Part I – Metropolis Infrastructure

1) Formal models of executions of Metropolis meta-model

Formal semantics of metropolis meta-model is being developed. The semantics is automata based, i.e. for each syntactic construct in the meta-model, we define an automaton specifying its semantics. The goal of this project is to be able to automatically generate these automata for a meta-model netlist. Ideally, automata would be specified in an analyzable format, e.g. SMV. Then they could be used as an abstraction for formal verification.

Number of people: 1
Mentors: Felice Ballarin (felice@cadence.com)

2) FSM generation for Metropolis constraints

Part of Metropolis meta-model is a set of constraints that could be specified in a logic called LOC (Logic Of Constraints). Not all LOC formulas can be represented by FSMs, but some of them can. Your goal is to define a subset which may be converted to FSMs and to develop the conversion procedure. Alternatively, you may try to find a conversion procedure from an LOC subset to a decidable temporal logic like LTL.

Number of people: 1
Mentors: Felice Ballarin (felice@cadence.com)

3) Hardware synthesis from Metropolis processes

This project has the goal of extending the existing translator from the Metropolis meta-model to the CDFG form used by high-level synthesis. In particular, it must handle multi-dimensional array accesses, function calls and port methods. It must also implement a translator back from the scheduled CDFG into Metropolis process code, this time using a synchronous (clocked) semantics.

Pre-requisites: Java programming (no knowledge of HLS is required)

Number of people: 1
Mentors: Yosinori Watanabe (watanabe@cadence.com)
Luciano Lavagno (luciano@cadence.com)

4) Co-simulation interface in Metropolis meta-model

In Metropolis, one can model both function and architecture specifications made of heterogeneous components using the same modeling mechanism called the meta-model. Tools are available to generate semantically equivalent descriptions in many different
languages, such as Java, C++, SystemC, which can be executed for simulation. It is often the case, however, the designs captured in Metropolis need to be co-simulated with the rest of the system (or environment) that are modeled in the same target language (say SystemC) but not within the Metropolis environment. Typical examples of this case are when such designs are provided by third-parties that do not want to disclose the source code, or when the part being designed in Metropolis has to interact with code inherited from products of the previous generation. This project investigates various approaches to achieve such co-simulation. In one way or another, an interface must be defined between the two designs. A generic interface that can be used for many design scenarios will be attractive for its versatility, while specific one may have advantages for efficiency, and such trade-off will be experimented.

**Number of people: 1**
**Mentors**: Yosinori Watanabe (watanabe@cadence.com)

5) Metroshell to explore platform-based design space

Metroshell is the text-based user interface of the Metropolis environment. It allows one to load or browse designs for both functions and architecture, specify refinement or abstraction, relate two platforms, successively apply various checks on the designs or call simulation, synthesis, verification tools. It is written in Jacl, the Java version of Tcl script language.

This project enhances the capability of Metroshell, which involves necessary enhancement of the compiler of the Metropolis language. It will add more interactive features such as design debugging and more commands for design validity checking. The resulting features will be tested on design examples being specified in the Metropolis project.

**Number of people: 1** (Knowledge on compilers and Java programming recommended)
**Mentors**: Yosinori Watanabe (watanabe@cadence.com)

6) Synchronous Modeling in the Metamodel and Esterel

This project involves the development of a library of synchronous constructs for use in the Metropolis Metamodel, their translation into an appropriate input for an Esterel compiler, and the re-incorporation of the result of the compiler back into the MMM. The latter would be in the form of a single process, with the same behavior as the original synchronous composition of processes.

**Number of people: 2**
**Mentors**: Trevor Conrad Meyerowitz tcm@ic.EECS.Berkeley.EDU
Luciano Lavagno (luciano@cadence.com)
7) Interfacing SUIF (and/or Trimaran) with Metropolis

SUIF is a publicly available compiler infrastructure that has been developed by several universities, including Stanford and Illinois, to support research in the software compiler domain, with a particular emphasis on parallelizing compilers. Trimaran is also publicly available, but it is an effort sponsored in particular by HP for its EPIC VLIW architecture (one of the roots of Intel's Itanium architecture). Both of them provide powerful algorithms for dismantling high-level program into low-level simple instruction, discovering parallelism, performing data flow analysis, and so on. An interface with those tools would significantly enhance the Metropolis toolbox both for software and for hardware compilation and synthesis.

The task of this project is to pick one (or both, if more than one person is interested) of these compilation frameworks, and build translators from the MMM Abstract Syntax Tree to their Intermediate Representations, and vice-versa.

Background: programming in C++.

Number of people: 1
Mentors: Luciano Lavagno (luciano@cadence.com)
Harry Hsieh <harry@cs.ucr.edu>
Watanabe (watanabe@cadence.com)

Part II – Metropolis Applications

8) Modeling continuous-time sub-systems in Metropolis

In several application domains it is difficult or inconvenient to model the discrete and continuous components in separate environments. A typical example is offered by the real-time control community: in order to access how the performance of a controller is affected by architectural and mapping choices, the closed loop system should be considered. Realistic and quantitative answers to this question during the early phases of the development are a precious tool for product development. The first step in allowing the two types of subsystems to co-exist is defining the way they can communicate/interact. The students should define a set of interfaces to effectively exchange information across the discrete-continuous boundary. Communication media across the two worlds should implement some of these interfaces. The students should also test this interaction model by defining a set of process templates to model ordinary linear differential equations. The combination of media and processes
defines a platform for modeling discrete-continuous interaction. It is expected that models built within this platform can be simulated using one of the simulation backend tools in Metropolis. Hence some mechanism to solve (i.e. “integrate”) linear differential equations should be provided as part of the platform either by extending one of the simulators or by implementing the integration in the metropolis processes/media. The development of backend tools to facilitate other types of analysis (e.g. reachability) would be a nice addition.

Some of the possible issues to be addressed can be the following:

1. what is an intuitive and efficient representation for a continuous time system in the context of the Metropolis meta-model? Are the metropolis black box structures adequate?
2. How should an interface between a continuous time block and the rest of the simulator be structured?
3. Should we allow a continuous time system to initiate a communication (generate events)? Or should we restrict ourselves to a more passive model where outputs are generated only when requested by a discrete component?

**Number of people: 1-2**

**Mentors:** Yosinori Watanabe (watanabe@cadence.com)
Felice Balarin (felice@cadence.com)
Howard Wong-Toi (howard@cadence.com)
Claudio Pinello (pinello@eecs.berkeley.edu)

**9) Protocol Implementation in Metropolis**

Siegmund and Mueller propose to represent protocols stacks in a mixed declarative-imperative formalism (as in other related work), and extend the SystemC language to support this style of specification. By doing so, they provide a mean to simulate a design at various levels of detail. At the same time, the protocol specification can be used to automatically synthesize a model of a hardware implementation, which in turn can be used for simulation.

We propose to implement a similar hierarchical technique (not necessarily in declarative form) in Metropolis by using the notion of refinement and of mapped behavior, and the concept of a communication medium. In particular, refinement makes it possible to replace a higher level description with a more concrete description, thus trading-off the performance of simulation and analysis with the level of detail. In Metropolis each layer of the protocol stack can be represented as a platform. The layers that sit above the platform use the services that the platform offers to implement their services. The notions of mapping and refinement are used to describe the relationships between the layers.

The goal of this project is to define a methodology for specifying protocol stacks using refinement and mapping, and to test the specification with some reasonable examples.
10) The Modeling and Refinement of a Simple Microprocessor Using YAPI in Metropolis

Kahn Process Networks are a powerful model for concurrent programming and modeling. Among their most useful properties is that of being fully deterministic. YAPI is an extension of process networks developed by Philips that has been used for modeling multimedia applications. The first task in this project is to implement an untimed model of a simple microprocessor in the YAPI library in metropolis. From here the model will be refined to include timing, pipelining, and other microarchitectural features. There is also the opportunity to try mixing the YAPI simulation with other models in Metropolis such as FSM's and SDF.

Number of people: 1
Mentors: Trevor Conrad Meyerowitz tcm@ic.EECS.Berkeley.EDU

11) Analysis of PicoRadio Design in Metropolis

Wireless sensor networks have many applications such as smart homes, interactive museums, personal-environment offices, etc. PicoRadio, a research project currently being carried out at Berkeley Wireless Research Center (BWRC), is about how to design the next-generation ad-hoc ultra low-energy wireless sensor networks. With no doubt, a formal design methodology is one of the most important factors to ensure the success of the Picoradio project.

The goal of this project is to analyze the Picoradio design from the point of view of platform-based design, review its protocol specification, inspect its software and hardware implementation, and present a better design methodology.

Your tasks are to (1) find refinement relationship between protocol layers, (2) model architectures, (3) map the protocols onto different architectures.

Skills: basic knowledge of network protocols and some programming

Number of people: 1
Mentors: Rong Chen rongchen@eecs.berkeley.edu, Marco Sgroi sgroi@eecs.berkeley.edu,
Part III – QSS Studies

12) Environment modeling in quasi-static scheduling

Quasi-static scheduling (QSS) is a powerful technique to reduce overhead caused by communication and context switches for a network of processes[1]. It takes as input a set of communicating processes, where each process is associated with a sequential program, and generates a single sequential program (schedule) by interleaving operations of the processes. Currently, it employs a simple assumption about the environment that the inputs from the environment are either fully controllable or fully uncontrollable while the outputs are always fully controllable. In practice, the inputs and outputs are mutually dependent, and thus the resulting schedule may not correctly interact with the environment in general because of the over-simplified assumption.

This project investigates conditions that guarantee the correctness of the schedule when interacting with the environment. Specifically, for a given subsystem $S'$ that is a part of a closed system $S$, one wants to apply the quasi-static scheduling to $S'$ and derive a schedule that is consistent with the function of $S'$ within $S$. The $S/S'$ part of system $S$ is treated as the environment for $S'$ during scheduling and must be properly modeled. Two approaches to be explored are:

1. Explicit environment modeling
   Represent the control flow of the overall system $S$ by a Petri net (PN).
   Define the set of reduction rules for the PN model of the environment $S/S'$. Develop QSS technique for scheduling of $S'$ together with its reduced environment model. Do experiments with benchmark examples.

2. Implicit environment modeling
   Formulate environment properties that are sufficient to guarantee that the schedule derived with the current QSS approach (or its minor modification) would always be consistent with the environment. Check these properties on MPEG and COSY benchmarks.


Mentors: Yoshi Watanabe (watanabe@cadence.com), Alex Kondratyev (alex@cadence.com)

13) Quasi-static Scheduling with Lossy Communication

Quasi-static scheduling (QSS) is a powerful technique to reduce overhead caused by communication and context switches for a network of processes[1]. It takes as input a set of processes communication through FIFOs, where each process is associated with a sequential program, and generates a single sequential program (schedule) by interleaving
operations of the processes. It employs Petri net as the underlying formalism to model the specification.

The objective of this project is to extend the technique in order to handle lossy communication semantics between processes. A Petri net structure must be defined to model this semantics and the Petri net generator must be extended so that this structure is used when the communication semantics is lossy. There is a good example from the picoradio project of Berkeley Wireless Research Center, where a software implementation of six processes with this communication semantics turned out to be very inefficient because of overhead of context switches. Once the extension of the QSS technology is made, it will be applied to this example to evaluate the efficiency of the resulting implementation with respect to the original one.


Number of people: 1
Mentors: Yosinori Watanabe (watanabe@cadence.com)  
Alex Kondratyev (kalex@cadence.com)

Part IV – DRAFTS Project

14) Implementing run-time support for DRAFTS

The DRAFTS (Distributed Real-time Applications Fault Tolerant Scheduling) project aims at automating synthesis from a dataflow-like model to executable code, in such a way that the implementation is fault tolerant and meets real-time constraints. It is based on some assumptions on the (distributed) run-time environment like message handling, semi-static scheduling, coordination. The aim of the proposed class-project is to code a prototype of such an environment. An executable specification running in Metropolis meta-model is available as a reference, this should be translated to a convenient language (preferably C/C++) to run on a network of Unix (Linux) workstations connected through TCP/IP links.

Prerequisites: some experience in programming networking applications

Number of students: 2
Mentor: Claudio Pinello (pinello@eecs.berkeley.edu)

15) Define a formal calculus for dealing with deadlines, WCET and timeouts

In the scope of the DRAFTS project a dataflow-like model is replicated and scheduled to meet real-time constraints and to tolerate faults in the architecture.
Starting from worst case execution times (WCET) and deadlines, a given schedule and scheduling policy and some fault scenario it is possible to derive for each actor:

1) earliest & latest start time
2) earliest & latest completion time
3) latest deadline (admissible completion time)

The student will formulate the problem formally (e.g. using max-plus algebra) and devise an algorithm to efficiently compute the times listed above.

**Number of students:** 1  
**Mentor:** Claudio Pinello (pinello@eecs.berkeley.edu)

---

**Part V – Simulink Exploration**

**16) STARS analysis for Simulink designs**

Simulink is a design environment for reactive real-time controllers, which provides the user with a rich library of blocks, ranging from adders, multipliers and muxes to FIR filters and integrators. STARS is a method for worst-case execution time that computes a conservative upper bound on the number of activations of a block given a netlist and an abstract model (based mostly on event counters) of each block.

The goal of this project is to analyze a Simulink netlist, using the Real-Time Workshop API, and generate an abstract STARS netlist that instantiates a STARS model of each Simulink block. A suitable subset of the full block library to be supported will be defined at the beginning of the project.

**Prerequisites:** knowledge of Matlab/Simulink.

**Number of people:** 1  
**Mentors:** Felice Balarin (felice@cadence.com), Luciano Lavagno (luciano@cadence.com)

---

**17) Definition of the Simulink MOC in Metropolis**

Simulink is the most widely used specification capture tool for embedded control. It allows designers to model discrete and continuous time controllers, and has a statechart editing facility. The goal of this project is two-fold:
- define the semantics of block activation (Model of Computation) of Simulink in Metropolis, using the meta-model facilities (await, communication media, constraints and so on),
- create an export script in Simulink that will generate an appropriate MMM netlist using the defined MOC.
Depending on the number of people taking the project, the latter could involve just one activation mechanism (e.g., discrete time single-rate) or multiple methods, and Statecharts as well.

Background: Simulink

**Number of people:** 1-3

**Mentors:**
- Luciano Lavagno (lavagno@polito.it)
- Roberto Passerone (roby@eecs.berkeley.edu)
- Howard Wong-Toi (howard@cadence.com)

**Part VI – UML Modeling**

**18) UMMMM: UML Modeling of the Metropolis Meta-Model**

You know the first step of embedded system design is specification, but you may wonder how exactly such a process starts. A good design specification is crucial to the overall success of a design process, and thus it has always been an active research area. UML, an object-oriented modeling language standardized by the Object Management Group (OMG), has captured much attention in the embedded system design as a possible solution for raising the abstraction level of the specification, and thus, facilitating the following synthesis and verification processes.

The OMG has recently approved a UML profile for resources, schedulability and time. This has several similarities (and several differences) with respect to the Metropolis Meta-Model. The former is more abstract and devoted mostly to software, while the latter is more concrete and spans the HW/SW boundary.

The goal of this project is to explore more deeply differences and similarities between the two, and (ideally) demonstrate that the MMM can be expressed within the framework of the UML profile.

Your tasks are to (1) compare UML-Platform (a particular UML profile) with UML-RT (the OMG adopted profile) to analyze their similarities and differences, (2) create a complete class hierarchy for the main behavioral concepts of the MMM, and show how it can be instantiated to represent behavior, mapping, communication, performance. The tasks will be done by using a simple design example.

**Skills:** basic knowledge of UML is desirable but not absolutely necessary

**Number of people:** 1

**Mentors:**
- Rong Chen (rongchen@eecs.berkeley.edu)
- Marco Sgroi (sgroi@eecs.berkeley.edu)
- Luciano Lavagno (lavagno@polito.it)
Part VII – Network Discovering

19) Efficient low-power network-discovery algorithm for wireless embedded sensor networks

The goal of this project is to explore efficient low-power algorithms for discovering the nodes and the connections of a wireless embedded sensor network (WESN) within a pre-defined local (Flocal) part of the deployment field (F).

An ad-hoc sensor network is a network of numerous wireless nodes where each node has sensing, storage, computation, communication and possibly actuating resources. Each node has a limited power source that determines the lifetime of the node. Therefore, energy is the most critical factor in sensor networks, in all of its design levels (i.e. hardware, network layers and applications). Numerous studies in sensor networks have already demonstrated that the communication between the nodes is the dominating source of energy consumption. While traditional wireless network architecture has been based on system of static base stations, it appears that that multi-hop networks where each node communicates with a few close nodes is most efficient in terms of energy saving and bandwidth reuse. In multi-hop network, each node communicates with geographically distant other nodes using intermediate nodes to build a communication path.

In many sensor network applications, in order to save power, network just reactively responds to unusual events. Since the nodes are not always active and the network might be dynamic, each local application would require exploring the connectivity and coverage of the local neighborhood (Flocal) at its execution time (the local neighborhood for the target applications is already statistically modeled and determined). The algorithm minimizes the power cost of the local procedure that explores the network within the local field (Flocal) while it (the algorithm) completely discovers the network within Flocal. There are already a number of proposed algorithms for this problem. A project would consist of implementing and comparing at least two of these discovery algorithms or even better, proposing a new more efficient discovery algorithm. Basic understanding of algorithm and programming skills in C/C++ is required.

Number of People: 1
Mentor: Farinaz Koushanfar (farinaz@eecs.berkeley.edu)

Part VIII – Current Spectrum

20) A Methodology for the Computation of an Upper Bound on Noise Current Spectrum of CMOS Switching Activity **

Background: During its switching activity a digital block injects noise current into the power supply network and into the substrate. This current may affect the operation of other blocks (both analog and digital) and needs, therefore, to be estimated. The problem is not trivial since the injected noise current depends on the input test vector applied to the circuit.
In particular, for some application what is needed is the estimation of an upper bound of the current spectrum in the frequency domain. In [1], a methodology has been proposed to estimate such noise current spectrum upper bound over the whole input space. It consists of two major steps:

1. Library characterization for noise spectrum
2. Algorithm to compute the upper bound estimation for a given circuit.

**Goal of the project:** Perform library characterization for noise spectrum and verify the validity of proposed model. Implement the algorithm proposed in [1] and verify its validity. Explore possible improvements in accuracy and/or computation time to the proposed methodology.

**Number of people:** 1-2

**Mentors:** Alessandra Nardi (nardi@eecs.berkeley.edu)


**Part IX – Survey**

21) Review of current practice in modeling and analysis of discrete and continuous components.

Some design environments support modeling of both continuous and discrete components, examples are VerilogA, Simulink, etc. The student should review what typical uses of these tools are, what features are the most useful and what are the most desired by designers (e.g. reachability analysis, stability analysis, schedulability analysis, validation by simulation, performance assessment, etc.). It is expected that these features and uses change per application domain. The student should focus on a few domains (e.g. Multimedia, Automotive Control, etc.) and collect and organize the information. The outcome of the review should be a set of guidelines for the development of the Metropolis framework to include support for analog/continuous components.

**Number of people:** 1

**Mentors:** Yosinori Watanabe (watanabe@cadence.com)
Felice Balarin (felice@cadence.com)
Howard Wong-Toi (howard@cadence.com)

Note: A project marked with “**” can be taken as a project in either EE249 or EE219A. If you want to take it as a joint project in both EE249 and EE219A, you need to first get permission from the professor and then discuss with mentors to increase the project workload correspondingly.