David Kebo Houngninou

Instructional Associate Professor
Department of Computer Science and Engineering
Texas A&M University, College Station, TX

Phone: 214-686-9611
Email: davidkebo@tamu.edu
Website: https://people.engr.tamu.edu/davidkebo

Education

Ph.D. in Computer Engineering December 2017
Area of research: Hardware Formal Verification
Southern Methodist University, Dallas, TX

Master of Science in Computer Engineering December 2010
Washington University in St. Louis, St. Louis, MO

Bachelor of Science in Computer Engineering December 2008
University of Evansville, Evansville, IN

Professional Experience

Instructional Associate Professor August 2018 - Present
Department of Computer Science and Engineering, Texas A&M University, College Station, TX

Teaches core undergraduate and graduate-level courses in data structures, algorithms, programming languages, operating systems, computer security, and hardware verification. Develops curriculum and assignments to build students' theoretical knowledge and practical abilities. Incorporates the latest advancements in computer science education to achieve learning outcomes.

Adjunct Lecturer August 2015 - May 2018
Department of Computer Science and Engineering, Southern Methodist University, Dallas, TX

Developed and taught undergraduate computer science courses, including microprocessor architecture and interfacing, digital systems design, software engineering, and discrete computational structures. The courses focus on theoretical foundations and practical applications, such as designing hardware interfaces, using HDLs, and applying mathematical logic/proofs. Prepared curriculum teaching ARM architecture, FPGA architecture, and combinatorics.

Design Engineer (Multicore and DSP design and verification) May 2017 - September 2017
Texas Instruments, Dallas, TX

Verified multicore SoC and DSP architectures, including PCIe and Cortex-R5 cores, by generating stimuli to activate cache coherence and performing randomized stress testing pre-silicon to uncover complex interaction bugs.

Researcher (Temporal and structural graph analysis) June 2014 - September 2014
IBM Research Lab, Cambridge, MA

Developed interactive data visualizations using D3.js to graphically display governance, risk, and compliance data, enabling dynamic sorting/analysis of risks, ratings, descriptions, and comments with exportable tables to improve accessibility and risk management workflow.

Software Developer May 2012 - August 2012
NXP Semiconductors, Austin, TX

Developed and maintained Grails-based software applications implementing MVC frameworks with Groovy/Java, integrating RESTful APIs for the NXP Semiconductor fabrication plant.

Software Developer January 2011 - October 2011
Wirevibe, Irving, TX

Maintained back-end systems of web applications using PHP, MySQL frameworks, RESTful APIs, and MVC architectures.

Hardware Engineer August 2008 - May 2010
Magellan Integration, Evansville, IN

Led the IT infrastructure and lab design projects, including networking, security, and surveillance systems. Spearheaded the design and documentation for various CCTV low-voltage installations, including custom security and surveillance solutions to meet clients' specifications across multiple corporate locations.

Teaching Experience

Teaching Interests: micro-architecture, functional verification, formal verification, emulation.

Department of Computer Science and Engineering, Texas A&M University

CSCE 689: Hardware-based verification using emulation and prototyping

This graduate course covers fundamental principles of emulation and prototyping for comprehensive IP/SoC design verification and system validation. The course's main objective is to expose students to industry best practices for emulating and prototyping SoC designs.

CSCE 616: Hardware-based verification using emulation and prototyping

This graduate course covers hardware functional verification, case studies on verification in integrated circuit design, introduction to industry best practices and functional verification. Key topics include verification cycle & methodology, stimuli generation, UVM, assertion-based verification, coverage models, and regression.

CSCE 465: Computer and network security

Fundamental concepts and principles of computer security, operating system and program security, malware, network security, secret key and public-key cryptographic algorithms, hash functions, authentication, firewalls, and intrusion detection systems, web and application security.

CSCE 410: Operating systems

The course teaches aspects of operating systems, such as system calls, processes, memory management, scheduling, threading, synchronization, file systems, and I/O devices. The labs implement extensions to open-source Linux kernels such as the RISC-V xV6 and operating system libraries in the user space.

CSCE 313: Introduction to computer systems

Introduction to system support for application programs, both on a single node and over a network, including OS application interface, inter-process communication, introduction to systems and network programming, and simple computer security concepts.

CSCE 312: Computer organization

This course integrates notions from algorithms, computer architecture, operating systems, compilers, and software engineering in one framework. The course explores techniques used to design of modern hardware and software systems. The material covers the following topics: Introduction to computer systems, data representation, machine language, processor architecture, memory hierarchy, assembler, virtual machines, compiler, and operating systems.

CSCE 222: Discrete structures for computing

This course provides the mathematical foundations from discrete mathematics for analyzing computer algorithms, for both correctness and performance, models of computation, including finite state machines, and Turing machines. The course covers logic, proofs, sets, algebraic structures, graph theory, and combinatorics.

CSCE 221: Data structures and algorithms

Specification and implementation of abstract data types and their associated algorithms, including stacks, queues, lists, sorting and selection, searching, graphs, and hashing; performance tradeoffs of different implementations and asymptotic analysis of running time and memory usage; includes the execution of student programs written in C++.

CSCE 121: Introduction to programming design and concepts

Computation to enhance problem-solving abilities, computational thinking, understanding how people communicate with computers, and how computing affects society. Design and implementation of algorithms; data types, program control, iteration, functions, classes, and exceptions; understanding abstraction, modularity, code reuse, debugging, maintenance, and other aspects of software development; development and execution of programs.

CSCE 110: Programming I

This introductory computer science course teaches students to use computational thinking and programming to solve problems. Through learning the fundamentals of Python programming, students gain strategies and tools to analyze problems, break them down into logical components, and develop algorithmic solutions. The course focuses both on building core programming knowledge as well as applying those skills to enhance critical thinking and problem decomposition abilities. The topics covered include core language syntax, data types, variables, conditional logic, loops, functions, classes, modules, algorithms, recursion, and basic data structures.

Department of Computer Science and Engineering, Southern Methodist University

CSE/EE 7385: Microprocessor architecture and interfacing

Graduate course on the ARM Microprocessor Architecture and Interfacing. Covers memory structure and interfacing, bus systems, support chips, tools for hardware design, analysis, simulation, implementation, and debugging. Includes a laboratory to design and analyze interfaces to processors, memories, and peripherals.

CSE/EE 7387: Digital systems design

This graduate course covers combinational logic synthesis using Verilog, FPGA architecture, and finite state machine design. Teaches the use of HDLs for circuit specification and automated synthesis tools. Includes laboratory experiments and a final design project.

CSE 4345: Software engineering

Software system development and overview of development models and their stages. Covers system feasibility and requirements, architecture and design, validation and verification, maintenance, and evolution. Includes project management and a review of current software engineering literature.

CSE 2353: Discrete computational structures

Application of the concepts of discrete mathematics to computer science problems. Focuses on mathematical principles central to computer science, including sets, logic, and proofs. Covers additional topics, such as an introduction to graph theory and combinatorics.

Course Development

Department of Computer Science and Engineering, Texas A&M University

CSCE 110 Programming I - online course June 2020

An online introductory course designed for students with little or no programming experience. This online course provides students with an understanding of the role computation plays in solving problems and helps students, regardless of their major, feel confident writing programs. The course uses the Python programming language. The material developed includes recorded lectures, lecture notes, auto-graded labs, and practice activities.

CSCE 689 Hardware-based verification using emulation and prototyping August 2023

Graduate course developed in collaboration with Synopsys to teach students state-of-the-art FPGA-based Emulation Systems for verification. This course equips students with skills and knowledge essential for the semiconductor industry, focusing on hardware-based verification using emulation and prototyping. The content and methodologies align with both academic theories and current industry standards.

Service and Professional Activities

Doctoral Consortium Deputy Chair January 2023 - Present
CMD-IT/ACM Richard Tapia Celebration of Diversity in Computing Conference

Led the consortium, allowing Ph.D. candidates to present their research proposals and experimental results to a panel of researchers and industry professionals. The candidates receive feedback on their work and guidance for future research.

Diversity, Equity, and Inclusion Committee member August 2022 – Present
Department of Computer Science and Engineering, Texas A&M University

Advise the department leadership on improving diversity, such as recruiting and retaining underrepresented groups, ensuring equity in resource allocation and opportunities, and building an inclusive departmental culture. Track representation data and trends, identify specific areas needing attention, and develop evidence-based recommendations.

Undergraduate Curriculum and ABET Committee member August 2021 - Present
Department of Computer Science and Engineering, Texas A&M University

Review proposed updates to the computer science undergraduate curriculum and coordinate with the Accreditation Board for Engineering and Technology (ABET).

Undergraduate Admissions Committee member August 2021 - Present
Department of Computer Science and Engineering, Texas A&M University

Review candidates for admission into the computer science and Engineering program.

Computer Engineering Coordinating Committee member August 2018 – May 2019
Department of Computer Science and Engineering, Texas A&M University

Present reviews of proposed changes to the computer engineering undergraduate curriculum and coordinate with the Accreditation Board for Engineering and Technology (ABET).

Founder, President January 2022 – Present
Teamup Nonprofit, College Station, TX

Lead the nonprofit organization in promoting project-based learning and empowering K-12 and college students to build technology to drive positive social change. The mission of Teamup is to prepare students for STEM careers while empowering them to create technology for social good.

Conference workshops and talks

Early introduction to computer architecture in K-12 March 2023
SIGCSE Technical Symposium on Computer Science Education

Introduced an educational framework for K-12 and undergraduate college students to learn computer architecture by building custom processors, exploring computer subsystems, and observing how programs are simulated in real-time.

FLIP: A RISC-V visual computer architecture simulator for K-12 March 2023
SIGCSE Technical Symposium on Computer Science Education

Presented an interactive simulator that allows students to build and run programs on a custom RISC-V processor. The increasing interest in the RISC-V ISA and its fast adoption in chip design makes this instruction set a great candidate to support this educational tool for beginners.

A RISC-V Open-Source Architecture Design Space Exploration Toolbox September 2021
CMD-IT/ACM Richard Tapia Celebration of Diversity in Computing Conference

Introduced the RISC-V architecture and its latest applications in SoC designs. Trireme is a RISC-V architecture design exploration suite for education and research. The session presented a RISC-V simulator, its functionalities and ran hands-on design exploration examples with the attendees. The simulator introduced in the workshop allows students and researchers to experiment with the RISC-V ISA features and quickly bring up complete and fully working architectures.

Publications

David Kebo Houngninou. 2023. FLIP: A RISC-V Visual Computer Architecture Simulator for K-12. In Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2023). Association for Computing Machinery, New York, NY, USA, 1271. https://doi.org/10.1145/3545947.3573252

David Kebo Houngninou, Maristela Holanda, and Dilma Da Silva. 2023. Early Introduction to Computer Architecture in K-12. In Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2023). Association for Computing Machinery, New York, NY, USA, 1331. https://doi.org/10.1145/3545947.3576277

Thornton, M. A., Houngninou D. K. and Miller D. M. "Computing the Reed-Muller Spectrum / Algebraic Normal Form: Functional Methods" in Advances in the Boolean Domain (B. Steinbach, editor), Cambridge Scholars Publishing, 2022.

Houngninou, D. K., Thornton M. A. and Miller D. M. "Extracting the Reed-Muller Spectrum / Algebraic Normal Form from a Circuit Specification" in Advances in the Boolean Domain (B. Steinbach, editor), Cambridge Scholars Publishing, 2022.

Houngninou D. K., Miller D. M., Thornton M. A., "ANF Computation of Cryptographic Switching Functions using a Netlist Representation," Bell System Technical Journal 28 (1), 59-98, 2021.

Houngninou D. K. and Thornton M. A., "Simulation of switching circuits using transfer functions," 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), Boston, MA, 2017, pp. 511-514.

Houngninou D. K. and Thornton M. A, "Implementation of switching circuit models as transfer functions," 2016 IEEE International Symposium on Circuits and Systems (ISCAS), Montreal, QC, 2016, pp. 2162-2165.

Projects

RISC-V visual computer architecture simulator

FLIP is a hardware simulator designed for educational purposes. FLIP is web-based and includes several tools that can be used independently for teaching computer architecture.
Chip Builder and Navigator: The chip builder is designed to teach students the purpose of the low-level components of a computer system. The user can choose a predefined core or build a custom system from scratch with (ALU, caches, main memory, and configuration parameters). The experience is guided step-by-step to assist the learner. The navigator is an interactive browser that allows the user to explore all the components of the computer system. The learner can view the current state of the RAM, ALU, registers, cache, and buses while executing the program instructions.
Code Editor and Simulator: The editor and simulator are designed to show students how instructions are loaded and executed incrementally on a processor. The built-in code editor allows the user to write programs to run on the target device or load sample programs. To facilitate the understanding of the execution, the student can visualize the execution of assembly language code step by step with additional labels and explanations. The speed and the level of detail for the animations can be controlled in the user settings.

Professional Society Memberships

American Society for Engineering Education (ASEE)

ACM Special Interest Group on Computer Science Education (SIGCSE)