It is now generally agreed that commercial database systems are not sufficient to meet the requirements of design applications.

quite a few years. However, a major weakness of these earlier systems was their inflexible choice of the set of representations that they chose to support. For example, systems have been built in which the design has been represented as a network of devices; as a network plus a logic schematic; as an interconnection of parts from a standard catalog; as a collection of devices, interface pins, interconnections, and signals; as a collection of parts, interconnection nets, wiring, and layout artwork; and as a description that spans logical, physical, and electrical circuit representations. These representations have been "wired" into the database structure, making it almost impossible to incorporate a new representation into an existing database without a major (and potentially catastrophic) reorganization. Such an approach is not acceptable for VLSI-based systems because of the rapid evolution of the design methods and their associated representations. For example, it would be very difficult to integrate a new "sticks"-based layout editor into one of these existing databases.

The current view is that existing database systems are not well-matched to the needs of engineering design applications. It is only recently that database specialists have become interested in applying and extending their techniques for design applications. We will focus on the emerging solutions next.

Design data management: Research in progress

It is now generally agreed that commercial database systems are not sufficient to meet the requirements of design applications. Two approaches are being actively pursued: either extend the database system to better handle such applications, or create a special purpose "database-oriented" file system for design applications. By extending an existing system, it is possible to greatly reduce the effort in building a suitable design management system. However, by retaining much of its general-purpose mechanisms, such a system may not be as efficient as a special-purpose solution.

A number of common themes can be identified across the approaches.

(1) Rather than support a record-oriented interface, as is common with commercial database systems, they provide access to design objects. Design objects are collections of design data that are logically related and that can be manipulated as a single entity. For example, the collection of VLSI geometries that implement an adder could constitute a design object. The geometries are stored as a complex interconnection of records that allow them to be manipulated as a logical group. Design objects form the natural units of data creation, access, and manipulation.

(2) The description design often includes unformatted data, for example, engineering drawings and documentation text. While easily supported by a byte-oriented file system, database systems do not support such data very well, because they have been designed to manage large collections of regularly formatted data (for example, large part catalogs, bank account records, the social security records of every person legally employed in the United States, etc.). The solution is to extend the system with support for long data items, that is, uninterpreted byte streams that can be (almost) arbitrarily long.

(3) Finally, the mode of interaction between a "user" and the database is quite different in the design environment than for typical transaction processing environments (for example,