# A KNOWLEDGE BASED DESIGN SYSTEM FOR DIGITAL ELECTRONICS*

Milton R. Grinberg

Department of Computer Science
University of Maryland
College Park, Maryland 20742

## 1. Overview and Goals of the SADD System

In digital design, the goal of a human expert is to translate a goal, such as "build a digital alarm clock", or "build a video display interface" into a working digital circuit. This is a problem solving activity in which the expert's general purpose problem solving abilities interact with his rich knowledge of the digital world. By translating the initial ideas through successively more refined "sketches", the human expert gradually arrives at a chip-level digital schematic that realizes the initial goals, and which may have bugs that will only be discovered after the circuit has been simulated or built and tested.

The Semi-Automatic Digital Designer (SADD) is an experimental interactive knowledge-based design system, whose domain of expertise is digital electronics. The SADD project has two goals. First, I want to provide an intermediate digital design problem solver for which a human expert can interactively provide the high-level functional descriptions of a circuit, and which can take this high-level circuit description and refine it into a circuit schematic that performs the required task. Second, I am attempting to discover and express in a computer model that knowledge that makes the human designer an expert in digital design. This includes the generic information for each high-level digital function (e.g., counter, clock) and the methods needed to transform these functional descriptions into realizable circuits. As a first case study in design, I adopted a relatively sophisticated TV video display circuit called the Screensplitter. This is a real circuit of moderate overall complexity.

## 2. Digital Design

How does a digital designer go about the task of designing a circuit? The designer generally starts with a well-defined goal for a circuit with only an ill-defined solution for accomplishing that goal. Using his design experience, the designer can pinpoint some of the required and hence pivotal components needed in the circuit. The circuit is slowly sketched around these pivotal components. The sketching process allows the designer to specify where the inputs come from and where the outputs go. The designer often makes notes concerning characteristics of the components. During this sketch phase, the expert discovers that other components are needed and adds them to the design. Eventually the designer starts to refine each component by selecting and interconnecting chips to implement the component.

What are the "primitives" that a designer uses? At the implementation (i.e., final circuit) level there are three "primitives": (1) a chip and its input and output pins, (2) a wire, and (3) a signal. The chip is a function whose outputs are defined by its current state and its current inputs. A wire is a physical connection among pins, defining an electrical equivalence. A signal is the electrical information present on a wire.

At the sketchpad level, there are two additional "primitives": (1) a function and (2) a connection. A function performs some designated digital task. I have concentrated on nine different functions (i.e., counter, shifter, memory, combinational, switch, register, divider, selector, and clock) that are required in the Screensplitter. These are the pivotal components that a designer uses in the sketch phase. A connection is an information path between two functions. It identifies where information flows and when it is allowed to flow.

## 3. SADD Organization

There are three distinct design phases in the SADD system:

(1) specification acquistion
(2) circuit design
(3) circuit simulation

In the specification acquistion phase, the user describes the functional structure of the circuit in English. During this phase, frames are introduced, relevant information about the frames is filled in, and the interrelationship among the frames established. From this description, SADD builds a model of the circuit, expressed as a semantic net.

In the circuit design phase, the conceptual description is used to construct a circuit schematic. This is accomplished in two steps. First, an implementation strategy for each component is selected using the conceptual function characteristics (which may have to be deduced from the characteristics provided by the designer and the functions relationship to the other functions in the model). Then the circuit schematic is implemented using the selected strategy and conceptual function description.

In the circuit simulation phase, the correctness of the circuit is determined by simulating the circuit on a conceptual simulator. If the designed circuit proves not to fulfill the goal of the designer during the simulation phase, it can be modified and redesigned.

### 3.1. Screensplitter Circuit

In order to develop SADD, the Screensplitter was chosen as a benchmark. Fig. 1 illustrates one of the 12 logical components (the Display Character

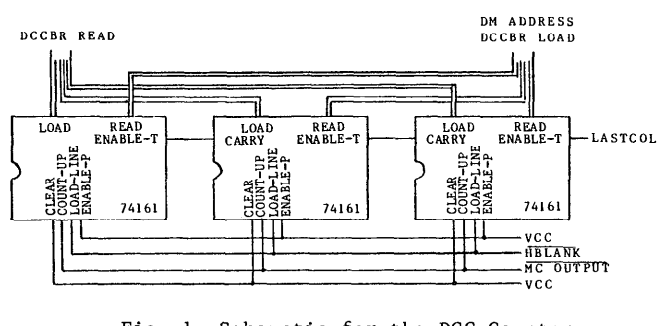


Fig. 1. Schematic for the DCC Counter

*This research is supported by the Office of Naval Research under Grant N00014-76C-0477.

Counter) from the original Screensplitter circuit schematic. Using the circuit schematic, a verbal design scenerio was developed that describes the functional components, characteristics of these functional components and the interconnections. Fig. 2 shows the relevant portion of the input

1. THE DISPLAY CHARACTER COUNTER (DCC) COUNTS FROM 0 TO 3519.
2. WHEN THE COUNT OF THE SLC EQUALS 4, THE COUNT OF THE DCC CAPTURED-IN THE LOAD OF THE DCC-BASE REGISTER (DCCBR) WHEN THE HORIZONTAL BLANK (HBLANK) BEGINS.
3. THE DCC CAPTURED-FROM THE DCC-BASE REGISTER (DCCBR) WHEN HORIZONTAL BLANK (HBLANK) BEGINS.
4. EACH-TIME THE COUNT OF THE PIXEL COUNTER (PC) EQUALS 5 ENDS, THE DCC INCREMENTS.
5. THE COUNT OF THE DCC CAPTURED-IN THE ADDRESS OF THE DISPLAY MEMORY (DM).

Fig. 2. Scenerio for the DCC Counter

description for the Display Character Counter. The complete scenerio is 41 sentences.

### 3.2. Parser

A phrase-keyword parser using procedurally encoded case-frameworks for the verbs was developed to interpret the input. Phrase-keyword means that the parser is always trying to build up one of seven types of phrases that are common in the digital design world. It uses the keywords to help identify the beginnings and endings of these phrases.

After a sentence has been parsed into phrases, the procedure associated with the verb in the sentence is applied. This processing first verifies that the sentence is semantically acceptable (that the phrase types are legitimate for the verb). Then those circuit objects not currently in the circuit model are introduced into the model, and the verb's manipulations of the model are processed.

The parser uses 5 directives to manipulate the model. These directives either add new information to the model or interrelate existing parts of the model. The directives are:

(a) Specify a function - a function (e.g., clock, counter) is introduced into the model.
(b) Assign a value to a function's aspect - one of the function's characteristics is assigned a value.
(c) Define a conceptual signal - a global name is assumed to be a port of either a known or unknown function.
(d) Define the source of a conceptual signal - the source function of a global signal is identified.
(e) Make a connection - identify an information path between two functions and a condition gating the information flow.

### 3.3. Example - Specification Acquisition

To illustrate the specification acquistion phase, the five sentences shown above for the Display Character Counter are reduced to their effects on the model. The effects for each sentence are preceded by a letter which references the corresponding directive type from the above list of directives.

1. THE DISPLAY CHARACTER COUNTER (DCC) COUNTS FROM 0 TO 3519.
   (a) Introduce a COUNTER function named DISPLAY CHARACTER COUNTER (DCC).
   (b) Assign a value to the Aspect COUNT-SEQUENCE of (0 3519).

2. WHEN THE COUNT OF THE SLC EQUALS 4, THE COUNT OF THE DCC CAPTURED-IN THE LOAD OF THE DCC-BASE REGISTER (DCCBR) WHEN THE HORIZONTAL BLANK (HBLANK) BEGINS.
   (c) Define a conceptual signal named HORIZONTAL BLANK (HBLANK).
   (e) Make a connection between (DCCBR LOAD) and (DCC COUNT) under the condition (AND (HIGH SLC DS4) (RISING UNKNOWN HBLANK)).

3. THE DCC CAPTURED-FROM THE DCC-BASE REGISTER (DCCBR) WHEN HORIZONTAL BLANK (HBLANK) BEGINS.
   (e) Make a connection between (DCC LOAD) and (DCCBR READ) under the condition (RISING UNKNOWN HBLANK).

4. EACH-TIME THE COUNT OF THE PIXEL COUNTER (PC) EQUALS 5 ENDS, THE DCC INCREMENTS.
   (a) Introduce COUNTER function named PIXEL COUNTER (PC).
   (e) Make a connection between (DCC COUNT-UP) and T under the condition (FALLING PC DS5).

5. THE COUNT OF THE DCC CAPTURED-IN THE ADDRESS OF THE DISPLAY MEMORY (DM).
   (a) Introduce MEMORY function named DISPLAY MEMORY (DM).
   (e) Make a connection between (DM ADDRESS) and (DCC COUNT).

The model of the circuit after just these five sentences are processed is shown in Fig. 3. After

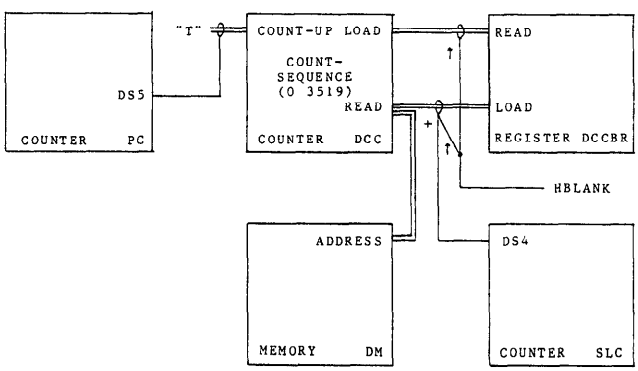


Fig. 3. State of the Model - (1, 2, 3, 4, and 5)

this description, the DCC has been completely specified and its implementation is discussed in the next section.

### 3.4. Function

Each function type has an associated frame structure that provides the prototypical knowledge about that function. There are two components to the function frame: the ASPECT and the CONPT. The ASPECT identifies the important characteristics of the function which might be mentioned by the user in the input scenerio. The CONPT identifies the ports (the input and output lines) that are associated with the function. The width of the CONPT is also identified as either 1 or the value of one of the ASPECTs of the function. When a particular function is introduced into the model, a copy of the prototype is copied into the database and instantiated with the name of the function. Fig. 4 is the instantiation of the DCC counter as represented in the model after sentences 1, 2, 3, 4, and 5 have been processed. The asterisked items are those that were modified during the sentence processing.

The expert has a conceptual view of the component being described. The function frame with the associated filled-in aspect values represents that conceptual view the expert has of the component.

## 4. Designing a Circuit for a Function

There are two phases involved in the implementation of a function. First a method for implementing the function must be selected. Then the selected method must be processed to produce the chip-level design.

```
 (COMPONENT DCC SADD COUNTER)
 (TYPE DCC DEVICE (DISPLAY CHARACTER COUNTER))
*(DESC-VAL WIDTH DCC IO-2)
*(DESC-VAL COUNT-SEQUENCE DCC (0 3519))
 (ASPECT WIDTH DCC NIL)
 (ASPECT OUTPUT-CODING DCC NIL)
 (ASPECT COUNT-SEQUENCE DCC NIL)
 (ASPECT COUNT-DIRECTION DCC NIL)
 (ASPECT DISTINGUISHED-STATES DCC NIL)
 (ASPECT LOADABLE DCC NIL)
 (ASPECT RESETABLE DCC NIL)
*(CONPT LOAD DCC WIDTH (G150))
*(CONPT READ DCC WIDTH (G265 G171))
*(CONPT COUNT-UP DCC 1 (G145))
 (CONPT COUNT-DOWN DCC 1 NIL)
 (CONPT LOAD-LINE DCC 1 NIL)
 (CONPT CARRY DCC 1 NIL)
```

where

```
IO-2 = 12
G150 - (CONNECT (DCC LOAD) (DCCBR READ)
                           (UP UNKNOWN HBLANK))
G145 - (CONNECT (DCC COUNT-UP) T (DOWN PC DS5))
G171 - (CONNECT (DCCBR LOAD) (DCC COUNT)
                             (HIGH COMB1 OUTPUT))
G265 - (CONNECT (DM ADDRESS) (DCC COUNT) T)
```

Fig. 4. DCC Counter Function - (1, 2, 3, 4, and 5)

## 4.1. Implementation Method

The first step in the selection process is to deduce values for many of the ASPECTs associated with the function and to verify that there are no inconsistencies in the ASPECT values. The deductions made for the DCC are that it is loadable, resetable, has a binary output, and that the count direction is up. The set of implementation methods associated with the type of function is then considered.

An implementation method has 4 components: Prerequisites, Eliminators, Traits, and Procedure. Fig. 5 illustrates two implementation methods for a counter, one based on a 74161 loadable counter and the other on a 7493 counter. The Prerequisites are

```
(STRATEGY BIN74161 COUNTER
   (PREREQUISITES
       (COUNT-DIRECTION UP)
       (OUTPUT-CODING BIN))
   (ELIMINATORS )
   (TRAITS (RESETABLE YES)
           (LOADABLE YES)
           CARRY)
   (PROCEDURE $BIN74161)
   )

(STRATEGY BIN7493 COUNTER
   (PREREQUISITES
       (COUNT-DIRECTION UP)
       (OUTPUT-CODING BIN))
   (ELIMINATORS )
       (LOADABLE YES)
   (TRAITS (RESETABLE YES))
   (PROCEDURE $BIN7493)
   )
```

Fig. 5. Example Counter Implementation Methods

a list of ASPECTs and their values that must be valid for the function in order to select the method. In the BIN74161 implementation method, the function being implemented must require a binary output coding that only counts up. This is also true for the BIN7493 implementation method.

The Eliminators are a list of ASPECTs and their values which if matched by the function under implementation, eliminate that method from being selected to implement the function.

The Traits are a list of ASPECTs and their values that are true if the method is selected and a list of ports that can be explicitly available if the method is selected. The Procedure is the recipe used to build the circuit. It is represented procedurally in LISP code and is only referenced by name in Fig. 5.

Each method that can be used to implement a function is processed in the selection phase by checking the Prerequisites and Eliminators. A list of all acceptable methods is compiled during the selection. If there is only one acceptable method it is used to implement the function. If more than one method is acceptable then the Traits of all candidates are used to find the most appropriate method. If this does not narrow the list to a single method then one is randomly chosen or the user chooses the method. In this example for the DCC assuming that there are only the two implementation methods from Fig. 5 available to implement a counter, the BIN74161 method is the only acceptable method and is selected.

The procedure associated with the selected method is then run. This procedure consults the frame associated with the function and the connections involving the function, and from this information constructs a real digital circuit, introducing and interconnecting chips that collectively implement the function.

## 4.2. Implemented Circuit

After the first two phases, there are three levels of the DCC design. At the top level there is the DCC function and the related connections. At the bottom level are the chips and wires used to implement the DCC. The intermediate level interfaces the other two levels by providing interconnections between chip pins or wires and the connection points on the function. Hence the hierarchy of the components is maintained and, if necessary, any circuit fragment can be altered without much effect on the rest of the circuit. The implementation for the DCC as designed by the BIN74161 implementation method is illustrated in Fig. 6.

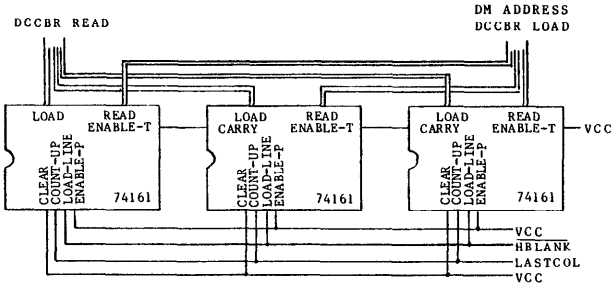


Fig. 6. DCC Implementation

## 5. Conclusion

SADD is a general purpose design system based on the ideas of structured, modular circuit design via an interactive user interface. The current 41 sentences in the input scenerio have been run successfully through the parser creating a database of approximately 600 entries. The design of the 3 counters in the Screensplitter are completed and circuits functionally equivalent to those actually used in the Screensplitter have been designed. The design of a symbolic simulator is currently in progress. This simulator will allow the designer to test and debug the circuits and will complete the design process envisioned for SADD. When completed, SADD will provide a general purpose and extensible knowledge-based system for digital design.