.e. an integer part of 7 and a fractional part of 0.8125.

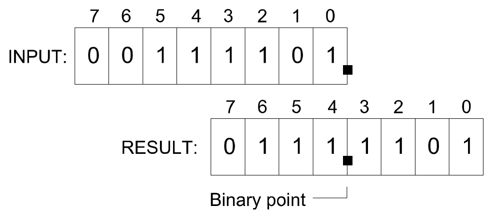

Figure 2: 8-bit square-root example

Values are sampled on the rising clock-edge of *clk* when *en* is high. The number of register stages in the pipeline may be modified in order to trade off maximum speed against the total resource used. The overall pipeline latency of the square root function is given by the following formula:

$$Latency = (dw-1)/reg\_stages$$

The total latency should be rounded up to the nearest whole number. For instance, when the generic parameter *dw* = 16 and *reg_stages* is set to 3, then the function has a fixed cycle latency of 16-1/3 = 5 clock cycles. In other words, the result of the square-root will take 5 clock cycles to appear at the output.

Note that while the latency may change depending on the *reg_stages* setting, the throughput is always maintained at one output result per clock.

Functional Timing

Figure 3 demonstrates the calculation of √57728. In this example, the generic parameter *dw* has been set to 16 and *reg_stages* is set to 5. Note that the result has a latency of 3 clock cycles. The result is given as 61509, which as an 8.8-bit value represents the number 240.27.

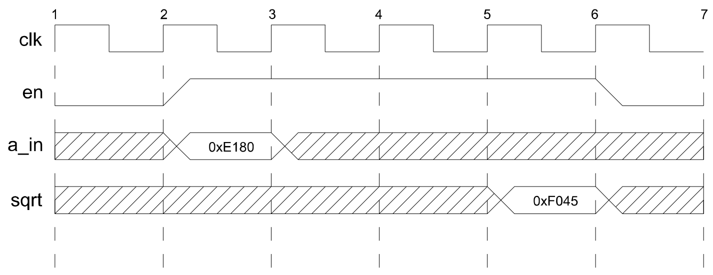

Figure 3: Calculation of the square-root of 57728 with a data width of 16-bits

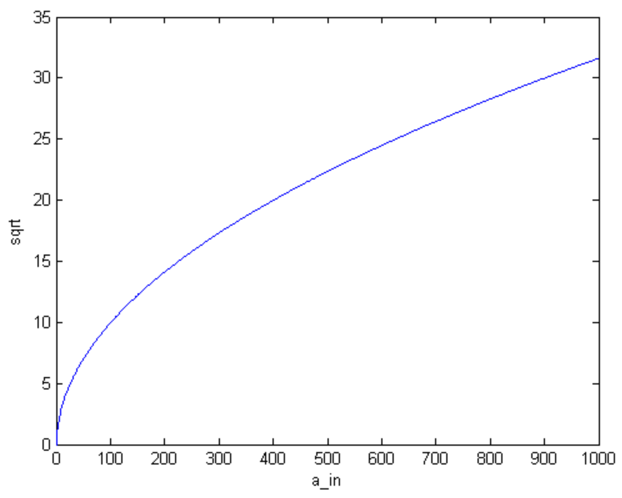

Figure 4: Plot of test results for function: f(x) = √x

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_sqrt_subsquare.vhd	Subtract square block
pipe_sqrt.vhd	Top-level block
pipe_sqrt_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_sqrt_subsquare.vhd
2. pipe_sqrt.vhd
3. pipe_sqrt_bench.vhd

The VHDL testbench instantiates the square-root 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 square root function will be driven with a randomized sequence of input values. The test terminates automatically.

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

Figure 4 shows the results of the square-root function for the first 1000 natural numbers with the generic parameter dw = 16.

Synthesis

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

- pipe_sqrt.vhd
  - pipe_sqrt_subsquare.vhd

The VHDL core is designed to be technology independent. However, as a benchmark, synthesis results have been provided for the Xilinx Virtex 5 and the Altera Stratix III series of FPGA devices. The lowest and highest speed grade devices have been chosen in both cases for comparison.

Note that the generic parameter *reg_stages* will have a signification effect on the speed and area of the synthesized design. For the fastest possible design, the generic parameter reg_stages should be set to 1. For the smallest design, then reg_stages should be set to equal the data width, dw.

Trial synthesis results are shown with the generic parameters set to dw = 16 and reg_stages = 1. Resource usage is specified after Place and Route.

VIRTEX 5

Resource type	***Quantity used***
Slice register	329
Slice LUT	49
Block RAM	0
DSP48	15
Clock frequency (worst case)	161 MHz
Clock frequency (best case)	264 MHz

STRATIX III

Resource type	***Quantity used***
Register	360
ALUT	303
Block Memory bit	0
DSP block 18	26
Clock frequency (worse case)	125 MHz
Clock frequency (best case)	180 MHz

Revision History

Revision	***Change description***	***Date***
1.0	Initial revision	23/07/2008
1.1	Added new opt_mode generic parameter	06/10/2008
1.2	Renamed *opt_mode* parameter to *reg_stages*. Design now supports any number of register stages up to the maximum *dw*	01/11/2011