

## Key Design Features

- Synthesizable, technology independent VHDL Core
- Function  $y = \cos(x)$
- Input range  $0 \leq x \leq \pi/2$  (Quarter wave)
- Output range  $0 \leq y \leq 1$
- Based on a quadratic polynomial with dynamic coefficients
- Input values as 16-bit unsigned fractions
- Output values as 16-bit unsigned fractions in radians
- Accurate to within 0.0002
- High-speed fully pipelined architecture
- 3 clock-cycle latency

## Applications

- Fixed-point mathematics
- Quadrature signal generation in digital communications
- Alternative to using a 65536 x 16-bit LUT (128kbytes)

## Pin-out Description

Pin name	I/O	Description	Active state
clk	in	Synchronous clock	rising edge
en	in	Clock enable	high
x_in [15:0]	in	Input value in radians	data
y_out [15:0]	out	Output value	data

## Functional Specification

Value	Type	Valid range
x_in [15:0]	16-bit unsigned fraction in [16 15] format	$[0, \pi/2]$
y_out [15:0]	16-bit unsigned fraction in [16 15] format	$[0, 1]$
		Accuracy to within 0.0002

## Block Diagram

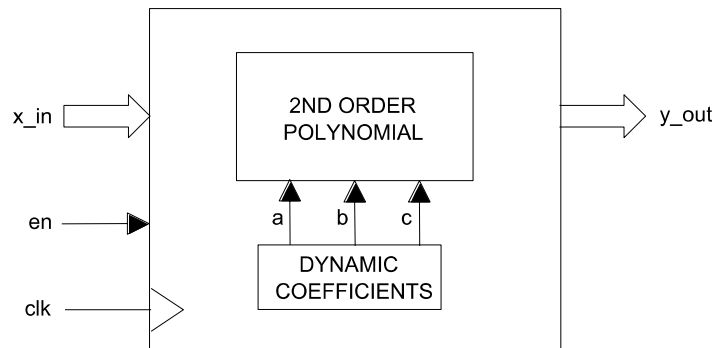


Figure 1: cos\_x function architecture

## General Description

COS\_X (Figure 1) calculates the cosine of an angle in radians. It has a fully pipelined architecture and uses fixed-point mathematics throughout. Input values are accepted as 16-bit unsigned values in the range 0 to  $\pi/2$ . Output values are 16-bit unsigned values in the range 0 to 1. For input values greater than  $\pi/2$ , the output clips to 0. Both input and output values are in [16 15] format with 1 integer bit and 15 fraction bits. As an example the input value 0xC000 would signify the value 1.5.

Internally, the function uses a 2<sup>nd</sup> order polynomial of the form:

$$y = ax^2 + bx + c$$

The coefficients a, b and c dynamically change with respect to the input value in order to generate a more accurate approximation. The output result is accurate to within 0.0002.

Values are sampled on the rising clock-edge of *clk* when *en* is high. The function has a 3 clock-cycle latency.

## Functional Timing

Figure 2 demonstrates the computation of  $y = \cos(x)$ , where  $x = 0x37E4$  (0.4366 as a decimal fraction). The result, 0x73FD (0.9062 in decimal) has a latency of 3 clock cycles.

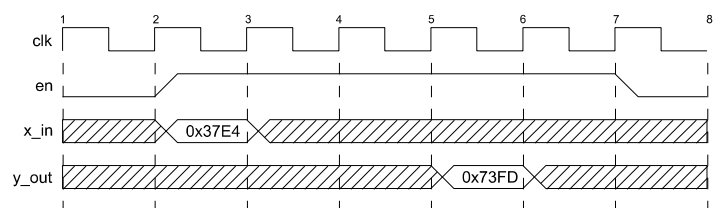


Figure 2: Calculation of  $y = \cos(x)$

## 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
cos_x.vhd	Top-level block
cos_x_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. cos\_x.vhd
2. cos\_x\_bench.vhd

The simulation must be run for at least 1 ms during which time a 16-bit input stimulus in the range 0 to 65535 will be generated. The test terminates automatically.

The simulation generates a text file called *cos\_x\_out.txt*. This file contains the output results captured during the test. The results of the test are shown graphically in Figure 3 below:

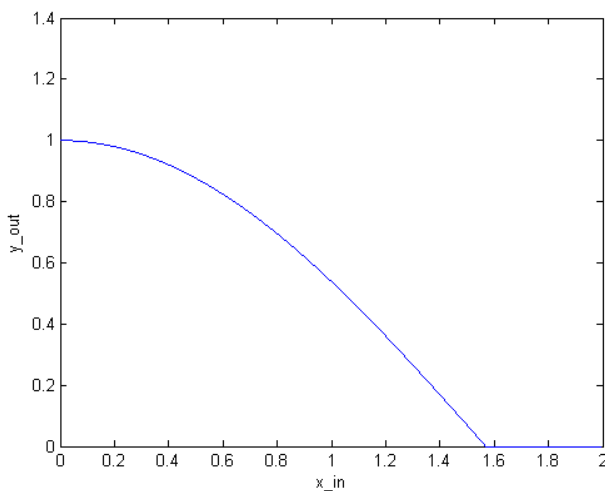


Figure 3: Plot of test results for cos\_x function

## Synthesis

The source file 'cos\_x.vhd' is the only file required for synthesis. There are no sub-modules in the design.

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.

Resource usage is specified after Place and Route.

### VIRTEX 5

Resource type	Quantity used
Slice register	31
Slice LUT	96
Block RAM	0
DSP48	3
Clock frequency (worst case)	346 MHz
Clock frequency (best case)	472 MHz

### STRATIX III

Resource type	Quantity used
Register	65
ALUT	74
Block Memory bit	0
DSP block 18	6
Clock frequency (worse case)	201 MHz
Clock frequency (best case)	210 MHz

## Revision History

Revision	Change description	Date
1.0	Initial revision	28/04/08
1.1	improved accuracy and updated synthesis results	07/04/09