INTRODUCTION

We are pleased to provide this brochure highlighting our second year of the Corporate Sponsored Senior Project Program! Our students have worked very hard during their time at UC Santa Cruz earning their engineering degree and fulfilling this capstone design sequence.

“If students are to be prepared to enter new-century engineering, the center of engineering education should be professional practice, integrating technical knowledge and skills of practice” (Sheppard et al., 2009). Students who have participated in this Corporate Sponsored program have been provided with a unique opportunity to experience working on real-world projects that involve design, budgets, deadlines, teamwork and reviews with their team mentor. They have come away with a sense of professionalism and pride in their work, new technical skills that may have been challenging, and experience in entrepreneurship and the implications of intellectual property.

Throughout this academic year, the students have interacted with their teammates, some have made visits to their corporate sponsor’s worksite and all have solved problems that arose along the way. The students take great pride in their completed projects and all they have accomplished during their time at UC Santa Cruz and the Baskin School of Engineering.

We also take great pride in what the students have accomplished. We are very grateful to our corporate sponsors for their willingness to sponsor this year-long program, mentor our students and provide them with challenging projects to work on.

Arthur P. Ramirez
Dean
Baskin School of Engineering

ACKNOWLEDGEMENTS

We would like to acknowledge and thank the faculty and staff who have been so instrumental in the Corporate Sponsored Senior Project Program:

SENIOR DESIGN FACULTY

Patrick Mantey—Director, Corporate Sponsored Senior Project Program and Associate Dean for Industry Programs
soe.ucsc.edu/people/mantey

David Munday—Teaching Fellow, Computer Engineering

Linda Werner—Adjunct Professor, Computer Science
soe.ucsc.edu/people/linda

CORPORATE SPONSORED SENIOR PROJECT PROGRAM STAFF

Brenna Candelaria—Project Assistant

Tim Gustafson—BSOE Technical Lead/BSOE Webmaster

Carolyn Hall—Director of Resource Planning and Management

Liv Hassett—Associate Campus Counsel

Frank Howley—Senior Director of Corporate Development

Heidi McGough—Program Coordinator/Executive Administrative Assistant, Dean’s Office

David Meek—Development Engineer, Baskin Engineering Lab Support

Christian Monnet—Development Engineer, Baskin Engineering Lab Support

Arthur Ramirez—Dean, Baskin School of Engineering

Lynne Sheehan—Network Administrator, Facilities/Machine Room

Anna Stuart—Senior Budget Analyst

Bob Vitale—Director of Laboratories and Facilities

AFFILIATED SENIOR DESIGN FACULTY AND TEACHING ASSISTANTS

Paul Naud—Teaching Assistant

Ethan Papp—Teaching Assistant

Stephen Petersen—Lecturer, Computer Engineering

Anujan Varma—Professor, Computer Engineering
soe.ucsc.edu/people/varma

John Vesecky—Professor, Electrical Engineering
soe.ucsc.edu/people/vesecky
SPECIAL THANKS to our sponsors for your generous support of our Corporate Sponsored Senior Projects Program. Your time, experience, and financial support were beneficial to our students and the success of their Senior Design Projects.
The goal of this project was to examine openCV’s face detection software and identify the time insensitive section of the code. Then take those sections and attempt to accelerate the face detection algorithm by running it hardware instead of software. In order to accomplish this task Altera gave us a Cyclone IV PCI express board. Our final goal being able to start the face detection in software, having it ship data to the Cyclone IV FPGA logic to perform part of the face detection algorithm and then back to software where it would display the results.

Introduction

The first step was to analyze the OpenCV implementation of the code and determine which portion of the software had the longest run time. The HaarDetectObjects function took significantly longer to execute than any other data load or image processing functions. To accelerate this function, the design was split into three modular functions:
- Transferring the image data in memory and translating its address
- Transferring the Haar Classifier data in memory and encoding it into our design
- Implementing the Verilog code for the classification algorithm

Methodology

The initial task was to simulate the Haar Classifier algorithm in hardware before trying to optimize and parallelize the design. The block diagram in the above figure represents the design of the Verilog project. A stage comparator master state machine operates at the upper level to manage which classifiers should be extracted from the feature classifier data at what time. Once the feature classifier data has been loaded, the master calls the feature classifier slave to perform the slave functions of calculating the area sum of the rectangles and the classifier’s right and left threshold. When the window region has finished its testing, the slider will iterate to a new region in the image to test. If the slider reaches the end of the image, it will enlarge the image and undergoes the same procedure again. This continues until all possible image windows have been exhausted.

Face Detection

While there are a number of different methods for face detection, for this project Haar Cascade Classifiers for face detection was chosen. This method was selected for a number of reasons, the main one being the fact that Haar Cascade Classifiers was one of the first algorithms capable of real-time face detection in software. It did this by forgoing a large complex mathematical equation, in favor of thousands of small, fast computations. Since complex math can be difficult and expensive in hardware, this algorithm was a perfect candidate for hardware acceleration.

In software, Haar Cascade Classifiers works by comparing thousands of Haar-like features, or rectangular sections of the picture, to a value from an input XML file created by a training program. It does this sequentially in 22 separate stages, breaking out of the algorithm if any one of the stages fail to detect a face.

Results

Algorithm times before acceleration:

<table>
<thead>
<tr>
<th>Stage</th>
<th>Time Before Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage 1</td>
<td>2.2 ms</td>
</tr>
<tr>
<td>Stage 2</td>
<td>2.3 ms</td>
</tr>
<tr>
<td>Stage 3</td>
<td>3.1 ms</td>
</tr>
<tr>
<td>Stage 4</td>
<td>3.6 ms</td>
</tr>
<tr>
<td>Stage 5</td>
<td>5.2 ms</td>
</tr>
<tr>
<td>Stage 6</td>
<td>12.5 ms</td>
</tr>
</tbody>
</table>

Predicted performance boost:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Speed Up Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detect Haar Objects</td>
<td>48x</td>
</tr>
</tbody>
</table>

*results include a 22ms PCIe transfer delay and are estimated from the runtime measurements of Detect Haar Objects and similar speed up tests performed by Junguk Cho et al. and Jason Oberg et al.*

Conclusion

From the above results, it is clear that a noticeable performance boost is achieved in Detect Haar Objects by running it in hardware. However, because of constraints on the size of our FPGA and limited memory access it was impossible to completely parallelize the algorithm like originally hoped, with all 22 stages of the algorithm running simultaneously in hardware. Instead each stage and Haar-classifier had to be run sequentially, because of this, it is possible to improve upon these results even farther by increasing either memory accessibility to allow multiple simultaneous accesses to memory, or increasing the number of logic cells on the FPGA to fit more instantiations of our Haar-Classifier module in the hardware which would allow more Haar comparisons to happen concurrently.

Acknowledgments

Clive Davis – Altera Sponsor
Adam Titley – Altera Sponsor
David Munday - Senior Project Mentor
Ethan Papp -Senior Project Assistant
OLED Luminaire

William McGrath  Randy Chen  George Weichhardt  Andy Jia Liang Shen  Trenton Louie

JACK BASKIN SCHOOL OF ENGINEERING
Team OLED Senior Design Project

Motivation

The organic light-emitting diode (OLED) panel emits soft, even light that is pleasing to the eye, making them ideal for personal lighting applications. It is a largely unexplored technology in the lighting industry, making them expensive to manufacture and difficult to purchase, and scarce in product design. This project aims to showcase the potential OLEDs have in the lighting industry by designing an innovative luminaire. The luminaire functions as a desk lamp and has the ability to control the light. This information is used to adjust the brightness of OLED panels.

Approach

• Design power supply to regulate power from mains electricity (120V AC) and power an array of OLEDs.
• Design a capacitive touch slider and a light sensor to adjust the brightness of OLED panels.

Power Distribution

Mains — AC-DC Rectifier — 12V LDO — OLED Array
— DC DC Converter — 5V LDO — Microcontroller & Sensors

After 120V AC RMS is rectified, the first voltage reference linearly converts 170V DC RMS to 170V. A source-follower power MOSFET and zener diode in reverse breakdown set this 10V to drive the power management integrated circuit (PMIC).

Results

• The custom OLED driver is automatically dimmable by ambient light detection or manually dimmable with a capacitive touch slider. The dimming functionality allows for an adjustment of output lux from 15 to 250. The lux output and corresponding power can be observed in figure 7.
• A mechanical design allows the lamp to illuminate a desk or user’s face. This information is achieved through collaboration with a local machinist and a local lamp designer.
• Reverse bias research indicates that the capacitance of these panels is too large and are not suited to switch at the target rate.

Switching Mode Power Supply (SMPS)

Back controller SMPS topology uses the LM3445 as a power management integrated circuit (PMIC).

Experiment:
• Used a trans-resistance amplifier to measure the photocurrent.
• Panel discharge off time, light measurement time, and charge on time were measured on panels in zero bias and reverse bias configurations in figures 5 and 6 respectively.
• The LG panels used in the final prototype were measured to have more than 400 ohm of capacitance.
• Placed capacitance in series and resistance in parallel with the panels to reduce the RC response which decreased the on and off charge time from 12 ms to 5 ms.
• The amplifier output took less than 3 seconds to stabilize for a light measurement.

Future Work

Large capacitance in the panels prevents fast switching. The capacitance associated with each panel is related to the size of the panel and the internal characteristics. A reverse bias switching configuration is feasible with smaller panels that have reduced capacitance or panels made from compounds with smaller dielectric constants.

Acknowledgements

David Munday, Patrick Munday, University of California at Santa Cruz;
Dr. Robert Jon Vasse, Applied Materials;
Mike Liu, Peter Nguyen, Enegle-Hai, Axensity Brands;
Chris Brightman, Illuminée;
Bob Lund, Amstrit Corp.
Sponsored by Dell KACE, ITNinja is a rapidly growing community where IT professionals ask questions and share ideas related to enterprise software topics, systems management, and other technical topics and new technologies. Each month, half a million IT professionals come together to discuss topics related to setup and deployment while also sharing experiences and solutions with IT management peers around the world.

Dell KACE asked us to create a tool and an API that would allow users (without programming skills) to query any data in the ITNinja database and display that data in a portable widget that can be placed anywhere on the web.

Creating this widget will also increase exposure and traffic to ITNinja.com

Usability
• Creates a way for users to experience and use data in a new interactive format that’s customized and personalized while requiring no programming experience from the user.
• Tailored towards a wide spectrum of people including technical developers and non-technical IT administrative staff.
• Users can proudly display their contributions to the ITNinja community on their own blog or website.
• Provides ability to extract data from ITNinja in thousands of different ways and share that data anywhere on the Internet.

How and What

User interaction Flow Diagram

Systems Interaction Diagram

Portable Code

The widget is divided into two parts, the list view pane and the preview pane. In its inert state, the widget will only show the list view. Upon mouse-over on the widget, the preview pane will be displayed.

The list view shows the title, author, and how long ago the post was created, while the preview panel gives a short preview of the selected content.

Technologies Used

Front-End:
• JQuery, CSS 3, HTML 5, and JavaScript for building the widget.
• AJAX and JSON for passing JS code to the server.
• CodeMirror for formatting the portable code.

Back-End:
• PHP 5.3 for interacting with the MySQL database
• PHP Data Objects for preventing data injection
• MySQL as ITNinja’s database
• Slim Framework for formatting the data and using RESTful practices with ITNinja’s content management system

Acknowledgements
Senior Software Director Brian Link for being our corporate sponsor and mentoring us for this project.
Professor Linda Werner for being our advisor and providing resources to complete this project
On eBay, a seller has the freedom to use any photo to represent their item-for-sale. Our project is the design and creation of an Internet accessible application that uses crowdsourcing and gamification to assist eBay in the discovery of the most appealing photo to represent each specific item. We have created a game which uses a racing game mechanic to motivate players to select a photo they believe to most accurately represent an item. Motivated to win, the player competes against other ‘virtual players’ to choose the “best” photo, and once selected, a new round of photos appear along with a new listed item. The overall photo scores are saved, and we believe the most frequently chosen images are those most likely to carry common aspects of what makes a photo ideal.

Abstract

Software Architecture

Technologies Used

• Node.js (server)
  – Socket.io (connection)
  – DB-mysql (data storage)
• Ebay API
• MySQL (ebay api cache database)
• Django (game-client host)
• JavaScript (game client)
• HTML (client GUI)
• CSS (HTML style)
• Amazon ec2

To advance, the player is given a round of pictures for an item named in the textbox. Depending on prior crowd-sourced data and ranking, the player advances with a weighted amount of boost. The player’s vote will be sent to the database. Once the choice is made, a new set of pictures for a different item is displayed.

Acknowledgements

• Corporate Contacts: Gyanit Singh, Dr. Neel Sundaresan, Dr. Yana Huang
• Faculty Advisor: Dr. Linda Werner
Abstract

Echelon Corporation's LonWorks technology has been a popular control system technology with over 100 million devices installed around the world in many diverse applications since the 90s. Echelon's customers have created wireless transceivers for LonWorks networks, but Echelon, until now, has not.

The purpose of this project is to expand Echelon's portfolio by integrating the LonTalk network protocol with RF radio over IPv6, as well as integrating Echelon's Interoperable Self Installation protocol (ISI) which is used to simplify the device installation for all users.

Overview

LonWorks, a networking platform built upon the LonTalk and Interoperable Self Installation protocols, is used for automated control systems such as smart home, smart grid, smart city and smart control.

LonTalk provides more reliability over User Datagram Protocol (UDP) sockets with its LonTalk Services, which include Unacknowledged Repeated packets and Acknowledged packets. In addition, its Services also include Authentication, Request/Response messages, Network Management and Addressing management, and end-to-end reliability. The Network Layer handles node address information and routing.

Interoperable Self Installation protocol (ISI) adapts a peer-to-peer control algorithm that allows end devices to automatically or manually form a network between sensors, actuators, or controllers depending on the application. Figure 1 illustrates an example topology of a LonTalk Network Group, where each device (blue box) within the network (cloud) publishes information through the ISI protocol and makes decisions based on the received information.

Figure 1 – Example Topology of LonTalk Network Group

Figure 2 – LonTalk/ISI Stack Block Diagram

The BEFORE diagram above shows LonTalk's existing networking stack. The LonTalk Services are implemented throughout the stack. The Session Layer contains request/response message handling and retransmissions. The Transport Layer contains duplicate-packet detection, addressing, management, and end-to-end reliability. The Network Layer handles node address information and routing.

The AFTER diagram shows the LonTalk stack after integration with the Jennet radio modules. The major changes that were made include: a new adaptation layer which interfaces the bottom of the LonTalk stack with the Jennet module's UDP sockets. This allows for packets of the LonTalk Services to be sent with IPv6 through the Jennet modules.

Figure 3.a and 3.b shows the non-transient state diagrams of a host and its subscriber ISI devices. By establishing a connection, an ISI device becomes a host. A host asks for an assembly with associated network variables and its addressing mode throughout the network. Other devices in the network will detect the request and decide whether or not to accept the invitation. If a device has an available connection and decides to accept the invitation, it becomes a subscriber of the host and sends back an acceptance message to the host. Next, the host detects the acceptance message and responds with a confirmation message to the subscriber. In addition, the host will bind the assembly to its connection table. When the subscriber receives the confirmation message from the host, it will bind the assembly to its connection table. The connection is then established.

Devices join a network group using the Interoperable Self-Installation (ISI) protocol, which interfaces with LonTalk's application layer. Each device will select a random IPv6 address and send it using ISI's "fire and forget" algorithm, which broadcasts the device IP address to all devices in the group without waiting for a receipt confirmation. If an address collision occurs, the receiving device will change its address and resend its new IP address. The entire network will make this adjustment until all devices in the network have a unique address.

Figure 3.a – ISI Host State Diagram

Figure 3.b – ISI Member State Diagram

Acknowledgements

- Prof. Patrick Murray and Anjana Verma, UC Santa Cruz
- Bob Delin, Bob Walker, Glen Riley and Bernd Grauweiler, Echelon Corporation
- Sam Morris and Sho Smith, NXP Semiconductors
- NELS, David Munday, Paul Naud and Ethan Papp, Baskin School of Engineering

Results

Our final protocol supports most of the functionalities of the original LonTalk and ISI protocols. We were however unable to incorporate some other services due to the small code size requirement for the protocol in order to leave a small footprint on the modules and provide the maximum amount of space for the customer application.

Our major achievement includes:
1. Using IPv6 address space which minimizes address collision within a network group.
2. Implemented ISI-S and core LonTalk Services.
3. Small source code which maximizes the flash space for customer application.

Hardware

We work with Jennet IEEE 802.15.4 RF radio modules by NXP Semiconductors.

- The radio module supports IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) which significantly shrinks the IPv6 packet size by reducing the overhead.
- The radio module has exposed UDP sockets in its hardware API.

Integration

The integration of LonTalk began by putting over minimal functionality to get each LonTalk Service working one at a time. We attempted to transmit unacknowledged messages by sending the data packet through the application layer on the sender's end, and having the message go down the stack to the network layer. After getting packet down to the network layer, we modified and saved the bottom of the LonTalk stack to the Jennet UDP Socket with an adaptation layer, which defined the packet to fit the Jennet Interface. Thus, the next step was to send the packet back up the stack on the receiver end and ensure the header was being read and taken off the packet correctly. Once the packet had made its way to the application layer on the receiver's end, the data was sent to the appropriate application. These steps were repeated for each new addition of the LonTalk Services.

The source code for ISI was written for Echelon's custom hardware, Neuron chip, which uses Echelon's Conference language called Neuron-C. The hardware API is written in Echelon's Assembly language called Neuron Assembly. Our main task in the ISI integration is to translate source code into C and reproduce C functions that emulate Neuron-functions that ISI use.
Portable Environmental Data Logger
Carol Owens, Brian Chen, Nathan Krueger, Hart Wanetick, Cody Harris
JACK BASKIN SCHOOL OF ENGINEERING
Senior Design Project

Motivation
Many environmental data loggers are large, expensive, stationary and mostly unavailable to the public. While agencies like the United States Environmental Protection Agency (EPA) operate loggers to monitor air quality in many cities, the data is presented as an average over a large region, often using a small number of stationary data sources for an entire city.

The Portable Environmental Data Logger project (PEDL) was created for Google as an end-to-end system that measures, stores, and visualizes measurements of environmental phenomena on a local scale. PEDL is portable and can be mounted to cars or other vehicles and travel around a region collecting measurements.

PEDL aims to increase public awareness about air quality by presenting the data online using simple interactive visualizations enabled by Google technologies. Most importantly, people may be more interested in environmental data if they can easily see data collected in their own neighborhood.

Objectives

Low Cost: PEDL was designed to be portable and lower cost than conventional data loggers. The logger is weather-proof and rugged for practical everyday outdoor use. Both the software and the hardware were designed to be modular so they would be open to future expansion.

Reproducible: The system was designed to be easily reproducible; all source code, hardware schematics and the bill of materials are freely-available for Google and the public’s benefit.

Comprehensive: PEDL’s sensor suite measures climate and pollution which creates a comprehensive picture of the local environment at the time of the measurements taken.

Future Work

Two PEDL units and a public-facing software GUI were built over the course of 20 weeks. PEDL could be modified relatively easily to run without the need for an external Linux PC that controls the system. This could be accomplished by combining the software and running it all on the BeagleBone. By also incorporating a cellular modem, the system could be made completely autonomous incorporating a cellular modem and an onboard Ethernet controller for communicating with the sensors and an onboard Ethernet controller for communicating with the host Linux PC.

Software Implementation

The Linux PC and embedded Linux board in PEDL communicate over TCP sockets. The Linux PC sends commands to the BeagleBone; receives sensor data via serialized Google Protocol Buffer messages, and uploads it to Google Cloud Storage.

Once the data is uploaded and buffered to Google’s Cloud Storage, it is parsed and inserted into a Google Cloud SQL database. A web app running on Google App Engine queries this database and renders geospatial and time-series visualizations of the measured data.

Hardware Implementation

PEDL uses both analog and digital embedded sensors. The five digital sensors measure temperature, pressure, humidity, radiation and carbon dioxide. Two analog electrochemical sensors measure ozone and nitrogen dioxide.

Development of hardware for the PEDL system involved multiple revisions of custom printed circuit boards, the designs for which are all open-source. The latest system circuitry uses three custom PCBs along with the BeagleBone development board and dedicated PCBs for each of the CO2, NO2, O3, and radiation sensors.

The system uses several protocols to read sensor measurements. Three digital sensors communicate using I2C, a popular and straightforward communication protocol. The carbon dioxide sensor communicates using the Modbus protocol and a two-wire serial protocol is used to communicate with the humidity sensor. The two analog electrochemical gas sensors are multiplexed and read by an ADC that also communicates over I2C.

A BeagleBone, a popular embedded Linux platform, communicates with a host Linux PC (as specified by Google) and collects data from the environmental sensors. The BeagleBone has GPUs, dedicated UARTs and PCI hardware used for communicating with the sensors and an onboard Ethernet controller for communicating with the host Linux PC.

The PEDL system is entirely enclosed by two NEMA-rated weatherproof boxes that are joined together. All gas sensors are given samples of the outside air by two diaphragm pumps. Intake and outtake ports for the pumps are angled glands coming out of the main box that prevent moisture build-up and filter the air for large particles. The climate sensors are positioned in a small vented box that is coupled to, but sealed off from, the main enclosure.

Acknowledgements

We would like to thank:
- Google for sponsoring us and Matt Thrailkill and Karin Tuxen-Bettman for supplying us with the idea for this project, mentorship, and technical support
- David Munday for his guidance, insight, and mentorship
- Michael Lisk for helping us test many of the sensors used on this project
- The Bay Area Air Quality Management District, specifically Stanley Yamazaki and Linda Ruth for helping us test our toxic gas sensors
- Santa Cruz Institute for Particle Physics for helping us calibrate our Geiger counters
- Baskin Engineering Lab Support for their assistance and support
What2Watch: A Mobile Application Recommendation System

Cullen Glassner, Quentin Rivers, Sam Sanders with Aryeh Hillman, Chris Lopez

Abstract

Netflix is one of the world’s leading video streaming services. As part of Netflix’ constant search to develop innovative and interesting ways to provide recommendations, we designed and created What2Watch, an application for Apple’s mobile operating system, iOS. What2Watch is a movie recommendation application designed to give users an easier way to discover new movies and television shows about which they otherwise would not have heard. The What2Watch application uses short interactions to utilize a user’s small downtimes. Our goal was to create a modular system that will facilitate the testing of several different recommendation algorithms and user experiences with as few confounding factors as possible. We created the application adhering to the Netflix aesthetic standards and technologies for an easy integration into the larger Netflix ecosystem.

Introduction

This project addresses the issues of testing an AI and UI within a mobile application recommendation system. Our goal was to create an application for iOS devices that could be loaded through the Netflix application, so “plug-and-play” different AI algorithms and user interfaces. Our team used Scrum, an agile process, to manage the tasks and workflow throughout this two quarter project. This form of management involves short, regular meetings between all engineers, with daily to weekly updates. Learning to create user interfaces was a challenge. This involved learning new technologies such as HTML, CSS, jQuery, etc. Programming for mobile devices was another new area for each team member. Through this project we learned much about programming for mobile applications as well as managing groups, and working dynamics.

Future Work

• Add the Netflix home page to more fully integrate with the Netflix user experience.
• Use user test feedback to improve the recommendation algorithms.
• Develop new recommendation algorithms and incorporate them into the existing application.
• Develop alternate or additional UI views for the app, based on user testing feedback.
• Implement the ‘genius button’ algorithm. The genius button would serve as an instant recommender.
• Integrate What2Watch into the larger Netflix ecosystem and user experience.

Architectural Design

We took many different factors into account when we were designing What2Watch. Some of these factors were functional requirements, but many of them were non-functional. We had to design for interoperability, extensibility, understandability, ease of learning, general usability, and scalability.

Interoperability and scalability were important considerations since What2Watch needed to be able to be integrated into the larger Netflix ecosystem. This meant that we had to use technologies which allow for a seamless integration and reorganization with other Netflix products and technologies. It also meant that What2Watch had to either be currently scalable to millions of users, or easily changed to be scalable to millions of users. This need for integration also partly led to the importance of understandability, ease of learning, and general usability. What2Watch would not be fully integrated into the Netflix ecosystem if its user experience were jarringly different in terms of style or quality, so we had to maintain the same levels of understandability and usability. Further, What2Watch will be used to test and compare user satisfaction of various recommendation systems. In order for the results to be useful, the user experience needs to have a negligible impact on user satisfaction.

Using What2Watch as a platform for testing and evaluating AI algorithms also meant the system had to be extensible. Without sufficient extensibility, it would be difficult to readily change out AI algorithms without having to redo a large amount of work. Further, insufficient extensibility would not allow for the use of different sources of data for AI algorithms.

All of these non-functional requirements were met by creating an extensively modular application. There were three large components in What2Watch - the web application, the data and logic server, and then the Netflix servers with user data. The first two components were designed and created by us, and were themselves deeply modular in nature to meet these non-functional requirements. By making all of the components so modular, we made adjusting the understandability, ease of learning, and general usability extremely easy. We can simply adjust the user interface to adjust these factors. No back-end portions of the system ever needed to be touched in order for us to adjust the user interface.

In the same way, modularity allowed for the interoperability and scalability that was required. By maintaining a separate data and logic server, when we need to scale to millions of users, we can just change out our current back-end for a more scalable back-end. Further, since the AI is separate from the REST portion of the server, it will be incredibly easy for us to change the data and REST portion of the server while maintaining our AI algorithm.

This deep modularity also solved one of the most important non-functional requirements - extensibility. Since What2Watch is intended to first act as a test bed for recommendation algorithms before going live, having the AI separate from the rest of the system allows us to change out AI algorithms quickly and easily. By being able to swap out algorithms so easily, we can rapidly prototype and iterate over many different recommendation algorithms.

Technologies

• Apache HttpClient 4.2.3 - Java http client, used to communicate with the Netflix API.
• Java 1.4 - Programming language used to implement the server and AI.
• JSON.simple 1.1.1 - Java JSON parser
• jQuery 1.7.2
• Jetty 9.0.2 - Java web server
• HTML
• Java/crdt
• LESS 1.3.3 - CSS extender
• Git 8.2.3 - Version control

User Interface

The user interface for this application is loaded through an iOS application, and as an HTML web page. This allowed for us to program and test on non-iOS devices. Many different options for the interface were present, using different methods of obtaining user input, as well as varying forms of displaying the intelligent results from our AI algorithms. Below is one example of the flow of this application.

Acknowledgments

Thanks to the following,
At Netflix:
• Chris Jaffe, Director - Product Innovation
• Sam Pan, Director - User Experience
• Mike Cohen, Senior Software Engineer
• Joel Beukelman, Senior User Experience Designer
• Jackie Joyce, Senior Manager - Enhanced Content

At UCSC:
• Linda Werner, Faculty Advisor

*Taken from Netflix presentation January 17th, 2013
Abstract

The aim of this project is to increase the speed of Oracle Number divisions by developing a fully custom hardware solution. Oracle Numbers is a proprietary high-precision decimal floating-point format used in Oracle servers. The achieved cycle latency of our final solution is 25 fan-out of four (FO4) delays, targeting high-precision decimal floating point format used in Oracle servers. The achieved performance metrics for both the Nikmehr and Lang dividers. In fact, Figure 4 shows that 22 copies of the Lang divider block are equivalent in area to one Nikmehr divider block. The relatively low area of the Nikmehr divider block suggests that we may use multiple copies of the Lang divider block to pipeline the design, yielding an improved throughput proportional to the number of divider block copies implemented. This idea for improvement is left as future work for this project.

Results

Both of our designs offer a significant latency improvement over the current software implementation, with the Nikmehr divider taking 105 ns for a full-length division and the Lang divider taking just 76.5 ns for a full-length division. Figure 5 shows this speedup of 352.9 times faster than the current implementation, which is performed in software.

Oracle Numbers have variable length and precision, with the capability of storing up to forty decimal digits of precision in the mantissa. Each two adjacent decimal digits of the mantissa are encoded and stored in base-100 values called Oracle Digits (ODs). The entire number is prefaced with two eight-bit header fields, the first specifying the number of Oracle Digits in the mantissa, and the second containing the sign and exponent of the number.

Overview

Our design consists of three phases: decoding, division, and encoding. In the decoding phase, the Oracle Digits are converted to binary coded decimal (BCD) digits, and Oracle Numbers which have less than the maximum number of Oracle Digits are extended with trailing zeros to their full length. Once the operands are in the form of forty BCD digits, factors of the divider are calculated for use in the next stage. The division stage is iterative, yielding one quotient digit per iteration. More detail on this stage is given in the following sections. In the encoding stage, we round the least significant mantissa digit and convert the BCD digits back to base-100 Oracle Digits. The length and sign / exponent fields are also adjusted accordingly in this stage.

Design Overview

Nikmehr’s implementation of SRT is similar to the classic long division algorithm. The implementation requires us to generate nine factors of the divisor, from $a^7$ to $a^0$. Nine three-digit comparison multiples are generated based on these divisor factors, which partition the range of the partial remainder. The Quotient Digit Selection (QDS) unit determines the interval in which the previous partial remainder falls to select a quotient digit. The divisor factor corresponding to this new quotient digit is subtracted from the previous partial remainder scaled by ten to calculate the new partial remainder.

Design 1: Nikmehr

The partial remainder formation unit is the same as the Nikmehr divider block. In the QDS unit, each quotient digit is formed by separating the selection process into two components, a high-component which selects from the set {$\pm 2, \pm 1, 0$}, and a low component which selects from the set {$\pm 2, \pm 1, 0, \pm 1$}. By separating the quotient digit into multiple components we are able to save time and power by removing the need to store a full divisor factor set.

In order to operate QDS and the decimal partial remainder formation unit in parallel, a truncated binary representation of the partial remainder must be calculated separately from the decimal partial remainder so as to avoid the delay that comes with a full decimal carry save addition. However, to ensure the accuracy of that truncated partial remainder as the frame of magnitude drifts from the initial calculation, we must compensate by periodically updating the partial remainder.

Design 2: Lang

Acknowledgements

The authors acknowledge support from the following organizations for their contributions:

- Hooman Nikmehr, UC Santa Cruz
- Ali Celebioglu, UC Santa Cruz
- Jack Baskin School of Engineering

References

Machine Learning Algorithm Classification

Shelby Thomas  Jeff Johnson  David Lau

JACK BASKIN SCHOOL OF ENGINEERING
Senior Design Project

Abstract
With the ability to gather and store vast amounts of data comes an analysis in order to find current and possible future trends. Many of these data processing techniques fall into the broad category of machine learning algorithms. These algorithms, while not new, have only recently increased usage outside of scientific research. Example of these algorithms can be found in recommendation systems, self-driving cars, and natural language processing. While the applications of machine learning algorithms are vast, they are computationally expensive. The motivation for this project is to find which computations found in these algorithms have the largest performance impact on a modern processor. Previous work in this field of study is limited to high level benchmarking rather than benchmarking at an architectural level. The goal of this project is to examine the modern Intel CPU architecture and determine where the most improvement can be had with respect to branching, and caching. This information would allow future optimization and provide insight for new machine learning algorithms.

Algorithm Hotspots
Support Vector Machines
Support vector machine is an algorithm whose goal is to find a linear separator to achieve binary classification given a set of training data. The biggest bottleneck of this algorithm resides in the way the algorithm trains in data using sequential minimal optimization. Sequential minimal optimization, or SMO is an algorithm that trains and creates the linear classifier. The algorithm utilizes a CBLAS library call CBLAS_DDOT which consumes more than 90% of the total runtime and results in a last level cache miss rate of almost 49%.

Optimization Potential
Two methods exist, the first method includes re-processing the data in order to minimize the amount of calls to the dot product function. The second involves taking a mathematical approach and finding out if there is an alternate training method other than SMO that uses fewer dot product calls.

ECLAT
ECLAT is an Association Rule Discovery (ARD) algorithm which recognizes patterns in transactional data. There are many applications for ARD algorithms which include finding recommendations for videos, friends, and television shows.

With an input dataset of 66612 Transactions and 600 items, the largest bottleneck in the algorithm is the high last level cache (LLC) miss rate. The bottleneck is caused by adding items to Transaction ID’s (TID) in a linear fashion. Since each TID in an object, many TID objects become evicted from the L1, L2, and LLC after many iterations of the algorithm. It is clear that this causes a problem with temporal locality.

Optimization Potential
One way to fix the temporal locality problem is to make sure that a subset of TID’s is accessed for many iterations before moving on to the next subset of TID’s. By only accessing smaller blocks of TID’s performance can be improved by reducing the number of cache evictions and ultimately LLC misses.

HOP
HOP is a clustering algorithm which seeks to assign particles in 2 or 3 dimensions to a group based on each particle’s proximity to other particles.

Given an input set of just over 64,000 particles, HOP performed most with regard to CPU branch mispredictions. The bottleneck was found in a macro called very often which adjusts a variable used in filling a priority queue. The macro contains roughly 17 tested conditions (7 total if, else, else-if, else statements).

Optimization Potential
One way to decrease the number of branch mispredictions is to simply eliminate the branches in question. Because the branch patterns are dependent on the input data, a computational substitute could replace the testing of the branch conditions. At least a 9% decrease in runtime could be achieved solely by eliminating the branch miss penalties occurring in the macro.

Data Collection
The algorithms were classified using Intel’s VTune Amplifier. When collecting the data the metric used were Cycles Per Instruction (CPI), L1 Miss Rate, LLC Miss Rate, Cache Miss Impact and Branch Miss Rate. The data was then compared with several different algorithms to find the 3 that performed the worst: Support Vector Machine, ECLAT and HOP.

Results and Future Work
Support Vector Machines
The first step to ensuring the findings were accurate was to pre-compute the dot product results and store them in a 2D array. Then let the training algorithm use these results, rather than the dot product call. Although this is not an ideal solution it provides a good indication on how much the runtime could be decreased if the frequency of these calls are reduced.

This method resulted in a runtime decrease of 66% and reduced the last level cache miss rate by almost 90%.

Future work for this algorithm includes looking at alternative ways to train the data outside of just using SMO. There are other widely used training algorithms including SVMLight and chunking. The algorithm can be parallelized using these training algorithms and possibly creating a hybrid implementation that reduces vector-vector operations.

ECLAT
To optimize ECLAT array tiling was utilized. This method works by only accessing a small subset of the array (called blocks) at any given time. In the ECLAT code, the array was an array of Transaction ID Objects.

This resulted in a runtime decrease of 37% and reduced the last level cache miss rate by 64%.

Future work for this algorithm includes finding better data structures to store the TID-ham data and improving the run time of other portions of the algorithm.

HOP
In order to confirm that the misses occurring in HOP were primarily in the INTERSECT macro the conditional branches were converted into indirect jumps. Since VTune can distinguish between these two events, it was possible to easily track any shifts. Initially the macro path was one-hot encoded and those integers were put into an array in an include header file. These were used as flags for computing a target for an unconditional jump. Those pointers were then multiplied by the flag and examined. The result is in the address of the correct code to execute calculated using data from previous runs.

Future work for researching branch prediction should include testing different architectures and older chip designs. This would allow a comparison across a variety of branch predictors. Alternative implementations of the macro without conditional branches should also be explored.

Acknowledgements
1. Thanks to Oracle for sponsoring and fraction Host Guild for monitoring us.
2. Thanks to David Murphy and Ethan Pokluda for their guidance.
3. Thanks to Professors Ivan Ranon, David Hembold, and Herbie Lee for making themselves available to answer questions.
4. Thanks to Northwestern University for providing support for the machine learning benchmarking suite, Mahout.
Abstract
The objective of this project was to develop an ultrasonic modem, meaning a way to transmit digital data via sound waves above the range of human hearing. The ultrasonic modem could then be used to create a wireless data link between computers, phones or other devices as an alternative to RF technology, using only built-in speakers and microphones to send and receive.

The approach consisted of a research phase involving exploration of modulation techniques and solidifying the signal processing algorithms, while modeling the system in MATLAB. Binary phase shift keying (BPSK) was chosen as the system’s modulation scheme, with a carrier frequency of 20 kHz. BPSK is a simple and easy to debug approach for developing the system and 20 kHz is above the hearing range for humans, but is still low enough to use standard audio hardware.

Several obstacles were overcome during the project including low power output of standard speakers at the required frequency range and locking on to the carrier frequency in the receiver.

The final product performs short range data transfers.

System Overview

Proof of Concept
In order to understand BPSK modulation and demodulation, the first step was running simulations in MATLAB of transmitting and receiving data. During the simulation stage, the team researched filters, carrier tracking methods, and symbol timing recovery methods. MATLAB provided a good resource for testing and visualizing each stage of processing while providing a template for programming in C for the devices.

To transmit data, the binary data is first encoded by changing the 0’s to -1’s. Each bit of the data is then inserted into a carrier, the carrier frequency is 20kHz. To receive the data, the signal is bandpass filtered to diminish ambient noise, then run through a phase-locked loop (PLL) and down-converted to baseband. The received data is pulse shaped again through a matched square root raised cosine filter and then down sampled to the data rate for decision making and decoding.

Theory of Operation
There are many ways to embed digital information in a signal for transmission. The modulation technique that is used in this project embeds the information into the phase of the carrier signal. Binary phase shift keying (BPSK) uses a constant frequency cosine figure in order to eliminate high frequency harmonics of the signal. This results in a square wave with more gradual transitions. This signal then gets upconverted to the carrier frequency.

The phase-locked loop (PLL) is vital to receiver design. The figure below demonstrates the locking of a PLL. The incoming signal is offset by 50Hz from the initial PLL frequency. The figure shows the PLL starting with an error and tracking the carrier signal until error is reduced to zero. The PLL starts with a best guess of the carrier frequency and then uses error signal generated by the phase error to adjust the local oscillator.

The loop filter is fundamental to PLL performance and controls the settling time and overshoot of the phase estimation. It is a first-order feedback system, making the PLL overall a second-order system.

Final Design

The final design was based on the research and simulation done in MATLAB. This receiver configuration provided the best performance for the system in order to reliably lock on to the carrier and extract the data.

Only the receiver block diagram is shown in detail because it was the most complex part of the project and the transmitter was not too different from the overview representation.

Conclusion and Future Work
Over the past 5 months the team has developed a multiplatform system to transmit and receive data using an acoustic data link at 20 kHz. This can be a useful tool for short range data transfer between computers, smartphones, and other compact devices. One of the primary obstacles was poor performance of standard speakers at high frequencies, so experimentation with specialized transducers and faster ADC’s and DAC’s could prove very interesting. Future work could include implementing more complex modulation schemes for higher bitrates with greater bandwidth efficiency as well as incorporating adaptive equalization for better performance with noise.

Acknowledgments
- Sponsors: Raytheon/Applied Signal Technology and Michael Ready
- Faculty advisor: Patrick Mantey
- Thank you to everyone who helped us along the way
Introduction
This objective of this project was to test the interrupt latency of a relatively new development board, featuring Xilinx’s Zynq-7000 Programmable System on a Chip, called the ZedBoard. We measured the interrupt latency of the board configured as a stand alone system, with the Zynq-7000 as an embedded processor. As well as the interrupt latency of the board configured for running Real-Time and Standard Linux operating systems. For the Linux software interrupt testing, we used Cyclic-test, a Real-Time Linux test program. For the Linux hardware interrupt testing, we created a custom test bench using an interrupt signal generated from the onboard AXI timer, and a custom Linux driver that used the AXI timer data to measure the interrupt latency.

Initial Ideas
- Test interrupt latency using the A9 processor’s built in timer.
- Test interrupt latency using the onboard AXI timer.

Linux
- Test Linux hardware interrupt latency using the onboard AXI timer.
- Test Linux software interrupt latency using Cyclic-test.

Theory of Operation
The interrupt latency is the time when the interrupt signal is seen by the system to when the interrupt signal is answered. We obtain the interrupt latency by using the AXI timer, which creates an interrupt signal when it reaches terminal count. The AXI timer then rolls back to its reset value and continues to increment. The time after the AXI timer rolls over to when the interrupt handler answers the interrupt and reads from the AXI timer is the interrupt latency.

Methods

Conclusions
- Real-Time Linux software increases interrupt latency, but lowers the interrupt latency variance.
- Standard Linux software has lower interrupt latency, but at the cost of increased latency variance.
- Real-Time Linux hardware interrupt latency generates a consistent bell-curve, but at an increased average latency compared to Standard Linux.
- Bare Metal interrupt latency is constant after the caches warm up.
- Bare Metal interrupt latency shows cache warming behavior even when caches are disabled.

Results
Bare Metal

Acknowledgements
Mike Matera - Xilinx Sponsor
David Munday - Senior Project Mentor
Ethan Papp - Senior Project Assistant
Yaskawa Select: A Product Selection Application for the iPad

Dominic Amsden, Sylvie Boenke-Bowden, Kevin Perkins, Nelson Pollard

Abstract

Yaskawa is one of the largest makers of AC and DC servomotors and control systems in the world. The specification of the appropriate servomotor or control system to meet Yaskawa's customers' needs involves access to a large product list with many dependencies among the subassemblies. When Yaskawa's sales representatives are in the field, they need to have access to a laptop and the Internet in order to configure system part numbers to the customer's specifications. A fast, simple, offline method for configuring system part numbers in the field was required.

Using Scrum as our development process framework and weekly virtual meetings with management, engineering, and sales representatives from Yaskawa, we have designed and implemented YaskawaSelect as a standalone mobile application for the iPad that we believe addresses these challenges.

Required Features

- Search motors by motor family
- Select from available motor characteristics including wattage, voltage, and encoder options
- Select motor accessories - cables, amplifiers
- Generate a report based on selections
- Advanced search by performance characteristics
- Search by max torque, intermittent speed, load inertia
- Calculate most desirable motors based on input
- Dynamically edit/add selections
- Stretch goal: Sizing & graphs based on motor data

User Interface

1. Choose Search Method
2. Select Motor Family
3. Select by Performance
4. Touchscreen Motion Profile Graphs
5. Enter Parameters: Torque, Speed, Load Inertia
6. Select one or more motors to compare
7. Motor Query: Wattage, Cables, Amplifiers
8. Results List Bill of Materials

Motion Profile

A motion profile is a graphical representation of a customer's mechanical requirements, relating speed and torque over time. Our application will allow the customer to set up and manipulate these graphs via the iPad's touchscreen. The app will then translate the entered data into traditional motor query parameters.

Architecture

Workflow Diagram

Technologies Used

- Database Design
  - SQLite
- Python scripting language to load database
  - Parses through data and creates easier to process datasheet.
- Database API
  - Object oriented
- Mobile Application and Motor Search
  - Apple Xcode
  - User Interface
  - Storyboard in Xcode
  - Written in Objective-C
  - Core Plot from Google for graphing

Results

- Created App that queries motor & accessory database based on sophisticated search parameters.
- Designed User Interface for Salesperson to use with customer in the field without Internet connectivity.
- Offers a range of motor options & gives recommendations based on customer’s needs.
- What's next?
  - Motor graphs updating in real time.
  - Use iPad's touch screen for motion profile graphs.

Acknowledgements

Scott Carlberg
Product Marketing Manager

Jeffrey Pike
Manager, Motion Group Marketing

Edward Nicolson, Ph. D.
Senior Director, Development

Mark Wilder
Regional Motion Engineer

Michael Miller
Supervisor, Regional Motion Engineer

Jennifer Piane
Software Engineer

Faculty Advisor Dr. Linda Werner