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, analogous 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.

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. 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.

The final assumption arises from the goal of immunity to clock skew and races. To achieve this goal the discipline must define a design which will work under all possible circuit delay conditions. Accordingly, we must not assume any given circuit path is faster or slower than 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 (s) 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 s, providing setup and hold times required by the latch. Such a signal becomes valid on phase 1, denoted s1.

Figure 2 illustrates a representative set of clocking types. Clock types (c and qualified c) are signals that establish the time and sequence
What timing here... guarantees "latching" here?

Figure 1. A $\varphi_1$ clocked dynamic latch.

references for the data types (valid $\varphi$ and stable $\varphi$). 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.

$p$

$q$

$v$

$s$

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, if a $v_p$ signal is valid some delay after $\varphi_1$ rises. By stretching the $p_1$ period, we can guarantee that the $v_p$ signal is settled before the setup time required by a $p_1$ latch. Similarly, by stretching the gap between $p_1$ falling and $p_2$ 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 $s_p$ will have outputs that are $s_p$ (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 $v_p$ or $s_p$, the output is $s_p$. This is the foundation of the two-phase clocking discipline. $\varphi$-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 $s_p$ signal may not change value during the entire $p_1$ period. When an $s_p$ signal is ANDed with a $p_1$ or $p_2$, clock (see Figure 3c), we get an output that fits the definition of a $q_{p_1}$ clock. The output cannot glitch. It either mimics a $p_1$ 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 $p_1$, $p_2$, $q_1$, $q_2$, $s_p$, and $s_{p_2}$ 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, one color for $p_1$ and $s_p$, paths, another for $p_2$ and $s_{p_2}$. 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.
Let us examine the top branch first. G1 has two inputs: \( p_1 \) and \( s_{p2} \). It does not fit any of the composition rules given in Figure 3, so it is illegal. An \( s_{p2} \) signal can change value several times during the \( p_1 \) 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 \( q_{p1} \). However, a \( q_{p1} \) signal is not allowed as input to T2: only \( v_1 \) and \( s_{p1} \) signals may be latched by a memory element. The race between the falling edges of \( Eq_{p1} \) and \( Cq_{p1} \) through G2 makes the value stored at \( N2 \) indeterminate.

It should be clear that if signals A and C were both \( s_{p1} \), 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 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 multiplexer 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.

<table>
<thead>
<tr>
<th>Result</th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clocking simulation and chip ok</td>
<td>3</td>
</tr>
<tr>
<td>Non-fatal clocking error: simulation and chip ok</td>
<td>2</td>
</tr>
<tr>
<td>Fatal clocking error: simulation could detect</td>
<td>2</td>
</tr>
<tr>
<td>Fatal clocking error: simulation could not detect</td>
<td>1</td>
</tr>
</tbody>
</table>

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.
...however, 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