LogiCORE IP Linear Algebra Toolkit v1.0

DS829 March 1, 2011 **Product Specification**

Introduction

The Xilinx LogiCORE™ IP Linear Algebra Toolkit (LAT) v1.0 implements matrix-matrix addition, matrix-matrix subtraction, matrix-matrix multiplication, and matrix-scalar multiplication. It provides a high degree of flexibility, allowing the IP generated to be tailored to a vast range of end user applications. All functions generated are compliant with the AXI4-Stream specification, facilitating system integration.

Features

- Drop-in module for Virtex®-6, Virtex-7, and Kintex™-7 FPGAs
- AXI4-Streaming Protocol compliant core
- Fully synchronous, single clock domain design with synchronous active low reset
- Fixed-point matrix-matrix addition/subtraction, matrix-scalar multiplication, matrix-matrix multiplication
- Configurable matrix dimension up to 32x32
- Real or complex data support
- Two's complement fixed-point maximum of 24-bit input and 48-bit output
- Symmetric, asymmetric, convergent, and truncation options for output rounding with output saturation support
- Clock Enable (CE) input, can pause core and resume on-the-fly by deasserting/reasserting CE
- User selectable folding factor for resource optimization
- Data I/O Support: serial or parallel
- Uses SIMD mode of DSP48E1 to implement complex addition on a single slice
- DSP48E1 slice or LUT based implementation for matrix-matrix addition/subtraction
- Folding factors supported for matrix size MxN are: 1, 2, 3, …, M, N, MN
- Folding factors supported for matrix multiplication $C_{MxN} = A_{MxL} \times B_{LxN}$ are: 1, 2, 3, …, M, MN
- Bit-accurate C Model available
- Customer demonstration test bench

LogiCORE IP Facts Table				
Core Specifics				
Supported Device Family[1]	Kintex-7, Virtex-7, Virtex-6L, Virtex-6			
Supported User Interfaces	AXI4-Stream			
	Resources[2]			**Frequency**
Configuration	**LUT-FF Pairs**	**DSP Slices**	**Block RAMs**	**Max. Freq.[3]**
Matrix-Matrix Addition (6x5, Real)	4,213	30	0	471
Matrix-Scalar Multiplication (6x5, Complex)	8,785	120	0	471
Matrix-Matrix Multiplication (8x8 with 8x1, Real)	3,550	32	0	440
Provided with Core				
Documentation	Product Specification, Getting Started Guide, User Guide			
Design Files	Netlist			
Example Design	Not Provided			
Test Bench	VHDL			
Constraints File	Not Provided			
Simulation Model	Verilog/VHDL Model and Bit Accurate C Model			
Tested Design Tools				
Design Entry Tools	CORE Generator™ tool 13.1			
Simulation	ISim 13.1, Mentor Graphics ModelSim 6.6d			
Synthesis Tools	XST 13.1			
Support				
Provided by Xilinx, Inc.				

1. For a complete listing of supported devices, see the release notes for this core.
2. Resources listed here are for Virtex-6 devices. See Table 7 for complete device performance numbers.
3. Performance numbers are for Virtex-6 FPGAs. See Table 7 for complete device performance numbers.

Applications

- LAT base functions (addition/subtraction, scalar and matrix multiplications) IP core ideally suited for high performance streaming matrix-data processing, various types of digital signal processing needs on 1-D and 2-D data. This highly scalable and configurable core provides a wide choice of cost-performance trade-off for various types of Radar, Sonar and other horizontal signal and data processing requirements.
- Provides high-performance and well optimized base functions as building blocks for MIMO (Multiple Input Multiple Output) transmit-receive processing in wireless communication system.

Functional Description

The LAT v1.0 supports fixed-point matrix-matrix addition, subtraction, matrix-scalar multiplication, and matrix-matrix multiplication. These operations are represented by the following equations.

Table 1: **Matrix-Matrix Addition Subtraction, Matrix-Scalar Multiplication, Matrix-Matrix Multiplication Equations**

Matrix-Matrix Addition Subtraction	$C_{M \times N} = A_{M \times N} \pm B_{M \times N}$ $C_{M \times N}(i,j) = A_{M \times N}(i,j) \pm B_{M \times N}(i,j)$
Matrix-Scalar Multiplication	$C_{M \times N} = A_{M \times N} \times b$ $C_{M \times N}(i,j) = A_{M \times N}(i,j) \times b$
Matrix-Matrix Multiplication	$C_{M \times N} = A_{M \times L} \times B_{L \times N}$ $C_{M \times N}(i,j) = \sum_{k=1}^{L} A_{M \times L}(i,k) \times B_{L \times N}(k,j)$

These operations are four functional modes (`OP_MODE`) of the core. A and B (becomes scalar input b for matrix-scalar multiplication) are two input matrices, and C is the output matrix. Matrix dimensions are configurable through three dimension parameters M, N, and L. Once `OP_MODE` and dimension parameters are selected, individual matrix dimensions are decided automatically as shown in Table 1. The next set of parameters is the data width of individual matrices, `A_WIDTH`, `B_WIDTH` and `C_WIDTH`, representing precision of elements of matrices A, B, and C, respectively. There are two AXI4-Stream compliant slave ports for input matrices and one master port for the output matrix C (port A, port B, and port C, respectively) as shown in the block diagram of the core in Figure 4. Data bus (TDATA) widths for all three ports are decided internally based on the data width, matrix dimension, and folding factor. Folding factor (F) is a user configurable parameter for deciding the resource reuse (mostly DSP48E1 slices) of the core. Folding factor also dictates the processing latency and streaming latency (or conversion rate) of the mode. As this is a fully real-time streaming core, input matrices must be fed to the core in F (or less) contiguous cycles of the input clock and this constraint applies to both A and B matrices. The next batch of matrices can be fed in the immediate following clock cycle making the core fully streaming capable. Similarly, the output matrix C is made available at the output port over F (or less) contiguous cycles. The core can be paused and resumed dynamically using a clock enable (CE, active high) signal at the input of the core. This feature can be used for flow control if needed; the CE can be tied to HIGH if not necessary. The two I/O modes, serial and parallel, are for data entry and exit to and from the core. In serial mode, matrix elements are entered serially, element-by-element and the output matrix exits the core in similar fashion. While in parallel mode, matrix elements are packed and a parallel bus is formed for each of the three ports. Packing is row-wise, the first element of the row goes to the least significant position (LSP), followed by the next element until the bus width is fully populated. Consequently, a fully packed port A (in parallel mode) has the following packing ([MSP to LSP]):

$[A_{M\times N}(2,3)\ A_{M\times N}(2,2)\ A_{M\times N}(2,1)\ A_{M\times N}(1,5)\ A_{M\times N}(1,4)\ A_{M\times N}(1,3)\ A_{M\times N}(1,2)\ A_{M\times N}(1,1)]$

A detailed description of the four functional modes is provided in subsequent sections.

Matrix-Matrix Addition and Subtraction

In this mode, matrices A, B, and C have the same dimension, MxN (M is the number of rows and N is the number of columns), and the same number of elements (MN). The data widths (i.e., precision) `A_WIDTH`, `B_WIDTH`, and `C_WIDTH` are abbreviated as A_w, B_w, and C_w, respectively. The elements of the corresponding matrix entering (or exiting for C) in one clock cycle are NE_A, NE_B, and NE_C. For this mode, NE_A=NE_B =NE_C and computed as:

$$NE_{A/B/C} = ceil\left(\frac{M\times N}{F}\right) \qquad \text{Equation 1}$$

in parallel I/O mode, which indicates as many elements are packed to form the TDATA bus (of the corresponding port) as elements (adder/subtracter) required for the chosen mode. The selection of $NE_{A/B/C}$ occurs inside the core once `OP_MODE`, dimensions, data width, folding factor, and I/O modes are selected. The core selects the optimum number of elements based on F and the real-time streaming requirement specifying that matrix elements enter (and exit) in F clock cycles or less. TDATA bus widths for three ports are computed accordingly once $NE_{A/B/C}$ are computed. `S_AXIS_A_TDATA`, `S_AXIS_B_TDATA` and `M_AXIS_C_TDATA` (as shown in Figure 4) bus widths are denoted as $ABUS_w$, $BBUS_w$ and $CBUS_w$. The core computes these bus widths as shown in the following equations:

$$BBUS_w = NE_B \times (1 + CMPLX) \times ceil8(B_w) \qquad \text{Equation 2}$$

$$ABUS_w = NE_A \times (1 + CMPLX) \times ceil8(A_w) \qquad \text{Equation 3}$$

$$CBUS_w = NE_C \times (1 + CMPLX) \times ceil8(C_w) \qquad \text{Equation 4}$$

Where, CMPLX is the data type parameter to be configured to zero for real data type and to one for complex data type and *ceil8*(.) is the operator for ceiling (equal to or greater) to an integer divisible by 8. The number of clock cycles required for matrix A/B/C is determined by the following equation (meeting the real-time streaming requirement):

$$K_{A/B/C} = ceil\left(\frac{M\times N}{NE_{A/B/C}}\right) = ceil\left(\frac{M\times N}{ceil\left(\frac{M\times N}{F}\right)}\right) \le F \qquad \text{Equation 5}$$

Serial I/O mode is a special case of parallel I/O mode with NE_A=NE_B =NE_C=1, F=MN and $K_{A/B/C}$=MN, and all previous equations are true with bus widths scaling down accordingly. While entering the data to ports A and B, corresponding port TVALID is asserted High for $K_{A/B/C}$ number of contiguous clock cycles. Similarly, on the output port C, TVALID is asserted High by the core for the same number of contiguous clock cycles once the output is available. Also, TREADY is asserted HIGH (by external AXI4 slave) for these many cycles (at least) for the proper functioning of the core in this mode.

Matrix-Scalar Multiplication

This mode is similar to the matrix-matrix addition mode with the exception of port B where input is scalar; consequently, NE_B=1 and K_B=1. The rest of the parameters for ports A and C remain the same as those for addition mode. In serial mode, port B parameters are unchanged (NE_B=1 and K_B=1), while port A and B parameters scale down as in the addition case with $NE_{A/C}$=1 and $K_{A/C}$=MN.

Matrix-Matrix Multiplication

In this mode, unlike addition mode, the dimensions of matrices A, B and C can be dissimilar and folding factor F scales matrix rows only, giving matrix elements per cycle as:

$$NE_A = ceil\left(\frac{M}{F}\right) \times L \qquad \text{Equation 6}$$

$$NE_B = ceil\left(\frac{L}{F}\right) \times N \qquad \text{Equation 7}$$

$$NE_C = ceil\left(\frac{M}{F}\right) \times N \qquad \text{Equation 8}$$

The bus width equations (Equation 2, Equation 3, and Equation 4) hold true for this mode as well. However, the number of clock cycles going in to matrices A and B and out of matrix C are changed as follows:

$$K_A = ceil\left(\frac{M \times L}{NE_A}\right) = ceil\left(\frac{M \times L}{ceil\left(\frac{M}{F}\right) \times L}\right) = ceil\left(\frac{M}{ceil\left(\frac{M}{F}\right)}\right) \le F \qquad \text{Equation 9}$$

$$K_B = ceil\left(\frac{L \times N}{NE_B}\right) = ceil\left(\frac{L \times N}{ceil\left(\frac{L}{F}\right) \times N}\right) = ceil\left(\frac{L}{ceil\left(\frac{L}{F}\right)}\right) \le F \qquad \text{Equation 10}$$

$$K_C = ceil\left(\frac{M \times N}{NE_C}\right) = ceil\left(\frac{M \times N}{ceil\left(\frac{M}{F}\right) \times N}\right) = ceil\left(\frac{M}{ceil\left(\frac{M}{F}\right)}\right) \le F \qquad \text{Equation 11}$$

This choice of scaling matches the rates of A and C ports, that is K_A=K_C (only in parallel mode). Serial I/O mode is a bit different in this operation. While elements per cycle in serial mode are equal (NE_A=NE_B =NE_C =1), the number of cycles needed for input to the matrix and the folding factor change according to the matrix dimensions. Consequently, K_A=ML, K_B=LN and K_C =MN and the folding factor is as follows:

$$F = max(K_A, K_B, K_C) \qquad \text{Equation 12}$$

In the previous four functional modes, when K_A is less than F, no data should be sent on the cycles beyond K_A in port A. The core also deasserts `S_AXIS_A_TREADY` beyond these K_A cycles. Similarly, when K_B is less than F, no data should be sent on the cycles beyond K_B. The core also deasserts `S_AXIS_B_TREADY` beyond these cycles and when K_C is less than F, no data will be sent on the cycles beyond K_C. The core will also deassert `M_AXIS_C_TVALID` beyond these cycles. Folding factor support is 1-M for parallel I/O mode. However, other folding factors 1-N can be obtained (if needed) by transposing the input matrices (C' = B'A'). Consequently, output matrix C' will also be in the transposed form.

Symmetric rounding to zero, symmetric rounding to infinity, asymmetric rounding to zero, asymmetric rounding to infinity, convergent rounding to even, convergent rounding to odd and truncation modes are supported for rounding. Saturation is performed by default. All internal computations are performed in full precision and rounding-saturation is performed only at the last stage of computation. Data grows through the internal computing stages where full data precision is maintained until the last stage assuming worst case data growth. This is handled intelligently without requiring any extra computing resources. For example, the worst case output precision $C_{w,max}$ for different `OP_MODES` (given input precisions A_w,B_w) are as follows:

`OP_MODE`: matrix-matrix addition/subtraction,

$$C_{w,\ max} = max(A_w, B_w) + 1 \qquad \text{Equation 13}$$

`OP_MODE`: matrix-scalar multiplication,

$$C_{w,\ max} = A_w + B_w + CMPLX \qquad \text{Equation 14}$$

`OP_MODE`: matrix-matrix multiplication,

$$C_{w,max} = A_w + B_w + ceil(log2(L)) + CMPLX \qquad \text{Equation 15}$$

In these four `OP_MODES`, output is allowed to grow up to $C_{w,max}$ (as shown previously) before rounding to the user selected C_W at the last stage of computing.

In matrix-matrix addition-subtraction mode, input data widths (A_w and B_w) can be up to a maximum of 24 bits. The user has the flexibility to choose an implementation based on fabric or DSP48E1 slices. For the complex input case, when DSP48E1 slice based implementation is used, each DSP48E1 slice is configured in the SIMD mode to optimize the resource usage. Rounding and saturation are implemented on the fabric.

In matrix-scalar multiplication mode, A_w can be up to 24 bits and B_w can be up to 18 bits. This limitation is from the DSP48E1 slice, which uses an internal 25x18 multiplier. The implementation is always based on DSP48E1 slices. Rounding and saturation are implemented on the fabric.

In matrix-matrix multiplication mode, A_w and B_w can be up to 24 bits. However, for any given implementation, the choice of A_w and B_w should be such that unit multiplication can be accommodated within one DSP48E1 slice, which uses an internal 25x18 multiplier. Consequently, if A_w is greater than 18 bits, B_w should be less than or equal to 18 bits and vice versa. The implementation is always based on DSP48E1 slices. Rounding and saturation are implemented on the fabric. Output precision C_w can be up to 48 bits.

The user has the flexibility to choose any C_w bits over the range [0, $C_{w,max}$-1] for output using the parameter LSB. C_w can be chosen between [2, $C_{w,max}$]. The parameter LSB indicates the least significant bit location of the output relative to the internal full precision output value, as shown in Figure 1. C_w number of bits, starting from the LSB are sent to the output. As a result, given C_w, LSB can vary from [0, $C_{w,max}$ - C_w].

The user also has the flexibility to choose a folding factor for a given matrix operation to optimize resource utilization when operating in the parallel I/O mode. The folding factors supported are [1 to M, N and MN] for matrix addition, subtraction and scalar multiplication modes, while in matrix multiplication mode, supported folding factors are [1 to M]. However, the folding factor should be selected so it does not violate the AXI4 port TDATA width limitations. In this version, maximum port width is set to 1024 bits for all the three ports (ports A, B and C). In serial I/O mode, the user does not have the flexibility to choose the folding factor, which scales to MN for matrix addition, subtraction and scalar multiplication modes, while for matrix multiplication mode, it is determined by Equation 12.

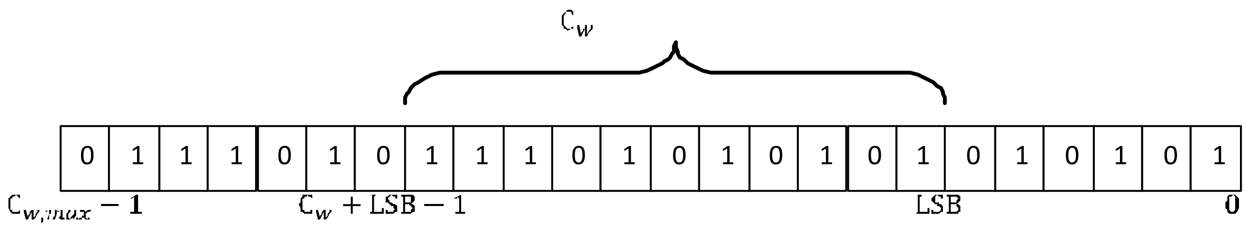


Figure 1: **Output Precision Format**

GUI Parameters

There are a number of parameters available for customizing an implementation of the LAT v1.0. These parameters can be configured using the two pages of the GUI. In addition, the GUI also contains three informational tabs. Tooltips appear when hovering the mouse over parameters that require additional explanation.

Tab 1: IP symbol

IP symbol tab illustrates the core pinout.

Tab 2: Implementation

The implementation tab displays the following details:

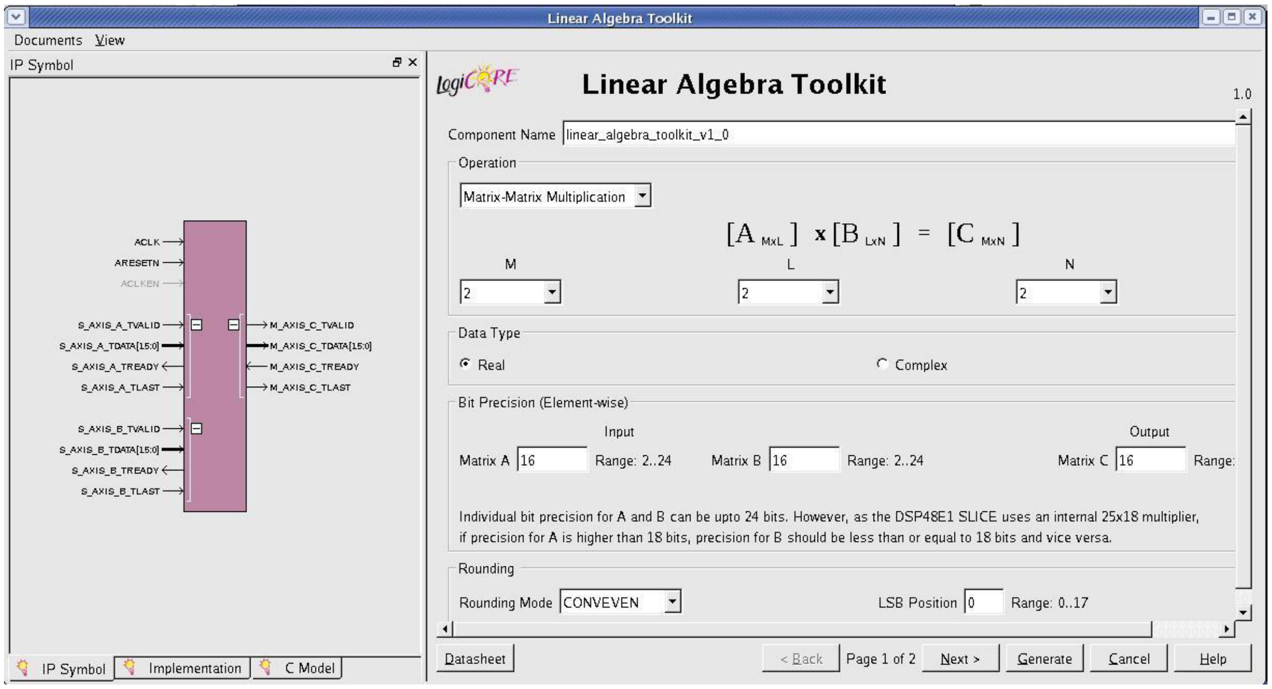
- Number of DSP48E1 slices and Block RAM elements required to implement the design
- Latency Information for the core

Tab 3: C Model

This tab points to information regarding the Bit Accurate C Model for the LAT.

Page 1

The first page of the GUI is used to define the functional mode of the LAT core as shown in Figure 2. Broadly, this page covers the matrix operation to be performed, the dimensions, bit precision of the matrix elements and the rounding options. The parameters are explained in greater detail in Table 2. All together, this page defines the complete functionality of the core.



Component Name: linear_algebra_toolkit_v1_0

Operation: Matrix-Matrix Multiplication

$[A_{MxL}] \times [B_{LxN}] = [C_{MxN}]$

M: 2 L: 2 N: 2

Data Type: [✓ Real] Complex

Bit Precision (Element-wise)

Input: Matrix A 16 Range: 2..24 Matrix B 16 Range: 2..24

Output: Matrix C 16 Range:

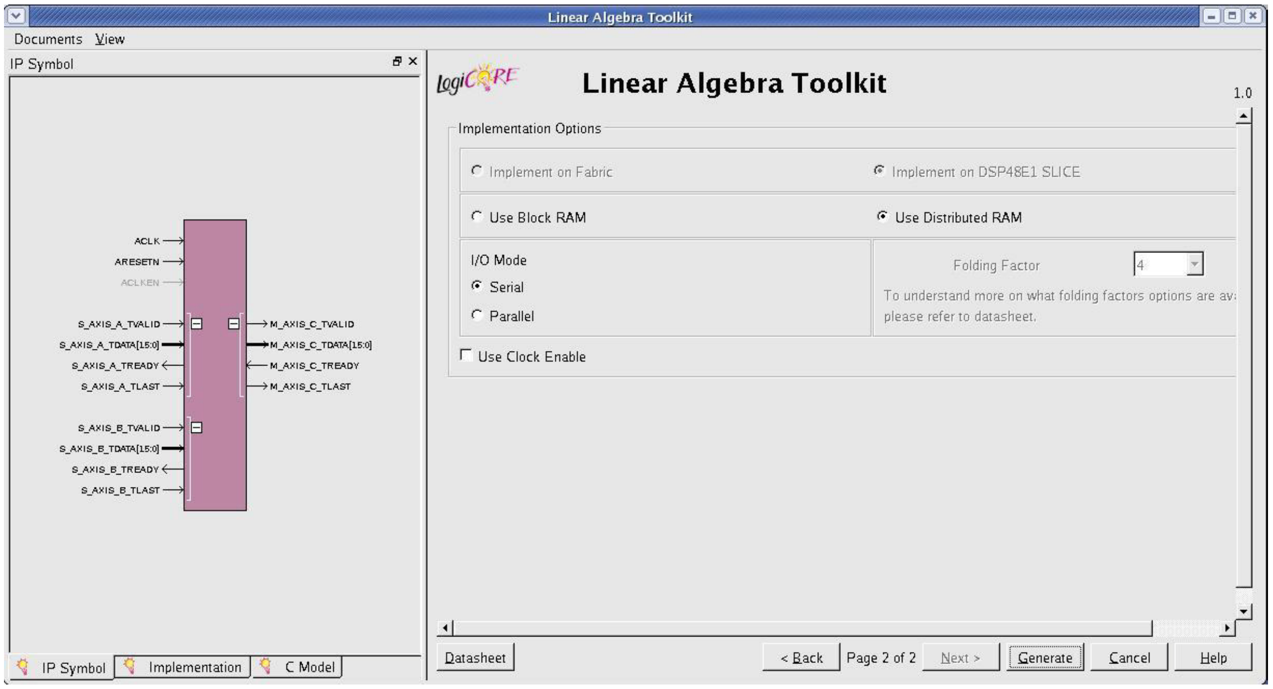
Individual bit precision for A and B can be upto 24 bits. However, as the DSP48E1 SLICE uses an internal 25x18 multiplier, if precision for A is higher than 18 bits, precision for B should be less than or equal to 18 bits and vice versa.

Rounding

Rounding Mode: CONVEVEN LSB Position: 0 Range: 0..17

Figure 2: **LAT GUI First Page**

The second page of the GUI (shown in Figure 3) is used to define the implementation details for the core operation that was defined in Page 1. For the Matrix-Matrix Addition/Subtraction, it is possible to choose between a LUT (fabric) based implementation and a DSP48E1 slice based implementation. For the Matrix-Matrix Multiplication operation, it is possible to choose between a Block RAM based implementation or a Distributed RAM based implementation. In addition, the user has the flexibility to choose between a parallel and serial I/O mode. When in parallel mode, the user can also choose a folding factor as a trade-off between throughput and resource utilization. The parameters with ranges and default settings are explained in detail in Table 2.



Linear Algebra Toolkit 1.0

Implementation Options

Implement on Fabric [✓ Implement on DSP48E1 SLICE]

Use Block RAM [✓ Use Distributed RAM]

I/O Mode: [✓ Serial] Parallel

Folding Factor: 4

To understand more on what folding factors options are av
please refer to datasheet.

☐ Use Clock Enable

Figure 3: **LAT GUI Second Page**

Table 2: **LAT GUI Parameters**

Parameter	Range of Values	Description	XCO Parameter
General Parameters			
Component Name	ASCII text using characters: a..z, 0..9 and "_" starting with a letter	linear_algebra_toolkit_v1_0	component_name
OP_MODE	Matrix-Matrix Addition, Matrix-Matrix Subtraction, Matrix-Scalar Multiplication, Matrix-Matrix Multiplication	This parameter decides which core operation to be performed.	op_mode (The corresponding XCO values are MADD, MSUB, SMULT, MMULT)
M	1-32	Number of rows of input Matrix A, which is same as number of rows of output Matrix C	m
L	1-32	Applicable only to Matrix-Matrix Multiplication where it denotes the columns of A or rows of B (or the inner dimension of matrix multiplication). This value is ignored in other modes.	l

N	1-32	Columns of input Matrix B, which is same as number of columns of output Matrix C.	n
Data Type	Real, Complex	This denotes the data type of elements of Matrix A, Matrix B and Matrix C.	cmplx
Input-Output Precision			
A_WIDTH	2-24	Input bit-width precision for elements of Matrix A. For complex data type it indicates real (or imaginary) field precision as both real and imaginary fields are assumed to be of same precision.	a_width
B_WIDTH	2-18 for Matrix Scalar multiplication, 2-24 otherwise	Input bit-width precision for elements of Matrix B. For complex data type it indicates real (or imaginary) field precision as both real and imaginary fields are assumed to be of same precision.	b_width
C_WIDTH	2-MAX_PRECISION	Output bit-width precision for elements of Matrix C. For complex data type it indicates real (or imaginary) field precision as both real and imaginary fields are assumed to be of same precision. MAX_PRECISION is 25 bits for addition/subtraction modes and 48 for scalar and matrix multiplication modes.	c_width
LSB	0-MAX_PRECISION-C_WIDTH	Determines the LSB location of the output relative to the internal full precision output value.	lsb
Rounding Mode	SYMZERO SYMINF ASYMZERO ASYMINF CONVEVEN CONVODD TRUNC	SYMZERO: Symmetric Rounding, Round down in absolute value SYMINF: Symmetric Rounding, Round up in absolute value ASYMZERO: Asymmetric Rounding, rounds towards negative infinity ASYMINF: Asymmetric Rounding, rounds towards positive infinity CONVEVEN: Convergent Rounding towards even CONVODD: Convergent Rounding towards odd TRUNC: Remove bits from LSB-1 down to zero	rnd_mode
Implementation Specific Parameters			

Implement on Fabric/DS48E1 Slice	Radio Button	For Matrix-Matrix Addition/Subtraction the implementation can be on Fabric or DSP48E1 slice. Only one of these two options can be selected, default is implementation on fabric. Note, matrix-scalar multiplication and matrix-matrix multiplication are implemented on DSP48E1 and hence this choice is not available for these modes.	impl_mode
I/O Mode	Serial, Parallel	Selects whether data will be transferred serially or in parallel. Default I/O mode is serial.	io_mode
Folding Factor	1-M, N (addition/subtraction and scalar multiplication modes) 1-M (matrix multiplication mode)	Implies the amount of reuse of the resources. Roughly, F implies (1/F)*100% use of resources compared to Folding Factor 1. This is applicable only when I/O Mode is parallel, as the Folding Factor gets decided automatically in serial I/O mode.	fold_fac
Use Clock Enable	Enable/Disable	Adds or Removes Clock Enable port from the core. Default is Disable (no Clock Enable port).	c_has_ce
Use Block RAM/Use Distributed RAM	Radio Button	Available only in matrix-matrix multiplication mode, where user has a choice between Block RAM and Distributed RAM for internal data storage. Default option is use Distributed RAM.	ram_type

Core Symbol and Port Definitions

The block diagram in Table 4 provides the input/output signals for the core. Signals are categorized into Control Signals (clock, reset and clock enable), Port A Signals, Port B Signals and Port C Signals. Table 3 provides detailed information for the signals.

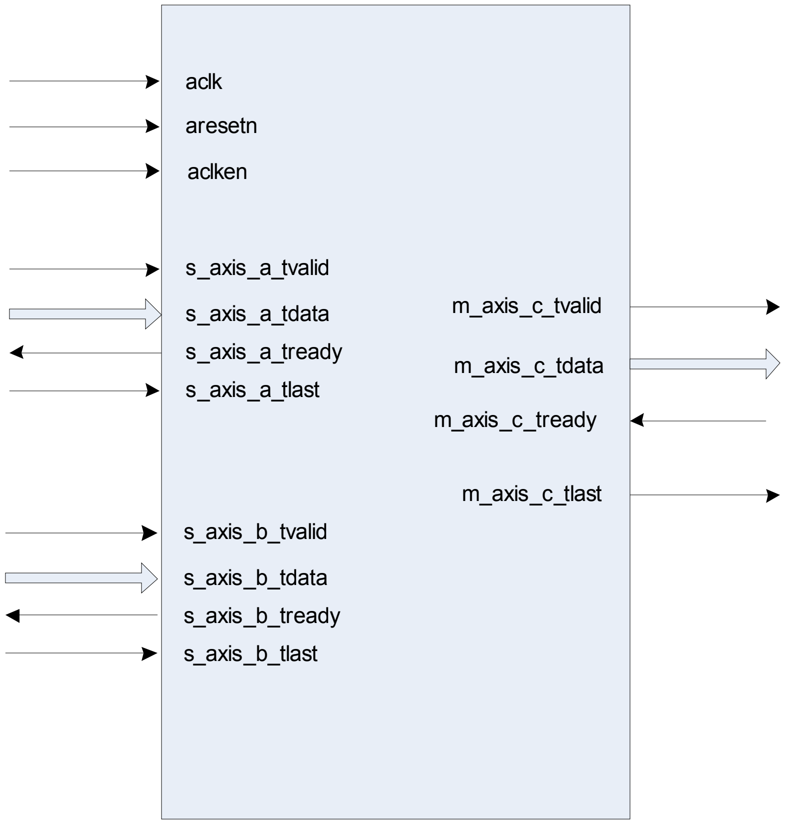


Figure 4: **LAT IP Core Block Diagram**

Table 3: **LAT Interface Signals**

Signal	Bit Width	I/O	Description
Control Signals			
ACLK	1	I	Global clock for the AXI4 interface. The same clock is used internal to the core as single clock domain fully synchronous design. The whole design operates only at the positive edge of this clock.
ARESETN	1	I	Active-low synchronous reset. When ARESETN is asserted (ARESETN = 0), the interface is forced into the reset state irrespective of ACLKEN. When ARESETN is de-asserted, the interface comes out of reset state. ARESETN is synchronous to ACLK. ARESETN has to be asserted for at least 4 clock cycles for the core to get reset. ***Note:*** ARESETN takes precedence over ACLKEN.
ACLKEN	1	I	Clock enable for the IP. This is an optional signal. ACLKEN is active-high and is synchronous to ACLK. All activity is suspended when ACLKEN is de-asserted. This signal to be used for flow control for the core, which is designed for real time streaming.

Table 3: **LAT Interface Signals**

Port A Signals (for input Matrix A)			
S_AXIS_A_TVALID	1	I	AXI4 data valid signal for port A. Active high signal synchronous to ACLK, to be held high for $K_A$contiguous cycles by external master.
S_AXIS_A_TDATA	[0: $ABUS_w$ - 1]	I	Data port for Matrix A (max width 1024 bits). Data comes synchronously w.r.t ACLK over K_A contiguous cycles.
S_AXIS_A_TREADY	1	O	Core ready signal for port A, synchronous to ACLK, stays asserted for K_A contiguous cycles and gets deasserted for F-K_A cycles. Active high signal.
S_AXIS_A_TLAST	1	I	Indicates the last cycle of K_A contiguous cycles, synchronous to ACLK, gets asserted for the last cycle of K_A contiguous cycle. Active high signal.
Port B Signals (for input Matrix B)			
S_AXIS_B_TVALID	1	I	AXI4 data valid signal for port B. Active high signal synchronous to ACLK, to be held high for K_B contiguous cycles by external master.
S_AXIS_B_TDATA	[0: $BBUS_w$-1]	I	Data port for Matrix B or scalar b (max width 1024 bits). Data comes synchronously w.r.t ACLK over K_B contiguous cycles.
S_AXIS_B_TREADY	1	O	Core ready signal for port B, synchronous to ACLK, stays asserted for K_B contiguous cycles and gets deasserted for F-K_B cycles. Active high signal.
S_AXIS_B_TLAST	1	I	Indicates the last cycle of K_B contiguous cycles, synchronous to ACLK, gets asserted for the last cycle of K_B contiguous cycles. Active high signal.
Port C Signals (for output Matrix C)			
M_AXIS_C_TVALID	1	O	AXI4 data valid signal for port C. Active high signal synchronous to ACLK, held high for K_C contiguous cycles by the core.
M_AXIS_C_TDATA	[0: $CBUS_w$-1]	O	Data port for Matrix C (max width 1024 bits). Data comes out synchronously w.r.t ACLK over K_C contiguous cycles from the core after initial delay of T_D cycles.
M_AXIS_C_TREADY	1	I	AXI4 streaming ready signal for port C from external slave, synchronous to ACLK, must stay asserted for K_C (at least) contiguous cycles from the beginning of M_AXIS_C_TVALID assertion. Active high signal.
M_AXIS_C_TLAST	1	O	Indicates the last cycle of K_C contiguous cycles, synchronous to ACLK, gets asserted for the last cycle of K_C contiguous cycles. Active high signal.

Using the LAT Core

The CORE Generator software GUI performs error-checking on all input parameters. Resource estimation and implementation details 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 Model

The core has a number of options for simulation models:

- VHDL UniSim structural model
- Verilog UniSim structural model

The models required may be selected in the CORE Generator tool 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 models might produce incorrect results if simulation with a resolution other than 1 ps. See the "Register Transfer Level (RTL) Simulation Using Xilinx Libraries" section in Synthesis and Simulation Design Guide for more information. This document is part of the ISE Software Manuals set available at:

www.xilinx.com/support/documentation/dt_ise.htm

Demonstration Test Bench

When the core is generated using the CORE Generator tool, a demonstration test bench is created. This is a simple VHDL test bench that exercises the core. The demonstration test bench source code is a VHDL file: `demo_tb/tb_`*`component_name`*`.vhd` in the CORE Generator software output directory. The source code is comprehensively commented.

Using the Demonstration Test Bench

The demonstration test bench instantiates the structural model for the customized LAT v1.0 core. The netlist can be simulated within the demonstration test bench. The VHDL netlist *`component_name`*`.vhd` is generated by CORE Generator. If the option to generate the structural model was not set, generate a netlist using the NetGen program. For example, in Unix:

```
netgen -sim -ofmt vhdl component_name.ngc component_name.vhd
```

Compile the netlist into the work library (see your simulator documentation for more information). Compile the demonstration test bench into the work library. Then simulate the demonstration test bench. View the test bench signals in your simulator's waveform viewer to see the operations of the test bench.

Demonstration Test Bench Detailed Information

The demonstration test bench performs the following tasks:

- Instantiate the core
- Generate a clock signal
- Drive the core's input signals to demonstrate core features
  - Reset behavior
  - Clock Enable support
  - AXI4-Streaming support

More information on the demonstration test bench is available in the `TB_README.txt` file that is generated as part of the CORE Generator output.

AXI4-Stream Considerations

Conversion to AXI4-Stream interfaces brings standardization and enhances interoperability of Xilinx IP LogiCore solutions. Other than general control signals such as `ALCK`, `ACLKEN` and `ARESETN`, all inputs and outputs to the LAT core are conveyed via AXI4-Stream channels. A channel consists of `TVALID` and `TDATA` always, plus several optional ports and fields. In the LAT core, the optional ports supported are `TREADY`, `TLAST`. Together, `TVALID` and `TREADY` perform a handshake to transfer a message, where the payload is `TDATA` and `TLAST`. The LAT core operates on the operands contained in the `TDATA` fields (Port A and Port B) and outputs the result in the `TDATA` field (Port C) of the output channel. This facility is expected to ease use of the LAT core in a system. The field `TLAST` indicates last cycle of packetized data, asserted by the AXI4-Streaming master for the last cycle. Note, this is not a mandatory requirement for the core and can be left connected to LOW (for ports A and B) if not used (or available with upstream master).

For further details on AXI4-Stream interfaces, see the Xilinx AXI Reference Guide (UG761) and the AMBA® 4 AXI4-Stream Protocol Version: 1.0 Specification.

Basic Handshake

Figure 5 shows the transfer of data in an AXI4-Stream channel. `TVALID` is driven by the source (master) side of the channel and `TREADY` is driven by the receiver (slave). `TVALID` indicates that the value in the payload fields (`TDATA` and `TLAST`) is valid. `TREADY` indicates that the slave is ready to receive data. When both `TVALID` and `TREADY` are true (asserted) in a cycle, a transfer occurs. The master and slave will set `TVALID` and `TREADY` respectively for the next transfer appropriately. Reset needs to be asserted at least for four clock cycles. Note that since this is a streaming core, it does not need reset once it is done after powering up. In matrix-matrix multiplication mode, resetting the core also clears the pipeline (that is, matrices in the output pipeline will be cleared). Avoid resetting the core in the middle of a transfer (as this is a real-time core). Reset needs to be asserted at least for F cycles (only for matrix-matrix multiplication mode) if the core has to be reset during a transfer. As shown in the following timing diagram, the clock enable signal must stay asserted when the reset is issued.

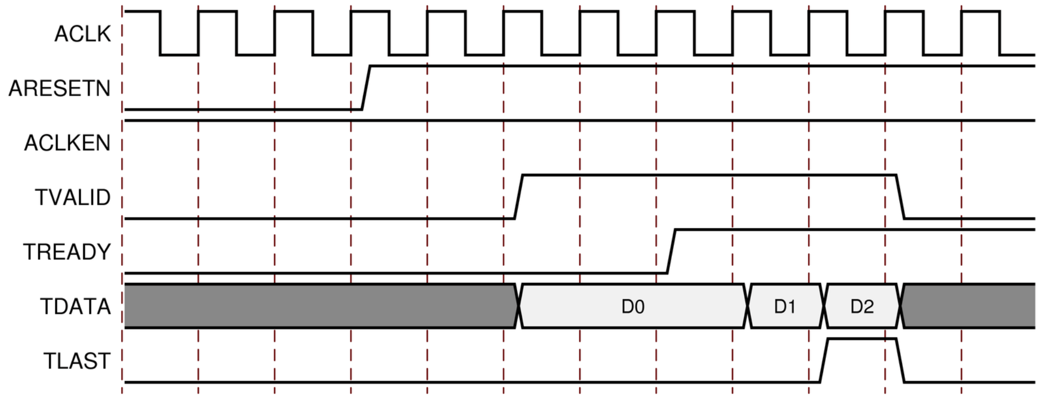


Figure 5: **Data Transfer in an AXI4-Stream Channel**

TDATA Packing

The TDATA field of the AXI4-Stream interface carries packed data. In complex data types, the real component is in the least significant position and the imaginary component follows, each are separate subfields. To ease interoperability with byte-oriented protocols, each subfield within TDATA that could be used independently is first extended to fit a bit field that is a multiple of 8 bits. For example, say the LAT core is configured to have an A operand width of 11 bits. Each of the real and imaginary components of A is 11 bits wide. The real component would occupy bits 10 down to 0. Bits 15 down to 11 would be ignored. Bits 26 down to 16 would hold the imaginary component and bits 31 down to 27 would also be ignored. Sub-field packing and byte-rounding are shown in Figure 6. Note that the bits added by byte orientation are ignored by the core and do not result in additional resource use.

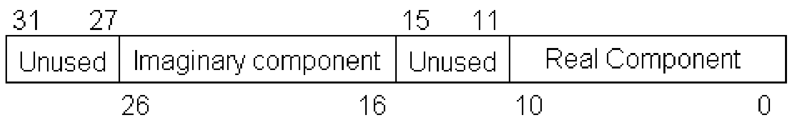


Figure 6: **TDATA Packing for A, B and C Ports**

TDATA Structure for A, B and C Ports

Input ports A, B, and output port C carry packed and byte-rounded data in their respective TDATA fields. In parallel I/O mode, matrix elements (after byte-rounding and real-imaginary packing for complex data) are also packed row-wise in the TDATA field. Consequently, for a 4x5 matrix fully packed, TDATA looks ([MSP to LSP]) as follows:

$$[A_{4x5}(2,3)\ A_{4x5}(2,2)\ A_{4x5}(2,1)\ A_{4x5}(1,5)\ A_{4x5}(1,4)\ A_{4x5}(1,3)\ A_{4x5}(1,2)\ A_{4x5}(1,1)]$$

This packing is subjected to the condition that the maximum width support for TDATA is 1024 for this core.

Control, Interface, and Timing

The LAT v1.0 core has fully AXI4-Streaming compliant interfaces for inputs and output. The core has two AXI4 slave interfaces for inputs (port A and port B) and one master interface (port C) for the output; which must be connected to external masters (for ports A and B) and slave (for port C), respectively. This core is designed for real time streaming data with the assumption that data is continuous without any interruptions once initiated. However, flow control can be achieved by the clock enable signal (ACLKEN), which pauses the core when deasserted externally (by a master).

Figure 7 shows the timing diagram for control and port A, B, C signals for matrix addition operation. As shown in the diagram, matrices A and B are input over data buses S_AXIS_A_TDATA and S_AXIS_B_TDATA and corresponding S_AXIS_A_TVALID and S_AXIS_B_TVALID signals are held High by the external AXI4-Streaming master for K_A and K_B cycles (both are equal in the addition case). Corresponding ready signals (S_AXIS_A_TREADY and S_AXIS_B_TREADY) from the core stay asserted as the core is always ready for external inputs as $K_A = K_B = F$ in this case. Signals S_AXIS_A_TLAST and S_AXIS_B_TLAST are asserted by the external masters for the last cycle of input data, indicating the end of the current matrix. The output matrix exits from the core over the M_AXIS_C_TDATA bus after a delay of T_D cycles. The signal M_AXIS_C_TVALID is asserted by the core indicating valid data cycles over the data bus lasting for K_C cycles. The external slave for outputs keeps the M_AXIS_C_TREADY signal asserted for receiving real-time streaming data from the core. The core asserts the M_AXIS_C_TLAST signal for the last cycle of K_C cycles indicating the last cycle of the current transfer of the output matrix.

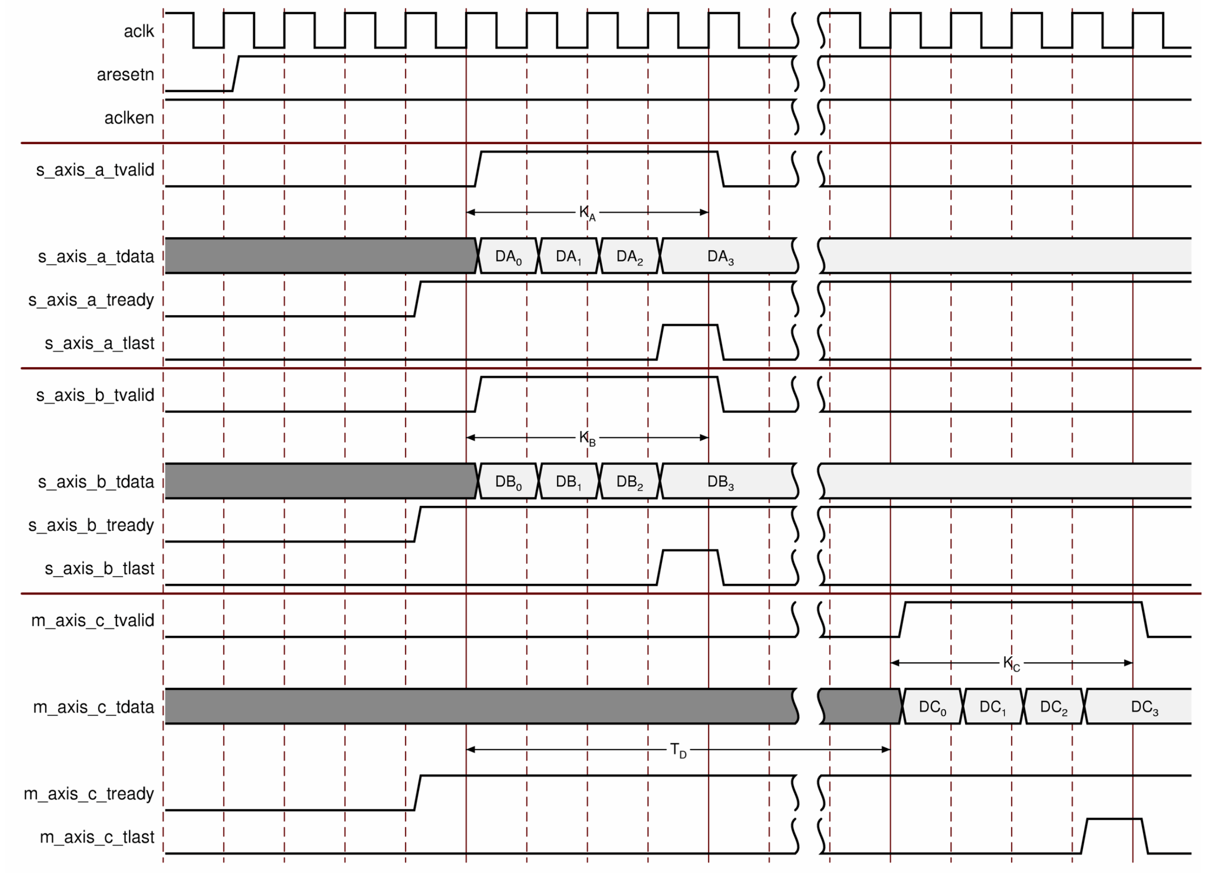


Figure 7: **Timing Diagram of Matrix Addition Mode**

Figure 8 shows continuous real-time streaming of two matrices in matrix-scalar multiplication mode. In this example, $K_A = K_C = 2$, $K_B = 1$ and $F = 3$. On port A, the signal `S_AXIS_A_TVALID` stays asserted for K_A and then goes down for $F - K_A$ cycles before the next matrix is fed in. Similarly, for scalar b, `S_AXIS_A_TVALID` goes down for $F - K_B$ cycles before the next scalar is made available. Corresponding ready signals `S_AXIS_A_TREADY` and `S_AXIS_B_TREADY` show similar behavior indicating the core is busy processing the current data. The signal `S_AXIS_B_TLAST` gets asserted for a single cycle like corresponding valid signal as K_B=1. On the output port C matrices come out one in every F cycles where `M_AXIS_C_TVALID` has a cadence of K_C cycles of high and $F - K_C$ cycles of low.

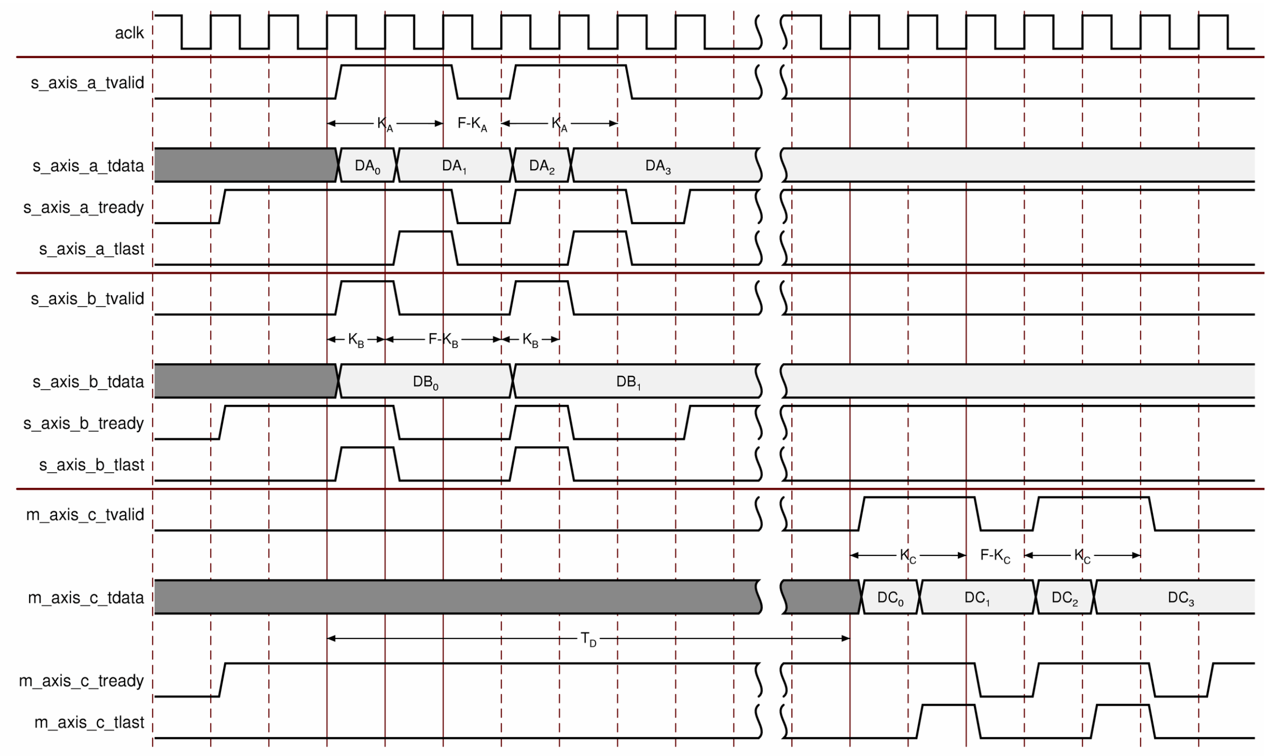


Figure 8: **Timing Diagram of Matrix-Scalar Multiplication Mode**

Continuous real-time streaming of two matrices is shown in Figure 9 for matrix-matrix multiplication mode. $K_A = K_C = 3$, $K_B = 4$ and $F = 5$ in this case. Signals `S_AXIS_A_TVALID` and `S_AXIS_B_TVALID` stay asserted for K_A and K_B cycles respectively. On port C, output matrices come out after initial delay T_D maintaining the streaming rate of one matrix in every F cycles as in other functional modes.

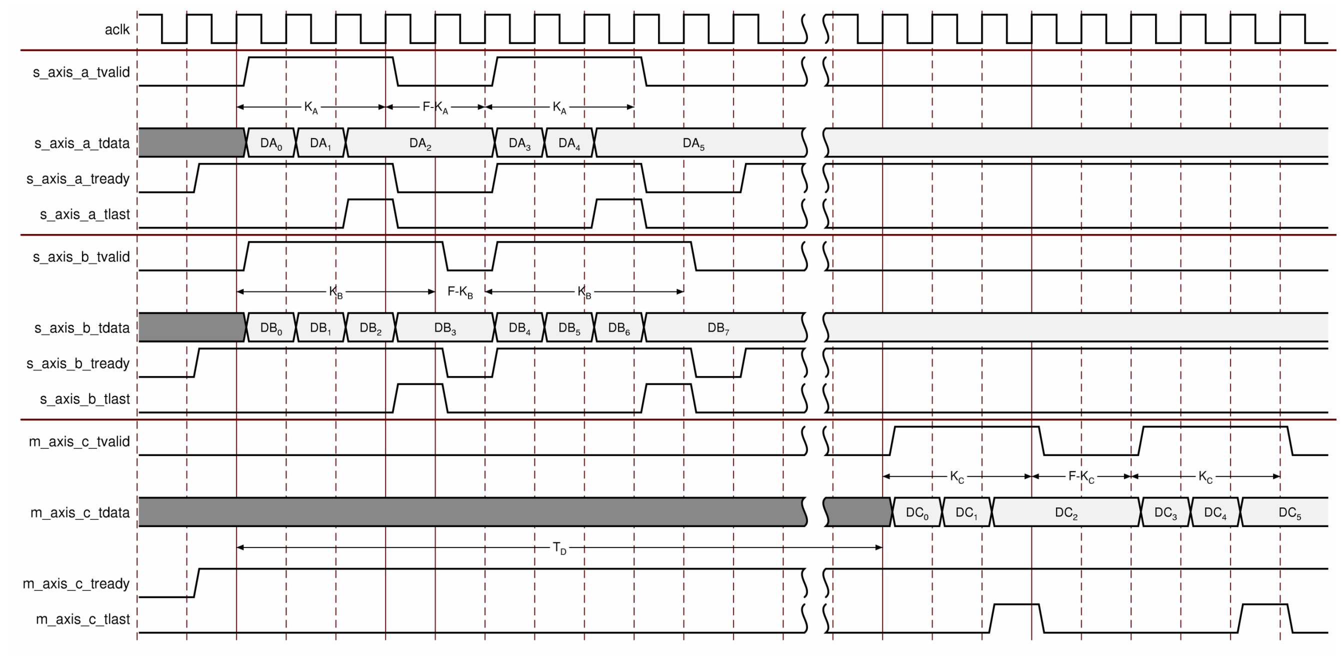


Figure 9: **Timing Diagram of Matrix-Matrix Multiplication Mode**

Figure 10 shows pause-and-resume mode of the core using clock enable signal. De-assertion of the signal ACLKEN freezes the core pausing both data and control paths. All the input signals (both control and data) are don't care when the core is paused. Output valid signal M_AXIS_C_TVALID is de-asserted if in the middle of a transfer when the core is paused. As this is a fully synchronous design, output pauses after a delay of T_{CE} cycles from the de-assertion of clock enable signal. The transfer resumes (after the same delay of T_{CE}) when the clock enable signal is transitioned to the asserted state. T_{CE} is 1 clock cycles for addition/subtraction and scalar multiplication modes and for matrix-matrix multiplication this is 2 clock cycles. This feature can be used when data comes at a decimated rate.

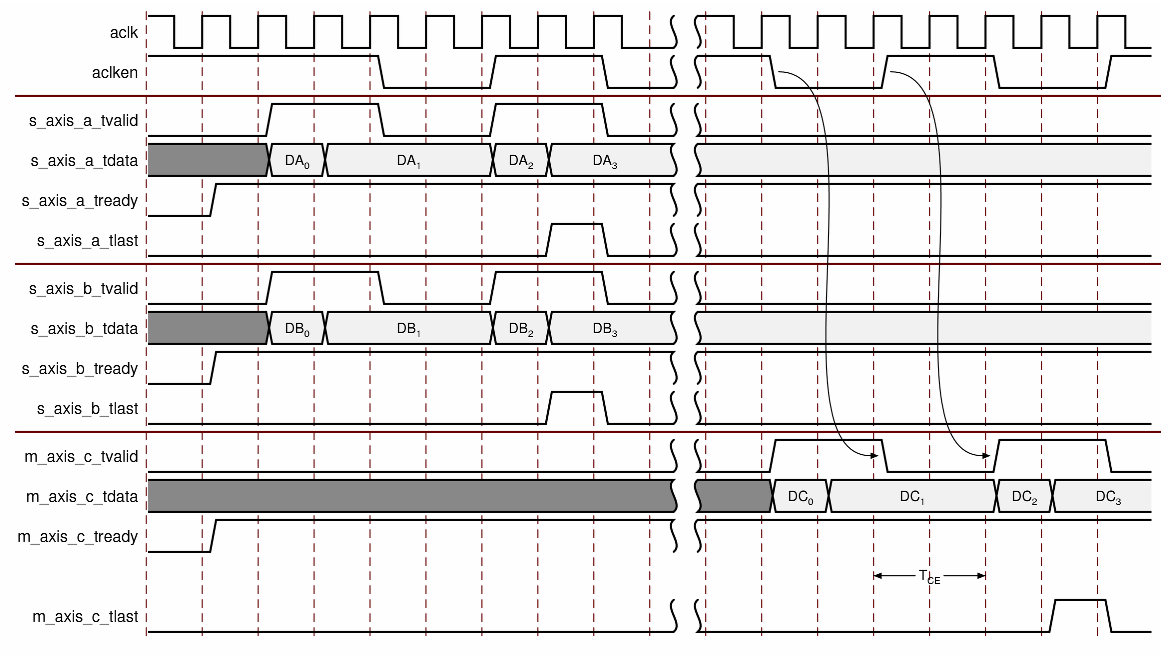


Figure 10: **Interface Timing Diagram With Pause-and-Resume**

Latency, Performance, and Resource Utilization

This section describes latency, performance and resource utilization of the core in various modes and configurations. Since the LAT core is fully real-time streaming capable, the streaming latency (minimum lag in clock cycles between two successive matrices) is always the same as the folding factor F cycles. However, initial latency of the core is a function of various core parameters.

Latency

Let T_D (in clock cycles) denote the total first-in-first-out (initial latency) delay of the core. This is the latency from the first data input to the core to the first output observed at the output of the core. The initial latency T_D has two components, one, delay (in clock cycles) T_{core_delay} of the functional-mode core itself (based on the choice of M, N and L) and the other, a fixed part (based on routing and registering stages of the core) T_f. Thus, total latency can be expressed as:

$$T_D = T_{core\_delay} + T_f \qquad \text{Equation 16}$$

Functional mode-wise break-up of initial latency is given below:

- Matrix-Matrix Addition/Subtraction

  In this mode, core delay is constant and does not scale with matrix dimension, T_{core_delay}=2 and the fixed delay T_f is between [0,4] based on implementation.
- Matrix-Scalar Multiplication

  For this op-mode, the core delay parameter is a function of the number of elements of the A matrix input per cycle (NE_A) (as shown in the following table) and the fixed delay T_f is between [0,4] as in the case of addition/subtraction.

Table 4: **Matrix-Scalar Mode Core Delay**

NE_A	T_{core_delay}
≤ 4	4 + 2 x CMPLX
≤ 8	5 + 2 x CMPLX
≤ 16	6 + 2 x CMPLX
≤ 32	7 + 2 x CMPLX
> 32	8 + 2 x CMPLX

- Matrix-Matrix Multiplication

  In this op-mode, the folding factor directly affects T_{core_delay} since the core waits for the input matrices to get filled up completely before starting to process them. Consequently, for parallel I/O:

$$T_{core\_delay} = F + (CMPLX + 1) \times L + N + ceil\left(\frac{M}{F}\right) \qquad \textit{Equation 17}$$

  and for the serial I/O case, this equation is simplified to:

$$T_{core\_delay} = F + (CMPLX + 1) \times L \qquad \textit{Equation 18}$$

  The maximum value of T_f is 11 and 17 clock cycles in parallel and serial modes, respectively.

Timing Performance and Resource Usage

This section provides data on the timing performance and resource utilization of the core. Performance has been obtained on one representative device from the Virtex-6, Virtex-7, and Kintex-7 families of FPGAs. The following table lists the devices used for characterization.

Table 5: **Devices Used For Characterization**

Virtex-6	Virtex-7	Kintex-7
xc6vlx75t	xc7v285t	xc7k70t

All devices were used with speed grade -1.

General Notes on Characterization

While obtaining the maximum frequency numbers for different configurations and op-modes for the core, the general guideline was to use default settings of XST, MAP, and PAR tools for almost all of the tool options to make the results independent of tool versions. The key required setting was specifying high effort levels for MAP and PAR. This is typically a standard practice used in the ISE flow when trying to achieve best timing performance. The tool and speed file version is provided in the following table.

Table 6: **Tool Options For Core Performance**

Tool	Options
XST	Default
MAP	-w -logic_opt off -ol high -t 1 –xe n -xt 0 -register_duplication off -r 4 -global_opt off -mt off -ir off -pr off -lc off -power off
PAR	-w -intstyle ise -ol high –xe n-mt off

The maximum achievable clock frequency and the resource counts may be affected by other tool options, additional logic in the FPGA device, using a different version of Xilinx tools, and other factors. Note that clock frequency does not take clock jitter into account and should be derated by an amount appropriate to the clock source jitter specification. Relatively higher matrix dimensions with low folding factors or high values of I/O precision (bit widths) results in increased routing congestion especially for the case of matrix-matrix multiplication. This may result in poorer maximum achievable frequency.

Rounding Modes

Characterization was performed by configuring the core to the default "convergent rounding to even" mode.

Memory Usage

FIFOs are used in matrix-matrix multiplication mode, which can either be mapped to Block RAM or Distributed RAM. Selection of Distributed RAM mode realizes FIFOs using LUT/FFs on the FPGA fabric and gives better maximum frequency, while choice of Block RAM provides better resource utilization at the cost of some reduction in maximum achievable frequency.

Table 7: **Virtex-6 Maximum Frequency and Resource Utilization, Device: xc6vlx75t-ff484-1, Speed Version: PRODUCTION 1.11d 2010-12-06**

OPMODE	Configuration								DSP48E1 Slices	Block RAMs	LUT-FF Pairs	Fmax (MHZ)	Fmax (without clock enable)
	M	L	N	Folding Factor	Data Type	A Width	B Width	C Width					
Matrix-Matrix Addition (DSP48E1 based)	6	NA	5	1	Complex	16	16	16	30	0	4,213	471	471[2]
	8	NA	5	2	Complex	24	24	24	20	0	4,256	471	471[2]
	8	NA	10	4	Complex	24	24	24	20	0	4,258	471	471
	8	NA	10	80[1]	Complex	24	24	24	1	0	225	471	471[2]
Matrix-Matrix Addition (LUT based)	6	NA	5	1	Complex	16	16	16	0	0	7,153	471	471
	8	NA	5	2	Complex	24	24	24	0	0	7,016	471	471
	8	NA	10	4	Complex	24	24	24	0	0	7,018	471	471
	8	NA	10	80[1]	Complex	24	24	24	0	0	363	471	471
Matrix-Scalar Multiplication	6	NA	5	1	Complex	16	16	16	120	0	8,785	471	471[2]
	8	NA	5	2	Complex	24	24	24	80	0	8,288	471	471[2]
	8	NA	10	4	Complex	24	24	24	80	0	8,289	471	471[2]
	8	NA	10	80[1]	Complex	24	24	24	4	0	324	471	471[2]
Matrix-Matrix Multiplication	8	4	1	1	Real	12	12	16	32	0	6,300	435	445
	8	8	1	2	Real	16	16	16	32	0	3,550	440	450
	4	4	4	4	Real	16	16	16	16	0	1,750	450	455
	8	8	8	64[1]	Real	16	16	16	8	0	2,500	455	455
	8	4	1	8	Complex	16	16	16	16	0	2,450	440	445
	4	4	4	4	Real	16	16	16	16	16	1,526	400	400[2]

[1] Serial I/O
[2] Component Switching Limit (471 MHz for DSP48E1 and 400 MHz for Block RAM in read-first mode)

Table 8: **Virtex-7 Maximum Frequency and Resource Utilization, Device: xc7v285t-ffg484-1, Speed Version: PREVIEW 0.74 2010-11-17**

OPMODE	Configuration								DSP48E1 Slices	Block RAMs	LUT-FF Pairs	Fmax (MHZ)	Fmax (without clock enable)
	M	L	N	Folding Factor	Data Type	A Width	B Width	C Width					
Matrix-Matrix Addition (DSP48E1 based)	6	NA	5	1	Complex	16	16	16	30	0	4,213	417	417
	8	NA	5	2	Complex	24	24	24	20	0	4,256	417	417
	8	NA	10	4	Complex	24	24	24	20	0	4,258	417	417
	8	NA	10	80[1]	Complex	24	24	24	1	0	225	417	417
Matrix-Matrix Addition (LUT based)	6	NA	5	1	Complex	16	16	16	0	0	7,153	417	417
	8	NA	5	2	Complex	24	24	24	0	0	7,016	410	410
	8	NA	10	4	Complex	24	24	24	0	0	7,018	417	417
	8	NA	10	80[1]	Complex	24	24	24	0	0	363	417	417
Matrix-Scalar Multiplication	2	NA	5	1	Complex	16	18	16	40	0	8,785	390	390
	4	NA	5	1	Complex	24	18	24	40	0	8,288	360	360
	8	NA	10	1	Complex	24	18	24	40	0	8,289	417	417
	8	NA	10	80[1]	Complex	24	18	24	4	0	324	417	417
Matrix-Matrix Multiplication	8	4	1	1	Real	16	16	16	32	0	6,300	380	380
	8	8	1	2	Real	16	16	16	32	0	3,550	375	375
	4	4	4	4	Real	16	16	16	16	0	1,750	380	380
	8	8	8	64[1]	Real	16	16	16	8	0	2,500	375	375
	8	4	1	8	Complex	16	16	16	16	0	2,450	380	360

[1] Serial I/O

Table 9: **Kintex-7 Maximum Frequency and Resource Utilization, Device: xc7k70t-fbg484-1, Speed Version: PREVIEW 0.52 2010-11-17**

OPMODE	Configuration								DSP48E1 Slices	Block RAMs	LUT-FF Pairs	Fmax (MHZ)	Fmax (without clock enable)
	M	L	N	Folding Factor	Data Type	A Width	B Width	C Width					
Matrix-Matrix Addition (DSP48E1 based)	6	NA	5	1	Complex	16	16	16	30	0	4,213	400	400
	8	NA	5	2	Complex	24	24	24	20	0	4,256	390	390
	8	NA	10	4	Complex	24	24	24	20	0	4,258	390	390
	8	NA	10	80[1]	Complex	24	24	24	1	0	225	400	400
Matrix-Matrix Addition (LUT based)	6	NA	5	1	Complex	16	16	16	0	0	7,153	417	417
	8	NA	5	2	Complex	24	24	24	0	0	7,016	400	400
	8	NA	10	4	Complex	24	24	24	0	0	7,018	417	417
	8	NA	10	80[1]	Complex	24	24	24	0	0	363	417	417
Matrix-Scalar Multiplication	2	NA	5	1	Complex	16	18	16	40	0	8,785	390	390
	4	NA	5	2	Complex	24	18	24	40	0	8,288	390	390
	8	NA	10	4	Complex	24	18	24	40	0	8,289	417	417
	8	NA	10	80[1]	Complex	24	18	24	4	0	324	417	417
Matrix-Matrix Multiplication	8	4	1	1	Real	16	16	16	32	0	6,300	375	375
	8	8	1	2	Real	16	16	16	32	0	3,550	375	375
	4	4	4	4	Real	16	16	16	16	0	1,750	400	400
	8	8	8	64[1]	Real	16	16	16	8	0	2,500	410	410
	8	4	1	8	Complex	16	16	16	16	0	2,450	380	360

[1] Serial I/O

Reference Documents

1. Xilinx AXI Reference Guide (UG761)
2. AMBA® 4 AXI4-Stream Protocol Version: 1.0 Specification

Support

Xilinx provides technical support for this LogiCORE IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

Refer to the IP Release Notes Guide (XTP025) for further information on this core. There will be a link to all the DSP IP and then to the relevant core being designed with. For each core, there is a master Answer Record that contains the Release Notes and Known Issues list for the core being used. The following information is listed for each version of the core:

- New Features
- Bug Fixes
- Known Issues

Ordering Information

This core may be downloaded from the Xilinx IP Center for use with the Xilinx CORE Generator software v13.1 and later. The Xilinx CORE Generator system is shipped with Xilinx ISE Design Suite development software.

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

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
03/01/2011	1.0	Initial Xilinx release.

Notice of Disclaimer