Robert B. Hitchcock, Sr.
Gordon L. Smith
David D. Cheng

Timing Analysis of Computer Hardware

Timing Analysis is a design automation program that assists computer design engineers in locating problem timing in a clocked, sequential machine. The program is effective for large machines because, in part, the running time is proportional to the number of circuits. This is in contrast to alternative techniques such as delay simulation, which requires large numbers of test patterns, and path tracing, which requires tracing of all paths. The output of Timing Analysis includes "slack" at each block to provide a measure of the severity of any timing problem. The program also generates standard deviations for the times so that a statistical timing design can be produced rather than a worst case approach. This system has successfully detected all but a few timing problems for the IBM 3081 Processor Unit (consisting of almost 800 000 circuits) prior to the hardware debugging of timing. The 3081 is characterized by a tight statistical timing design.

1. Introduction

As the usage of large scale integration (LSI) in computers continues to increase, debugging of timing problems on actual hardware becomes more and more difficult. LSI creates a scarcity of probe points; and since timing problems tend to be identified serially on hardware, the increased time required to redesign and rebuild components containing many circuits can have a serious impact on schedules.

Timing Analysis (TA) is a design automation program which provides an alternative to the hardware debugging of timing problems. The program establishes whether all paths within the design meet stated timing criteria, that is, that data signals arrive at storage elements early enough for valid gating but not so early as to cause premature gating (see Fig. 1). Although Timing Analysis may be used as part of many different hardware design verification methodologies, it was developed as part of the IBM 3081 design verification methodology [1–3].

Because the usage of clocked designs has been widespread since the earliest electronic computers, the verification of path timings has always been important. Previously reported approaches have sought hardware-independent alternatives other than hand timing of questioned paths. One approach centered on the analysis of the detailed electrical properties of the components along a path, using an approach which traced all paths [4–6]. Other approaches have taken the delays of the blocks to be well defined numbers and have attempted to optimize these delays within a power constraint [7, 8]. A simulator accepting the delays as stochastic variables has been used [9] to allow the probabilistic comparisons of clock and data setting signals.

Another approach [10] used a block-oriented algorithm on limited models. In this algorithm there is a dependency on the function (AND and OR) of each block. An interesting approach which does not ignore logic function yet is operationally efficient is described in [11]. This approach does not deal with statistics, however. Two papers [12, 13], which resulted from different statistical approaches to the block model of the authors, deal with the problems of placing bounds on the distributions of the model.

The TA program is based on a block-oriented algorithm which leads to a running time essentially proportional to the number of logic blocks. Unlike the approaches based on path tracing or delay simulation, the result is a running time acceptable for even very large machines, since the block-oriented algorithm of Timing Analysis does not pay the excessive costs associated with tracing all paths or associated with the modeling of a large number of patterns in delay simulation.

The basic TA block-oriented algorithm for computation of arrival times and slacks (see Section 2) for non-probabilistic cases involving combinational logic is analogous to the Program Evaluation and Review Techniques (PERT) [14] algorithm when run non-probabilistically.

Timing Analysis is primarily a tool for aiding the engineer who is charged with producing a design to meet a specified clock cycle. The output of the TA program not only identifies any logic with timing problems, but also provides a measure of the severity of each problem. Thus, the engineer is directed to the problem areas and is given a measure of the changes required in the path timing.

Timing Analysis also incorporates methods of handling delay statistics which allow the creation of a statistical timing design with a shorter clock cycle than that of a worst case design. These are approximation methods which have provided good results on the 3081 in which the statistical terms are a significant but not overwhelming portion of the path delays.

The program is complete in the sense that the timing of all paths is effectively verified. A methodology is provided, however, to allow the engineer to alter the propagation characteristics of individual blocks when it is known that logic causes paths to never be sensitized or to meet abnormal timing criteria. Unlike delay simulation, the program does not require input of a set of test patterns, nor does it overlook paths because patterns were omitted.

Because of the characteristics just described, TA is an effective aid to engineers who must create a valid timing design, even if the machine consists of a large number of random logic circuits. In order to apply the TA methodology, the design must be a synchronous, sequential machine and it must be possible to associate explicit clock times with each storage element.

Section 2 describes the basic block-oriented algorithm of Timing Analysis. Section 3 discusses the system in which the algorithm is embedded. Section 4 explains the application of Timing Analysis and discusses the errors diagnosed by it. Section 5 provides a summary of TA's characteristics and summarizes our experiences.

2. Block-oriented algorithm

Our description of the basic algorithm starts with the treatment of late signals, that is, long paths. Figure 2 shows a simple example of combinational logic which serves as a basis for the discussion. At this point, storage elements need not be considered.

The lower portion of Fig. 2 contains a key defining the three values associated with each of the blocks on the

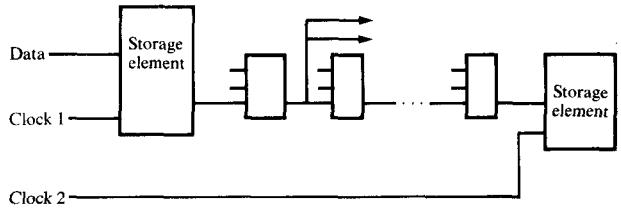

Figure 1 The timing verification problem. Make certain that every transition stored by one clock will arrive at the next storage element in time to be gated by the other clock.

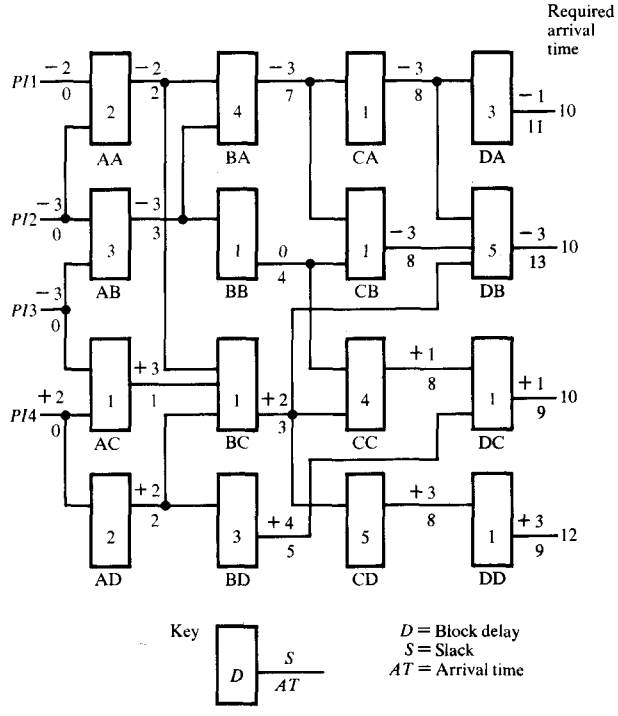

Figure 2 Forward and backward propagation.

diagram. The more complex calculations have been eliminated to permit concentrating on the timing analysis algorithm itself. For example, only one delay D is shown for a block, rather than one for rising and one for falling transitions. The number, AT, written beneath the output of each block is the maximum arrival time (in whatever units of time are chosen as appropriate). The primary input (PI) arrival times and the primary output (PO) required arrival times, which are normally derived from clock arrival times, are assumed to be explicitly given. The slack S is discussed subsequently.

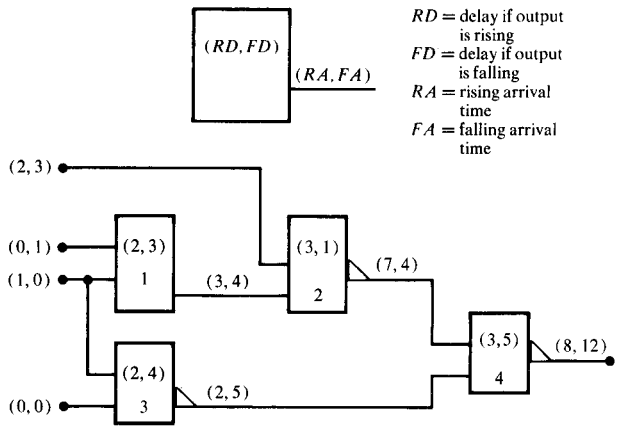

Figure 3 Using rising and falling delays.

The order in which blocks are processed by Timing Analysis ensures that every input to a block will have been processed before the block itself. The arrival time of a block is computed by

1. Establishing the arrival time at the block output for each input to the block by propagating each input signal through the block,
2. Selecting the maximum such arrival time.

Thus, blocks in the leftmost column of Fig. 2 will have an arrival time equal to the delay of the block plus the maximum arrival time of any primary input (which are all shown as zero). Block BA, which has one input at time two and the other at time three, is typical of other blocks in the figure. Three is greater, the internal delay is four, therefore seven is the maximum output arrival time for this block.

These calculations yield the outputs of blocks DA, DB, DC, and DD, at times eleven, thirteen, nine, and nine, respectively. (Notice that two signals are late and two signals are early.)

The slack S is defined as the difference between the required arrival time and the actual arrival time. The slack value is computed so that a negative number indicates a problem; that is, the signal is too late. The output of block DA arrives at time eleven, one unit late (slack = -1). Block DB is three units late (slack = -3); block DC is one early (slack = $+1$); and DD is three early (slack = $+3$). The slack information is then propagated back through the block graph, with all blocks being reprocessed in order opposite to that used for generation of the arrival times.

When each block is processed, the slacks are computed for the sources for each input, one at a time. For example, block DA has a delay of three and a required arrival time at its output of ten; therefore, the required arrival time at its input is seven (the output required arrival time minus the internal delay). With respect to this block, the output of block CA is one unit late, so the slack is minus one. Block DB has a delay of five, so the required arrival time at its input is five, and the signal feeding from block CA is three units late. Three units late (slack = -3) is worse than one unit late, so the program stores the most negative value (-3) at the output of block CA. The slack value stored at the output of the block corresponds to the slack value for the worst path going through that block.

Note that each block is processed once forward and once backward; therefore, the process runs in a time proportional to the number of blocks. Thus, a solution to a path-oriented problem has been achieved using a block-oriented technique.

From Fig. 2, one can observe a funneling of negative slack values through block BA. If BA could be replaced by a circuit having a smaller delay, all the negative slacks could be eliminated without changing the functional design at all. Clearly, there are also alternative techniques for resolving this timing problem.

In addition, notice that there are many blocks with positive slacks. These blocks could use circuits with increased delay and still be within the timing constraints [9, 10].

If we are looking for early signals caused by short paths, the above procedures are modified slightly. The arrival time at a block is computed from the minimum arrival time of any input to the block rather than the maximum). Slacks are again defined so that a negative slack indicates a problem; that is, the signal is too early.

Figure 3 elaborates the Timing Analysis principles illustrated by Fig. 2 by showing how the rising and falling delays and the inverting properties of a block are included in the computations. As the key at the top of Fig. 3 indicates, the first number inside the block is the delay if the output is rising; the second number is the delay if the output is falling; and the arrival times are kept in a two-tuple with the rising arrival time in the first position and the falling arrival time in the second position. The fact that block 2 inverts is indicated by the triangular wedge symbol at its output. This means that the rising output will be the greater of the two falling inputs (at times three and four) plus the block delay (three), so that the rising arrival time will be seven. The falling output will be the greater of the two rising inputs (at times two and three) plus the delay (one), so the falling arrival time will be four. The same calculation can be done for each of these blocks so that we can compute the rising

and falling primary output arrival times of eight and twelve, respectively. Note that if ten were the required arrival time for both the rising and falling arrival times, one would be early and the other late. For this reason the TA diagnostics distinguish between block output rising and falling arrival times, not just between blocks.

The calculation of arrival times also includes statistics. Each block delay, which has been referred to as a single-delay number up to this point, actually consists of two numbers—a nominal delay and a delay standard deviation (sigma). The calculation of a rising or falling arrival time at each block of the model involves the calculation of a nominal arrival time and an arrival-time sigma. Nominal arrival times are computed by simply adding block nominal delays to previous nominal arrival times. The arrival-time sigmas are computed by applying standard convolutions.

The arrival time actually stored at a block output is the result of propagating one of the input arrival times through the block. When the TA program is looking for long paths, the one selected is the one having the latest output arrival time. The latest arrival time is the one for which the expression

(nominal arrival time)

$$+ \text{(confidence level)} \times \text{(arrival time sigma)}$$

is the greatest. ("Confidence level" is a designer-supplied input parameter.) This expression is used to compute an arrival time whenever one is to be printed or a slack is to be computed. Conclusions about the probability of a timing failure can be drawn from the confidence level under the assumption that the distributions are Gaussian. When the program is looking for short paths, the expression is modified by replacing the plus sign with a minus sign.

3. The timing analysis system

A complete TA model consists of the combinational logic, the storage elements, and the clocks. The reference clock times for the input storage elements are known as propagate reference times; these determine when signals begin to propagate into the combinational logic. The reference clock times for the output storage elements are known as compare reference times; these determine when signals must arrive at the output of the combinational logic.

Timing Analysis views circuits as propagation elements, not as logic functions. Therefore, TA requires the explicit identification of those points in the model where tests are needed to verify that the data arrive before the clock. A means is provided for flagging those points within each storage element or at each clock gate where timing tests are required. These are necessarily points at which data and clock meet.

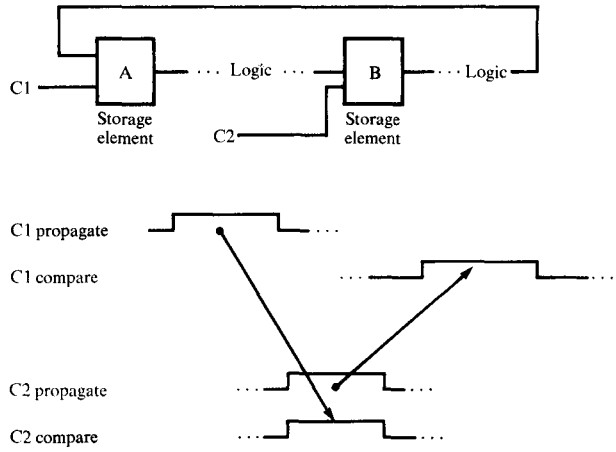

Figure 4 Clock descriptions.

Figure 4 shows two storage elements, A and B, and some combinational logic connecting them into a loop. Clock 1 (C1) gates storage element A, and clock 2 (C2) gates storage element B. The propagate and compare reference times are shown in the lower half of the figure. The path from A to B and the path back from B to A can be checked in the same Timing Analysis run without requiring a special test for each of the separate loops; the only requirement is a specification of the clocks in a consistent manner so that the signals resulting from the propagate time from C1 will be compared with the compare time of C2, and the signals resulting from the propagate time of C2 will be compared with the compare time for C1.

The TA program is a complete system in the sense that all paths are potential candidates for identification as problem paths. This follows from the TA approach which views storage elements as special functions and ignores the actual Boolean properties of other logic blocks with the exception of their inverting properties.

There may be paths, however, which are identified as problem paths by TA which can never be sensitized by logic or which meet special timing criteria at the path outputs. An example of a special timing criterion is represented by the so-called two-cycle path in which the path exceeds the cycle time but logic always ensures that two machine cycles are available for data to propagate through the path. Therefore, a means is provided whereby the engineer can insert delay modifiers at inputs or outputs of the blocks in the model which prevent the propagation of arrival times or which adjust arrival times by fixed constants (*e.g.*, minus one cycle for the two-cycle path). These delay modifiers are placed in

the model based upon an understanding of the logic by the engineer. They are processed within the context of the basic block algorithm of TA.

In certain cases, it may not be possible to place delay modifiers in a model without having an undesirable effect on other paths. For example, it may not be possible to locate any block pin in a two-cycle path where the placement of a delay modifier will not adversely affect a non-two-cycle path. It is then necessary to make one or more extra runs in which delay modifiers in the path are set to different values.

As used in the present methodology, no automatic check is provided to ensure that delay modifiers do not erroneously conceal timing problems. Thus, the completeness of TA can be compromised by the usage of delay modifiers.

When a model of the total design to be analyzed is too large for storage, it is still possible to analyze the entire model. The large model is broken up into more manageable small models that will fit within storage. In the case of the model of the IBM 3081, it has been found convenient to make the smaller models correspond to thermal conduction modules (TCMs) [15], which can contain as many as 45 000 circuits. The values computed at the primary outputs (inputs) of one model are saved and automatically fed to the primary inputs (outputs) of the other models connected to it so that the total analysis effectively processes the entire model.

4. Application of timing analysis

The timing analysis program is used by engineers as an aid in the generation of a design which meets stated timing criteria. Slacks produced by the TA program are used for pinpointing timing problems and for assessing their seriousness. The engineers make use of both slacks and arrival times as listed in the output for each block as an aid for designing changes to eliminate identified timing problems. Information is also provided by TA for establishing whether or not the timing performance goal is achievable for the machine. Additionally, TA provides information for assessing various design alternatives.

This timing analysis method is applicable to a broad class of clocked sequential machines, including, in particular, machines of large size. The domain of applicability of this technique is primarily determined by the following three characteristics of the timing analysis method:

1. *Block delay modeling*
   The accuracy of TA results is dependent upon the accuracy of the delay parameters calculated for each block on the basis of the block circuit type and the block circuit environment. Also, if block delay distributions significantly differ from Gaussian distributions, then the ability to use the confidence level to draw conclusions about the probability of a timing failure is weakened. However, the Central Limit Theorem [16] of the theory of statistics indicates that this effect is usually minimal.
2. *Timing Analysis statistical approximation*
   The algorithm of Timing Analysis is a statistical approximation. In particular, the effect of parallel paths is not reflected within the algorithm. For example, the output of a block with two independent but identical Gaussian inputs differs slightly from that of a Gaussian distribution, and the output nominal delay and standard deviation differ slightly from the values computed by TA. As a second example, the probability that all paths of a machine with many paths are free of timing problems is generally lower than the probability that any given path is free of timing problems. Thus, the assurance given by a successful TA run that each individual path in isolation meets or exceeds the desired probability of freedom from timing problems does not necessarily result in the same assurance for the entire design.
3. *Non-sensitized paths and special timing criteria*
   The number of delay modifiers and the number of multiple runs required because of non-sensitized paths or special timing criteria have an impact on the ease of model setup and results analysis. It is easier to set up TA runs and interpret the output when most of the design contains (a) only storage elements where a single pair of propagate and compare times can be assigned and (b) paths where delay modifiers are not required.

The analysis of the 3081 system utilizing statistical timing criteria has been successfully accomplished using TA. Almost all timing problems have been detected via TA prior to hardware debugging of timing [3]; the few exceptions were traced to errors in generation of block delays or to misplacement of delay modifiers within the model. The resulting design is such that any further shortening of the clock cycle would impact a very large number of paths and therefore be extraordinarily difficult to accomplish.

The output provided (slacks and arrival times) has been found to be very effective as a design aid to the engineer. Gaussian distributions have served as adequate block delay approximations, and it has been possible to generate block delays with high precision. However, block delay definition represented a major part of the development effort because of the need to achieve the shortest cycle possible with the technology. A significant part of this effort results from the inclusion of wiring effects in the delays. The slacks generated at each block as a result of the TA statistical approximations are almost always equal to, or very close to, the slacks for the worst path through that block—much closer than a worst case statistical error analysis indicates. Also,

reasonable engineering judgments adequately handle the parallel statistical effects when establishing the timing performance of the entire machine. Finally, the number of delay modifiers and multiple runs required because of non-sensitized paths and special timing criteria has been sufficiently small so as to present a minimal burden.

It is believed that the TA approach has applicability to many hardware systems; any system with the appropriate clocking is a candidate.

5. Summary and conclusions

Timing Analysis is a block-oriented algorithm for timing all paths within a clocked sequential machine; and since it is a block-oriented algorithm, it is practical for very large machines. It effectively checks all paths and output arrival times and slacks for each block. Statistical approximation and methods for dealing with logic with special timing characteristics are both built into the TA program.

The timing analysis approach has been successful on the 3081 because

- It has found all but a few timing problems prior to the hardware debugging of timing.
- It has successfully allowed the generation of a tight timing design.
- The basic algorithm is much more efficient than delay simulation or path tracing for finding timing problems.
- The presentation in the output of slacks and arrival times for each block presented the engineers with meaningful parameters for assessing and understanding each timing problem.

Within the domain of applicability (see the previous section), the TA approach described in this paper is believed to be effective for solving the timing design problem for many LSI machines.

6. Acknowledgments

The authors acknowledge the significant contribution of W. Donath in developing (along with one of the authors—Hitchcock) a predecessor experimental block-oriented system [17] at the IBM Thomas J. Watson Research Center. This system, which included storage elements, was designed to analyze critical paths.

The authors also acknowledge the contributions of the following to the early formation of TA: W. Donath, R. Bahnsen and H. Ofek. The management support of A. Fitch, M. Monachino, B. Golnek, and R. Russo, and the management of the program implementation by A. Brown were all key to the final success. It is not possible to name all the many others on the 3081 project and in the design automation (EDS) and technology groups who contributed to TA. These functions are located in Poughkeepsie, East Fishkill, and Endicott.

References

1. M. S. Pittler, D. M. Powers, and D. L. Schnabel, "System Development and Technology Aspects of the IBM 3081 Processor Complex," *IBM J. Res. Develop.* **26,** 2–11 (1982, this issue).
2. R. N. Gustafson and F. J. Sparacio, "IBM 3081 Processor Unit: Design Considerations and Design Process," *IBM J. Res. Develop.* **26,** 12–21 (1982, this issue).
3. Michael Monachino, "Design Verification System for Large-Scale LSI Designs," *IBM J. Res. Develop.* **26,** 89–99 (1982, this issue).
4. D. J. Pilling and H. B. Sun, "Computer-Aided Prediction of Delays in LSI Logic Systems," *Proceedings of the 10th ACM/IEEE Design Automation Workshop,* Portland, OR, 1973, pp. 182–186, and discussion held after presentation of paper.
5. Ryotaro Damikawai, Minoru Yamada, Tsuneyo Chiba, Kenichi Furumaya, and Yoji Tsuchiya, "A Critical Path Delay Check System," *Proceedings of the 18th ACM/IEEE Design Automation Conference,* Nashville, TN, June 1981, pp. 118–123.
6. M. A. Wold, "Design Verification and Performance Analysis," *Proceedings of the 15th ACM/IEEE Design Automation Conference,* Las Vegas, NV, June 1978, pp. 264–270.
7. A. E. Ruehli, P. K. Wolff, Sr., and G. Goertzel, "Analytical Power/Timing Optimization Technique for Digital System," *Proceedings of the 14th ACM/IEEE Design Automation Conference,* New Orleans, LA, 1977, pp. 142–146.
8. B. J. Agule, J. D. Lesser, A. E. Ruehli, and P. K. Wolff, Sr., "An Experimental System for Power/Timing Optimization of LSI Chips," *Proceedings of the 14th ACM/IEEE Design Automation Conference,* New Orleans, LA, 1977, pp. 147–152.
9. Brengt Magnhagen, "Practical Experience from Signal Probability Simulation of Digital Designs," *Proceedings of the 14th ACM/IEEE Design Automation Conference,* New Orleans, LA, 1977, pp. 216–219.
10. T. I. Kirkpatrick and N. R. Clark, "PERT as an Aid to Logic Design," *IBM J. Res. Develop.* **10,** 135–141 (1966).
11. T. M. McWilliams, "Verification of Timing Constraints on Large Digital Systems," *Proceedings of the 17th ACM/IEEE Design Automation Conference,* 1980, pp. 139–147.
12. Arthur Nádas, "Probabilistic PERT," *IBM J. Res. Develop.* **23,** 339–347 (1979).
13. A. Nádas, "Random Critical Paths," *IEEE International Symposium on Circuits and Systems Proceedings,* Houston, TX, 1980, pp. 32–35.
14. D. G. Malcolm, J. H. Roseboom, C. E. Clark, and W. Fagar, "Application of a Technique for Research and Development Program Evaluation," *Oper. Res.* **7,** 647–669 (1959).
15. A. J. Blodgett and D. R. Barbour, "Thermal Conduction Module: A High-Performance Multilayer Ceramic Package," *IBM J. Res. Develop.* **26,** 30–36 (1982, this issue).
16. W. Feller, *Probability Theory and Its Application,* Vol. 1, John Wiley & Sons, Inc., New York, 1950.
17. W. E. Donath and R. B. Hitchcock, "Method for Determining the Characteristics of a Logic Block Graph Diagram to Provide an Indication of Path Delays Between the Blocks," U. S. Patent 4,263,651, 1981.

Received September 5, 1980; revised August 18, 1981

Robert B. Hitchcock, Sr., is located at the IBM General Technology Division laboratory, Endicott, New York 13760. Gordon L. Smith is located at the IBM Data Systems Division laboratory and David D. Cheng at the IBM General Technology Division laboratory, both in Poughkeepsie, New York 12602.