Multiply Generator V2.0

November 3, 2000 Product Specification

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

Features

- Drop-in module for Virtex, Virtex™-II, Virtex™-E and Spartan™–II FPGAs
- Generates parallel multiplier, sequential and serial-sequential multiplier, fixed and reloadable constant co-efficient multiplier.
- Supports 2's complement signed and unsigned operations
- Supports inputs ranging from 1 to 64 bits wide
- Supports outputs ranging from 1 to 129 bits wide
- Generates purely combinatorial and fully pipelined implementations.
- Optional registered output with optional clock enable and asynchronous and synchronous clears.
- Optional handshaking signals.
- Incorporates Xilinx Smart-IP technology for maximum

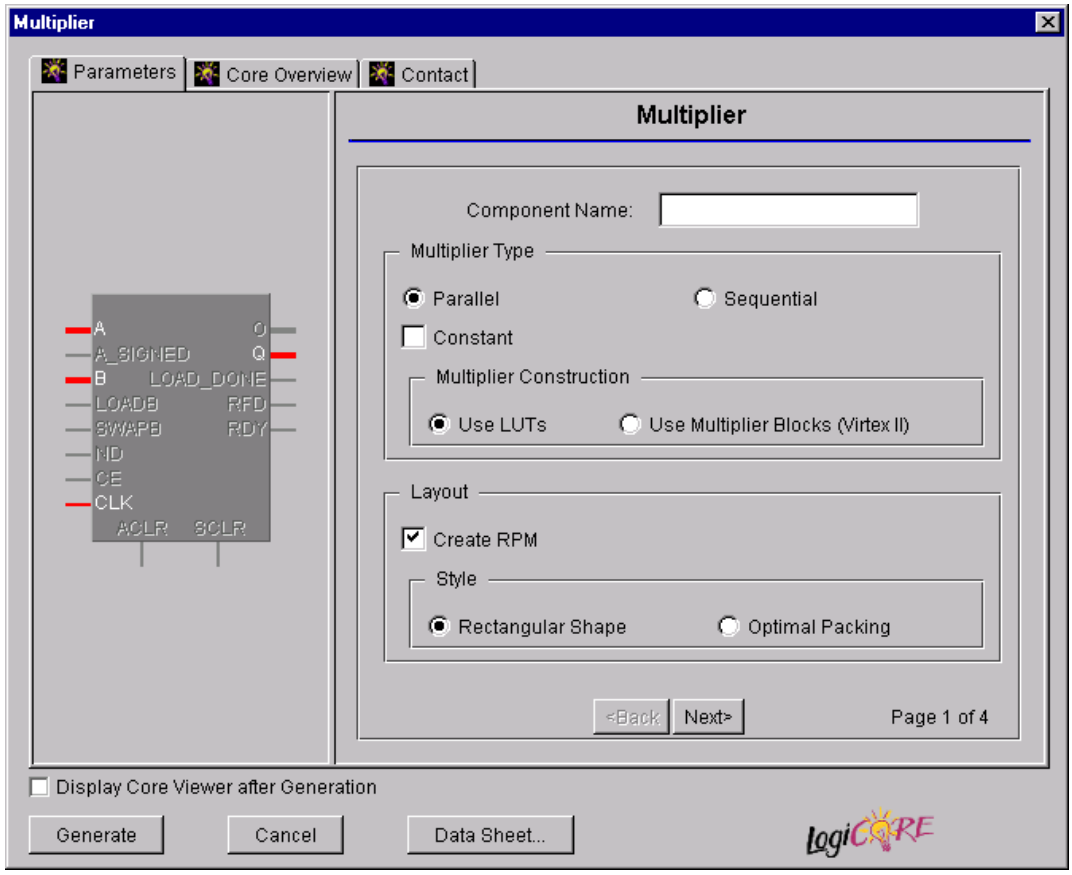

Figure 1: Main GUI Customization Window.

performance
- To be used with version 3.1i and later of the Xilinx CORE Generator System

Functional Description

The multiplier generator core can generate three different types of multiplier:

- Parallel Multiplier. This takes 2 input busses, A and B, and generates the multiplication of the values on these busses. The multiplier is generated from look-up tables or from the discrete multiplier blocks on the Virtex™-II chip.
- Constant Coefficient multiplier. This takes the A input bus and multiplies it by a user-supplied constant value. The constant can be fixed or reloadable. In the reloadable case, the value on the B input bus when a load signal is asserted is used as the constant.
- Sequential multiplier. In the sequential multiplier, the multiplication is broken down into a series of smaller partial multiplications. The final result is the summation of these partial products. This leads to a smaller multiplier and is particularly useful in designs where size is crucial. In the serial-sequential multiplier, the data on the A bus is input serially.

Timing diagrams for all three types of multiplier can be found in Figures 8-12.

Pinout

Signal names for the schematic symbol are shown in Figure 2 and described in Table 1.

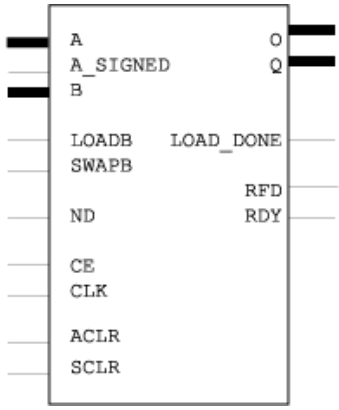

Figure 2: Core Schematic Symbol.

CORE Generator Parameters

The main CORE Generator parameterization screen for this module is shown in Figure 1. The parameters are as follows:

Table 1: Core Signal Pinout

Signal	Signal Direction	Description
A[N:0]	Input	A Input bus. N = port_a_width-1
A_SIGNED	Input	A Input sign control
B[M:0]	Input	B Input bus. M = port_b_width-1
CLK	Input	Clock - rising edge clock signal
CE	Input	Clock enable
ND	Input	New data signal. The user can assert this to indicate that there is new data present on the input to the multiplier.
ACLR	Input	Asynchronous Clear - forces the output of the registers in the circuit to a low state when driven
SCLR	Input	Synchronous Clear - forces register outputs to a low state on next concurrent clock edge
LOADB	Input	When LOADB is asserted the value on the B input is written into the memory as the new constant value. The number of clock cycles taken to perform this operation is dependent on the size of the B input.
SWAPB	Input	The reloadable constant coefficient multiplier can be generated with two banks of memory each storing a different constant. The user can assert SWAPB to switch between the two banks.
RDY	Output	Data ready signal. When this signal goes high the result of the current multiplication is present on the output.
RFD	Output	Ready for data signal. When this signal is high the multiplier is ready to accept data.
O[P:0]	Output	Asynchronous output. P = output_width-1.
Q[P:0]	Output	Synchronous output. P = output_width-1.
LOAD_DONE	Output	A high on this output indicates that the load process has been completed.

Note:
All control inputs are Active High. Should an Active Low input be required for a particular control pin an inverter must be placed in the path to the pin. The inverter will be absorbed appropriately during mapping.

- **Component Name**: The component name is used as the base name of 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 "_".
- **Multiplier Type**: Select the appropriate radio button for the type of multiplier required. If **Parallel** is selected, the multiplication will be calculated in parallel. Here, the circuitry to calculate all the partial products is present in the design. If **Sequential** is selected, a smaller circuit is generated by generating the circuit for only one partial product. The A input is fed into this circuit in stages and the results for each stage are summed together. The full product appears at the output after n cycles, where n is equal to **CLK_Cycles_per_input**. This parameter is set later. The default setting is **Parallel**. If a constant coefficient multiplier is required, the **Constant** check box is checked.
  - **Multiplier Construction**. If the design is to be implemented on a Virtex™-II chip, the user can select between using Look-Up Tables (LUTs) and the dedicated multiplier blocks.
- **Layout Options**
  - **Create RPM**. When this box is checked, the module will be generated with relative location attributes attached. The default operation is to create an RPM. Note that when a module is created as an RPM it is possible that one or more of the module dimensions may exceed those of the device being targeted. If this is the case, mapping errors will occur and the compilation process will fail. In this case, the module should be regenerated with the **Create RPM** checkbox unchecked.
  - **Shape:** If the Create RPM check box is selected, the user has two shape options. If **Rectangular** is selected, the design will be roughly rectangular. In this configuration, the LUTs are spaced quite loosely. If **Optimal** is selected, the design will be more densely packed. This option is not valid for a sequential type multiplier where the logic is always densely packed.

When a constant coefficient multiplier has been selected, the parameterization screen shown in Figure 3 is displayed when the Next button is clicked.

- **Constant B Options**:
  - **Constant Value**: When the **Constant** check box is checked, Port B is set to the value that is typed into this text box. The Constant Value must be entered in positive or negative decimal format and the binary equivalent value must not exceed the specified **Port B Width**. In most cases, specifying Port B to be a constant will create a module without Port B. The only exception to this is when reloading functionality is requested, as Port B is needed to provide the new

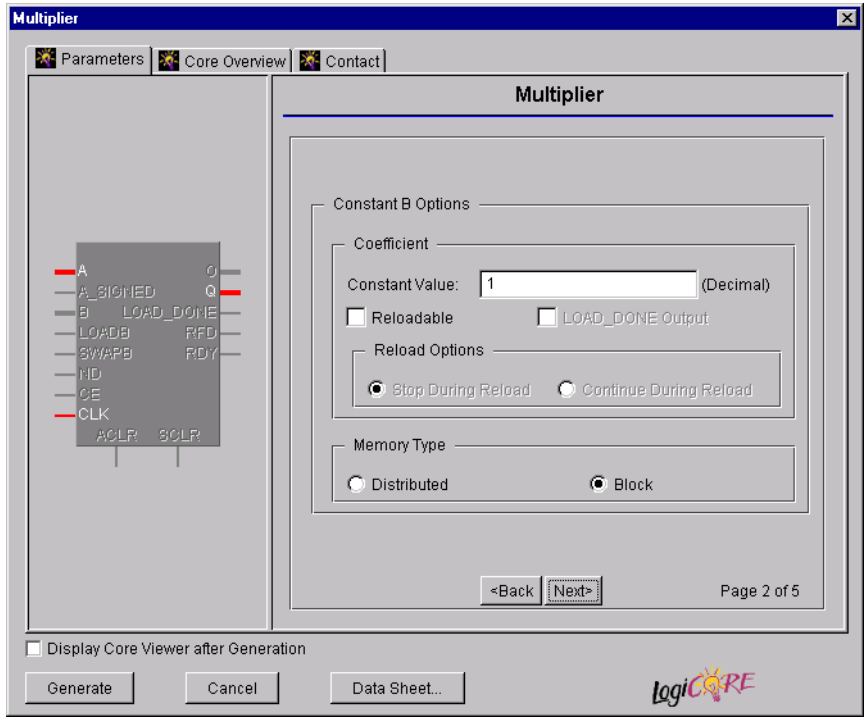

Figure 3: Constant Coefficient Parameterization Window.

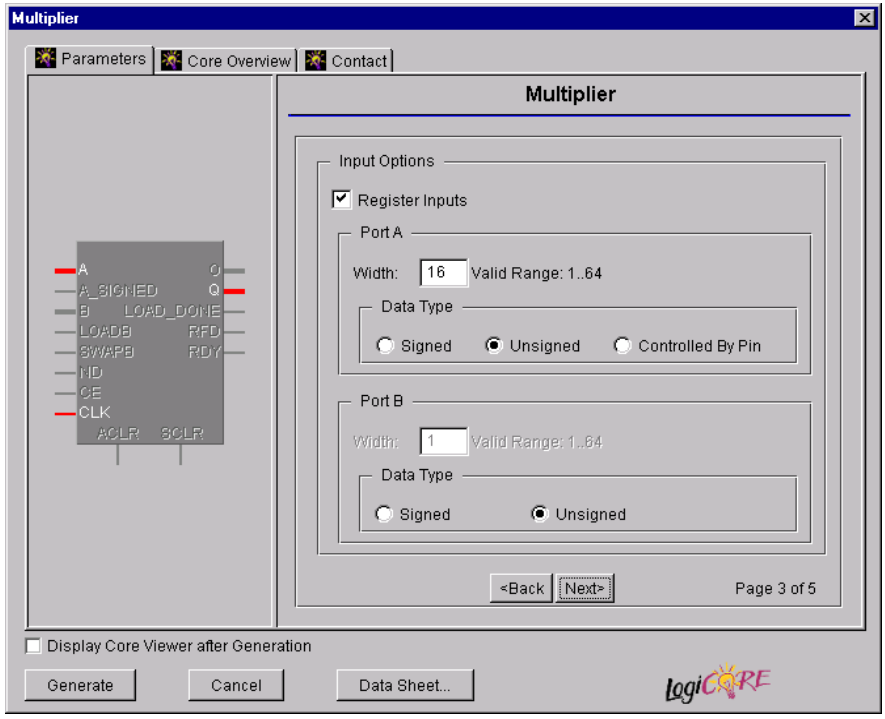

Figure 4: Input Options Parameterization Screen.

constant data in such cases. The default setting is for the Port B value to be provided via Port B.

- **Reloadable**: If this check box is checked, the multiplier will be instantiated with a LOADB pin. When this pin is driven high, the value on the B input port will be loaded into the module and used as the new constant value.
- **LOAD_DONE Output**: If this is selected, the module will generate a LOAD_DONE output. This indicates that the loading procedure is complete.
- **Reload Options**: If the **Stop During Reload** check box is selected, the multiplier will be generated using one bank of memory. The value on the output during the reloading process will be incorrect. If **Continue During Reload** is selected, the module will be generated with two banks of memory. During the reload process, the new constant will be loaded into the memory that is not being used. This results in correct data at the output during the reload cycle. The module is instantiated with a SWAPB pin which allows the user to toggle between the two banks of memory.
- **Memory Type**: The module can be generated using **Distributed** or **Block** memories. Distributed memory utilizes the LUTs configured as memory modules. The block memory uses the predefined memory block on the Xilinx chip.

If a non constant coefficient multiplier is selected, the screen shown in Figure 3 is skipped. The screen shown in Figure 4 is displayed next.

- **Input Options**:
  - **Register Inputs**. If the Register Inputs check box is checked, the A, B, A_SIGNED and ND inputs are registered. The registers are clocked on the rising edge of the CLK signal.
  - **Port A Width**: Enter the width of the Port A input. The valid range is 1 to 64. The default value is 16.
  - **Port A Data Type**: Enter the sign of the Port A input. The data can be interpreted as one of three types. These are **Signed**, **Unsigned,** and **Pin**. When **Pin** is selected, the sign of the input is determined by the value on the A_SIGNED pin. If this pin is driven high, the data is treated as a signed number; if it is driven low, the data is unsigned. The default value is **Unsigned**.
  - **Port B Width**: Enter the width of the Port B input. The valid range is 1 to 64. The default value is 16.
  - **Port B Data Type**: Enter the sign of the Port B data. This can either be **Signed** or **Unsigned**. The default value is **Unsigned**.

If the user has selected a sequential multiplier, the next screen is the one shown in Figure 5. If a parallel multiplier has been selected, this screen is skipped and the next screen will look like the one in Figure 6.

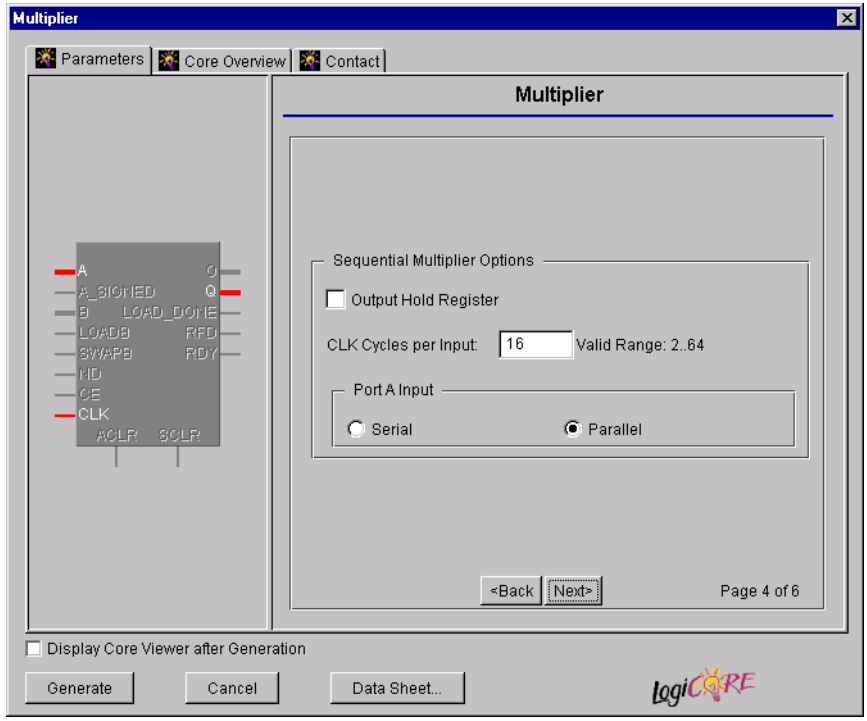

Figure 5: Sequential Multiplier Options

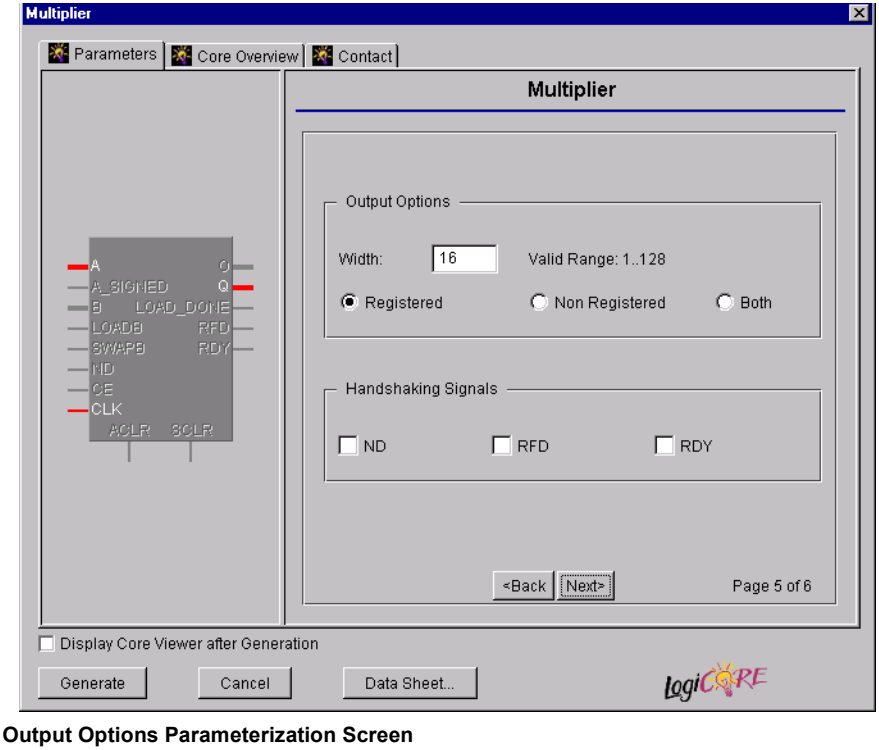

Figure 6: Output Options Parameterization Screen

Sequential Multiplier Options:

- **Output Hold Register.** If this box is checked, the Q output will be held constant between calculations. If it is unchecked, the Q output will reflect the O output as it accumulates the partial products. Note that this is not valid for non-registered ouputs
- **CLK Cycles per input**. The number of cycles that each multiplication should take. Note that this does not indicate the latency of the core. It simply is a guide that the A input will be split up into equal parts (rounded up from the division A_width/ CLK_cycles_per_input) for separate multiplication. There are cases when this multiplication can be carried out in less than the cycles entered. For example, if CLK_cycles_per_input = 4 and A_width = 5, then the multiplication process will be carried out with 2 bits of the A input at a time. This means that the multiplication will actually be carried out in 3 clock cycles rather than the 4 specified.
- **Port A Input**. The A input can be parallel or serial. In the parallel case, the width of the A bus is specified by the **Port A Width** value. Internally, the module splits the A input into slices. The number of slices is dependent on the **CLK Cycles per input** setting. When a serial input is requested, the module is instantiated with a 1-bit-width A input. In this case, the multiplication is performed 1 bit at a time. The **CLK Cycles per input** value is set to **Port A Width** in this case.

- **Output Options**
  - **Width**: For signed and unsigned inputs, the output width can be any number of bits from 1 to 128. If the A inputs sign is determined by a pin, an output width of 129 can be selected.
  - **Registered**: If this check box is checked, the module will be generated with registers on all the outputs.
  - **Non-Registered**: If this check box is checked, the module will be generated with an asychronous output only.
  - **Both**: If this check box is checked, an asynchronous and a synchronous output will be generated. If a RDY signal is requested, it is driven high when the correct product is ready at the registered output.
- **Handshaking Signals**
  - **ND**: New Data. If ND is checked, the multiplier will wait for ND to go high before processing the inputs. When there are no registers in the design, this signal is ignored.
  - **RFD**: Ready for Data. The RFD signal is produced by the module when it can accept new data. In the case of the sequential multiplier, this signal is driven low for the number of cycles it takes to perform the calculation. It is then driven high when it can accept a new input value.
  - **RDY**: Ready. This signal is driven high when the correct product is present at the output; *i.e.,* at the end of a sequential calculation or N cycles after a valid new data is accepted, where N is the latency of the module. If an ND pin is not present and a parallel multiplier is selected, this pin will be tied high.

The final parameterization screen is shown in Figure 7.

- **Pipeline Options**:
  - **Maximum Pipelining**: If this check box is checked, the module will be generated with registers between each stage of the multiplier. This leads to an increase in circuit speed at the cost of space.
  - **Minimum Pipelining**: If minimum pipelining is selected, the multiplier will be generated with the minimum amount of registers in the data path. Output and Input registers can still be requested.
- **Register Options**:This panel is enabled only when a registered output has been requested via the **Output Options**
  - **Asynchronous Clear**. An ACLR pin is generated. When driven high, this will reset all the registers in the design asynchronously.
  - **Synchronous Clear**. An SCLR pin is generated. This will reset all the registers in the design on the rising edge of the CLK signal.
  - **CE Overrides**: This parameter controls whether or not the SCLR input is qualified by CE. This parameter is only enabled when a **Clock Enable** input has been requested. When **CE Overrides SCLR** is selected, an active level on SCLR will be acted upon only when the CE pin is Active. Note that this is not the way that the dedicated inputs on the flip-flop primitives work, and so setting the **CE Overrides** parameter to **CE Overrides SCLR** will force the SCLR functionality to be implemented using logic in the look-up tables (LUTs) preceding the output register. This results in increased resource utilization even when an asynchronous clear is not present. When **SCLR Overrides CE** is selected, an active level on SCLR will be acted upon irrespective of the state of the CE pin. This setting is more efficient when ACLR is not present because it allows the dedicated inputs on the flip-flop primitives to be used for the synchronous clear. It is less efficient when ACLR forces the synchronous control functionality to be implemented using logic in the LUTs preceding the output register. This is because the CE signal has to be gated with SCLR so that it can generate a CE signal to the flip-flops, slowing down the CE path and resulting in slower overall operation of the module. The default setting is **SCLR Overrides CE** so that a more efficient implementation can be generated.

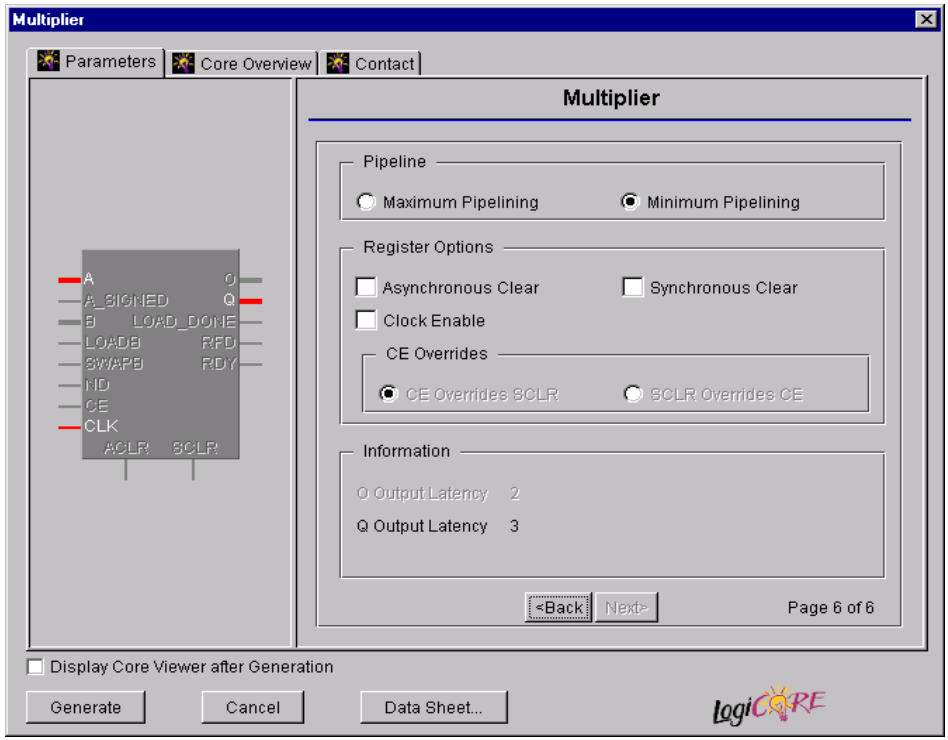

Figure 7: Final Parameterization Screen

Parameter Values in the XCO File

Names of XCO file parameters and their parameter values are identical to the names and values shown in the GUI,

except that underscore characters (_) are used instead of spaces. The text in an XCO file is case insensitive.

Table 2 shows the XCO file parameters and values, and summarizes the GUI defaults. The following is an example of the CSET parameters in an XCO file:

```
CSET component_name = abc123
CSET port_a_width = 16
CSET port_a_data = Unsigned
CSET port_a_input = Parallel
CSET port_b_width = 16
CSET port_b_data = Unsigned
CSET port_b_constant = false
CSET port_b_constant_value = 1
CSET rdy = false
CSET create_rpm = true
CSET output_width = 32
CSET reload_options = Stop_During_Reload
CSET asynchronous_clear = false
CSET synchronous_clear = false
CSET output_options = Registered
CSET output_hold_register = false
CSET ce_overrides = CE_overrides_SCLR
CSET rfd = false
CSET register_input = true
CSET reloadable = false
CSET clk_cycles_per_input = 1
CSET nd = false
CSET clock_enable = false
CSET multiplier_type = Parallel
CSET load_done_output = false
CSET multiplier_construction = Use_LUTs
CSET style = Rectangular_Shape
CSET memory_type = Distributed_memory
CSET pipelined = Minimum
```

Core Resource Utilization

For an accurate measure of the usage of primitives, slices, and CLBs for a particular point solution, check the **Display Core Viewer after Generation** checkbox, in CoreGen.

Ordering Information

This core is downloadable free of charge from the Xilinx IP Center (www.xilinx.com/ipcenter), for use with version 3.1i and later versions of the Xilinx Core Generator System. The Core Generator System is bundled with the Alliance and Foundation implementation tools.

To order Xilinx software, contact your local Xilinx sales representative. For information on the Xilinx sales office nearest you, please refer to http://www.xilinx.com/company/sales.htm.

Table 2: Default Values and XCO File Values

Parameter	XCO File values	Default GUI Setting
component_name	ASCII text starting with a letter and based upon the following character set: a..z, 0..9 and _	blank
port_a_width	Integer in the range 1 to 64	16
port_a_data	One of the following keywords: Unsigned, Signed, Controlled_by_Pin	Unsigned
port_a_input	One of the following keywords: Parallel, Serial	Parallel
port_b_width	Integer in the range 1 to 64	16
port_b_data	One of the following keywords: Unsigned, Signed	Unsigned
port_b_constant	One of the following keywords: True, False	false
port_b_constant_value	Decimal value whose value does not exceed $2^{port\_b\_width}$ - 1 for an unsigned b input or is between $-(2^{port\_b\_width-1})$ and $2^{port\_b\_width-1}$ - 1 when b is signed.	1
output_options	One of the following keywords: Non_Registered, Registered, Both	Registered
output_width	Integer in the range 1 to 129	32
register_input	One of the following keywords: true, false	true
output_hold_register	One of the following keywords: true, false	false
multiplier_type	One of the following keywords: Parallel, Sequential	Parallel
rfd	One of the following keywords: true, false	false
rdy	One of the following keywords: true, false	false
nd	One of the following keywords: true, false	false
create_rpm	One of the following keywords: true, false	true
clock_enable	One of the following keywords: true, false	false
ce_overrides	One of the following keywords: CE_Overrides_SCLR, SCLR_Overrides_CE	CE_Overrides_SCLR
asynchronous_clear	One of the following keywords: true, false	false
synchronous_clear	One of the following keywords: true, false	false
clk_cycles_per_input	Integer in the range 1 to Port_A_Width	1
reloadable	One of the following keywords: true, false	false
reload_options	One of the following keywords: Stop_During_Reload, Continue_During_Reload	Stop_During_Reload
multiplier_type	One of the following keywords: Parallel, Sequential	Parallel
pipelined	One of the following keywords: Minimum, Maximum	Minimum
multiplier_construction	One of the following keywords: Use_LUTs, Use_Multiplier_Blocks_Virtex2	Use_LUTs
style	One of the following keywords: Rectangular_shape, Optimal_packing	Rectangular_shape
memory_type	One of the following keywords: Distrubuted_memory, Dual_port_block_memory	Dual_port_block_memory
load_done_output	One of the following keywords: true, false	false

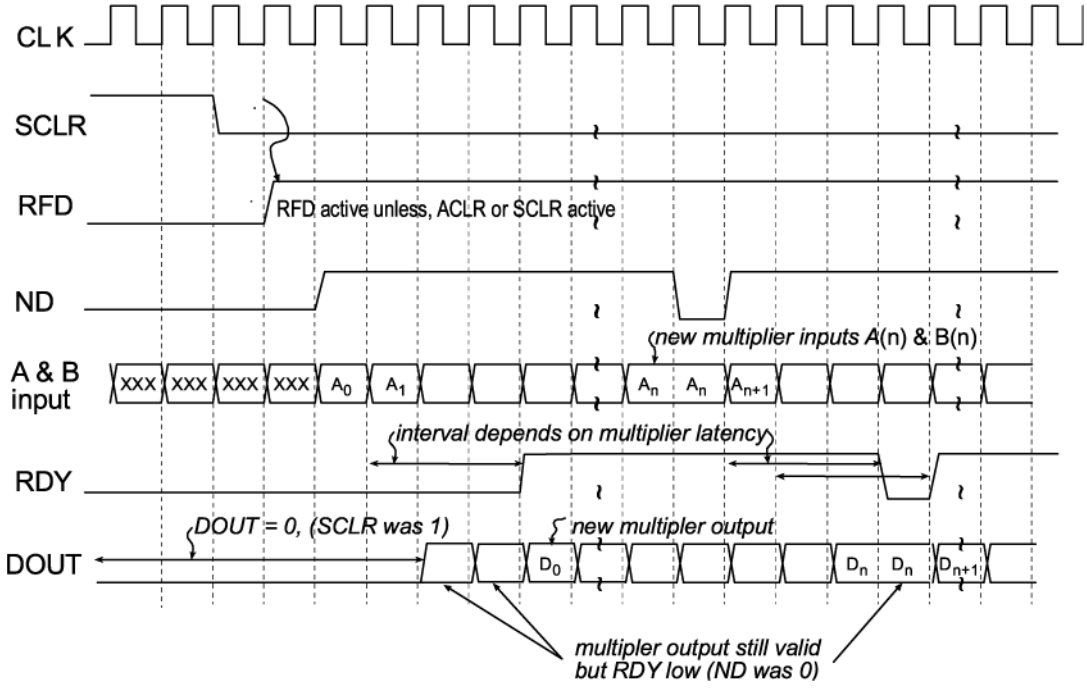

Figure 8: Parallel multiplier timing diagram

Note, that if a parallel constant multiplier, or a sequential multiplier is selected, the RFD signal will go active asynchronously after a clear.

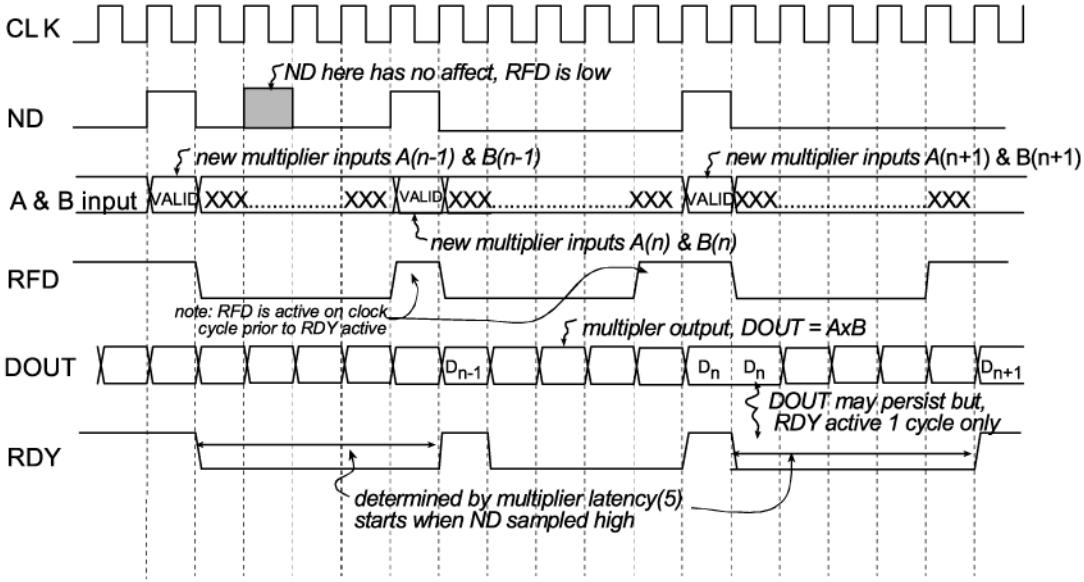

Figure 9: Sequential multiplier with minimum pipeline timing diagram

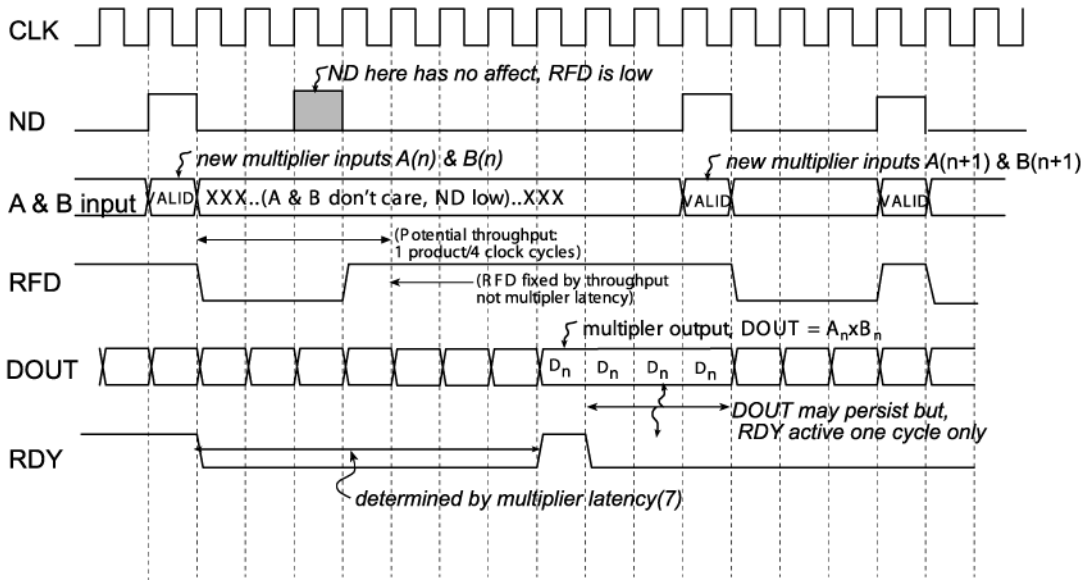

Figure 10: Sequential multiplier with maximum pipeline timing diagram

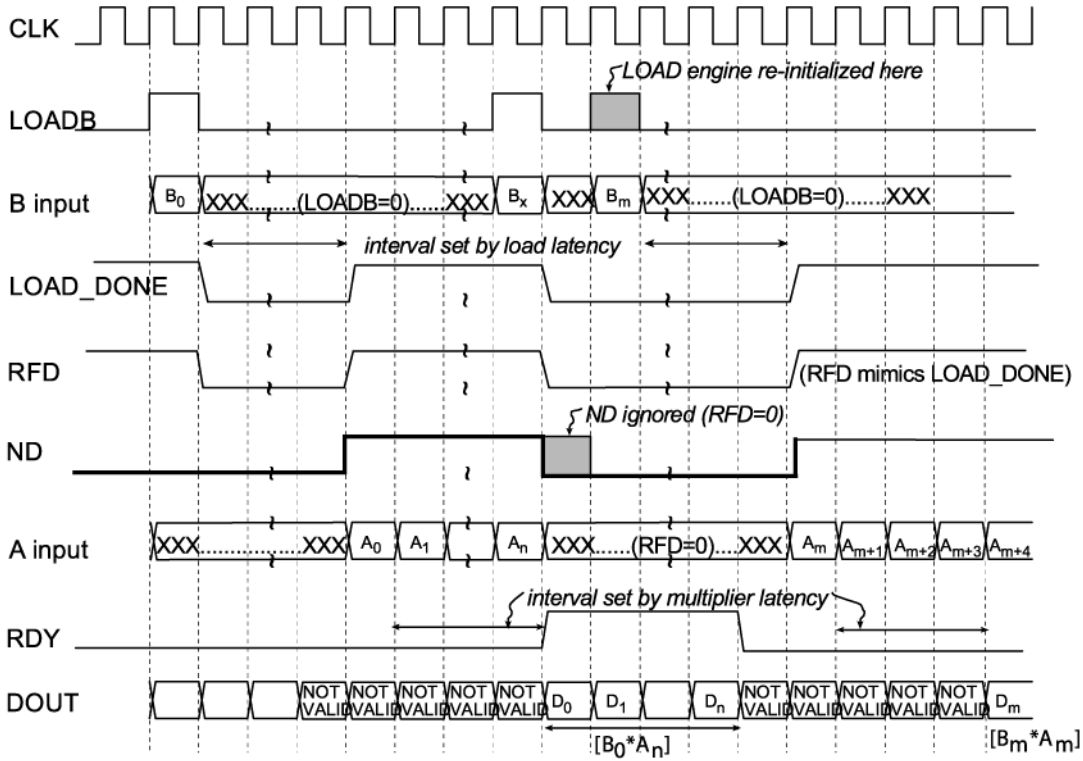

Figure 11: Parallel constant multiplier with reloadable b input timing diagram

Note, load latency is the time it takes the multiplier to reload the b constant value. This time is dependent on the following function:

load latency = 2^w

if (a_width >= rom_addr_width) w=rom_addr_width

else w=a_width

a_width : the width of the a input

rom_addr_width : 4 if memory type = distributed

8 if memory type = block and family = Virtex

9 if memory type = block and family = Virtex2

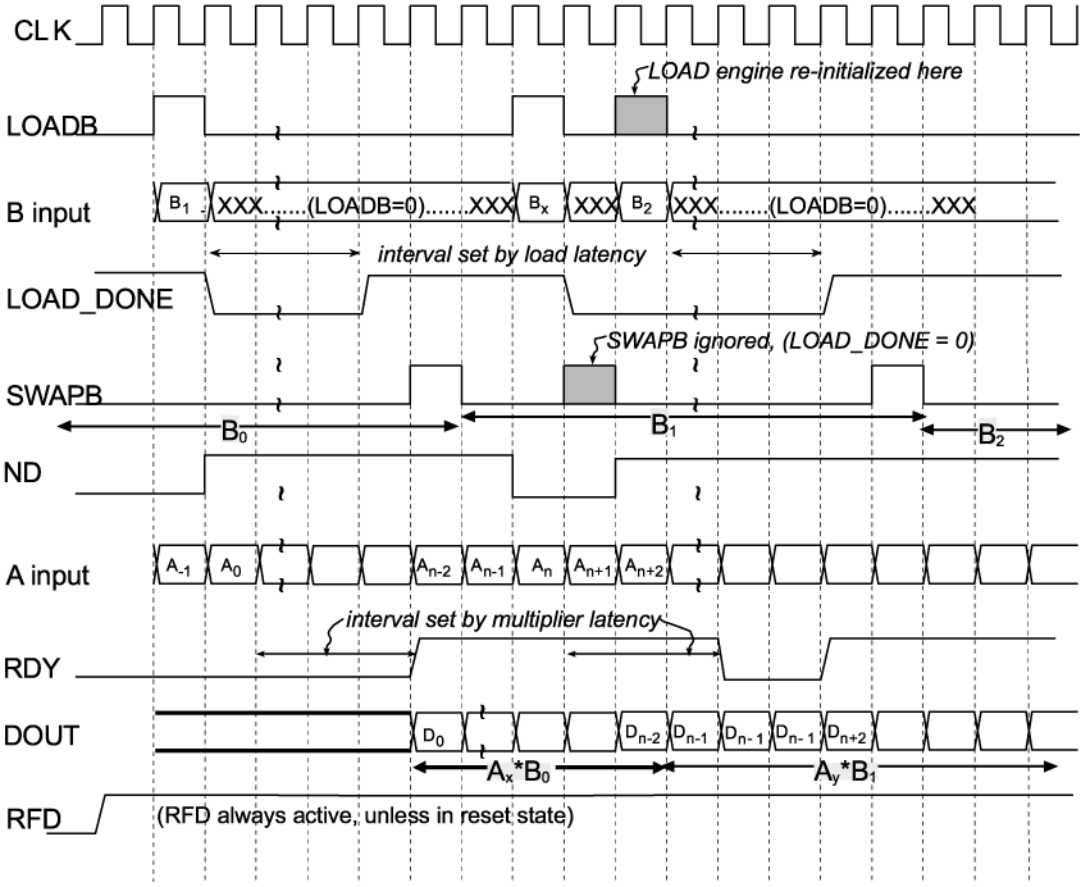

Figure 12: Parallel constant multiplier with reloadable swap b input timing diagram

Note, that when a sequential multiplier is used with a reloadable b input, it is possible to reload the b constant value while in the middle of a multiplication. In this case, all available control signals will be cleared, enabling the next multiplication to begin once the new b constant value is available.