An Intelligent Aid for Circuit Redesign

Tom M. Mitchell, Louis I. Steinberg, Smadar Kedar-Cabelli
Van E. Kelly, Jeffrey Shulman, Timothy Weinrich

Department of Computer Science
Rutgers University
New Brunswick, NJ 08903

Abstract

Digital circuit redesign is a task that requires knowledge of circuit structure, function, and purpose, and of the interrelationships among these. We describe a knowledge-based system, REDESIGN, which assists in the redesign of digital circuits to meet altered functional specifications. REDESIGN assists the user in focusing on an appropriate portion of the circuit, generating possible local changes within the circuit, ranking these possible changes, and detecting undesirable side-effects of redesigns. It provides this assistance by combining two modes of reasoning about circuits: (1) causal reasoning involving analysis of circuit operation, and (2) reasoning about the purposes, or roles, of various circuit modules within the larger circuit. We describe these two modes of reasoning, and the way in which they are combined by REDESIGN to provide aid in circuit redesign.

I Introduction

The AI/VLSI group at Rutgers is exploring Artificial Intelligence approaches to a new generation of design and debugging aids for digital circuits. This work has led to a general exploration of methods for representing and reasoning about complex artifacts, and about the interrelationships among their purpose, function, and structure. Over the past few years we have developed a prototype intelligent assistant (called REDESIGN) for the functional redesign of digital TTL circuits. This paper reports on the REDESIGN system, its capabilities for representing and reasoning about digital circuits, and its use of these capabilities to assist in circuit redesign.

A. The Problem

In the functional redesign problem the system is given the schematic of a working digital circuit (e.g., a computer terminal), and its functional specifications (e.g, the fact that it displays 80 characters per line, 25 lines per screen, displays the cursor at a programmable address, etc.). The system is also given a data structure called a *design plan*, which relates the circuit schematic to its specifications. Given a desired change to the functional specifications (e.g., require that the terminal display 72 characters per line), the task is then to redesign the circuit so that it will meet these altered specifications.

*This material is based on work supported by the Defense Advanced Research Projects Agency under Research Contract N00014-81-K-0394. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

The formulation of the design problem presented here is very similar to planning problems in the AI literature, and the issues addressed in this work are related to those addressed by others working in the areas of planning and design, such as [2, 4, 6, 7, 8, 9, 10]. Our work is also related to that of [1], which deals with recognizing circuits rather than designing them, and which addresses the relations among circuit function, structure, and purpose.

Design and redesign are closely related problems. In addressing redesign rather than design, we have focused more on how to describe usefully the interconnected subgoals and constraints which characterize the solution to a design problem, and less on control of search in design. One view of functional redesign is that it is a form of analogical problem solving. In particular, the original circuit provides a solution to the problem of implementing the original specifications. Given a new set of specifications which is nearly identical, the hope is to implement these specifications in a closely analogous fashion (i.e., by making only minor changes to the original circuit schematic). The key to using the analogy effectively lies largely in having recorded the essentials of the original solution in a fashion that will allow determining which portions can be reused, and which changes will lead to undesirable interactions among subgoal solutions.

The next section discusses the representation of circuits, and the notions of circuit behavior and specifications. The subsequent section describes the two modes of reasoning about circuits employed by REDESIGN: causal reasoning and reasoning about purpose. We then illustrate the use of these modes of reasoning by REDESIGN, by tracing its use for a specific redesign problem.

II Representing Circuits, Behaviors and Specifications

The structure of a circuit is represented by a network of *modules* and *data-paths*. A module represents either a single component or a cluster of components being viewed as a single functional block. Similarly, a data-path represents either a wire or a group of wires. The data flowing on a data-path is represented by a *data-stream*, and the operation performed by a module is represented by a *module function*. These representations are described in [3, 5]. One aspect of this circuit representation that has been important in REDESIGN is that data-streams represent the entire time history of data values on a data-path, rather than a single value at a single time, as in many circuit simulators. This has proven to allow considerable flexibility in reasoning about circuit behavior over time.

In reasoning about redesign, REDESIGN must distinguish between what happens to be true of the circuit (we refer to this as the circuit *behavior*), and what must be true for that circuit to work correctly (we refer to this as the circuit *specifications*). Therefore, for each module function and datastream, both behavior and specifications are recorded. For example, the behavior of a particular module may state that its output will be the sum of its inputs, delayed by 100 nanoseconds, while the specifications for that module may simply require that the output be delayed by less that 500 nanoseconds.

While these definitions look straightforward, the notion of a specification must be defined more precisely. It is useful to think of the specification of a module as giving the *range* within which the behavior of that module can be altered without making the circuit as a whole malfunction. However, the range of acceptable behaviors depends upon what else in the circuit is allowed to change. One could define a module's specifications as the range of acceptable behaviors, assuming the *specifications* of all other modules remain unchanged, but allowing other modules to have any behavior within their own specifications. We term this kind of specification an *s-specification*. Or, one could define a module's specification as the range of acceptable behaviors, assuming every other *behavior* in the circuit remains fixed. We term this kind of specification a *b-specification*.

Notice that the s-specifications of a module will always be at least as restrictive as its b-specifications (since s-specifications are based on weaker assumptions regarding the surrounding circuitry). Thus, it is possible for a component to violate its s-specifications (but not its b-specifications), and for the circuit as a whole to still operate correctly. While top down design systems typically must deal with s-specifications (since b-specifications are not defined until the final circuit implementation is known), in REDESIGN we have found it most useful to record b-specifications. This is because in considering possible changes to an individual module within a completed design we make the default assumption that the rest of the circuit, and hence the rest of the behaviors, will remain unchanged. Of course, if changes are made in two modules then one must keep track of how changes in each one affect the b-specifications of the other.

III Two Modes of Reasoning about Circuits

A variety of types of questions arise when redesigning a circuit. REDESIGN uses two separate modes of reasoning to answer these questions -- one to analyze circuit operation based on a causal model of the circuit, and one to reason about the purposes of circuit submodules (i.e. their roles in implementing the global circuit specifications). These two modes of reasoning are combined to provide assistance at various stages of the redesign process.

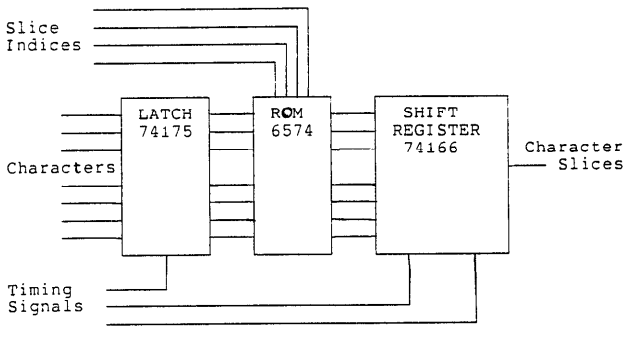

Figure III-1: The Character Generator Module

A. Causal Reasoning

Causal reasoning answers questions such as "If input X is supplied to the circuit module, what will the output be?" and "If output Y is desired, what must be provided as inputs to the module?", where X and Y are complete descriptions of input and output datastreams. It also answers questions of the form, "Given only the start-time of the input (or output), what can be determined about the output (or input)?" The first of these questions, where a completely described input is given, is the type of question answered by standard circuit simulators. But redesigning a circuit requires answering the other kinds of questions as well. For example, if the circuit specifications call for the circuit output to have a certain duration, it is important to be able to determine which properties of the upstream signals will assure this duration. The collection of routines that provides answers to these kinds of questions about circuit operation is called CRITTER [3]. CRITTER propagates full and partial descriptions of data-streams through the circuit, and can test whether a given data-stream specification is satisfied by its behavior. It also maintains a *Dependency Network* that records, for each specification, both its source and the path in the circuit through which it was propagated. See [3, 5] for more information on CRITTER.

B. Reasoning about Purpose

A second kind of reasoning important in redesign concerns the roles, or purposes, of various circuit modules in implementing the overall circuit specifications. Questions of this sort that arise during redesign include "What is the purpose of circuit module M?" and "How are the circuit specifications decomposed into subspecifications to be implemented by separate sections of the hardware?". Questions of this sort can be answered by REDESIGN, by examining the *Design Plan* of the circuit.

The Design Plan is a data structure that shows how circuit specifications are decomposed and implemented in the circuit, as well as the conflicts and subgoals that arise during design. It contains enough information to allow "replaying" the original design, and is characterized in terms of a set of *implementation rules* that embody in executable form general knowledge about circuit design tactics. This Design Plan must be provided to REDESIGN, as part of the characterization of the circuit which is to be redesigned.

In order to illustrate the form of the Design Plan, consider the simple Character Generator Module (CGM) circuit shown in figure 3-1. This circuit is similar to a standard circuit used in most video computer terminals -- it is the part of the terminal that translates the ASCII character codes into the corresponding dot matrix to be displayed on the screen. This circuit accepts as input (1) a stream of ASCII encoded *Characters*, (2) a stream of binary encoded integers, called *Slice-Indices* that specify which horizontal slice of the character dot matrix is to be displayed, and (3) several clock signals used for synchronization. The circuit must produce a stream of *Character-Slices*, each of which is a bit string corresponding to the dots to be displayed on the terminal screen for the selected horizontal slice of the input Character.

The heart of the CGM design is a read-only memory, the ROM6574. This ROM6574 stores the definition of the character font (the dot matrix to be displayed for each character), one Character-Slice per byte of memory. To retrieve the Character-Slice corresponding to a given Character and Slice-Index, the ASCII code for the character is concatenated with the binary representation of the Slice-Index, and used to address the ROM6574. The other components in this circuit are used to interface the ROM6574 to the desired input and output formats. For example, the CGM specifications require serial output while ROMs produce parallel output. Therefore, a shift register (SHIFT-REGISTER-74166) is used to convert the output data to serial. Also, because the address inputs to the ROM6475 must be stable for at least 500 nsec. while the input Characters are stable for only 300 nsec., a latch (LATCH74175) is used to capture the input Characters, and hold these data values stable for an acceptable duration.

The above paragraph summarizes the purpose of each circuit component and the conflicts and subgoals that appear during design. *This is precisely the kind of summary that must be captured in the Design Plan, in order to allow the REDESIGN program to reason effectively about the design and about the purposes of individual circuit components.*

Figure III-2 illustrates the Design Plan used to describe the CGM circuit to REDESIGN. Each node in the Design Plan corresponds to some abstracted circuit module whose implementation is described by the hierarchy below it. The topmost node in this Design Plan represents the entire CGM, and its functional specifications. The bottom most nodes in the Design Plan represent individual components in the circuit. Each solid vertical link between modules in the Design Plan corresponds to some implementation choice in the design, and is associated with some general implementation rule which, when executed, could recreate this implementation step. For example, the vertical link leading down from the topmost module in the figure represents the decision to use a Read-Only Memory (ROM) to implement the CGM. This implementation choice is associated with the implementation rule which states "IF the goal is to implement some finite mapping between input and output data values, then use a ROM whose contents store the desired mapping" (note this leaves open the choice of the exact type of ROM.)

Each dashed link in the Design Plan represents a conflict arising from some implementation choice or choices, and leads to a design subgoal, represented by a new circuit module with appropriate specifications. For example, a conflict follows from the implementation choice to use a ROM, and leads to the subgoal module labelled "Parallel-to-Serial-Subgoal". The conflict in this case is the discrepancy between the known output signal format of ROMs (i.e., parallel) and the required output signal format of the CGM (i.e., serial). The specifications of the new subgoal module are therefore to convert the parallel signal to serial. In a similar fashion, the implementation choice to use the specific ROM6574 leads to another conflict, and to the resulting subgoal to extend the duration of the input data elements.

By examining the Design Plan of a circuit, REDESIGN is able to reason about purposes of various circuit modules, and about the way in which the circuit specifications are implemented. The general implementation rules used to summarize the design choices can be used to "replay" the Design Plan for the similar circuit specifications, and thus allow for a straightforward kind of design by analogy.

IV Redesigning a Circuit

This section illustrates the use of both causal reasoning and reasoning about purpose in redesigning a circuit. It traces the actions of the REDESIGN program as it took part in a particular redesign of the Video Output Circuit (VOC) of a computer terminal. The Video Output Circuit (which contains the Character Generator Module discussed earlier) is shown in figure 4-1. It is the part of the computer terminal that produces the composite video information to be displayed on the terminal screen. It produces this output from its combined inputs, which include the characters to be displayed, the cursor position, synchronization information for blanking the perimeter of the terminal screen, and special display commands (e.g., to blink a particular character).

In this example, we consider redesigning the VOC to display characters in an italics font rather than its current font. Given a redesign problem, REDESIGN guides the user through the following sequence of five subtasks: (1) focus on an appropriate portion of the circuit, (2) generate redesign options to the level of proposed specifications for individual modules, (3) rank the generated options, (4) implement the selected redesign option, and (5) detect and repair side effects resulting from the redesign. A more complete trace and discussion of this example is given in [5].

Focus attention on appropriate section(s) of the circuit. In many cases, the most difficult step in functional redesign is determining which portions of the circuit should be ignored. Focusing on relevant details in one locality of the circuit while ignoring irrelevant details in other localities can greatly simplify the complexity of redesign. In order to determine an appropriate focus, REDESIGN "replays" the Design Plan by reinvoking the recorded implementation rules with the changed circuit specifications. During this replay process, whenever an abstract circuit module is produced by some implementation step, its purpose is compared with the purpose of the corresponding module in the original Design Plan. If the purpose is unchanged, then the original implementation of this module will be reused without

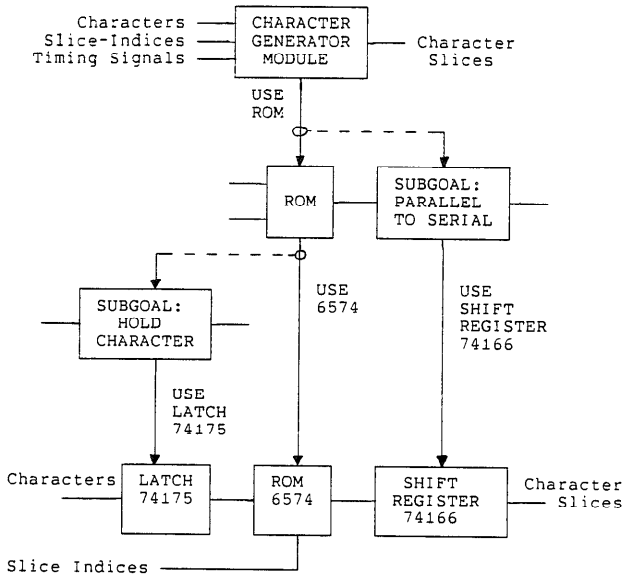

Figure III-2: Design Plan for the CGM

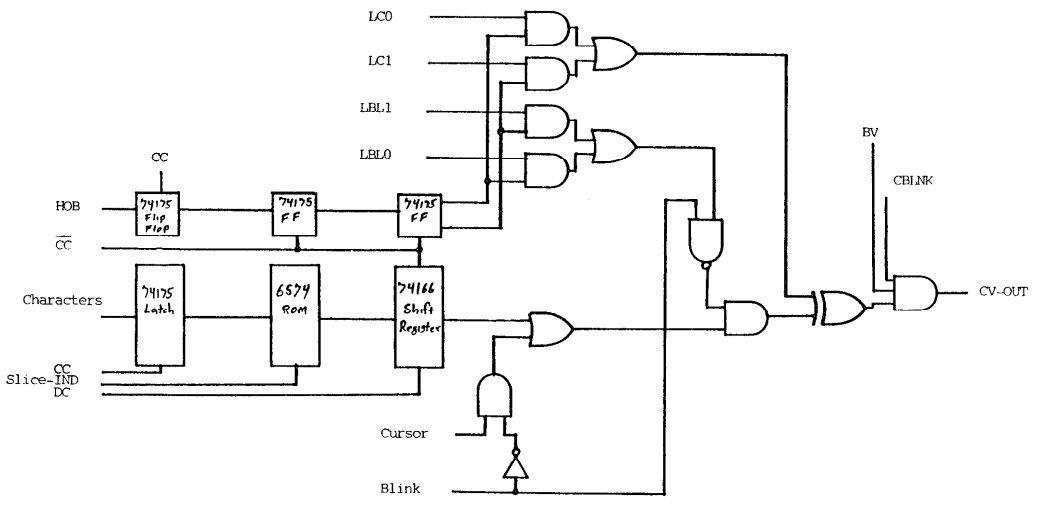

Figure IV-1: Video Output Circuit

change in the new design**. If the new module has a different purpose than the corresponding module in the old Design Plan, (e.g., the new CGM must implement a different character font), an attempt is still made to apply the same implementation rule as in the original design (e.g., still try to use a ROM). If this implementation rule is not useful in the new design (as with the rule that suggests using the specific ROM6574), then REDESIGN stops expanding this portion of the Design Plan, and marks the corresponding portion of the circuit as a portion to be focused on for further redesign. The use of the Design Plan as sketched above leads in the current example to a focus on redesigning the abstract ROM module within the CGM within the VOC circuit. This abstract ROM module is implemented in the current circuit by two components as shown in figure III-2 (the ROM6574 and LATCH74175). A second method of focusing is possible, by using the Dependency Network produced by CRITTER. This method involves isolating those points in the circuit that possess specifications derived from the changed specification on the output datastream. The resulting focus is generally broader than that determined from the Design Plan, because out of the many places in the circuit that can impact any given output specification, only a small proportion of these involve circuitry whose main purpose is to implement that specification.

Generate redesign options to the level of proposed specifications for individual circuit modules. Once an initial focus for the redesign has been determined, redesign options are generated which recommend either altering the specifications of individual modules, or adding new modules with stated specifications. In both cases, only the new functional specifications are determined at this point -- the circuitry to implement these specifications is determined later. The constraint propagation capabilities of CRITTER provide the basis for generating these redesign options. In the current example, once REDESIGN has focused on the section of the VOC including the ROM6574 and LATCH74175, it considers the new output specification for this circuit segment, and propagates it back through this segment. Before each propagation step, REDESIGN considers the option of breaking the wire at that point and inserting a module to transform the values on that wire to values satisfying the required specification. In addition, it considers the option of altering the module immediately upstream, so that it will provide the required signal at that point. For each of the generated options, the new functional specifications are defined in terms of (1) the new specification to be achieved, and (2) a list of unchanged specifications found in the original Dependency Network, which are to be maintained. In the current example, the option generation process produces a list of five candidate redesign options. This list includes redesign options such as "replace the ROM6574 by a module which stores the new character font", and "introduce a new module at the output of the ROM6574, which will transform the output values into the desired font" (these options are described by the program in a formal notation, and the above are only English summaries).

Rank the generated redesign options. Heuristics for ranking redesign options can be based on a variety of concerns: (1) the estimated difficulty of implementing the redesign option (e.g., components with zero delay cannot be built), (2) the likely impact of the implemented redesign on global criteria such as power consumption and layout area, and (3) the likelihood and severity of side effects that might be associated with the redesign***. In the current example, the heuristic that selects the appropriate redesign option suggests "Favor those redesign options that replace existing modules whose purpose has changed." In this case, since the purpose of the ROM6574 has changed, the option of replacing this component is recommended. The recorded Dependency Network and Design Plan also provide very useful information for estimating the relative

---

**One must still make certain that changes elsewhere in the design do not interact dangerously with the implementation of this module. In REDESIGN, this is accomplished without having to directly examining the implementation of the module. Instead, design changes elsewhere in the circuit are checked for consistency with the constraints recorded in the Dependency Network produced by CRITTER.

***The current REDESIGN system has only a primitive set of heuristics for ranking redesign options.

severity of various changes to the circuit. Because the Design Plan shows the dependencies among implementation decisions (e.g., the purpose for the LATCH74175 is derived from the decision to use the specific ROM74175) it provides a basis for ordering the importance of components and associated constraints in the overall design (e.g., if the ROM6574 is removed, the LATCH75174 may no longer have a purpose for existing). This ordering of circuit modules, and of the datastream constraints that they impose, provides an important basis for estimating the relative extent of side effects associated with their change.

Implement the selected redesign option. The above steps translate the original redesign request into some set of more local (and hopefully simpler) specification changes. While the implementation rules that REDESIGN possesses can be used for design****, we have not focused on automating this step. Thus, the user is left to implement the redesign option.

Detect and Repair Side Effects Arising from the Redesign Once the redesigned circuit is produced, REDESIGN checks the new circuit segment to try to determine (a) that it does achieve the desired new purpose, and (b) that it does not lead to undesirable side effects. Undesirable side effects are detected as violations of the Dependency Network specifications at the inputs and outputs of the altered circuit segment. If a specification is violated, the new circuitry might be redesigned, or the specification might itself be modified or removed by redesigning a different portion of the circuit. The Dependency Network can be examined to determine the source of the violated specification, and to determine the locus of circuit points at which the specification could be altered.

V Summary

REDESIGN is a research prototype system that demonstrates the feasibility of providing intelligent aids for redesign and design of digital circuits. It provides aid in focusing attention on an appropriate portion of the circuit, in generating and ranking redesign options, and in monitoring and manipulating the many constraints involved in making a design work. While the current REDESIGN system has many limitations (e.g., in the size of circuits it can handle, its inability to help with certain classes of redesigns, shortcomings of its causal reasoning methods, incompleteness of its knowledge base of implementation rules, etc.) the basic representations and approaches to reasoning appear useful.

Several aspects of our approach have contributed to the success of REDESIGN. The most apparent of these is the combined use of reasoning about *causality* in the circuit, and reasoning about the *purposes* of parts of the circuit. There are also some important aspects to how REDESIGN reasons about causality and purpose. In reasoning about causality, REDESIGN describes both the behavior and the specifications for a data stream, in a way that allows it to describe entire histories, not just data stream values at particular time instants. REDESIGN can propagate these descriptions through the circuit, to build a Dependency Network showing how the specifications for each data stream are derived from the behaviors of the modules and the specifications for the circuit as a whole. In reasoning about purposes, we have viewed the original design process essentially as a planning problem, with subgoals derived both from the decomposition of parent goals and from conflicts between other subgoals. The Design Plan provides REDESIGN with an explicit summary of this planning process, with detail enough to replay the process, and to examine the particular relationships among design goals and subgoals.

****We have recently begun an effort to build a VLSI Design Consultant system which uses similar rules for automated design.

References

[1] de Kleer, Johan, *Casual And Teleological Reasoning In Circuit Recognition,* Ph.D. dissertation, Massachusetts Institute Technology, January 1979.

[2] Green, C., et al., "Research on Knowledge-Based Programming and Algorithm Design", Research Report KES.U.81.2, Kestrel Institute, September 1982.

[3] Kelly, V., Steinberg, L., "The CRITTER System: Analyzing Digital Circuits by Propagating Behaviors and Specifications," *Proceedings of the National Conference on Artificial Intelligence,* August 1982, pp. 284-289, Also Rutgers Computer Science Department Technical Report LCSR-TR-30, and Re-Design Project Working Paper #6.

[4] J. McDermott, "Domain Knowledge and the Design Process," *Proceedings of the 18th Design Automation Conference,* IEEE, Nashville, 1981.

[5] Mitchell, T., Steinberg, L., Kedar-Cabelli, S., Kelly, V., Shulman, J., and Weinrich, T., "REDESIGN: A Knowledge-Based System for Circuit Redesign", Technical Report DCS-TR, Rutgers Univ., April 1983.

[6] Mostow, D.J., and Lam, M., "Transformational VLSI Design: A Progress Report", Technical Report, USC-ISI, November 1982.

[7] Rich, Charles; Shrobe, Howard E.; Waters, Richard C., "Computer Aided Evolutionary Design For Software Engineering", AI Memo 506, Massachusetts Institute Technology, January 1979.

[8] Stefik, Mark Jeffrey, *Planning With Constraints,* Ph.D. dissertation, Stanford University, January 1980.

[9] Sussman, Gerald Jay; Holloway, Jack; Knight, Jr., Thomas F., "Computer Aided Evolutionary Design For Digital Integrated Systems", AI Memo 526, Massachusetts Institute Technology, May 1979.

[10] Wile, David S., "Program Developments as Formal Objects", Technical Report, Information Sciences Institute, July 1981.