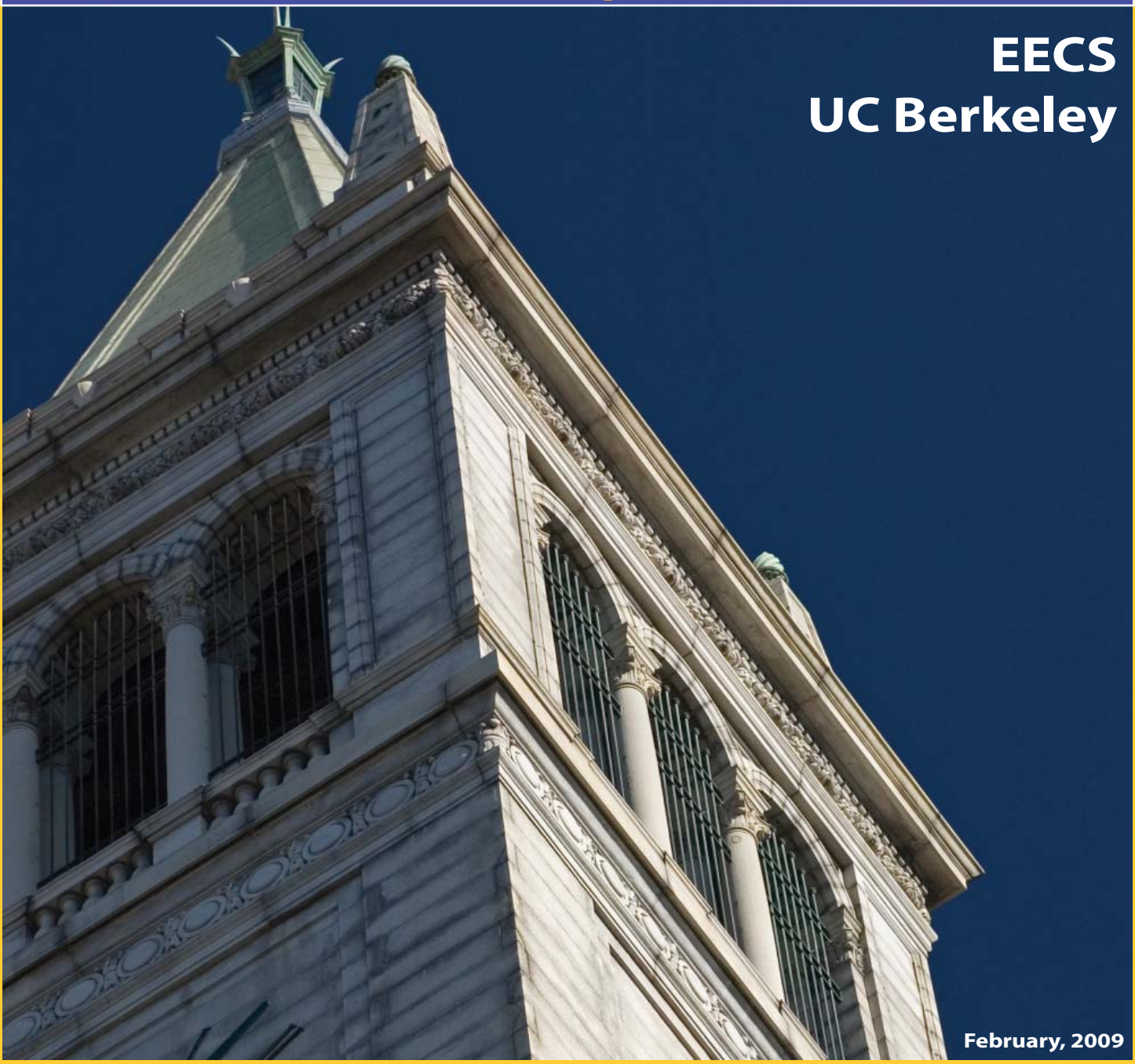
Chess

Center for Hybrid and Embedded Software Systems

— 2009 Prospectus —



EECS
UC Berkeley

February, 2009

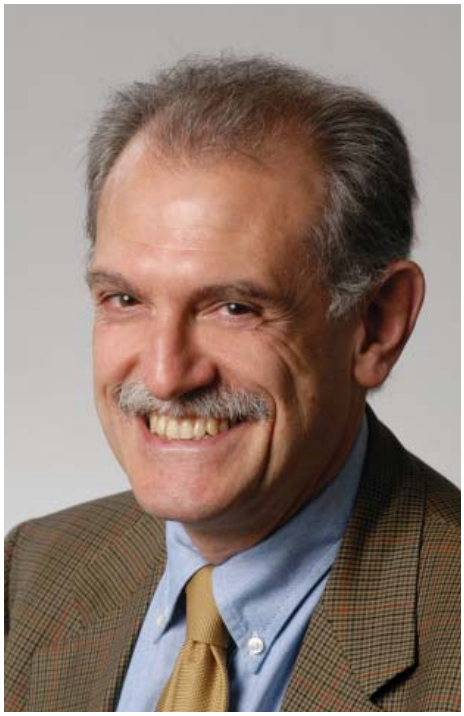
Letter from the Directors

CHESS was founded in 2002 to build foundational theories and practical tools for systems that combine computation, networking, and physical dynamics. The US **National Science Foundation (NSF)** provided the seed funding for the first five years under the Information Technology Research (ITR) program. CHESS is now thriving, with a rich portfolio of funded research projects and active collaborations with corporate partners. The current members include **Bosch**, **HSBC**, **Lockheed-Martin**, **National Instruments**, and **Toyota**, with additional projects funded by **Agilent** and **IBM**. NSF continues to fund projects in CHESS, as do the US **Air Force Office of Scientific Research**, the **Air Force Research Lab**, the **Army Research Laboratory**, and the State of California **Micro** Program. This report is an overview of these projects.

In the systems of interest in CHESS, embedded computers and networks monitor and control physical processes in feedback loops where physical processes affect computations and vice versa. For the last 30 years or so, computers have been increasingly embedded in stand-alone, self-contained products. We are poised, however, for a revolutionary transformation as these embedded computers become networked. The transformation is analogous to the enormous increment in the utility of personal computers with the advent of the web. Just as personal computers changed from word processors to global communications devices and information portals, embedded computers will change from small self-contained boxes to *cyber-physical systems* (CPS), which sense, monitor and control our intrinsically distributed human environment.

Edward A. Lee

S. Shankar Sastry



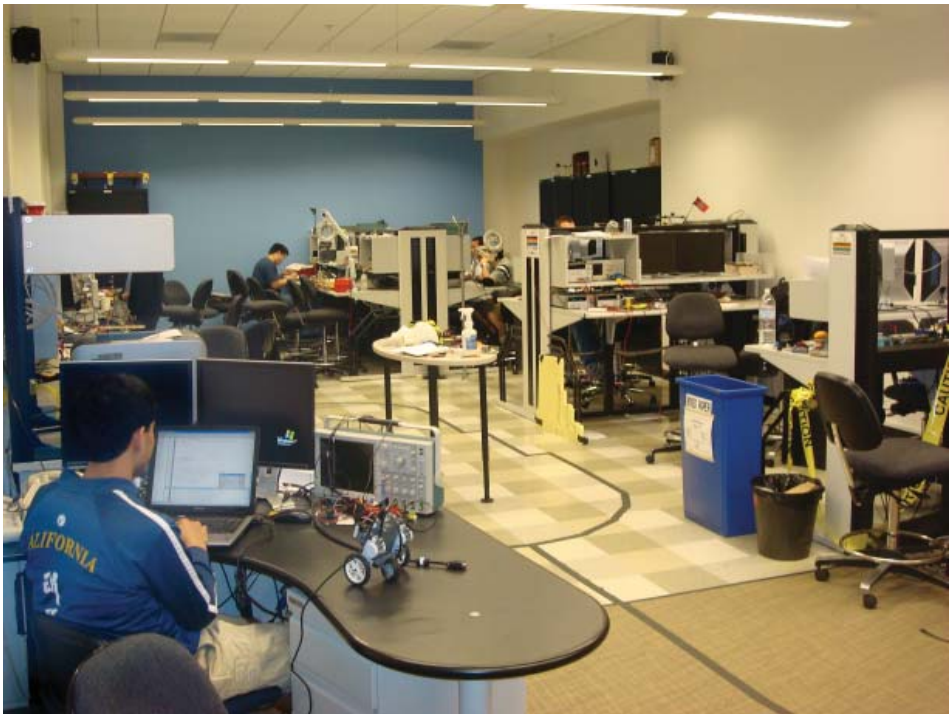
Alberto Sangiovanni-Vincentelli

Claire Tomlin

What is CHESS?

The Berkeley Center for Hybrid and Embedded Software Systems (CHESS) focuses on model-based and tool-supported design methodologies for real-time fault tolerant software on heterogeneous distributed platforms. We are bridging the gap between computer science and systems science by developing the foundations of a modern systems science that is simultaneously computational and physical. This represents a major departure from the current, separated structure of computer science (CS), computer engineering (CE), and electrical engineering (EE): it reintegrates information and physical sciences. The center includes corporate partnerships and conducts a number of funded research projects. It is directed by a board of directors, with day-to-day operations run by an executive director.

Christopher Brooks
CHESS Executive Director



Newly opened National Instruments Embedded Systems Design Laboratory, being used in this picture for an undergraduate mechatronics class

Bridging Research & Education

Embedded Systems Curriculum

CHESS leads the way towards a comprehensive undergraduate and graduate curriculum in embedded systems. Supporting this effort, CHESS partner **National Instruments** and its CEO **Dr. James Truchard** funded a new instructional lab (see pictures on this page) dedicated to teaching embedded systems. Complementing this infrastructure effort, we have developed courses that now form the backbone of the embedded systems curriculum at Berkeley.

Dr. James Truchard, CEO of National Instruments, with CHESS director Lee, then Chair of EECS, at the opening of the National Instruments Embedded Systems Design Laboratory in March 2008.

Introduction to Embedded Systems EECS 149

CHESS directors **Lee** and **Tomlin**, and participating faculty **Sanjit Seshia** developed a new undergraduate course in embedded systems. This was first offered in Spring 2008. It introduces students to the design and analysis of computational systems that interact with physical processes. A major theme of this course is on the interplay of practical design with models of systems, including both software components and physical dynamics. A major emphasis is on building high confidence systems with real-time and concurrent behaviors. Through projects and lab exercises, students are introduced to applications in robotics, consumer electronics, control systems, and instrumentation.

Embedded System Design: Models, Validation, and Synthesis EECS 249

CHESS director **Sangiovanni-Vincentelli** has led the development of our core graduate course in embedded systems. This class presents approaches to a new system science based on theories, methods and tools that were in part developed by CHESS. It emphasizes modeling, design, and analysis techniques where heterogeneity, concurrency, and multiple levels of abstraction play an important role. Correct-by-construction refinement techniques are introduced as a way of substantially reducing design time errors.

Concurrent Models of Computation EECS 290N

CHESS director **Lee** has led the development of an advanced graduate course focused on the models and techniques of embedded systems design and analysis. This course studies concurrent models of computation (MoCs), including how to use concurrent MoCs to design software systems, how to implement concurrent MoCs, how to analyze designs for boundedness, deadlock, and determinacy, formal semantics (fixed point semantics and metric-space models), and language design (type systems, higher-order components, and structured design). The MoCs covered include process networks (PN), threads, message passing, synchronous/reactive (SR), concurrent state machines (statecharts and ERG), dataflow (several variants), rendezvous (like CSP, CCS), time-triggered models (like Giotto), discrete-event (DE), and continuous-time (CT). The course studies heterogeneous models, including hybrid systems, and how these MoCs are used in languages such as StreamIt, LabVIEW, Lustre/SCADE, Esterel, Signal, and Simulink. Applications to embedded systems design, cyber-physical systems modeling, and parallel and distributed software are considered.

Entrance of the National Instruments Embedded Systems Design Laboratory.

High-Confidence Design of Adaptive, Distributed Embedded Control Systems (HCDDES)

The design of embedded systems is challenging for high-confidence aerospace systems, where technology leadership is the key to superiority. Next generation unoccupied air vehicles (UAVs) and space vehicles (SVs) are complex systems of systems involving multiple synchronous and asynchronous processes in an architecture distributed across several processors both within a single UAV (SV) and across groups of UAVs (SVs). Additionally, autonomy, fault tolerance, and resource optimization require that mixed-criticality functions are resident on the same processors in this architecture. Guaranteeing the provably correct behavior of the overall embedded system is extremely difficult, especially with respect to timing constraints and their relationship to safety of the physical systems, and performance specifications associated with mission-level objectives. Traditional software engineering methods cannot solve these problems because they do not address properties key to the interaction of physical processes with software, such as timing, mixed criticality functions, and concurrent processes.

This project is developing a comprehensive approach to the design of high-confidence complex embedded systems, that is, systems that are correct-by-construction, fault tolerant, secure, and degrade gracefully under fault conditions or information attack. Our approach integrates verification, validation, and test procedures throughout the complete design, development and maintenance cycle, from requirements capture to deployment and life cycle updates. This MURI project has the following research areas:

1. Hybrid and embedded systems theory.
2. Model-based software design/verification.
3. Composable Tool Architectures.
4. Testing and Experimental Validation.

The CHESS component of this project is focused on an experimental quadrotor aircraft called the Starmac, shown below, designed by CHESS director **Tomlin** and her research group.

AFOSR MURI

MURI is a Multidisciplinary University Research Initiative operated by the Air Force Office of Scientific Research (AFOSR). This MURI project is a collaboration between CHESS, Vanderbilt University, Carnegie Mellon, and Stanford.

Starmac team, left to right: Steven Waslander, Vijay Pradeep, Haomiao Huang, Michael Vitus and Gabe Hoffmann.

Jeremy Gillula is missing from the photo. All were Stanford graduate students or postdocs at the time.

Ptolemy II

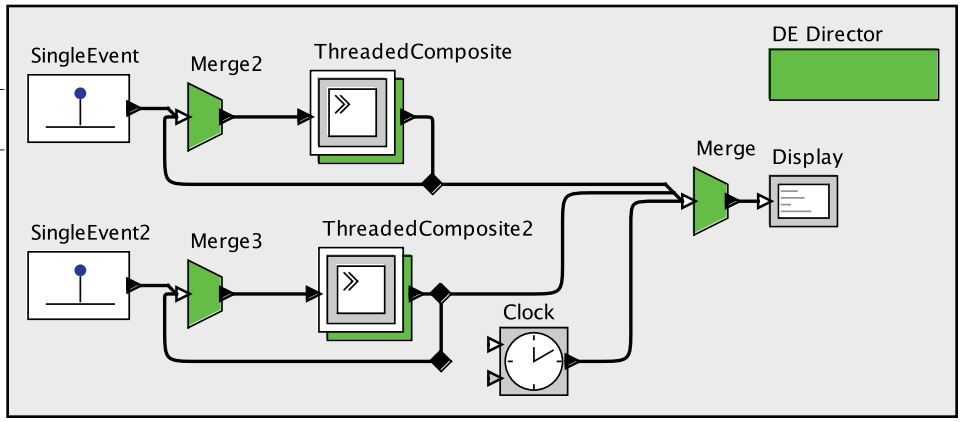
A Software Laboratory

The Ptolemy Project, led by CHESS director **Lee**, is developing and maintaining a software framework called Ptolemy II that serves as an extensible laboratory for experiments with design and analysis techniques. This effort is supported by the **Air Force Research Lab** under the suppo Modeling and Analysis Framework (EMAF) project and by CHESS affiliate **HSBC**. This project provides infrastructure for several other research projects in CHESS.

Ptolemy II supports design and assembly of concurrent components. The key underlying principle is the use of well-defined models of computation (MoCs) that govern the interaction between software components. A major problem area addressed is the use of heterogeneous mixtures of models of computation.

Current efforts include:

- Providing a flexible and retargetable code generation framework that can translate Ptolemy II models into standalone C or Java programs.
- Facilitating development, maintenance, and evolution of large models, leveraging concepts such as higher-order components and model transformation tools to get scalability.
- Effectively exploiting through self-configuration, multicore and networked computing infrastructure for high performance.

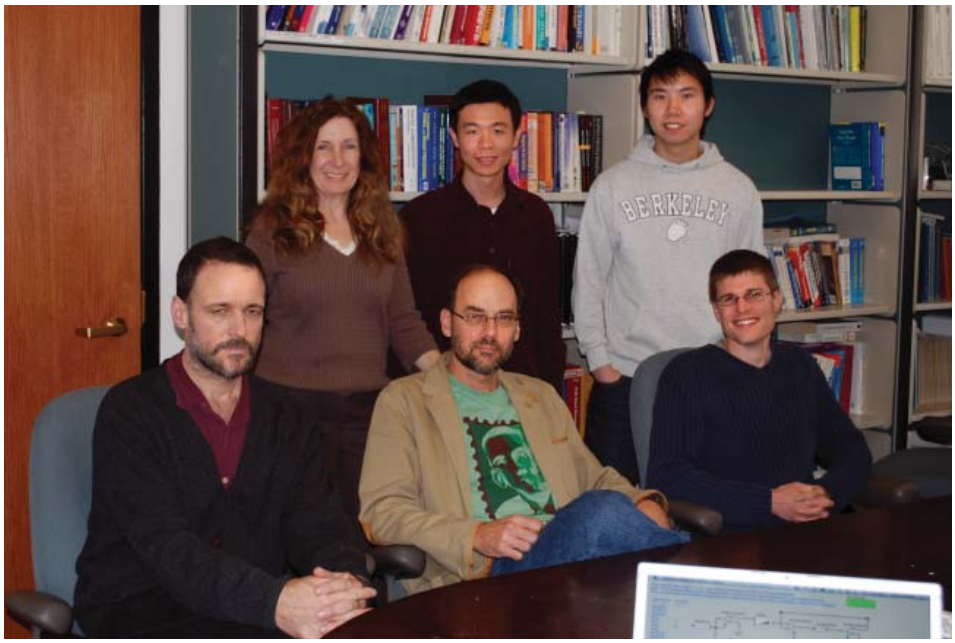


Ptolemy II model shown in Vergil, an editor that supports graphical editing of Ptolemy II models. This model shows the newly added ThreadedComposite higher-order actor. This actor supports concurrent and parallel execution under many models of computation. It enables designers to leverage multicore architectures with parallel execution and to integrate asynchronous and real-time I/O with Ptolemy II models.

- Incorporating legacy analysis tools and code as components within more complex workflows.
- Customizing configurations of Ptolemy II to provide specialized or domain-specific functionality.
- Tools and techniques for model-engineering, including model refactoring tools, model analysis, formal verification, and support for reusable design patterns.
- Experimentation with visual syntaxes and visual editors for models, with a focus on organizing and handling large data sets and constructing and visualizing large models.
- Integration of databases, web services, and cluster computing infrastructure into models.

Ptolemy II is open-source software with an open SVN repository and hundreds of contributors.

The core support and development pteam is shown below. From left to right, top to bottom, Mary Stewart, Thomas Huining Feng, Jackie Mankit Leung, Christopher Brooks, Edward A. Lee, and Bert Rodiers.



Hybrid Control Models and Computational Tools for Biological Regulatory Networks

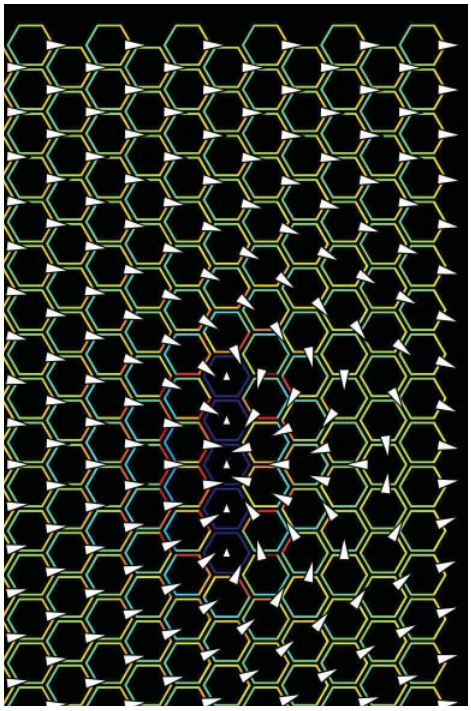
Systems theory for biology

With funding from NSF, CHESS Director **Tomlin** is leading a project focused on designing control theoretic mathematical models and associated computational tools for the systems involved in intra- and inter-cellular regulatory circuits in biological development.

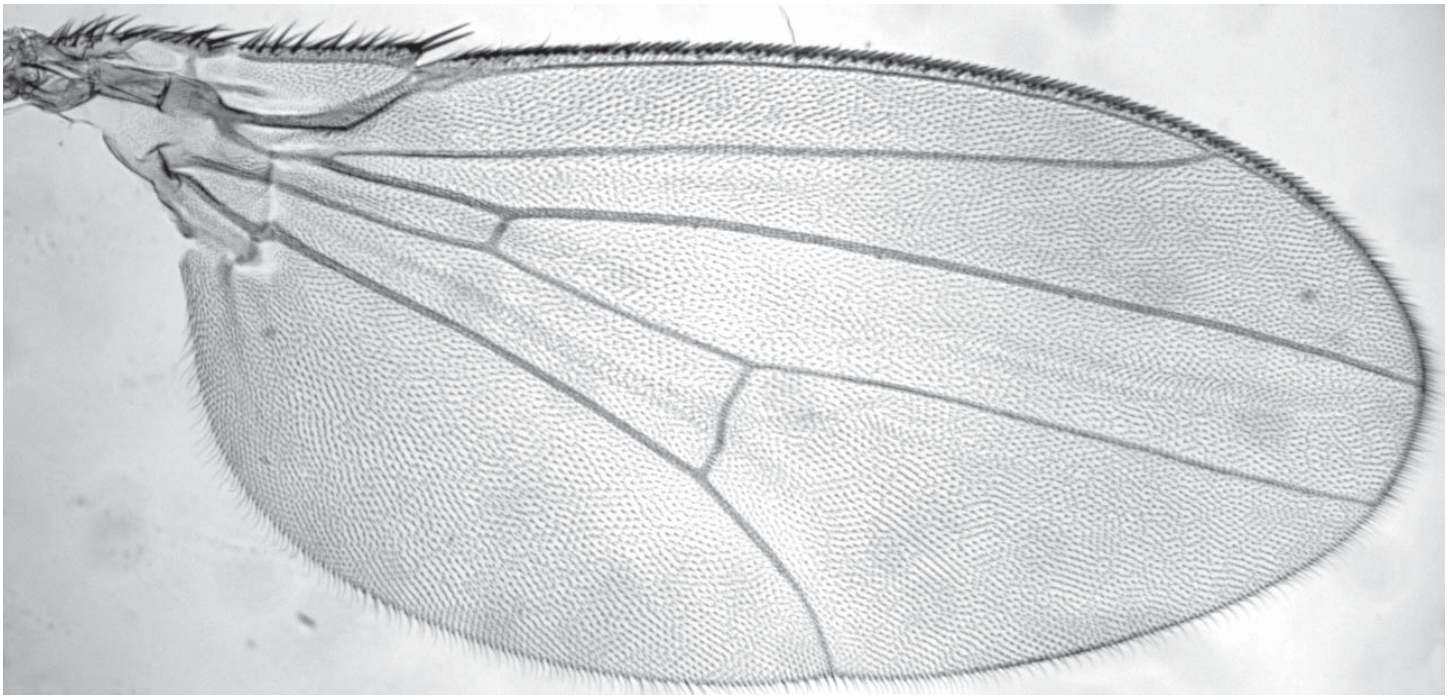
The majority of biological research today focuses on single point organisms or diseases. Even in the emerging field of synthetic biology, researchers concentrate on well-characterized components, which when connected in certain ways, yield point solutions. Often these solutions are extremely valuable, such as the discovery of a mechanism which controls segmentation in embryos, or the design of bacteria which cheaply manufacture useful medicines and fuels. Yet we believe that there is a strong need for general purpose models, analysis methods, and software tools to represent, computationally dissect, and understand the processes by which organisms develop. This will lead to a better understanding of genetic defects which cause disease, leading to effective personalized health care, and could help immensely in the engineering of new biological systems. A deeper understanding of the enviable robustness of biological development could also help in complex software systems design.

The focus of this project is on modeling and computation, in collaboration with several biologists. These collaborations are studying a range of developmental mechanisms, from early Drosophila development, to cell polarity determination in developing Drosophila and mice tissues, as well as in cell cultures, to the transition from low grade to high grade lymphoma in both mice and humans.

This project focuses on (i) development, analysis, control, and simulation of stochastic hybrid system models for properties needed for cell regulation; and (ii) design of efficient computational methods for large scale analysis of the corresponding biological cell networks. We use a hierarchy of interconnected models, and a set of computational algorithms, which describe the operation of cells and cell networks from a molecular level to an organismal level. Our approach combines novel new ideas from hybrid systems theory and developmental biology, as well as control theory, stochastic analysis, and numerical analysis.



The figure above shows a simulated representation of the Drosophila wing cells and associated hair growth, shown for an actual wing below (courtesy of Professor Jeffrey Axelrod, Stanford University School of Medicine). A strip of 5 cells in the center of the simulation are mutant for one of the core developmental proteins, and a disruption in polarity in the neighboring cells results. Axelrod and Tomlin are investigating cell polarity through mathematical models and computational tools.



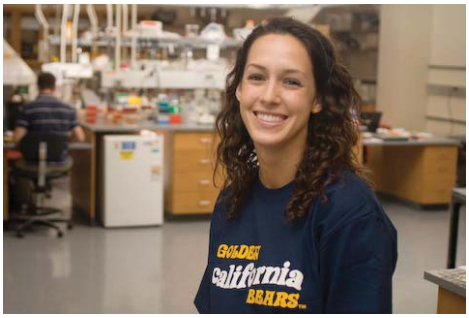
Platform-based Design of Synthetic Biology Tools

Assembly of biological parts is key to the future of synthetic biology. Biologists are faced with the challenge of organizing, sorting, and editing these parts from an ever-growing collection. This project is building on technology originally developed for design of electronic systems to assist biologists to do these jobs.

Clotho is a toolset that lays the foundation for a design flow using standard biological parts. This tool includes multiple views of a design, a number of unique interfaces for part management, a complete algorithm development manager, and a plug-in system based on the Java Plug-In Framework (JPF). CHESS researchers initially created Clotho as a project for the International Genetically Engineered Machine (iGEM) competition at MIT in 2008, where it received 1st place for software tools and was one of 16 teams (out of 84) to receive a gold medal overall. It continues as an active research project.

Clotho provides a platform-based design paradigm that consolidates diverse tools into one working, integrated toolbox. It is based on a core-and-hub system that manages multiple connections between tools. Examples of connections include sequence viewers/editors, parts database managers, assembly algorithm GUIs, and much more. Computer-savvy biologists can create new connections that will then be integrated into Clotho. These pieces can be kept locally in a lab or made public and shared with the community.

Key is how biological functionality can be implemented by the selection and assembly of standard parts. CHESS is focused on developing the correct abstractions for these systems and defining interconnections, interfaces, hierarchy, composition, and libraries. This includes developing efficient algorithms for the assembly of standardized biological parts and developing combinational logic representations for biological circuits. This work leverages VLSI CAD techniques to deal with structural duplication, carrier signal selection, and chemical diversity. Clotho has become an "associated project" within the SyntheticBiology Engineering Research Center (**SynBERC**) at Berkeley, and receives additional funding from **NSF** and **Genentech**.



Above: Anne Van Devender, who worked on this project via the SUPERB program (Summer Undergraduate Program in Engineering Research at Berkeley). Below: Clotho winning team, left to right: Doug Densmore (chancellor's postoctoral fellow and team lead), Matthew L. E. Johnson, and Nade Sritanyaratana.

Precision-Timed (PRET) Machines

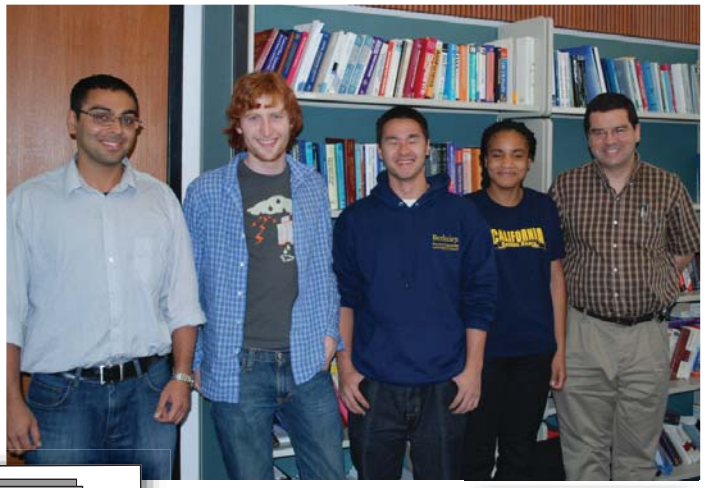
Embedded computer architectures with repeatable timing

Most abstractions in computing hide timing properties of software. For example, a C program does not directly provide any timing semantics. A programmer must step outside the programming abstraction to specify timing properties. The program itself describes only the transformation of data, and "correct" execution of the program has nothing to do with timing. As a result, computer architects and compiler and language designers use clever techniques to improve the average-case performance at the expense of predictable and repeatable timing. We find these techniques to be problematic for real-time embedded computing. The absence of timing in the computing abstraction layers results in unpredictable and non-repeatable behavior and brittle embedded systems.

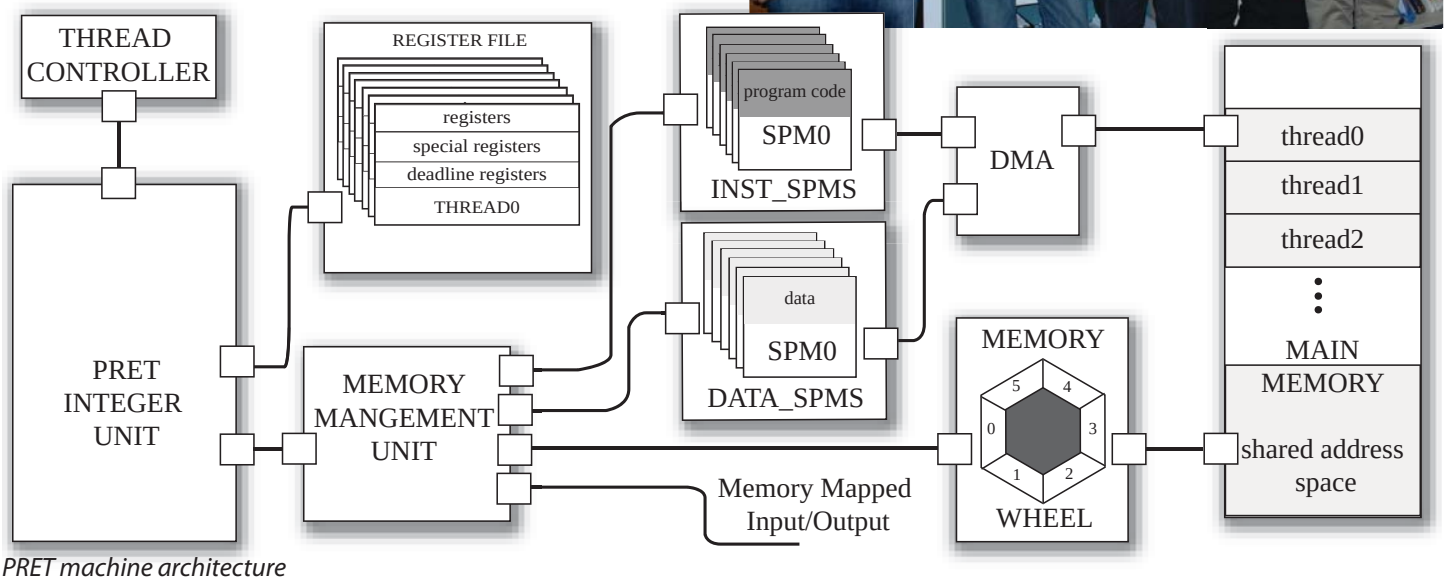
Our approach treats time as a first-class property of embedded computing. Our focus is on the hardware design of a real-time embedded processor that we call the precision timed (PRET) architecture. The PRET project, funded by **NSF**, is a collaborative effort lead by CHESS Director **Lee** and **Prof. Stephen Edwards** from Columbia University.

Our goal is to reintroduce timing predictability and repeatability by judiciously adopting architectural optimization techniques to deliver performance enhancements without sacrificing timing predictability and repeatability. Our approach includes extending the instruction-set architectures (ISA) with control over execution time. We believe that timing predictability and repeatability are not at odds with performance.

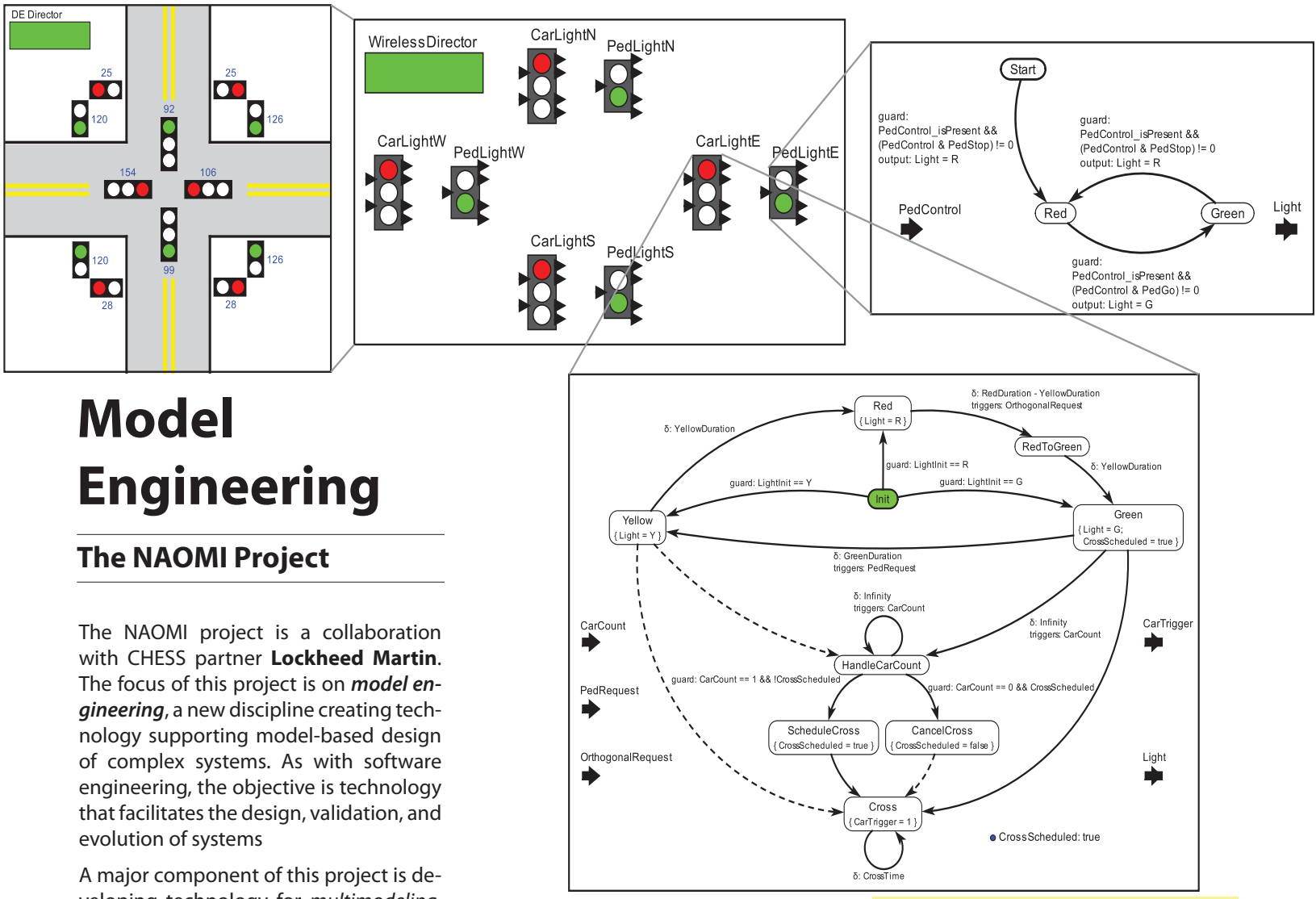
We have released a cycle-accurate model and simulator of a thread-interleaved PRET architecture based on the SPARC ISA. The architecture has multiple software-managed on-chip memories, a memory wheel to arbitrate access to main memory, and extensions to the ISA with timing instructions. Programs are written in C and compiled using GCC's SPARC compiler. We provide examples, including a video graphic driver example and RSA encryption/decryption algorithms to avoid timing-channel attacks. We plan to implement this as a soft core on FPGA. We will show that PRET architectures executing software components can be seamlessly integrated with what would traditionally have been purely hardware designs, thus greatly improving the expressiveness and usability of FPGA-based design flows. PRET provides a starting point for a revolution that will make timing predictability and repeatability central features of embedded processors.



The CHESS PRET team includes, from left to right, Hiren Patel, Ben Lickly, Isaac Liu, Shanna-Shaye Forbes, and Hugo Andrade (National Instruments)



PRET machine architecture



Model Engineering

The NAOMI Project

The NAOMI project is a collaboration with CHESS partner **Lockheed Martin**. The focus of this project is on ***model engineering***, a new discipline creating technology supporting model-based design of complex systems. As with software engineering, the objective is technology that facilitates the design, validation, and evolution of systems

A major component of this project is developing technology for *multimodeling*, which is the simultaneous use of multiple models and modeling techniques. We have identified three forms,

Hierarchical multimodeling: hierarchical compositions of distinct modeling styles, combined to take advantage of the unique capabilities and expressiveness of each style.

Multi-view modeling: distinct and separate models of the same system are constructed to model different aspects of the system.

Meta modeling: use of models together with models of the modeling language.

Specifically, this project is developing technology to facilitate identification of design problems well before system integration. The approach is to systematize multimodeling by defining mechanisms for exchanging modeling data. This enables the use of domain specific modeling languages (DSMLs) in combination.

Specific problems being addressed by this project include:

- Technology for composition of design patterns.
- Technology for maintaining relationships between simulation models and abstractions suitable for formal verification.
- Support for parameterization of models with models. Two versions of these are **aspect-oriented multimodeling**, where models are woven together to generate new models, and **higher-order components**, where components in a model are parameterized by other models.
- **Model ontologies**, which annotate data shared across models with semantic information and check for compatible uses of the data.
- Development and application of the **event relationship graph (ERG)** model of computation for representing control logic and controlling model transformation workflows.

*The model above is a test case constructed by CHESS graduate student **Thomas Huining Feng**. It shows a hierarchical multimodel that can publish data and subscribe to data from other multi-view models via the NAOMI software framework. This model shows a form of hierarchical multimodeling where the decision logic of a simple traffic light system is represented heterogeneously using both finite state machines (FSMs) and event relationship graphs (ERGs). The decision logic models are composed within the wireless domain of Ptolemy II, which is an extension of the discrete-event (DE) domain that models wireless communication. This wireless deployment model is embedded within a discrete event model that provides test data and animates a simulation. Using the NAOMI framework, this model can import or export parameters from or to entirely different modeling tools.*

Scalable Composition of Subsystems

Software Producibility

This project, which started in February 2007, is funded by the **Army Research Office** in connection with the **OSD Software-Intensive Systems Producibility Initiative**. The mission is to research scalable techniques in software engineering based upon the concepts inherent in model-based composition. The overarching goal is to show that these techniques will result in predictable and understandable behaviors in Systems-of-Systems (SoS) environments. The focus is on interaction between components (rather than the conventional focus on transformation of data), and on composition, which in this domain needs to be intrinsically concurrent (rather than the conventional thread-based applique of concurrency on imperative models).

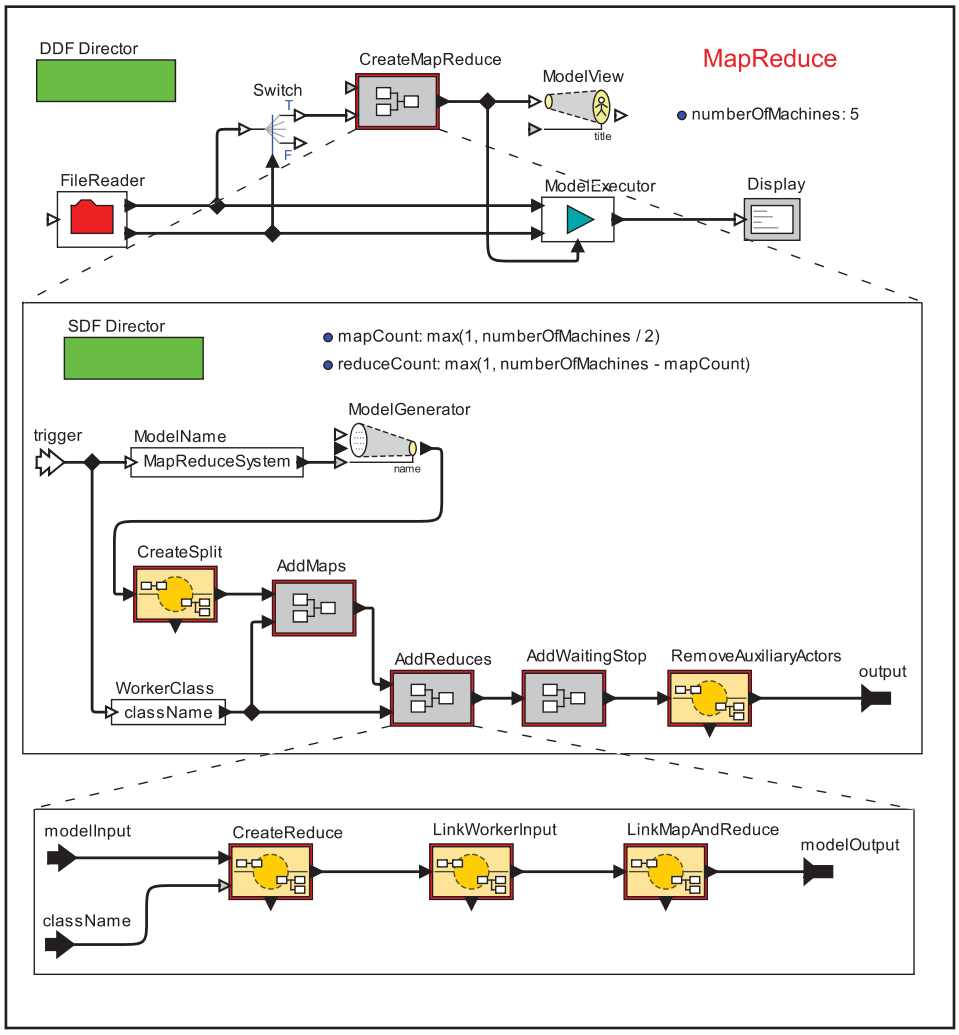
Composition is based on (1) integration technologies for legacy and custom subsystems that provide an understanding of the interaction of subsystems; (2) scalable composition mechanisms for system-of-systems architectures; (3) interface formalisms through which compatibility of subsystems can be checked and properties of compositions can be determined from properties of the subsystems; (4) Ontology models for the organization of components together with a semantic type system for the data on which they operate; and (5) Hybrid models for designing and analyzing the dynamics of subsystem interactions with their physical environment. We assume that subsystems have arbitrary granularity, and can range from simple software components to entire legacy systems appropriately wrapped.

We focus on **actor-oriented (AO) models**, which complement object-oriented (OO) component architectures with intrinsically concurrent subsystem coordination mechanisms. In OO, components interact principally through transfer of control (method calls). At coarse granularity, distributed objects interact through middleware that extends OO principles with proxies that mask the distributed nature of the system. At still coarser granularity, middleware services support a variety of strategies for managing distributed data and concurrent interactions, including message passing, web services, transparent data replication, event marshalling, transaction services, and distributed file systems. AO mechanisms focus instead on concurrent semantics with disciplined, well-founded concurrency and communication mechanisms.

One major outcome of this project has been a **model transformation technology** (see box) that has applications to model optimization, scalable model construction, joint management of product families, design refactoring, and workflow automation. In the mechanisms that we have developed, models construct models. The same modeling languages are used to specify a scalable composition of components as are used to specify the component interactions.

Potential applications include large complex software systems, distributed systems, and scalable parallel programming.

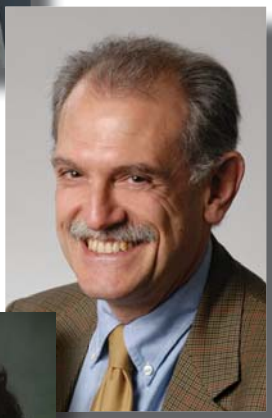
The figure to the right illustrates a model transformation workflow that constructs an arbitrarily sized computational model following the MapReduce pattern of Dean and Ghemawat. The top-level model is a dynamic dataflow (DDF) model that creates a MapReduce model and then displays and executes it. It creates the model by invoking the mid-level model, which is a synchronous dataflow (SDF) model that constructs the MapReduce model. This model uses model-transformation components that accept Ptolemy II models at their inputs and produce modified Ptolemy II models at their outputs. The modifications to the model are specified using a triple-graph grammar. This example and the model transformation infrastructure was constructed by CHESS researcher **Thomas Huining Feng**.



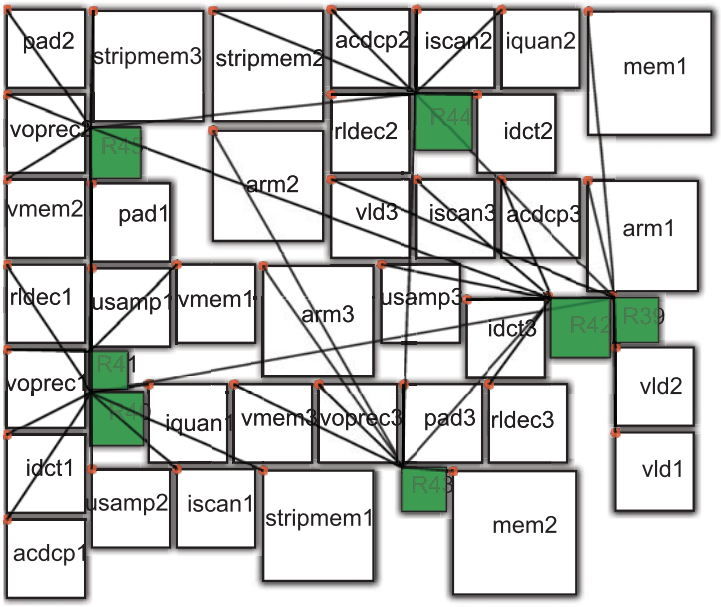
Communications Synthesis Infrastructure

The ***communication synthesis infrastructure*** (COSI) framework, is a public-domain design framework for design exploration and synthesis of interconnection networks. COSI is based on ***platform-based design*** and allows researchers and designers to contribute, combine, and compare optimization algorithms, communication protocols, partial designs, and models for interconnection design. Specifically, COSI enforces a clean separation among network specification, the library of building blocks that can be instanced and composed to derive the network implementation, the models of performance and cost associated with each of them, and the optimization algorithms that are used to explore the design space. Adopting this methodology allows comparing different interconnection topologies and different building blocks, thus smoothing out preconceived ideas about efficiency of particular interconnection schemes. The design paradigm advocated by COSI is implemented in the form of a software library that defines the data structures and their organization.

The general methodology for the design space exploration of networks is based on theoretical results. The formulation of the design problem can be shown to be independent of the particular application domain. To demonstrate the resulting flexibility and generality, the release of COSI includes two design flows for on-chip communications and building automation networks.



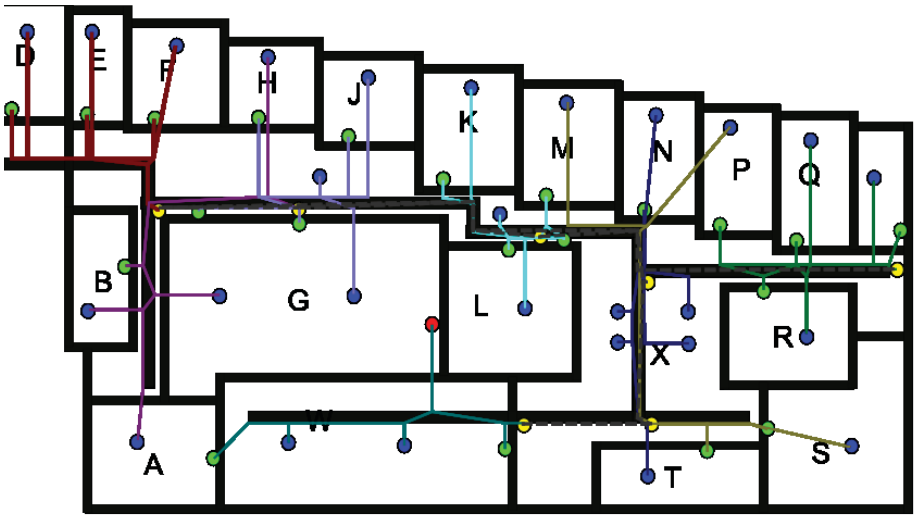
COSI researchers, clockwise from the top: Alessandro Pinto (UTRC), CHESS director Alberto Sangiovanni-Vincentelli, and Luca Carloni (Columbia).



The on-chip communication design flow (top right) allows reading the specification of a System-on-Chip in terms of the IP cores (including their geometrical properties), the communication requirements, and the models of the on-chip communication components. Several synthesis algorithms are provided to generate an optimal network with different properties. Code generators are also provided to produce numerical results as well as SystemC executable simulations. The figure shows the topology of an optimized interconnection network obtained by COSI starting from the communication requirements among cores (represented by white squares). The green squares represent on-chip routers.

The building automation network design flow allows reading the specification of the communication design problem in a graphical format (including the floorplan of a building, the wiring restrictions, the position of sensors and actuators and the communication requirements among sensors, actuators and controllers). Several synthesis algorithms are provided to synthesize an optimal network. Code generators are available to provide graphical and numerical results. The figure at the bottom shows the communication infrastructure of a sensor network for a hypothetical HVAC system of the Don Pederson Center in the Department of EECS, where most CHESS researchers reside. Green dots represent sensors, blue dots actuators and the red dot a centralized controller. The result obtained by COSI is a set of (possibly heterogeneous) sub-networks such as wired connections that use power lines and wireless connections based on the ZigBee protocol, indicated by different colors, connected by a higher level network.

For more details and software releases, please visit http://embedded.eecs.berkeley.edu/cosi/



Mechanical Systems Control

Professors **Karl Hedrick** and **Masayoshi Tomizuka** and their students in the **Mechanical Systems Control (MSC) Laboratory** of the Berkeley Mechanical Engineering Department are collaborating with CHESS on a project funded by CHESS partner **Toyota** to improve modeling and embedded software design for automobile engines and transmissions. The overall objective is to develop engineering methods for creating extremely reliable embedded real-time software with well-understood behavior within the dynamics of the overall automotive system. CHESS director **Lee** and affiliated faculty member **Koushik Sen** provide hybrid systems modeling, embedded software code generation, and software verification technologies.

In the approach in this project, suitable models of computation are chosen that can better represent the systems at hand, that is, the vehicle subsystems. The plant dynamics are modeled using hybrid systems. In this way, the combination of discrete events, such as gear shifting, and continuous dynamics, such as the catalytic converter temperature, can be handled in a single framework. Our work also includes experimental validation using a Toyota engine (see box).

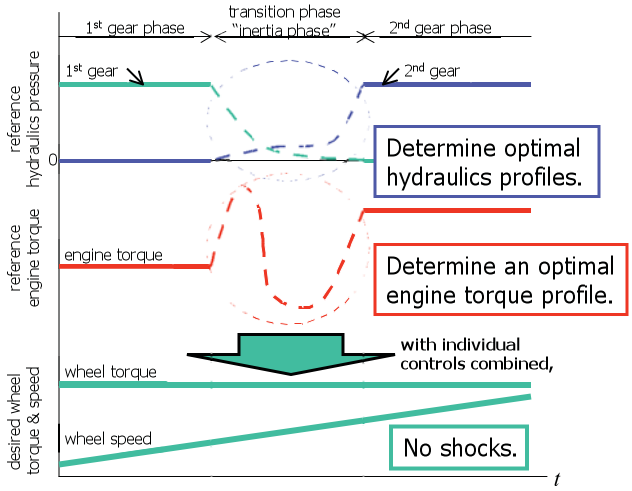
Cold Start

In the cold start period of a gasoline automotive engine, the first minutes of operation, a large percentage of the cumulative hydrocarbon (HC) emissions are produced. This project has developed a hybrid systems model of the engine and of the catalyst that reflects this behavior. This model is designed to be flexibly adapted to different engines and to be coupled with models of control systems for the engine, thus supporting model-based design. Joint engine-controller models enable investigation of strategies that to reduce the engine hydrocarbon (HC) cold start emissions. A second objective is to reduce the cost of design, development and implementation of the control systems software.

Advanced Transmissions

In this effort, CHESS researchers are developing a model-based approach to automotive powertrain control related to the automatic transmission and the engine. The objective is control algorithms that achieve comfortable, powerful, yet economical driving, all of which are primal criteria and concerns for determining what is really "good" automobile driving. This project investigates the use of new sensors to measure the engine torque or the use of torque estimation techniques based on cylinder pressure sensors, and new models of the mechanical systems. To advance disturbance observer techniques that are used to achieve robustness and performance, the project has recently proposed an application of iterative learning control to further improve the performance of disturbance observers. The proposed scheme detects dynamical model discrepancies between an actual engine and its nominal model, and compensates for them to realize nominal plant dynamics.

The Toyota test cell facility used in this project consists of a Toyota Camry engine (2.4 L, 4-cylinder, DOHC 16 valve) with variable valve timing (VVT) and intake air valve control (IAVC) features. It can produce up to 117 kW(157hp) @ 5,600rpm and support a peak torque of 220 Nm(162lb.-ft.) @ 4,000rpm. The engine is coupled to an eddy current dynamometer through a clutch and a 5-speed manual transmission. The dynamometer can absorb up to 225 kW and also drive the engine to a peak torque of 156 Nm. A Digalog dyno-controller is used to control the speed and the torque of the dynamometer. The engine can be cooled quickly by means of a cooling pump separated from the main drive belt.



Smooth gear shifting through intelligent collaboration by the engine and the automatic transmission.

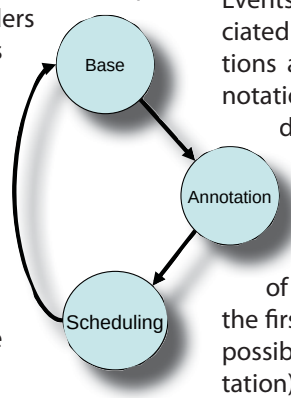
Metro II: A Next-Generation Design Framework for Platform-based Design

The design of embedded systems is becoming more difficult as design complexity increases, time-to-market pressures continue, and development teams with diverse backgrounds are assembled. The platform-based design methodology (PBD), developed under the direction of CHESS director **Sangiovanni-Vincetelli**, is a technique to combat these challenges. This methodology advocates the separation of concerns between an architectural platform – a collection of architectural primitives configured to provide a set of services – and the functionality – a description of what the design does defined in terms of the same services.

By taking these two portions of a design through a set of clearly defined abstraction/refinement steps which culminate in mapping, correct-by-construction design as well as structured design space exploration are enabled. To support the design and analysis of heterogeneous systems CHESS researchers, in collaboration with the **Gigascale Systems Research Center (GSRC),** developed the **METROPOLIS** Design Framework. It is based on the ideas of PBD and orthogonalization of concerns, specifically communication-computation, function-architecture, and behavior-performance. It features a flexible and formal semantics that supports a wide variety of models of computation.

Based on experience with **METROPOLIS**, CHESS graduate students **Abhijit Davare, Douglas Densmore, Trevor Meyerowitz, Alessandro Pinto, Guang Yang, Haibo Zeng**, and **Qi Zhu** have been developing a second generation framework called **METRO II**. Key objectives include:

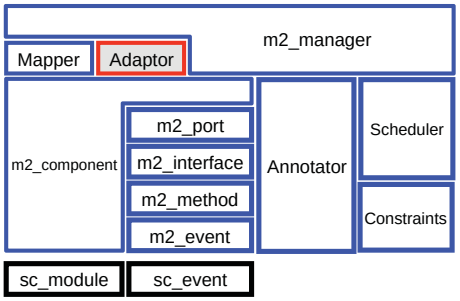
1. The ability to import pre-designed IP. IP providers develop their models using languages and tools that are domain specific. Requiring a singular form of design entry in a system-level environment results in significant effort to translate the original specification into the new language while making sure that semantics are preserved. If different designs can have different semantics, heterogeneity has to be supported by the new environment.



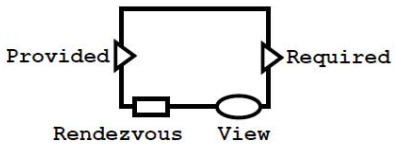
2. The ability to separate cost from behavior when carrying out design. In a system-level framework that supports multiple abstraction levels, many implementations of the same basic functionality will have the same behavioral representation at higher levels of abstraction. For instance, different processors will be abstracted into the same programmable component. What distinguishes them is the performance vs. cost tradeoff. Moreover, not all metrics are optimized at the same time. It should be possible to introduce performance metrics during the design process from specification to implementation.

3. The ability to explore the design space in a structured manner. This requirement is divided into two main parts: facilitating correct-by-construction abstraction/refinement and efficiently relating the functional and architectural portions of the design together. The first part is crucial to guarantee that the points explored in the design space are legal.

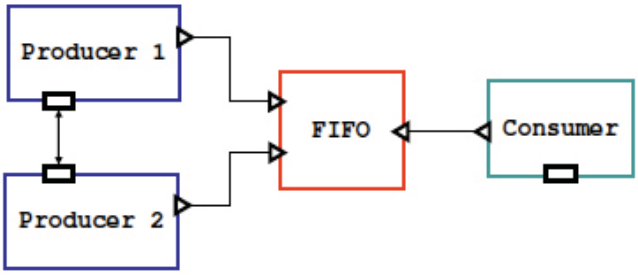
The key concept underlying METRO II is an event. An event denotes an action taken by a process. Events may be associated with annotations and state. Annotations correspond to quantities in the design, such as time or power. State includes variables that are in the scope of an event. Based on the treatment of events, the design is partitioned into three phases of execution, shown to the left. In the first phase (Base), processes propose possible events; the second phase (Annotation) associates tags with the proposed events; and the third phase (Scheduling) allows a subset of the proposed events to execute.



Metro II architecture, above. The implementation platform, sc_module and sc_event, is SystemC 2.2. The Metro II core is the set of components above that.

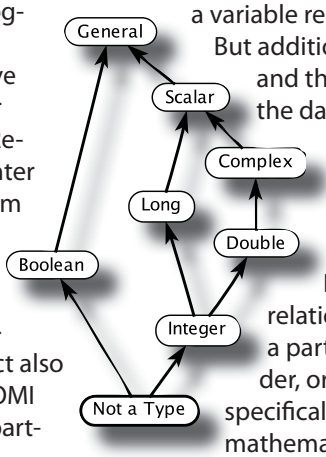


Metro II components can contain rendezvous ports (used to synchronize with other components), required ports (to indicate methods that must be provided by other components), provided ports (to provide methods to other components), and view ports (to expose internal events to the outside world). Below is a simple producer-consumer example built with such components, where the two producers coordinate the production through their rendezvous ports.



Ptolemy Hierarchical Orthogonal Multi-Attribute Solver (PtHOMAS)

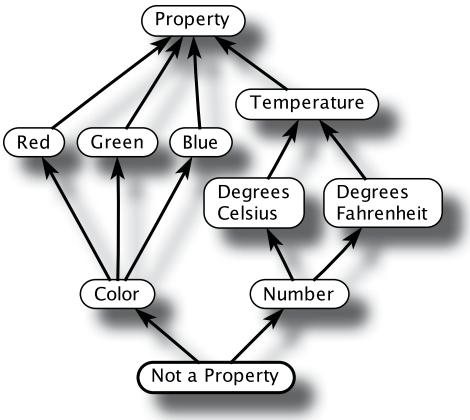
Ptolemy Hierarchical Orthogonal Multi-Attribute Solver (PtHOMAS) is a collaborative project with CHESS partner **Bosch** through the Bosch Research and Technology Center with additional funding from the **Army Research Office** in connection with the **OSD Software-Intensive Systems Producibility Initiative**. This project also has applications in the NAOMI collaboration with CHESS partner **Lockheed-Martin**.

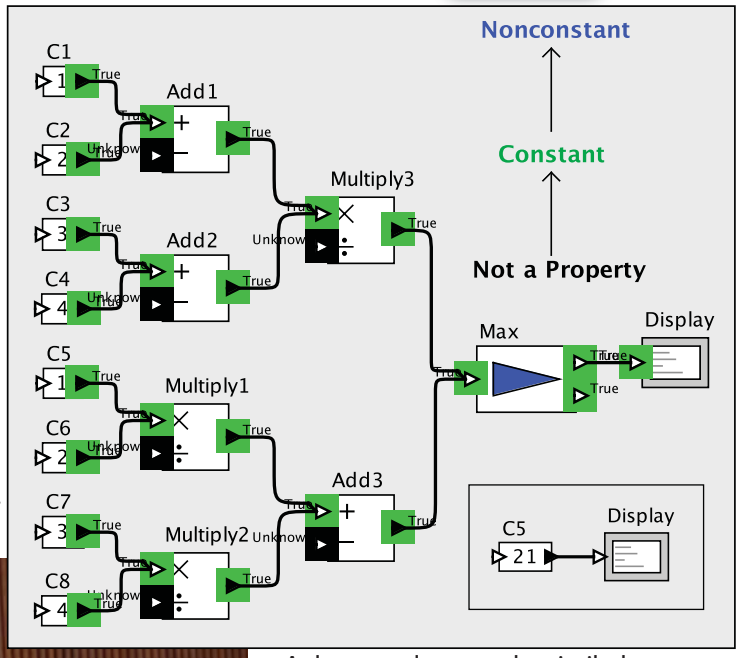


This project is focused on enhancing model-based design techniques with the ability to include in models semantic information about data (what the data means), to check consistency in the usage of data across models, and to optimize models based on inferences made about the meaning of the data.

The approach is to build on classical data type systems, which address this problem up to a point. It is easy to check, for example, whether two models agree that a variable represents an integer value. But additional meaning like the units and the semantic interpretation of the data is not captured in the type system. An organization of such meaning is referred to in computer science as a data ontology.

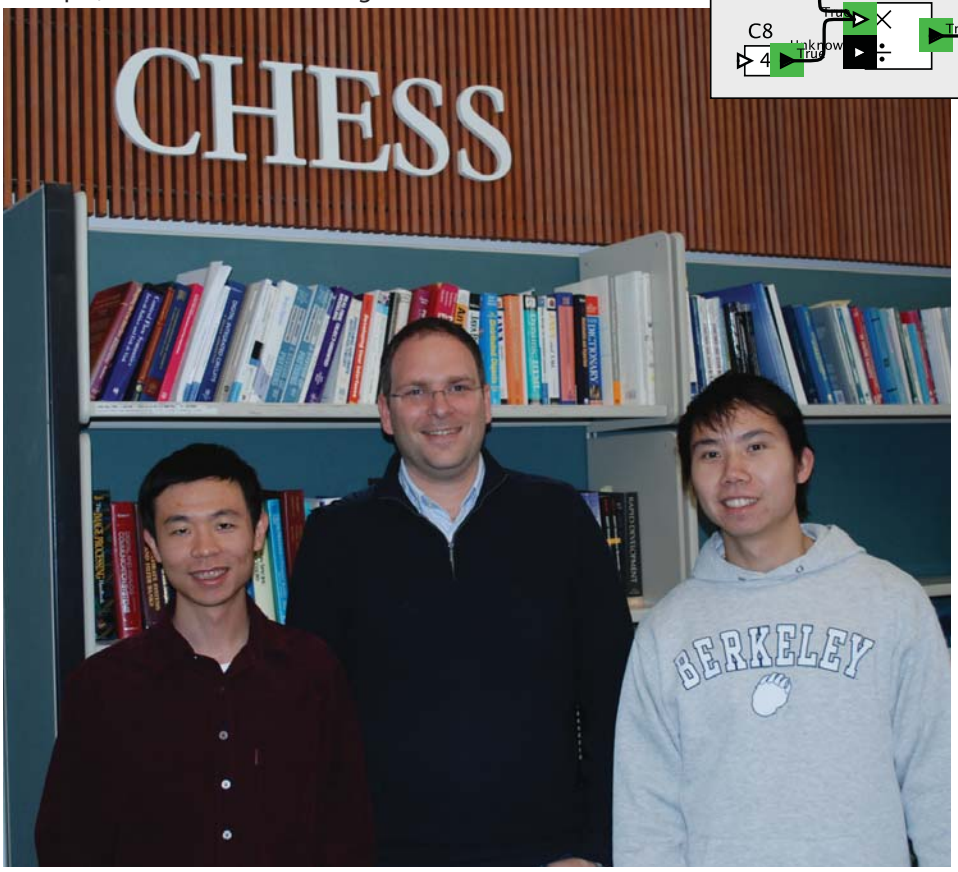
In classical type systems, the relationship between types form a partial order, or more specifically, a mathematical lattice, as shown above left for small subset of a typical type system. Type checking and type inference algorithms are based on efficient algorithms for solving systems of inequalities over such partial orders.





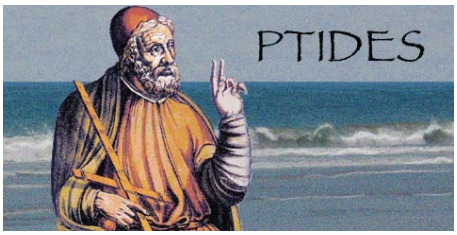
A data ontology can be similarly constructed as a lattice, as shown in the example on the upper right. In this example, each node in the ontology specifies an interpretation of a piece of data and the relations between nodes indicate "is a" relationships between these interpretations. In the PtHOMAS project, we associate such ontologies with models and apply consistency checks, inference algorithms, and model optimizations based on the results of the inference.

The above Ptolemy II model, for example, has a simple 3-element lattice associated with it, and based on inference results, can be automatically transformed to the much simpler model shown in the inset.



PtHOMAS researchers Thomas Huining Feng (CHESS graduate student), Thomas Mandl (Bosch), and Jackie Mankit Leung (CHESS graduate student).

Programming Temporally Integrated Distributed Embedded Systems (PTIDES)



This project, which has seed funding from **NSF**, **Agilent,** the **Army Research Office,** and **IBM** is developing programming models for real-time distributed systems, and studying algorithms that can statically check whether a given model is deployable over a network of nodes.

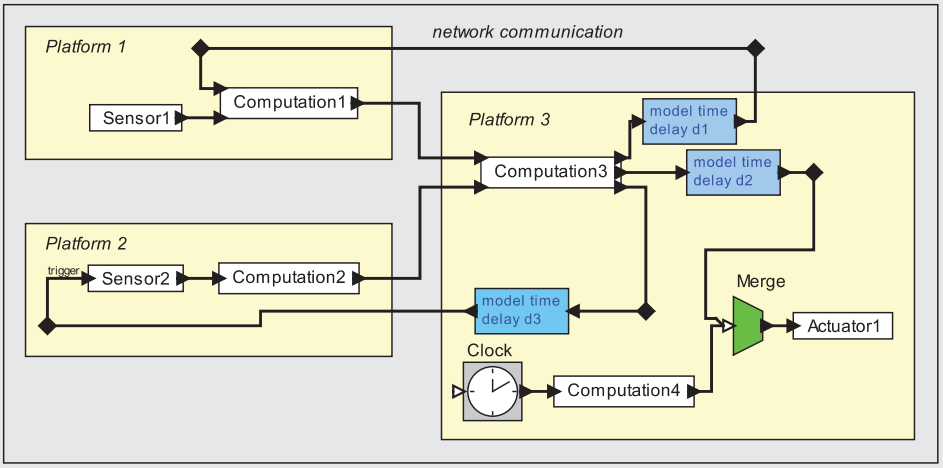
The introduction of network time protocols such as NTP (at a coarse granularity) and IEEE 1588 (at a fine granularity) and time-triggered networks like FlexRay and TTA results in a relatively consistent global notion of time in distributed systems. This project is addressing the question how such time synchronization changes the way that real-time distributed software is developed.

CHESS researchers, led by CHESS director **Lee,** and originally by graduate student **Yang Zhao**, developed a model called PTIDES (Programming Temporally Integrated Distributed Embedded Systems), which leverages time synchronization over distributed platforms. PTIDES uses discrete-event (DE) models as specifications for distributed real-time systems and extends DE models with the capability of mapping certain events to physical time. It uses *model time* to define execution semantics, getting determinate concurrent models, and adds constraints that bind certain model time events to physical time. It limits the relationship of model time to physical time to only those circumstances where this relationship is needed. The formal foundation is based on the concepts of *relevant dependency* and *relevant order* developed by CHESS graduate student **Ye (Rachel) Zhou**.

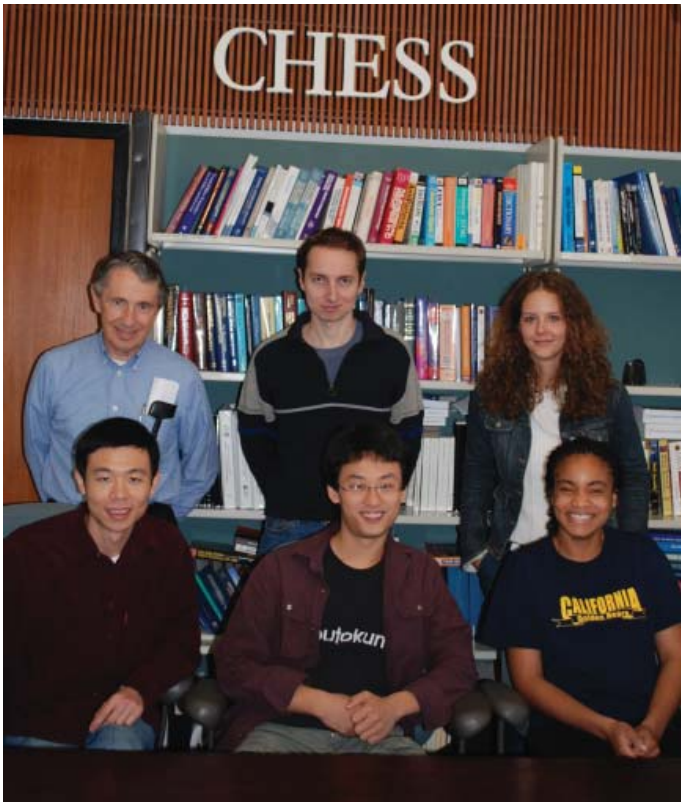
Implementations of IEEE 1588 have demonstrated time synchronization as precise as tens of nanoseconds over networks that stretch over hundreds of meters, more than adequate for many manufacturing and instrumentation systems. Such precise time synchronization enables coordinated actions over distances that are large enough that fundamental limits (the speed of light, for example) make it impossible to achieve the same coordination by conventional stimulus-response or client/server mechanisms.

We have begun implementing a distributed operating system based on Ptides that we call **PtidyOS**.

Potential applications of Ptides include manufacturing, instrumentation, surveillance, multi-vehicle control, avionics systems, automotive systems. robotics, and scientific experiments.



The figure above illustrates a typical Ptides scenario. We have three computers on a network (labeled "platforms" in the figure). Platform 1 collects data from a sensor and, based on feedback from platform 3, performs some calculations with that sensor data. It sends the result to platform 3 over a network, which combines the data with similar data from platform 2. After some calculation, it (possibly) issues a command to an actuator. That command is merged (deterministically, in time-stamp order) with locally generated commands to the actuator. The Ptides model is robust in that failures on platforms 1 and 2 cannot prevent platform 3 from processing the local commands.



Ptides researchers include, from left to right, top to bottom, John Eidson (Agilent Visiting Industrial Fellow), Slobodan Matic (CHESS postdoc), Patricia Derler (Visiting Student Researcher, University of Salzburg), and CHESS graduate students Thomas Huining Feng, Jia Zou, and Shanna-Shaye Forbes.

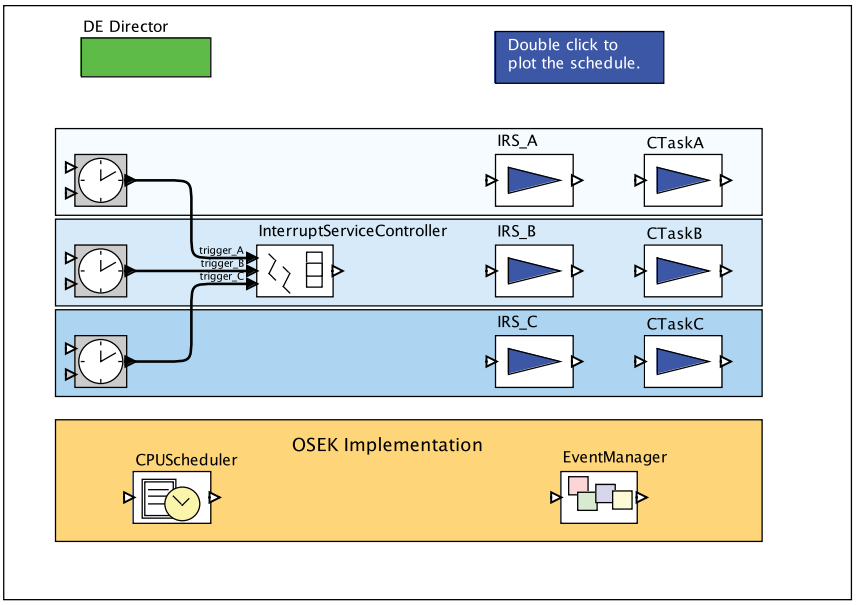
APES-LESS: Access Point Event Simulation of Legacy Embedded Software Systems

Modeling Legacy C Code

The implementation of a functional software model (with concurrency and timing properties) on a sequential execution platform may exhibit behaviors unaccounted for in the model, due to influences of platform-specific parameters. In particular, dataflow and timing properties of the model may be violated in the implementation as a result of the interplay of execution times and preemptive scheduling. In the embedded systems industry, the established way to validate properties of an implementation is extensive testing, which is done either by simulating the application mapped on a detailed model of the platform (usually using an instruction set simulator) or by using a hardware-in-the loop setup. Both processes are expensive and brittle. Small changes in the assumptions or the setup can result in big changes in the results.

This project, funded by CHESS partner **Toyota**, is developing an approach for fast simulation of software models mapped to platform abstractions. In particular, the project focuses on legacy automotive control software in C, running on an OSEK-compliant operating system. We refer to a line of source code containing an I/O access (with respect to the owner task) as an *access point*. In a run of the software, an "access point event" (APE) occurs whenever the code of an access point starts executing. The simulation framework being developed is based on three ingredients: (1) The ability to catch all APEs during an execution, (2) The ability to stop and then resume the execution of a task at any access point, and (3) The ability to determine the (platform-specific) execution time of the portion of code between any two consecutive access points of the same task. We achieve (1) by instrumenting the application code, i.e., inserting a callback to the simulation engine before every access point. Feature (2) is obtained by executing each application task in a separate (Java) thread, which can be paused and then resumed as necessary in the callback function. Objective (3) can be achieved by leveraging existent methods for estimation of execution times.

The simulation engine employs the discrete-event domain in Ptolemy II to process the APEs generated by the application tasks, with timestamps given by the execution times determined at (3) and reported to the engine as parameters of the instrumentation callbacks. Task management operations are provided by a partial OSEK implementation in Ptolemy II.



Above: Model of program with three interrupt service routines and three scheduled tasks. Below: APES-LESS researchers Patricia Derler (Visiting Student Researcher from the University of Salzburg) and Stefan Resmerita (CHESS Postdoc).

Foundations of Cyber-Physical Systems

Cyber-Physical Systems (CPS) are integrations of computation and physical processes. Embedded computers and networks monitor and control the physical processes, with feedback loops where physical processes affect computations and vice versa. The economic and societal potential of such systems is vastly greater than what has been realized, and major investments are being made worldwide to develop the technology. The technology builds on the older (but still very young) discipline of *embedded systems*, computers and software embedded in devices whose principle mission is not computation, such as cars, toys, medical devices, and scientific instruments. CPS integrates the dynamics of the physical processes with those of the software and networking, providing abstractions and modeling, design, and analysis techniques for the integrated whole.

As a discipline, CPS is an engineering discipline, focused on technology, with a strong foundation in mathematical abstractions. It shares many of these abstractions with computer science, but requires adapting them to embrace the dynamics of the physical world. Computer science, as rooted in the Turing-Church notion of computability, abstracts away core physical properties, such as the passage of time, that are required to include the dynamics of the physical world in the domain of discourse.

CHESS is pursuing foundational research in the abstractions and analytical techniques of CPS. Two such efforts are described here.

CHESS graduate students Yasemin Demir, Jeff Jensen, and Eleftherios Matsikoudis.

Timed Discrete-Event Systems

In the context of timed discrete-event systems, processes are allowed to realize functions that are not order-preserving with respect to the prefix ordering relation on the communicated sequences of values. This property renders naive applications of traditional domain-theoretic models inadequate for the semantic interpretation of such systems. Yet, interesting results have been obtained by imposing a fixed lower bound on the reaction time of the involved processes, effectively precluding Zeno behavior.

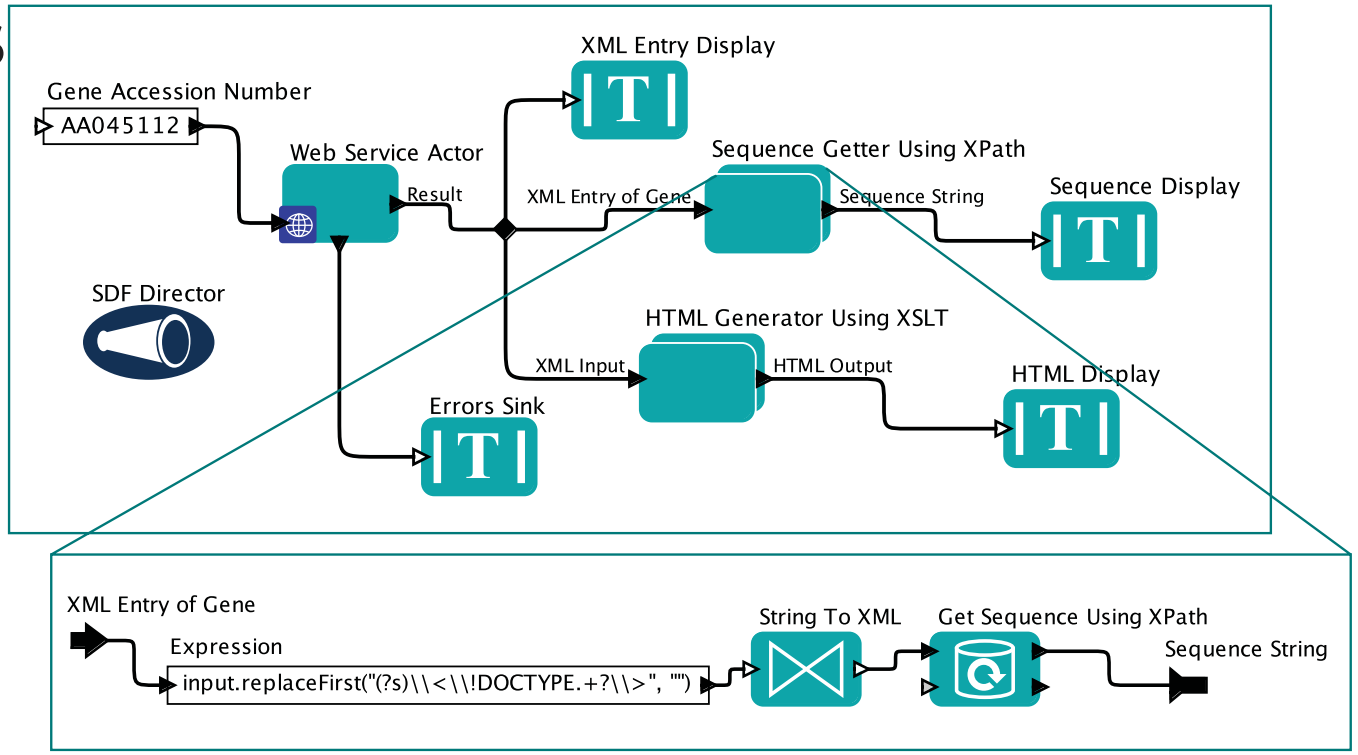
This work focuses on relaxing this requirement to obtain semantic interpretations even in the presence of Zeno conditions. The underlying aim is to establish a canonical denotational definition of timed discrete-event programming languages, thereby providing the means for reasoning about the correctness of the individual implementations, as well as allowing hidden commonalities of seemingly different timed systems to emerge.

Compositional Dataflow

Dataflow is a model of computation where a collection of actors communicate through unidirectional first-in first-out (FIFO) channels. Each dataflow actor is defined as a set of firing rules and a firing function. Firing rules specify what tokens must be available at the inputs for the actor to fire. The firing function specifies the input tokens consumed and the output tokens produced when the actor fires, and possibly a new set of firing rules along with a new firing function, thus reflecting an update to the state of the actor. Under suitable conditions, these actors define continuous functions, typically referred to as dataflow processes, over the complete partial order (CPO) of all finite and infinite sequences of tokens under the natural prefix order, and therefore provide for an alternative implementation of Kahn process networks.

Despite the abundance of dataflow applications in the embedded systems community, dataflow does not qualify as a programming language. Unless an evaluation strategy is specified, it is impossible to provide a semantic interpretation that will unambiguously assign a meaning to a dataflow network. A considerable amount of research has centered around evaluation strategies that will adhere to Kahn's least fixed-point solution. However, tailoring the operational semantics of a language to fit the simplicity and mathematical convenience of a particular denotational semantics is somewhat questionable. Park's bounded scheduling policy stands out for promoting bounded execution at the expense of convergence to Kahn's least fixed-point semantics. Nonetheless, it compromises compositionality and renders dataflow programming and execution intractable to formal reasoning.

This work focuses on a formal operational semantics for a dataflow programming language. The objective is to identify a set of computation rules that can yield bounded execution whenever possible, and at the same time remain uncoupled from the low-level machine details, and hence be more amenable to formal analysis. The notion of compositionality serves as a compass in this process, offering a solid correctness criterion for the proposed formalism. Questions on optimality issues are also to be addressed.



Kepler

Scientific Workflows

Above is a simple illustrative Kepler model created by Ilkay Altintas of the San Diego Supercomputer Center (SDSC) at UC San Diego that shows the use of web services and data transformation actors. This workflow demonstrates the use of the remote genomics data service to retrieve a genetic sequence. The Kepler actor uses the WSDL description of the web service to make its capabilities available to the model builder. In this simple example, the genetic sequence is displayed in three different ways: first in its native format (XML); second as a sequence element that has been extracted from the XML format; and third as an HTML document that might be used for display on a web site. The latter two transformation operations are performed using a composite actor that hides some of the complexity of the underlying operation. These composites can be thought of as 'sub-workflows' that execute a potentially complex set of tasks when called. One of these uses XPath to perform the transformation and one uses XSLT, thus illustrating the ability to leverage pre-existing infrastructure within Kepler workflows.

The Kepler Project is dedicated to furthering and supporting the capabilities, use, and awareness of the free and open source, scientific workflow application called Kepler. Kepler builds on the **Ptolemy II** system created within CHESS.

Kepler is designed to help scientists, analysts and computer programmers create, execute, and share models and analyses across a broad range of scientific and engineering disciplines. Kepler can operate on data stored in a variety of formats, locally and over the internet, and is an effective environment for integrating disparate software components, such as merging "R" scripts for statistical analysis with compiled C code, or facilitating remote, distributed execution of models via web services and computational grid infrastructures.

Using Kepler's graphical user interface, users simply select and then connect pertinent analytical components and data sources to create a "scientific workflow"—an executable representation of the steps required to generate results. The Kepler software helps users share and reuse data, workflows, and components developed by the scientific community to address common needs. And it helps scientists track and record the provenance of the data produced by their computational experiments.

Kepler is an open-source Java-based application that is maintained for the Windows, Mac OS X, and Linux operating systems. The Kepler Project supports the official code-base for Kepler development, as well as providing materials and mechanisms for learning how to use Kepler, sharing experiences with other workflow developers, reporting bugs, suggesting enhancements, etc.

Kepler is an open collaboration with many contributors from diverse domains of science and engineering. The project was started by researchers at the **National Center for Ecological Analysis and Synthesis** (**NCEAS**) at UC Santa Barbara, the **San Diego Supercomputer Center** (**SDSC**) at UC San Diego, and UC Davis as part of the **SEEK** (Science Environment for Ecological Knowledge) and **SDM** (Scientific Data Management) projects. Kepler participants include the above organizations plus CHESS, the **Lawrence Livermore National Labs (LLNL)**, the **University of Kansas**, **Monash University** (Australia), the **Univ. of New Mexico**, **James Cook University** (JCU, Australia), and **North Carolina State University**. The effort is supported by CHESS, the National Science Foundation and the Department of Energy.

Dissertations & Selected Publications

PhD Dissertations [2007-2008]:

Probabilistic Reachability for Stochastic Hybrid Systems: Theory, Computations, and Applications
Alessandro Abate

Learning Data Driven Representations from Large Collections of Multidimensional Patterns with Minimal Supervision
Parvez Ahammad

Platform Based Design for Wireless Sensor Networks
Alvise Bonivento

A Framework for Compositional Design and Analysis of Systems
Arindam Chakrabarti

Stochastic Omega-Regular Games
Krishnendu Chatterjee

Actor-Oriented Programming for Wireless Sensor Networks
Elaine Cheong

Automated Mapping for Heterogeneous Multiprocessor Embedded Systems
Abhijit Davare

A Design Flow for the Development, Characterization, and Refinement of System Level Architectural Services
Douglas Michael Densmore

A Hierarchical Coordination Language for Reliable Real-Time Tasks
Arkadeb Ghosal

A Platform-Based Approach to Low-Power Receiver Design
Yanmei Li

Compositionality in Deterministic Real-Time Embedded Systems
Slobodan Matic

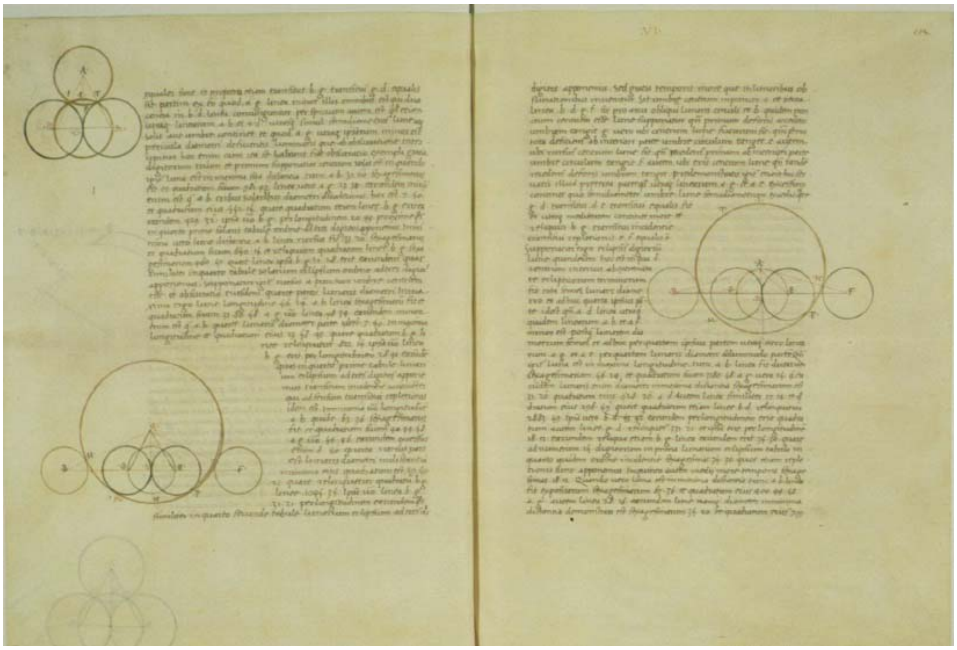
Single and Multi-CPU Performance Modeling for Embedded Systems
Trevor Conrad Meyerowitz

Distributed Networked Sensing and Control Systems: Robust Estimation and Real-time Control
Songhwai Oh

A Platform-Based Approach to Communication Synthesis for Embedded Systems
Alessandro Pinto

Games for the Verification of Timed Systems
Vinayak Prabhu

Attacks and Defenses of Ubiquitous Sensor Networks
Tanya Gazelle Roosta



Synchronous Reactive Communication: Generalization, Implementation, and Optimization
Guoqiang Wang

Composing and Validating Orthogonal Concerns and Heterogeneous Models
Guang Yang

Probabilistic Timing Analysis of Distributed Real-time Automotive Systems
Haibo Zeng

Operational Semantics of Hybrid Systems
Haiyang Zheng

Partial Evaluation for Optimized Compilation of Actor-Oriented Models
Gang Zhou

Interface Theories for Causality Analysis in Actor Networks
Ye Zhou

Optimizing Mapping in System Level Design
Qi Zhu

A Few of Many Published Papers:

M. Oishi, I. Mitchell, A. M. Bayen, and C. Tomlin, "Invariance-preserving abstractions of hybrid systems: Application to user interface design," *IEEE Trans. Control Systems Technology*,16(2) pp. 229-244, March 2008.

Y. Zhou and E. A. Lee, "Causality Interfaces for Actor Networks," *ACM Transactions on Embedded Computing Systems* (TECS), pp. 1-35, April 2008.

A. Sangiovanni-Vincentelli, "Is a Unified Methodology for System-Level Design Possible?," *IEEE Design and Test of Computers*, 25(4) pp. 346-358, July-August 2008.

C. Yeo, P. Ahammad, K. Ramchandran, and S. S. Sastry, "High-speed action recognition and localization in compressed domain videos," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 18, no. 8, pp. 1006-1015, Aug. 2008.

S. Tripakis, C. Pinello, A. Benveniste, A. Sangiovanni-Vincentelli, P. Caspi and M. Di Natale, "Fault-Tolerant Distributed Deployment of Embedded Control Software," *IEEE Transactions on Computers*, 57(10) pp. 1300-1314, October 2008.

A. Abate, M. Prandini, J. Lygeros, and S. S. Sastry, "Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems," *Automatica*, 44(11) pp. 2724-2734, Nov. 2008.

X. Liu, E. A. Lee, "CPO semantics of timed interactive actor networks," *Theoretical Computer Science* 409 (1): pp.110-25, December 2008.

Etc....
See http://chess.eecs.berkeley.edu for complete publications

Center for Hybrid and Embedded Software Systems
University of California at Berkeley
http://chess.eecs.berkeley.edu