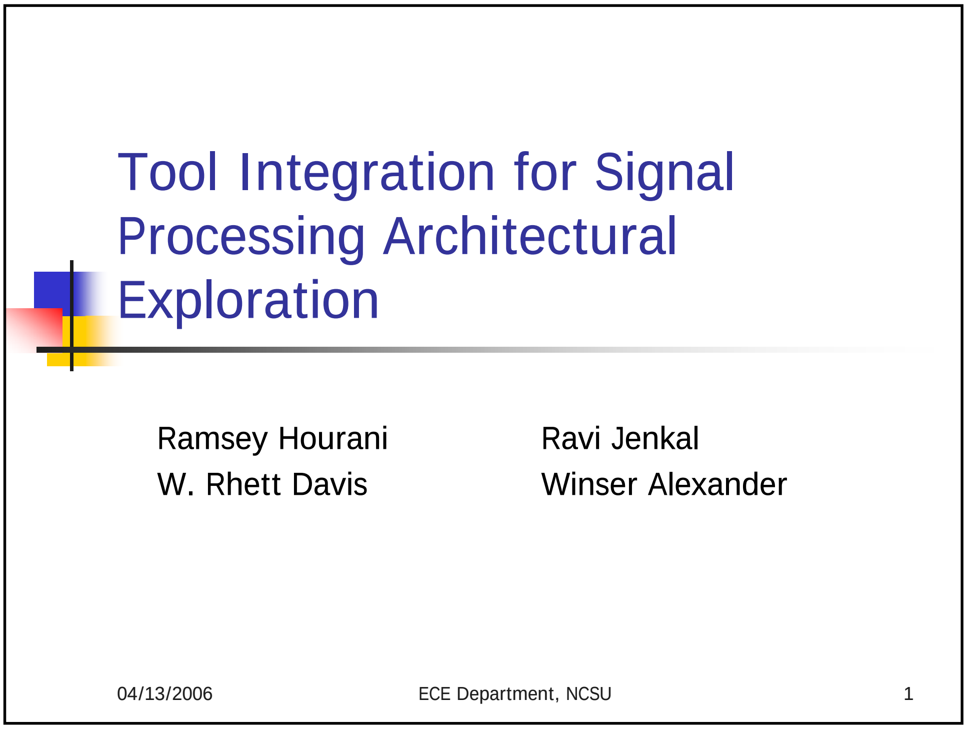
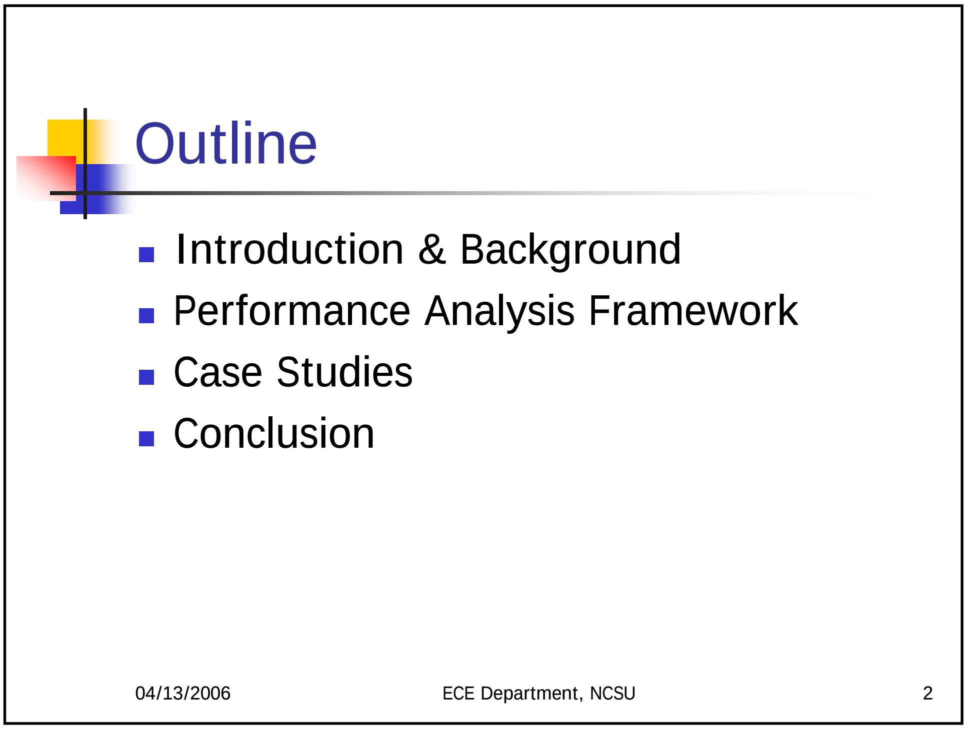


Tool Integration for Signal Processing Architectural Exploration

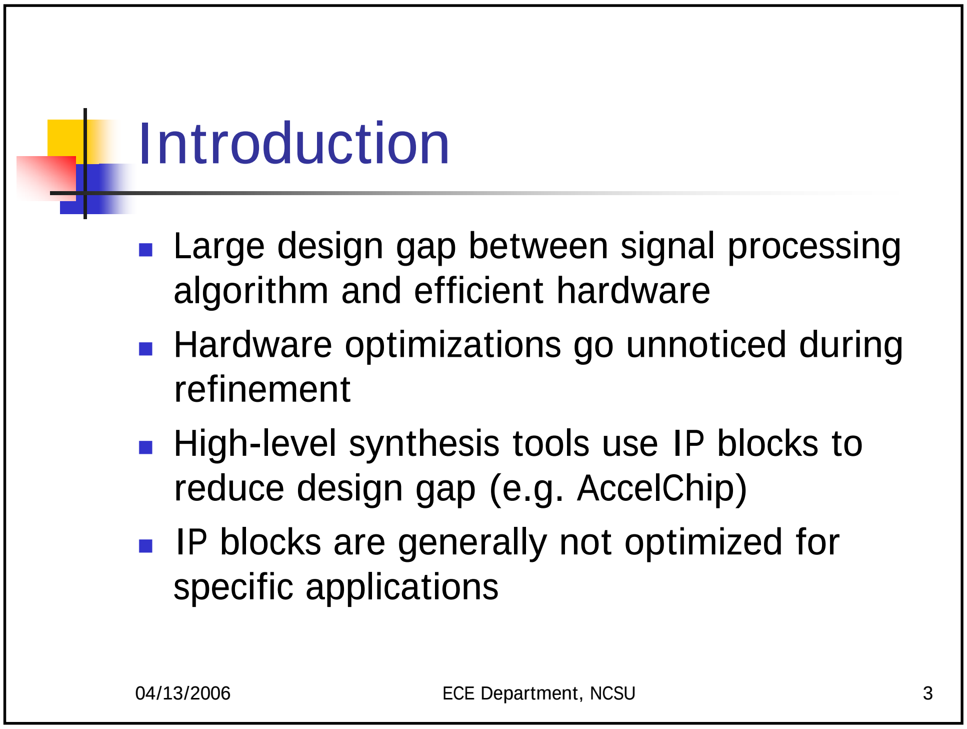
Ramsey Hourani
W. Rhett Davis

Ravi Jenkal
Winser Alexander



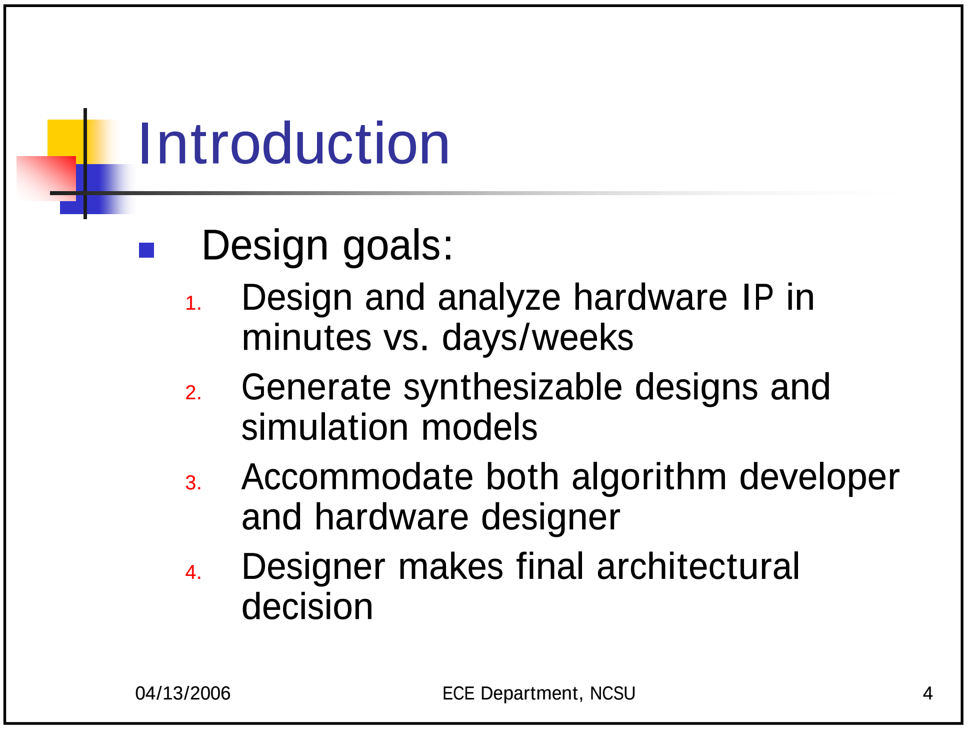
Outline

- Introduction & Background
- Performance Analysis Framework
- Case Studies
- Conclusion



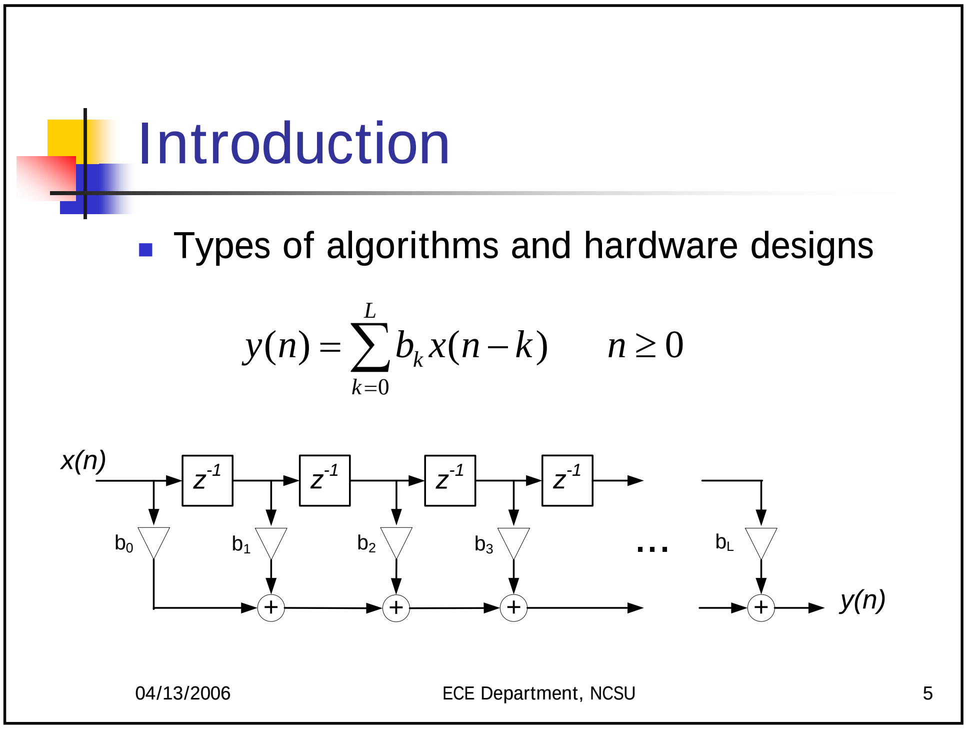
Introduction

- Large design gap between signal processing algorithm and efficient hardware
- Hardware optimizations go unnoticed during refinement
- High-level synthesis tools use IP blocks to reduce design gap (e.g. AccelChip)
- IP blocks are generally not optimized for specific applications



Introduction

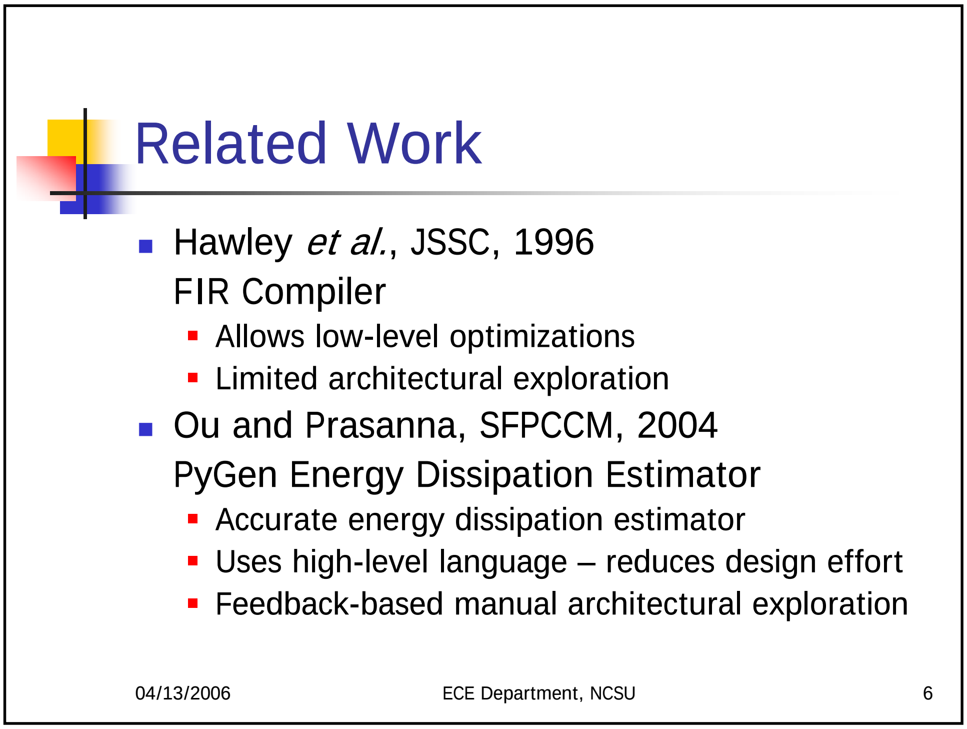
- Design goals:
  1. Design and analyze hardware IP in minutes vs. days/weeks
  2. Generate synthesizable designs and simulation models
  3. Accommodate both algorithm developer and hardware designer
  4. Designer makes final architectural decision



Introduction

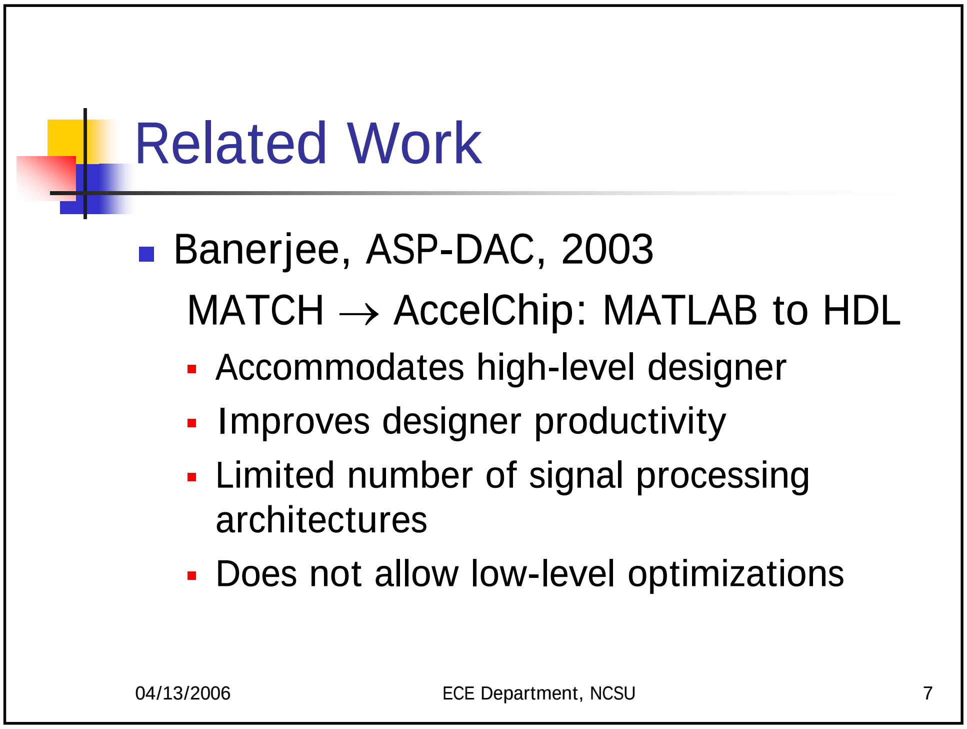
- Types of algorithms and hardware designs

$$y(n)=\sum_{k=0}^{L} b_k x(n-k) \qquad n \ge 0$$



Related Work

- Hawley *et al.*, JSSC, 1996
  FIR Compiler
  - Allows low-level optimizations
  - Limited architectural exploration
- Ou and Prasanna, SFPCCM, 2004
  PyGen Energy Dissipation Estimator
  - Accurate energy dissipation estimator
  - Uses high-level language – reduces design effort
  - Feedback-based manual architectural exploration

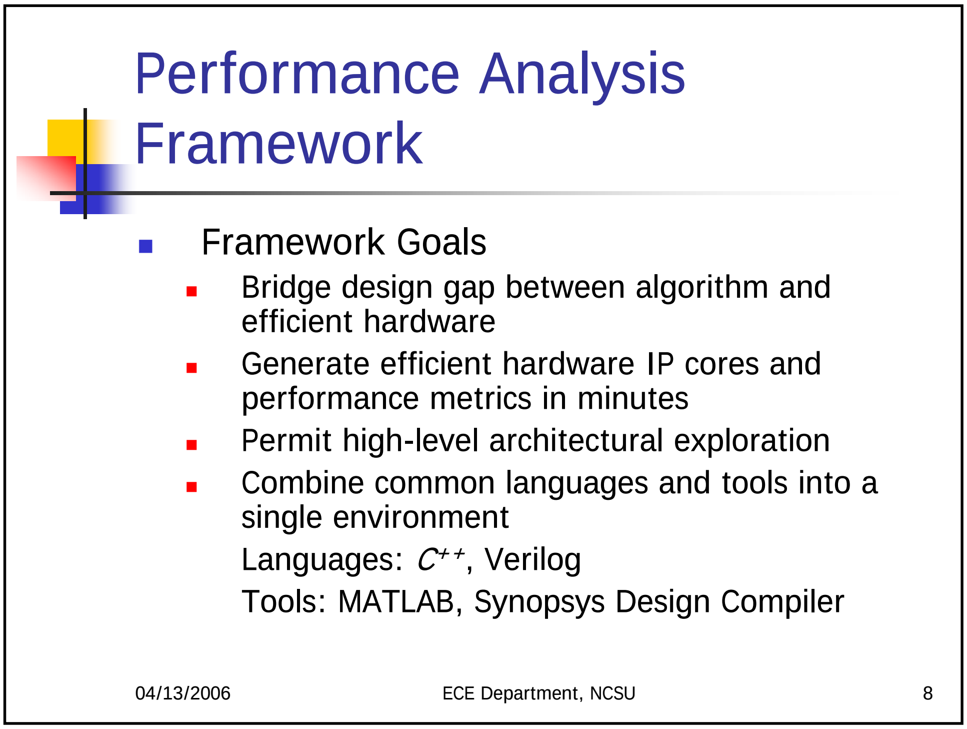


Related Work

- Banerjee, ASP-DAC, 2003

  MATCH → AccelChip: MATLAB to HDL

  - Accommodates high-level designer
  - Improves designer productivity
  - Limited number of signal processing architectures
  - Does not allow low-level optimizations

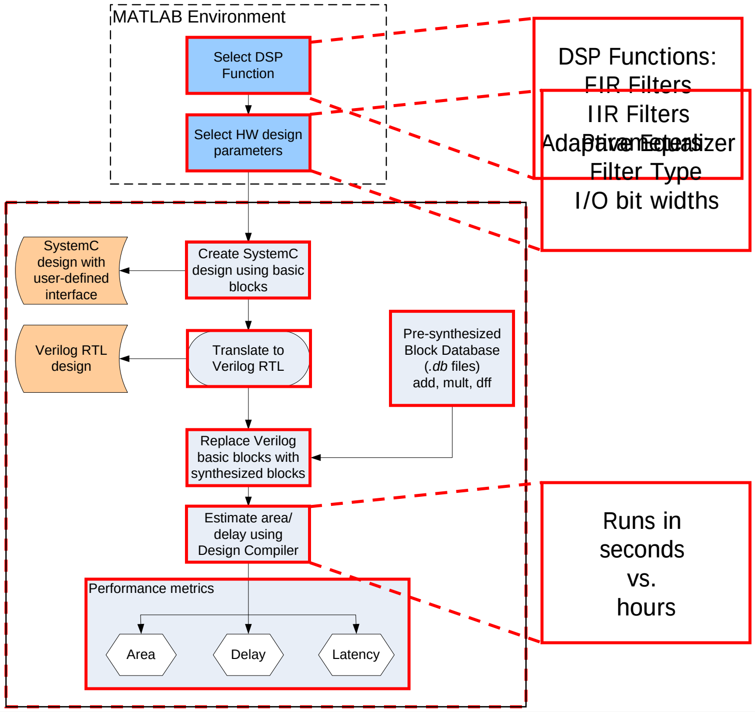


Performance Analysis Framework

- Framework Goals
  - Bridge design gap between algorithm and efficient hardware
  - Generate efficient hardware IP cores and performance metrics in minutes
  - Permit high-level architectural exploration
  - Combine common languages and tools into a single environment

    Languages: *C++*, Verilog

    Tools: MATLAB, Synopsys Design Compiler



MATLAB Environment

Select DSP Function

Select HW design parameters

SystemC design with user-defined interface

Create SystemC design using basic blocks

Verilog RTL design

Translate to Verilog RTL

Pre-synthesized Block Database (*.db* files) add, mult, dff

Replace Verilog basic blocks with synthesized blocks

Estimate area/ delay using Design Compiler

Performance metrics

Area

Delay

Latency

DSP Functions:
FIR Filters
IIR Filters
Adaptive Equalizer

Parameters
Filter Type
I/O bit widths

Runs in seconds vs. hours

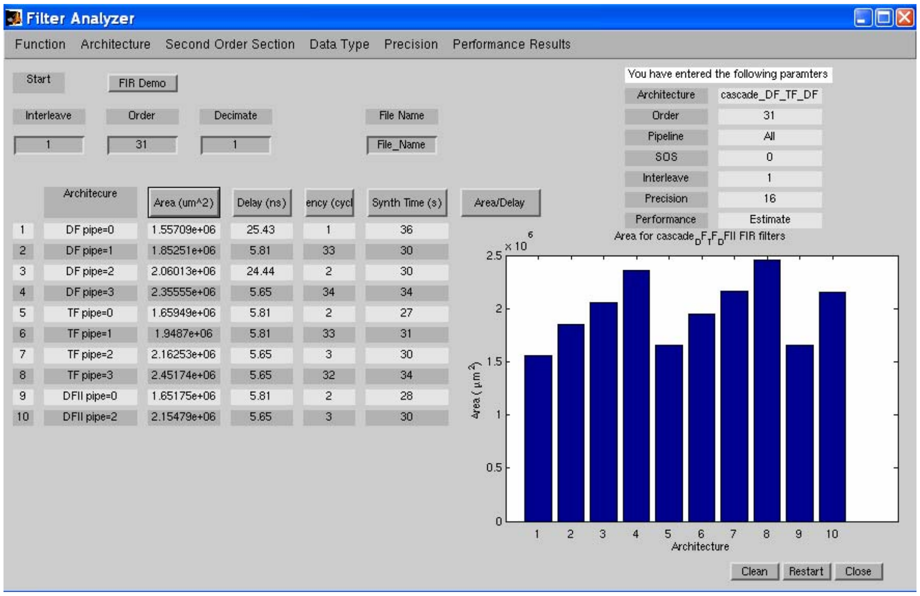
Case Studies

- FIR Filter: 32-tap 16-bit
- Adaptive Channel Equalizer
- Goals:
  - IP architectural exploration
  - Minimize design effort and analysis time

Case Study 1: FIR Filter

- Input Parameters:
  - Filter type: FIR Filter
  - Filter order: 31
  - Input/Output bit-width: 16/32 bits
- Framework automatically generates FIR filter structures with performance metrics
  - Direct Form/ Transpose Form
  - Different levels of pipelining

Framework GUI



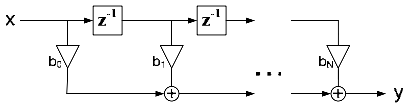
Interleave	Order	Decimate	File Name
1	31	1	File_Name

	Architecure	Area (um^2)	Delay (ns)	ency (cycl	Synth Time (s)
1	DF pipe=0	1.55709e+06	25.43	1	36
2	DF pipe=1	1.85251e+06	5.81	33	30
3	DF pipe=2	2.06013e+06	24.44	2	30
4	DF pipe=3	2.35555e+06	5.65	34	34
5	TF pipe=0	1.65949e+06	5.81	2	27
6	TF pipe=1	1.9487e+06	5.81	33	31
7	TF pipe=2	2.16253e+06	5.65	3	30
8	TF pipe=3	2.45174e+06	5.65	32	34
9	DFII pipe=0	1.65175e+06	5.81	2	28
10	DFII pipe=2	2.15479e+06	5.65	3	30

You have entered the following paramters

Architecture	cascade_DF_TF_DF
Order	31
Pipeline	All
SOS	0
Interleave	1
Precision	16
Performance	Estimate

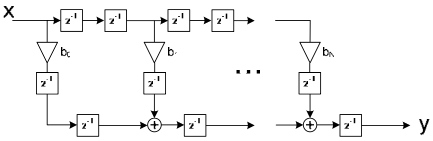
Arch 1:



No. of Mult: N+1
No. of DFF: N
Delay: $N \text{x} T_{add} + T_{mult}$

No. of Add: N
Latency: 1 clk

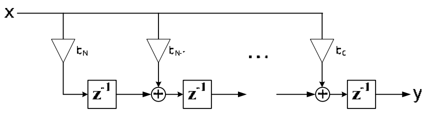
Arch 2:



No. of Mult: N+1
No. of DFF: 4xN
Delay: T_{mult}

No. of Add: N
Latency: N clk

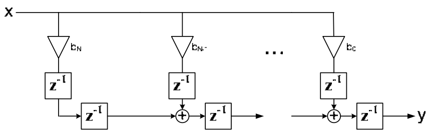
Arch 5:



No. of Mult: N+1
No. of DFF: N
Delay: $T_{add} + T_{mult}$

No. of Add: N
Latency: 1 clk

Arch 7:



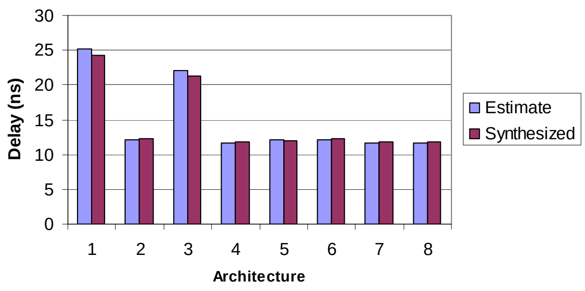
No. of Mult: N+1
No. of DFF: N
Delay: T_{mult}

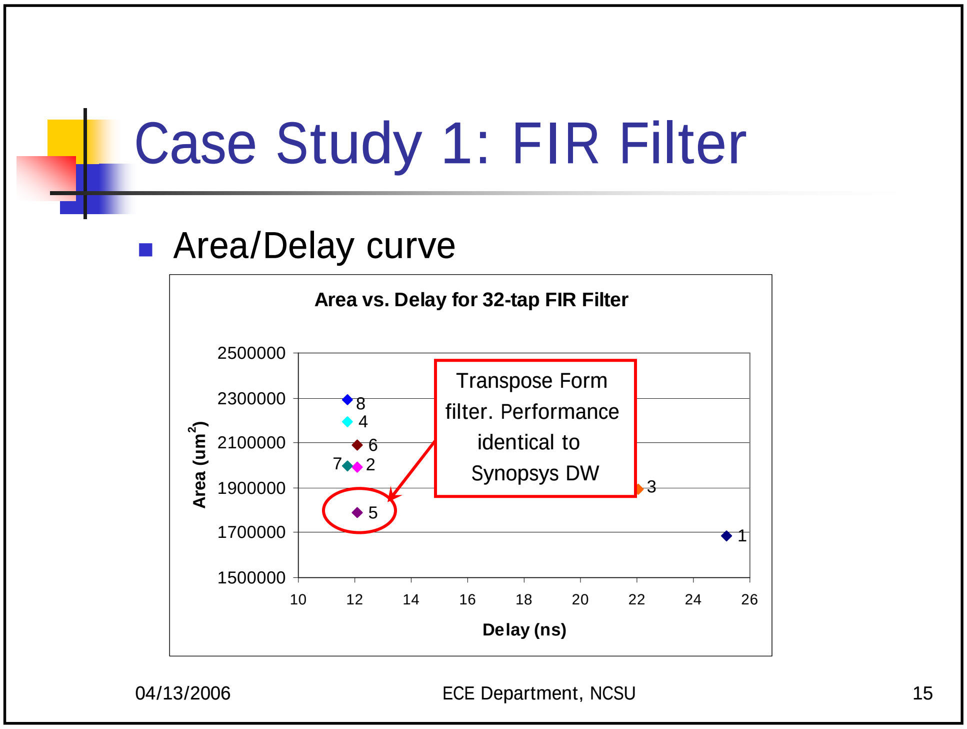
No. of Add: N
Latency: 2 clk

Case Study 1: FIR Filter

- Estimated Metrics vs. Synthesized

Estimated vs. Synthesized Delay for 32-tap FIR filter



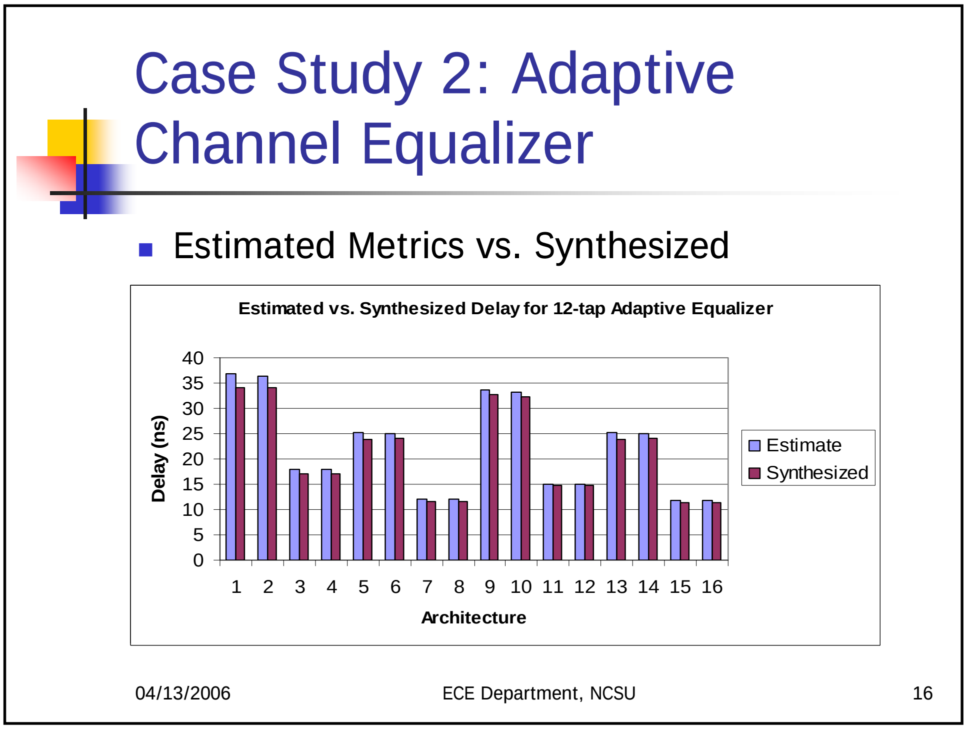


Case Study 1: FIR Filter

- Area/Delay curve

Area vs. Delay for 32-tap FIR Filter

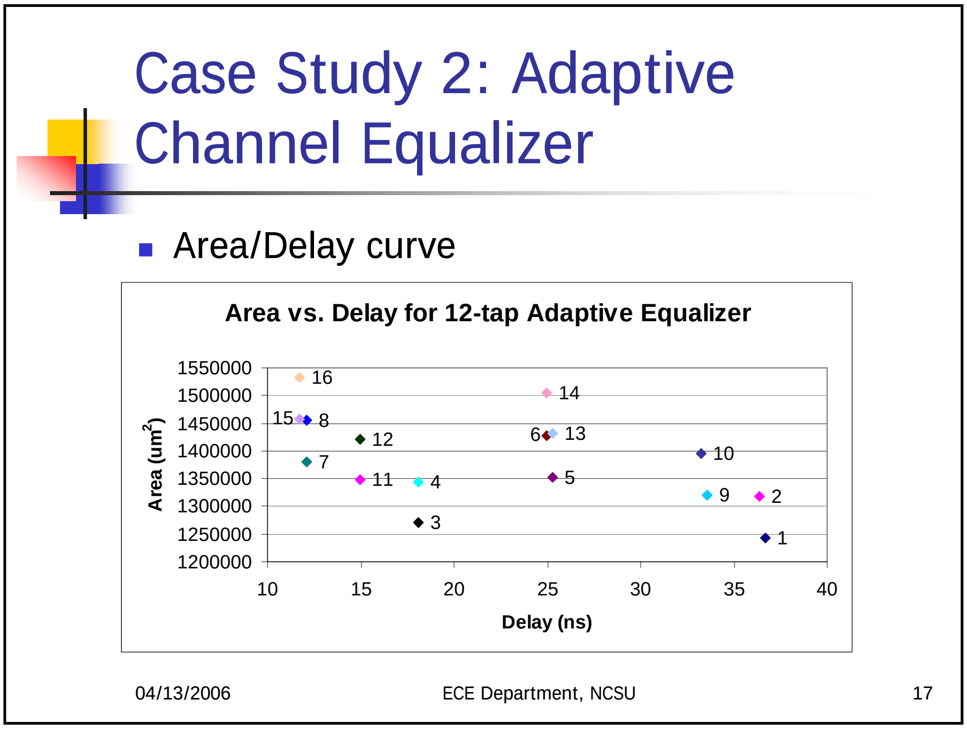
Transpose Form filter. Performance identical to Synopsys DW



Case Study 2: Adaptive Channel Equalizer

- Estimated Metrics vs. Synthesized

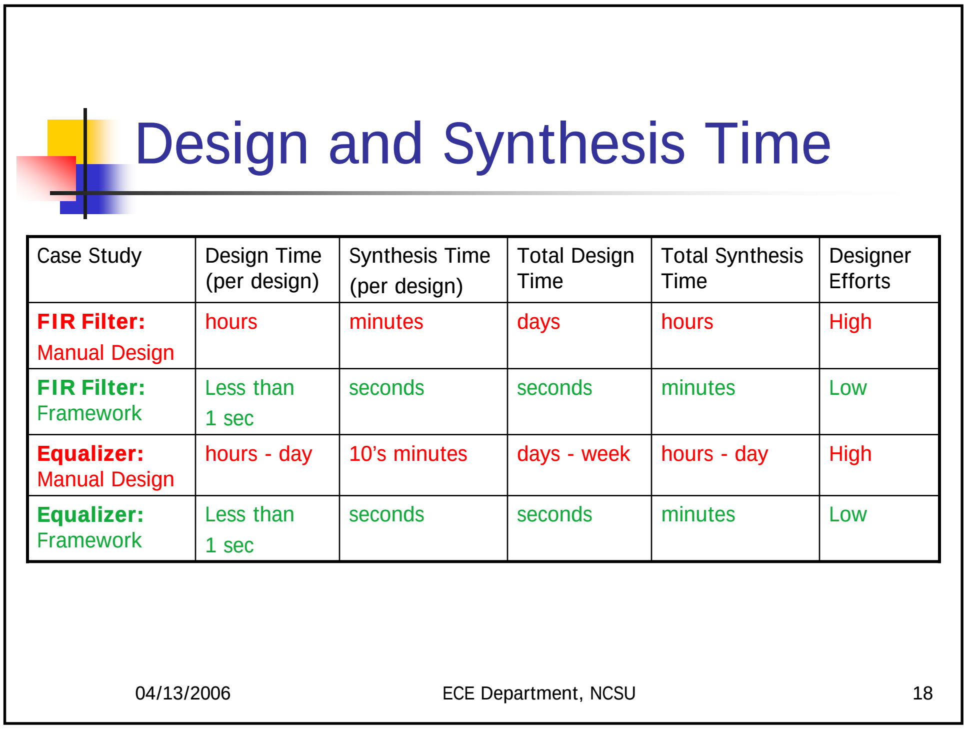
Estimated vs. Synthesized Delay for 12-tap Adaptive Equalizer



Case Study 2: Adaptive Channel Equalizer

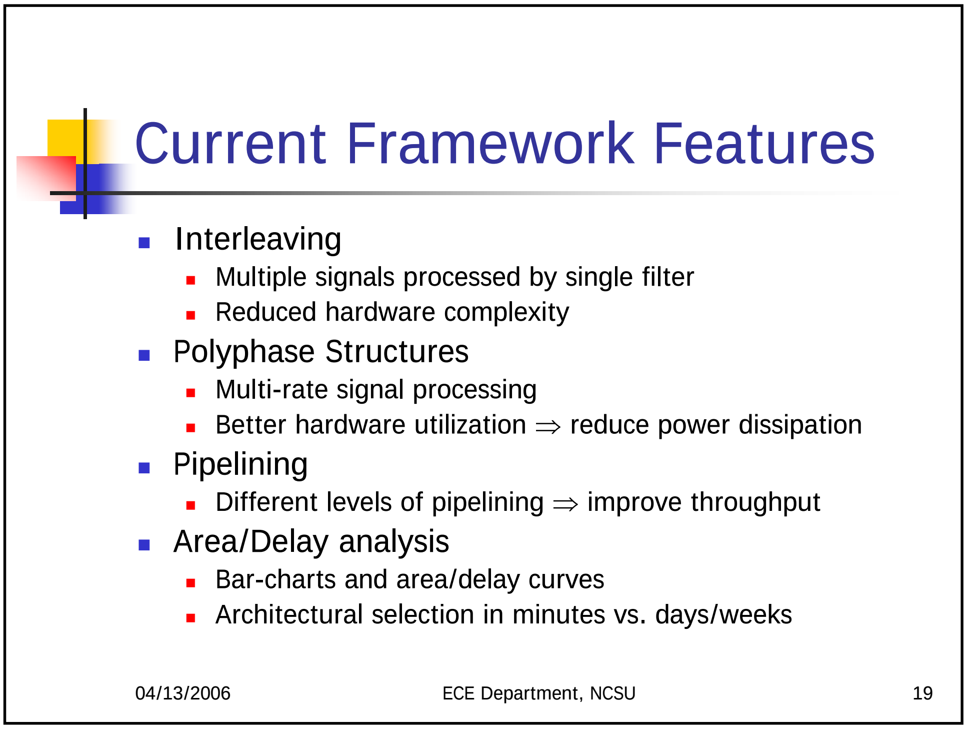
- Area/Delay curve

Area vs. Delay for 12-tap Adaptive Equalizer



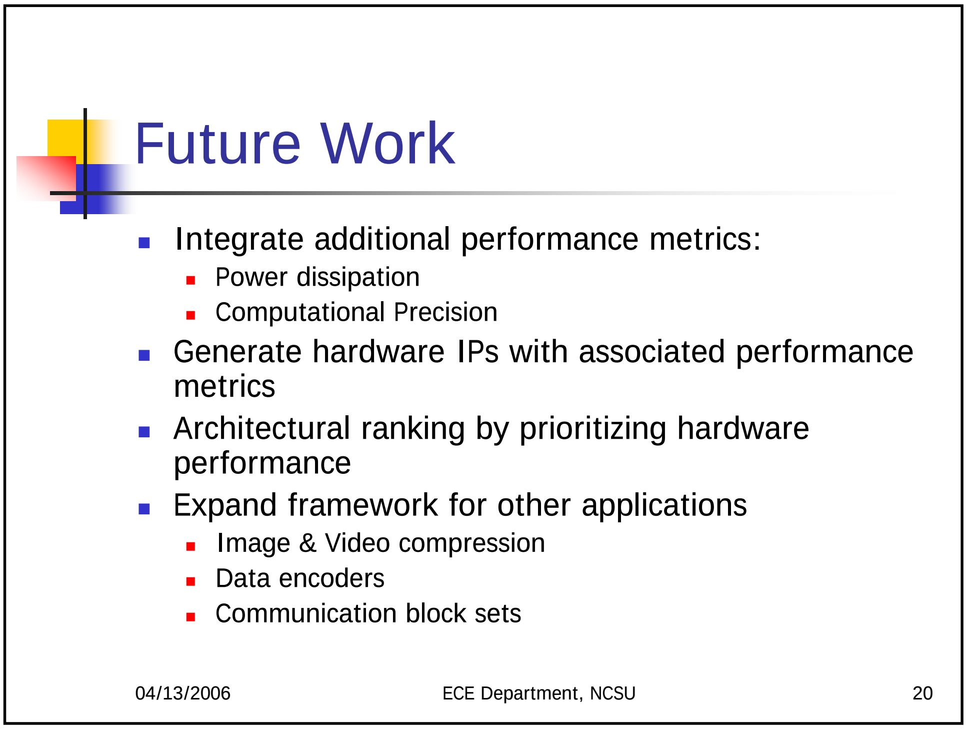
Design and Synthesis Time

Case Study	Design Time (per design)	Synthesis Time (per design)	Total Design Time	Total Synthesis Time	Designer Efforts
FIR Filter: Manual Design	hours	minutes	days	hours	High
FIR Filter: Framework	Less than 1 sec	seconds	seconds	minutes	Low
Equalizer: Manual Design	hours - day	10's minutes	days - week	hours - day	High
Equalizer: Framework	Less than 1 sec	seconds	seconds	minutes	Low



Current Framework Features

- Interleaving
  - Multiple signals processed by single filter
  - Reduced hardware complexity
- Polyphase Structures
  - Multi-rate signal processing
  - Better hardware utilization ⇒ reduce power dissipation
- Pipelining
  - Different levels of pipelining ⇒ improve throughput
- Area/Delay analysis
  - Bar-charts and area/delay curves
  - Architectural selection in minutes vs. days/weeks



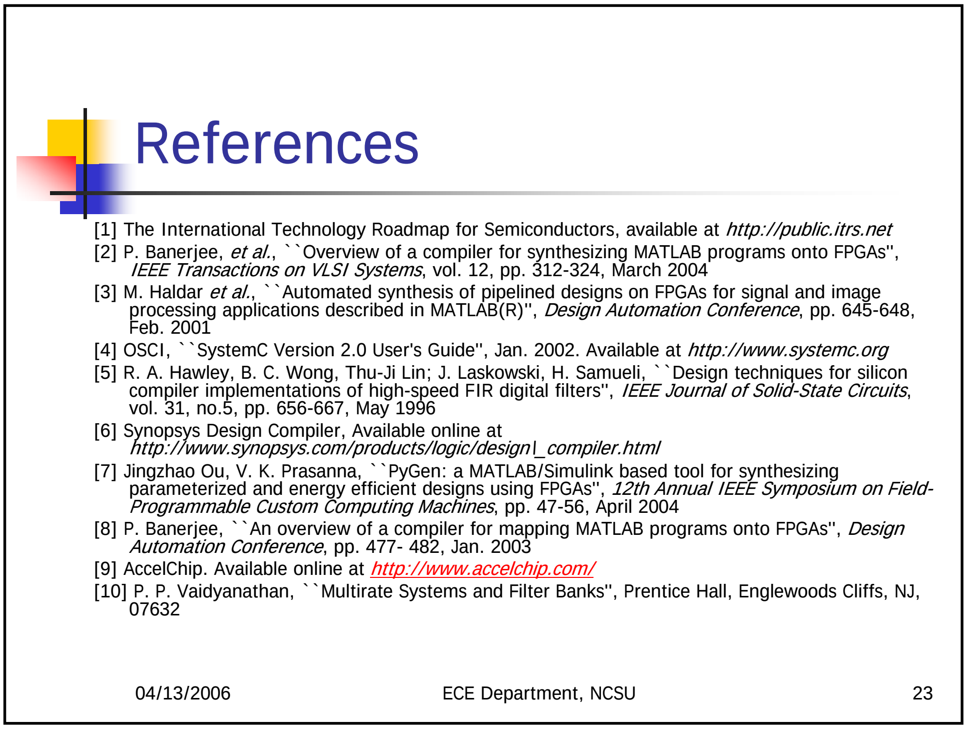
Future Work

- Integrate additional performance metrics:
  - Power dissipation
  - Computational Precision
- Generate hardware IPs with associated performance metrics
- Architectural ranking by prioritizing hardware performance
- Expand framework for other applications
  - Image & Video compression
  - Data encoders
  - Communication block sets

Conclusion

- Reduce design gap between algorithm and hardware
- Framework for generating IP cores and analyzing performance
- Reduce design and analysis times
- Accommodate users with different hardware expertise

Questions?



References

[1] The International Technology Roadmap for Semiconductors, available at *http://public.itrs.net*

[2] P. Banerjee, *et al.*, ``Overview of a compiler for synthesizing MATLAB programs onto FPGAs'', *IEEE Transactions on VLSI Systems*, vol. 12, pp. 312-324, March 2004

[3] M. Haldar *et al.*, ``Automated synthesis of pipelined designs on FPGAs for signal and image processing applications described in MATLAB(R)'', *Design Automation Conference*, pp. 645-648, Feb. 2001

[4] OSCI, ``SystemC Version 2.0 User's Guide'', Jan. 2002. Available at *http://www.systemc.org*

[5] R. A. Hawley, B. C. Wong, Thu-Ji Lin; J. Laskowski, H. Samueli, ``Design techniques for silicon compiler implementations of high-speed FIR digital filters'', *IEEE Journal of Solid-State Circuits*, vol. 31, no.5, pp. 656-667, May 1996

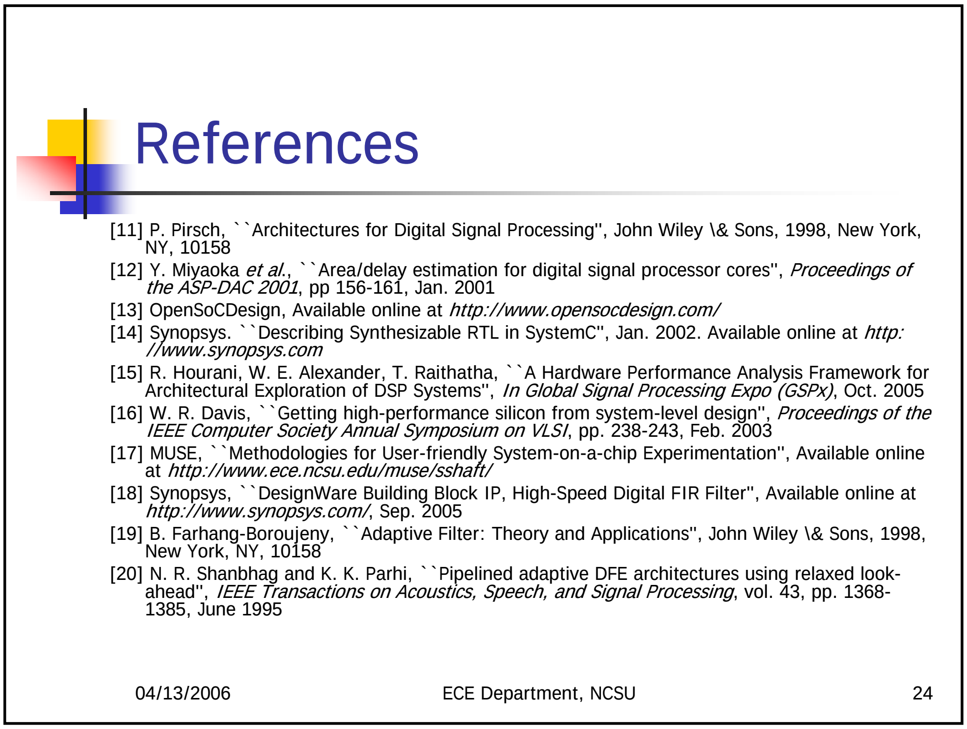
[6] Synopsys Design Compiler, Available online at *http://www.synopsys.com/products/logic/design_compiler.html*

[7] Jingzhao Ou, V. K. Prasanna, ``PyGen: a MATLAB/Simulink based tool for synthesizing parameterized and energy efficient designs using FPGAs'', *12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pp. 47-56, April 2004

[8] P. Banerjee, ``An overview of a compiler for mapping MATLAB programs onto FPGAs'', *Design Automation Conference*, pp. 477- 482, Jan. 2003

[9] AccelChip. Available online at *http://www.accelchip.com/*

[10] P. P. Vaidyanathan, ``Multirate Systems and Filter Banks'', Prentice Hall, Englewoods Cliffs, NJ, 07632



References

[11] P. Pirsch, ``Architectures for Digital Signal Processing'', John Wiley \& Sons, 1998, New York, NY, 10158

[12] Y. Miyaoka *et al.*, ``Area/delay estimation for digital signal processor cores'', *Proceedings of the ASP-DAC 2001*, pp 156-161, Jan. 2001

[13] OpenSoCDesign, Available online at *http://www.opensocdesign.com/*

[14] Synopsys. ``Describing Synthesizable RTL in SystemC'', Jan. 2002. Available online at *http://www.synopsys.com*

[15] R. Hourani, W. E. Alexander, T. Raithatha, ``A Hardware Performance Analysis Framework for Architectural Exploration of DSP Systems'', *In Global Signal Processing Expo (GSPx)*, Oct. 2005

[16] W. R. Davis, ``Getting high-performance silicon from system-level design'', *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, pp. 238-243, Feb. 2003

[17] MUSE, ``Methodologies for User-friendly System-on-a-chip Experimentation'', Available online at *http://www.ece.ncsu.edu/muse/sshaft/*

[18] Synopsys, ``DesignWare Building Block IP, High-Speed Digital FIR Filter'', Available online at *http://www.synopsys.com/*, Sep. 2005

[19] B. Farhang-Boroujeny, ``Adaptive Filter: Theory and Applications'', John Wiley \& Sons, 1998, New York, NY, 10158

[20] N. R. Shanbhag and K. K. Parhi, ``Pipelined adaptive DFE architectures using relaxed look-ahead'', *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 43, pp. 1368-1385, June 1995