EE 574/ EE 674: SYSTEM LEVEL DESIGN AUTOMATION, HIGH LEVEL SYNTHESIS, AND SOFTWARE-HARDWARE CODESIGN

Marek A. Perkowski

Department of Electrical Enginering.

- The course teaches new ideas and techniques in high-level synthesis, system-level synthesis, formal design and verification and software-hardware codesign. Part of it has been organized according to industrial requests and emphasizes digital design and simulation using modern tools and hardware description languages. Working knowledge of VHDL is welcome, but is not mandatory as a prerequisite.
- System-related subjects will be discussed that are of interest to students. New books by Marc Davio and Giovanni De Micheli will be used, together with instructor's lecture notes and journal/conference papers. Lecture notes will be also available as WWW Pages.
- The class has been now made independent and it does not require other classes as prerequisities.
- The class is heavily project-oriented and topics change from year to year. There will be several software/hardware design mini-projects that will vary from year to year.

TOPICS BY HOUR:

1. Comprehensive design automation systems (3 hours).
2. Regular expressions, Petri Nets and other hardware description methods (3 hours).
3. Problems of system and high-level synthesis and verification (4 hours).
4. Register-transfer notation and design (3 hours).
5. Integrated approaches to formal design and hardware verification (5 hours).
6. Methods and languages to specify hardware, system hardware-software codesign (3 hours).
7. Data path design (4 hours).
8. Scheduling and allocation (3 hours).
9. Design methods for systolic and pipelined computers (2 hours).
10. FPGA technologies, FPGA-based prototyping and reconfigurable computing (5 hours).
11. Cellular and dynamic architectures (3 hours).
12. Microprograming and microprogram optimization (2 hours).

PREREQUISITE: graduate standing in EE.

TEXTBOOKS:

- Davio, “Digital System Design” – available from the professor.
- Giovanni De Micheli - "Synthesis and Optimization of Digital Circuits" McGraw-Hill, 1994.
- Marek Perkowski – Lecture notes.

GOALS:

- This is the third course in a sequence, but can be taken as a stand-alone class, because it is graded mostly on projects. The class will include lectures by the professor, student presentations and discussion of projects and tools.
- Continuation of projects from previous classes of this sequence is encouraged. New students will take new projects.
- Students learn about advanced topics related to the design automation of general-purpose and special processors. The special emphasis is for FPGA-based circuits for robotics, telecommunication and image processing.
- The emphasis is on using various tools, tool integration, software-hardware co-design, fast prototyping methods, hardware verification, formal design and verification, and use of simulation methodologies, use of various programmable devices and design of innovative computer architectures.
- Substantial and comprehensive group project that leads to the design of the state of the art software tool, processor or software-hardware system.

LABORATORY PROJECTS

- There is no formal laboratory hours, but the students work on a comprehensive project in Intelligent Robotics Lab.
- These projects vary from year to year.
- In the past the software projects included various point tools in the following areas: logic synthesis, state machine design, silicon compilers, simulators, technology fitters, technology mapping, microprogramming, and high-level synthesis.
- There were also projects related to integration of the tools to systems and testing them in designs.
- In the past, the hardware projects in Robotics included the following:
    1. Concurrent reactive state machine for control of a walking robot,
    2. Co-processor interface to PC,
    3. Cube Calculus Machine for solving combinational problems,
    4. Satisfiability Machine,
    5. ESOP minimizer based on breadth-first hardware search,
    6. Petrick Function minimizer based on counting search.
    7. Image matcher based on maximum cliques,
    8. Multiplier of multidimensional polynomials,
    9. Array multiplier,
    10. Systolic multiplier,
    11. Systolic Matrix operations using Faddeev algorithm,
    12. Systolic LU decomposer,
    13. Rough Set Machine,
    14. Ashenhurst Decomposer,
    15. Various microprogrammed units of special applications.

The hardware projects in Telecommunication included the following:

1. Digital FIR and IIR filters,
2. Digital stack filter,
3. Median filter,
4. Digital convolvers,
5. Fast Fourier processor,
6. Fast Fast Walsh processor,

7. Fast Reed-Muller processor,
8. General-purpose Digital Signal Processor,
9. Image coding and compressing processors,
10. Viterbi ,
11. Manchester
12. Spread-Spectrum Transmitter,
13. Internet switch,
14. Various coding circuits.

The hardware projects in Image Processing included the following:

1. Hough Transform Processor for finding parameters of straight lines in image,
2. Image filtering and other transformations based on convolution,
3. Image histogramming.

- This year most projects are tool-related and related to software-hardware codesign, and include:
  - Use Mentor Tools for automatic insertion of BIST to VHDL
    (this project requires some knowledge of Unix and VHDL).
    This project is preferably for students who have already taken my Test class.
  - Use Summit Tools (and/or Xilinx Tools) for synthesis of verifiable controllers.
    Verification will be by hand.
    This project is preferably for students who take my 510FP class, but
    other students are also welcome. No other prerequisites are expected.
  - Use ORCAD Tools for designing of a logic machine in ALTERA or Xilinx.
    (this project requires some knowledge of VHDL, but can be done by mostly schematic capture).
    Choice of a logic machine is to be discussed, it can be a sorter/absorber,
    satisfiability machine, covering machine or other of your choice.
    This project does not require other prerequisities than some elementary
    VHDL knowledge on the level of EE 171.
  - Re-compile Diades tools of PSU for new Lisp and make them available from WWW.
    Re-Design totally automatically all 20 ADL examples from high-level specification.
    Investigate the role of encoding, state-minimization and logic synthesis.
    (this project requires some knowledge of Unix, html and Lisp).
    It is for students with interest in hardware-software co-design and software implementations
    of CAD tools.

COMPUTER USAGE:

SUN SPARC Stations are used to run Mentor, Cypress, Orcad, DEC, Analogy Inc., and various public-domain and university design tools and simulators.

ESTIMATED CONTENT:

- Engineering Science: 2 credits or 50%.
- Engineering Design: 2 credits or 50%.

ADDITIONAL INFORMATION:

- Besides the textbook, there are very extensive class notes available from Smart Copy, that include descriptions of all projects and many references.

- The students work in groups on the comprehensive projects. There are no tests or homework. There are graded class presentations by every student. Every student has to write several reports and deliver piece of computer code or a hardware design.
- The grade is based on student's performance in presentations, reports and final project.

The grading policy is as follows:

- 20 %. Written individual reports that document reading and understanding of literature, class and off-the-class group discussions, new ideas and theoretical contributions, working computer code, and results of testing the programs.
- 30 %. Project, either software, hardware or software/hardware.
- 10 %. Class lectures presented by students. They are based on previous reports of the students. Grading takes into account both the quality of material, the use of figures, transparencies, and other materials prepared by students, and also the quality of the oral delivery, and how well the student is able to handle questions of the class and the professor.
- 10 %. Participation in class and off-the-class discussions. (Evey week there is an additional meeting just to present the progress of work on reports).
- 30%. Final report, that takes into account the critics of the professor, additional results from the computer, and feedback from the class.

MORE DETAILED SPECIFICATION OF TOPICS THAT MAY LEAD TO FUTURE CLASS PROJECTS:

1. **Comprehensive design automation systems.** Systems that start from high-level specifications, role of VHDL and Verilog. Systems for state machines and logic level. Systems for FPGAs and PLDs. Silicon compilers. CAD on system design level. Role of Artificial Intelligence and new software methodologies. Internet.
2. **Methods and languages to specify hardware.** Examples of languages: VHDL, Verilog, ADL, RUBY, Esterel and ZEUS. Behavioral, functional, structural and register-transfer descriptions. Algorithmic State Machines. Fundamentals of Hardware and System Description Languages. Current extensions of language VHDL and expandable languages. Specification of controllers and data path architectures. Hierarchy of languages, regular languages, regular expressions. Formal and intuitive synthesis of Finite State Machines from regular expressions.
3. **Petri Nets as input specifications.** Parallel machines and tokenized machines. State Charts and Spec-Charts. Other hardware description methods on high and medium level. Problems of system and high-level synthesis and verification.
4. **Predicate calculus methods to specify hardware.** Automatic theorem proving. Modal operators. Use of higher order logic. Examples of symbolic simulation, analysis and verification of combinational logic.
5. **Register-transfer notation and design.** Use of invariants in verification and optimization of hardware. Systematic design of Glushkov machines with control realized as a finite state machine.

6. **Hardware Verification.** Verification of Boolean Functions, Multiple-Valued Functions, Fuzzy Functions and Finite State Machines. Use of Decision Diagrams. Verification of microprograms. Automatic System Verification.
7. **Data path design.** Design of arithmetic and logic operators. Use of similarities of structures and operators. Generalized factorization. Compilation of Data Path of a processor in VHDL. Algorithms for optimal and quasi-optimal scheduling of processor operations in time. Hardware scheduling and allocation. Logical, statistical and knowledge-based approaches. Simulated annealing and genetic algorithms. Exact and approximate algorithms. Optimal and quasi-optimal allocation of operations to hardware modules. Applications to optimize an FFT processor. Various projects.
8. **Design methods for systolic and pipelined computers.** Specification of a systolic processor for sorting and median filtering in VHDL. Optimization of pipelines.
9. **FPGA technologies and Computing.** FPGA-based prototyping and emulation. Array processors. Xputers. Data Flow techniques. Very Long Instruction Processors in FPGA. Neural Nets and Fuzzy logic processors. Emulators and Xputers. Project: VHDL for systems from Field Programmable Gate Arrays.
10. **Cellular and dynamic architectures**. Cellular logic. Cellular automata. Game of life. Modern realizations of cellular logic and automata.
11. **Microprograming and microprogram optimization**. VHDL description of a microprogrammed controller. Using stack and recursion in hardware. Examples of architectures with stacks. Techniques of microprogram optimization.
12. **Simulation.** Levels of system simulation. Project: write a special-purpose simulator for multiple-valued, quantum or single electron transistor systems.
13. **Software-hardware co-design.** Hierarchical design of a symbolic computer.