Multiply Accumulator v4.0

DS336 April 28, 2005 **Product Specification**

Features

- Drop-in module for Virtex™, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-4, Spartan™-II, Spartan-IIE, Spartan-3, and Spartan-3E FPGAs
- Parallel multiply accumulator module
- **A** and **B** inputs use Unsigned or Signed data up to 32 bits wide
- Automatic system-level performance matching
- Optional user-defined pipelining
- Result Rounding Modes
  - convergent
  - truncation
  - round away from zero
- Support for saturation arithmetic
- Hand-shaking signals allow easy interfacing with other modules
- Fully configurable using v7.1i or higher of the Xilinx CORE Generator™ system

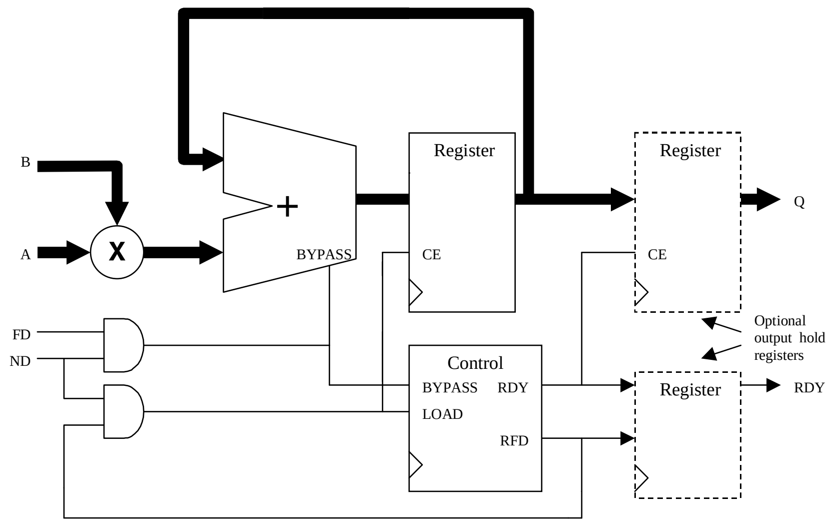


Figure 1: **MAC Block Diagram**

Functional Description

The overall function of the Multiply Accumulator (MAC) is shown by the following equation:

$$Q = \sum_{n=0}^{count-1} (\pm 1) * A(n) * B(n) \qquad \text{Equation 1}$$

Q is the primary data output of the core. **A** and **B** are multiplied together and the product added or subtracted from the current result. A simplified schematic of the core is shown in Figure 1.

The *count* value in Equation 1 is set to a fixed value by a parameter.

Multiply

A(n) and **B(n)** are data bus input ports which are sampled on the rising edge of the clock when the **ND** (New Data) input signal is asserted. Both ports are independently configurable as either signed or unsigned data, with bit precision in the range 2 to 32 bits. With Virtex-4 devices the input width is limited to 18 bits.

Multiplier Pipelining

The module can optionally be user or automatically pipelined. When pipelining is enabled the critical paths in the circuit are reduced, therefore increasing the maximum attainable core clock frequency. The introduction of pipelining registers increases the MAC latency.

Automatic pipelining is the default mode and uses the user specified clock cycle frequency to determine what pipelining stages are needed to meet performance. The higher the specified processing clock frequency the more pipeline stages are added.

The user may disable the automatic pipelining mode and then selectively enable various pipeline options. These include input, output and internal multiplier registers.

The output register option adds a register to the results port and will hold the MAC operation result until the next **RDY** (Ready Output) signal is asserted.

Accumulator

The accumulator accepts a full resolution output result from the multiplier and performs a sum-of-products calculation. For performance and logic placement reasons the user can optionally define parameters that will force the accumulator to be pipelined.

Table 1 shows the accumulator type resulting from all possible data input types.

Table 1: **Input and Accumulator Data Types**

Port A Type	Port B Type	Accumulator Output
Unsigned	Unsigned	Unsigned
Don't Care	Signed	Signed
Signed	Don't Care	Signed

Rounding Modes

The MAC output result width is controlled by the **c_mac_result_width** (integer) .xco parameter. The MAC output defaults to full-precision, with this bit field supporting the maximum positive and most negative values implied by the MAC function being computed. The full resolution width is: (A width + B width + log base 2 of the **c_mac_count** (integer) parameter)-bits. When the **c_mac_result_width** parameter is defined to be less than the full precision result width, rounding is applied to the full precision accumulator value to generate the reduced precision output value presented on the core **Q** port. The rounding mode is defined by the **c_round_operation** xco parameter**.** The following three rounding modes are supported:

- Truncation
- Round away-from-zero
- Convergent Rounding

Truncation

Truncation removes the low order bits of the MAC result. For example, if the full precision result is 36 bits and the user specifies a 32-bit result, then the low order four bits, bits 0 to 3, would be removed before presenting the result on the **Q** port.

Round-Away-From-Zero

The *round-away-from-zero* rounding mode supports simple rounding of a fractional value to the nearest integer quantity. For a positive-valued full precision result x, $x \geq 0$, the rounded result q(x) is defined as $q(x) = \lfloor x + 0.5 \rfloor$ where $\lfloor . \rfloor$ denotes the *floor* operator. For a negative-valued full-precision result x, $x < 0$, the rounded result is computed according to $q(x) = \lceil x - 0.5 \rceil$ where $\lceil . \rceil$ is the *ceiling* operator. Table 2 summarizes the rules employed in the *round-away-from-zero* algorithm and provides some simple examples.

It should be noted that when the fractional component of a full-precision value is precisely ½ the round-away-from-zero algorithm will introduce a statistical bias into the output time series because these borderline cases are always rounded towards $+\infty$ for positive full-precision values and towards $-\infty$ for negative full-precision samples.

Table 2: **Round-Away-From-Zero Summary and Examples**

Result	Fractional Field	Operation	Example
positive	< ½	truncate	102d = 01100110b, round to 4 bits => 0110.0110b = 6.375, LSBs = .0110b = .375 < ½ , after rounding result equals 0110b = 6d
positive	> = ½	Add 1 to result bits	106d = 01101010b, round to 4 bits => 0110.1010b = 6.625, LSBs = .1010b = .625 > ½ , after rounding result equals 0111b = 7d
negative	< ½	truncate	-110d = 10010010b, round to 4 bits => 1001.0010b=6.875, LSBs = .0010b = .125 < ½ , after rounding result equals 1001b = -7d
negative	> = ½	Subtract 1 from the result bits	-104d = 10011000b, round to 4 bits => 1001.1000b = -6.5, LSBs = .1000b = .5 = ½ , after rounding result equals 1000b = -8d

Table 3: **Convergent Rounding Summary and Example**

Fractional Field	Operation	Example
< ½	Truncate	102d = 01100110b, round to 4 bits => 0110.0110b = 6.375, LSBs = .0110b = .375 < ½ , after rounding result equals 0110b = 6d
> ½	Add 1 to result bits	106d = 01101010b, round to 4 bits => 0110.1010b = 6.625, LSBs = .1010b = .625 > ½ , after rounding result equals 0111b = 7d
= ½	Truncate or add 1 to result based on LSB of result	-104d = 10011000b, round to 4 bits => 1001.1000b = -6.5, LSBs = .1000b = .5d = ½ , the result LSB bit is 1 therefore add 1 to the result: 1001 + 0001 = 1010 = -6d

Convergent Rounding

Convergent rounding is a bias-free rounding scheme that is sometimes referred to as *round-to-nearest even*, which is frequently shortened to simply *round-to-even*. It is similar in nature to the round-away-from-zero algorithm, however, the borderline case when the fractional component of a value is exactly ½ is treated differently. A shortcoming of the roundaway-from-zero algorithm involves how to round a number that is midway between two possibilities. For example, if the value is 1.5, should the rounded result be 1 or 2? The convergent rounding algorithm adopts the convention where a number with a fractional component equal to ½ is rounded up or down so that the least significant digit of the result is even. Thus, 1.5 and 2.5 are both rounded to 2, but 3.5 is rounded to 4. Convergent rounding avoids the introduction of a statistical bias in the output sequence since it will round upward about 50% of the time and downward about 50% of the time for most real world applications. This rounding mode is preferred in many signal processing applications, particularly when the MAC is used in a multistage multirate filter that is implementing a large sample rate change. Table 3 provides a summary of the convergent rounding algorithm along with several examples.

Saturation

With either the round-away-from-zero or convergent rounding modes there is a potential for the rounded result to overflow the dynamic range of the accumulator (accumulator wrap-around). To avoid this issue, the rounding modes may be used in conjunction with saturation arithmetic.

For an 8-bit signed number, 01111111b = +127d and 10000000b = -128d are the most positive and most negative values respectively. Now consider a 12-bit precision MAC result that is to be rounded to 8 bits. If the output word is 01111111.1010b = 127.625d, in the ideal case, both round-away-from-zero and convergent rounding would produce a rounded output value of +128. However, this exceeds the dynamic range of the 8-bit result field. Saturation arithmetic will ensure that the rounded result does not experience word wrapping. In this case the rounded output would be clamped to the maximum positive value of +127.

Interface and Control

A number of control signals allow interfacing with external logic. There are two control input signals, **FD** (First Data) and **ND** (New Data). **FD** is asserted for the first input to a new accumulation. **ND** indicates that new data is available to the module at port **A**.

The interaction between the control signals **FD** and **ND** is illustrated in Figure 2. In this example, the MAC is configured to multiply and accumulate four values (**MAC Count** = 4). The output, **Q**, is registered (**Output Hold Registers** = 1).

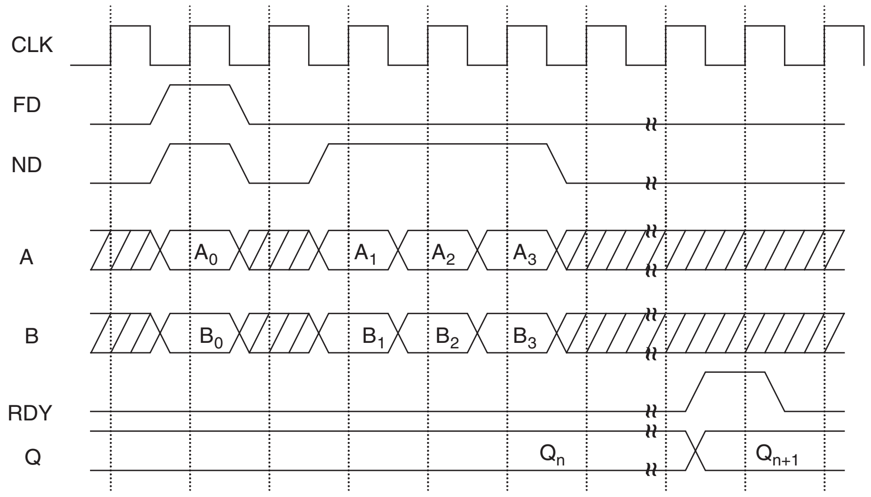


Figure 2: **Control Timing Example: MAC**

Core Pinout

A representative symbol with signal names is shown in Figure 3. Optional ports are illustrated with dotted lines.

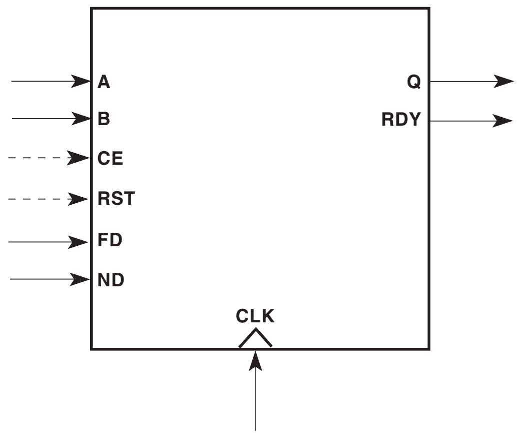


Figure 3: **Core Schematic Symbol**

The core signals are defined in Table 4 and described in detail in the remainder of this section. The mandatory ports are essential to the core; optional ports depend on the parameters used.

Table 4: **Core Signal Pinout**

Name	Direction	Description
Mandatory Signals		
A	Input	Primary multiplier data input bus
B	Input	Secondary multiplier data input bus
FD	Input	**First Data**: indicates the start of a new MAC operation (active high)
ND	Input	**New Data**: indicates that new input data is present (active high)
CLK	Input	**Clock**: Active on the rising edge
Q	Output	MAC data output bus
RDY	Output	Signals that a new result is available on 'Q'
Optional Signals		
CE	Input	**Clock Enable**
RST	Input	**Synchronous Clear**

The following is a short description of each pin and GUI parameters that affect the core's operation.

A and B Data Input

A(n) and **B(n)** are input data ports with independently definable width set for each port through the a_width and b_width parameters. The data ports are sampled on the rising edge of the clock when the **ND** and **CE** are asserted high.

Q Output

The **Q(n)** output port presents the results of the sum-of-products operation. The width of the port is defined by the user parameter **result_width**. In the case where the **result_width** is defined to be less then full precision then the result will be rounded according to the user defined method before presenting on the **Q(n)** port.

FD (First Data) Input

The **D** input identifies the first pair of operands being present on the **A** and **B** data ports. When **FD, ND,** and **CE** (if enabled) are all asserted the product of the **A** and **B** values is used to load the accumulator. Note that the user may assert the **FD** signal for any data pair to effect the loading of the accumulator. **FD** is also used to create the **RDY** signal after **mac_count** sample pairs have been loaded. If the user asserts the **FD** signal before all **mac_count** sample pairs have been loaded then the **RDY** signal will not be generated for the prior data set. The user may optionally also not assert the **FD** signal after all mac_count sample pairs have been loaded. In this case the MAC will continue to accumulate new sample result values, **RDY** will be asserted based on the **mac_count**.

ND (New Data) Input

The **ND** signal qualifies that the data on the **A(n)** and **B(n)** is to be inserted into the MAC. **ND** is internally qualified with the **CE** (if enabled) signals.

RDY (Ready) Output

The optional **RDY** output is asserted High when the result of an accumulation is ready. **RDY** indicates that **Q** is valid.

CLK (Clock)

This signal determines the rate of operation of the MAC core.

CE Input

An optional clock enable capability can be added to the core. The clock enable input port **CE** is active high and is used to control the entire MAC implementation. When the signal is asserted and on a rising edge of the clock the MAC state can change. If **CE** is de-asserted then the state of the MAC will not change. Note that in the case where the reset (**RST**) capability is added the assertion of the **RST** signal does override the **CE** signal.

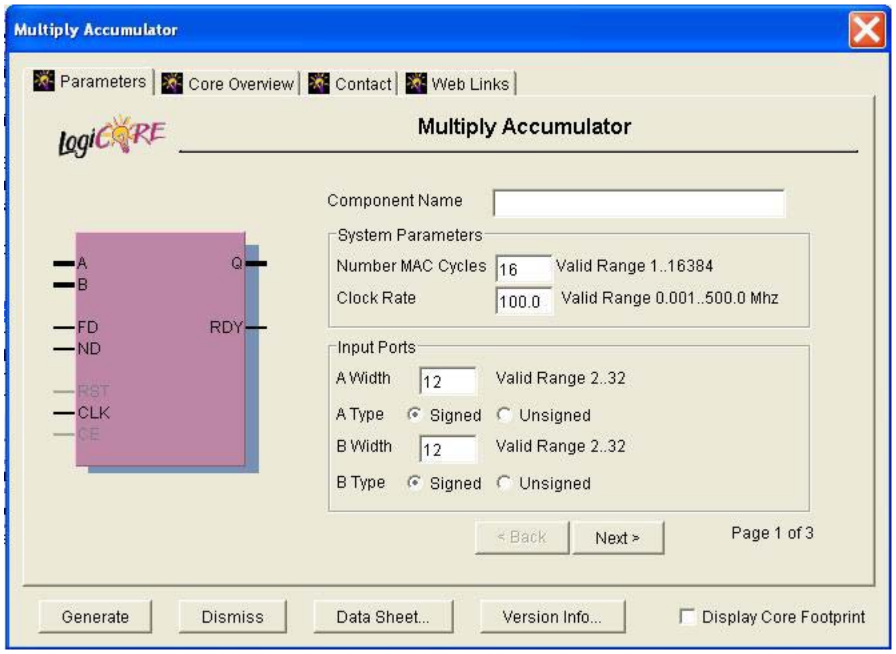
RST Input

This optional Synchronous Clear input is active high. When asserted, the content of the accumulator is cleared to zero.

Core Parameters

The core GUI provides a number of parameters to allow a wide selection of MAC modules to be generated. This section describes the function and the range of each parameter.

The first window of the MAC GUI is shown in Figure 4 and the related options are described below.



Component Name

System Parameters

Number MAC Cycles 16 Valid Range 1..16384

Clock Rate 100.0 Valid Range 0.001..500.0 Mhz

Input Ports

A Width 12 Valid Range 2..32

A Type [✓ Signed] Unsigned

B Width 12 Valid Range 2..32

B Type [✓ Signed] Unsigned

Page 1 of 3

Display Core Footprint

Figure 4: **MAC GUI Window 1**

- **Component Name**

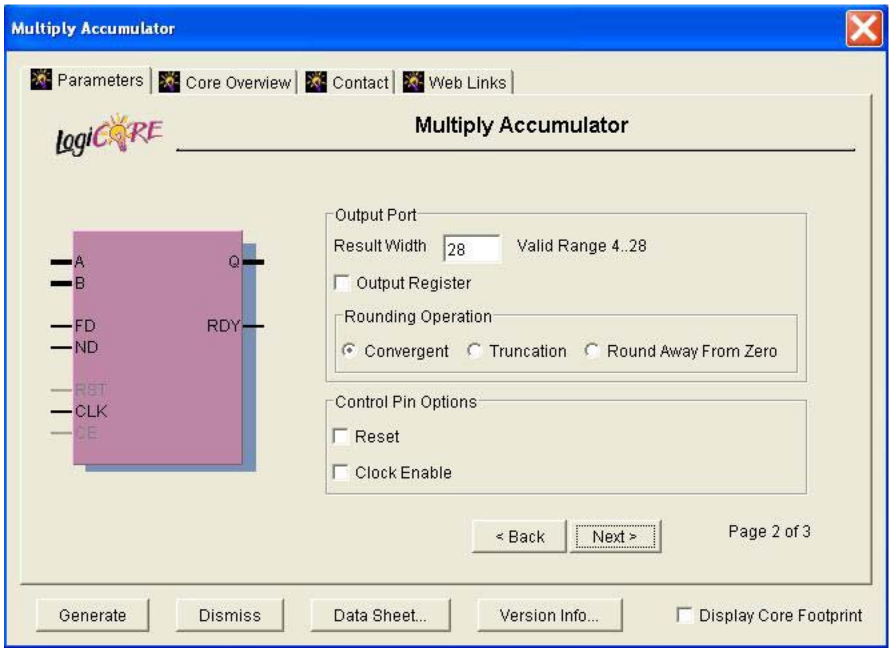
  The component name is used as the base name of the output files generated for this module. Names must begin with a letter and must be composed from the following characters: a to z, 0 to 9, and "_".

- **System Parameters**
  - **Number of MAC Cycles**: The number of multiply-accumulator operations between first data signals.
  - **Clock Rate**: defines the MAC clock frequency that the user requires.

- **Input Ports**
  - **Port A Width**: The width of the **A** input. The valid range is 2 to 32 bits. For Virtex-4 devices, the input range is 2 to 18 bits for signed inputs and 2 to 17 bits for unsigned input.
  - **Port A Type**: The type of A input data. The default value is Signed. **A** input data type may be Unsigned or Signed.
  - **Port B Width**: The width of the **B** input. The valid range is 2 to 32 bits. For Virtex-4 devices, the input range is 2 to 18 bits for signed inputs and 2 to 17 bits for unsigned input.
  - **Port B Type**: The type of **B** input data. The default value is Signed. **B** input data type may be Unsigned or Signed.

The second window of the MAC GUI is shown in Figure 5 and the related options are described below.

- **Output Port**
  - **Result Width**: Number of bits required on the output.
  - **Output Register**: When selected, the results of the MAC operation will be held in a result register.
- **Rounding Operation**: When the result width is defined to be smaller than full precision, the user can select the type of rounding: convergent, truncation, or round away from zero.
- **Control Pin Options**
  - **Reset**: When selected, a Synchronous clear capability is added to the implementation.
  - **Clock Enable**: When selected a clock enable capability is added to the implementation.


Multiply Accumulator

Output Port
Result Width 28 Valid Range 4..28
Output Register [ ]
Rounding Operation: [● Convergent] Truncation Round Away From Zero

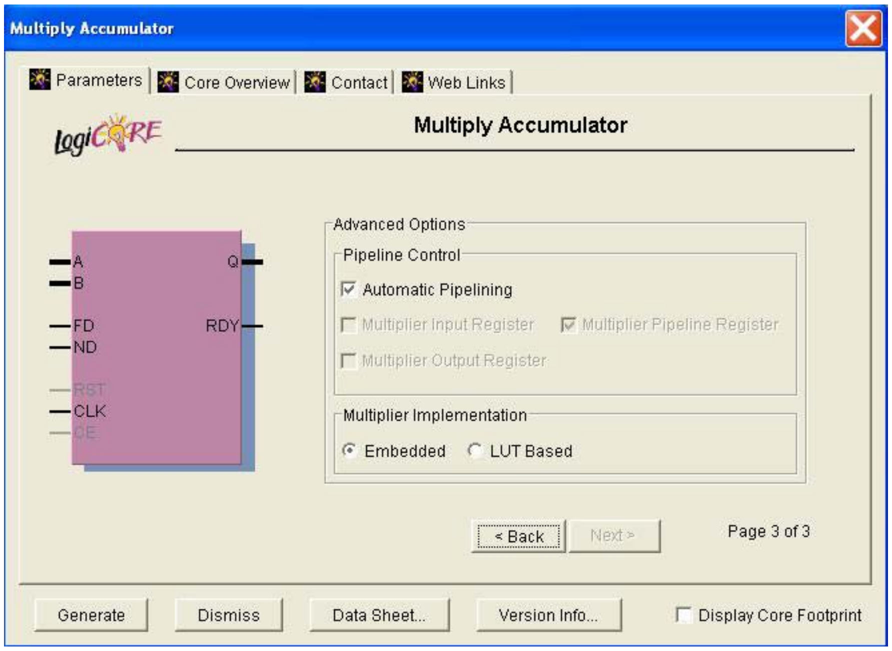
Control Pin Options
Reset [ ]
Clock Enable [ ]

Page 2 of 3

Display Core Footprint [ ]

Figure 5: **MAC GUI Window 2**

The third window of the MAC core is shown in Figure 6 and the options are described below.



Advanced Options

Pipeline Control

[✓] Automatic Pipelining

[ ] Multiplier Input Register [✓] Multiplier Pipeline Register

[ ] Multiplier Output Register

Multiplier Implementation

[✓ Embedded] LUT Based

Figure 6: **MAC GUI Window 3**

- **Advanced Options**
  - **Pipelined Control**
    - **Automatic Pipelining**: When selected, the core will use the user defined clock frequency performance to determine where and how many pipeline stages need to be added to the implementation.
    - **Multiplier Input Register**: If the Automatic pipelining option is disabled then the user can select to add an input register for the A and B values.
    - **Multiplier Output Register**: If the Automatic pipelining option is disabled then the user can select to add an output register for the results of the multiplier.
    - **Multiplier Pipelined Register**: If the Automatic pipelining option is disabled then the user can select to add internal registers in the multiplier.
  - **Multiplier Implementation**
    - **Embedded**: Forces the implementation to use embedded multipliers if available in the target architecture.
    - **LUT Based**: Forces the implementation to construct a multiplier from LUT resources.

Parameter Values in the XCO File

Names of XCO parameters and their parameter values relate directly to the names and values shown in the GUI, except that the underscore character, "_," is used in place of spaces. The text in an XCO file is case insensitive.

Table 5 shows the XCO file parameters and values and summarizes the GUI default settings.

The following is an example of an XCO file:

```
CSET component_name = mac1
CSET a_width = 8
CSET b_width = 8
CSET a_type = signed
CSET b_type = signed
CSET has_output_register = false
CSET has_mult_input_register = false
CSET has_mult_pipeline_register = false
CSET has_mult_output_register = false
CSET mac_count = 4
CSET round_operation = Convergent
CSET result_width = 18
CSET use_embedded_mult = Embedded
CSET clock_frequency = 100.0
CSET auto_pipeine = true
CSET has_clock_enable = false
CSET has_reset = true
```

Table 5: **Default Values and XCO File Values**

Parameters	XCO File Value	Default GUI Settings
Clock_Frequency	Desired MAC operational frequency in megahertz. : 0.001 to 500.0 MHz	100.0 MHz
Auto_Pipeline	One of the key words: true or false	True
Mac_count	Integer from 1 to 16383	16
A_Width	Integer from 2 to 32	16
B_Width	Integer from 2 to 32	16
A_Type	One of the key words: signed or unsigned	Signed
B_Type	One of the key words: signed or unsigned	Signed
Result_Width	Integer from 4 to 80	28
Round_Operation	One of the key words: "Convergent", "Truncation" or "RoundAwayFromZero"	Convergent
Use_Embedded_Mult	One of the key words: true or false	True
Has_Clock_Enable	One of the key words: true or false	False
Has_Reset	One of the key words: true or false	False
Has_Mult_Input_Register	One of the key words: true or false	False

Table 5: **Default Values and XCO File Values *(Continued)***

Parameters	XCO File Value	Default GUI Settings
Has_Mult_Pipeline_Register	One of the key words: true or false	False
Has_Mult_Output_Register	One of the key words: true or false	False
Has_Output_Register	One of the key words: true or false	False

Table 6: **Performance and Size Characteristics**

Multiplier Size	Accumulator Size	Target Device	MHz	Slice Count	User Defined Clock Frequency
16x16	36	2VP20 –7	344 MHz	77	320 MHz
18x18	40	2VP20 –7	343 MHz	85	320 MHz
18x18	44	2VP20 –7	319 MHz	90	300 MHz
18x18	48	2VP20 –7	289 MHz	94	275 MHz
18x18	50	2VP20 –7	263 MHz	98	250 MHz

Performance and Size Characteristics

Table 6 provides performance and logic resource utilization for several MAC configurations. Note that these are optimal numbers and overall utilization of the device may slightly affect performance. In all cases the core was configured for automatic pipelining based on the user XCO parameter **clock_frequency**.

Behavioral Models

When the core is generated by the CORE Generator system, VHDL and Verilog functional behavioral models may also be generated, depending on the CORE Generator options selected. When the Verilog model is compiled there may be some warnings referring to components within the Verilog model, similar to the one shown below. These are a feature of the automatically generated Verilog model and can be ignored.

```
WARNING: ./mac_v2_0_model.v(352): [TFMPC] -
 Too few port connections.
# Region: /testbench/uut_bhv_vlog/BU203
```

Ordering Information

This core can be downloaded, free of cost, from the Xilinx IP Center for use with the Xilinx CORE Generator™ System v7.1i and later. The CORE Generator system is bundled with all Xilinx Foundation Series Development Software packages.

Please contact your local Xilinx sales representative or visit the Xilinx IP Center for information about additional Xilinx LogiCORE modules.

Revision History

Date	Version	Revision
04/28/05	1.0	Initial Xilinx release in corporate template.