Variable Parallel Virtex Multiplier V1.0.2

October 15, 1999 | Product Specification

Xilinx Inc.
2100 Logic Drive
San Jose, CA 95124
Phone: +1 408-559-7778
Fax: +1 408-559-7114
E-mail: coregen@xilinx.com
URL: www.xilinx.com/ipcenter

Features

- Drop-in module for Virtex, Virtex-E and Spartan2 families
- Multiplies multiplicand (A) by multiplier (B) to produce a product (P)
- Multiplicand value can range from 2 to 32 bits
- Multiplier value can range from 2 to 32 bits
- Product value can range from 4 to 64 bits
- Supports unsigned and 2's complement signed numbers
- Optional combinatorial architecture
- Optional pipelined architecture for increased throughput
- Optional registered outputs
- Optional: Clock Enable, Asynchronous Clear, Asynchronous Set, Synchronous Clear and Synchronous Set
- High performance and density using Xilinx Relational Placed Macro (RPM) mapping and placement technology
- To be used with Xilinx Core Generator V2.1i System and later versions

Functional Description

This parameterized module multiplies an M-bit wide multiplicand variable by an N-bit wide multiplier variable to produce an M+N bit result.

An area-efficient, high-speed algorithm is used to give an efficient, tightly packed design. Each stage can also be pipelined for maximum performance.

Output registers can be selected to maintain synchronicity between modules.

A clock-enable can halt any pipelined stage or output stage. Partially completed results are stored internally when CE is taken low, and the multiplier resumes operation when CE returns high.

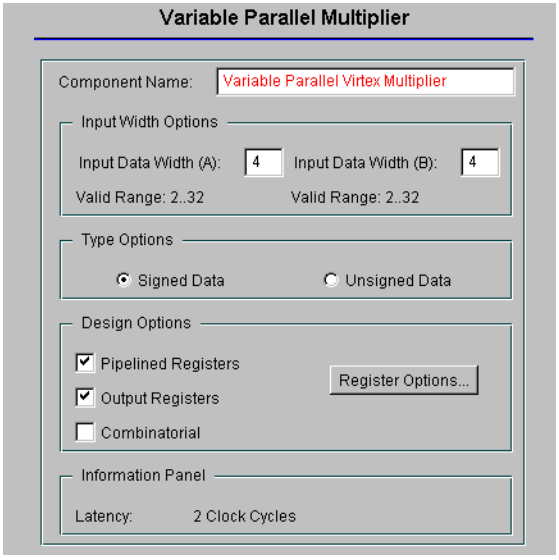

Figure 1: Variable Parallel Multiplier Parameterization Screen

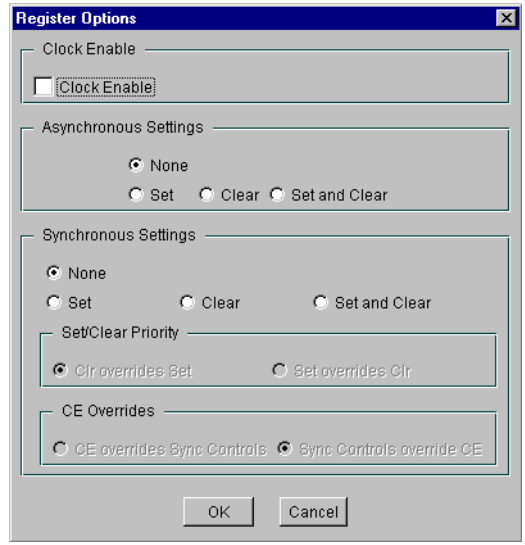

Figure 2: Register Options Parameterization Screen

Synchronous or asynchronous clears and resets can be selected to conform with the user's methodology.

Pinout

Signal names are shown in Figure 3 and described in Table 1.

Table 1: Core Signal Pinout

Signal	Signal Direction	Description
A(N-1:0)	Input	Input data (multiplicand)
B(M-1:0)	Input	Input data (multiplier)
ce	Input	Active High clock enable
aclr	Input	Active High asynchronous clear
aset	Input	Active High asynchronous set
sclr	Input	Active High synchronous clear
sset	Input	Active High synchronous set
P((N+M)-1:0)	Output	Output data (product)

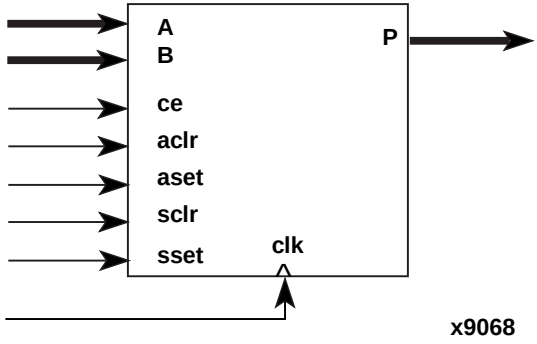

Figure 3: Core Schematic Symbol

CORE Generator Parameters

The main CORE Generator parameterization screen for this module is shown in Figure 1.

The GUI parameters are described below. The corresponding XCO file parameter names are listed in Table 5.

- **Component Name**: Enter a name for the component. The component name is used as the base name for the output files generated for this module. Names must begin with a letter and must be composed from the following characters: a to z, 0 to 9 and "_".
- **Input Data Width (A)**: Type the multiplicand variable width into the text box provided. The valid range is 2 to 32. The default value is 4.
- **Input Data Width (B)**: Type the multiplier variable width into the text box provided. The valid range is 2 to 32. The default value is 4.
- **Signed Data**: Set the sign of the input and output data to be signed. The default setting is checked.
- **Unsigned Data**: Set the sign of the input and output data to be unsigned. The default setting is unchecked.
- **Pipelined Registers**: Switch pipelining ON or OFF. The default setting is checked.
- **Output Registers**: Switch output registers ON or OFF. The default setting is checked.
- **Combinatorial**: Switch all registers OFF. The default setting is unchecked.

The Register Options parameterization screen for this module is shown in Figure 2. The parameters are as follows:

- **Clock Enable**: When this box is checked the module is generated with a clock enable input. The default setting is unchecked.
- **CE Overrides**: This parameter controls whether or not the SSET and SCLR inputs are qualified by CE or not. This parameter is only enabled when a Clock Enable input has been requested along with SSET and/or SCLR. When **CE Overrides Sync Controls** is selected, an Active High level on any of the synchronous control inputs will only be acted upon when the CE pin is active. This is not the way the dedicated inputs on the flip-flop primitive work, and so setting the **CE Overrides** parameter to **CE Overrides Sync Controls** will force the synchronous control functionality to be implemented using logic in the Look Up Table (LUT) preceding the flip-flop. This results in increased resource utilization even when asynchronous controls are not present. When **Sync Controls Override CE** is selected, an active level on any of the synchronous control inputs will be acted upon, irrespective of the state of the CE pin. This setting is more efficient when asynchronous inputs are not present because it allows the dedicated inputs on the flip-flop primitive to be used for the synchronous control functions. It is less efficient when the presence of asynchronous inputs force the synchronous control functionality to be implemented using logic in the LUT preceding the flip-flop. This is because the CE signal has to be gated with the synchronous control inputs to generate a new, qualified CE signal to the multiplier flip-flop. This slows down the CE path and results in slower overall operation of the module. The default setting is **Sync Controls Override CE** so that the more efficient implementation can be generated.
- **Asynchronous Settings:** All asynchronous controls are implemented using the dedicated inputs on the flip-flop primitive. The module can be generated using the following asynchronous control inputs by clicking on the appropriate button:
  - **None**: No asynchronous control inputs. This is the default setting.

- **Set**: An ASET input pin is generated.
- **Clear**: An ACLR input pin is generated.
- **Set and Clear:** Both ASET and ACLR input pins are generated. ACLR has priority over ASET when both are asserted at the same time.

- **Synchronous Settings**: When no asynchronous controls are implemented (i.e. when **Asynchronous Settings** is set to **NONE**) the synchronous controls can be implemented using the dedicated inputs on the flip-flop primitive. There are exceptions to this, as described under the **Set/Clear Priority** and **CE Overrides** parameters.

  When asynchronous controls are present synchronous control functionality must be implemented using logic in the LUT preceding the output flip-flop.

  The module can be generated with the following synchronous control inputs by clicking on the appropriate button:

  - **None**: No synchronous control inputs. This is the default setting.
  - **Set**: An SSET input pin is generated.
  - **Clear:** An SCLR input pin is generated.
  - **Set and Clear**: Both SSET and SCLR input pins are generated. SCLR/SSET priority is defined by the setting of the **Set/Clear Priority** parameter.
- **Set/Clear Priority**: By selecting the appropriate radio button, the priority of synchronous clear with respect to synchronous set can be controlled. This parameter is only enabled when both synchronous set and synchronous clear have been requested. It is not possible for Set to Override Clear when the synchronous control functionality is implemented using dedicated inputs on the flip-flop primitive. The required logic can only be implemented using the LUT preceding the flip-flop, resulting in increased resource utilization even when asynchronous controls are not present. The default setting is **Clear Overrides Set** so that the more efficient implementation is usually generated.

Note: If selected the asynchronous control inputs are active at all times. Any assertion results in an immediate change, regardless of the state of the clock, clock enable, data or any synchronous control signal.

Parameter Values in the XCO File

Table 5 shows the XCO file parameters and values and summarizes the GUI defaults. **Note:** The text in an XCO file is case insensitive.

The following is an example of an XCO file:

```
CSET pipelined_registers = registered
CSET synchronous_settings = none
CSET component_name=mult_vgen_v1_0
CSET a_width = 4
CSET ce_overrides = sync_controls_override_ce
CSET signed_type = signed
CSET set_clear_priority = clear_overrides_set
CSET clock_enable = true
CSET b_width = 4
CSET output_registers = registered
CSET asynchronous_settings = none
```

Latency

The total latency (number of clock cycles required to get the first output) is a function of the width of the 'B' input and the pipelining and output register selections, see Table 2.

Table 2: Latency

B data width (# bits)	Latency (# Clocks)	
	Pipelined	Pipelined + output reg.
2	N/A	1
3 and 4	1	2
5 to 8	2	3
9 to 16	3	4
17 to 32	4	5

Performance & Size Characteristics

It is important to set a maximum period constraint on the core's clock input. Table 3 and Table 4 show speeds that can be achieved when this is done.

Information onVirtex slice count and RPM dimensions is listed in Table 3 for several Virtex multipliers.

Table 3: Virtex Fully Registered Multiplier

A Width	B Width	Slice Count	CLBs (Rows, Columns)	XCV300-6
8	8	40	R8xC4	167MHz
16	16	152	R16xC8	143MHz
32	32	576	R32xC16	111MHz

Table 4: Virtex Combinatorial Multiplier

A Width	B Width	Slice Count	CLBs (Rows, Columns)	XCV300-6
8	8	36	R6xC4	83MHz
16	16	140	R12xC8	59MHz
32	32	544	R24xC16	43MHz

Core Resource Utilization

Select ‘A’ (multiplicand) and ‘B’ (multiplier) inputs carefully. To reduce the number of stages and therefore the size and latency of the multiplier, connect the smaller bit size input to ‘B’. However, each stage will contain a larger carry chain. Therefore, to get maximum speed performance between stages, connect the larger input size to ‘B’. Generally keep the ‘B’ input as small as possible.

The equations shown below can be used to approximate the size of the multiplier i.e. number of virtex slices required. The calculation is a worst case count. Values for ‘B’ that are not equal to a power of two or combinatorial multipliers will produce a smaller slice count than the equations predict. Designs requiring both synchronous and asynchronous control signal will produce a large slice count than the equations predict.

Step 1: Calculate total number of stages required(i.e., the FinalStageNo.) Round the ‘B’ input to the nearest power of two.

$$B(bitsize)=2^{FinalStageNo}$$

Step 2: Calculate the total number of Virtex slices for each stage (i.e., for each StageX).

$$StageX = \left[\frac{A(bitsize)+2^{StageNo}}{2}\right] x \left[\frac{2^{FinalStageNo}}{2}\right]$$

Step 3: Sum all stage values (stage(1) + stage(2)... +stage(finalstage)) to obtain the total slice count.

Ordering Information

This core is downloadable free of charge from the Xilinx IP Center (www.xilinx.com/ipcenter), for use with the Xilinx Core Generator System version 2.1i and later. The Core Generator System 2.1i tool is bundled with the Alliance 2.1i and Foundation 2.1i implementation tools.

To order Xilinx software contact your local Xilinx sales representative at www.xilinx.com/company/sales.htm.

Table 5: Default Values and XCO File Values

Parameters	XCO File Values	Default GUI Settings
component_name	ASCII text starting with the letter and based upon the following character set: a…z, 0…9 and "_"	blank
a_width	Integer in the range of 2 to 32	4
b_width	Integer in the range of 2 to 32	4
pipelined_registers	One of the following key words: registered, non-registered	registered
output_registers	One of the following key words: registered, non-registered	registered
signed_type	One of the following key words: signed, unsigned	signed
clock_enable	One of the following key words: true, false	false
asynchronous_settings	One of the following key words: none, set, clear, set_and_clear	none
synchronous_settings	One of the following key words: none, set, clear, set_and_clear	none
set_clear_priority	One of the following key words: clear_overrides_set, set_overrides_clear	clear_overrides_set
ce_overrides	One of the following key words: sync_controls_override_ce, ce_overrides_sync_controls	sync_controls_override_ce