Sandeep Dasgupta

Email: sdasgup3@illinois.edu
Email: sandigame.123@gmail.com
Webpage: http://webhost.engr.illinois.edu/~sdasgup3
Github: https://github.com/sdasgup3
Phone: (+1) 2174172344

Interested In

- Compiler (LLVM, GCC) (Frontend/Backend/Optimizations)
- Static/Dynamic Program Analysis
- Design of Interpreter & Dynamic typed languages
- Data Analysis using Statistical tools
- Binary Code Decompilation
- Language Formal Semantics
- Parallel & Power Aware Computation
- Symbolic Execution

Languages proficient with

- Go, C++, Swift, Haskel, C, Assembly
- Perl, Python, R
- Pthread, OpenMP, MPI, Charm++

Career Overview

PhD Computer Science – GPA 3.95/4.00 *Aug'13 - May'20 (Expected)*
CS @ University of Illinois Urbana Champaign

- Working with Prof. Vikram S. Adve on ALLVM Research Project
- Current Research
  - Formal semantics of X86-64 ISA. ***Public Repository***
  - Validating the decompilation from binary to LLVM IR: Given the importance of binary analysis, establishing the faithfulness of a binary lifter (or decompiler) is most desirable, and is the current focus of my work. Specifically, we are developing techniques and tools for the translation validation of binary decompilers to LLVM IR, including defining the semantics of LLVM IR and X86-64 ISA, which is required for that validation.

Software Engineering Intern, with Stephane Eranian, Kernel performance team *Sept'18 - Dec'18*
Google Inc.

- First, Developed from scratch an "Intel Topdown" converter that we will be released to the open-source community. This will make the Intel Topdown bottleneck analysis methodology more visible and successful.
- Second, Developed proof-of-concept prototypes to achieve data profile symbolization. Given a set of hot load/store instructions (generated by sampling the application using tools like perf), the problem is to find the type of the data address associated with those instructions. Such information is useful in optimizing memory allocation strategies or cache utilization. Deployed LLVM based tools implementing approaches based on: (1) Allocation Based Profiling and (2) Dwarf debug Info.
- Received manager's recognition via spot bonus.

Software Engineering Intern, Tipp Moseley, Continuous profiling team *May'17 - Aug'17*
Google Inc.

- Augmented the profiling tools with analysis capable of guiding the service owners to pinpoint unexpected performance changes (called "havocs" in this work) in CPU time.

  Developed a statistical Go tool which determines potential "havocs" with reduced false positive rate. Came up with an idea of correlating the statistically detected trend changes in the data with build CL changes which not only gives confidence on trend change detection but also helps in reducing the false positives.

Compiler Engineer Intern, with Roman Levenstein, Swift performance team *April'16 - Aug'16*
Apple Inc.

- Achieved functional completeness in converting swift to JS. Involves converting swift programs to LLVM bitcode using the swift compiler infrastructure and then use emscripten to convert the result to JS. The challenges include (but not limited to):
  - Converting Swift standard and runtime library into LLVM IR
  - Enhancing emscripten type legalization for small integer types like i3
  - Enhancing emscripten address relocations support
  - Supporting dynamic loading of JavaScript generated from emscripten
  - Adjustments in runtime library to support Type and protocol conformance metadata

Component Design Engineer *Aug'11 - Jun'13*
Intel Technology India Pvt. Ltd.

- Worked as Design Automation Engineer for Formal Equivalence Verification (FEV) of hardware designs for Broadwell (BDX) and Skylake (SKL) projects. Involves building flows and methodologies and interacting closely with global FEV teams to provide FEV solutions.
- Owner of tools and infrastructures for driving FEV central runs for BDX. Invloves working closely with teams of various design styles.

M.Tech Computer Science & Engineering – CPI 9.00/10.00 *Aug'09 - Jun'11*
Indian Institute Of Technology Kanpur, Kanpur, India.

- Secured **rank** 1 in M. Tech 2009 Batch, IIT Kanpur.

Senior Member Of Technical Staff *Aug'06 - Jul'09*
Interra Systems India Pvt. Ltd.

- Developer of Interra's premiere front-end analyzer products - Cheetah (SystemVerilog) and MVV(Mixed Verilog Vhdl) and provided support for several new constructs of System Verilog IEEE-1800-2005.
- Worked with Atrenta (I) Pvt. Ltd. (service project) for the development of System Verilog features in Spyglass DFT.
- Awarded **Interra Humming Bird Award** in recognition of & appreciation for providing excellent support to Atrenta (I) Pvt. Ltd. in the project "IEEE compliance for Spyglass".

B.E. Computer Science & Engineering – *First Class with Honors*, 85.625/100.00 *Aug'02 - Jun'06*
***Indian Institutes of Engineering Science and Technology, Shibpur*, West Bengal, India.**

- Awarded **University Gold Medal** for securing 1st Rank in BE, Computer Science & Engineering, '02 batch.
- Awarded **Best Student Award**, sponsored by Tata Consultancy Services, for outstanding performance in BE, Computer Science & Engineering, 2002 batch.

Publications

Conferences

- Sandeep Dasgupta, Daejun Park, Theodoros Kasampalis, Vikram S. Adve & Grigore Rosu."**A Complete Formal Semantics of x86-64 User-Level Instruction Set Architecture**", in *Programming Language Design and Implementation*, PLDI 2019.
  [PDF] [DOI] [BIB] [PLDI'19]
- Sandeep Dasgupta & Amey Karkare."**Precise shape analysis using field sensitivity**", in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC 2012.
  [PDF] [DOI] [BIB] [SAC'12]
- Barnali Basak, Sandeep Dasgupta & Amey Karkare."**Heap Dependence Analysis for Sequential Programs**", *International Conference on Parallel Computing*, ParCo 2011.
  [PDF] [DOI] [BIB] [ParCo'11]

Journals & Workshops

- Andrew H. Miranti, Sandeep Dasgupta. "**Formalizing x86-64 Instruction Decoder in K**, in *Workshop on Instruction Set Architecture Specification*, SpISA 2019.
[PDF] [SpISA'19]
- Sandeep Dasgupta, Amey Karkare & P. Vinay K. Reddy. "**Precise shape analysis using field sensitivity**.", in *Innovations in Systems and Software Engineering*, ISSE 2013.
[PDF] [DOI] [BIB] [ISSE'13]

Invited Talks

- Sandeep Dasgupta. "**The K Framework**.", in *Workshop on Declarative Program Analysis*, DPA 2019.
[PDF] [DPA'19]

Poster Submissions

- Sandeep Dasgupta and Vikram S. Adve. "**A Complete Formal Semantics of User-Level ISA**.", in *Office of Naval Research (ONR), Total Platform Cyber Protection Program Meet*, 2019.
[PDF]
- Sandeep Dasgupta, Joshua Cranmer, Edward Schwartz and Vikram S. Adve. "**Binary Decompilation to LLVM IR** ", LLVM Developers' Meeting, 2016.
[PDF] [LLVM Dev Meet'16]
- Sandeep Dasgupta, Barnali Basak & Amey Karkare. "**Dependence Analysis for Parallelization of Sequential Programs**", the 8th ASIAN Symposium on Programming Languages & Systems, APLAS 2010.
[PDF] [APLAS'10]

Other Projects & Services

Projects

- **x86-64 Instruction Semantics** [Pull requests on Stoke]
`Abstract:` Porting my work on most complete x86-64 instruction semantics (Dasgupta et al., PLDI'19) to an open source x86 superoptimizer tool, Stoke, inorder to benefit downstream tools built on top of it.
- **Profiler Using *Clang* based AST Instrumentation** [Github]
`Abstract:` *Clang* based AST reformatting tool used for injecting instrumentation code in C/C++ programs. The goal is collecting profiles (run-times and execution frequencies) on C/C++ programs.
- ***LLVM* based Dwarf Type Reader** [Github]
`Abstract:` To read type information from *debug info* section of executables using *LLVM* based APIs.
- **Decompiling binary to "rich" LLVM IR** [Github]
`Abstract:` One of the limitations of existing binary-to-LLVM IR decompilers is that the extracted IR misses high level informations like variables and types. Moreover, some of them uses a flat array to model the runtime process stack which is shared by all the procedures. This inhibits many aggressive optimizations on the modeled stack because of potential aliases between procedures. Towards our goal of extracting richer LLVM IR from binary, we have implemented the following IR transformation passes on the decompiled LLVM IR: (1) Distinguish individual stack frames, and (2) Identify local variables in each stack frame.
- **Pointer Analysis Debugger** [Presentation] [Github]
`Abstract:` To detect bugs in LLVM's pointer analysis using a mix of static program analysis and dynamic symbolic execution.
- **Partial Redundancy Elimination(PRE)** [Report] [Github]
`Abstract:` Implemented a PRE optimization pass in LLVM and measured results on a variety of applications and did a detailed study on the performance numbers.
- **Extended GCC's Global Dataflow Analyzer** [Report] [Github]
`Abstract:` Extended the Generic Data Flow Analyzer GDFA (of gcc) to the data flow frameworks where data flow information can be represented using bit vectors but the frameworks are not bit vector frameworks because they are non-separable e.g., faint variable analysis, possible undefined variable analysis, strongly live variable analysis.

- **Designing an Interpreter of a dynamic language for Graph algorithms** [Report] [Github]
  `Abstract:` Designed a dynamically typed language and an interpreter for it and achieved a slowdown of 2X w.r.t the execution time of statically compiled C language. This is obtained by providing built-in compiled functions for simple graph computation which in turn help to build complex ones.
- **Mitigating Impact of Heterogeneity Across Power-constrained Nodes on Parallel Applications through Load Balancing** [Report] [Github]
  `Abstract:` Different processors across the nodes have different execution times for the same work-loads. This performance imbalance is seen only when the CPU power is capped to low values. This performance imbalance causes increased execution times of the parallel applications. We did a detailed study and proposed a power aware load balancer (using Charm++) which minimized the performance imbalance at the lower power caps by tackling this heterogeneity.
- **Designing superscalar processor** [Github]
  `Abstract:` Developed a superscalar processor (with an issue rate of 2) using verilog hdl, and an assembler for that processor using flex and bison.
- **Graph Coloring Using State Space Search** [Report] [Github]
  `Abstract:` Leveraged the state space search model for implementing graph coloring in parallel in Charm++. Some of the challenges for efficient exploration of space by chares include intelligent pruning of the state space, load balancing, grain-size control and low-overhead communication between chares. Evaluated multiple options for each of these, and come up with design decisions which would work for a large category of real-life graph applications.

Services

- Sub-reviewer, The 30th International Conference on Concurrency Theory [CONCUR 2019]
- Sub-reviewer, Workshop on Instruction Set Architecture Specification [SpISA 2019]

My Hobbies

Kuk Sool Won

- At the moment I am actively learning "Kuk Sool Won", a Korean martial arts system, which involves systematic study of all traditional fighting arts. This incorporates physical exercise, as well as mental development and traditional weapons training.

Dancing

- I am also fond of dancing and choreography, preferably on Bollywood Beats. Have a look
  - Group@UIUC Durga Puja Festival 2016
  - Group@UIUC Tagore Festival 2016
  - Solo@UIUC New Year Festival 2015
  - Solo@IIT Kanpur, Hall Festival 2011

Table Tennis

- I have spent considerable amount of time playing Table Tennis with excellent players, so ping pong comes naturally to me.