Floating Point Processing using FPGAs

Michael Parker

Altera Corp

HotChips Conference

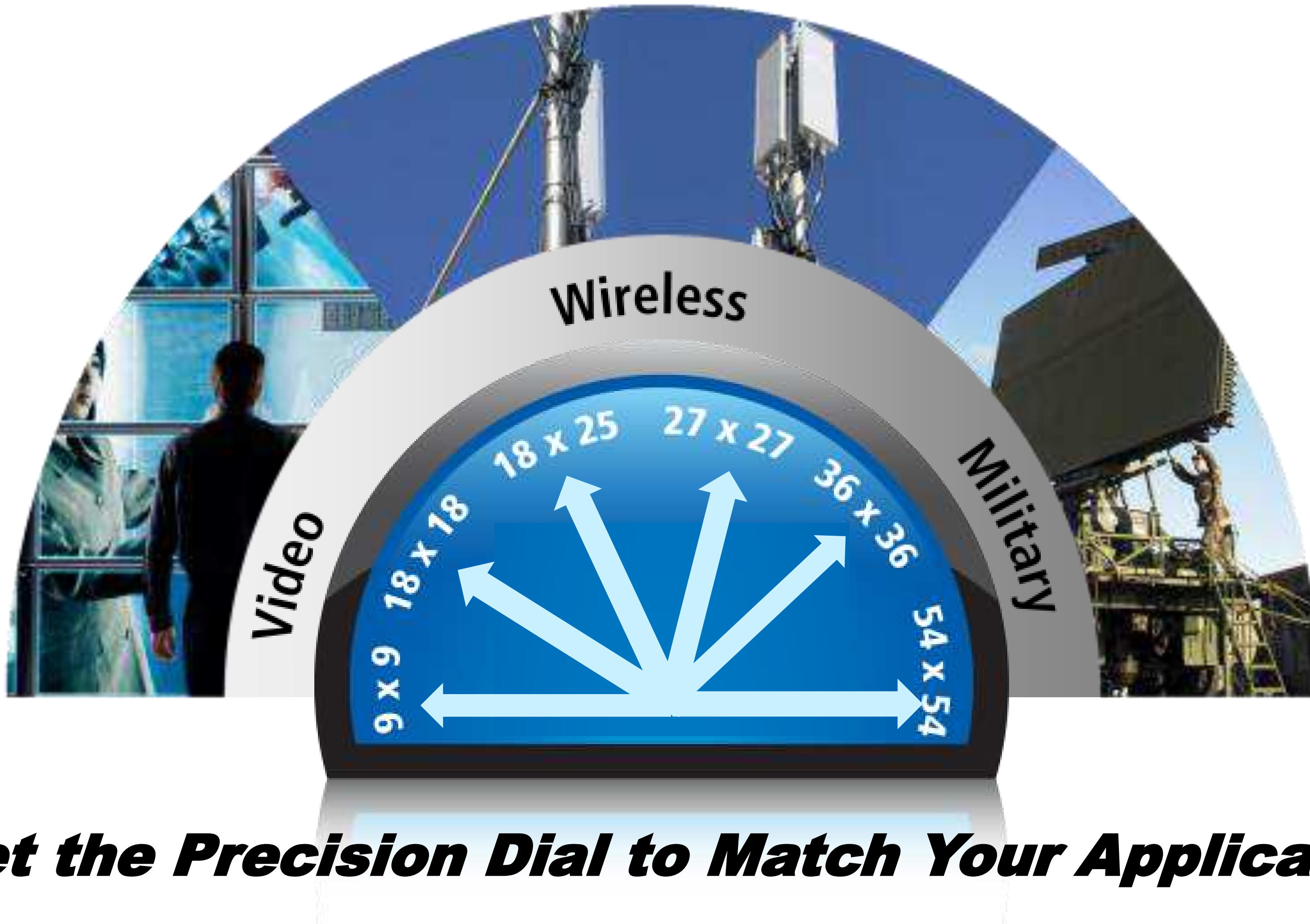
August 29, 2012

Agenda

- Stratix V FPGA architecture for Floating Point
- New Approach: “Fused Data Path”
- Throughput, GFLOPs, GFLOPs/W
  - FFT
  - Cholesky Decomposition
  - QR Decomposition
- Computational Accuracy
- Third Party Benchmarking

Stratix V architecture enhancements for floating point

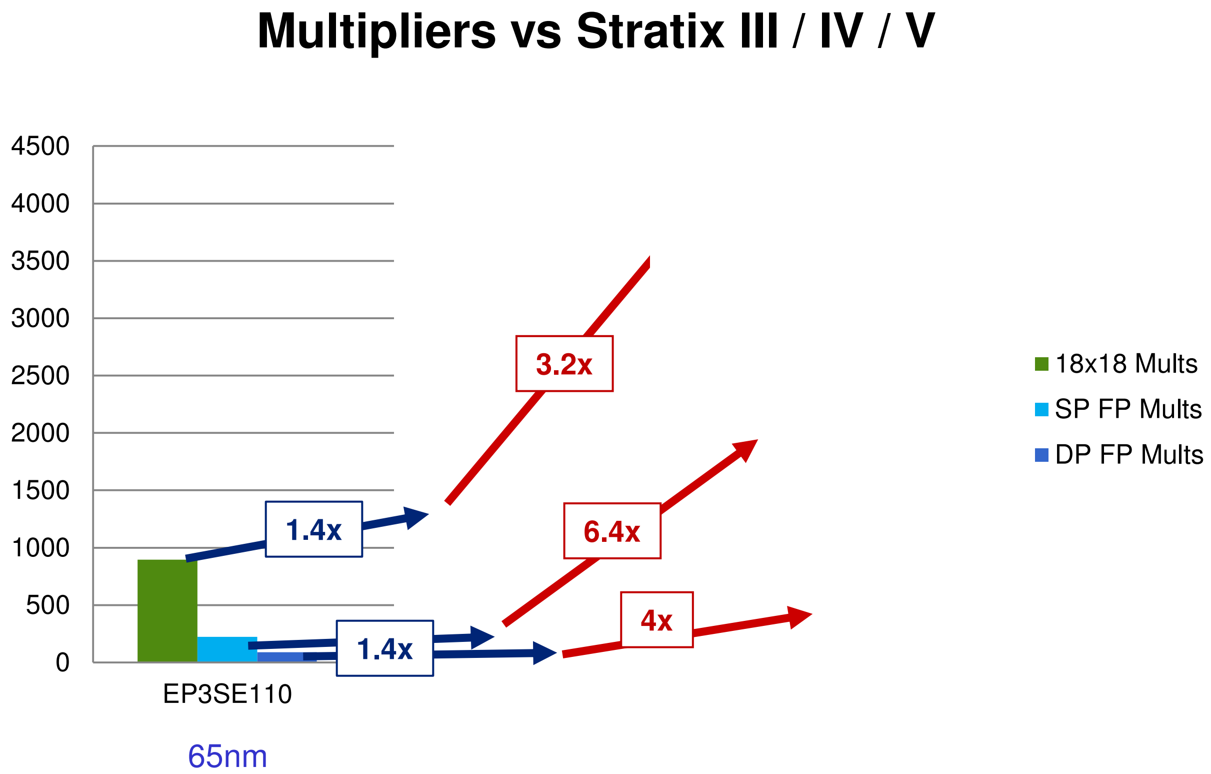
Altera's Variable-Precision DSP Block



Set the Precision Dial to Match Your Application

Why Floating Point at 28nm ?

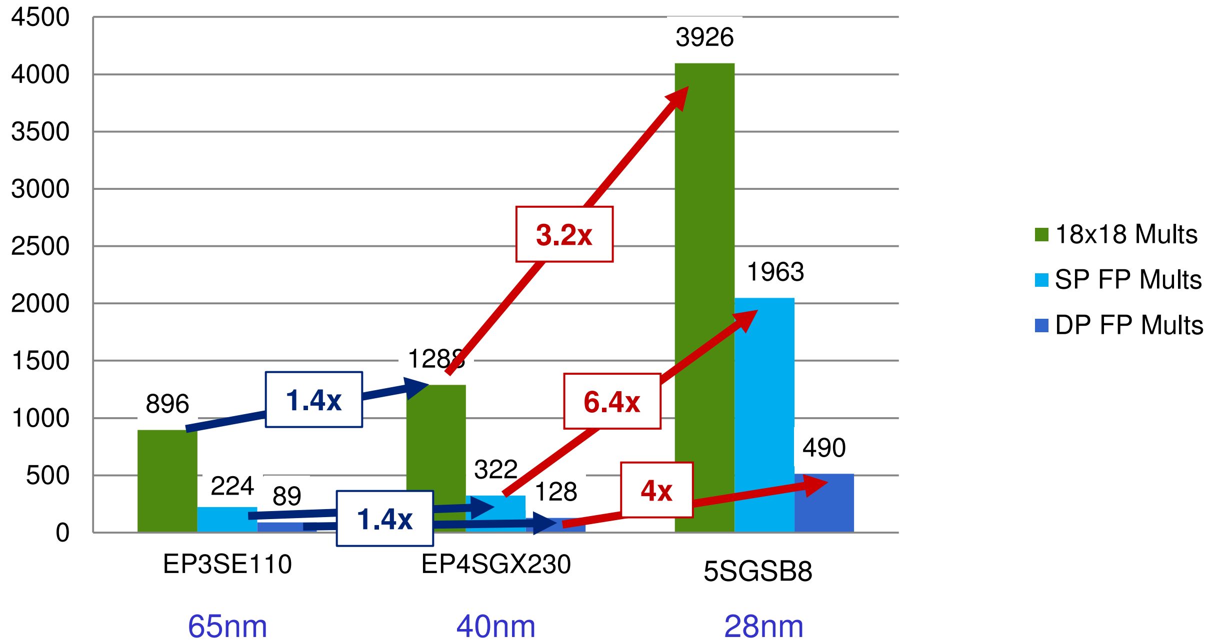
- Floating point density determined by hard multiplier density
- Multipliers must efficiently support floating point mantissa sizes



Floating Point Multiplier Capabilities

- Floating point density determined by hard multiplier density
- Multipliers must efficiently support floating point mantissa sizes

Multipliers vs Stratix III / IV / V



Introducing Fused Datapath

Allows High Performance Floating-Point in FPGAs

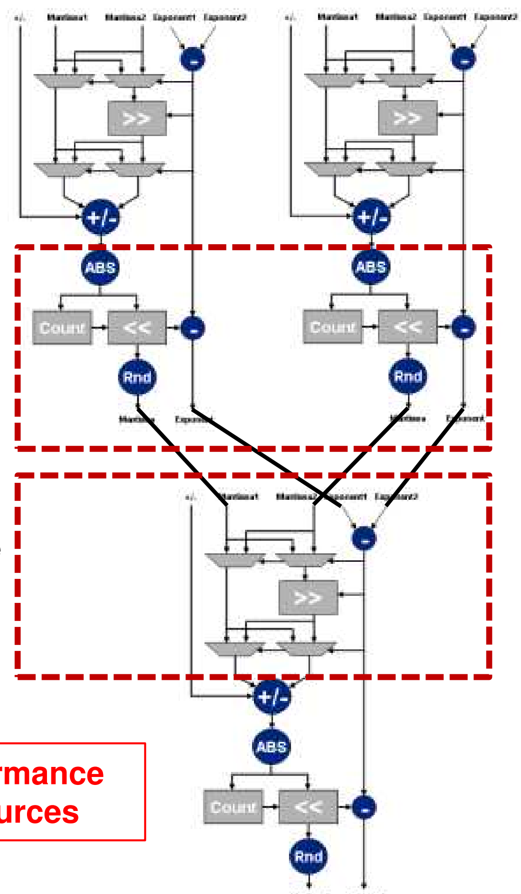
ALTERA®

New Floating-Point Implementation

Processor: Each Operation IEEE754

Denormalize

Normalize



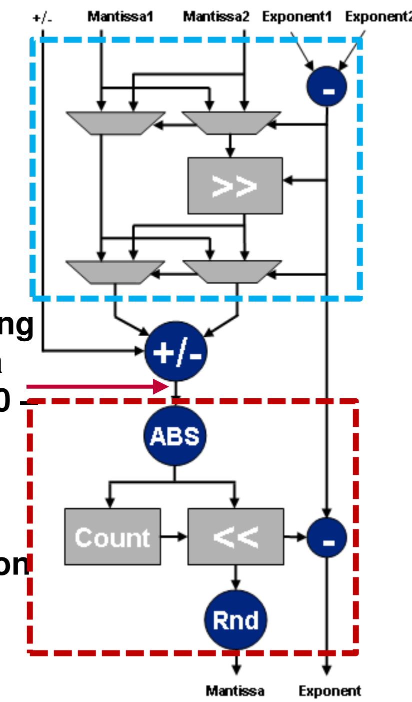
Low Performance
High Resources

Altera Floating Point: Fused Datapath

Slightly Larger – Wider Operands

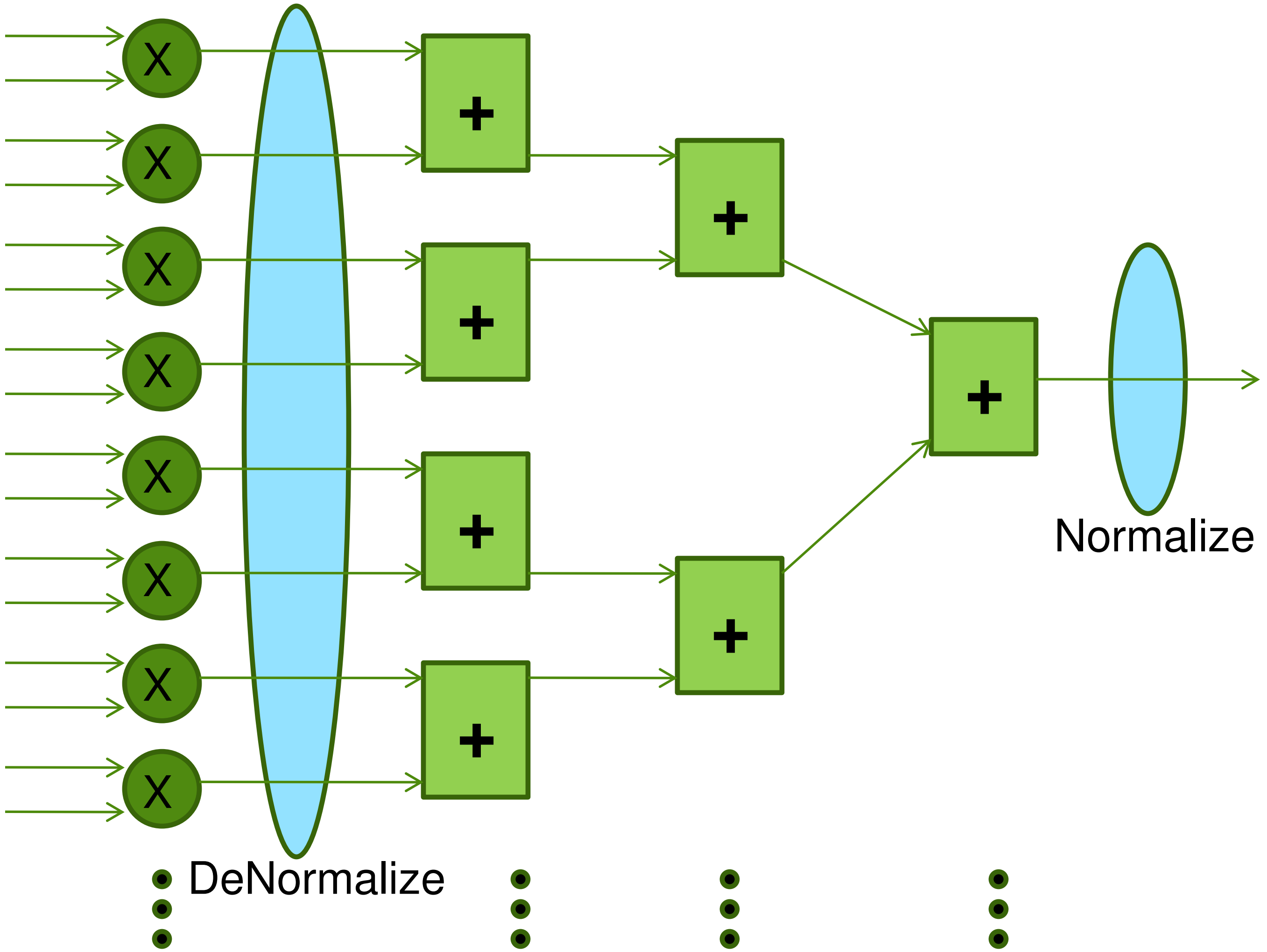
True Floating Mantissa (not just 1.0 - 1.99..)

Remove Normalization



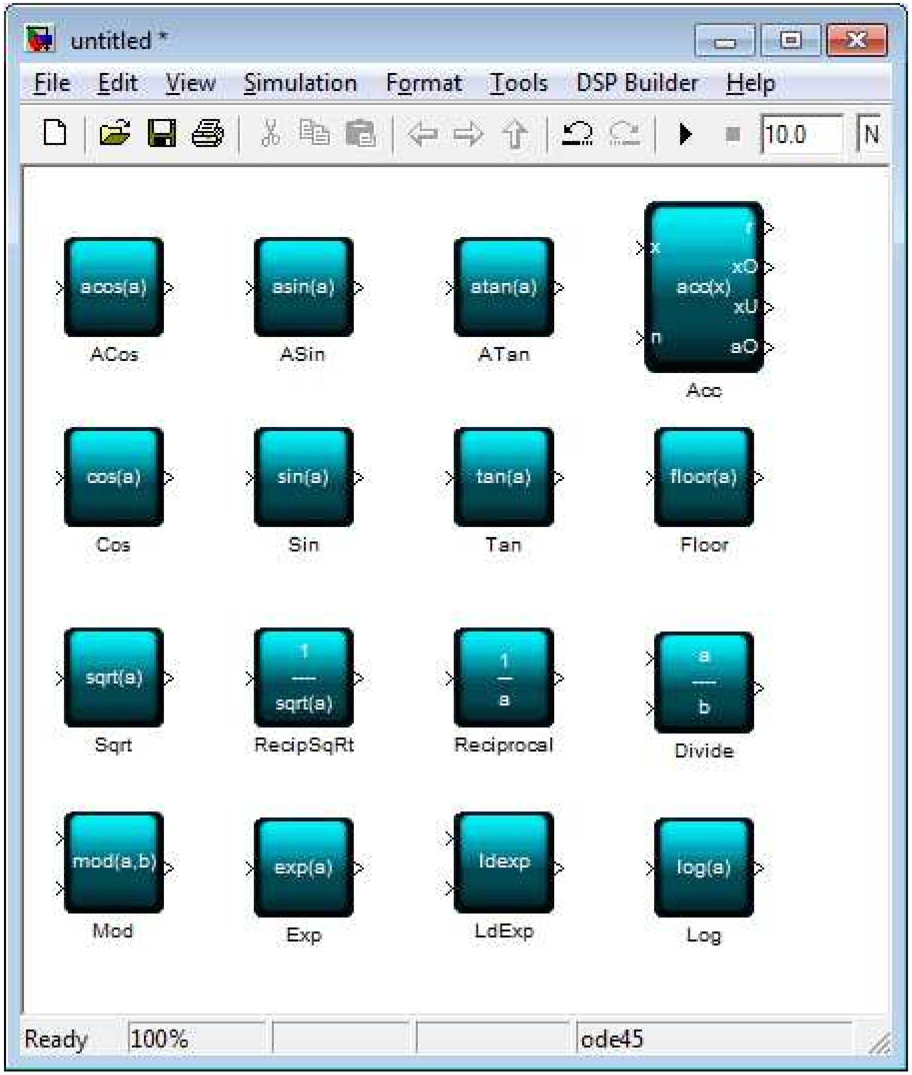
High Performance
Low Resources

Vector Dot Product Example



Floating Point Functions

- Math.h
  - SIN
  - COS
  - TAN
  - ASIN
  - ACOS
  - ATAN
  - EXP
  - LOG
  - LOG10
  - LDEXP
  - FLOOR
  - CEIL
  - SQRT
  - 1/SQRT
  - DIVIDE
  - MOD



Implemented using "Fused Datapath"

Stratix V Floating Point Performance Benchmarks

Fast Fourier Transform (FFT)

Matrix Inversion algorithms

- Cholesky Decomposition
- QR Decomposition

Altera 28nm high end FPGAs

Stratix V "GS" Family					
Part Number	LEs / ALUTs	ALUTs / Registers	DSP Multiplier Count	Mbits / M20 memory blocks	14 GBps Transceiver Count
5SGSD3	236K	178K / 356K	1200	13 / 688	24
5SGSD4	360K	272K / 543K	2088	19 / 957	36
5SGSD5	457K	345K / 690K	3180	39 / 2014	36
5SGSD6	583K	440K / 880K	3550	45 / 2320	48
5SGSD8	695K	525K / 1050K	3926	50 / 2567	48

Fast Fourier Transform (FFT) Performance (Mid-size Stratix V, full Floating Point)

FFT MegaCore Device: 5SGSD5	14 Single Precision Floating-point FFT cores, 1,024 pt		
	Usage	Max	%
Logic utilization	317,332	345,200	92%
ALUT	259,844	345,200	76%
Reg	289,781	690,400	42%
Mem bits	1,954,120	41,246,720	5%
M20K	1,190	2,014	59%
18x18 Multipliers	448	3,180	28%
f_{MAX}	**304 MHz**		
Transform time per core	**3.4 us (0.24 us aggregate transform time)**		

28 nm Stratix V FPGA: ~1W per Floating-Point FFT Core

FPGA verses DSP Processor

Device	Altera Stratix V 5SGSD8	Texas Instruments TMS320C6678
Resources	695 kLEs 50 Mbits block mem 3926 multipliers 48 TRX (14 GSPS)	8 cores, fixed and SP floating point 1.25 GHz
Peak GMACs (16x16 or 18x18)	**2350** (3926 multipliers @ 600 Mhz)	**320** (40 GMACs per core)
Peak GFLOPs Rating (single precision)	**1000** (see 1 TeraFlop whitepaper)	**160** (20 GFLOPs per core)
1024 length floating point FFT performance (single precision)	**3.41 us** (1024 clock cycles@ 300 MHz)	**10.26 us** (12800 clock cycles @ 1.25 GHz)
Aggregate 1024 length FFT transform time	**0.17 us** (20 FFTs per device)	**1.28 us** (8 FFTs per device, 1 per core)

The Cholesky Decomposition

- The Least Squares solution for x in Ax = b
- A must be Hermitian (conjugate symmetric)
  - Only lower triangular matrix is needed for calculation
- If A is positive definite, it can be decomposed into lower triangular matrix L and conjugate transpose L' (A= L * L')
- With Cholesky decomposition, x is solved via forward and backward substitution with decomposed matrices L and L'
- Cholesky decomposition method is more efficient than LU decomposition methods which are suitable for any matrix.

Solving Diagonal Elements

$$A = \begin{bmatrix} L_{11} & 0 & 0 & 0 \\ L_{21} & L_{22} & 0 & 0 \\ L_{31} & L_{32} & L_{33} & 0 \\ L_{41} & L_{42} & L_{43} & L_{44} \end{bmatrix} \begin{bmatrix} L_{11} & L_{21} & L_{31} & L_{41} \\ 0 & L_{22} & L_{32} & L_{42} \\ 0 & 0 & L_{33} & L_{43} \\ 0 & 0 & 0 & L_{44} \end{bmatrix} = \begin{bmatrix} \boxed{L_{11}^2} & & & \text{ConjugateSymmetric} \\ L_{21}L_{11} & \boxed{L_{21}^2 + L_{22}^2} & & \\ L_{31}L_{11} & L_{31}L_{21} + L_{32}L_{22} & \boxed{L_{31}^2 + L_{32}^2 + L_{33}^2} & \\ L_{41}L_{11} & L_{41}L_{21} + L_{42}L_{22} & L_{41}L_{31} + L_{42}L_{32} + L_{43}L_{33} & \boxed{L_{41}^2 + L_{42}^2 + L_{43}^2 + L_{44}^2} \end{bmatrix}$$

$$A_{jj} = \sum_{k=1}^{j} L_{jk} * L'_{kj}$$

where j is the column index of the matrix

$$A_{jj} = \sum_{k=1}^{j} L_{jk} * \mathrm{conj}(L_{jk})$$

The first non-zero element, at the top of each column can be obtained by:

$$L_{jj} = \sqrt{A_{jj} - \sum_{k=1}^{j-1} L_{jk} * \mathrm{conj}(L_{jk})}$$

Equation 1

$$L_{11} = \sqrt{A_{11}}$$

Off-diagonal Elements

$$A = \begin{bmatrix} L_{11} & 0 & 0 & 0 \\ L_{21} & L_{22} & 0 & 0 \\ L_{31} & L_{32} & L_{33} & 0 \\ L_{41} & L_{42} & L_{43} & L_{44} \end{bmatrix} \begin{bmatrix} L_{11} & L_{21} & L_{31} & L_{41} \\ 0 & L_{22} & L_{32} & L_{42} \\ 0 & 0 & L_{33} & L_{43} \\ 0 & 0 & 0 & L_{44} \end{bmatrix} = \begin{bmatrix} L_{11}^2 & & & \text{ConjugateSymmetric} \\ \boxed{L_{21}L_{11}} & L_{21}^2 + L_{22}^2 & & \\ \boxed{L_{31}L_{11}} & \boxed{L_{31}L_{21} + L_{32}L_{22}} & L_{31}^2 + L_{32}^2 + L_{33}^2 & \\ \boxed{L_{41}L_{11}} & \boxed{L_{41}L_{21} + L_{42}L_{22}} & \boxed{L_{41}L_{31} + L_{42}L_{32} + L_{43}L_{33}} & L_{41}^2 + L_{42}^2 + L_{43}^2 + L_{44}^2 \end{bmatrix}$$

$$A_{ij} = \sum_{k=1}^{j} L_{ik} * L'_{kj}$$

where i and j are the row and column indices of the matrix

$$A_{ij} = \sum_{k=1}^{j} L_{ik} * \mathrm{conj}(L_{jk})$$

where L_{jk} is the transpose of L_{kj}

Equation 2

$$L_{ij} = \frac{A_{ij} - \sum_{k=1}^{j-1} L_{ik} * \mathrm{conj}(L_{jk})}{L_{jj}} \Rightarrow L_{ij} = \frac{A_{ij} - \sum_{k=1}^{j-1} L_{ik} * \mathrm{conj}(L_{jk})}{\sqrt{A_{jj} - \sum_{k=1}^{j-1} L_{jk} * \mathrm{conj}(L_{jk})}}$$

Forward Substitution

We now have L and L' thus $A * x = b \rightarrow L * L' * x = b$

If we define: $y = L' * x \rightarrow L * y = b$

L is the lower triangular matrix, y and b are column matrices and b is known in the system so y can be solved by forward substitution

$$y_j = \frac{b_j - \sum_{k=1}^{j-1} y_k * L_{jk}}{L_{jj}}$$

Equation 3

Note that solving for y is very similar to solving for L shown below

$$L_{ij} = \frac{A_{ij} - \sum_{k=1}^{j-1} L_{ik} * \mathrm{conj}(L_{jk})}{L_{jj}}$$

Equation 2

Since equations are similar, Cholesky decomposition and forward substitution are combined into the same process. The only difference is that Eq 2 is conjugated

Backward Substitution

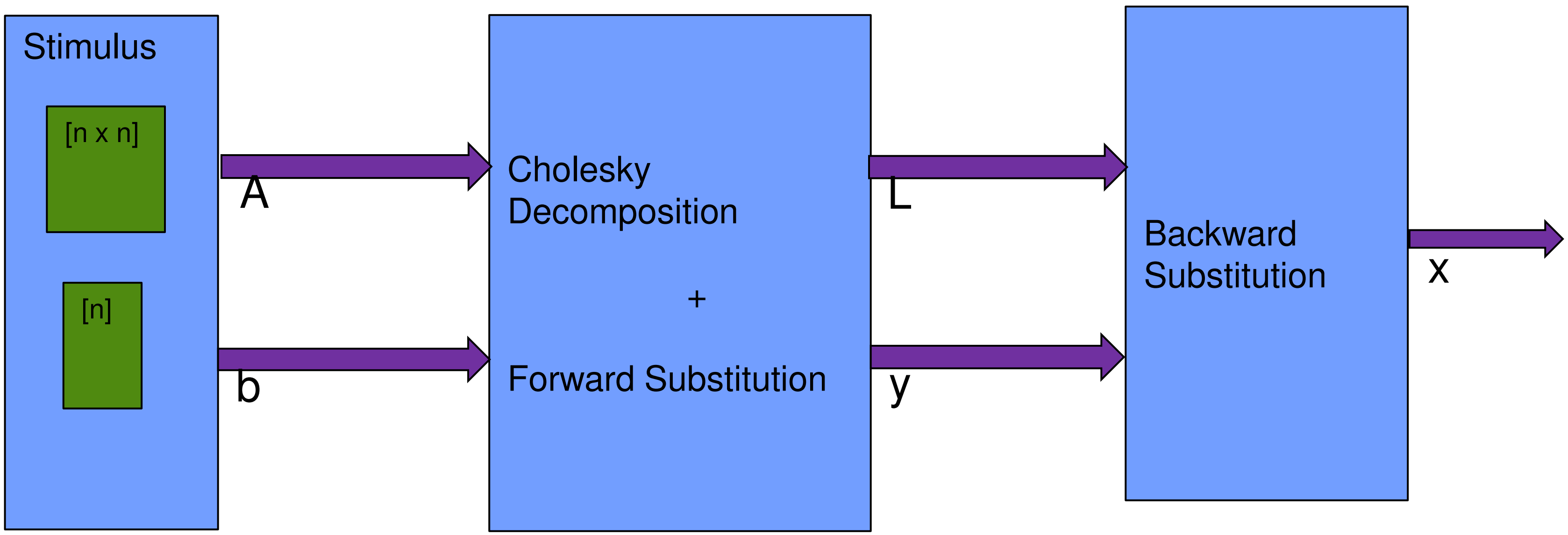
x can be solved by backward substitution, $L' * x = y$

Since L' is an upper triangular matrix, x has to be solved from the bottom to the top, hence why it's called back substitution

$$x_j = \frac{y_j - \sum_{k=j+1}^{VS} x_k * L'_{jk}}{L'_{jj}}$$

Equation 4

Cholesky Block Diagram



Solve for x in Ax = b where A is conjugate symmetric

Performance and FPGA Resources

Cholesky Decomposition Parameterizable Core using 5SGSD5					
Complex Input Matrix Size	Vector Size	ALUTs / Memory blocks / 27x27s	% ALUTs / % Memory blocks / % 27x27s	Latency @ Operating frequency	GFLOPS per core (complex single precision)
30x30	30	76.5K 793 M20K 146 DSP	22% 39% 9%	255 us @ 250 MHz	21.7
60x60	60	141K 955 M20K 268 DSP	41% 47% 17%	328 us @ 235 MHz	39.0
240x240	60	154K 1820 M20K 268 DSP	45% 90% 17%	922 us @ 220 MHz	74.2
360x360	90	204K 1411 M20K 391 DSP	59% 70% 25%	1103 us @ 190 MHz	91.8
400x400	100	220K 1619 M20K 430 DSP	64% 80% 27%	1342 us @ 190 MHz	103

GFLOPs and GFLOPs/Watt

Cholesky Decomposition Parameterizable Core using 5SGSD5					
Complex Input Matrix Size (n x n)	Vector Size	Through-put (Matrix per second)	GFLOPS per core (complex single precision)	Core power consumption as measured using Altera 5SGSD5 eval board	GFLOPs/Watt
30x30	30	472,464	21.7	7.7 W	2.8
60x60	60	118,858	39.0	13.6 W	2.9
240x240	60	8,467	74.2	14.0 W	5.3
360x360	90	1142	91.8	14.7 W	6.2
400x400	100	1182	103	16.1 W	6.4
Complex Cholesky FLOPs = $4/3n^3 + 8n^2$					

Competive Results: Nvidia GPU

Cholesky Decomposition (single precision)			
Matrix Size	GFLOPs with LAPACK Library	GFLOPs with Magma Library	GFLOPs with Nvidia OpenCL Library
512x512	20	22	58
768x768	20	39	82
1024x1024	36	57	68
2048x2048	60	117	96
Cholesky FLOPs = 4 $N^3/3$, where N is matrix dimension			

- Results in about 0.25 GFLOPs/Watt (512x512)
- Nividia GTX480 rated at 977 GFLOPs
- Intel Pentium4 3.7GHz rated at 14.8 GFLOPs

High Performance Relevance Vector Machine on GPUs
Depeng Yang, Getao Liang, David Jenkins,Gregory D. Peterson, and Husheng Li
U of Tennessee, Knoxville

More Nvidia Results

LU Decomposition (single precision)			
Matrix Size	CPU GFLOPs	GPU GFLOPs	GPU speedup
1024x1024	24.2	51.4	3.1
2048x2048	26.5	111.7	5.2
3072x3072	27.5	151.6	6.5
4032x4032	29.96	183.02	7.1
Using Magma 1.0 RC5 library			

- Nvidia Fermi Tesla C2050, 1147.0 MHz clock
- AMD Quadro NVS 290, 918.0 MHz clock

MAGMA LAPACK for GPUs
Stan Tomov, Research Director, Innovative Computing Laboratory
Department of Computer Science
University of Tennessee, Knoxville

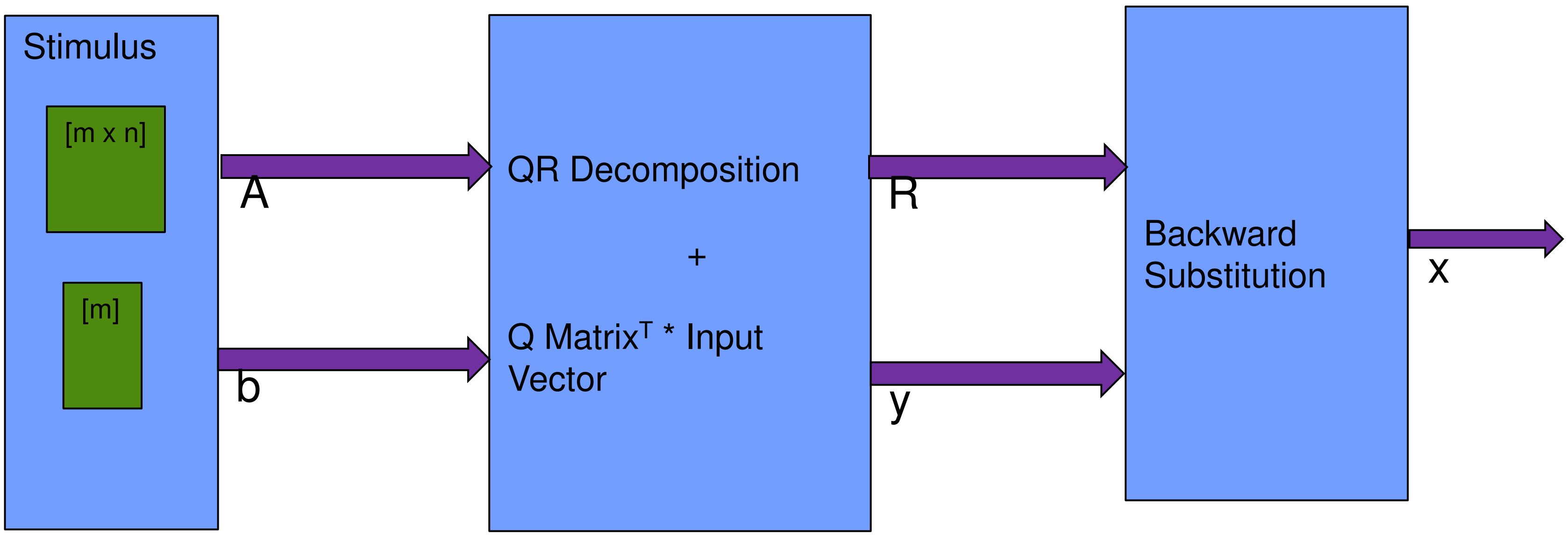
QR Decomposition

- QR Solver finds solution for Ax=b linear equation system using QR decomposition, where Q is ortho-normal and R is upper-triangular matrix. A can be rectangular.

- Steps of Solver
  - *Decomposition:* $A = Q \cdot R$
  - *Ortho-normal property:* $Q^T \cdot Q = I$
  - *Substitute then mult by* Q^T*:* $Q \cdot R \cdot x = b$ $R \cdot x = Q^T \cdot b = y$
  - *Backward Substitution:* $Q^T \cdot b = y$ *solve* $R \cdot x = y$

- *Decomposition is done using Gram-Schmidt derived algorithms. Most of computational effort is in "dot-product"*

Block Diagram



Solve for x in Ax = b where A is non-symmetric, may be rectangular

Performance and FPGA Resources

QR Decomposition Parameterizable Core using 5SGSD5					
Complex Input Matrix Size	Vector Size	ALUTs / Memory blocks / 27x27s	% ALUTs / % Memory blocks / % 27x27s	Latency @ Operating frequency	GFLOPS per core (complex single precision)
50x100	50	105K 230 M20K 227 DSP	30% 11% 14%	45 us @ 250 MHz	43.8
100x200	50	106K 304 M20K 228 DSP	31% 15% 14%	213 us @ 250 MHz	64.3
100x200	100	202K 504 M20K 428 DSP	58% 25% 27%	173 us @ 200 MHz	91.9
250x400	100	200K 858 M20K 428 DSP	58% 43% 27%	1586 us @ 200 MHz	106
400x400	100	203K 1566 M20K 428 DSP	59% 78% 27%	4029 us @ 200 MHz	106

GFLOPs and GFLOPs/Watt

QR Decomposition Parameterizable Core using 5SGSD5					
Complex Input Matrix Size (n x m)	Vector Size	Through-put (Matrix per second)	GFLOPS per core (complex single precision)	Core power consumption as measured using Altera 5SGSD5 eval board	GFLOPs/Watt
50x100	50	31,681	43.8	10.8 W	4.1
100x200	50	5,920	64.3	13.9 W	4.6
100x200	100	8,467	91.9	21.0 W	4.4
400x400	100	310	106	25.2 W	4.2
450x450	75	165	80.0	20.2	4.0
Complex QRD FLOPs = $5.33mn^2 + 8mn - 2n + 4n^2$					

Accuracy, Validation, and summary

Computational error analysis

QR Decomposition Accuracy			
Complex Input Matrix Size (n x m)	Vector Size	MATLAB using computer Norm/Max	DSPBA generated RTL Norm/Max
50x100	50	5.01e-5 / 6.42e-6	4.87e-5 / 6.02e-6
100x200	100	2.3e-5 / 1.24e-6	1.68e-5 / 9.97e-7
400x400	100	8.8e-5 / 4.81e-6	7.07e-5 / 4.03e-6
using Frobenius norm $\|E\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}\left|e_{ij}\right|^2}$			

Cholesky Decomposition results are similar

Summary

- High performance floating point designs can be built using FPGAs
  - High density of 27x27, 36x36, 54x54, 72x72 multipliers available at 28nm
  - New floating point toolflow reduces routing density to sustainable level
  - Availability of optimized math.h library of floating point functions
- FPGA Fixed point parellelism performance benefits now carry over into floating point
- Best in class GFLOPs / Watts
- Real-world, not marketing, floating point benchmarks for comparison