I want to study programming languages, compilers, and program verification with a focus on their practical applications to hardware design and distributed systems. Although coming from a Computer Science background, my undergraduate education included coursework in Electrical and Computer Engineering and Physics, equipping me with a multidisciplinary foundation to approach complex problems from multiple perspectives. My projects—ranging from designing a language framework for typesetting systems and proposing an augmented type system for JavaScript, to building a mini kernel and reconstructing an NES console on FPGA—have deepened my technical expertise and broadened my understanding of the interplay between software and hardware. By pursuing graduate studies at UIUC, I aim to advance my knowledge and contribute to innovative research in programming languages and their role in enabling robust hardware and system designs.

During my undergraduate years, a comprehensive curriculum provided me with a broad academic foundation and advanced coursework in programming languages, compilers, systems, and electrical engineering. In Programming Language Design (CS 422), I created languages in different paradigms using the K framework. In Compiler Construction (CS 426), I developed a functional KOOL compiler. In Operating System Design (CS 423), I built several practical Linux kernel modules, and in ECE 391 (Computer Systems Engineering), I gained a deeper understanding of kernel development by implementing a mini operating system. My coursework in Digital Systems (ECE 385) further enhanced my understanding of FPGA programming and chip design, providing me with hands-on experience in hardware implementation and logic synthesis. Together, these courses have equipped me with the theoretical foundation and practical skills necessary to explore the intersection of programming languages, hardware systems, and computation.

Building on the foundational knowledge from my coursework in programming languages and compilers, I began an exploratory project to address challenges I encountered with LaTeX during technical writing. The idea of creating a typesetting system to replace LaTeX emerged from my frustration with its poor readability and limited flexibility when writing complex ODE formulas, big-step semantics, and TikZ graphs. Lines of dense macros made source files unreadable, while simple adjustments, like spacing, often required opaque “black magic.”

To address these issues, I began designing a new language that improves LaTeX's readability and expandability. I studied The TeXbook and LuaTeX documentation. It introduces some effective typesetting concepts like TeX's Glue-Box model but relies on an outdated macro system and lacks modern Unicode support. Based on these insights, I designed a Lisp-like domain-specific language, which allows user-defined syntax and semantics to capture frequent constructs, such as those in ODE formulas or big-step semantics. This could fold cumbersome expressions into human-readable syntax, making the language significantly more readable and usable. I also prototyped an IR that generates instructions for rendering in browsers or PDFs according to OpenType font specification. While the project is still preliminary, it has strengthened my understanding of modular language design and broadened my perspective on translating high-level constructs into graphical outputs.

Beyond the typesetting project, I also explored language design through a proposal to address limitations I observed in TypeScript, a typed superset of JavaScript that I frequently use in web programming. While its subtyping system shows promise for refined typing that could better capture the behavior of JavaScript APIs, the lack of type operations often leads to convoluted "type gymnastics" or manual workarounds, undermining its expressiveness.

To address these limitations, I proposed a predicate-based type system, where compile-time predicate functions could be used to evaluate type constraints based on metadata of checked variables. Specific type operations could be implemented in predicate functions, while commonly used types are treated as syntactic sugar, enhancing the language's flexibility and usability. It also unifies imperative paradigms with type checking, where TypeScript's system is often inconsistent. The proposal deepened my understanding of predicate-based systems and its application in practical type system design.

In addition to my projects, I have worked on several smaller projects that further developed my technical skills. I built an AI-based interactive storyteller that generates story pieces based on user prompts. In a course project, I implemented a mini kernel for x86 architecture, which deepened my understanding of operating systems and low-level programming. Additionally, I reconstructed the Nintendo NES console on FPGA as part of a digital systems course, where I gained hands-on experience in hardware design. The NES console uses a 6502 CPU and a custom Pixel Processing Unit (PPU). Through this project, I learned how to decompose CISC instructions into simplified internal operations, design microcode, and generate microinstruction tables automatically. This experience significantly enhanced my understanding of hardware-software integration and provided practical insights into CPU architecture and hardware implementation. These projects have broadened my perspective in hardware, systems, and AI, complementing my academic and research preparation.

My coursework and projects have deepened my interest in the application of programming languages and compilers in different systems, as well as my broader interest in software-hardware systems. I am particularly interested in designing programming languages that simplify and streamline hardware design, enabling efficient development of hardware systems and improving their verification processes. Additionally, I want to explore verification techniques in different contexts such as operating systems, cryptography, and embedded systems. UIUC's strong curriculum and collaborative research environment make it the ideal place to pursue these goals.

As an current UIUC undergraduate student applying to graduate school, my experience with UIUC's strong curriculum and its research groups shows that it will be the ideal place for me to pursue my goals in application of programming languages and compilers in hardware and systems. I am particularly excited about the opportunity to work with Professor Nam Sung Kim, whose research on power-efficient computing and Processing-In-Memory (PIM) aligns with my interest in designing tools that optimize the interaction between software and hardware. Professor Ning Luo’s work on formal methods and programming languages for achieving security and verifiability resonates with my goal of exploring verification techniques for operating systems and embedded systems. Professor Deming Chen’s contributions to reconfigurable computing and system-level design methodologies inspire me to investigate how programming languages and compilers can streamline hardware design and improve system efficiency. The combination of

UIUC's exceptional faculty and research-driven environment will provide the perfect foundation to advance my research interests and contribute meaningfully to the field of electrical and computer engineering.