DAT110
Methods for Electronic System Design and Verification

Per Larsson-Edefors
VLSI Research Group

Lecture 3:
Term Paper Information.
Practical Advice for Lab Exercises.

Term paper information

Term Paper Work - Overview

- In depth studies of selected topic;
  using sources like textbooks and research papers.
- Group work.
- Lecture and workshop on technical writing (Anne Hsu Nilsson).
  - The writing training will prove useful in your future career,
    not least in the EESD project and in the MSc thesis work.
- Oral presentation of selected topic.
- Written presentation of selected topic.
  - Resubmission possible.
  - Final deadline for term paper, Friday Jan. 11, 2019.

Why Practice Presentation?

11.5 Design of Efficient Microarchitectures

Konferenz 3 1400 - 1530

Chair: **Dionisios Pnevmatikatos,** Technical University of Crete, GR
Co-Chair: **Todd Austin,** University of Michigan, US

The microarchitecture session presents innovative ideas for the efficient design of computing components. The first paper presents a viable prediction technique to deactivate cache ways in order to save energy without compromising performance. The second paper proposes a micro-architectural extension for approximate computing that reduces bit-error-rate while providing the power benefits of extreme voltage scaling techniques. The third paper presents a faster and accurate version of logarithmic number unit (LNU) design and implementation using a co-transformation scheme.

1400 **PRACTICAL WAY HALTING BY SPECULATIVELY ACCESSING HALT TAGS**
Speaker: Daniel Moreau, Chalmers University of Technology, SE
Authors: Daniel Moreau[1], Alen Bardizbanyan[1], Magnus Själander[2], Dave Whalley[3] and Per Larsson-Edefors[1]
[1]Chalmers University of Technology, SE; [2]Uppsala University, SE; [3]Florida State University, US

1430 **LAZY PIPELINES: ENHANCING QUALITY IN APPROXIMATE COMPUTING**
Speaker: Georgios Tziantzioulis, Northwestern University, US

Oral presentation under time pressure is a central skill today.

Term Paper Work Flow 1(2)

Phase 1:

- Select technical topic with two fellow students.

Phase 2:

- Browse relevant portions of book, chapter and/or papers. Remember to take notes as you read the material.
- Distribute within the group what parts to read in detail...

Phase 3:

- ... read your sources in more detail.
- Use CHANS, IEEE Xplore and ACM Digital Library to identify more sources (study the bibliographies!).

Term Paper Work Flow 2(2)

Phase 4:

- Based on the complete set of sources, write a draft version.
- Participate in the writing workshop Tuesday Dec. 18.

Phase 5:

- Prepare a 15-minute oral presentation of the term paper topic.
- Presentations: mid December.

Phase 6:

- Complete term paper based on feedback from oral presentation and writing workshop.
- Submit term paper on PingPong no later than Jan. 11, 2019.

Get Started

- Consider the topics and make your selection ...
  - no later than Lecture 4, Friday Nov. 16.
  - sign up on the paper that I bring.
- There is a LaTeX quick start guide on PingPong, in case you want to use LaTeX to write your documents.
  - There is this software called Word out there. If you intend to work with figures, stay away from Word!
- Remember I have one consultation hour on Mondays 1-2pm and one on Tuesdays 1-2pm (any exceptions listed on PP).

Oral Presentation

- At the term paper seminar ...
  - bring presentation on a laptop.
  - deliver the presentation.
  - answer questions during a 5-minute Q&A.

- *Oral presentation guidelines will be given in a later lecture.*

Term Paper

- Each group writes a term paper. Start early!
  - Consider the term paper work flow mentioned earlier.
  - Make sure you consider the comments you get on your oral presentation and on the workshop.
  - Contain the term paper within 4 pages, including the bibliography. No cover/title page should be used.
  - Check spelling and grammar very carefully.
  - Obey the basic rules, such as: State title, Identify yourself, Give date (and/or version number), Use page numbering, Make a proper bibliography, Use citations in the body text, Strive to be concise.

Term Paper Deadline Summary

- Select term paper at next lecture, Friday Nov. 16.
- The term paper should be submitted (as PDF) via PingPong by Friday Jan. 11, 2019, at the latest.
  - I can give feedback on draft term papers sent to me well before the deadline. But use email for draft term papers; don't use the PingPong assignment function here.
  - No feedback once past the deadline.
  - Missed deadline directly translate into reduced grades.

Term Paper Grading Principles

3. Basic understanding of the topic area.
   - Presentation/Q&A: Show basic understanding.
     Term paper: Show basic understanding. Use > 6 sources.
4. Good understanding of the topic area.
   - Presentation/Q&A: Show good understanding.
     Term paper: Show good understanding. Use > 12 sources.
5. Very good understanding of the topic area.
   - Presentation/Q&A: Show very good understanding.
     Term paper: Show very good understanding.
     Include reflections. Use > 16 sources.
     Use your own example in the term paper.

Term Paper Grading Function

- Grading function on term paper work:
  - 70% term paper.
  - 30% presentation (including answers during Q&A).
- Plagiarism disqualifies reports (Urkund).

Topic Proposal 1

- Power estimation and minimization at microarchitecture level.

  1. Main textbook: Vol 1/Ch 13 and Vol 2/Ch 3 (3.2).
  2. “A Framework for Power-Gating Functional Units in Embedded Microprocessors”, by S. Roy et al., IEEE TVLSI 2009.
  3. “Variation-Aware System-Level Power Analysis”, by S. Chandra et al., IEEE TVLSI 2010.
  4. “A Multi-Granularity Power Modeling Methodology for Embedded Processors”, by Y.-H. Park et al., IEEE TVLSI 2011.
  5. “A Study on the Use of Performance Counters to Estimate Power in Microprocessors”, by R. Rodrigues et al., IEEE TCASII 2013.

Topic Proposal 2

- Low-power digital circuit implementation.

1. Main textbook: Vol 1/Ch 13 (13.5.3) and Vol 2/Ch 3.
2. ”Low Power Design Essentials”, by J. M. Rabaey, Springer, 2009.
3. “Low Power and Power Management for CMOS - An EDA Perspective”, by J. Kawa, IEEE TED 2008.
4. “Ultra Low Power Circuit Design using Tunnel FETs”, by R. Mukundrajan et al., ISVLSI 2012.
5. “Energy-Efficient Design Methodologies: High-Performance VLSI Adders”, by B. Zeydel et al., IEEE JSSC 2010.
6. “A 65 nm Standard Cell Library for Ultra Low-power Applications”, by M. Vohrmann et al., ECCTD 2015.

Topic Proposal 3

- Memory considerations for low power systems.

1. ”Low Power Design Essentials”, by Rabaey, Springer, 2009.
2. “Digital Computation in Subthreshold Region for Ultralow-Power Operation”, by S. K. Gupta et al., Proc. IEEE 2010.
3. “A 14 nm FinFET 128 Mb SRAM With Vmin Enhancement Techniques for Low-Power Applications”, by T. Song et al., JSSC 2015.
4. “Analysis Towards Minimization of Total SRAM Energy Over Active and Idle Operating Modes”, by N. Verma, IEEE TVLSI 2011.
5. “Low power memory implementation for a GHz+ Dual Core ARM Cortex A9 processor...”, by G. Yeung et al., VLSI-DAT 2011.
6. “A 65 nm 32 b Subthreshold Processor With 9T Multi-Vt SRAM and Adaptive Supply Voltage Control”, by S. Lutkemeier et al., JSSC’13.

Topic Proposal 4

- ESL synthesis and analysis.

  1. Chs 2+3 of “High-Level Synthesis: From Algorithm to Digital Circuit”, by P. Coussy et al., Springer, 2008.
  2. “High-Level Synthesis: Past, Present, and Future”, by G. Martin et al., IEEE D&T Comp. 2009.
  3. “Electronic System-Level Synthesis Methodologies”, by A. Gerstlauer et al., IEEE TCAS 2009.
  4. “System synthesis from UML/MARTE models: The PHARAON approach”, by H. Posadas et al., ESLsyn 2013.
  5. “Improving ESL power models using switching activity information from timed functional models”, by S. Schürmans et al., SCOPES’14.

Topic Proposal 5

- Scaling and design for manufacturability (DFM).

1. Main textbook: Vol 2/Chs 18+19.
2. “Design for manufacturability: from 1D to 4D for 90-22 nm technology nodes”, by A. Balasinski, Springer, 2014.
3. “Science and engineering beyond Moore's law”, by R. K. Cavin et al., Proc. IEEE 2012.
4. “Physically-aware analysis of systematic defects in integrated circuits”, by W. C. Tam et al., IEEE D&T Comp. 2012.
5. “EUV and e-beam manufacturability: Challenges and solutions”, by Y.-W. Chang et al., DAC 2015.

Topic Proposal 6

- Hardware/software codesign (ESL).

  1. “A Practical Introduction to Hardware/Software Codesign”, by P. R. Schaumont, Springer, 2013.
  2. “HArtes: Hardware-Software Codesign for Heterogeneous Multicore Platforms”, by K. Bertels et al., IEEE Micro 2010.
  3. “Hardware/Software Codesign: The Past, the Present, and Predicting the Future”, by J. Teich, Proc. IEEE 2012.
  4. “A comprehensive integration infrastructure for embedded system design”, by J. Barba et al., Microprocessors and Microsystems (Elsevier) 2012.

Topic Proposal 7

- EDA for 3D integrated circuits.

1. “Three-dimensional integrated circuit design: EDA, design and microarchitectures”, editor Yuan Xie et al., Springer, 2010.
2. “Design Tools for the 3D Roadmap”, by L. McIlrath et al., IEEE Int. Conf. on 3D System Integration 2009.
3. “The road to 3D EDA tool readiness”, by C. Chiang et al., Asia and South Pacific Design Automation Conf. 2009
4. “Design issues in heterogeneous 3D/2.5D integration”, by D. Milojevic et al., ASP-DAC 2013.
5. “Pathfinding: A design methodology for fast exploration and optimisation of 3D-stacked ICs”, by D. Milojevic et al., ISSOC 2009.
6. New chapter in course textbook’2016! Vol 2/Ch 9.

Topic Proposal 8

- EDA for signal and power integrity solutions.

  1. Main textbook: Vol 2/Chs 20 + 21.
  2. Ch 8 “Signal Integrity Simulations” of “The Foundations of Signal Integrity”, by P. Huray, Wiley-IEEE Press, 2010.
  3. “Signal Integrity Flow for System-in-Package and Package-on-Package Devices”, by P. Pulici et al., Proc. IEEE 2009.
  4. Practical system in “On-chip power integrity evaluation system”, by Y. Nabeshima et al., EMC Compo 2011.
  5. “Prediction of Power Supply Noise From Switching Activity in an FPGA”, L. Ren et al., IEEE T Electromagnetic Compatibility 2014.

Topic Proposal 9

- Design for test.

  1. Main textbook: Vol 1/Chs 21 + 22.
  2. ”System-on-chip test architectures: nanometer design for testability”, by Wang et al., eds., Morgan Kaufmann Publishers, 2008.
  3. “Introduction to Mixed-Signal IC Test and Measurement”, by G. Roberts et al., Oxford University Press, 2012.
  4. “Test Challenges for 3D Integrated Circuits”, by H.-H.S. Lee et al., IEEE D&T Comp. 2009.
  5. “DFT Architecture with Power-Distribution-Network Consideration for Delay-based Power Gating Test”, by V. Tenentes et al., IEEE TCAD 2015.

Topic Proposal 10

- TLM 2.0 (ESL).

1. Main textbook: Vol 1/Ch 17.
2. Reference Manual for the OSCI TLM 2.0.
3. “Quantitative Analysis of the Speed/Accuracy Trade-Off in Transaction-Level Modeling”, by G. Schirner et al., ACM TECS 2008.
4. “Fast and Accurate Transaction-Level Model of a Wormhole Network-on-Chip with Priority Preemptive Virtual Channel Arbitration”, by L. S. Indrusiak et al., DATE 2011.
5. Link to relevant tool for abstraction of RTL: http://www.hifsuite.com/

Topic Proposal 11

- EDA for FPGA.

1. Main textbook: Vol 2/Ch 13.
2. “Physical design for FPGAs”, R. Jayaraman, ISPD 2001.
3. “On power and fault-tolerance optimization in FPGA physical synthesis”, by M. Jose et al., ICCAD 2010.
4. “The Effect of Compiler Optimizations on High-Level Synthesis for FPGAs”, by Q. Huang et al., FCCM 2013.
5. “An efficient and effective analytical placer for FPGAs”, by T-H. Lin et al., DAC 2013.
6. “StML: Bridging the gap between FPGA design and HDL circuit description”, by D. Peterson et al., FPT 2013.

Topic Proposal 12

- EDA for reconfigurable computing.

  1. “Introduction to reconfigurable computing”, by C. Bobda, Springer, 2007.
  2. Ch. 3 of “Adaptable embedded systems”, by Antonio Carlos Schneider Beck et al., Springer, 2013.
  3. “Compiling for Reconfigurable Computing: A Survey”, by J. M. P. Cardoso et al., ACM CSUR 2010.
  4. “Concepts, architectures, and run-time systems for efficient and adaptive reconfigurable processors”, by L. Bauer et al., AHS 2011.

Topic Proposal 13

- Implementation of signal processing algorithms.

1. Ch. 11 of “High-Level Synthesis: From Algorithm to Digital Circuit”, by P. Coussy et al., Springer, 2008.
2. “Synthesis and optimization of DSP algorithms”, by G. Constantinides et al., Kluwer, 2004.
3. HDL-coder (datasheet), Mathworks, 2012.
4. “Design Tradeoffs and Challenges in Practical Coherent Optical Transceiver Implementations”, by A. Morero et al. IEEE JLT 2016.
5. “Technology Driven DSP Architecture Optimization within a High-Level Block ...”, D. Markovic et al., Asilomar 2006.
6. Ch. 8 of “DSP for Embedded and Real-Time Systems”, editor R. Oshana, Elsevier, 2012.

Topic Proposal 14

- Adaptive circuits and systems.

  1. “Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation”, by D. Ernst et al., MICRO 2003.
  2. “A Self-Tuning DVS Processor Using Delay-Error Detection and Correction”, by S. Das et al., JSSC 2006.
  3. “Variation-Aware Adaptive Voltage Scaling System”, by M. Elgebaly and M. Sachdev, TVLSI 2007.
  4. “Implementing Minimum-Energy-Point Systems With Adaptive Logic“, by L. Koskinen et al., TVLSI 2016.
  5. “Bitcell-Based Design of On-Chip Process Variability Monitors for Sub-28 nm Memories”, by S. Gupta et al., TCAS 2016.

Topic Proposal 15

- Neuromorphic systems.

  1. “A 45nm CMOS Neuromorphic Chip with a Scalable Architecture for Learning in Networks of Spiking Neurons”, by J. S. Seo et al., CICC 2011.
  2. “Design Tools for Artificial Nervous Systems”, by L. K. Scheffer, DAC 2012.
  3. “Specifications of Nanoscale Devices and Circuits for Neuromorphic Computational Systems”, by B. Rajendran et al., TED 2013.
  4. “Memory and Information Processing in Neuromorphic Systems”, by G. Indiveri et al., Proc. IEEE 2015.

Topic Proposal 16

- Approximate computing.

  1. “A Low-Power DCT Core Using Adaptive Bitwidth and Arithmetic Activity Exploiting Signal Correlations and Quantization”, by T. Xanthopoulos et al., JSSC 2000.
  2. “Scalable Effort Hardware Design”, by V. Chippa et al., TVLSI 2014.
  3. “Designing Approximate Circuits using Clock Overgating”, by Y. Kim et al., DAC 2016.
  4. “Computing Approximately, and Efficiently”, by S. Venkataraman et al., DATE 2015.
  5. “Near/Sub-Threshold Circuits and Approximate Computing”, by J. Schlachter et al., ISVLSI 2015.

Topic Proposal 17

- SoC voltage regulation.

  1. Part IV “Power Delivery Circuits” of “On-Chip Power Delivery and Management”, editor I. P. Vaisband et al., Springer 2016.

  2. “System-Level Power Analysis of a Multicore Multipower Domain Processor With ON-Chip Voltage Regulators”, by A. Paul et al., TVLSI 2016.

  3. “Smart Grid on Chip: Work Load-Balanced On-Chip Power Delivery”, by D. Pathak et al., TVLSI 2017

Practical advice for lab exercises

Large-Scale Project Organization

- Design engineers and verification engineers belong to different teams and have different missions.
- The design and verification teams can gain from different work methods: Using different implementation approaches may increase the chance of spotting bugs.
- Limit to how much design and verification teams should be separated: Outsourcing verification work is not a good idea.

Document Implementation/Verification

- Document all blocks:
  From smallest block, to complete system.
- Document testbenches and simulations
  (stimuli and expected output).
- Make it possible to trace updates;
  list who verified the block and when.
- List block dependencies, since changing a component
  forces a re-verification of everything
  above it in the design hierarchy.
- Document odd requirements and behavior.

Bottom-Up Verification

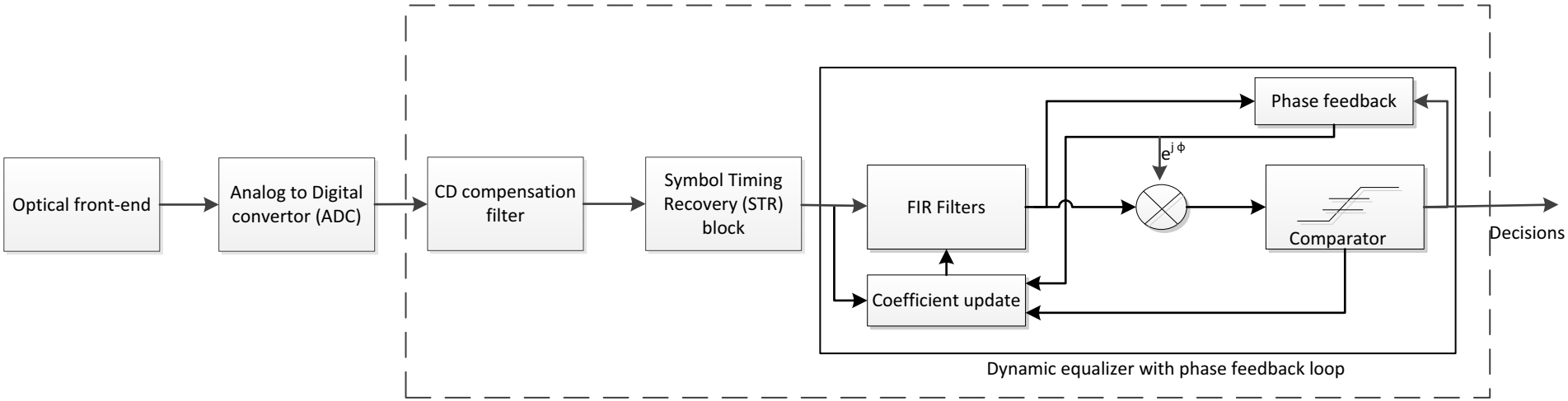
- When designing the ALU it is good practice to conceptualize it top down, but in terms of verification, the ALU and its sub-blocks should be verified bottom up.
- The order in which sub-blocks should be implemented and verified should be clear from the paper design.
- When functionality of individual sub-blocks has been verified, verify functionality of the combination of sub-blocks.
- Finally, you will reach the top design.

Build Testbenches

- A testbench for HDL code consists of non-synthesizable code that can interface to input test vectors and expected output vectors.
- Several testbenches, for the same HDL design, can exist:
  - The chance to spot errors (even in the testbench!) increases with alternate testbenches.
    Conformity does not always work in your favor!
  - Verification of functionality may warrant a different testbench than one which is used to establish maximal speed (RCA vs Sklansky adder).
  - An ALU, for example, can be verified stand-alone or in the context of preceding and/or succeeding hardware.

SCENARIO

- Take for example the DSP system from Lecture 1.



- Assume a testbench is to be constructed to verify the FIR filter.

TESTBENCH 1(2)

```
entity testbench is
end testbench;

architecture behavior of testbench is

-- Declare component of Design Under Test (DUT)
   component FIR is
      port()
   end component;

-- Declare internal signals
   signal clk: STD_LOGIC:='0';

-- Define variables
   constant clk_period : time := 1 ns;
```

TESTBENCH 2(2)

```
   begin
      -- Instantiate DUT
      u0: FIR port map ();

      -- Process definitions
      ‘generate clock and reset’
      ‘read in test vectors‘

      -- Testbench main process
      ‘loop: read file test vector ->
      send test vector to DUT ->
      read DUT output and compare to reference’
      ‘use assert for comparison’
      ‘write errors to file’
   end;
end behavior;
```

Testbench Structure

- Initialization.
- Input stimuli.
- Response assessment.
- Verification utility - repository of common functions.
- Clock generation and synchronization.
  - Make the clock period statement a variable so it is easy to change this parameter:

```
constant clk_period : time := 1 ns;
```

- Interface between testbench and design implementation.

Script Example - Run Inside RTL Compiler

```
set REPORT_PATH reports/
##set your own parameters

source anotherscript.s
##run other scripts

report power > ${REPORT_PATH}/power.rpt
##report things

shell rm some_file
##run Linux commands
```

Makefile Example

```
all:
   lib rtl_syn elab sim

elab:
   ncelab WORKLIB.ALU_TB:BEHAVIORAL

sim:
   ncsim WORKLIB.ALU_TB:BEHAVIORAL

lib:
   ncvhdl /chalmers...../CORE9GPHS_VHDL_FUNCT.vhd

rtl_syn:
   ncvlog double_check_this_netlist.v
   ncvhdl ALU_TB_2367ps.vhdl

clean:
   rm –r INCA_libs
```

Makefile - Run in Linux

- Execute (two examples):

```
[perla@remote12 sims]$ make all
[perla@remote12 sims]$ make clean
```

- Other Linux commands:
  - `ls` (show directory content, `ls -l` shows details)
  - `cd` (change directory)
  - `mkdir` (make new directory)
  - `rm` (remove file)
  - `mv` (move files and directories)
  - `vi` or `gedit` or `emacs` (some texteditors)