USC
UNIVERSITY
OF SOUTHERN
CALIFORNIA

Area and Power Performance Analysis of Floating-point based Applications on FPGAs

Gokul Govindu, Ling Zhuo, Seonil Choi, Padma Gundala,
and Viktor K. Prasanna
Dept. of Electrical Engineering
University of Southern California
September 24, 2003

http://ceng.usc.edu/~prasanna

USC UNIVERSITY OF SOUTHERN CALIFORNIA

Outline

- **Floating-point based Applications on FPGAs**
- **Floating-point Units**
  - **Area/Power Analysis**
- **Floating-point based Algorithm/Architecture Design**
- **Area, Power, Performance analysis for example kernels:**
  - **FFT**
  - **Matrix Multiply**
- **Conclusion**

Floating-point based Applications on FPGAs

Applications requiring

- High numerical stability, faster numerical convergence
- Large dynamic range

Examples:

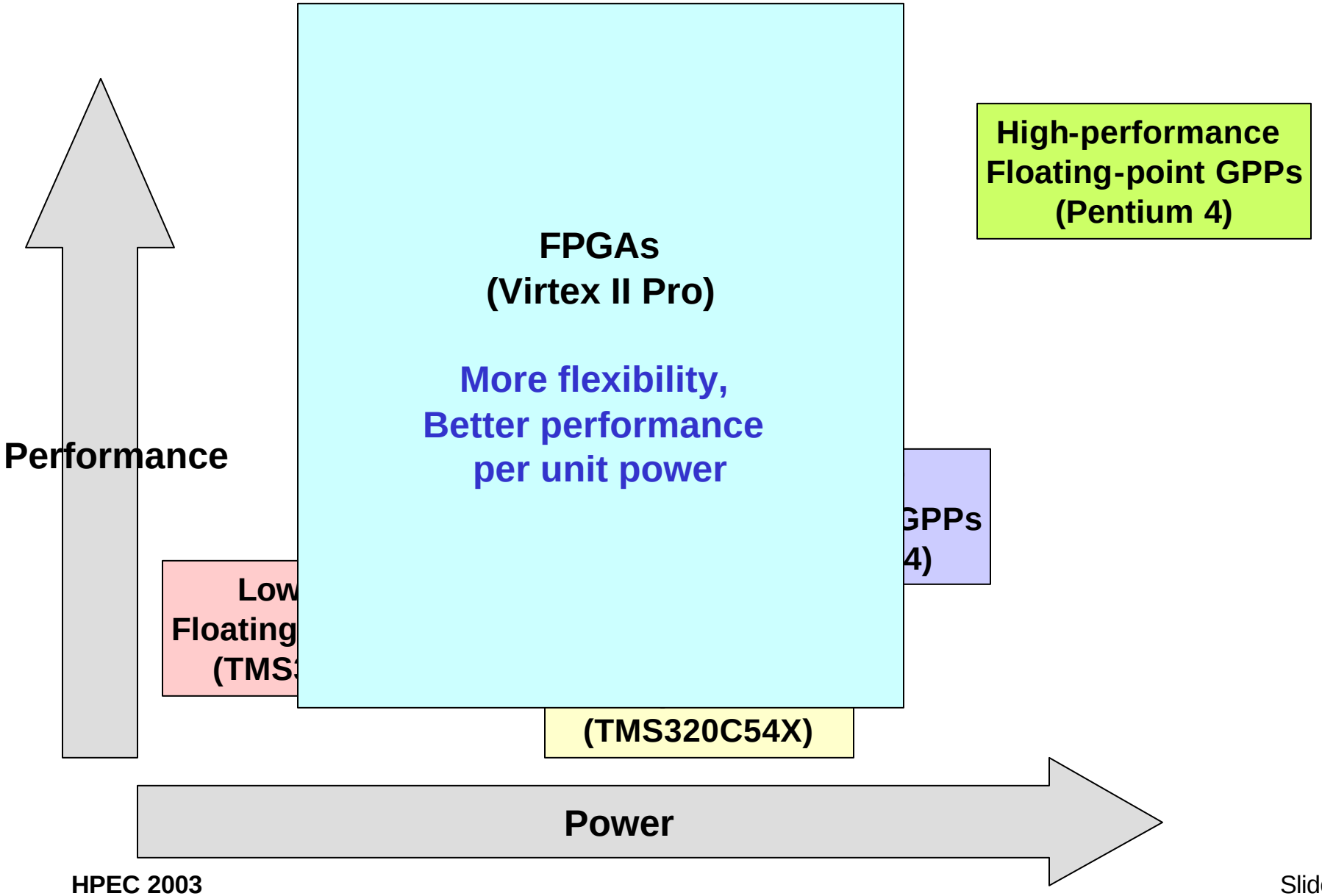
- Audio/Image processing, Radar/Sonar/Communication, etc.

Fixed-point vs. Floating-point

- Resources
  - Slices
- Latency/Throughput
  - Pipeline stages
  - Frequency
- Precision
- Design complexity of fixed/floating-point units

Energy – Area – Performance Tradeoffs

Floating-point Device Options



Need for FPU Design in the Context of the Kernel

Integration

- Latency
  - Number of pipeline stages as a parameter
- Frequency
  - FPU frequency should match the frequency of the kernel/application's logic
- Area/Frequency/Latency tradeoffs

Optimal Kernel Performance

- High throughput
  - Maximize frequency
- Minimize Energy
  - Architectural tradeoffs - FPUs parameterized in terms of latency/ throughput/ area
- Optimize F/A for FPU
  - Maximize the performance of the kernel

Algorithm/Architecture Design

- Re-evaluation of the algorithm/architecture
  - Tolerate latencies of FPU - low area vs. high frequency tradeoffs
  - Re-scheduling

USC UNIVERSITY OF SOUTHERN CALIFORNIA

Outline

- Floating-point based Applications on FPGAs
- Floating-point Units
  - Area/Power Analysis
- Floating-point based Algorithm/Architecture Design
- Area, Power, Performance analysis for example kernels:
  - FFT
  - Matrix Multiply
- Conclusion

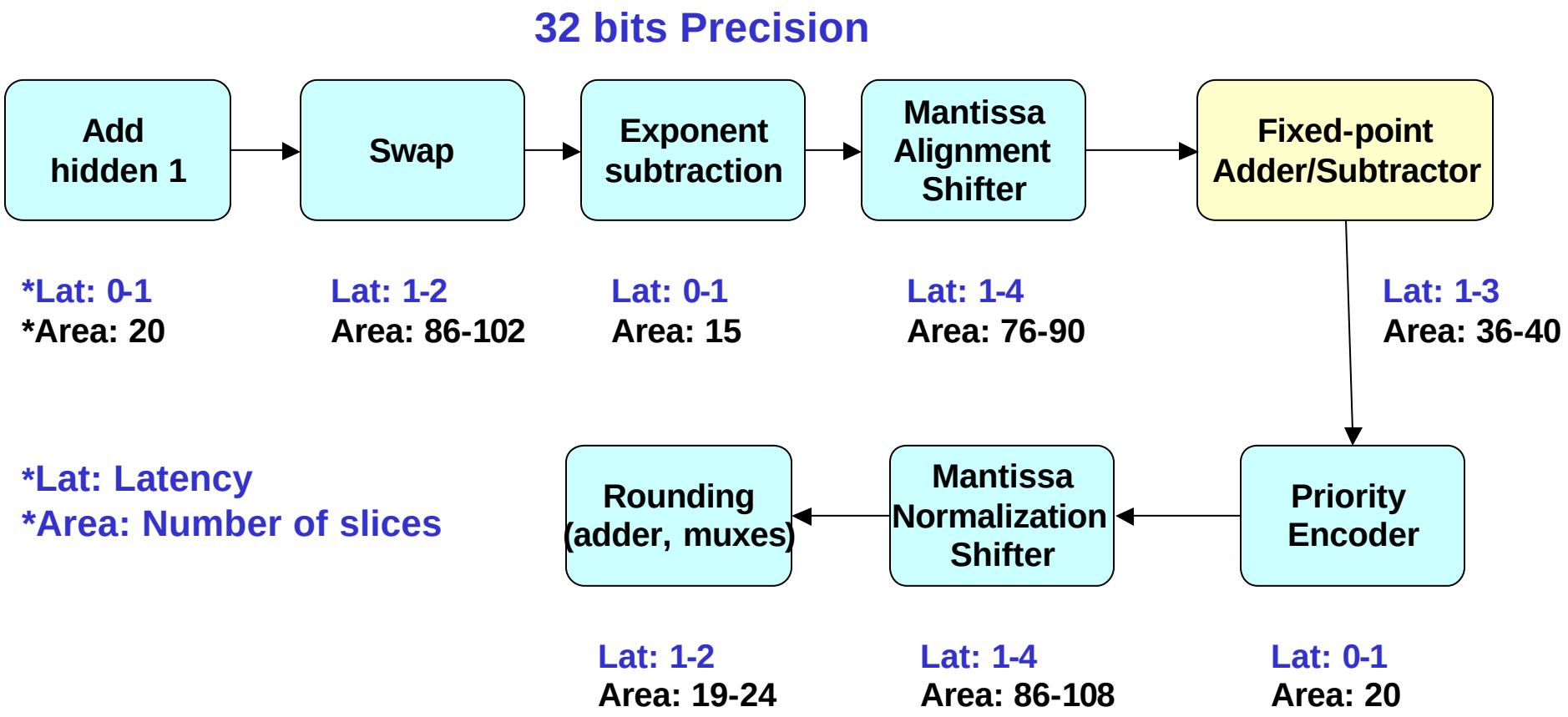
Our Floating-point Units

- Now, easier to implement floating-point units on FPGAs
  - Optimized IP cores for fixed-point adders and multipliers
  - Fast priority encoders, comparators, shift registers, fast carry chains….

Our floating-point units

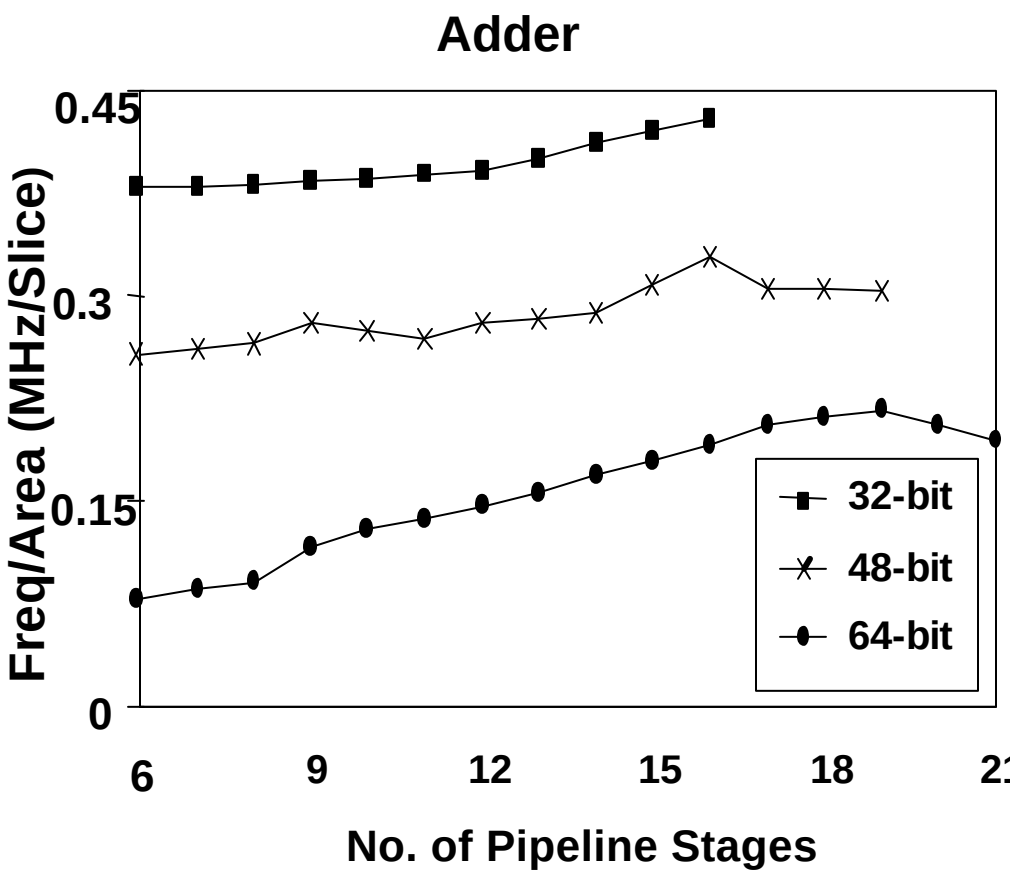
- Precision
  - Optimized for 32, 48 and 64 bits
- IEEE 754 format
- Number of pipeline stages
  - Number of pipeline stages parameterized
    - For easy integration of the units into the kernel
    - For a given kernel frequency, units with optimal pipelining and thus optimal resources, can be used
- Metrics
  - Frequency/Area
  - Overall performance of the kernel (using floating-point units)
  - Energy

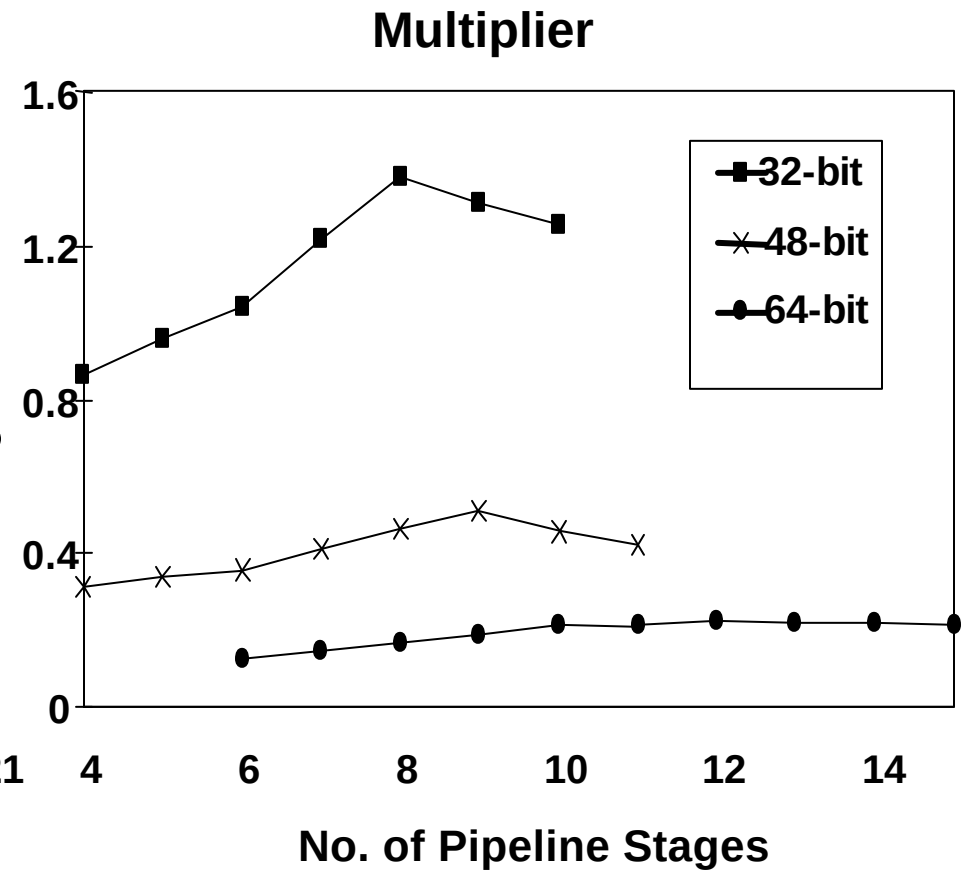
Floating-point Adder/Subtractor



- Pipeline stages: 6-18
- Area: 390- 550; Achievable frequency: 150-250MHz
- Xilinx XC2VP125 –7

Frequency/ Area vs. Number of Pipeline Stages





- Diminishing returns beyond optimal F/A
- Tools' optimization set as "balanced - area and speed"
  - -Area and Speed optimization give different results in terms of area and speed

Addition Units: Some Trade-offs

	Fixed-point		Floating-point		Floating-point	
	32 bits with 2 stages	64 bits with 4 stages	32 bits with 14 stages	64 bits with 19 stages	32 bits with 19 stages	64 bits with 21 stages
Area(slices)	36	139	485	933	551	1133
Max. Freq. (MHz) achievable	250	230	230	200	250	220
Power(mW) at 100MHz	23.48	102	200	463	254	529

Floating-point vs. Fixed-point

- Area : 7x-15x
- Speed: 0.8x-1x
- Power: 5x-10x

Multiplier Units: Some Trade-offs

	Fixed-point		Floating-point		Floating-point	
	32 bits with 5 stages	64 bits with 7 stages	32 bits with 7 stages	64 bits with 10 stages	32 bits with 10 stages	64 bits with 15 stages
Area(slices)/Embedded Multipliers	190/4	1024/16	180/3	838/10	220/3	1019/10
Max. Freq. (MHz) Achievable	200	130	220	175	220	215
Power(mW) at 100MHz	136.3	414	227	390	263	419

Floating-point vs. Fixed-point

- Area : 0.9x-1.2x
- Speed: 1.1x-1.4x
- Power: 1x-1.6x

A Comparison of Floating-point units

Our units vs. the units from the NEU library*

	USC 32 bits			NEU 32 bits			USC 64 bits			NEU 64 bits		
	F	A	F/A	F	A	F/A	F	A	F/A	F	A	F/A
Adder	250	551	.45	120	391	.35	200	933	.22	50	770	.07
Multiplier	250	182	1.4	95	124	0.6	205	910	.23	90	477	.18

F: Frequency
A: Slices

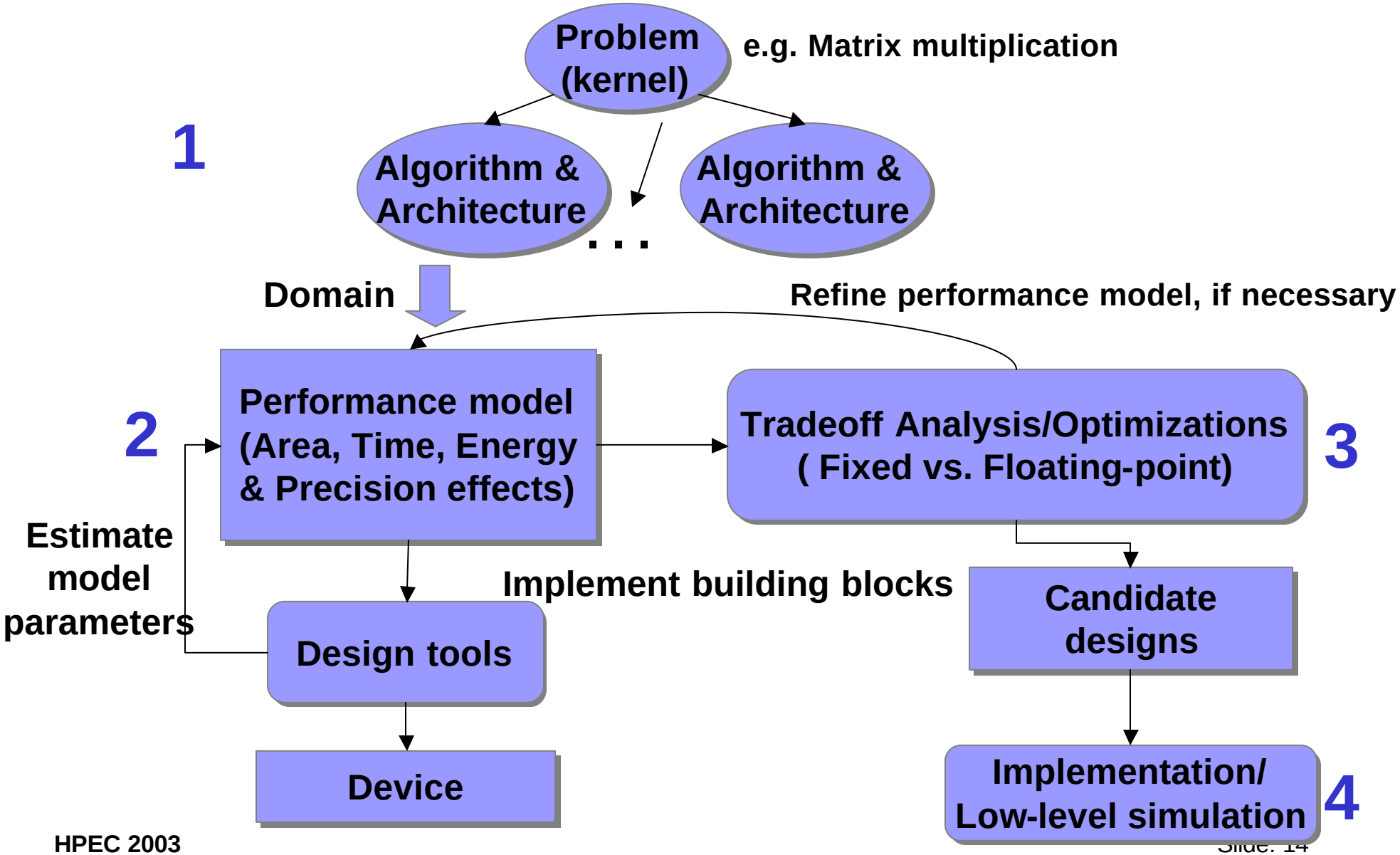
*** P. Belanovic, M. Leeser, *Library of Parameterized Floating-point Modules and Their Use*, International Conference on Field Programmable Logic (ICFPL), Sept., 2002**

USC UNIVERSITY OF SOUTHERN CALIFORNIA

Outline

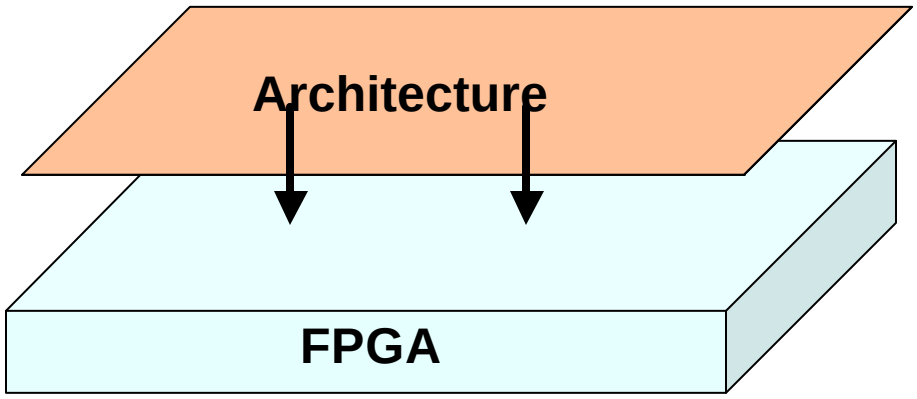
- Floating-point based Applications on FPGAs
- Floating-point Units
  - Area/Power Analysis
- Floating-point based Algorithm/Architecture Design
- Area, Power, Performance analysis for example kernels:
  - FFT
  - Matrix Multiply
- Conclusion

The Approach: Overview



1. Domain

- FPGA is too fine-grained to model at high level
  - No fixed structure comparable to that of a general purpose processor
  - Difficult to model at high level
- A family of architectures and algorithms for a given kernel or application
  - E.g. matrix multiplication on a linear array
- Imposes an architecture on FPGAs
  - Facilitates high-level modeling and high-level performance analysis



- Choose domains by analyzing algorithms and architectures for a given kernel
  - Tradeoffs in Area, Energy, Latency

USC UNIVERSITY OF SOUTHERN CALIFORNIA

2. Performance Modeling

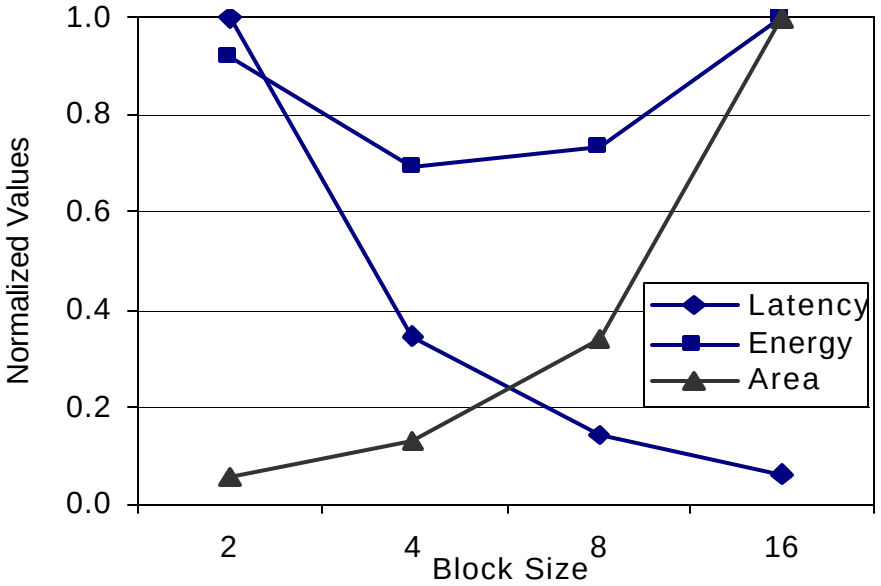
- **Domain Specific Modeling**

- **High-level model**
  - **Model parameters are specific to the domain**
  - **Design is composed based on the parameters**
  - **Design is abstracted to allow easier (but coarse) tradeoff analysis and design space exploration**
  - **Precision effects are studied**
  - **Only those parameters that make a significant impact on area and energy dissipation are identified**

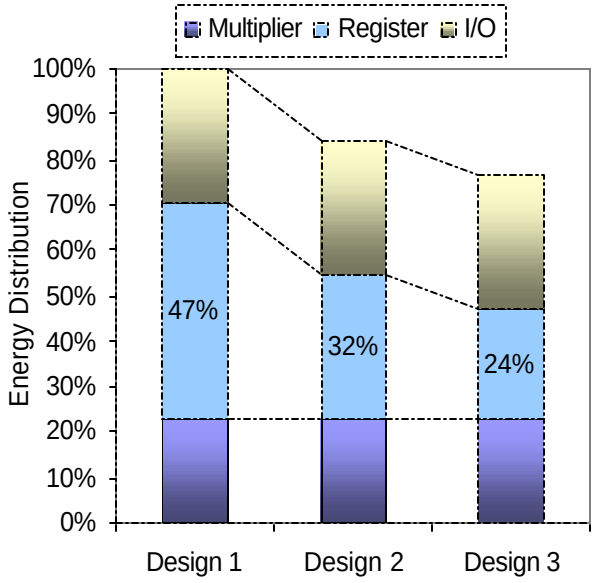
- **Benefit: Rapid evaluation of architectures and algorithms without low-level simulation**
  - **Identify candidate designs that meet requirements**

3. Tradeoff Analysis and Manual Design Space Exploration

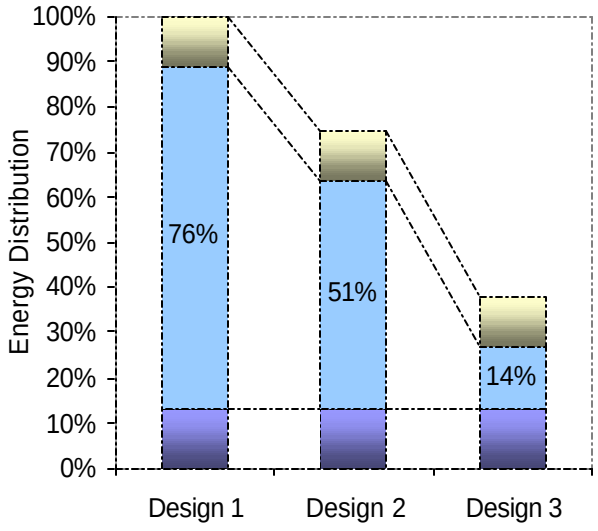
- Vary model parameters to see the effect on performance
- Analyze tradeoffs
- Weed out designs that are not promising



Example: Energy Tradeoffs



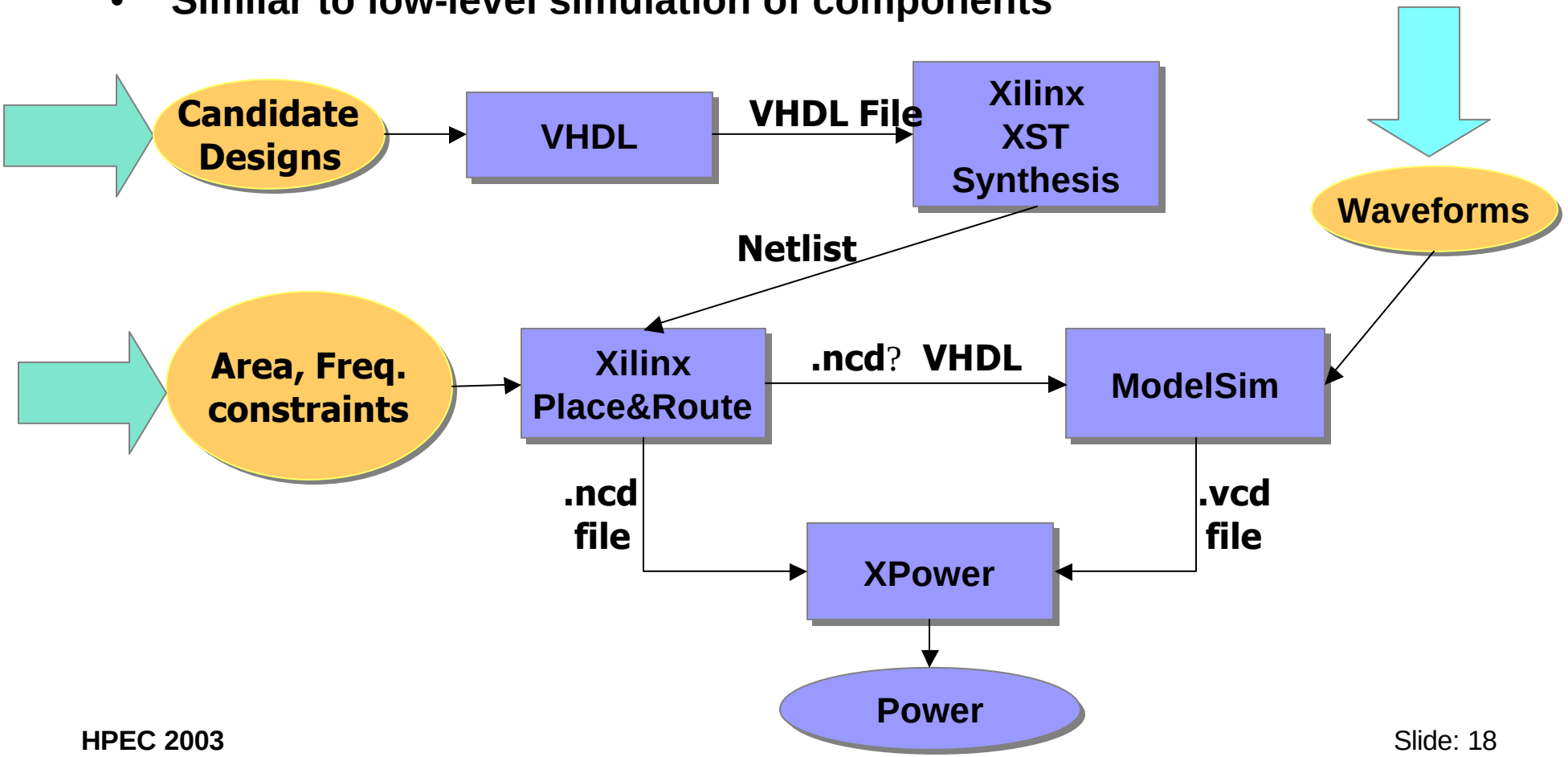
(a) 3x3



(b) 12x12

4. Low Level Simulation of Candidate Designs

- Verify high-level estimation of area and energy for a design
- Select the best design within the range of the estimation error among candidate designs
- Similar to low-level simulation of components



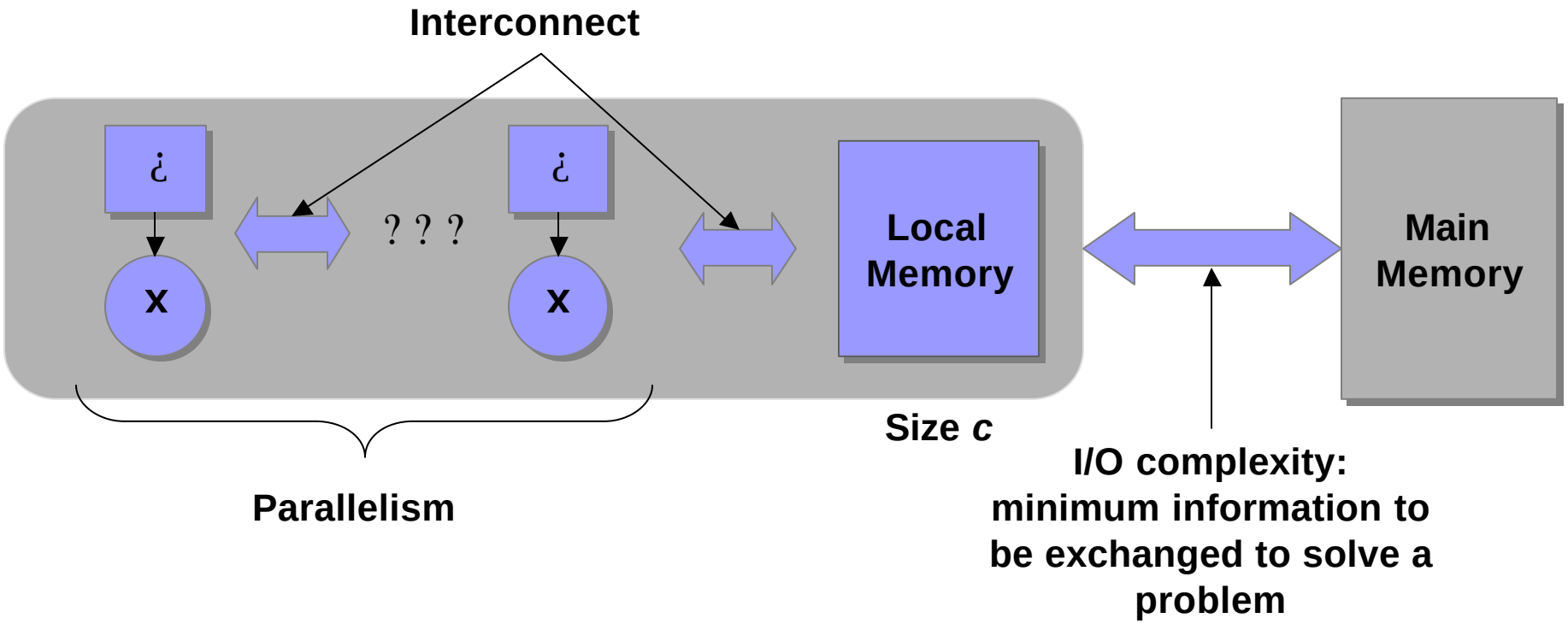
USC UNIVERSITY OF SOUTHERN CALIFORNIA

Outline

- Floating-point based Applications on FPGAs
- Floating-point Units
  - Area/Power Analysis
- Floating-point based Algorithm/Architecture Design
- Area, Power, Performance analysis for example kernels:
  - FFT
  - Matrix Multiply
- Conclusion

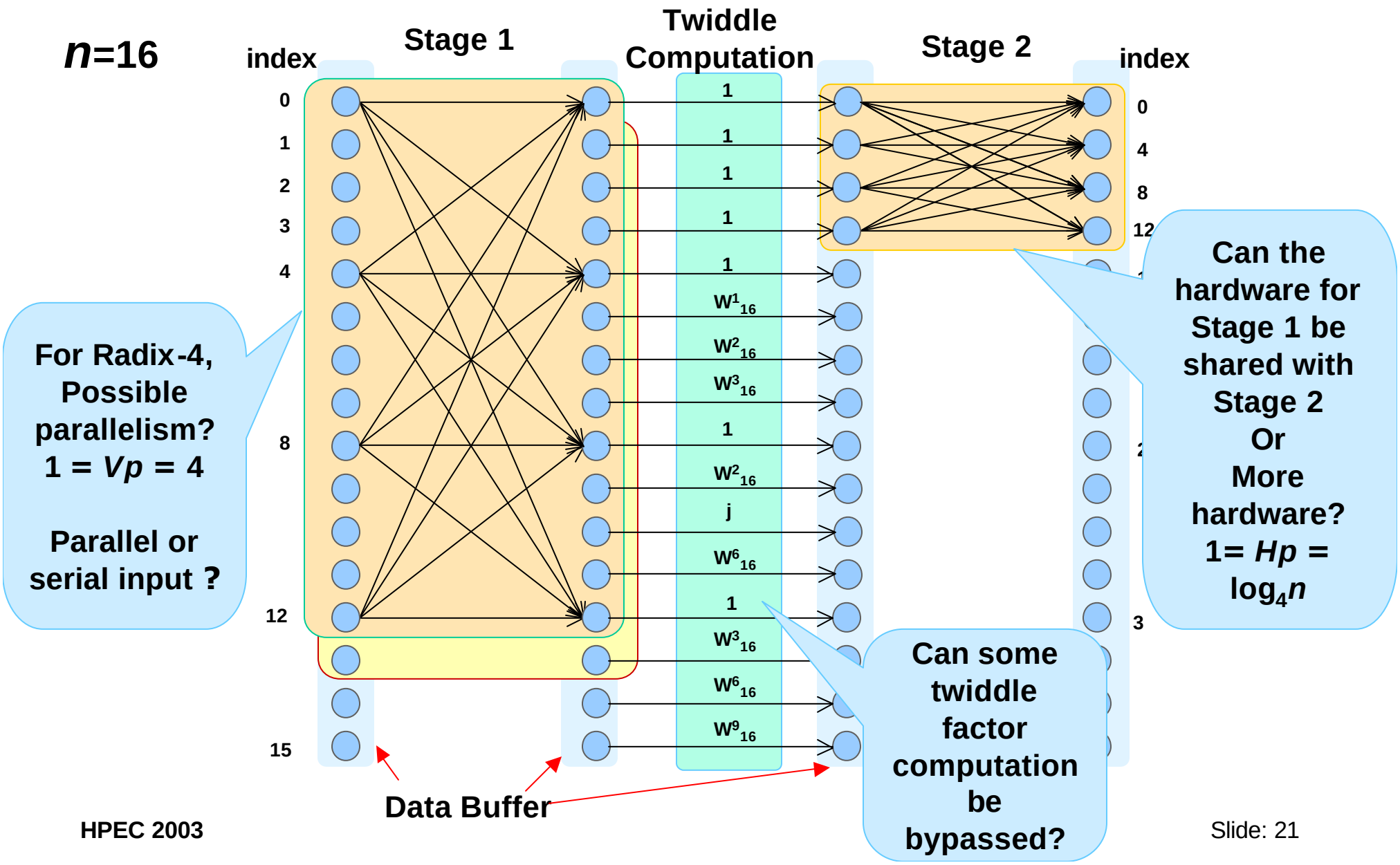
Example 1: FFT Architecture Design Tradeoffs

n-point FFT

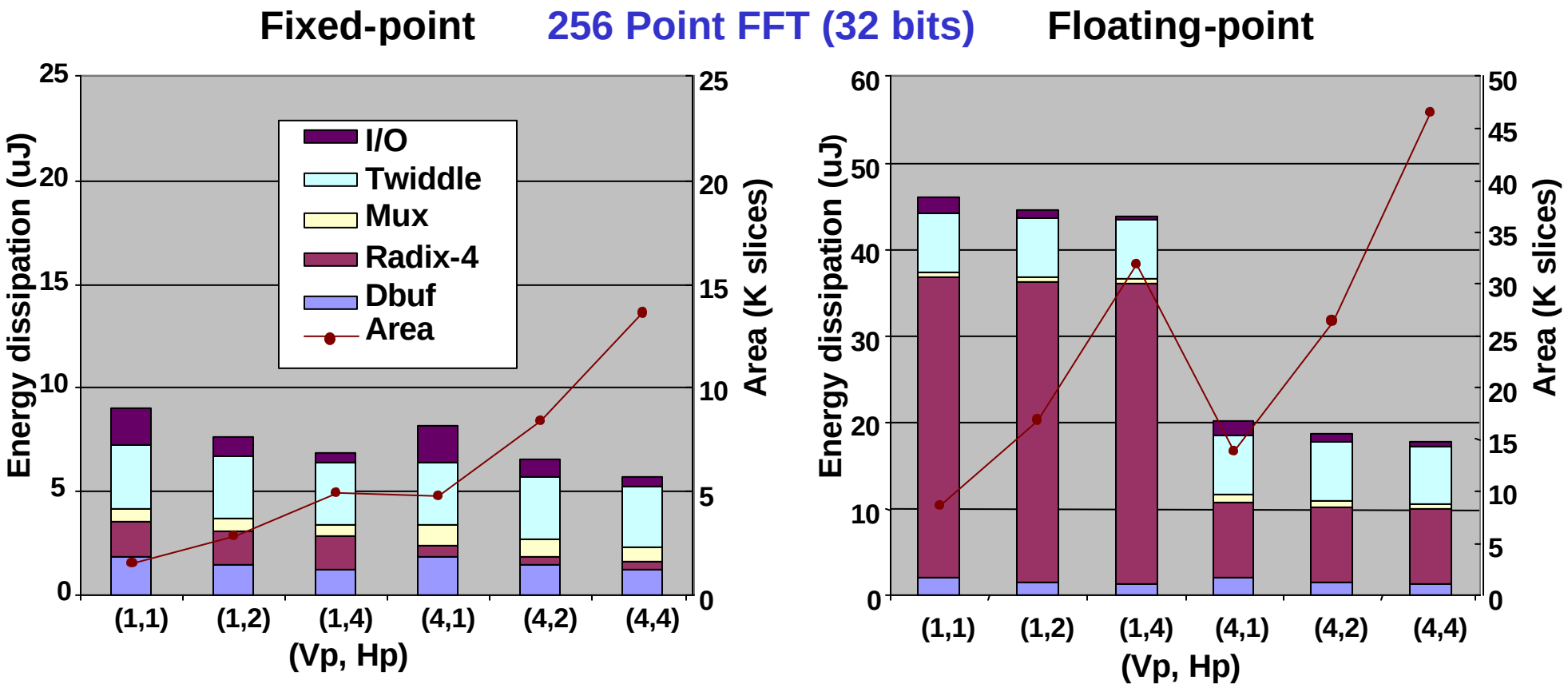


For *n*-point FFT, I/O complexity = ? (*n* log*n* / log*c*)

FFT Architecture Design Tradeoffs (2)



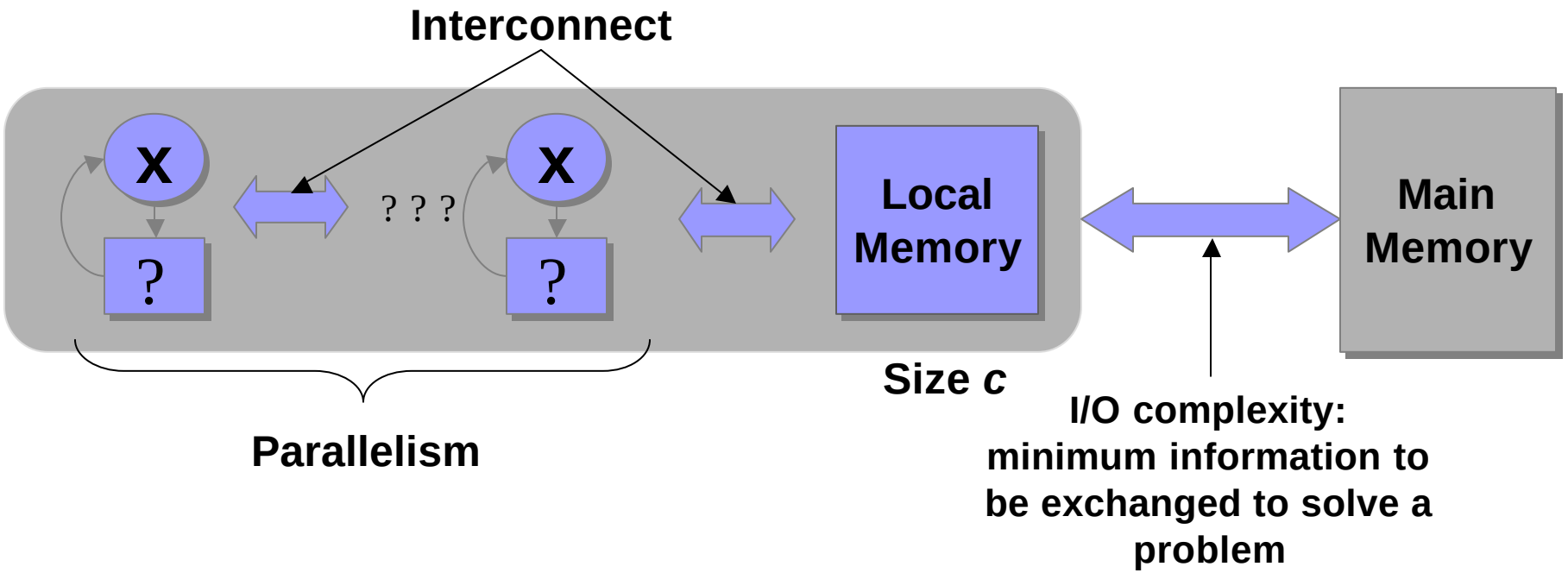
FFT Architecture Design Trade-offs (3)



- **Optimal FFT architectures with respect to EAT**
  - **Fixed-point: (Vp, Hp) = (1,4)**
  - **Floating-point: (Vp, Hp) = (4,1)**

Example 2: Matrix Multiplication Architecture Design (1)

I/O Complexity of Matrix Multiplication

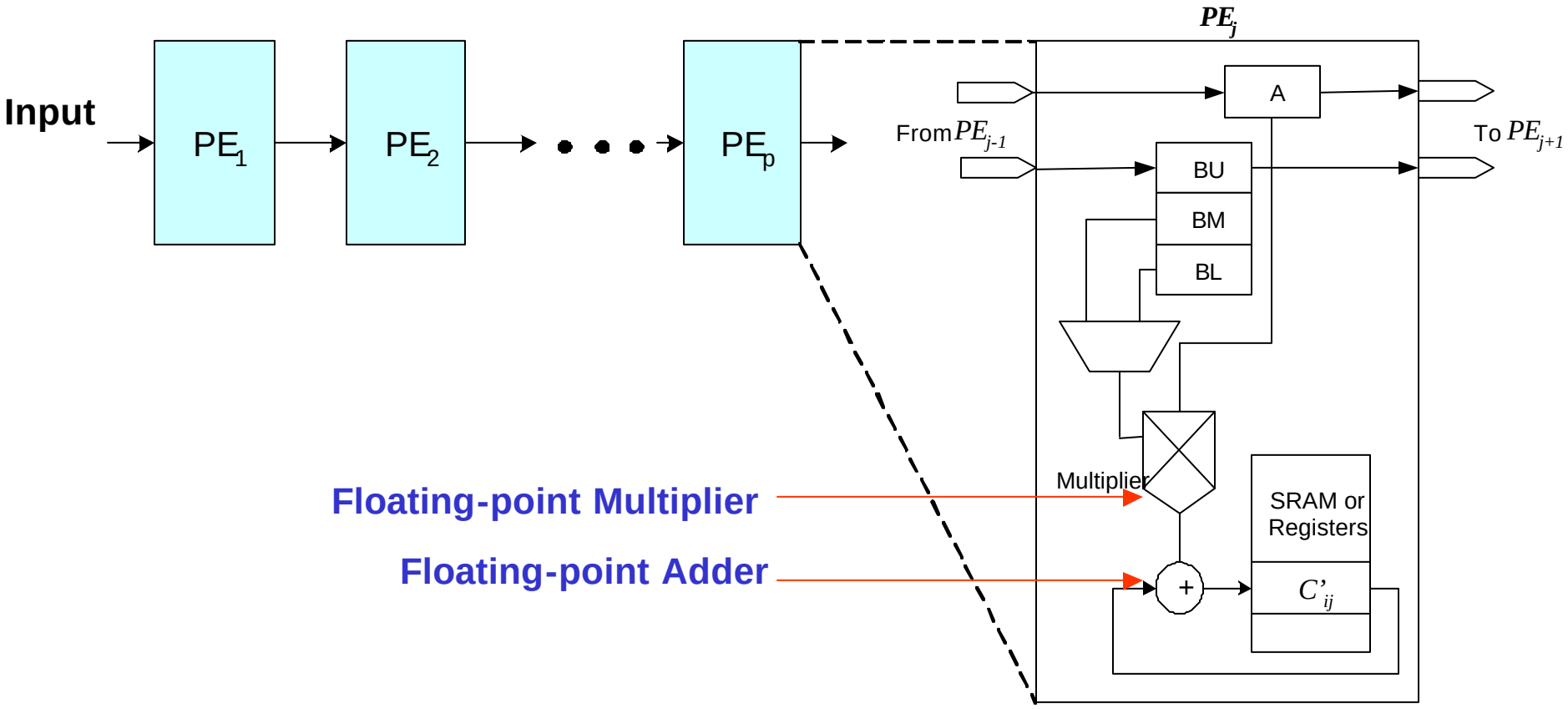


Theorem (Hong and Kung): For *n* ? *n* matrix multiplication

I/O complexity = ? (n^3/? *c*)

Matrix Multiplication Architecture Design (2)

Processing Element Architecture*



*** J. W. Jang, S. Choi, and V. K. Prasanna, *Area and Time Efficient Implementation of Matrix Multiplication on FPGAs,* ICFPT 2002.**

Matrix Multiplication Architecture Design (3)

- **Our design**
  - **Number of PEs = *n***
  - **Storage = ? (*n* ? *n*)**
  - **Latency = ? (n^2)**
- **For *n* x *n* matrix multiplication, I/O complexity = ? (n^3/? c)**
- **Our design has optimal I/O complexity**

Performance of 32, 64 bits Floating-point Matrix Multiplication (4)

Pipeline stages	32 bits XC2VP125 –7			64 bits XC2VP125 –7		
	Min	Max	Optimal	Min	Max	Optimal
Area(slices) of each Processing Element	718	991	933	1524	2575	2256
Max. No. PEs	77	56	59	36	21	24
Achievable Frequency (MHz)	90	215	210	50	190	180
Sustained Performance (GFLOPS)	13.8	24.1	24.7	3.6	8.0	8.6

The performance (in GFLOPS) is maximum for the design with floating-point units with maximum frequency/area.

FPGA vs. Processor

32 bits floating-point matrix multiplication on FPGA using our FPU and architecture

	FPGA XC2VP125 –7 230MHz	TI TMS320 C6713* 225 MHz	Analog TigerSharc * 500 MHz	Pentium 4 SSE2 * 2.53 GHz	PowerPC G4 * 1.25 GHz
GFLOPS	24.7 (sustained)	1.325 (peak)	1.0 (peak)	6.56 (peak)	6.22 (peak)
Power(W)	26	1.8 (core power)	2.4 (core power)	59.3	30
GFLOPS/W	0.95	0.7	0.4166	0.11	0.2

FPGA vs. Processor

- Performance (in GFLOPS): up to 24.7x
- Performance/Power (in GFLOPS/W): up to 8.6x

* From data sheets

USC UNIVERSITY OF SOUTHERN CALIFORNIA

FPGA vs. Processor

64 bits floating-point matrix multiplication on FPGA using our FPU and architecture

	FPGA XC2VP125 –7 200MHz	Pentium 4 SSE2 1.5 GHz*	AMD Athlon 1 GHz*
GFLOPS	8.6 (sustained)	2.0 (peak)	1.1 (peak)
Power(W)	26	54.7	60
GFLOPS/W	0.33	0.036	0.018

FPGA vs. Processor

- Performance (in GFLOPS): up to 7.8x
- Performance/Power (in GFLOPS/W): up to 18.3x

* From data sheets

USC UNIVERSITY OF SOUTHERN CALIFORNIA

Conclusion and Future Work

Conclusion

- Floating-point based implementations are not prohibitively expensive either in terms of area or latency or power
- High performance kernels can be designed with appropriate FPUs
- In terms of GFLOPS and GFLOPS/W, FPGAs offer significant over general purpose processors and DSPs

Future Work

- Floating-point based beamforming….
- Tool for automatic integration of FPUs into kernels

http://ceng.usc.edu/~prasanna

MILAN for System-Level Design: Design Flow

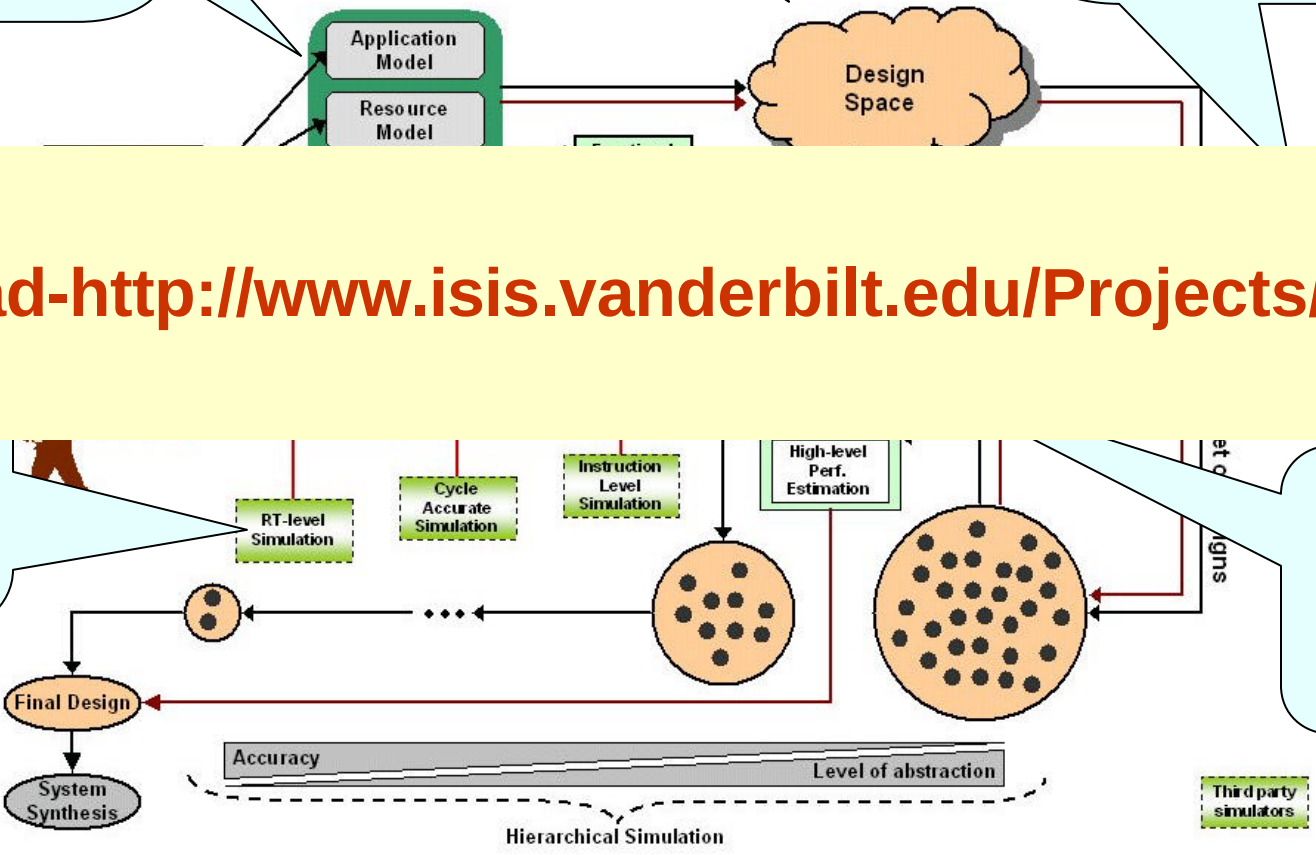
Model PARIS kernels, end-to-end application, hardware choices, mission parameters, etc.

PARIS design space

Dynamic programming based heuristics

Multi-rate application optimization

Interval arithmetic



Download-http://www.isis.vanderbilt.edu/Projects/milan/

VHDL and C implementations

Energy, latency, and area estimates

Enhanced HiPerE

High-level estimator for FPGAs

USC
UNIVERSITY
OF SOUTHERN
CALIFORNIA

Questions?

http://ceng.usc.edu/~prasanna