REPRESENTATIONS FOR REASONING ABOUT DIGITAL CIRCUITS

Tom M. Mitchell. Lou Steinberg. Reid G. Smith*,
Pet Schooley, Howard Jacobs", Van Kelly

Department of Computer Science
Rutgers University
New Brunswick, NJ 08903

ABSTRACT

We *are* interested in developing programs that reason about digital electronic circuits, in order to design, redesign, and debug them The first step toward developing such programs is to determine a useful way of representing the design and operation of circuits A useful representation must make apparent the roles of various circuit components in implementing the overall circuit function, and must allow a *program* to reason about the operation of the circuit at various levels of abstraction This paper summarizes our efforts to develop such a representation This work is closely related to other AI work on representing plans and on representing and reasoning about complex physical processes***

I INTRODUCTION

A variety of issues central to AI arise in considering the problem of reasoning about digital electronic circuits The practical significance of this problem is becoming increasingly clear as circuits of greater and greater complexity *are* mass produced at. low cost This paper considers the problem of *representing* the design and operation of digital circuits in a way that is useful for reasoning about their operation, design revision, debugging, testing, and maintenance

We report here on the elements of such a representation that is being developed as part of our longer term research toward automating the *functional redesign* of digital electronic circuits By functional redesign we mean the task of altering the design of an existing, well understood circuit, in order to meet a desired change to its functional specifications For example, consider the problem of redesigning a computer terminal in order to alter the number of characters displayed *per* line from 72 to 80. to alter the encoding of input characters from ASCII to EBCDIC, or to make the character font programmable rather than fixed

In order to reason about redesigning an existing circuit one must understand the basic operation of the circuit, as well as the relationship between the original design specifications *and* the existing circuit Thus, there are two major classes of representation problems in describing the design and operation of digital circuits:

Representing the basic operation of the circuit Representing the operation of a circuit over time requires describing the timing and synchronization of data-flow, as well as the function and interconnections of components. Circuit simulators, such as those typically reported in the Design Automation literature [1], represent circuit operation *m* a way that supports one kind of reasoning about circuit behavior; i.e., answering the question "What is the output of the circuit for a given input?'. We intend our representation of circuit operation to be useful for answering additional" questions such as 'Will two given signals always be high at the same time?', or "Whet change to the input of a given component will cause a desired change to its output?". We have developed a declarative representation for the behavior of signals over time and the synchronization of events, along with a representation of functions of circuit modules, that is more suited to answering such questions. Previous work in AI on representing and reasoning about complex processes includes, for example, [2, 3, 4]

Representing the design plan In addition to representing the basic operation of a circuit we must also represent the relationship between the design specification *and* the implemented circuit in a way that allows answering questions such as "What is the *role* of a given circuit component in implementing the overall design specifications?', end "How is a desired function decomposed into subfunctions, and implemented by a variety of circuit modules?" We represent the relationship between the design specifications and the implemented circuit in terms of a hierarchical *plan* for the design of the circuit This plan consists of several representations of portions of the circuit at different levels of abstraction, along with various kinds of links between the levels. Relevant work on representing hierarchical plans in domains other than electronics includes [5, 6, 7].

II OVERVIEW OF THE REPRESENTATION

Our circuit representation involves describing a given pircuit at several levels of abstraction, ranging from a very high level functional descriptions down to its realization *in* TTL level components ie.g flip flops, logic gates, etc.). A uniform representation scheme is used for any such level of description The relationship between different levels of description is characterized in terms of general rules of circuit design The following sections highlight the major notions underlying our representation

*With Defence Research Establishment Atlantic, Box 1012, Dartmouth, NS, Canada, B2Y 3Z7.

**With Bell Telephone Laboratories, HP 1C-110, Holmdel. NJ, 07733.

***This research is supported by the Defense Advanced Research Projects Agency under research contract N00014-81-K-0394

A. REPRESENTING THE STRUCTURE AND OPERATION OF THE CIRCUIT

Tha circuit is described in terms of four entities (1) circuit *modules*. (2) their associated *functions,* (3) *data-paths* between modules (eg, wires or busses), and (4) *data-streams* (the sequence of voltages or higher level data flowing on a particular data-path) Each abstraction of the circuit corresponds to a collection of these four entities A number of modules can be grouped together and viewed as a single module Similarly, a number of data-paths can be grouped together *and* viewed as a single data-path An abstraction of a function corresponds to a partial definition of that function Data-streams of one type (eg, a stream of bits) can be viewed *as* representing data-streams of another type (e.g.. a stream of characters)

Our representation of the flow of data within the circuit is based on a declarative characterization of the sequence of data on a given data-path For instance, we can explicitly represent the statement that, on some data path D, a new ASCII character appears every 10 milliseconds, synchronized with clock C This allows answering such questions as, "Why is latch L1 clocked by clock C?" or, "What is the minimum setup time ROM R1 will ever have to handle?"

Since module functions relate input data-streams to output data-streams, their representation is closely tied to the data representation In particular, function descriptions specify the times at which elements in the output data-stream begin, along with their data value and their time duration. These are given as a function of the value, start time, and duration of elements in the input data-streams Function descriptions can often be simplified greatly once the structure of input data-streams is known (eg once it is known that a particular input is a periodic sequence of characters) A module is actually described by two functions, one which describes the modules actual behavior and one which describes what the module must do for the circuit to work. Usually, the former will be a more detailed version of the latter.

B. REPRESENTING THE DESIGN PLAN

The designed circuit is represented in terms of a hierarchical plan for implementing the design specifications We do not require that this design plan reflect the process by which the design was actually created, only that the plan explain the design as it exists The topmost level of the hierarchy corresponds to a high level specification for the design; the bottom most level corresponds to the implemented circuit. Each intermediate node in the hierarchy corresponds to an abstraction of some portion of the circuit structure and function that represents some set of commitments about the circuit implementation Each such abstraction is consistent with the desired functional specifications of its ancestor, and may itself contain abstract circuit modules whose implementation is not yet specified

Successive abstractions in the hierarchy are tied together by rules that characterize circuit-independent knowledge about implementing modules and about encoding data For example, one such rule states that in order to convert a pacaltel bit-string into a aerial bit-string, one can use a shift register in a specified configuration. Each rule application represents some commitment regarding the circuit implementation, and leads to a leas abstract circuit description For any component, it should be possible to trace back through the rules that lead to it, in order to characterize that components role in implementing the specifications

III CURRENT STATUS

We are currently debugging our representation by using it to describe portions of the video display controller for a computer terminal Our representation system is embedded within UNITs [83. a general-purpose frame-based representation system A detailed description of our representation and its use to describe a specific circuit is available in [9] We also suggest there how this representation could be used in support of a system for automated functional redesign

IV ACKNOWLEDGMENTS

Other members of the Digital Design Project include Saul Amarel. Giles Lafue. Saul Levy. Jeff Shulman. and Tim Weinrich John Grason and Larry O'Neill have provided valuable suggestions and criticisms

REFERENCES

[13 *17th Design Automation Conference Proceedings.* IEEE. Minneapolis, Minnesota. June, 1980

[23 C Rieger and M. Grinberg, A System Of Cause-Effect Representation And Simulation For Computer-Aided Design," in *Artificial Intelligence And Pattern Recognition In Computer Aided Design.* J C Latombe. ed. North-Holland Publishing Company. 1978, 299-326

[33 G J Sussman. At The Boundary Between Analysis And Synthesis,' in *Artificial Intelligence And Pattern Recognition In Computer Aided Design,* J C Latombe. ed. North-Holland Publishing Company 1978. 261-290

[43 J. S. Brown and R. Burton, "Multiple Representations Of Knowledge For Tutorial Reasoning/ in *Representation And Understanding: Studies in Cognitive Science,* Academic Press. 1975

[53 M J Stefik. *Planning With Constraints,* PhD dissertation, Stanford University. 1980, Also available as a Computer Science Dept technical report, number STAN-CS-80-784

[63 E Sacerdoti *A Structure For Plans And Behavior.* Elsevier North-Holland. 1977.

[73 C Rich and H Shrobe "Initial Report On A LISP Programmer s Apprentice". Technical report TR-354, MIT. 1976

[83 R G Smith and P. Fnedland "Unit Package User's Guide". Technical Memorandum 80/L Defence Research Establishment Atlantic, Dec 1980, Also Stanford Heuristic Programming Project Memo *HPP-80-28*

[93 T. M. Mitchell. L Steinberg, R. G Smith, P Schooiey. H Jacobs, V Kelly, "Representations For Reasoning About Digital Designs", Forthcoming