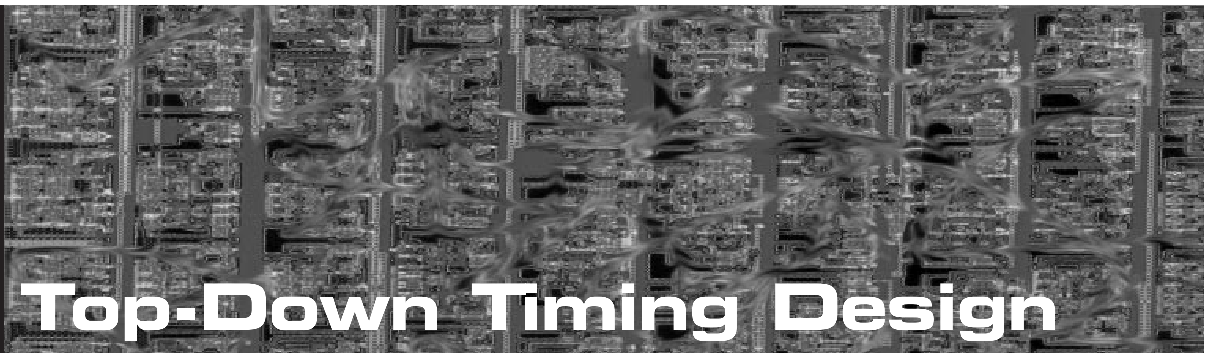
Reprinted with permission of *Integrated System Design*, July, 1997

Upfront timing analysis reduces timing errors and leads to thoroughly optimized designs.



Top-Down Timing Design

by Bryan Hoyer & Donna Mitchell

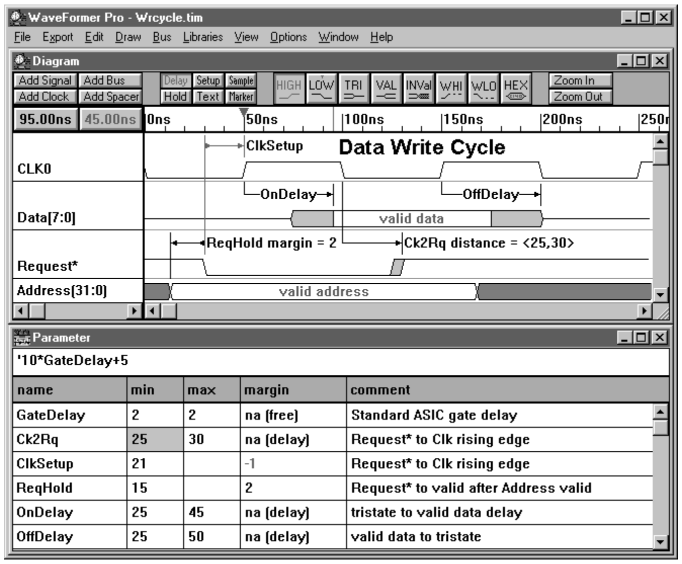
Generally when a design is started, a designer is given high-level timing requirements that must be met by his design. Examples of high-level timing requirements include computational throughput (millions of instructions per second), data transmission rates (megabytes per second), and response latencies to requests from external systems. A designer takes these times and converts them into cycle times (for example, 10 MIPS = 100 ns per instruction). These cycle times serve as "timing budgets" for the designer, and he must work within these constraints during the design of his system. As the design is broken down into smaller and smaller subsystems, the cycle times are subdivided and assigned to each subsystem. The process of moving through each successive level of timing is called "top-down timing design."

How does top-down timing design differ from back-end timing analysis? The key feature of a top-down timing design methodology is an up-front analysis of system timing (usually using timing diagrams) right at the beginning of the design cycle, where errors can be quickly corrected. Back-end timing analyzers such as digital simulators and static timing analyzers require the designer to have already committed to a specific circuit implementation, which is usually specified as a schematic or HDL netlist with associated simulation models. This requirement for a nearly completed design prior to beginning timing analysis means that any timing errors discovered during a back-end analysis may result in a significant redesign effort. In addition, when timing or logic errors are found at this stage, any circuit changes must be reflected in many different places: the schematic, model descriptions, testbench stimulus for the simulator or timing analyzer, and documentation accompanying the design. Top-down timing design helps the designer discover timing errors earlier in the design cycle, resulting in fewer changes and faster product development.

Back-end timing analyzers are most useful as design sign-off validators, rather than as design aids. It is difficult to use a back-end timing analyzer to maximize design speed because most of the design decisions have been made by the time a back-end timing analysis begins. Examination of significantly different design choices is not possible because each design alternative requires too much front-end design time before its performance can be analyzed. Front-end timing analyzers, such as WaveFormer Pro by SynaptiCAD Inc. (Blacksburg, VA), excel at timing optimization because comparatively little design information needs to be entered in order to analyze system performance.

Front-end timing analyzers don't invalidate the use of back-end timing analyzers. While front-end timing analyzers are quicker and require less design entry, back-end timing analyzers benefit from having complete design information and generally perform a more exhaustive, brute-force analysis of a design's performance that can detect errors that a designer may have missed during a front-end timing analysis. In fact, one of the biggest challenges of using a back-end timing analyzer is sifting through the data generated by the analysis to discover real timing errors. Designers, nevertheless, will benefit from the combination of both front-end and back-end timing analysis tools to achieve the highest level of confidence in the timing of a design.

Benefits of timing-diagram analysis Designers have traditionally used hand-drawn timing diagrams to perform their initial timing analysis. Timing diagrams offer the advantage of enabling the designer to see the effects of timing constraints on a system's signals. With a timing diagram, the cause-and-effect relationships between signal transitions are shown by timing parameters such as delays, setups, and holds, providing the designer with more insight into the operation of a system. This is particularly helpful because timing errors are found early in the design cycle. Also, critical paths that cause errors can be analyzed visually by following delay chains back through the diagram. Each delay in a critical path indicates a point at which a circuit change could alleviate a timing error or increase system performance.



'10*GateDelay+5

name	min	max	margin	comment
GateDelay	2	2	na (free)	Standard ASIC gate delay
Ck2Rq	25	30	na (delay)	Request* to Clk rising edge
ClkSetup	21		-1	Request* to Clk rising edge
ReqHold	15		2	Request* to valid after Address valid
OnDelay	25	45	na (delay)	tristate to valid data delay
OffDelay	25	50	na (delay)	valid data to tristate

Figure 1. Timing diagram used for top-down timing design. The screen capture of WaveFormer Pro shows margin calculations and delay parameters.

Timing-diagram analysis also gives you the advantage of being able to account for min/max delay tolerances at the same time. This is critical for asynchronous designs such as bus transaction protocols because most simulators can only simulate at the extreme timing end points, but not over the entire operational range of the design.

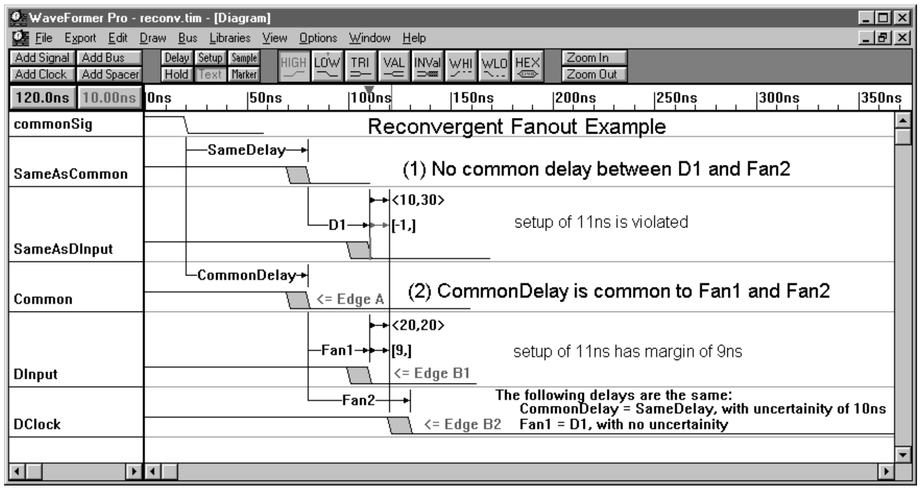
Many designs are never given a thorough timing-diagram analysis, despite the benefits, because of the amount of time required to perform such an analysis manually. Drawing timing diagrams by hand is a difficult and error-prone task, and the resulting diagrams are generally messy and hard to understand and analyze. The messiness is compounded when timing errors are discovered and fixed, causing the hand-drawn timing diagrams to be erased and re-drawn. In addition, the timing analysis of manually generated timing diagrams is usually overly pessimistic because it is difficult to find and remove common delays from setup and hold calculations.

Timing-diagram editors eliminate the need to manually generate timing diagrams. They allow the designer to easily create and change timing diagrams, and they reduce the chance for error in timing calculations. Timing diagrams can be created with a combination of point-and-click drawing, Boolean and temporal equations, and auto-generated clocks and buses. Designers can analyze system performance using just the timing parameter information available in data books without the need for schematic netlists and complex simulation models. Timing calculations are performed interactively, so it is easy to quickly assess the impact of a change in a timing parameter by updating the parameter's value or switching to a different part library.

Timing diagram editors can provide several types of timing and logic analysis:

- Automatic calculation of critical paths and constraint checking (setup and hold margins).
- Common delay removal.
- Delay correlation calculations.
- Combinatorial logic analysis using Boolean equations.
- Temporal equations for DSP analysis.
- Reuse of timing during simulation.

Calculation of critical paths and constraint checking Timing calculations in a timing diagram analyzer are performed using true min/max timing, and calculations are automatically adjusted to account for subtle timing effects such as reconvergent fanout. With timing-diagram editor output, it is easier to see timing relationships between signal transitions than with a simulator because the timing parameters (propagation delays, setup times, and hold times) graphically specify cause-and-effect relationships between signal transitions and timing constraints that must be met by the system. This is one reason why engineers who have access to powerful simulators still draw timing diagrams by hand. Timing-diagram editors also allow designers to verify min/max timing simultaneously so that the overall impact of IC-timing tolerances on a system's performance can be properly assessed.


WaveFormer Pro - reconv.tim - [Diagram]
File Export Edit Draw Bus Libraries View Options Window Help
Add Signal Add Bus Add Clock Add Spacer Delay Setup Sample Hold Text Marker HIGH LOW TRI VAL INVal WHI WLO HEX Zoom In Zoom Out
120.0ns 10.00ns
0ns 50ns 100ns 150ns 200ns 250ns 300ns 350ns
commonSig
SameAsCommon
SameAsDInput
Common
DInput
DClock
SameDelay
D1
<10,30>
[-1,]
CommonDelay
<= Edge A
Fan1
<20,20>
[9,]
<= Edge B1
Fan2
<= Edge B2

Reconvergent Fanout Example

(1) No common delay between D1 and Fan2

setup of 11ns is violated

(2) CommonDelay is common to Fan1 and Fan2

setup of 11ns has margin of 9ns

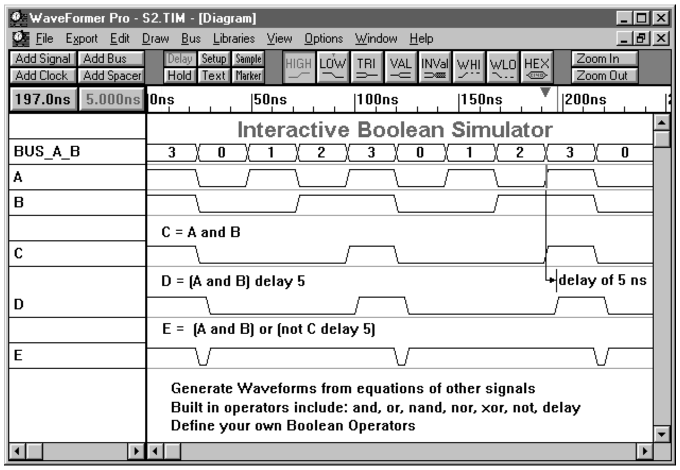
The following delays are the same:
CommonDelay = SameDelay, with uncertainity of 10ns
Fan1 = D1, with no uncertainity

Figure 2. Reconvergent fanout example. Reconvergent fanout adjustment removes common delays from calculation margin and distance values.

Reconvergent fanout Margin calculations in some circuits can be overly pessimistic if all the uncertainty times in a timing path are included in the calculations. For instance, if two signal transitions, B1 and B2, are caused by the same transition, A (see Figure 2), then margin calculations between B1 and B2 should not include the uncertainty of transition A, because no matter when A transitions, it will occur at the same time for both B1 and B2. When this happens, the circuit is said to have "reconvergent fanout" because this typically occurs when two signals diverge (fanout) from a common source and reconverge at the inputs of a gate. The adjustment of timing calculations to account for reconvergent fanout is referred to as "common delay removal" because the uncertainty created by delays common to both timing paths is removed. The ability to recognize and remove the pessimistic effects of reconvergent fanout is especially important for circuits that have demanding timing requirements. Performing common delay removal on manually drawn timing diagrams is tedious and error-prone, but ignoring this effect can lead to sub-optimal designs.

Delay correlation Another effect that is rarely accounted for in digital designs that can lead to overly pessimistic margin calculations is delay correlation. Delay correlation allows the designer to specify that the degree of variation in gate delays on a single chip is smaller than the variation of gate delays across multiple chips. This lower variation in on-chip delays makes it possible for designs to meet setup and hold-time requirements that could not be met using just the min/max delay values.

Delay correlation is expressed as a percentage value. By default, delays are considered uncorrelated (delay correlation = 0 percent). One hundred percent delay correlation among a group of gates would mean that all the gates are guaranteed to have exactly the same propagation delay time. Note that this delay time could be anywhere between the min and max time of the delay parameter, but whatever the value, it would be the same of all the gates. For example, two gates with a min/max delay equaling 5 to 15 ns and a correlation of 90 percent could have delay values anywhere in the range of 5 to 15 ns, but the delays of the gates are guaranteed to be within 1 ns ((15 ns − 5 ns) × 90% = 1 ns) of each other.


WaveFormer Pro - S2.TIM - [Diagram]
File Export Edit Draw Bus Libraries View Options Window Help
Add Signal Add Bus Add Clock Add Spacer Delay Setup Sample Hold Text Marker HIGH LOW TRI VAL INVal WHI WLO HEX Zoom In Zoom Out
197.0ns 5.000ns
0ns 50ns 100ns 150ns 200ns
BUS_A_B
A
B
C
D
E
3 0 1 2 3 0 1 2 3 0
delay of 5 ns

Interactive Boolean Simulator

C = A and B

D = (A and B) delay 5

E = (A and B) or (not C delay 5)

Generate Waveforms from equations of other signals
Built in operators include: and, or, nand, nor, xor, not, delay
Define your own Boolean Operators

Figure 3. Interactive Boolean simulator example. Interactive Boolean simulation aids in top-down timing analysis

Process and temperature variations across ICs account for the increased variation in across-chip gate delays. IC manufacturers do not generally publish delay correlation values for most standard TTL and CMOS parts, but they do publish delay correlation values for clock-buffer ICs (it is more important to have correlation between clock delays than to match absolute delay for clock trees, because uncorrelated delays increase clock skew). Clock-buffer delay correlation is usually specified indirectly as a min/max delay and an on-chip delay variation often called skew. For example, a clock buffer with a delay equaling 5 to 10 ns and an on-chip skew equaling 1 ns has a delay correlation of 80 percent (1 − (1 ns / (10 ns − 5 ns)) × 100) = 80 percent).

Designers using ASICs and FPGAs will benefit the most from taking advantage of delay correlation in their designs, since most of the delays in these designs are correlated. If your gate array or FPGA manufacturer doesn't publish this information in their data books, you should ask your manufacturer directly—they probably have the information available internally. The primary reason delay correlation information hasn't been published much in the past has been the lack of EDA tools that can take advantage of it.

Correlation factors in WaveFormer Pro are specified by first creating a correlation group name and assigning it a correlation percentage. Next, all the correlated delays (all the delays on the chip) are added to the correlation group. For gate array designers, WaveFormer Pro supports a default correlation group that applies to all delays in a design. After delay correlations are specified, WaveFormer Pro automatically adjusts all timing margins in a design.

Combinatorial logic analysis Early in the top-down design cycle, after the original timing interface has been analyzed, different parts of the circuit will start to take shape. WaveFormer Pro helps you in this phase by supporting interactive Boolean simulation (see Figure 3). Designers can enter Boolean equations that describe their design and immediately assess the impact of modifying logic, state, and timing information without having to change a schematic or create simulation models. The Boolean simulator includes support for propagation and interconnect delays, allowing any combinatorial logic to be modeled. Below is an example of a Boolean equation that models an `AND` gate with an input delay of 20 ns on one input and 10 ns on the other input:

```
SIG3 = (SIG0 delay 20 ns) and
     (SIG1 delay 10 ns)
```

Temporal equations for describing DSP waveforms Many waveforms are difficult to draw and work with because of the precise edge placement necessary to convey circuit information. For instance, a waveform that alternates between a frequency of 25 MHz and 50 MHz for 10 cycles would be very tedious to attempt to draw by hand. These types of waveforms can be described in WaveFormer Pro using temporal equations. The waveform described above could be generated using the following temporal equation:

```
CK25_50 [ (20 = H 20 = L) x
20 (10 = H 10 = L) x 20 ] x 10
```

Bryan Hoyer is president of Boulder Creek Engineering (Saratoga, CA).

Donna Mitchell is vice president of marketing for SynaptiCAD Inc. (Blacksburg, VA).

To voice an opinion on this or any *Integrated System Design* article, please email your message to miker@asic.com.