



# Double Precision Floating Point on FPGAs

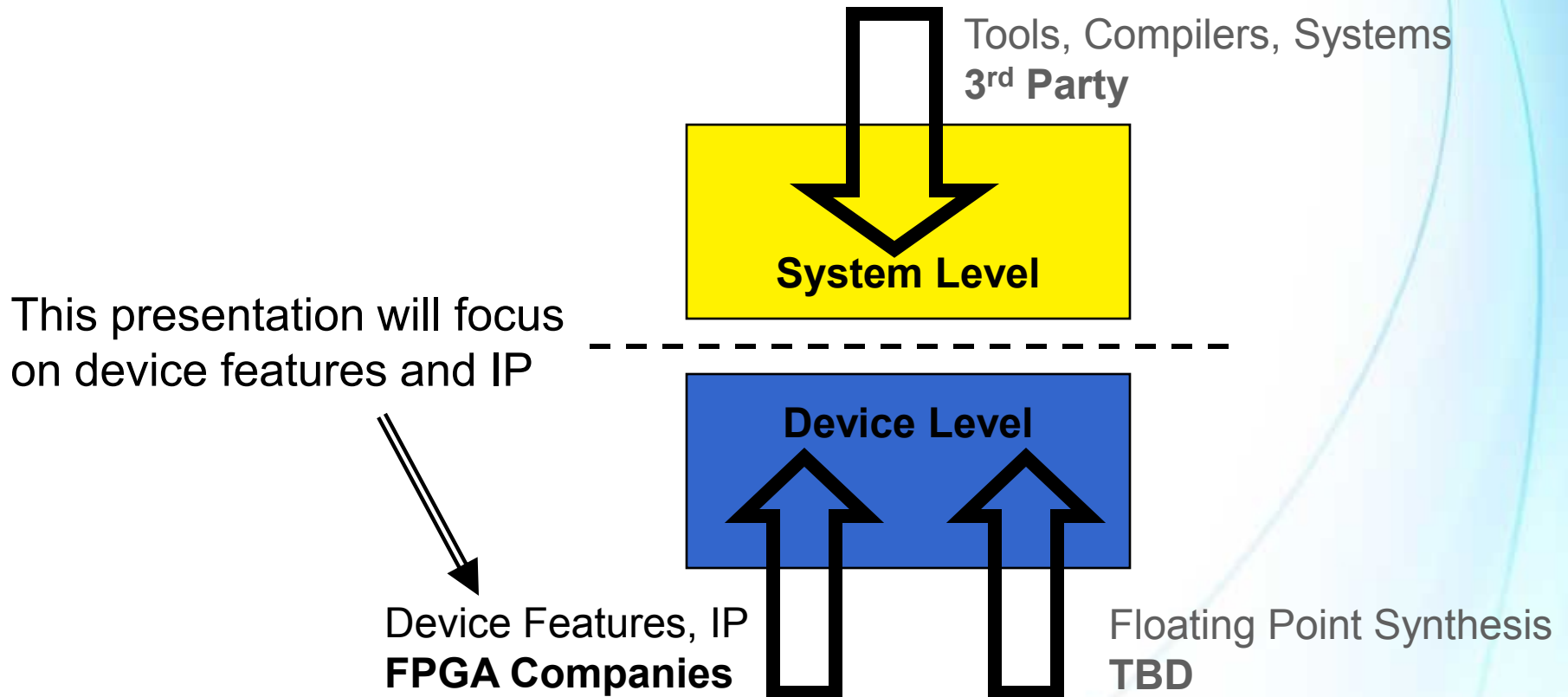
*Martin Langhammer*

*RSSI – July 18, 2007*



# High Performance Computing on FPGAs

- Three key technologies



# Agenda

- Floating Point Requirement
  - What do applications need?
  - What can FPGAs support?
- Floating Point Device Capability
- FPGA DSP Block Architecture
- FPGA IEEE754 Arithmetic Operators
- Matrix Multiply Benchmark Result



# Floating Point Requirements



# Application Requirements

## ■ Common Algorithms:

- FFT, FIR filters, SAR
- Monte Carlo, Black Scholes
- General Matrix Math

## ■ Resource Usage

- Typically a 1:1 ratio of multiplies to adds/subtracts
  - Dot Product/Matrix Multiply - 1:1
  - Matrix Determinant - 2:1
  - R4 Butterfly – 1:2

## ■ Metrics

- Raw GFLOPs
  - Must be supported by memory, interface bandwidth
  - Soft logic and memory resources for application implementation

# Trends in Floating Point

- Double Precision IEEE754 benchmark
- Power constraints
  - Improved GFLOPS/watt ratio
  - Performance density and power density severely limiting evolution of GPU based solutions
- Ease of use
  - Abstract programming model
  - Rapid application development and implementation
    - Multi-day fitting unacceptable

# FPGA Fabric

- Must support 54x54 unsigned structure
  - Readily decomposes to 18x18, 36x36
  - Commonly available on current generation FPGAs
- Performance derived from hard multipliers
  - 54x54 soft multipliers suffer latency, area, routing penalties
  - Some applications require more adder/subtractors than multipliers
    - Need sufficient soft logic resources to support differing multiplier:ALU ratios
- Ideal balance of resources (next slide)
  - Not just floating point functionality
  - Logic resources needed for applications implementation, interface, overhead for fitting

# Resource Requirements

- FPGA for HPC requires interface, FP and application logic
- Interfaces and Application Logic
  - HT 2.0 requires about 20K LUTs
  - Application logic (muxing, memory controllers, etc) will vary
- Double Precision Blocks
  - Adder/subtractor: >1K LUTs
  - Multiplier: 54x54 block <1K LUTs
  - Total Logic per operator pair + applications : 3K LUTs
- 3K LUTs is a 100% utilized design with 64 bit busses
- Many HPC applications require faster turnaround for design, fitting
  - Desired performance: 90nm @ 200 MHz, 65nm @ 250 MHz
- Real world design requirements : 60-80% full device
  - 4K-5K LUTs per 54x54 multiplier ratio



# Floating Point Benchmarking



# Pencil & Paper Evaluation (Theoretical)

- Article by Dave Strenski @ Cray
  - <http://www.hpcwire.com/hpc/1195762.html>
  - Compares Opteron Peak with Xilinx FPGA
  - Double Precision Floating Point
  - Use best combination of multipliers & logic
  - Includes logic budget for I/O interfaces
- Does not account for routing & fitting
  - Hard to route all the 64 bit data paths
  - “Constrained benchmarks” reduce utilization by 15%, reduce peak Fmax by 33% (57% sustained rate)
  - Still optimistic for large designs
- Let’s examine 1:1 Multiply:ALU ratio

# Constrained DP GFLOPS

| Device                            | Theoretical GFLOPs  | Constrained GFLOPs <sup>4</sup> |
|-----------------------------------|---------------------|---------------------------------|
| Opteron                           | 10                  | 10                              |
| Xilinx V4 LX200                   | 15.9 <sup>1,2</sup> | 9.1                             |
| Stratix <sup>®</sup> II EP2S180   | 25.2 <sup>3</sup>   | 14.4                            |
| Xilinx V5 LX330                   | 28.0 <sup>1,2</sup> | 16.0                            |
| Stratix <sup>®</sup> III EP3SE260 | 50.7 <sup>3</sup>   | 28.9                            |

**Notes:**

- 1 – Analysis by Dave Strenski, Cray.
- 2 – Multipliers using both hard and soft multiplier cores.
- 3 – Multipliers using only hard multiplier cores.
- 4 – Constraints : 15% logic reserve, 33% FMax degradation.

# DP GFLOPS/Watts (1/2 mult, 1/2 add)

| Device                | GFLOPS/<br>WATT | Sustained<br>GFLOPS | Peak<br>GFLOPS | Power |
|-----------------------|-----------------|---------------------|----------------|-------|
| AMD 2.5GHz            | 0.1             | 8?                  | 10*            | 68W   |
| 3Ghz Woodcrest        | 0.2             | 17**                | 24             | 80W   |
| Xilinx V4LX200        | 0.4             | 9.1 †               | 15.9*          | 25W   |
| Xilinx V5LX330        | 0.5             | 16.0 †              | 28.0*          | 30W   |
| Stratix® II EP2S180   | 0.6             | 14.4 †              | 25.2*          | 25W   |
| Stratix® III EP3SE260 | 0.9             | 28.9 †              | 50.7*          | 30W   |

Note \*: HPCWire, Dave Strenski;

Note \*\*: Clearspeed; ATI; Altera

Note †: peak with 33% Fmax decrease and 15% logic remaining

# Other Implementation Issues

- Pentium accuracy
- IEEE754 operations are non-associative
  - Different answers for sequential and parallel systems
  - Kapre and DeHon(\*), suggest that maintaining bit accuracy in a parallel flow will require about twice the resources
  - May be better to relax requirement for most applications
- Interface Bandwidth
  - Matrix Multiply requires 3 I/O operations per element
    - DGEMM requires 4 I/O operations
  - As parallelism increases, I/O can become bottleneck
    - Use matrix blocking to optimize for FPGA on board cache
    - Faster or multiple interfaces to accelerator card or FPGA
- Memory size and bandwidth
  - Size and bandwidth are two different issues

(\*) N. Kapre and A. DeHon, Optimistic Parallelization of Floating Point Accumulation. In *Proc. 18<sup>th</sup> IEEE Symposium in Computer Arithmetic*, pages 205-213, June 2007



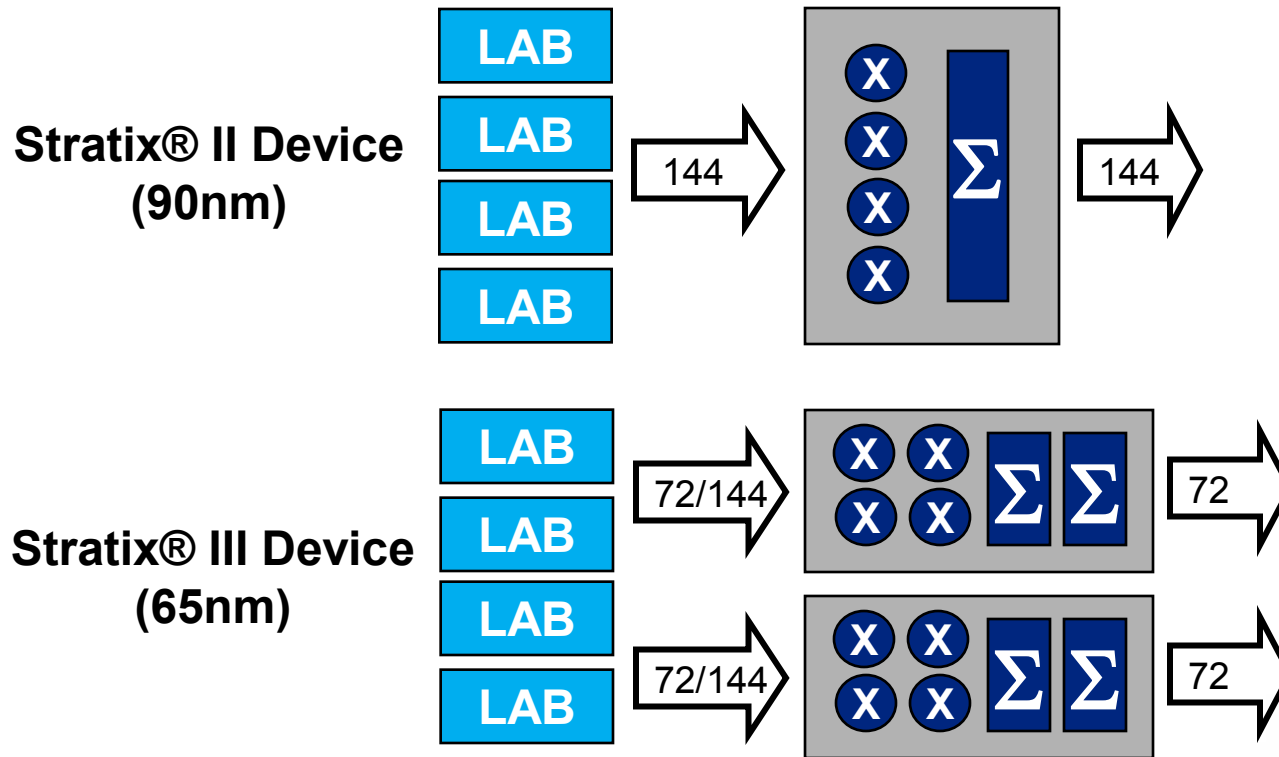
# FPGA DSP Block Architecture



# Stratix<sup>®</sup> III FPGA DSP Evolution

- Multipliers must be able to sum with other multipliers
- Routing is expensive
  - It takes up area
  - It forces aspect ratios and placement
- By decoupling routing from DSP resources i.e. not mandating a 1:1 ratio, we have more flexibility
  - Most of the functions we need to support don't have a 1:1 ratio
  - Sum of Multiplies: 2:3 routing: logic ratio
  - Complex Multiplies: 1:2 ratio
  - Large Multiplies: 1:2 ratio
- By allowing an input sharing scheme, almost all applications supported

# DSP Block Routing to Logic Interface



Routing for higher integration functions (36x36, 54x54, complex, FIR) now moved from interface into DSP logic. The aspect ratio of the block is no longer forced to align with other structures.

***Decoupling Logic and Routing Allows for Higher DSP Density***

# Stratix<sup>®</sup> III FPGA DSP Block

## ■ Basic Multiplier Modes

- 8 x (9x9)
- 6 x (12x12)
- 4 x (18x18)
- 2 x (36x36)
- 2 x complex (18x18)

## ■ Sum Modes

- 4 x Sum of Two (18x18)
- 2 x Sum of Four (18x18)

## ■ Accumulation

- 2 x Acc

## ■ Cascade Modes

- Input Cascade
- Output Cascade

## ■ Rounding

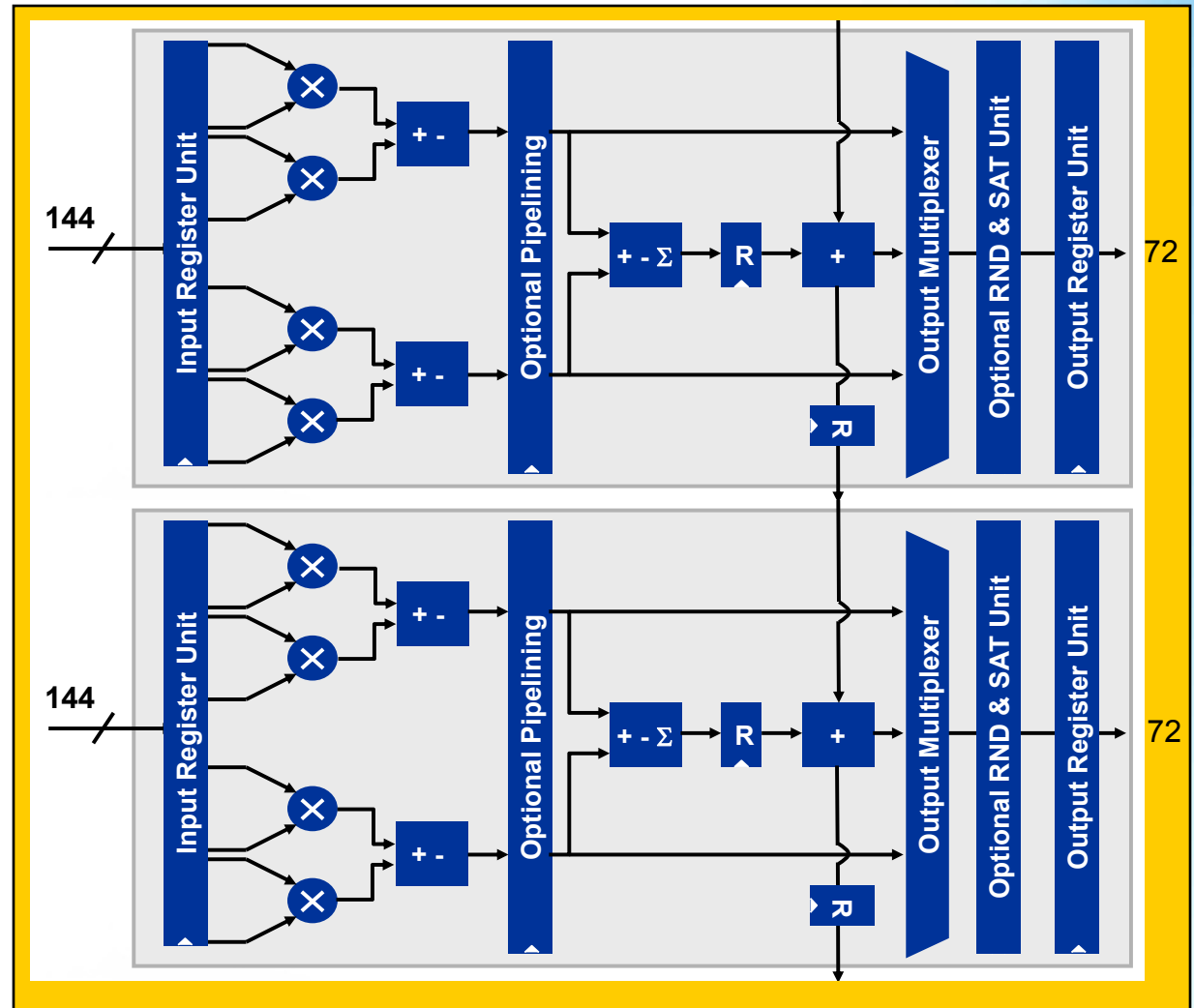
- Unbiased and Biased

## ■ Saturation

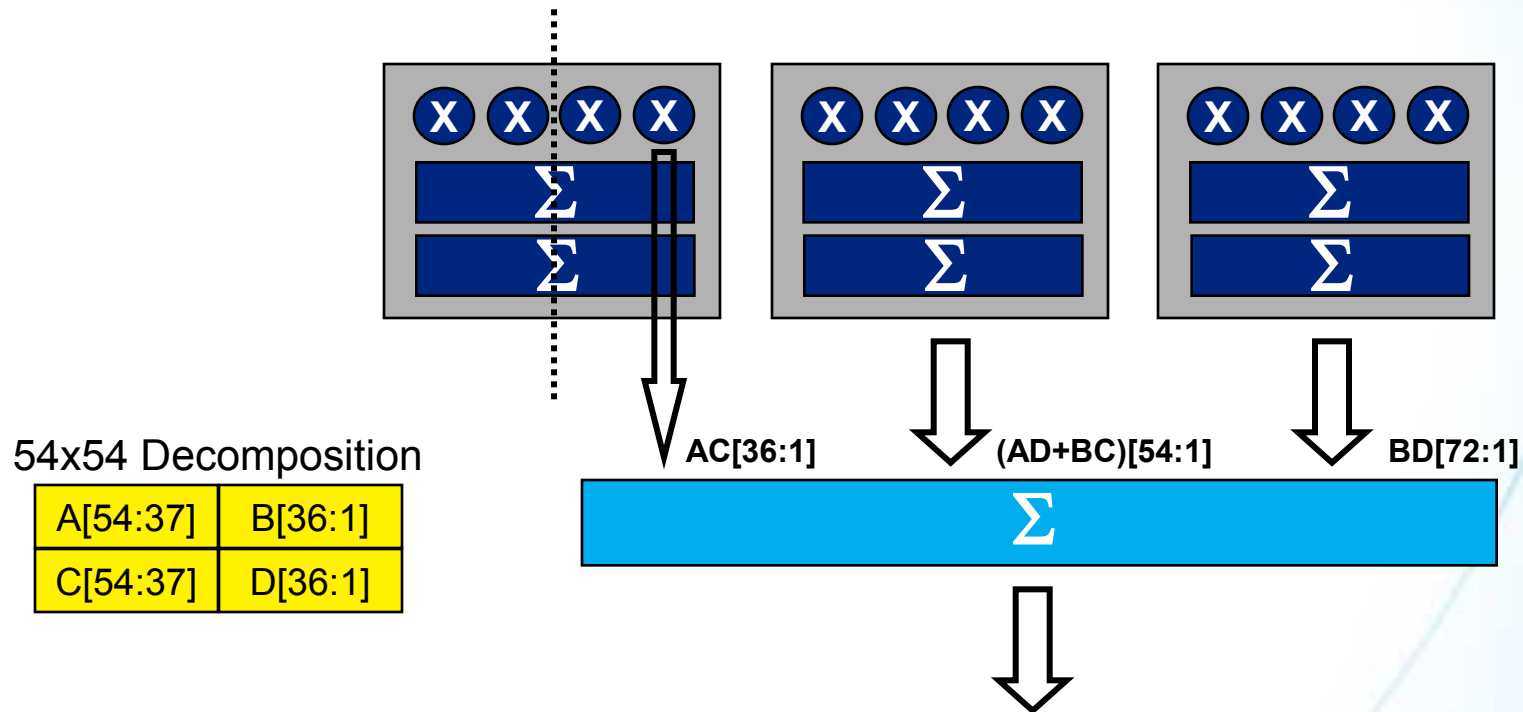
- Asymm and Symmetrical

## ■ Barrel Shifter

- Arithmetic, Logical and Rotation



# Stratix<sup>®</sup> III FPGA 54x54 Mode



## Notes

1. Almost all elements of the 54x54 multiplier are inside the DSP Blocks
2. Only a single external adder is required
3. Logic support and latency for double precision multipliers reduced



# FPGA IEEE754 Arithmetic Operators



# IEEE754 Cores

- Four basic arithmetic functions + sqrt
  - Add/Subtract
  - Multiply
  - Divide
  - Square root
- Parameterized support for double, single, extended, user defined precision
- Tested for 100% Pentium Bit Accuracy
- No denormalized number support
  - Denormal inputs and outputs treated as '0'
- Single Architecture for all Stratix® FPGAs
  - Stratix® III FPGAs have special 54x54 mode, more latency options

# Core Summary – Stratix® II FPGAs

| Operation           | Area (LUT) | Area (FF) | Latency | Performance |
|---------------------|------------|-----------|---------|-------------|
| ALU (Single)        | 533        | 496       | 10      | 355 MHz     |
| ALU (Double)        | 1239       | 1429      | 13      | 303 MHz     |
| Multiplier (Single) | 128        | 249       | 8       | 395 MHz     |
| Multiplier (Double) | 644        | 824       | 10      | 325 MHz     |
| Divider (Single)    | 775        | 1529      | 28      | 282 MHz     |
| Divider (Double)    | 2775       | 6323      | 57      | 212 MHz     |
| Root (Single)       | 497        | 908       | 28      | 310 MHz     |
| Root (Double)       | 1881       | 3569      | 57      | 201 MHz     |

For **Stratix® III** FPGAs (65nm) results, add 20% FMax



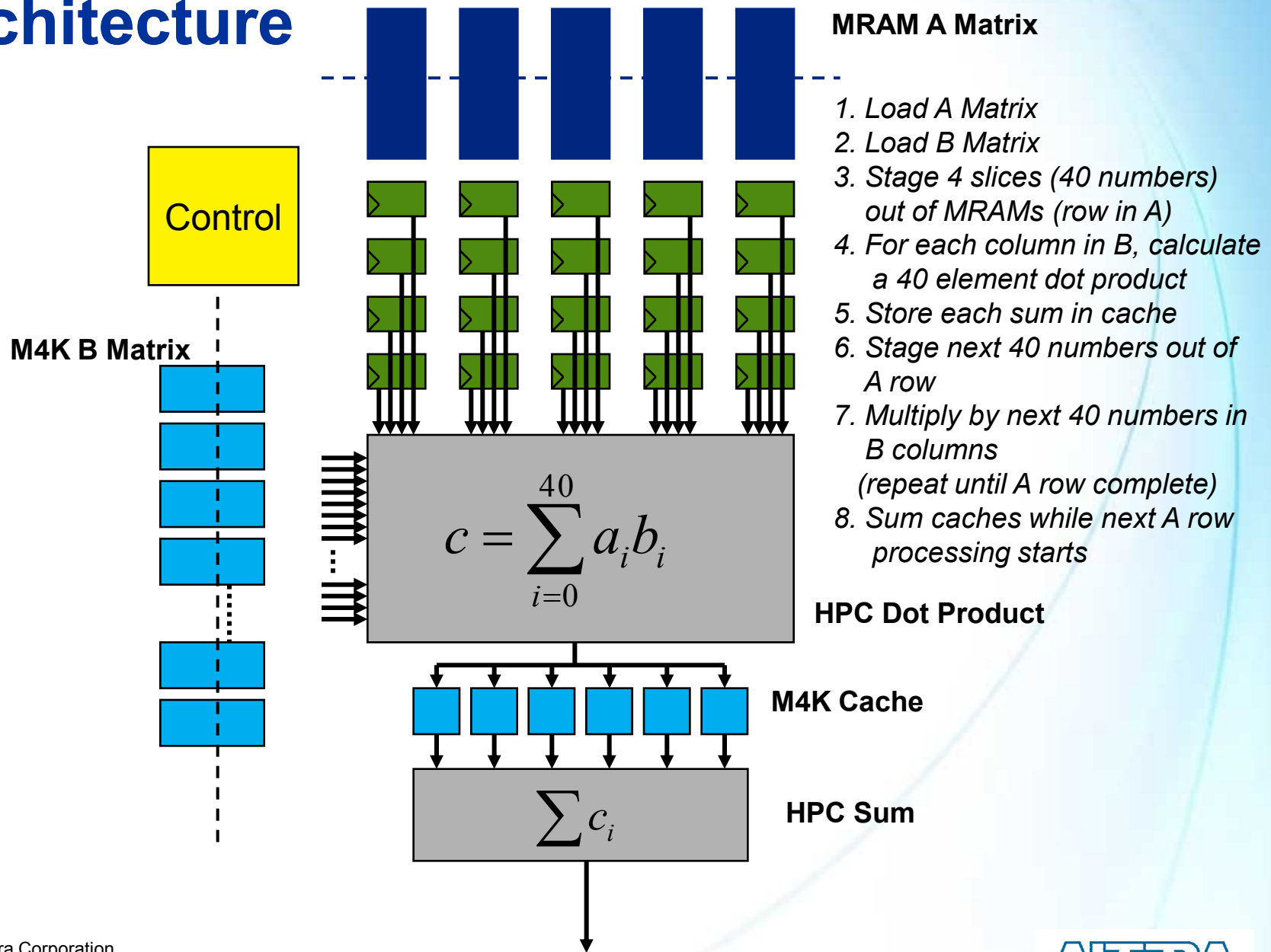
# Matrix Multiply Benchmark Result



# Maximum Device Capability

- Stratix<sup>®</sup> II EP2S180 – 384 18x18 multipliers
  - 9 18x18 multipliers required per 54x54 multiplier
  - Maximum of 42 54x54 multipliers per device
  - Based on 1:1 multiplier:adder/subtractor ratio, 84 operators are theoretically possible in a single device
    - 79K LUTs, 94K FFs
- Internal tool used to fit 40 multiplier cores and 45 adder cores
- Resulting design pushbutton 200 MHz
  - Application, including arithmetic portion, control, and interface 55% device utilization

# Architecture



# Blocked Matrix Performance

| Matrix Size | Sub Matrixes                         | Transfer Time | Processing Time | Ratio |
|-------------|--------------------------------------|---------------|-----------------|-------|
| 960x960     | A: 80x240<br>B: 240x40<br>C: 80x40   | 25600         | 19200           | 75%   |
| 1024x960    | A: 128x160<br>B: 160x64<br>C: 128x64 | 36864         | 32768           | 89%   |
| 1020x960    | A: 170x120<br>B: 120x85<br>C: 170x85 | 49300         | 43350           | 88%   |
| 1024x960    | A: 170x120<br>B: 120x64<br>C: 170x64 | 37120         | 32640           | 88%   |

Transfer time:

$$2 * (\text{brows} * \text{bcols} + \text{arows} * \text{bcols})$$

Processing time:

$$\text{arows} * \text{bcols} * \text{acols} / 40$$

**88% Sustained Throughput**

**80 Operators @% 100 Usage,  
2-6 operators @ 16-50% Usage =  
81 effective operators**

**200 MHz system speed \*  
81 operators \* 88% duty cycle**

**14.25 GFLOPs**



# Summary



# Summary

- FPGAs can implement double precision floating point operations effectively
- Increase in resources expected over next few years
  - Logic/DSP balance will define limits
- New tools and methods needed to make HPC effective on FPGAs
  - 100% resource utilization possible for pushbutton fits