Interface Synthesis:  
a vertical slice from digital logic to software components

Gaetano Borriello
Dep’t of Computer Science & Engineering  
University of Washington  
Seattle, WA 98195-2350 USA  
gaetano@cs.washington.edu

Luciano Lavagno
Dipartimento di Elettronica  
Politecnico di Torino  
Torino 10129 ITALIA  
lavagno@polito.it

Ross B. Ortega
Dep’t of Computing & Software Systems  
University of Washington  
Bothell, WA 98021-4900 USA  
ortega@u.washington.edu

ABSTRACT
Interface synthesis seeks to automate the process of interconnecting components. There are many levels of interconnection that must be considered including electrical, power, logic, register-transfer, device drivers, and higher software levels. This presentation will cover a vertical slice of the interfacing problem from digital logic up to coordinating communications between software components. The focus will be within an embedded systems context where the interfacing is between processors and memory and peripheral blocks as is the case in system-on-a-chip design.

The structure of the tutorial will parallel the history of CAD efforts in this area. We will begin with the early work in interface specification and logic synthesis then proceed on to the problems of interconnecting hardware to processors and their software, and finish with purely software interfaces involving inter-process communication and protocols between multiple processors. At each level we will discuss specification, synthesis, and verification aspects as well as highlight the currently available tools and on-going research efforts.

1. INTRODUCTION
As the complexity of integrated circuits and the systems in which they are employed continues to rise, interfaces and interface abstractions are becoming increasingly important in the design process. It has become clear that without appropriate packaging of system blocks into reusable, composable, and customizable components we will not be able to manage the designs of the future in a cost-effective manner. This embedded tutorial will seek to highlight the principal issues in interface-based design and provide a roadmap to existing and emerging approaches for dealing with these issues.

2. WHAT ARE INTERFACES?
Interfaces exist at many levels and represent the behavior of a physical or virtual block as seen from outside of the block itself. At lower levels, interfaces are concerned primarily with physical quantities such as logic voltage levels, current sinking/sourcing capability, and capacitive load. At higher levels, they are concerned with more abstract behaviors such as communication between concurrent processes possibly running on separate processors. Table 1 provides a partial list levels, they are concerned with more abstract behaviors such as communication between concurrent processes possibly running on separate processors. Table 1 provides a partial list

<table>
<thead>
<tr>
<th>Interface Level</th>
<th>Concerns</th>
<th>Synthesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>electrical</td>
<td>logic levels, current,</td>
<td>transistors wires</td>
</tr>
<tr>
<td></td>
<td>load</td>
<td></td>
</tr>
<tr>
<td>logical</td>
<td>Boolean relationships</td>
<td>gates latches</td>
</tr>
<tr>
<td>sequencing</td>
<td>partial ordering of events</td>
<td>latches</td>
</tr>
<tr>
<td></td>
<td>spacing between events</td>
<td>FSMs delays</td>
</tr>
<tr>
<td>timing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>data transaction</td>
<td>word-level transfer of</td>
<td>FSMs RTL</td>
</tr>
<tr>
<td></td>
<td>data</td>
<td></td>
</tr>
<tr>
<td>packet</td>
<td>transfer of block of data</td>
<td>RTL device driver</td>
</tr>
<tr>
<td>message</td>
<td>inter-process communication</td>
<td>device driver</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OS calls</td>
</tr>
</tbody>
</table>

Table 1: Interface levels and their associated concerns and synthesis artifacts

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

© 1998 ACM 1-58113-008-2/98/0011...$5.00
of interface levels, the principal concerns of the designer at each of the levels, and the types of synthesis artifacts generated.

It is clear from the above list that there will need to be several languages for interface specification and a corresponding set of CAD tools to deal with simulation, synthesis, optimization, and verification. The artifacts that need to be synthesized at each level span the full range of implementation technologies from hardware to software with different levels in each. The hardware level ranges from gates to flip-flops to delay lines while the software level ranges from memory-mapped I/O to device drivers to operating system primitives. This tutorial provides an introduction to all of the areas from the logic level on up.

3. WHAT ARE COMPONENTS?
Interfaces are a crucial element in making component-based design a reality. Abstractions hide unimportant details at a particular level of design and allow design efforts to focus on the aspects that are most crucial at that level. This is important to human designers in managing complexity and to CAD tools in permitting effective synthesis and optimization algorithms. The challenge is how to describe the behavior fully while divulging as few details as possible of the internal implementation of the block.

There are two principal reasons for hiding the internal details of blocks. First, more details require descriptions at different levels and thus adversely affect the complexity of CAD tools. Second, requiring internal details to be visible dampens efforts to commercialize the intellectual property that is central to the advantages of component-based design. Thus, there is a constant trade-off between expressiveness (and re-design or optimization capability) and simplicity and protection (powerful CAD tools and safeguarding the IP that makes the block interesting). On the one hand, a designer would like to know everything about a component to be able to not only use it but also optimize it within its system context. In the limit, this reaches all the way to being able to change the component's behavior for enhanced performance. On the other hand, an IP provider does not want to provide the knowledge that went into making the component and wants to only disclose enough to make it useful to others.

4. HOW CAN CAD HELP?
Interface synthesis is a growing area for CAD application. Designs are becoming more component-based, are spanning more implementation technologies, and have a growing complexity that requires the support of appropriate tools. The challenge posed to CAD researchers is to develop the appropriate abstractions for describing interfaces at a variety of levels, defining the library mechanisms that will enable a wide distribution of IP blocks, and provide the tools that take advantage of the abstractions to allow designers to more easily compose and evaluate systems.

By component-based, it is important to understand that this is not limited simply to hardware blocks for ASICs. Rather, components now entail not only custom and IP hardware but also custom and IP software. The target architectures for systems range from fully off-the-shelf to systems-on-a-chip and often involve multiple processing elements in the form of standard microprocessors, microcontrollers, and DSP processors, as well as custom data-paths and application-specific programmable processors. All these pieces must be packaged up for IP distribution.

Standard composition methods will be key to making this possible. At the lower levels, these correspond to bus and communication standards such as various serial line and networking protocols (e.g., USB and Firewire) and on-chip busses to interconnect the components of systems-on-a-chip (e.g., VSI). At higher levels, we need to consider the interfaces between communicating processes and state machines. Data-flow methodologies have accomplished much in making DSP applications more easily composable. The same must now be done for control-oriented applications.

5. CONCLUSION
This embedded tutorial presents the various levels of interface abstractions and the concerns at each level. The CAD tools needed at each level and the advantages these confer to the designer are highlighted. Primary among these is the further facilitation of a truly component-based design methodology spanning all levels of embedded system design. In closing, we describe the challenges for CAD research in this area.

6. REFERENCES


