

DESIGNING WIRELESS PROTOCOLS: METHODOLOGY AND APPLICATIONS

M. Sgroi, J. L. da Silva Jr., F. De Bernardinis, F. Burghardt,
A. Sangiovanni-Vincentelli, J. Rabaey
University of California at Berkeley

Abstract – Communication protocols are essential components of wireless systems. Present methods for protocol design are heuristic in nature and are not suited for next generation wireless systems where time-to-market concerns require correct-the-first-time implementations. In this paper we present a new design methodology for wireless protocols based on the principle of orthogonalization of concerns. In particular, the methodology separates function and architecture design and emphasizes the use of formal models to ensure correctness and reduce design time. Protocols are described using Co-design Finite State Machines (CFSMs), a model of computation that has been introduced to allow the efficient capture of both the control and the data processing parts of the specification. Furthermore, algorithms for automatic hardware and software synthesis from CFSMs are available. This allows a fast exploration of different HW/SW partitions and the analysis of tradeoffs involved. *Intercom*, a wireless system supporting full-duplex voice communication among different users, is presented and the design of its protocols is described. The design methodology presented here will be used for the design of *PicoRadio*, a low-power and highly adaptive network of sensors.

1. INTRODUCTION

Future wireless systems will provide reliable and mobile connectivity while supporting a broad variety of multimedia services at very low cost. Among next generation wireless systems, sensor networks are of special interest. Distributed, large-scale sensor information networks combine a number of seemingly contradictory implementation requirements. On the one hand the networks have to be versatile, self-organizing multi-functional, and dynamically reconfigurable. This implies that the communication and computational components of the sensor nodes need to be adaptive and programmable. On the other hand, extensive large scale coverage requirements which implies large numbers, require that the sensor nodes be inexpensive, have a very small footprint and consume a minimum amount of energy to extend their operational lifetimes.

Deep sub-micron integrated circuit technology allows implementing within a single silicon circuit a variety of heterogeneous functions, e.g. data processing, channel control, modem, interfacing, position location, thus providing a possible solution to the requirements stated above.

An essential problem in the design of wireless systems is the design of the protocol, not only because it is usually error-prone and time-consuming, but also because the design decisions made in this phase greatly affect the quality of the entire system implementation. Pushing performances/cost requirements and integration capabilities to the limit requires developing new design tools and methodologies. Our research efforts move indeed in two main directions:

- design of fully integrated wireless systems from specification to final chip implementation,

- development of a complete methodology for the design of low-power and dynamically adaptive protocols.

This paper provides an overview of both efforts. In the first part (Section 2) we describe the methodology we are developing for the design and the implementation of protocols. In the second part we describe *Intercom*, a wireless system for voice-based conferencing within a single-cell network, currently under development in Berkeley. The paper will be concluded with perspectives on future developments, especially with respect to *PicoRadio*, a low-power and highly adaptive network of sensors and monitors.

2. DESIGN METHODOLOGY

Protocols are currently designed using a rather informal approach. Starting from a textual description of the service that the protocol must provide, the design process is iteratively carried on through several steps. At every step the designer informally refines the protocol by adding further details and eventually removing existing errors. This process terminates only when the designer is convinced that the protocol is free of errors and satisfies all the system requirements. In this design method, the refinement process is carried out informally and, as a consequence, it is not guaranteed to maintain the properties of the initial specification. Hence, the final implementation may be incorrect or inconsistent with the initial specification. It is not surprising that most of the efforts of the protocol design research community have been devoted to formal verification. Besides the need of formal techniques, the design methodologies currently used lack of real support for performance analysis and therefore do not allow to address early in the design process important issues such as architecture selection and HW/SW partitioning.

Our goal is to define a *formal* methodology that allows to

- design *correct-by-construction* protocols
- implement them *efficiently*, i.e. using a minimal amount of physical resources (area and/or energy)

The design methodology we propose for wireless protocols is based on a *top-down* flow rooted on a clear separation between behavior and architecture. This paradigm can be applied at any stage of the design. In fact, we advocate starting thinking along these lines from the network level where the topology of the network is decided, or even at the application level. Network design is based on assumptions of the performances of the network nodes in terms of power and timing. If these assumptions are satisfied, then our design methodology ensures that the network will work correctly. These assumptions become constraints for the design of the nodes and are propagated down the design hierarchy. This guarantees that, if the constraints at the lower level of the hierarchy are satisfied, we do not have to verify the top level again. In addition, properties that are typically enforced at the top level of the hierarchy are extremely difficult to verify at lower levels where implementation details are overwhelming. We refer to this

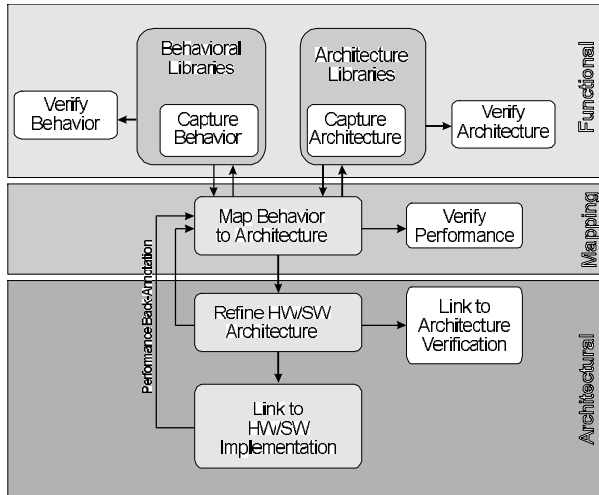


Figure 1 - Design phases for the proposed methodology

aspect of the methodology as *successive refinement*. Note that the architecture/function orthogonalization principle can be applied to all levels of the design hierarchy as well as the constraint “mapping” mechanism. We refer to this aspect as the “fractal nature” of design since the same pattern repeats itself on all scales.

As shown in Figure 1, the flow based on architecture/function co-design consists of two main phases. During the first one, the system specification is captured in a pure functional way and a set of alternative architectures that can be used to implement the specification is modeled. In the second phase, the components of the behavior are mapped onto the blocks of the candidate architectures and the performance of the resulting designs are estimated to evaluate the quality of the mapped design. The final selection is then synthesized and compiled into an actual implementation.

System specification is verified through functional simulation. At this level of abstraction there is no distinction between hardware and software since this distinction is the result of an architectural choice and of the consequent mapping of functionality to elements of the architecture. At the functional level, there is no explicit notion of time, and no assumption (explicit or implicit) is made about relative execution times of different blocks. The execution of each block is considered to be instantaneous and any synchronization between different blocks has to be realized by means of protocols. This adds some initial overhead to the system specification phase, but also allows an unbiased exploration of the design space. Some parts of the protocol are intrinsically dependent on time. In that case, time has to be modeled with signals connecting to timers. With functional simulation, we can verify that the *system specification is consistent and that interactions between different units of the system behave correctly*.

Selecting the **model of computation**, capturing the protocol functionality, is very important, as it has a great impact on the (possibly automatic) generation process of both hardware and software implementations. CFSM's are an extension of FSM's, hence, they are intrinsically good in modeling control paths [1]. CFSM's also can include data computations as part of the transitions, and can therefore model protocol data-paths as well. CFSM's networks are globally asynchronous, locally synchronous;

thus they are able to model effectively different hardware/software partitions, and asynchronous communication events. In short, CFSM's offer a model-of-computation that can be targeted indifferently towards hardware or software, and are a natural candidate for modeling our wireless protocols.

Architectures are described in terms of programmable units (CPU's), ASIC's, interconnect networks, and real time operating systems (schedulers). They capture the computational capabilities, the degree of parallelism, the sequentialization of blocks sharing the same resources, and the capacity of the inter-block communication channels.

Mapping consists of associating each functional block with an architectural resource. This mapping affects the behavior of the mapped system by introducing time delays. The effects of these implementation-induced delays can now be analyzed through performance simulations. It is in this phase of the design that trade-off analysis is carried out and an architecture is eventually selected in terms of its cost, performance, and energy consumption. To evaluate the quality of an architecture with respect to the specification, each component of the architecture has to be endowed with a method for evaluating delays, energy consumption and cost. If a function is mapped onto a programmable component such as a microprocessor, its implementation consists of software written in an appropriate language that may depend on the application domain and/or the microprocessor. In that case, an estimator predicting the execution time for the software on the selected microprocessor has to be available.

The paradigm described above is the basis of both the POLIS [1] system developed at UC Berkeley and the Virtual Component Composer (VCC) by Cadence Design Systems, Inc [2]. Both of them were used in our design experiments. If the functionality of the system is captured with CFSM's, automatic and accurate performance estimation, based on the same powerful software synthesis techniques that are used in the final implementation generation, is available. In the VCC environment, it is also possible to capture functionality of a functional module with a C program. Accurate performance estimation becomes more difficult, because of the somewhat unpredictable behavior of the C-compiler that will be used for the final code generation. However, even a ballpark evaluation is acceptable in this architecture exploration mode. Once a particular architecture is selected, more accurate estimation can be obtained by refining the information about the architecture with implementation details that make the evaluation process more precise.

Mapping of functional blocks into hardware can be done either manually by linking the function with a library element known to perform that function, or automatically by spinning off a synthesis process. If the behavior of the function is described in the CFSM format, it is relatively easy to generate a Verilog or VHDL file that can be fed into the behavioral or logic synthesis tools.

At the end of the exploration phase, a complete description of the system in both the hardware and the software components is available, as well as a first-order estimation of the performance and cost of the resulting system. From here, it is possible to go directly to the actual implementation along paths that eventually will be totally automatic.

In the next section we apply this methodology to the design of a wireless system of relevant complexity.

3. THE INTERCOM SYSTEM APPLICATION EXAMPLE

3.1. SPECIFICATION

The Intercom is a single-cell wireless network supporting full-duplex voice communication between mobile users located in a small geographical area. The network is composed of a number of units, called *remote terminals* that operate in one of the following modes:

- Idle (the remote is switched on but has not subscribed to the network and cannot participate to a conference)
- Active (the remote has subscribed to the network and therefore is enabled to participate to a conference)
- Communication (the remote is transmitting and/or receiving data).

Each remote can request one of the following services: subscription/unsubscription to enter active mode, query of active users, start/end conference with one or multiple remotes, and broadcast communication.

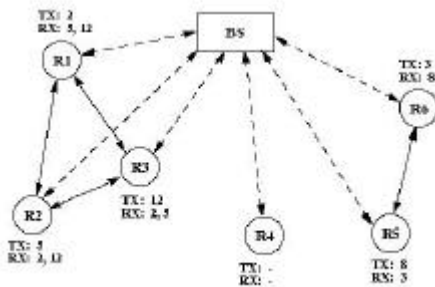


Figure 2 – Intercom Network

An example scenario is shown in Figure 2. Six remotes are currently active: R1, R2, and R3 are involved in a conference, R5 and R6 are in private conversation, and R4 is not communicating with anyone.

The system specification includes also the following requirements:

- number of users that can be simultaneously subscribed (20)
- minimum throughput and maximum latency of the data transfers, derived from standard quality voice requirements.

3.2 PROTOCOL STACK

The protocol stack is designed by applying communication refinement to the specifications above along the lines discussed in the previous section. According to the methodology, we identified the functionality of the layers, defined the interfaces between layers, and derived constraints for each layer in a top-down fashion from the system requirements. In this section, we motivate the design decisions that guided the protocol refinement process.

The network topology includes a unit, called *base station* that coordinates the network operation. The base station only provides control functionality, while voice channels are set up as peer-to-peer connections between remotes. This configuration helps to reduce the overall bandwidth requirements, compared to the star-connected network configuration typical in cellular environments. The base station keeps track of the evolution of the network con-

figuration using an internal database that stores information about the active users and the ID of the physical channels.

In general, all Intercom units have identical capabilities, i.e. each unit can be either a base-station or a remote (the mode of a unit is set on power-on). This design choice makes the protocol more robust in case of failures, as it enables active reconfiguration should the base-station fail.

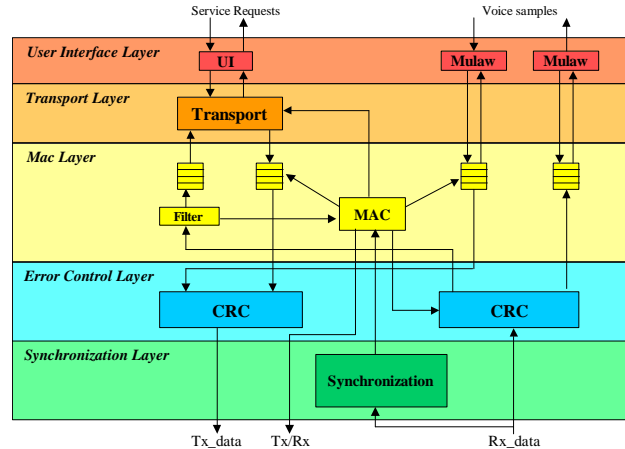


Figure 3 – Intercom protocol stack.

The protocol stack has two interfaces with the external world: one with the user, providing a voice channel and service requests (e.g. Subscribe, StartConference, EndConference), and one with the radio that forms the RF link. Figure 3 shows the structure of the protocol stack, which is composed of the following layers:

User Interface Layer. The UI module interfaces the remote with the environment from which it takes user service requests. The UI filters these requests so that only the relevant ones are forwarded to the lower layers (for example the UI discards a StartConference request if the user is not in active operation mode) and no resources are wasted. The Mulaw module performs a logarithmic quantification on a PCM-encoded bit stream, sampled at 64 kbps.

Transport Layer. The Transport Layer performs the network management by exchanging messages (containing information such as remote ID and type of service request) and forwarding them to the lower layers. The same message is retransmitted several times until an acknowledgment is received from the base station.

Mac Layer. A Time Division Multiple Access (TDMA) scheme addresses the shared medium access problem. Therefore, a set of two or more (depending on the parties involved in a conference) slots forms the ID of a physical channel corresponding to a virtual channel. Slots are allocated on a per-demand basis. When a remote wants to start a conference, it sends a request to the base station. If the base station detects that there is a physical channel (i.e. one slot per user) available, it communicates to each user the slots in which they can transmit and receive. This information, once received by the remote, is stored within the MAC layer so that at the beginning of each new slot it can activate the transmit

or the receive functions.¹ The picture includes also a set of queues at the MAC Layer, one queue for each flow of data. Queues are used to store the incoming data while the channel is used by other remotes. The slots are allocated as follows: the first two slots are allocated for control channels, one up-link (from the remote to the base station) and one downlink (from the base station to the remote) while the remaining eighteen are used for data transmission among users. A frame lasts 62.5ms frame, while each slot lasts 3ms. These values have been derived from the voice requirements on delay and throughput.

Consider again the scenario described in Figure 2. Annotations next to each remote show its transmission and reception slots. R1 can transmit in slot 2, R2 in slot 5, and R3 in slot 12. A remote can transmit in more than one slot if it needs a larger bandwidth and can receive in multiple slots (e.g. R1 in slots 5 and 12), e.g. in the case of a conference with multiple remotes.

Data Link Layer. This layer performs the (de)packetization of the bit-stream. It includes the following functions: (1) **Error Control**, consisting of a Cyclic Redundancy Check (CRC) function that allows to detect errors in the received packets and eventually discard them. (2) **Synchronization**, detecting frame and slots synchronization pilots in the bit stream coming from the wireless channel. It also includes a slot counter and notifies the MAC layer with the number of the current slot.

Physical Layer. The Physical Layer includes intensive bit-level data processing such as QPSK (de) modulation, timing recovery, phase/frequency correction. The Direct Sequence Spread Spectrum radio operates at a rate of 1.6 Mbps at 2.4 Ghz.

3.3 IMPLEMENTATION ISSUES

The Intercom has been designed using two different design methodologies. The first implementation was derived using an informal methodology, where the specification of the control protocols was done using SDL [3] based tools, and the executable 'C' code was generated directly from the SDL model. The MAC layer was supplied externally as a VHDL description. There was no real design partitioning and exploration, the system was implemented based on some initial estimation of complexity and no real alternative was quantitatively considered. The prototype system was realized with a StrongARM 1100 processor subsystem, a Xilinx XC4013XL FPGA, and a Proxim 1.6Mbps frequency hop radio.

The intercom system is currently being redesigned, integrating the entire system in two chips, one for the digital and one for the analog components. The final target, however, is a single-chip implementation. At this time, we have specified the functionality of the protocol using a CFSM model. Consequently, we can map this functionality to either hardware or software. Using the proposed methodology we are able to perform architecture exploration very quickly. With an effort of 4 man/months to learn the methodology, to get familiar with the tools and to model the protocol, we are now able to explore several architectural alternatives (about 3) every day! We are currently in the mapping phase, which will determine the correct partitioning between hardware and software.

¹ Connections can be completely remapped each frame by the base-station by broadcasting the content of the *slot set database*. The slot set database contains one transmit and one receive slot set for each remote.

Collecting a set of different partitions, we will be able to focus on the power consumption of each mapping and eventually select the optimal implementation for the system.

4. CONCLUSION AND FUTURE WORK

Given the recent technology advances, it becomes conceivable to build networks of heterogeneous nodes collecting different types of data (measurements, voice, etc.) to realize infrastructures for reactive environments. Building pervasive reactive environment requires a cheap, lightweight, and small network element, which we call a PicoRadio ("Pico" denotes that our target is a ultra low power implementation, possibly less than a few milliwatts range). To be integrated in reactive environments, each node must be as small as possible, easily re-locatable and consume very little energy. To achieve these goals, it is also important to investigate carefully analog/digital partitioning. Each PicoRadio has short range (3-10 meters), so an adaptive network infrastructure is required to allow it to communicate with other nodes. This is realized with a multi-hop, flexible and scalable network with hundreds of nodes.

The development of a network of ultra low power PicoNodes relies on the ability to save energy by trading-off communication and computation in all layers of the protocol stack (from network to physical), as well as on the conception of new low energy radio designs. To rise to this challenge, we need both highly reconfigurable devices available in the target architecture as well as a new design methodology allowing the quick and reliable exploration of a number of different architectures.

We presented here a design methodology that has the characteristics needed for the design of a complex wireless system as the PicoRadio network, where the design of the network and of its nodes should be carried out in a unified environment and with a common approach. We are testing the methodology by re-designing the Intercom, a simpler system that was designed a first time with an informal and ad hoc method. Preliminary results show that the advantages of the formal refinement-based methodology are very visible. Our future goals include the tuning of the existing and the development of novel models-of-computation tailored to wireless protocol design. In addition, we plan to develop new synthesis algorithms and to adjust the existing tools to provide a powerful design environment for academia and industry.

REFERENCES

- [1] F.Balarin, M.Chiodo,P.Giusto,H.Hsieh, A.Jurecska, L.Lavagno, C.Passerone, A. Sangiovanni-Vincentelli, E.Sentovich, K.Suzuki, B.Tabbara. *Hardware-Software Co-Design of Embedded Systems: the POLIS approach*. Kluwer Academic Publisher, 1997.
- [2] *Virtual Component Composer (VCC)*. Cadence Design Systems, Inc. <http://www.cadence.com/>
- [3] R.Saracco, J.R.W. Smith, R.Reed. *Telecommunications systems engineering using SDL*. North-Holland – Elsevier, 1989.