SYSTEM SIMULATION WITH MIDAS

John W. Bierbauer, Jonathan A. Eiseman, Faiq A. Fazal, and James J. Kulikowski

John W. Bierbauer, Jonathan A. Eiseman, Faiq A. Fazal, *and* ***James J. Kulikowski*** *are all associated with the Computer-Aided Engineering/Computer-Aided Design (CAE/CAD) Development Department in AT&T Bell Laboratories. Mr. Bierbauer, a supervisor at the Indian Hill West facility in Naperville, Illinois, is responsible for development of CAE tools for digital-system design verification. He joined AT&T in 1971 after earning a B.S. in electrical engineering at Illinois Institute of Technology, Chicago, and an M.S. in computer science at Northwestern University, Evanston, Illinois. Mr. Eiseman, who received a B.S. in computer science from Stevens Institute of Technology, Hoboken, New Jersey, is located at Murray Hill, New Jersey. He is a consultant, working on development, maintenance, and support of MIDAS. Mr. Fazal, who* (continued on page 47)

The desire to design systems right the first time and to produce designs that are readily manufacturable is a goal passionately pursued by all engineers. Designers of electronic systems are no exception to this rule; the number and sophistication of the computer-aided design tools used daily by chip designers confirm this. Recently, however, designers of larger electronic *systems* have also been turning to such tools in growing numbers. This article explores how a specific capability, simulation, has contributed to their success.

Introduction

When systems comprised simple, inexpensive parts, a designer could readily construct a working prototype in the lab. While such breadboarding techniques served well in the past, the cost and complexity of today's components prohibit this approach. Many designers are exploiting the benefits of modern electronic-device packaging to pack more functionality into existing space. Unfortunately, many of these parts are difficult to mount by hand and often use pin arrangements that "hide" connection points from manual probing—a serious constraint when debugging in the lab. Increased system clock and signal speeds also pose a problem for the would-be prototype builder, because lab breadboards often cannot provide the close tolerances and specific board layouts required by high-speed designs.

System simulation offers an effective alternative to hardware prototyping. By constructing a model of the system, the designer can judge whether the design is correct and can evaluate tradeoffs in an environment that faithfully reproduces the "real" world. Observability, the ability to monitor points within the circuit and even within the components, is another key reason for the upswing in simulation's popularity. As parts get more complex, the ability to display internal behavior becomes more critical.

Designers are more aware of market pressures in today's global economy. Stiff competition from traditional and nontraditional sources has dramatically shrunk the time available to develop a product. Latecomers to the scene cannot reap the higher profits and

market cornering of their faster paced competitors. Companies can no longer expect to charge premium prices over the life of their products. Product quality has also become an issue in many markets, since customers are no longer willing to put up with poorly designed products that must be debugged in the field.

Simulation addresses these market-related issues in a number of ways. New designs can be brought to the market in less time when fewer iterations are required to achieve the desired results. Thorough simulation increases the engineer's confidence that the design criteria have been met. Simulation also allows the designer to experiment more freely with alternative implementations, which may lead to lower costs, higher operating speeds, better reliability, etc. Such exploration in the breadboarding era was impractical. Better designs at lower cost with fewer bugs and more features are the promise of simulation early in the design process.

Panel 1. Acronyms and Terms

ASIC	application-specific integrated circuit
CAE	computer-aided engineering
ECON	efficient simulator console
EMSP	enhanced modular signal processor
ESIM	efficient simulator
FLAIRX	frame-level analysis and information retrieval on UNIX System
LAMP	logic analyzer for maintenance planning
MIDAS	min/max delay-ambiguity simulator
ODAN	output display and analysis
PLD	programmable-logic device
VLSI	very large-scale integration

A Brief History of System-Level Simulation

Computer programs that simulate the behavior of electrical circuits have been around about as long as the computer itself. Most of the early programs, however, concentrated on simulating the behavior of individual components rather than complete circuits. The advent of the transistor ushered in a new era, in which designs of unheard-of complexity could be achieved. Engineers wanted a way to verify that their new digital designs would really work as desired. Thus, in 1962 several engineers at AT&T Bell Laboratories in Indian Hill (Naperville, Illinois) collaborated to develop the logic analyzer for maintenance planning (LAMP), one of the first digital circuit simulators to be used within AT&T. The LAMP simulator understood how the basic building blocks of digital design (AND gates, OR gates, etc.) performed and was capable of predicting the *logical* behavior of a circuit that comprised an interconnection of such elements.

As circuits grew more complex and the integrated circuit became more popular, designers found that not only the functional behavior but also the *timing* of their designs were important. Early digital system simulators like LAMP did not provide adequate mechanisms for modeling circuit timing. Hence, a second generation of digital simulators was born that did address this need. The LAMP2 simulator, also developed at Indian Hill, allowed a designer to explore circuit timing using minimum, nominal, or maximum timing for all circuit components. Designers could now verify with more confidence that their designs indeed worked under selected timing conditions.

By the early 1980s, designers were discovering that yet another upgrade in simulation was required. The increased use of off-the-shelf very large-scale integrated (VLSI) components was taxing the limits of most gate-level simulators, since a single part might require over 100,000 gates to be properly modeled. Obviously, a circuit that used several such parts would soon exceed the maximum gate count allowed by most simulators. At this time, the concept of *behavioral* simulation was introduced to address this need. The efficient simulator (ESIM), developed by engineers at AT&T Bell Laboratories in Denver, Colorado, introduced behavioral simulation to many engineers throughout AT&T. Behavioral simulators offered a significant advantage over their

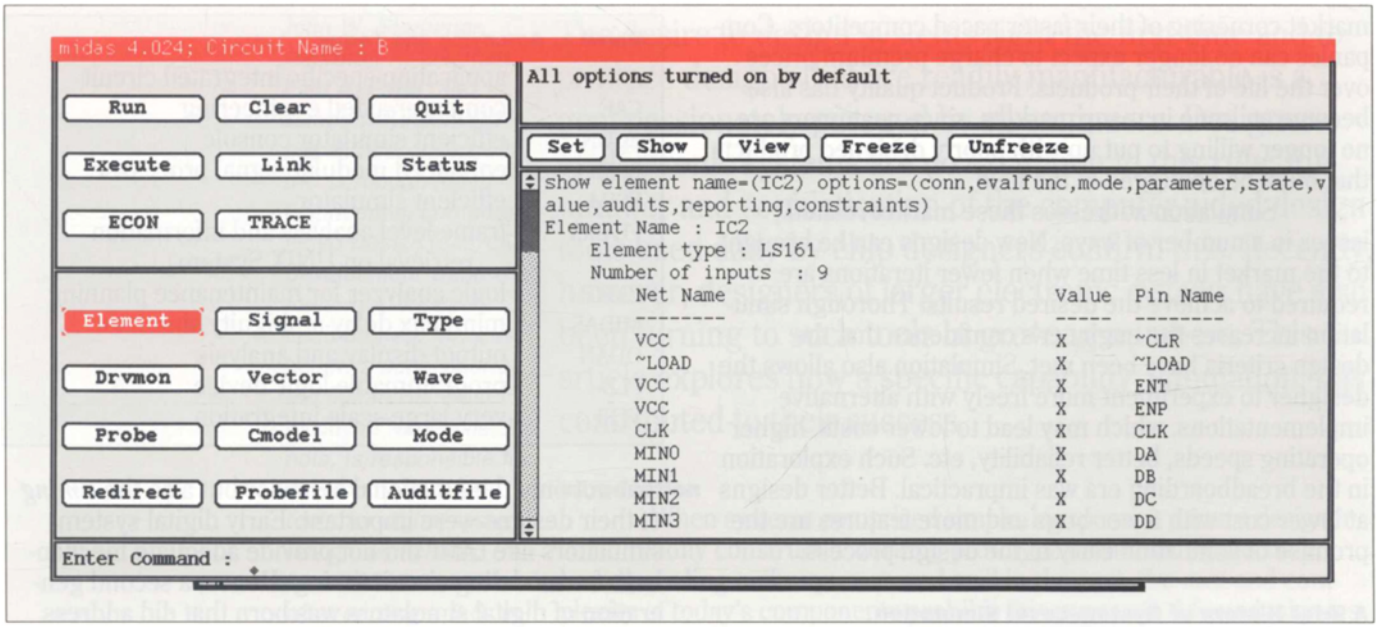


Figure 1. MIDASTOOL, MIDAS' graphical command interface, logically groups actions and options, from which a user can create a specific request. MIDASTOOL then displays the request as a formatted command and saves it in a journal file of the session.

predecessors: complex components could now be modeled in a straightforward way using a high-level language (C in the case of ESIM). The use of behavioral models speeds up the simulation while also affording the opportunity to model more than just individual components (buses, interfaces, and subcircuits, for example, are also easily modeled).

The timing capabilities introduced by LAMP2 and other simulators had one significant drawback. While these simulators allowed the designer to run a circuit under a selected set of timing conditions, it posed the question, "Which timing conditions should be chosen for any given circuit?" This question is addressed by the most recent development in simulation, which involves the simultaneous use of both minimum and maximum delays to predict circuit performance under a full range of timing possibilities. Known as "ambiguity-delay simulation," this technique, when properly applied, yields the most accurate view of circuit capability. See Appendix A for a detailed discussion of ambiguity simulation.

The MIDAS Simulation Environment

The min/max delay-ambiguity simulator (MIDAS), developed by the Computer-Aided Design and Test Laboratory in Murray Hill, New Jersey, was not the product of some abstract "ivory tower" exercise. Rather, MIDAS was born from a very specific list of requirements for a ambiguity-delay simulator. It was developed with the close cooperation of the enhanced modular signal processor (EMSP) project in Whippany, New Jersey. As MIDAS evolved, additional features were included in response to feedback from other projects such as the

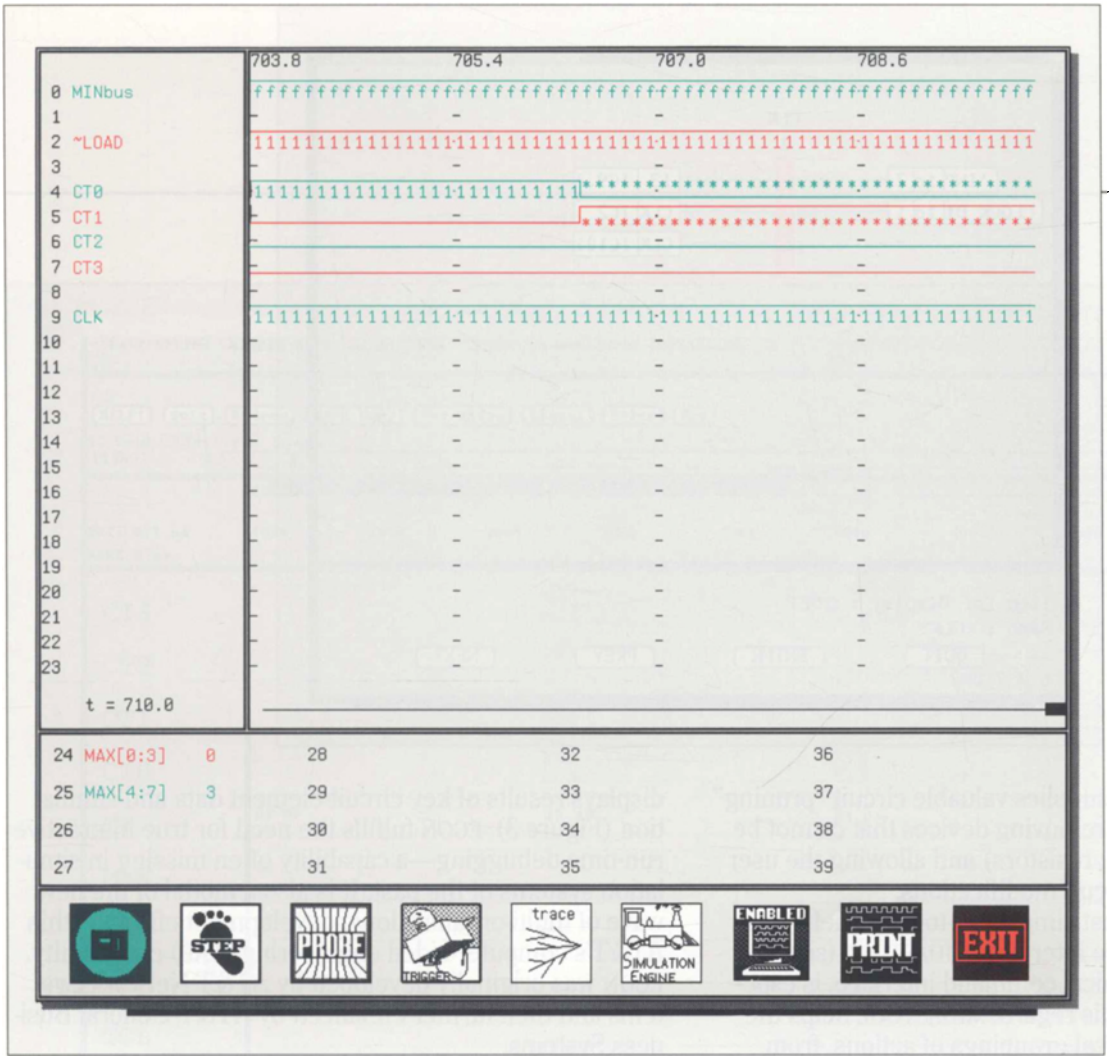


Figure 2. ECON is an interactive, run-time debugging tool that enables a user to communicate with MIDAS during simulation and to view a graphical display of simulation results as they occur.

application-specific integrated circuit (ASIC) hazard analysis project in Allentown, Pennsylvania. Like most complex tools, MIDAS is really just one part of a larger system that provides the complete solution to the circuit designer. This section explores that system and describes how the parts work together to provide an unparalleled simulation capability.

System Overview. Although MIDAS is just one part of a larger system, a variety of tools comprise the total MIDAS simulator. These tools include:

- GENMIDAS, which generates data needed by MIDAS from libraries and the circuit schematic.
- MIDASTOOL, which provides an interactive control panel for MIDAS.
- ECON, or efficient simulator console, which displays run-time control and graphical results for MIDAS.
- ODAN, or output display and analysis, which provides a graphical environment for display and analysis of MIDAS' postrun results.
- FLAIRX, or frame-level analysis and information retrieval on UNIX® System (UNIX is a registered trademark of UNIX System Laboratories, Inc.), which calculates delays for the paths between devices.

A more complete description of each tool follows.

GENMIDAS. Most simulators require some preprocessing of captured schematic data before simulation can begin. GENMIDAS performs this function for MIDAS. In addition to compiling schematic data into MIDAS binary

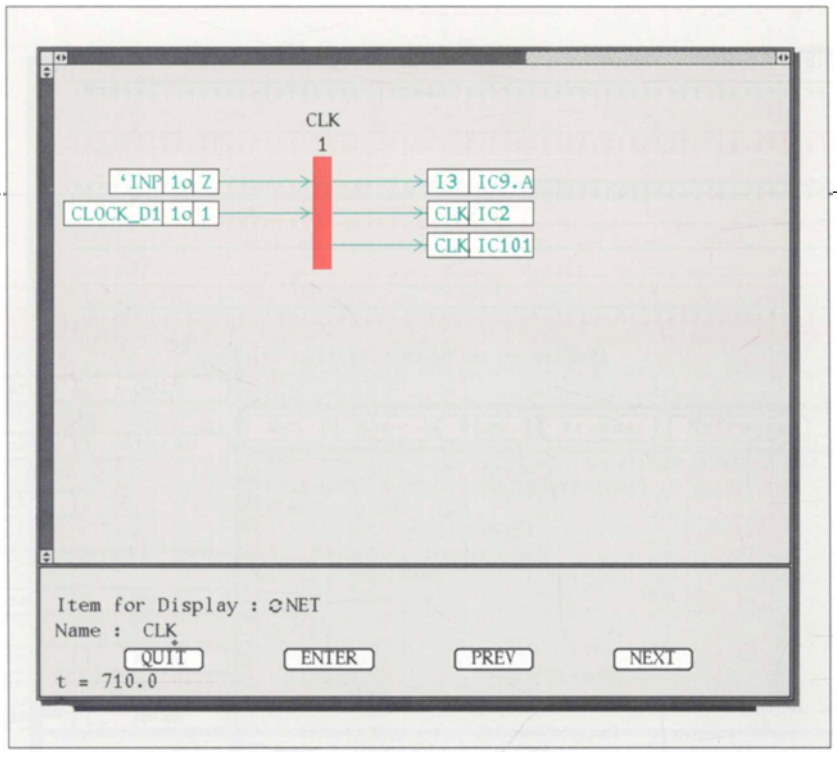


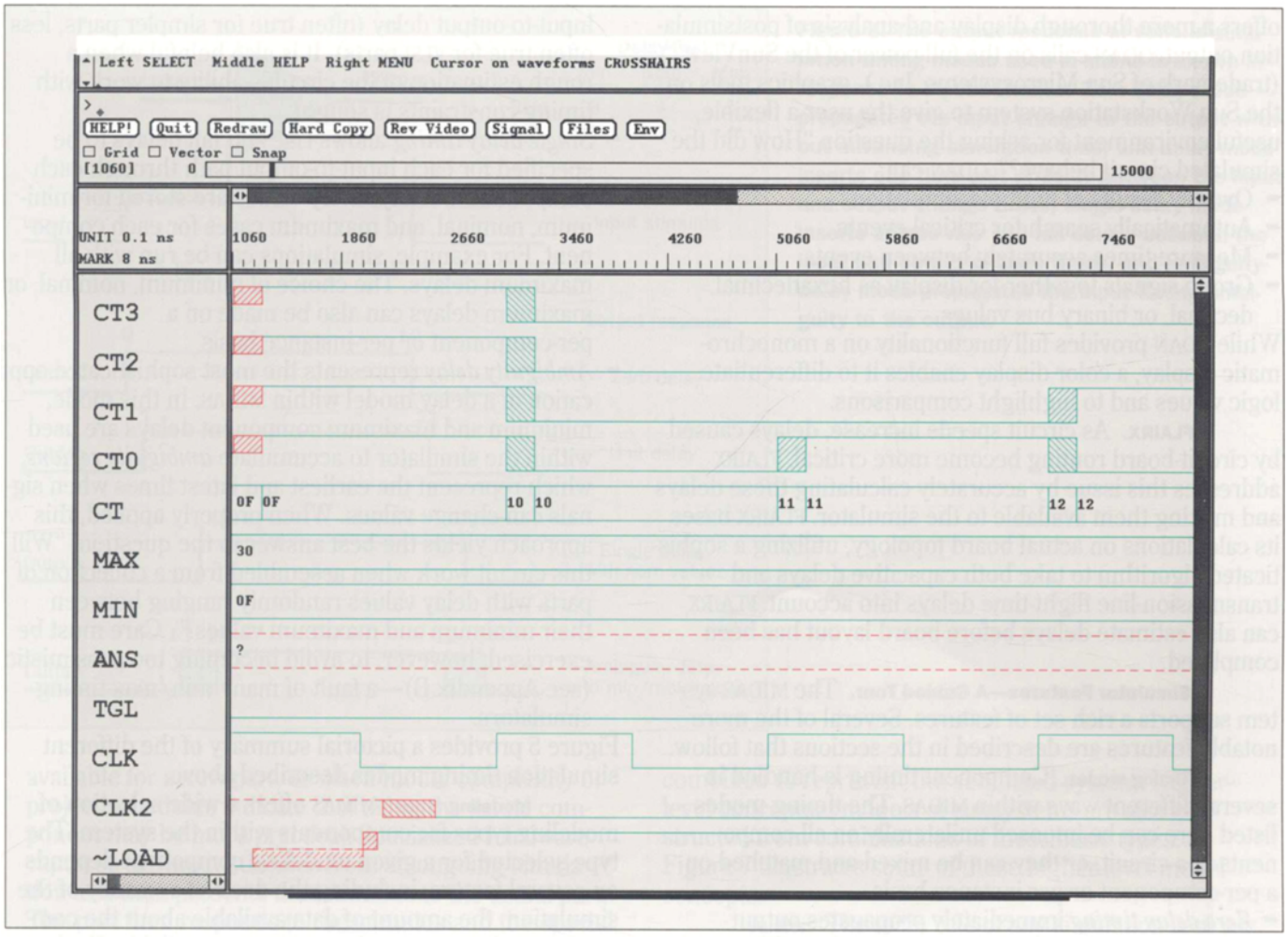
Figure 3. ECON allows the user to trace the route of the simulated circuit and its key elements, as well as view the results of the simulation.

format, GENMIDAS also supplies valuable circuit "pruning" services, automatically removing devices that cannot be simulated (for example, resistors) and allowing the user to specify additional circuit modifications.

MIDASTOOL. Most simulators today benefit from some form of interactive interface. MIDASTOOL (see Figure 1), MIDAS' graphical command interface, is especially user friendly in this regard. MIDASTOOL helps the user by presenting logical groupings of actions, from which the user can select options to form a specific simulator request. MIDASTOOL displays the fully formatted command constructed by the user and saves a journal file of all commands created during the session. Because the user can control MIDAS runs from both the previously stored journal files and the interactive MIDASTOOL control panel, there is less need to learn detailed MIDAS command syntax.

ECON. While MIDASTOOL affords a vehicle for communicating with the simulator during idle periods (before simulation begins or during predetermined pauses), ECON allows communication with MIDAS *during* simulation. Moreover, ECON provides two additional valuable features: run-time graphical display of simulation results (Figure 2), and a dynamic circuit-trace capability. The user can view circuit connectivity while the system displays results of key circuit element data and simulation (Figure 3). ECON fulfills the need for true interactive run-time debugging—a capability often missing in simulation systems of the past. It is also a model of the new wave of multi-organizational development efforts within AT&T's computer-aided engineering (CAE) community. ECON was originally developed by AT&T Network Systems and then further enhanced by AT&T General Business Systems.

MIDAS. The heart of this system is, of course, the MIDAS simulator itself. MIDAS supports a full range of modeling and timing options to allow the designer to call upon more capability incrementally as needed. Thus, a designer might begin with zero-delay simulations using a simple, high-level C-language model early in the design cycle. As the design progresses, the designer could introduce more detailed models and more accurate timing options. Finally, to achieve the highest possible level of confidence in the design, ambiguity-delay simulation could be invoked. While MIDAS has been designed to be feature rich, much attention also has been paid to performance. Thus, it is still feasible to simulate large circuits (100,000+ interconnects) in a Sun Workstation® system environment. MIDAS (and most accompanying tools) can also be executed in the UNIX System V environment on



an Amdahl mainframe. (Sun Workstation is a registered trademark of Sun Microsystems, Inc.)

ODAN. No suite of simulation tools would be complete without a mechanism for displaying simulation results. While ECON fills part of this need by displaying simulated logic values from selected points within the circuit while the simulation is running, ODAN (Figure 4)

Figure 4. ODAN gives the user a more thorough display and analysis of postsimulation results than ECON offers. ODAN can overlay the results of multiple simulations; automatically search for critical events; measure time between events accurately; and group signals together for display as hexadecimal, decimal, or binary bus values. ODAN works on both color and monochromatic displays.

offers a more thorough display and analysis of postsimulation output. ODAN calls on the full power of the SunView™ (trademark of Sun Microsystems, Inc.) graphics tools on the Sun Workstation system to give the user a flexible, useful environment for asking the question "How did the simulated circuit behave?" ODAN can:

- Overlay results of multiple simulations
- Automatically search for critical events
- Measure times accurately between events
- Group signals together for display as hexadecimal, decimal, or binary bus values.

While ODAN provides full functionality on a monochromatic display, a color display enables it to differentiate logic values and to highlight comparisons.

FLAIRX. As circuit speeds increase, delays caused by circuit-board routing become more critical. FLAIRX addresses this issue by accurately calculating these delays and making them available to the simulator. FLAIRX bases its calculations on actual board topology, utilizing a sophisticated algorithm to take both capacitive delays and transmission-line flight-time delays into account. FLAIRX can also estimate delays before board layout has been completed.

Simulator Features—A Guided Tour. The MIDAS system supports a rich set of features. Several of the more notable features are described in the sections that follow.

Timing modes. Component timing is handled in several different ways within MIDAS. The timing modes listed here can be imposed unilaterally on all components in a circuit, or they can be mixed and matched on a per-component or per-instance basis.

- *Zero-delay timing* immediately propagates output values when changes occur in input values. This mode is often used early in a design to verify functionality. Also, zero-delay simulation requires considerably less run-time than simulation with delays.
- *Unit-delay simulation* provides for a single time-unit's delay between any input change and the corresponding output change. This mode is useful when most components in the design exhibit about the same input-to-output delay (often true for simpler parts, less often true for VLSI parts). It is also helpful when a rough estimation of the circuit's ability to work with timing constraints is sought.
- *Single-delay timing* allows rise and fall delays to be specified for each input-to-output path through each component. Usually, delay values are stored for minimum, nominal, and maximum cases for each component. For example, simulations can be run with all maximum delays. The choice of minimum, nominal, or maximum delays can also be made on a per-component or per-instance basis.
- *Ambiguity delay* represents the most sophisticated application of a delay model within MIDAS. In this mode, minimum and maximum component delays are used within the simulator to accumulate *ambiguity regions*, which represent the earliest and latest times when signals can change values. When properly applied, this approach yields the best answer to the question, "Will this circuit work when assembled from a collection of parts with delay values randomly ranging between their minimum and maximum values?" Care must be exercised, however, to avoid becoming too pessimistic (see Appendix B)—a fault of many min/max timing simulators.

Figure 5 provides a pictorial summary of the different simulation timing modes described above.

Modeling types. MIDAS offers a wide selection of modeling types for components within the system. The type selected for a given modeling component depends on several factors, including the desired accuracy of the simulation, the amount of data available about the component, and the computing resources available. Each type is summarized here.

- *Behavioral models*, which are written in C language (see Figure 6), represent the backbone of the MIDAS modeling approach. Each model is a C function that accepts changes on component input pins and produces changes on output pins.
- *Hardware models* are used when a data sheet is not

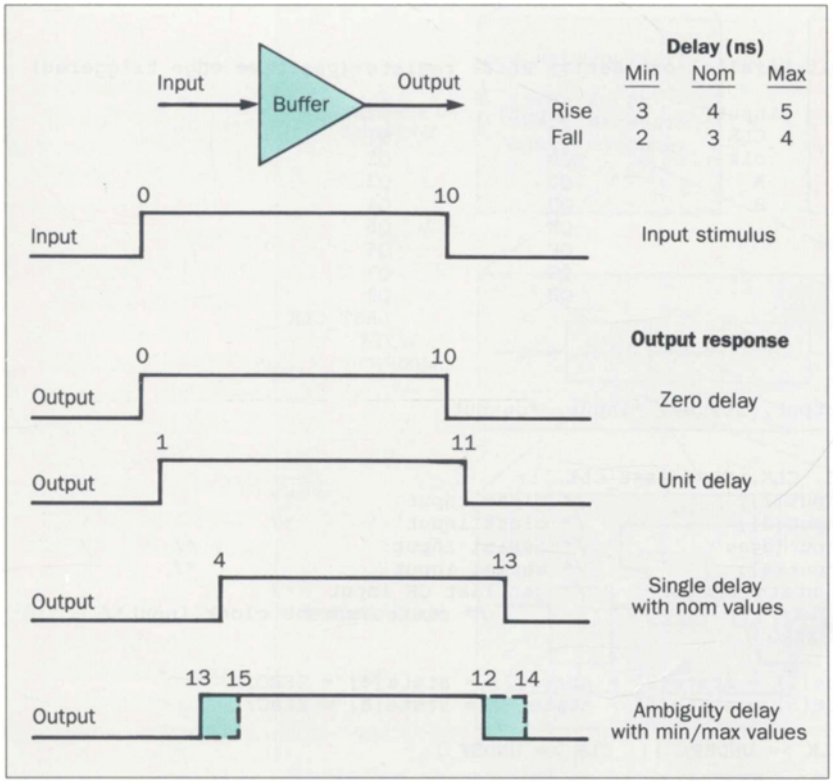


Figure 5. The output response of various simulation timing modes used in MIDAS is shown in response to input stimulus. Zero-delay mode propagates the input change to the output without advancing simulation time; unit-delay mode inserts one time unit of delay between the input and output change times; single-delay mode inserts unique rise and fall delays between the input and output change times; and ambiguity-delay mode propagates the input timing ambiguity to the output.

available for a new part, or when model complexity or project schedules indicate that using the actual component may be more practical. Specialized hardware captures the input stimulus from an ongoing simulation and then presents that stimulus to the actual part. The part then responds to the stimulus and the special hardware captures the new output, reflecting these values back into the simulation.

- *Table-driven models* use Boolean tables to describe the behavior of simpler parts. They are employed when simulation time is critical and when component timing need not be included in simulation. These models typically execute faster than their behavioral counterparts.

All the model types just described can be effectively combined to represent the simulated system. Higher-level conceptual models of subsystems are also constructed from combinations of these basic types. Figure 7 illustrates some of these higher-level model concepts.

Audits. The main role of any simulation is to ask the question, "Does this design work?" MIDAS helps to answer this question by providing several optional audits, or checks, which are invoked at run-time. The use of such audits unburdens the designer from having to scrutinize reams of simulator output to look for trouble areas. The audits consist of:

- *Setup audit*—Data inputs to sequential circuit elements are monitored to ensure that the data value

Figure 6. A typical MIDAS behavioral model, written in C-language.

```
/*      xx164() 8-bit Parallel-out Serial shift register(positive edge triggered)
 *
 *      index:          input:          output:         states:
 *        1               CLR^            QA              Q1
 *        2               clk             QB              Q2
 *        3               A               QC              Q3
 *        4               B               QD              Q4
 *        5                               QE              Q5
 *        6                               QF              Q6
 *        7                               QG              Q7
 *        8                               QH              Q8
 *        9                                               LAST_CLK
 */
#include "lib.h"
#include "vec.h"
int xx164(input, output, state), *input, *output;
STATE   *state;
{
        int     A, B, CLK, CLR, last_CLK, i;
        CLR     = input[1];             /* clear input          */
        CLK     = input[2];             /* clock input          */
        A       = input[3];             /* serial input                 */
        B       = input[4];             /* serial input                 */
        last_CLK  = state[9];           /* get last CK input    */
        state[9] = CLK;                         /* store current clock input*/
        if ( CLR == ZERO )
        {
                state[1] = state[2] = state[3] = state[4] = ZERO;
                state[5] = state[6] = state[7] = state[8] = ZERO;
        }
        else if ( CLK >= UNDEF  ||  CLR >= UNDEF )
        {
                state[1] = state[2] = state[3] = state[4] = UNDEF;
                state[5] = state[6] = state[7] = state[8] = UNDEF;
        }
        else  /* ( CLR == ONE )        */
        {
                if ( vrise[last_CLK][CLK] == 1 )  /* rising edge */
                {
                        /* perform shift        */
                        for (i = 8; i > 1; --  i)
                                state[i] = state[i-1];
                        state[1] = vand[A][B];
                }
        }
        /* set outputs to internal states */
        for (i = 1; i <= 8; ++i)
                output[i] = state[i];
        return(0);
}
```

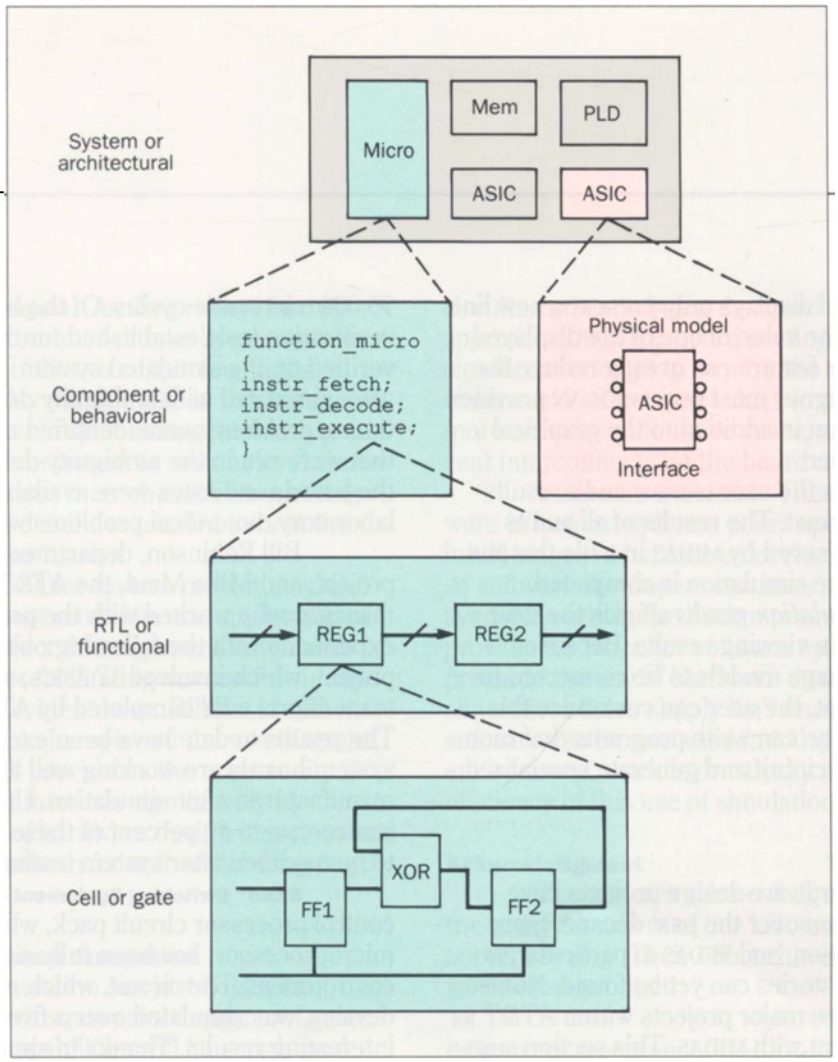


Figure 7. MIDAS uses a combination of modeling types—hardware models, behavioral models, and table-driven models—to represent a simulated system. The models shown here are higher-level conceptual models constructed from the modeling types.

reaches a known stable state for a specified period of time before the clock changes.

- *Hold audit*—Data inputs to sequential elements are monitored to ensure that the data value remains stable for a specified period of time after the clock changes.
- *Spike audit*—Uncertainty in output changes, often caused by timing anomalies, are reported.
- *Overlapping ambiguity*—If the controlling inputs to a device have ambiguity regions that overlap (i.e., a race condition), then the performance of that device may be compromised; all such occurrences are reported.

Output. The presentation of simulation results is indeed a critical factor. Most designers spend more time interacting with the output of the simulation than with the simulator itself. Thus, the design of the output subsystem is very important. MIDAS offers several output alternatives:

- *Run-time graphical output* is provided by ECON. ECON is used primarily as a debugging aid that allows the designer to halt the simulation if no additional value can be gained from it.
- *Postrun graphical display* is handled by ODAN, a powerful display and analysis tool invoked after a simulation run has been completed. ODAN provides several measurement and display options, including the ability to display data from any point in simulated time.
- *Textual display* is available for those designers who prefer textual display of simulation results. For

example, most textual displays only present a new line of text output when the value of one of the displayed signals changes. This feature can greatly reduce the amount of data a designer must review. ODAN provides a textual display option in addition to the graphical mode discussed earlier.

- *Audit reporting* allows the user to view audit results in a clear, concise format. The results of all audits described earlier are saved by MIDAS in a file that the user can view after the simulation is completed.
- *Custom output of simulation results* affords the user maximum flexibility in viewing results. Because MIDAS allows C-language models to be connected to any point in the circuit, the user can customize the output. Thus, he or she can write programs that monitor points within the circuit and generate specialized, formatted output.

Experience Within AT&T

Many AT&T hardware-design projects have benefited from simulation over the past decade. Since ambiguity-delay simulation, and MIDAS in particular, is very new, fewer case histories can yet be found. Nonetheless, we can cite three major projects within AT&T as having some involvement with MIDAS. This section summarizes their experiences.

The EMSP Project. This project was the first user of the MIDAS ambiguity-delay capability. In fact, EMSP provided the driving force for many key features in the simulator.

The project's extensive use of MIDAS firmly established the viability of the tool. More than one dozen circuit boards of varying complexity (many with ASICs) were simulated in MIDAS. Simulation was used extensively on this project, often simulating subsystems of multiple boards, in addition to running standalone simulations on each board.

Assembled test code was even loaded into simulated RAM to verify circuit operation. Running in this mode, 300,000 simulation vectors were required to cover 75,000 microcode cycles. Of the hundreds of hardware verification tests established for the project, most were verified on the simulated system. While not all boards were simulated with ambiguity delays, several significant timing problems were identified on some of the boards that were run in the ambiguity-delay mode. When all of the boards and ASICs were available to be tested in the laboratory, no critical problems were discovered.

Bill Robinson, department head in charge of the project, and Mike Maul, the AT&T Microelectronics manager who worked with the project, summed up their experience with the following observations. The EMSP project, which involved 17 ASICs, was one of the largest team efforts ever completed by AT&T Microelectronics. The results to date have been exceptional, in that the system boards are working well with the first ASICs manufactured after simulation. Under normal circumstances, up to 40 percent of these ASICs would have had to be modified after system testing.

5ESS® Switching Equipment Project. The common-control-processor circuit pack, which utilizes a Motorola microprocessor, has been fully simulated in the MIDAS environment. The circuit, which contains over 200 devices, was simulated over a five-week period with very interesting results. Thanks to simulation:

- Designers found several design flaws, including static inputs tied to the wrong logic state, logic errors in programmable-logic-device (PLD) equations, and glue-logic errors.
- Designers also uncovered many setup and hold timing violations as they simulated the circuit.
- When a wire-wrapped prototype board was finally crafted, only eight "white wires" were needed to bring up the circuit in the laboratory, and most of those were associated with a portion of the board that was not simulated.

In reflecting on their experience, Ron Munoz, one of the designers, stated, "Digital-circuit-pack simulation will become an integral part of our laboratory design process, much as chip simulation is today."

The AUTOPLEX® Project. As the product line of AUTOPLEX Cellular Telecommunications System moves from supporting analog radios to supporting digital radios, simulation will play a major role in the design of these new radio products. Project management has recognized that early commitment to system-level simulation significantly improves their chances for meeting the aggressive schedule deadlines needed to beat the competition in the cellular market.

The AUTOPLEX cellular system project may well prove to be the preeminent model for the application of simulation in a large project. Management vision and support, key elements for facilitating successful simulation in a project, is very evident here. Bill Zucker, the design manager who first championed the use of MIDAS, believes that, "MIDAS will provide us the opportunity to reduce our development intervals significantly by decreasing the number of hardware design iterations and by providing an early platform for software debugging and testing."

Future Directions of System Simulation

System-level simulation has just begun to become widespread. As this powerful design tool gains wider acceptance, new capabilities will be sought and developed. While the future is always difficult to predict, some emerging trends in this area can nonetheless be identified. Several of the more interesting possibilities are discussed here.

Mixed Digital/Analog Simulation. The MIDAS system simulates digital systems very efficiently. As digital circuits find broader, more diverse applications, the need to mix digital and analog components in the same simulation will increase. Analog techniques will also need improvement if they are to model higher-speed circuits more accurately.

Better Timing Models. While fairly accurate for circuits that operate at speeds up to 25 MHz (megahertz), the timing model used within MIDAS may prove inadequate to support the simulation of some of the faster circuits currently being designed. When circuit speeds approach 100 MHz, board-routing delay becomes significant enough to require the use of distributed capacitance and inductance values to model timing behavior more accurately. In these cases, crosstalk between adjacent interconnects on the board must also be modeled.

Software Verification. The extensive use of software in current systems poses specific needs. Today, the initial stages of hardware development are often rushed so software developers can have a working prototype available early in the design process. In fact, the need to provide hardware for software testing has been known to preclude the use of hardware-design verification via simulation: there simply isn't enough time. While system software can be executed in today's simulated-hardware environment, work is drastically needed to improve the efficiency of this use of simulation.

Acknowledgment

The authors acknowledge the contributions of the many members of our laboratory who helped make MIDAS a reality. Their efforts have established MIDAS as a world-class product.

Biographies (continued)

joined AT&T in 1982, is a distinguished member of technical staff responsible for design verification aids for digital designs. He earned a bachelor of technology degree in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur; he received an M.S. in computer science from the University of Pennsylvania, Philadelphia. Mr. Kulikowski, a member of technical staff at Indian Hill West, works on ambiguity-delay simulation, modeling issues, and the design/simulation process. He joined AT&T in 1977 after earning a B.S. in computer science from the University of Connecticut, Storrs. He also received an M.S. in computer science from Northwestern University.

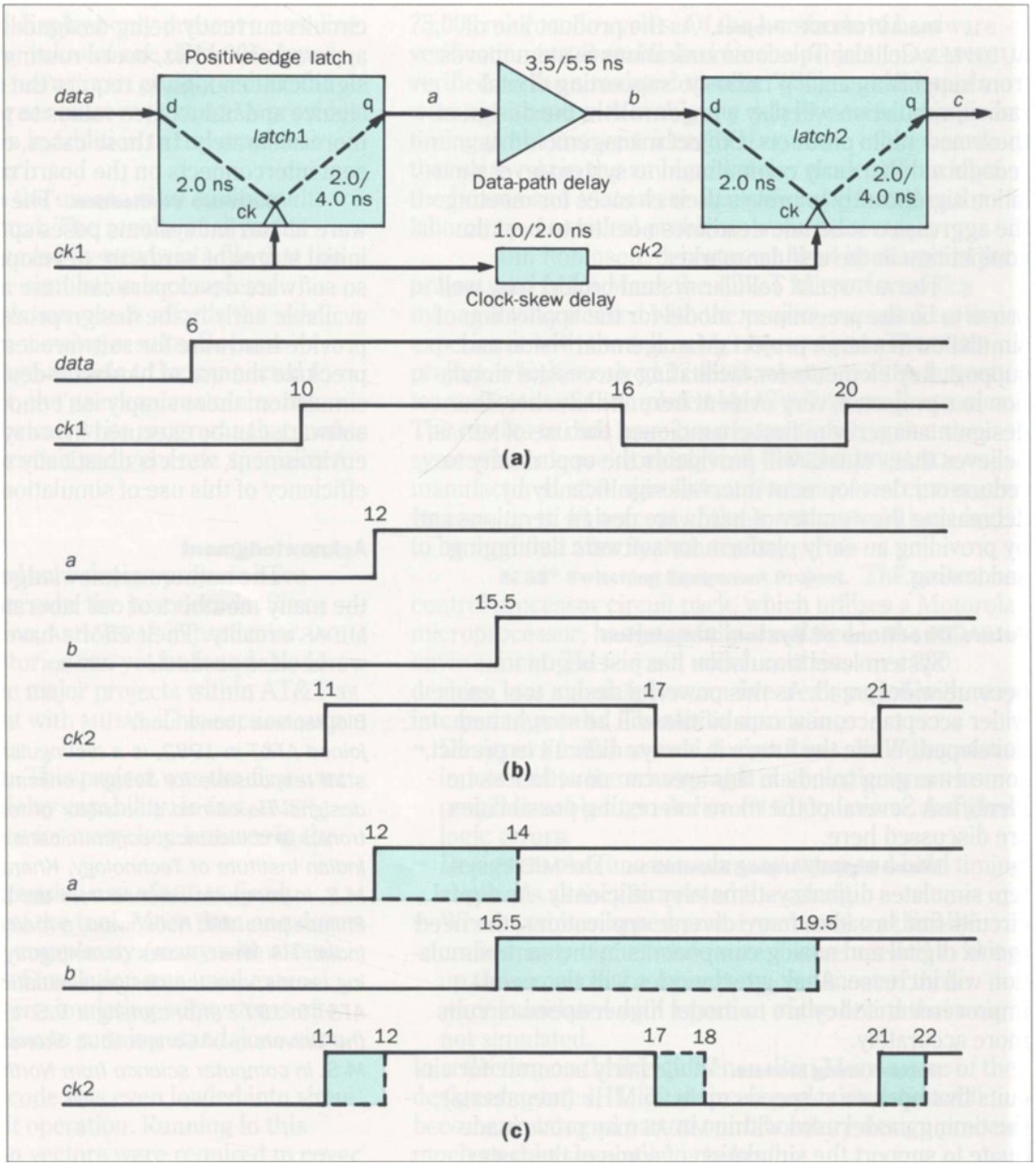


Figure A-1. This circuit demonstrates the importance of removing common ambiguity. (a) The sample circuit and stimulus. (b) Traditional single-delay simulation using min values for delays. (c) Ambiguity-delay simulation using min and max values for delays.

Appendix A. Ambiguity-Delay Simulation

An ambiguity-delay simulator allows a range of delay values to be associated with a circuit's individual elements within a single simulation run. The circuit shown in Figure A-1a has two instances of a positive, edge-triggered latch with a setup time of 2 ns (nanoseconds) between the data, *d*, and clock, *ck*, inputs. The delay between a rise of input, *ck*, and the corresponding change on the output, *q*, has a min value of 2 ns and a max value of 4 ns. For simplicity, the data path between the two latches is represented by a single buffer with min and max delays of 3.5 and 5.5 ns, respectively. The clock skew between the two latches is represented by a delay element with min and max values of 1 and 2 ns, respectively.

If the stimulus to the nets, *data* and *ck1*, is as shown in Figure A-1a, the circuit's response (shown in Figure A-1b) is as predicted by a traditional *single-delay* simulator using the min values for all delays. If such a simulator includes dynamic auditing of setup violations, it will find none for this run. The simulation of this circuit with max delays will be similar, differing only in the times at which transitions occur. However, if max delays are considered for the path *ck1*→*a*→*b*, while min delay is used for the path *ck1*→*ck2*, then the simulation run will show the transition of *b* at 19.5 ns and of the second 0→1 edge on *ck2* at 21 ns; a setup violation will be flagged.

It appears from the above discussion that a rigorous timing analysis of a circuit that includes elements with delay ranges requires simulation with all possible combinations of min and max delay values. These combinations pose an explosive proposition and are not feasible for any reasonably sized circuit. This is where ambiguity-delay simulation can help.

Ambiguity-delay simulation considers all possible delay values in a single simulation run. Figure A-1c illustrates this type of simulation. As a signal change traverses from one point in the circuit to another, the simulator tracks the min and max delays encountered along the path. It then uses these delays to compute the earliest and latest times for the signal transitions. Thus, for net *b*, the sharp transition at 15.5 ns of Figure A-1b is replaced by a boxed region in Figure A-1c. The bounding time values of 15.5 and 19.5 ns correspond to the min and max delays along the path *ck1*→*a*→*b*. In this type of simulator, a dynamic audit can detect a setup violation for *latch2* by comparing the latest transition time of *b* with the earliest time computed for the second *0*→*1* transition on *ck2*.

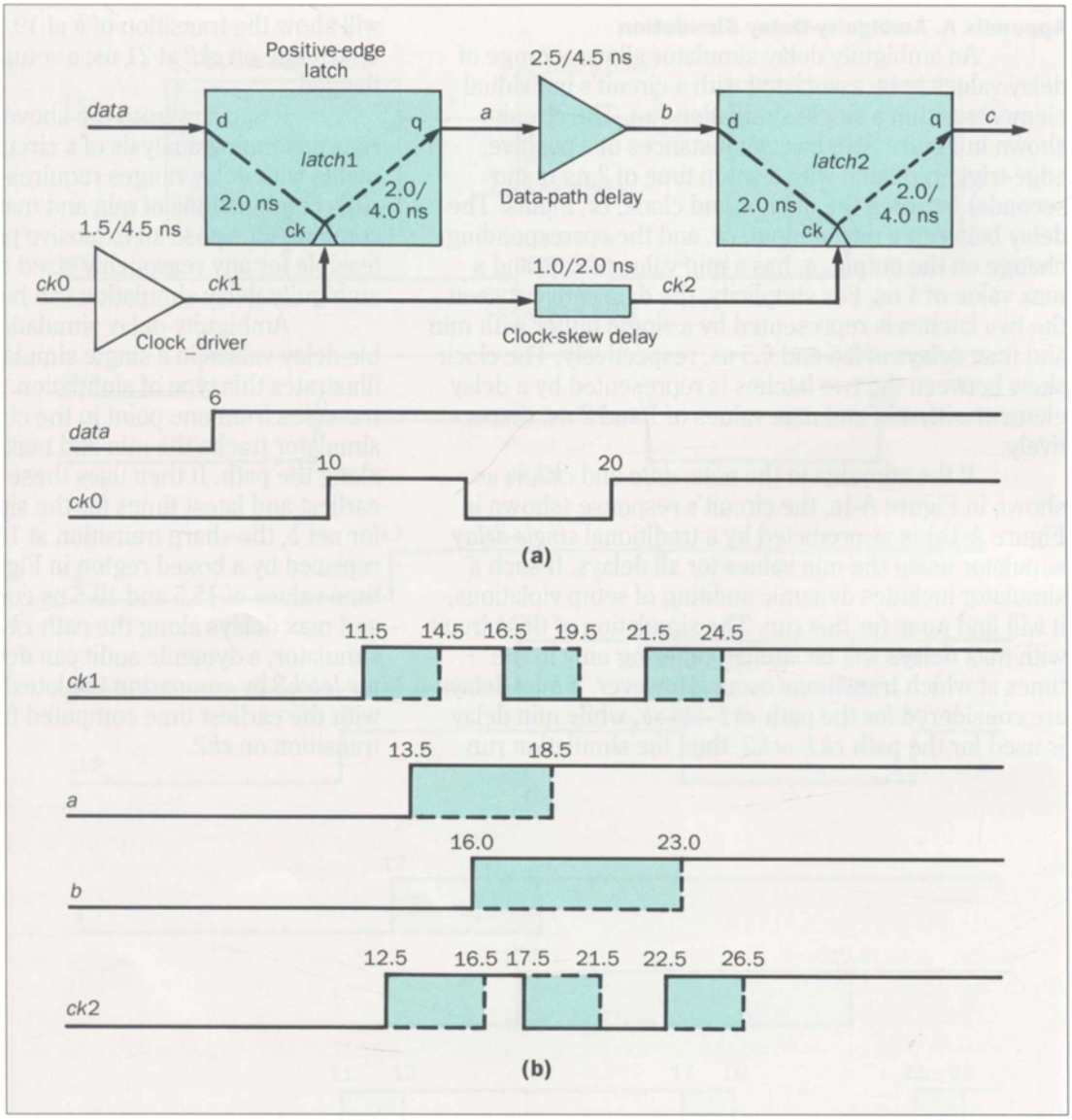


Figure B-1. Common ambiguity arises from path reconvergence. (a) A sample circuit and stimulus. (b) Ambiguity-delay response of the sample circuit.

Appendix B. The Common-Ambiguity Problem

Dynamic timing analysis compares transition times for two or more events. In ambiguity-delay simulation, this requires special processing, referred to as *common-ambiguity removal*. Common ambiguity may exist between two events if the circuit paths traversed by these events are identical, or if the paths have one or more elements in common. The difference in the min and max delays of the common portions of the paths represents the common ambiguity.

To appreciate the importance of removing common ambiguity, we will consider the circuit shown in Figure B-1a. Figure B-1b illustrates the ambiguity-delay response of this circuit.

The max delay of the data path *ck1*→*a*→*b* is 8.5 ns, which is 2.5 ns less than the clock period plus min delay of the clock path *ck1*→*ck2*. Since the setup time is 2 ns, we do not expect any data setup problems. However, from the waveforms, it appears that the transition on *b* can occur as late as 23 ns, while the second 0→1 transition of *ck2* can occur as early as 22.5 ns. While this implies that incorrect data may be latched, this conclusion is incorrect, since it assumes that the clock-driver element is simultaneously set to the max delay in the path *ck0*→*ck1*→*a*→*b* and to the min delay in the path *ck0*→*ck1*→*ck2*, which is not possible.

The difference of 3 ns in the min and max delays of the common clock-driver element in the two paths represents the common ambiguity (CA) in the transitions of *b* and *ck2*. The CA should be removed in any temporal comparison of the two events. This could be achieved by subtracting the CA of 3 ns from the latest transition time of the transition on *b*. On doing so, we note that the guaranteed min separation between the event on *b* and the second 0→1 event on *ck2* is 2.5 ns, as expected.

(Manuscript received October 11, 1990)