A CLOCKING DISCIPLINE FOR TWO-PHASE DIGITAL SYSTEMS

David Noice, Rob Mathews, and John Newkirk

Dept. of Electrical Engineering
Stanford University

ABSTRACT

A two-phase clocking discipline for digital ICs can guarantee freedom from clock skew, critical races, and other timing problems. In this paper, we present such a discipline: a notation, composition rules, and early results from verification tools.

INTRODUCTION

Sooner or later a designer of digital circuits must face the problems of clock skew, critical races, and hazards. Generally he does a timing analysis that uses the tedious and error-prone process of calculating delays and deriving timing diagrams to check for errors. Alternatively, he may perform timing simulations to test a circuit for a representative set of inputs, a process that may leave errors undetected.

This paper describes a different approach: to guarantee correct operation despite uncertainty about delays in the circuit. The result is a clocking discipline that deals with timing abstractions only. It is not based on delay calculations; it is only concerned with correct, synchronous operation at some clock rate*.

The examples we present are drawn from nMOS IC design. However, most of the ideas are applicable to any two-phase digital system.

THE CLOCKING DISCIPLINE

This clocking discipline is a method for producing circuits that are free from timing errors. It consists of two main parts: a notation for signal types, and composition rules for legal circuits. In effect, the notation and rules define a syntax of clocking-correct circuits that can be checked by auditing tools, analagous to the syntax and type checking done for computer programs.

A design that follows the discipline is guaranteed to have no errors due to clock skew, critical races, or hazards. A more accurate simulation results: a circuit that is free from timing errors can be simulated without fear of faulty race predictions.

This research was sponsored by DoD ARPA under contract DA 903-79-C-0680.

*This philosophy is also shared by self-timed asynchronous circuits [1,2].

Assumptions

The clocking discipline rests on three basic assumptions:

1) All input and initial values are digital.
2) The system is two-phase and synchronous.
3) All logic and wiring delays are positive and bounded (less than some fixed maximum value). No knowledge of relative delays among circuit paths is used.

The first assumption is important. The discipline guarantees continued digital operation, but it needs an initial starting point and digital inputs.

The second requirement has two bases. First, a two-phase clocked system is a very practical method for designing most MOS ICs[1]. Second, and more importantly, it can be shown that two phases are needed to avoid critical races if relative delays between different paths are unknown[3].

The final assumption arises from the goal of immunity to clock skew and races. To achieve this goal the discipline must insure a design will work under all possible circuit delay conditions. Accordingly, we must not assume any given circuit path is faster or slower that any other path. This assumption is especially important for IC designs where wiring delays can potentially be a dominant factor and are unknown until layout is finished.

This assumption gives rise to another constraint as well. To insure finite delays and digital signals the class of allowable combinational logic must be restricted. Combinational logic cannot have any unclocked feedback loops. Such loops can potentially oscillate, with no bounded settling time.

Notation

A notation of signal types and clocks derives from these assumptions. To motivate the notation, consider the dynamic latch shown in Figure 1. The input signal must satisfy timing requirements with respect to the clock (ϕ_1) to ensure that the pass transistor will properly latch the input value. In particular, the input must be valid in a window around the falling edge of ϕ_1, providing setup and hold times required by the latch. Such a signal becomes valid on phase 1, denoted w_{ϕ_1}.

Figure 2 illustrates a representative set of clocking types. Clock types (ϕ and qualified ϕ) are signals that establish the time and sequence

CH1813-5/82/0000/0108$00.75 © 1982 IEEE

Reprinted from IEEE INTERNATIONAL CONFERENCE ON CIRCUITS AND COMPUTERS ICCC '82, September-October 1982

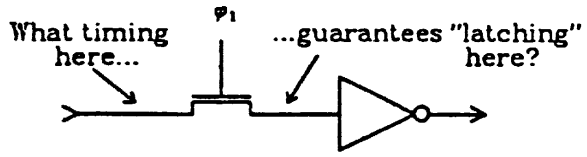

Figure 1. A φ_1 clocked dynamic latch.

references for the data types (valid φ and stable φ). The two sets (clock and data) are separate. A clock signal is never a valid data signal. This reflects the fact that a clock cannot satisfy the setup and hold time requirements with respect to another clock without a race.

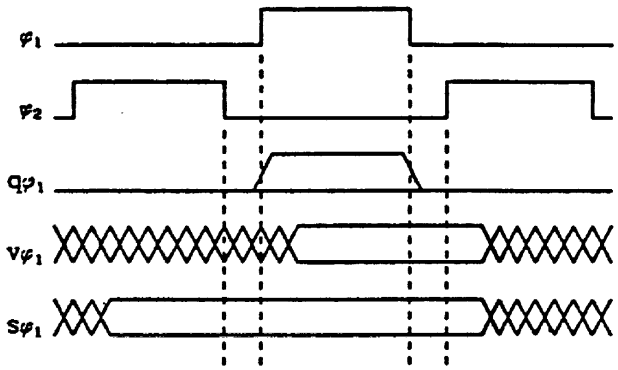

Figure 2. Some representative clocking types, showing the sequence of events. Q, v, and s stand for qualified, valid, and stable.

Note that all the signal types are invariant given arbitrary but bounded skew and delay. For example, a $v\varphi_1$ signal is valid some delay after φ_1 rises. By stretching the φ_1 period, we can guarantee that the $v\varphi_1$ signal is settled before the setup time required by a φ_1 latch. Similarly, by stretching the gap between φ_1 falling and φ_2 rising, we can guarantee sufficient hold time in the face of clock skew.

Composition Rules

Now let us investigate some of the properties of these signals. With a little thought we can see that none of the data signals change types when they go through combinational logic. Furthermore, combinational logic that has all its inputs $s\varphi_1$ will have outputs that are $s\varphi_1$ (see Figure 3a).

A memory element formed by a clocked pass transistor and inverter *will* change the signal type (see Figure 3b). If the input is $v\varphi_1$ or $s\varphi_1$, the output is $s\varphi_2$. This is the foundation of the two-phase clocking discipline. φ-clocked storage nodes are time-bounding elements. They are the points where values must have settled in time, and where values are held (and in some sense converted) for the opposite phase.

Note that an $s\varphi_1$ signal may not change value during the entire φ_1 period. When an $s\varphi_1$ signal is ANDed with a φ_1 or $q\varphi_1$ clock (see Figure 3c), we get an output that fits the definition of a $q\varphi_1$ clock. The output cannot glitch. It either mimics a φ_1 clock or remains low as shown in Figure 2.

The properties described above form composition rules for building clocking-correct circuits. They are the basic rules we can use to create a syntax of legal circuits.

A Simple Clocking Discipline

A simple clocking discipline can be constructed with only the six timing types φ_1, φ_2, $q\varphi_1$, $q\varphi_2$, $s\varphi_1$, and $s\varphi_2$ and the composition rules shown in Figure 3. A legal circuit comprises an interconnection of memory elements and combinational logic such that all inputs to memory elements receive signals of the proper types: clocks to the clock inputs and stable data signals to the data inputs. These structuring rules are equivalent to requiring that the circuit can be two-colored[4], one color for φ_1 and $s\varphi_1$ paths, another for φ_2 and $s\varphi_2$. All together, the notation and rules define a syntax of clocking-correct circuits (see Figure 4).

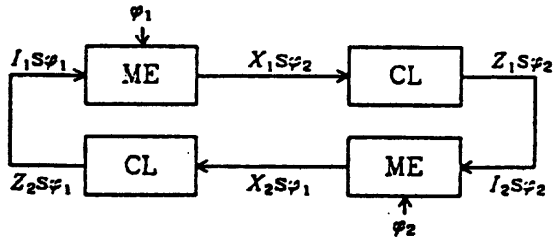

Figure 4. A typical two-phase finite state machine show the syntax and consistency of the clocking types and composition rules.

An analysis of the small example in Figure 5 will give the reader some feel for the utility of the

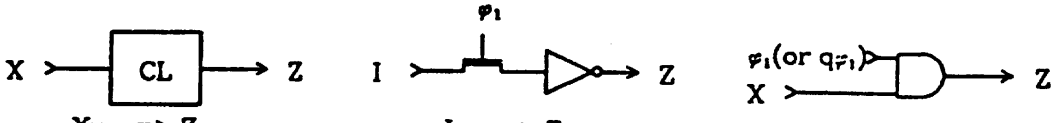

Figure 3. Composition rules for a) combinational logic with bounded delays, b) memory elements, and c) qualified clocks.

clocking discipline. The figure shows a memory element with storage at node 3. The node is periodically driven through T1 or T2. Will this circuit work?

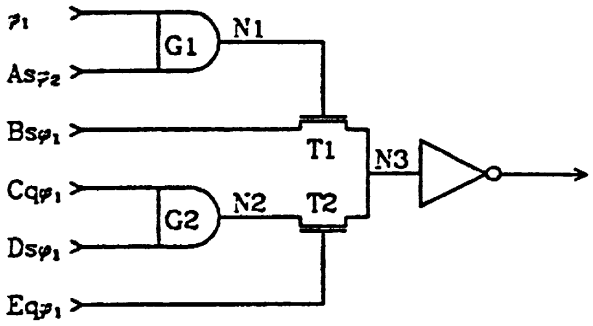

Figure 5. An example for analysis using the clocking discipline.

Let us examine the top branch first. G1 has two inputs: φ_1 and Asφ_2. It does not fit any of the composition rules given in Figure 3, so it is illegal. An sφ_2 signal can change value several times during the φ_1 period. The output at N1 is a "glitchy" qualified clock that can destroy the value stored at N3.

In the bottom branch, the inputs to G2 do fit the rule shown in Figure 3c, so N2 is qφ_1. However, a qφ_1 signal is not allowed as input to T2: only vφ_1 and sφ_1 signals may be latched by a memory element. The race between the falling edges of Eqφ_1 and Cqφ_1 through G2 makes the value stored at N2 indeterminate.

It should be clear that if signals A and C were both sφ_1, the circuit would work and fit within the clocking discipline. This example shows some of the usefulness of the discipline. The clocking discipline becomes increasingly valuable as the circuits analyzed become bigger and more complex.

The simple discipline described so far is too restrictive for MOS IC designs. In practice we have found it necessary to include such basic techniques as precharging, sharing of bus lines across clock phases, and feedback for static memory elements. These techniques have introduced new pitfalls (such as charge-sharing), but they are caught by the auditing tools described next.

Auditing Tools

The clocking discipline can help a designer produce cleaner, clocking-correct circuits, but mistakes will still happen. Auditing tools can eliminate those mistakes. These tools look for such errors as clocks used as data, data signals used on the wrong phase, *etc.* They also flag charge sharing, feedback loops, and the like for extra attention from the designer.

Once a design has passed the auditing tools it should be completely free from clock skew and race problems. A unit delay simulator[3] will now adequately predict the outcome of any races because they are not critical races—they have no effect on the final output.

It would be nice if the circuit could also be *exactly* modeled by a switch simulation. Unfortunately, there are still a few remaining problems, *e.g.*, drive fight (a multiplexor that allows two gate outputs to fight each other), and charge leakage off of a storage node.

The clocking discipline and auditing tools could preclude these problems, but the added restrictions would be costly. For example, we could eliminate the danger of charge leakage by requiring all memory elements to be refreshed every clock cycle. Currently, the designer is responsible for avoiding these problems. A switch simulator that could detect these conditions, combined with the clocking discipline, could guarantee a completely accurate functional simulation of a design's operation.

Results

The clocking discipline has been tested on eight student and research IC projects. During the design phase these projects made use of preliminary clocking discipline concepts, but the auditing tools were not yet available. Furthermore, time constraints prevented all but a few from doing even minimal simulation before fabrication.

During the Winter quarter of 1982 the fabricated ICs were received and complete electrical tests and simulations were done. At this point the paper designs were also checked with the auditing tools. The results are shown in Table 1.

Result	number
Clocking, simulation and chip ok	3
Non-fatal clocking error; simulation and chip ok	2
Fatal clocking error; simulation could detect	2
Fatal clocking error; simulation could not detect	1

Table 1. Results of testing eight IC designs with auditing tools, simulation, and electrical tests.

The experience with these designs has demonstrated that the clocking discipline is an effective and practical technique. The auditing tools are particularly useful because many errors can be caught at an early stage, before simulation.

It is interesting to note the types of errors found. In two cases non-fatal timing errors were flagged that pointed out some student misconceptions about precharged logic. Fortunately the mistakes only affected the performance of the chips rather than their operation. Three designs had fatal timing errors. In two of these cases additional simulation could have caught the error.

..owever, the third case had previously gone through extensive simulation and was expected to work. This design had a critical race. The simulator predicted it would work, but the actual chip failed. After the fact, the auditing tools pinpointed the location and nature of the race. In the of the race. In the future, the auditing tools will be used before simulation and fabrication.

The auditing check only takes about 5 to 10 CPU seconds on a VAX 11-780 for a design with 1000 transistors. The run time increases linearly with the number of transistors.

CONCLUSION

The different facets of the clocking discipline have all proven to be useful. First, the signal notation is a valuable aid in teaching, analyzing and designing two-phase circuits. Second, the composition rules and auditing tools guarantee that a design will be free from timing errors due to clock skew, races, or hazards as well as flagging other potential problems for extra attention.

The clocking discipline is a practical technique that guarantees clocking-correct designs, and more accurate simulations. The result will be reduced design time, fewer design iterations, and cheaper working designs.

REFERENCES

1. Mead, C.A. and Conway, L., *Introduction to VLSI Systems*, Reading: Addison Wesley, 1980.
2. Miller, R., *Switching Theory*, vol. 2, New York: Wiley, 1965, chapter 10.
3. Noice, D., Ph.D. thesis, in preparation.
4. Baldwin, B., "A representation for timing and topology", *Internal memo, LSI Systems Area*, Xerox PARC, January 1979.
5. Baker and Terman, "Tools for Verifying Integrated Circuit Designs", *Lambda*, Palo Alto, Fourth Quarter 1980.