**Operating System Libraries**

January 2014 – May 2014

Developed a command line interpreter to implement redirection, pipeline, and built-in UNIX commands.  
Implemented a memory allocator for the heap of a user-level process to perform the functions of malloc() and free().  
Wrote a multi-threaded code with spin-lock and compared it against pthread lock.  
Built a multithreaded web server based on HTTP with different scheduling policies such as FIFO, Smallest File First (SFF) and Smallest File First with Bounded Starvation (SFF-BS).  
Developed a UDP-based file server and client library to support file handling.

#### **System** **Software Projects**

Developed a shell that interprets the commands of the Unix command line, with support for pipeline, redirection and background processes.  
Implemented a memory allocator library that mimics the function of malloc() and free().  
Developed a multi-threaded web server based on the HTTP protocol, with three different scheduling policies - FIFO, Smallest File First (SFF) and SFF with Bounded Starvation...[**more**](https://www.linkedin.com/profile/view?id=140003200&authType=OUT_OF_NETWORK&authToken=znHU&locale=en_US&srchid=522862671410445063329&srchindex=1&srchtotal=5822&trk=vsrp_people_res_name&trkInfo=VSRPsearchId%3A522862671410445063329%2CVSRPtargetId%3A140003200%2CVSRPcmpt%3Aprimary)

**Operating Systems Libraries and Tools**

February 2014 – May 2014

Implemented malloc(), free() with worst-fit, best-fit and first-fit algorithms and compared  
the performance on different workloads.   
Modified the data structures used to improve storage performance while not adversely affecting allocation efficiency.  
Developed a shell that implemented the features of the UNIX shell and included add-on features to enable redirection, batch scripting and...[**more**](https://www.linkedin.com/profile/view?id=309719095&authType=name&authToken=2O1W&trk=prof-sb-browse_map-name)

#### **Power and Reliability-Aware Scheduling of Real-Time Periodic Tasks**

#### Developed a discrete event simulator in C++ to schedule tasks according to Earliest Deadline First (EDF) algorithm. Incorporated several static and dynamic power saving algorithms preserving reliability. Results for different utilizations and worst-case to best-case computation ratio were obtained and analyzed.

**Implementation of MIPS Processor (16 bit) using Altera – Quartus II**

September 2013 – December 2013

Implemented MIPS arithmetic, memory and control instructions on a 5 stage-pipelined RISC processor**system** with forwarding, register bypassing features.   
Designed a direct mapped write-back data cache using verilog HDL.   
Validated the overall functionality of processor using CAD tool – Quartus II.

#### **Design of 16-bit Pipelined RISC Processor**

Implemented a 16-bit data path and control path based on the MIPS Instruction Set Architecture.  
Developed on this to implement a pipelined processor with data forwarding and hazard detection  
units and verified its functionality.  
Implemented a branch prediction scheme to minimize control hazards.

**Pipeline Design of a Processor**

August 2013 – November 2013

Implemented a 16-bit pipelined RISC processor based on the MIPS Instruction Architecture.  
Implemented Data Forwardsing and Hazard Detection unit to handle data dependencies and veried its functionality.

#### **Design of MIPS pipelined processor**

Design in Quartus including pipelining, branch prediction and cache implementation. Included design of ALU, forwarding, register bypassing, stack operations and control unit.

#### **16-bit single cycle data path and pipelined processor**

November 2013

• Designed a 16-bit single cycle data path processor to implement 16 instructions using Altera Quartus.  
• 5 stage pipelined data path processor with forwarding unit to avoid data hazards and control hazards.  
• Direct mapped, write-back cache using Verilog code.

**Implementation of Adaptive Data Prefetching in L2 Caches**

January 2014 – May 2014

Designed a Stride Filtered Markov prefetcher for L2 data cache on an x86 based CPU using gem5 full**system** simulator and carried out performance and power analysis.  
Achieved up to 15% throughput improvement over no prefetching and 6% over the existing scheme across a set of compute and pointer intensive SPEC CPU 2006, PARSEC, SPLASH-2 and Rodinia benchmarks.

#### **Implementation and Evaluation of Sandbox Prefetcher in gem5 simulator**

January 2014 – May 2014

Implemented the Sandbox Prefetcher for L2 misses in gem5 architectural simulator using C++ and Python. Different configurations of the prefetcher were implemented and compared with the Stride prefetcher using Simpoints. Full system and system emulation modes where executed.

#### **Survey of Low-Power Cache Designs (ECE 752)**

A comprehensive survey of Low Power Cache Design Techniques covering Circuit-Level Techniques (State-Preserving and Non-State Preserving), Micro-architectural techniques, Compiler and OS-based techniques.

#### **Advanced Cache Design: Evaluation of Z-cache and Skewed Associative Caches**

 –

• Modified the source code of the gem5 simulator and added perfect shuffle and H3 hashing functions to model skewed-associativity in caches.   
• Implemented Random, Enhanced NRU and Bucketed LRU replacement policies. Added 2 level, 3 level replacement schemes as proposed in the Z-cache design.   
• Integrated McPAT with gem5 using a Perl script to generate power statistics of last level cache (LLC). McPAT tool was modified for each cache design by describing the CACTI models for LLC suitably.   
• Simulated SPEC 2006 benchmarks on our designs for a 2 core in-order processor and evaluated the performance - power trade-offs of different possible combinations.

#### **CACHE BEHAVIOR**

Implementing the 2-bit SRRIP block replacement policy in Gem5. Make changes in the file - “lru.cc”, and other places (like the block). Run Gem5 with the newly implemented cache replacement policy on benchmarks. Analyzing the LRU and SRRIP cache replacement policies.

**MOSIF Cache Coherence Protocol in**Gem5

The goal of our project was to implement a directory-based MOSIF cache coherence protocol with a two-level cache hierarchy. The L1 and L2 caches are private to a core and support exclusion i.e the L2 cache acts like a victim cache. The protocol was implemented by modifying the MOESI\_hammer files in gem5. To evaluate the benefit of adding an ‘F’ state, the protocol was compared against a base MOSI protocol.

#### **Signature based Hit Predictor High Performance Caching, ACM, MICRO ‘11,**Gem5**Simulator, X86 ISA**

 –

In a team of 3, studied and implemented the Signature based Hit Predictor, a cache insertion policy clubbed with LRU replacement policy for high performance caching. The cache policy was implemented on the Gem5 simulator for the X86 architecture. The performance was studied for parallel applications and compared with that of the LRU cache policy.

#### **Source Throttling of Network on Chip**

 – Present

- Traced the request to measure the congestion and estimate the idea case as reference  
- Achieved 6% performance improvement in terms of latency on self-built cases using tsim ocin  
- Collected information of metrics of baseline test benches under different protocols using gem5  
- Applying source throttling on real cases and tuning the parameters to achieve improvement

#### **Low Power Design for Networks-on-Chip in Chip Multiprocessors (Sponsored by Intel)**

 – Present

• Achieved around 75% energy savings with 5% performance degradation  
• Proposed a novel, low overhead, network status monitoring technique  
• Employed DVFS based PID control policy to manage power consumption   
• Implemented in C++ within GEM5 Full System Simulator

#### **Data Movement Micro-benchmark Development for cache-hierarchies in today’s CMPs (Linux and C):**

• A research project for the development of a data-movement micro-benchmark that measures the latency overhead involved in thread migration applied for temperature control of a core on a CMP. Two workloads were chosen for the benchmark- a compute intensive and a memory intensive workload.   
• Performance of the benchmark was analyzed to determine the effect of data-movement latency in caches during thread migration and compared the results with the performance of the workloads measured when enabling DVFS for temperature control.Results obtained were used to predict a temperature control model for CMPs suitable for both compute and memory intensive workloads.   
[**less**](https://www.linkedin.com/profile/view?id=250558705&authType=OPENLINK&authToken=hB04&locale=en_US&srchid=522862671411081268222&srchindex=13&srchtotal=345&trk=vsrp_people_res_name&trkInfo=VSRPsearchId%3A522862671411081268222%2CVSRPtargetId%3A250558705%2CVSRPcmpt%3Aprimary)

#### **Evaluation of multicore cache hierarchy using PARSEC benchmark simulation on**Gem5**(Linux, C and C++):**

Simulated a 16 core x86 based system with two levels of cache on Gem5 and the Ruby Memory System. The MESI cache coherence protocol was simulated on the cache hierarchy- 16 L1 private caches and a shared L2 last level cache.   
• Performance of the MESI cache-system for four PARSEC benchmarks was analyzed based on L2 cache miss rates, effect of cache-line size on L2 cache misses, true/false-sharing and its effect on the number of invalidates.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#### **Message Passing Interface (MPI) : Hodgkin-Huxely neuron model and ray tracer**

 –

Hodgkin-Huxely neuron model is implementation of neuron model in C using multiple processor system. Different partitioning schemes for ray tracing problem is developed in C using MPI and implemented in multiple processor system.  
Course: Multiple Processor systems

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#### **Design**TwoStageOp**-**Amp**by TSMC025 Technology with SC-integrator & R-2R DAC Application Examples and Demonstrate Geometric Optimization with 0.5um Level-1 model**

 –

First, design a two stage Op-Amp with Miller compensation using TSMC025 technology following a custom specification. Second, demonstrate 2-stage Op-amp design optimization with 0.5um Level-1 model with Mosek geometric MATLAB programming. Finally, design two mixed-signal circuit examples with the designed op-amp (TSMC025 technology) - Bottom-plate Switched-capacitor Integrator and R-2R Digital to Analog Converter.[**less**](https://www.linkedin.com/profile/view?id=184125313&authType=OUT_OF_NETWORK&authToken=q2DA&locale=en_US&srchid=522862671411095360130&srchindex=1&srchtotal=410&trk=vsrp_people_res_name&trkInfo=VSRPsearchId%3A522862671411095360130%2CVSRPtargetId%3A184125313%2CVSRPcmpt%3Aprimary)

#### TwoStageOP**-**AMP

• Designed a two stage folded cascode OP-AMP to meet requirements for DC Gain for first and second stages, slew rate, phase margin, unity gain frequency, input and output ranges. First stage is high gain and second stage is high swing. To choose optimal transistor sizes with acceptable bias currents individual circuits were built to perform DC analysis and parametric analysis. Assumption was made that first stage capacitance is due to the Miller Capacitor. All bias voltages were supplied using cascode current mirrors. Final design requirements were met and an AC signal was amplified to test the circuit

#### **Design of a**two**-**stage**Miller Operational Transconductance Amplifier(OTA) and its implementation in a 5th order active filter**

• Designed a two-stage Miller compensated OTA using 0.25µm CMOS technology.   
• The schematic and layout were completed using Cadence Virtuoso.   
• Schematic was simulated using Cadence Spectre.   
• A 5th order Butterworth filter was designed and then simulated in Cadence Spectre using the designed OTA.

#### **A Fast-Settling, High-Gain**Op**-**Amp

The project required designing of a fast-settling, DISO operational amplifier to be used in a highly linear voltage buffer. Using the cadence tool the design was design. A single-stage folded cascade Op-amparchitecture was adopted for the design. Challenge in the work was the trade-off between the 60-dB gain that dictates long-channel devices and the fast settling time of 10 ns that requires short-channel devices for high-speed operation.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_