Multiplier v11.0

DS255 April 24, 2009 Product Specification

Introduction

The Xilinx® LogiCORE™ IP Multiplier implements high-performance, optimized multipliers. A number of resource and performance trade-off options are available to tailor the core to a particular application.

Features

- Drop-in module for Virtex®-6, Virtex-5, Virtex-4, Spartan®-6, Spartan-3/XA, Spartan-3E/XA, Spartan-3A/AN/3A DSP/XA FPGAs
- Generates fixed-point parallel multipliers and constant-coefficient multipliers for two's complement signed or unsigned data
- Supports inputs ranging from 1 to 64 bits wide and outputs ranging from 1 to 128 bits wide with any portion of the full product selectable
- Configurable latency for all multiplier variants
- Resource estimation in the Xilinx CORE Generator™ graphical user interface (GUI)
- Supports symmetric rounding to infinity for Virtex-4, Virtex-5 and Virtex-6 multipliers when using the XtremeDSP™ slice
- For use with Xilinx CORE Generator, Xilinx AccelDSP™ Synthesis Tool and Xilinx System Generator for DSP™ v11.1 or later.

Overview

The Multiplier core can be configured in either of the following architectures:

- **Parallel**: The multiplier accepts inputs on buses A and B and generates the product of these two values. Implementations using slice logic (LUTs), embedded multiplier blocks in Spartan™-3, Spartan-3E and Spartan-3A FPGAs, or the XtremeDSP™ slice in Virtex-4, Virtex-5, Virtex-6, Spartan-6 and Spartan-3A DSP FPGAs can be selected. Hybrid multipliers constructed from a combination of slice logic and multiplier blocks can also be implemented.
- **Constant-Coefficient**: The multiplier accepts data on the A input bus and multiplies it by a user-defined constant value. The multiplier can be constructed from distributed memory, block memories in conjunction with slice logic, or from embedded multipliers.

Important: Multiplier v11.0 is not backward compatible with version 8.0 (and earlier) of the Multiplier core. See "Performance and Resource Utilization" for more information.

Core Symbol and Port Definitions

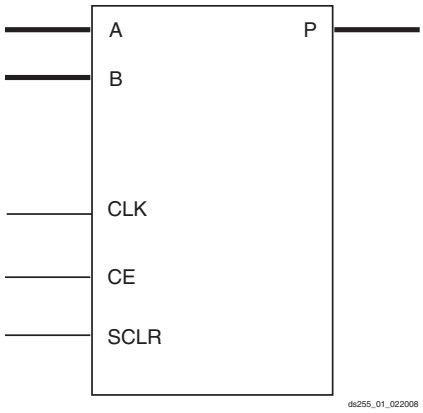

Figure 1: **Core Schematic Symbol**

Figure 1 and Table 1 illustrate and define the schematic symbol signal names. All control inputs are Active High. Should an Active Low input be required for a specific control pin, an inverter must be placed in the path to the pin and will be absorbed appropriately during synthesis and/or mapping.

Table 1: **Core Signal Pinout**

Name	Direction	Description
A[N-1:0]	Input	A operand input bus, N bits wide
B[M-1:0]	Input	B operand input bus, M bits wide (parallel multipliers only)
CLK	Input	Rising-edge clock input
CE	Input	Active high Clock Enable
SCLR	Input	Active high Synchronous Clear (SCLR/CE priority is configurable)
P[X:Y]	Output	Product Output - bit X down to bit Y

CORE Generator Graphical User Interface

The Multiplier core GUI has several pages with fields to set parameter values for the particular instantiation required. This section provides a description of each GUI field.

- **Component Name**: The name of the core component to be instantiated. The name must begin with a letter and be composed of the following characters: a to z, 0 to 9, and "_".
- **Multiplier Type**: Select between parallel and constant-coefficient multiplier options.
- **Input Options**: Select the required operand widths and whether the operands represent two's complement signed or unsigned data.
- **Parallel Multiplier Options**: These options are only visible when the multiplier type chosen is Parallel Multiplier.
  - **Multiplier Construction**: Allows the choice of LUTs or dedicated multiplier primitives to be selected for the core implementation.
  - **Optimization Options**: For a multiplier using dedicated multiplier primitives, speed, or area optimization can be selected for multiplier sizes up to 47x47. Speed optimization makes full use of multiplier primitives to provide the highest performance implementation. Area optimization uses a mixture of slice logic and dedicated multiplier primitives to reduce MULT18X18/XtremeDSP slice utilization, while still providing reasonable performance. For sizes above 47x47, only optimization for speed is allowed.
- **Constant-Coefficient Multiplier Options**: These options are only visible when the multiplier type chosen is Constant-Coefficient Multiplier.
  - **Coefficient**: Enter the integer value of the coefficient within the limits of the range shown. Positive and negative coefficients are supported. The input type (signed or unsigned) for the constant (B) port is automatically configured by the GUI based on the integer constant entered. The user may select if the A port is signed or unsigned.
  - **Memory Options**: Select if the multiplier should be implemented with distributed memory, block memory, or using embedded multiplier blocks.
- **Output Product Range**: The GUI automatically configures the output product width to represent the full product, based on the widths of the input operands.
  - **Use Custom Output Width**: The number of product bits can be customized if only a portion of the full product is required for an application by setting the MSB and LSB range.
  - **Use Symmetric Rounding**: For XtremeDSP slice-based parallel multipliers, the product can be symmetrically rounded towards infinity if required. This is the same behavior as the MATLAB *round* function. The multiplier must fit on exactly one XtremeDSP slice (maximum operand widths of 25x18 for Virtex-6 and Virtex-5, 18x18 for all other devices with XtremeDSP slices), and the LSB of the product must lie within the full-range product width.
- **Pipelining and Control Signals**:

  - **Pipeline Stages**: Select the level of pipelining for the multiplier instance. The label on the right provides feedback on the optimum number of pipeline stages for maximum performance. The core assumes that all inputs are registered.
    - Pipeline Stages = 0 implies that the core is combinatorial.
    - Pipeline Stages = 1 implies that only the core output is registered.
    - Pipeline Stages > 1 will cause registers to be inserted between input and output up to the optimum pipeline stages value. Adding more registers will improve achievable clock speed while increasing latency.
    - Pipeline Stages set to a value greater than the optimum value will fully-pipeline the core and cause SRL16-based shift registers to be added at the output to implement the extra latency.
  - **Clock Enable**: Select if all registers in the design have a clock enable control.
  - **Synchronous Clear**: Select if all registers in the design have a synchronous reset control.
  - **SCLR/CE Priority:** When both SCLR and CE pins are present, the priority of SCLR and CE can be selected. The fewest resources are used, and best performance is achieved, when SCLR overrides CE.
- **Resource Estimates Tab**: Clicking the resource estimates tab below the GUI symbol displays an estimate of the FPGA resources used for a particular multiplier instance. The values update instantaneously with changes in the GUI, allowing tradeoffs in implementation to be evaluated immediately.

Using the Multiplier IP Core

The CORE Generator GUI performs error-checking on all input parameters. Resource estimation and optimum latency information are also available.

Several files are produced when a core is generated, and customized instantiation templates for Verilog and VHDL design flows are provided in the .veo and .vho files, respectively. For detailed instructions, see the CORE Generator software documentation.

Simulation Models

The core has a number of options for simulation models:

- VHDL behavioral model in the xilinxcorelib library
- VHDL unisim-based structural simulation model
- Verilog unisim-based structural simulation model

The models required may be selected in the CORE Generator project options.

Xilinx recommends that simulations utilizing unisim-based structural models are run using a resolution of 1 ps. Some Xilinx library components require a 1 ps resolution to work properly in either functional or timing simulation. The unisim-based structural simulation models may produce incorrect results if simulated with a resolution other than 1 ps. See the "Register Transfer Level (RTL) Simulation Using Xilinx Libraries" section in *Chapter 6 of the Synthesis and Simulation Design Guide* for more information. This document is part of the ISE® Software Manuals set available at www.xilinx.com/support/software_manuals.htm.

XCO Parameters

Table 2 defines valid entries for the XCO parameters. Parameters are not case sensitive. Default values are displayed in bold.

Xilinx strongly suggests that XCO parameters are not manually edited in the XCO file; instead, use the CORE Generator GUI to configure the core and perform range and parameter value checking.

Table 2: **XCO Parameters**

XCO Parameter	Valid Values
component_name	ASCII text using characters: a..z, 0..9 and '_' ; starting with a letter
PortAWidth	1 - 64 (Unsigned data) 2 - 64 (Signed data) Default value is **18**
PortAType	**Signed**, Unsigned
PortBWidth	1 - 64 (Unsigned data) 2 - 64 (Signed data) Default value is **18**
PortBType	**Signed**, Unsigned
Use_Custom_Output_Width	**false**, true (must be set to 'true' to vary OutputWidthHigh/Low from full-precision)
OutputWidthHigh	PortAWidth+PortBWidth-1 (default value is **35**)
OutputWidthLow	0 to PortAWidth+PortBWidth-1 (default value is **0**)
MultType	**Parallel_Multiplier**, Constant_Coefficient_Multiplier
Multiplier_Construction	**Use_LUTs**, Use_Mults
OptGoal	**Area**, Speed
ConstValue	Integer constant representing any binary value up to 64 bits (**129**)
CcmImp	**Distributed_Memory**, Block_Memory, Dedicated_Multiplier
PipeStages	Integer in the range 0 to 30 (default value is **1**)
UseRounding	**false**, true
RoundPoint	Integer value less than OutputWidthHigh where binary point lies (**0**)
ClockEnable	**false**, true
SyncClear	**false**, true
SclrCePriority	**SCLR_Overrides_CE**, CE_Overrides_SCLR

Migrating to Multiplier v11.0 from Earlier Versions

Updating from Multiplier v9.0 and later

The CORE Generator core update feature may be used to update an existing Multiplier XCO file to version 11.0 of the Multiplier core. The core may then be regenerated to create a new netlist. See the CORE Generator documentation for more information on this feature.

Port Changes

There are no differences in port naming conventions, polarities, priorities or widths between versions.

Latency Changes

There are no latency differences between versions. The latency used for the previous multiplier core will be re-used when regenerating the core as v11.0. However, some cases may offer reduced latency in

v11.0 compared to previous versions. To verify that the latency used is the optimal figure, the updated XCO file may be loaded into CORE Generator and the latency on page 3 of the GUI compared with the optimum latency value.

Updating from versions prior to Multiplier v9.0

It is not currently possible to automatically update versions of the Multiplier core prior to v9.0. Xilinx recommends that customers use the Multiplier v11.0 GUI to customize a new core. Note that some features and configurations are unavailable in Multiplier v11.0. Also, some port names may differ between versions.

System Generator For DSP Graphical User Interface

This section describes each tab of the System Generator for DSP GUI and details the parameters that differ from the CORE Generator GUI.

The Multiplier core may be found in the Xilinx Blockset in the Math section. The block is called "Mult".

See the System Generator for DSP Help page for the "Mult" block for more information on parameters not mentioned here.

Tab 1: Basic

The Basic tab is used to specify the data types and control pins in a similar way to pages 1 and 3 of the CORE Generator GUI.

- **Precision:** Selecting "User Defined" allows Signed or Unsigned options to be selected. Both ports must be of the same type. Otherwise, System Generator for DSP automatically sets the input width parameters based on the signal properties of the "a" and "b" input ports.
- **Optional Port:** "Provide enable port" specifies if the core will have a clock enable pin (the equivalent of selecting the CE option in the CORE Generator GUI).
- **Latency:** Specify the latency required for the multiplier. This is equivalent to the Pipeline Stages setting in the CORE Generator GUI.

Tab 2: Advanced

The Advanced tab has no equivalent parameters on the CORE Generator GUI. The option to override with doubles applies to System Generator for DSP only.

Tab 3: Implementation

The Implementation tab is used to specify the optimization options in a similar way to page 2 of the CORE Generator GUI.

- **Use embedded multipliers:** Specifies if embedded multipliers / XtremeDSP slices should be used to construct the multiplier. If this is unchecked, LUTs will be used instead.
- **Optimize for speed/area:** Specifies if the multiplier, when built with embedded multipliers or XtremeDSP slices, should be optimized for speed (using more dedicated multiplier resources) or area (using a combination of dedicated multipliers and slice resources, where appropriate).
- **Test for optimum pipelining:** Verifies if the specified latency is the optimal selection for the hardware multiplier which will be created. Latency values that pass this test imply that the core produced would be optimized for speed of operation.

Core use through AccelDSP

The Multiplier core is available through Xilinx AccelDSP Synthesis Tool, a DSP synthesis tool that transforms a MATLAB® floating-point design into a hardware module that can be implemented in a Xilinx FPGA. The Multiplier core is invoked with a project option and automatically uses the core wherever the function is needed in the design. Certain parameters can be set through global directives. See the AccelDSP Synthesis Tool User's Guide or help menu for more information.

Performance and Resource Utilization

Table 3 through Table 8 provide performance and resource usage information for a number of different multiplier configurations.

The maximum clock frequency results were obtained by double-registering input and output ports to reduce dependence on I/O placement. The inner level of registers used a separate clock signal to measure the path from the input registers to the first output register through the core.

The resource usage results do not include the above "characterization" registers and represent the true logic used by the core to implement a single multiplier. LUT counts include SRL16s or SRL32s (according to device family).

The map options used were: "`map -pr b -ol high`."

The par options used were: "`par -t 1 -ol high`."

Clock frequency does not take clock jitter into account and should be derated by an amount appropriate to the clock source jitter specification.

The maximum achievable clock frequency and the resource counts may also be affected by other tool options, additional logic in the FPGA device, using a different version of Xilinx tools, and other factors.

The Virtex-5 FPGA test cases in Table 3 used an XC5VLX50-FF1153 (-1 speed grade) device and ISE speed file version "PRODUCTION 1.64 2009-02-10, STEPPING level 0".

Table 3: **Virtex-5 FPGA Family Performance and Resource Utilization**

Multiplier Configuration	Data Type	Core Latency (Cycles)	Maximum Clock Frequency (MHz)	LUT/FF Pairs	LUT6s	FFs	XtremeDSP™ Slices
9x9 Use LUTs	Signed	4	450	123	110	109	0
10x10 Use LUTs	Signed	4	447	133	129	117	0
12x12 Use LUTs	Signed	4	446	185	169	176	0
18x18 Use LUTs	Signed	5	389	398	390	366	0
18x18 Use Mults Optimize for Speed	Signed	3	450	0	0	0	1
20x20 Use Mults Optimize for Speed	Signed	4	450	17	0	17	2
20x20 Use Mults Optimize for Area	Signed	3	424	24	24	24	1
24x24 Use Mults Optimize for Speed	Unsigned	4	450	17	0	17	2
24x24 Use Mults Optimize for Area	Unsigned	5	403	239	170	236	1
25x18 Use Mults Optimize for Speed	Signed	3	450	0	0	0	1
25x18 Use Mults Optimize for Area	Signed	3	450	0	0	0	1
35x35 Use Mults Optimize for Speed	Signed	6	448	87	35	87	4
53x53 Use Mults Optimize for Speed	Unsigned	12	450	280	229	280	10

The Virtex-4 FPGA test cases in Table 4 used an XC4VLX40-FF1148 (-10 speed grade) device and ISE speed file version "PRODUCTION 1.69 2009-02-10, STEPPING level 1."

Table 4: **Virtex-4 FPGA Family Performance and Resource Utilization**

Multiplier Configuration	Data Type	Core Latency (Cycles)	Maximum Clock Frequency (MHz)	Slices	LUT4s	FFs	XtremeDSP Slices
9x9 Use LUTs	Signed	4	390	67	109	110	0
10x10 Use LUTs	Signed	4	367	67	129	117	0
12x12 Use LUTs	Signed	4	361	93	167	179	0
18x18 Use LUTs	Signed	5	311	200	388	370	0
18x18 Use Mults Optimize for Speed	Signed	3	400	0	0	0	1
20x20 Use Mults Optimize for Speed	Signed	6	400	62	20	72	4
20x20 Use Mults Optimize for Area	Signed	4	340	36	67	67	1
24x24 Use Mults Optimize for Speed	Unsigned	6	399	51	24	65	4
24x24 Use Mults Optimize for Area	Unsigned	6	309	219	355	402	1
25x18 Use Mults Optimize for Speed	Signed	4	400	17	0	17	2
25x18 Use Mults Optimize for Area	Signed	5	339	101	133	190	1
35x35 Use Mults Optimize for Speed	Signed	6	400	61	35	87	4
53x53 Use Mults Optimize for Speed	Unsigned	18	400	279	386	437	16

The Spartan-3A DSP FPGA test cases in Table 5 used an XC3SD3400A-FG676 (-4 speed grade) device and ISE speed file version "PRODUCTION 1.33 2009-02-10."

Table 5: **Spartan-3A DSP Family Performance and Resource Utilization**

Multiplier Configuration	Data Type	Core Latency (Cycles)	Maximum Clock Frequency (MHz)	Slices	LUT4s	FFs	XtremeDSP Slices
9x9 Use LUTs	Signed	4	211	67	113	104	0
10x10 Use LUTs	Signed	4	206	67	133	111	0
12x12 Use LUTs	Signed	4	195	93	180	164	0
18x18 Use LUTs	Signed	5	170	200	398	354	0
18x18 Use Mults Optimize for Speed	Signed	3	250	0	0	0	1
20x20 Use Mults Optimize for Speed	Signed	8	250	56	72	75	4
20x20 Use Mults Optimize for Area	Signed	4	200	36	67	67	1
24x24 Use Mults Optimize for Speed	Unsigned	8	250	55	65	72	4
24x24 Use Mults Optimize for Area	Unsigned	6	182	218	359	397	1
25x18 Use Mults Optimize for Speed	Signed	5	250	21	17	25	2
25x18 Use Mults Optimize for Area	Signed	5	197	101	135	188	1
35x35 Use Mults Optimize for Speed	Signed	8	250	70	87	105	4
53x53 Use Mults Optimize for Speed	Unsigned	24	250	298	492	454	16

The Spartan-3E FPGA test cases in Table 6 used an XC3S1600E-FG484 (-4 speed grade) device and ISE speed file version "PRODUCTION 1.27 2009-02-10."

Table 6: **Spartan-3E Family Performance and Resource Utilization**

Multiplier Configuration	Data Type	Core Latency (Cycles)	Maximum Clock Frequency (MHz)	Slices	LUT4s	FFs	MULT18X18SIOs
9x9 Use LUTs	Signed	4	231	67	110	108	0
10x10 Use LUTs	Signed	4	218	67	130	115	0
12x12 Use LUTs	Signed	4	209	93	169	176	0
18x18 Use LUTs	Signed	5	191	200	398	354	0
18x18 Use Mults Optimize for Speed	Signed	3	238	36	0	36	1
20x20 Use Mults Optimize for Speed	Signed	5	230	76	85	130	4
20x20 Use Mults Optimize for Area	Signed	4	197	93	101	181	1
24x24 Use Mults Optimize for Speed	Unsigned	5	209	95	104	169	4
24x24 Use Mults Optimize for Area	Unsigned	5	181	251	352	465	1
25x18 Use Mults Optimize for Speed	Signed	4	216	52	43	87	2
25x18 Use Mults Optimize for Area	Signed	4	193	118	164	223	1
35x35 Use Mults Optimize for Speed	Signed	5	163	143	158	265	4
53x53 Use Mults Optimize for Speed	Unsigned	7	128	595	801	1045	16

The Spartan-3 FPGA test cases in Table 7 used an XC3S1500-FG676 (-4 speed grade) device and ISE speed file version "PRODUCTION 1.39 2009-02-10."

Table 7: **Spartan-3 Family Performance and Resource Utilization**

Multiplier Configuration	Data Type	Core Latency (Cycles)	Maximum Clock Frequency (MHz)	Slices	LUT4s	FFs	MULT18X18s
9x9 Use LUTs	Signed	4	206	67	110	108	0
10x10 Use LUTs	Signed	4	191	67	130	115	0
12x12 Use LUTs	Signed	4	200	93	169	176	0
18x18 Use LUTs	Signed	5	181	200	394	358	0
18x18 Use Mults Optimize for Speed	Signed	2	246	36	0	36	1
20x20 Use Mults Optimize for Speed	Signed	4	157	76	85	130	4
20x20 Use Mults Optimize for Area	Signed	3	180	72	101	139	1
24x24 Use Mults Optimize for Speed	Unsigned	4	180	95	104	169	4
24x24 Use Mults Optimize for Area	Unsigned	5	158	251	386	465	1
25x18 Use Mults Optimize for Speed	Signed	3	200	52	43	87	2
25x18 Use Mults Optimize for Area	Signed	4	172	136	164	259	1
35x35 Use Mults Optimize for Speed	Signed	4	142	143	158	265	4
53x53 Use Mults Optimize for Speed	Unsigned	6	110	595	801	1045	16

The Spartan-3A FPGA test cases in Table 8 used an XC3S1400A -FG676 (-4 speed grade) device and ISE speed file version "PRODUCTION 1.41 2009-02-10."

Table 8: **Spartan-3A Family Performance and Resource Utilization**

Multiplier Configuration	Data Type	Core Latency (Cycles)	Maximum Clock Frequency (MHz)	Slices	LUT4s	FFs	MULT18X18SIO
9x9 Use LUTs	Signed	4	218	67	113	104	0
10x10 Use LUTs	Signed	4	211	67	133	111	0
12x12 Use LUTs	Signed	4	203	93	180	164	0
18x18 Use LUTs	Signed	5	169	200	398	354	0
18x18 Use Mults Optimize for Speed	Signed	2	251	36	0	36	1
20x20 Use Mults Optimize for Speed	Signed	4	226	76	85	130	4
20x20 Use Mults Optimize for Area	Signed	3	184	93	101	181	1
24x24 Use Mults Optimize for Speed	Unsigned	4	189	95	104	169	4
24x24 Use Mults Optimize for Area	Unsigned	5	176	251	352	465	1
25x18 Use Mults Optimize for Speed	Signed	3	209	52	43	87	2
25x18 Use Mults Optimize for Area	Signed	4	184	118	164	223	1
35x35 Use Mults Optimize for Speed	Signed	4	151	143	158	265	4
53x53 Use Mults Optimize for Speed	Unsigned	6	119	595	801	1045	16

Ordering Information

The Multiplier core can be downloaded from the Xilinx IP Center for use with Xilinx CORE Generator 11.1 and higher. The CORE Generator software is bundled with the ISE Foundation™ 11.1 software at no extra charge.

To order Xilinx software, contact your local Xilinx sales representative.

Information on additional Xilinx LogiCORE modules is available on the Xilinx IP Center.

Revision History

The following table shows the revision history for this document:

Date	Version	Revision
30/28/03	1.0	Revision History added to document.
02/27/04	1.1	Updated copyright, release date, UI screen shots and template.
05/21/04	1.2	Updated with Virtex-4 and CORE Generator system v6.2i.
04/28/05	1.3	Updated document to indicate support for Xilinx software v7.1i and Spartan-3E device.
01/18/06	2.0	Completely revised layout, removed UI screen shots, added VHDL generic information and direct instantiation usage details.
07/13/06	3.0	Updated with new generic information and performance data. Removed table of XCO parameters. Removed timing diagrams since handshaking control signals were deprecated.
02/15/07	3.1	Updated for ISE 9.1i support, Spartan-3A/3AN device, and updated the performance data. Added section on GUI parameters.
04/02/07	3.5	Added support for Spartan-3A DSP devices.
04/25/08	3.6	Updated for ISE v10.1 support and updated the performance data table. Removed section on direct instantiation.
04/24/09	4.0	Update for ISE 11.1 support.

Notice of Disclaimer