AccelWare IP cores provide a direct path to hardware implementation for complex MATLAB® toolbox and built-in functions. AccelWare cores deliver synthesizable, pre-verified DSP functions that enable true, top-down MATLAB architectural synthesis of FPGAs and ASICs. AccelWare IP includes Building Block, Advanced Math, Signal Processing and Communications toolkits.

Matrix Inverse (QR method)

The Matrix Inverse core computes the inverse of a real-valued, square input matrix. The implementation of the inverse computation is based on the triangular-orthogonal (QR) factorization of the input matrix followed by a product of the inverse of the matrix factors. (For applications where only QR factorization is required, see the data sheet for the **Matrix Factorization – QR method**).

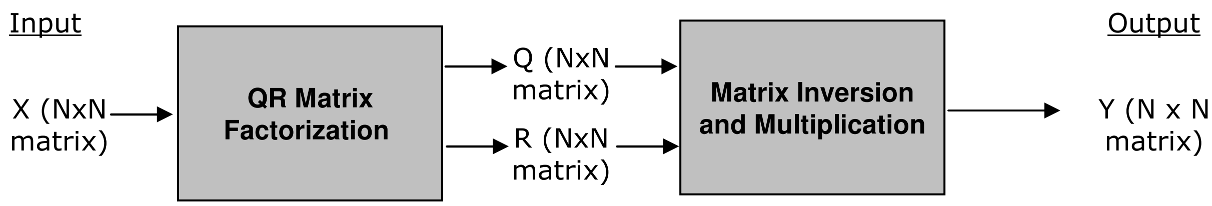


Figure 1: Matrix Inverse Block Diagram

The Matrix Inverse core uses a QR factorization algorithm based on Givens Rotations (GR) to produce the triangular (R) and orthogonal (Q) factors. The Givens Rotations are implemented in their *conventional* form using a COordinate Rotation DIgital Computer (CORDIC) for the vector rotations required to null elements below the diagonal and produce the upper triangular matrix R [1]. The resulting R and Q factors are then each inverted and multiplied to produce the output inverse matrix Y.

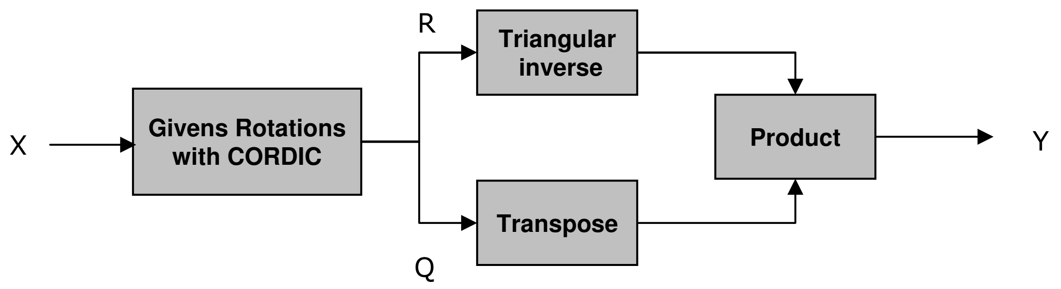


Figure 2: QR Factorization/Inversion Block Diagram

1] John G. Proakis et al. “Advanced Digital Signal Processing,” Macmillan Publishing Company, New York, New York, 1992.

Input

Signal Name	Signal Description	Type	Range
X	Input matrix to be inverted	Real	4 to 24 bits (fixed-point representation)

Output

Signal Name	Signal Description	Type	Range
Y	Output inverse matrix	Real	Up to 32 bits (fixed-point representation)

Implementation Parameters

Parameter Name	Parameter Description	Range
X quantizer	Input matrix quantization	Fixed-point precision, 2 to 24 bits of word-length
Implementation Algorithm	Algorithm used for factorization implementation	Conventional Givens Rotations
Input Data Type	Input matrix data type	Real
Matrix Size	Input matrix size (specifies number of rows and columns)	Integer in range 4 to 64
Input/Output Type	Input/output matrix dimension representation	1-D or 2-D
Output Precision	Number of bits for output matrix	Auto – sets value based on input matrix word-length

Input Matrix Quantization

The number representation of the input is defined by the input matrix quantization. The *qr_inverse* accepts real-valued, fixed-point input data with quantization parameters defined by this quantization.

Implementation Algorithm

The implementation algorithm used for matrix factorization is based on conventional Givens Rotations with CORDIC.

Input Data Type

The input matrix must be real-valued.

Matrix Size

The matrix size defines the number of rows and columns of the input and output matrices handled by the *qr_inverse*.

Input/Output Data Type

The *qr_inverse* model can be generated to accept input and generate output matrices as 1-D or 2-D arrays.

Output Precision

The parameters that define the numerical precision of the inverse output matrix are automatically computed during the generation of the AccelWare *qr_inverse* core. The output precision can also be affected during synthesis of the *qr_inverse* model with the AccelChip DSP Synthesis tool by setting quantization directives. This allows great flexibility to the user to explore numerical precision and hardware implementation area/speed tradeoffs.

Hardware Interfacing

A synthesizable AccelWare MATLAB model will typically be a design module that is part of a larger design on a chip. The flow of data into and out of the hardware ports is controlled by a protocol called DAP (Data Accept Protocol). Synthesizing the *qr_inverse* model with AccelChip in a stand-alone fashion will produce a Matrix Inverse hardware block with DAP interface signals ready for integration into a larger system. The following gives a description of DAP interface protocol.

Global Signals

The hardware module has one Clock input and one global Reset. Data transfers on each port are synchronized to the Clock. The global Reset returns all registers and flip-flops to a known state.

Input Synchronization Signals

ND (NewData) -This signal is controlled by the external design and indicates that data on the input data bus is valid. This causes the receiving device (the hardware module) to capture the data on the rising edge of the next clock cycle.

RFND (ReadyForNewData) -This signal is controlled by the hardware module and indicates that the module is ready to capture new data from the input bus. When the module sets RFND low, the external design should immediately stop sending new data. If the hardware module holds RFND constantly high, then new data will be captured on every clock cycle provided the sending device can send data that fast.

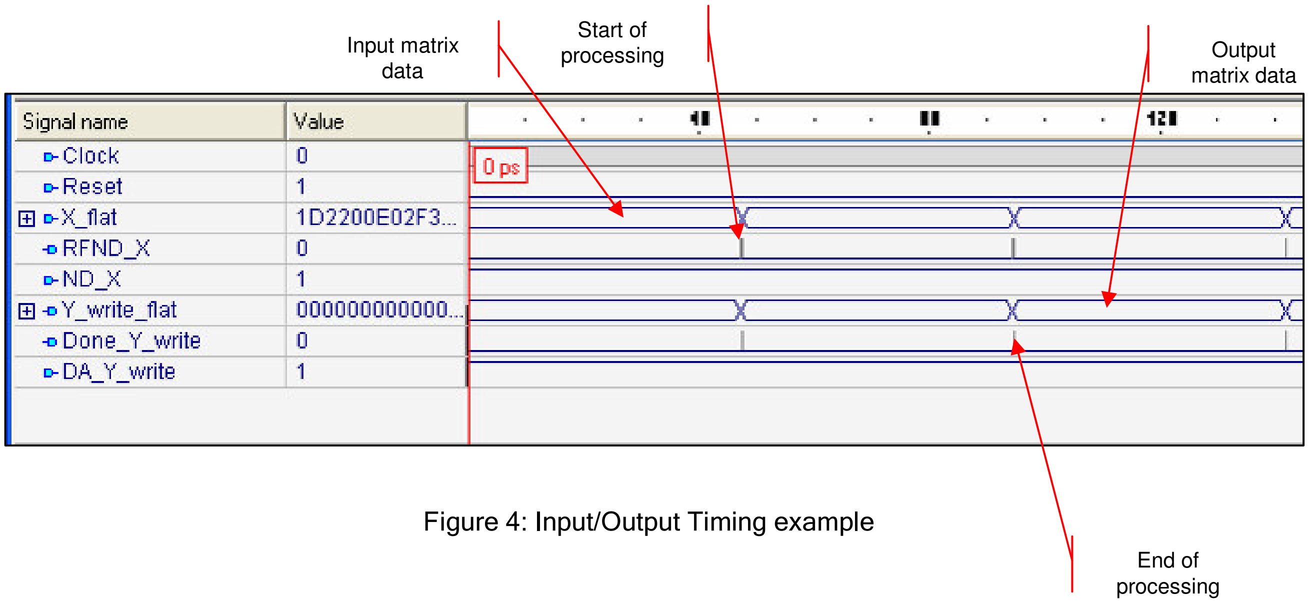
Output Synchronization Signals

Done -This signal is controlled by the hardware module and indicates that data on the output data bus is valid. Once Done is set high, it will remain high until the receiving device acknowledges the data capture by setting DA high. If the external design holds DA constantly high, then the hardware module will send data at the maximum possible rate, as governed by the module clock frequency and the latency of the computing algorithm.

DA (Data Accept) - This signal is controlled by the external design and indicates that the data on the output bus has been captured. If the external design holds DA constantly high, then the hardware module will send data out at the maximum rate possible, as governed by the module clock frequency and the latency of the computing algorithm.

Signal	**Direction**	**Description**
Clock	Input	Clock input
Reset	Input	Reset input
X_flat	Input	Input matrix data
RFND_X	Output	Ready for new data
ND_X	Input	New input data valid
Y_write_flat	Output	Output matrix data
Done_Y_write	Output	Done indication
DA_Y_write	Input	Data accepted indication

Figure 3: DAP Signals in *qr_inverse*



Signal name	Value
Clock	0
Reset	1
X_flat	1D2200E02F3...
RFND_X	0
ND_X	1
Y_write_flat	000000000000...
Done_Y_write	0
DA_Y_write	1

Figure 4: Input/Output Timing example

Differences in Operation between AccelWare *qr_inverse()* and MATLAB *inv()*

The MATLAB *inv* function can operate on input matrices with complex data. The AccelWare *qr_inverse* can currently operate on real-valued input matrices only.

The MATLAB *inv* function generates an indication when the input matrix is badly conditioned. The AccelWare *qr_inverse* does not currently generate a bad conditioning indication.

Ordering Information

The AccelWare *qr_inverse* core is included in the AccelWare Advanced Math Toolkit (AccelChip part number **AWAMT**) and is provided as an option to the AccelChip DSP Synthesis product (AccelChip part number **ACDSP**).

For further information on availability, contact your local AccelChip sales representative or send email to sales@accelchip.com.

AccelChip Incorporated
1900 McCarthy Blvd., Suite. 204, Milpitas, CA 95035 phone (408) 943 0700 option 1 fax (408) 943 0661