STATIC TIMING ANALYSIS FOR SELF RESETTING CIRCUITS

Vinod Narayanan
Barbara A. Chappell *
Bruce M. Fleischer

IBM T.J Watson Research Center
Yorktown Heights, NY 10598

ABSTRACT

Static timing analysis techniques [1, 2] are widely used to verify the timing behavior of large digital designs [11] implemented predominantly in conventional static CMOS. These techniques, however, are not sufficient to completely verify the dynamic circuit families now finding favor in high-performance designs [11]. In this paper, we describe an approach that extends static timing analysis to a high-performance dynamic CMOS logic family called self-resetting CMOS (SRCMOS *) [3, 4]. Due to the circuit structure employed in SRCMOS, designs naturally decompose into a hierarchy of gates and macros; timing analysis must address and preferably exploit this hierarchy. At the gate level, three categories of constraints on pulse timing arise from considering the effects of pulse width, overlap, and collisions. Timing analysis is performed at the macro level, by a) performing timing tests at macro boundaries and b) using macro-level delay models. We define various macro-level timing tests which ensure that fundamental gate-level timing constraints are satisfied. We extend the standard delay model to handle leading and trailing edges of signal pulses, across-chip variations, tracking of signals, and slow and fast operating conditions. We have developed an SRCMOS timing analyzer based on this approach; the analyzer was implemented as extensions to a standard static timing analysis program, thus facilitating its integration into an existing design system and methodology.*

1 INTRODUCTION

Static timing analysis is a method to verify the timing behavior of digital systems [1, 2]. In this approach, the timing behavior of the system is characterized in a pattern-independent fashion. Static timing analysis is significantly faster than traditional timing simulation, and is hence used extensively for the timing verification of large digital designs. Timing analysis tools for static CMOS circuit families are well developed and readily available commercially.

Dynamic CMOS circuits have inherent performance advantages over static CMOS; hence they are being widely used today in high performance designs. There are different families of dynamic circuitry; self-resetting CMOS (SRCMOS) is one such high-performance circuit family [3, 4, 5].

Currently available timing analysis tools do not yet deal effectively with dynamic circuit constraints. In this paper, we present our extensions of static timing analysis concepts to verify SRCMOS circuits. Our approach is implemented as an add-on module to an existing static timing analysis tool, and fits into existing design methodologies.

We begin the with a discussion of the basic operation and timing requirements of SRCMOS gates. We discuss the SRCMOS *macros*, a natural intermediate level in large designs. Our overall approach to timing is based on modeling the behavior of macros, and applying static timing analysis techniques at the macro boundaries. Many of the timing considerations we discuss for SRCMOS extend to other high-performance dynamic CMOS families, such as domino CMOS.

2 SELF-RESETTING CMOS LOGIC

2.1 Static vs Dynamic Logic Circuit Families

To operate correctly, a static CMOS gate must produce an output at a valid level and which is logically correct for any combination of valid input levels. The production of valid levels is usually trivial in static CMOS (with some complication if ratioed logic is used). As a result, logical verification can be treated as a purely topological problem, so that logic and timing are generally analyzed completely independently. In static CMOS, timing affects logic only when data is captured at latches. Therefore, all timing problems can be solved by adding delay at latch inputs, either to the data for fast paths or to the clock for slow paths.

While static circuits use *push-pull* configurations, dynamic circuits employ *uni-polar* switching. That is, logic inputs can cause transitions in only one direction. Uni-polar switching has several consequences. First, all circuits must be *reset* from one cycle to the next. Second, information is carried by the presence or absence of pulses, not a steady-state voltage. Uni-polar switching also allows performance in the active direction to be traded off against noise margin.

A dynamic CMOS gate will operate correctly only if certain constraints on device size and signal timing,

*Currently at Intel Corporation, Hillsboro, OR 97124 as well as topology, are met. For example, input pulses that are to act together must overlap by an amount that depends on the gate's device sizes and topology; they are not effective if they occur in the same cycle but without enough overlap. Timing problems can give rise to unexpected pulses or mask required pulses. In other words, timing problems can lead to logic errors which cannot be solved by adding delay to clock or data inputs to latches.

2.2 SRCMOS Circuit Fundamentals

The basic operation of SRCMOS circuits can be understood by referring to Figure 1 and Figure 2. A non-inverting SRCMOS logic gate, shown in Figure 1, is similar to a domino CMOS gate. The key difference between SRCMOS and domino is only implicit in the figure; the gate is reset by locally-generated reset signals rather than a global clock. Less important, but typical of SRCMOS, are the presence of a weak *static evaluate* pfet for testability [5], and the absence of a ground-interrupt device.

The gate has active-high pulsed inputs and output. It's (non-inverting) logic function depends on the topology of the nfet tree. If inputs go high in the right combination and at the same time, a conducting path is created from node (*TL*) at the top of tree to ground. If this conducting path exists for some minimum time, *TL* falls and the output rises to create the leading edge of the output pulse. Once enough input pulses end, the *RL* input can arrive (fall) to reset *TL*, and once *TL* is reset high, the *RH* input can arrive and terminate the output pulse.

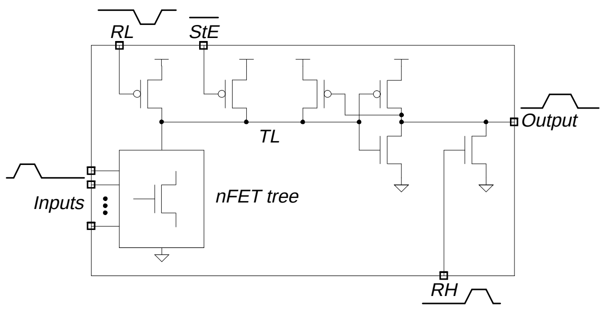


Figure 1: Basic SRCMOS Gate

In SRCMOS, there is a natural hierarchy level — the macro — above the gate level. An SRCMOS macro includes a number of SRCMOS gates and a local reset generator, as shown in Figure 2. The reset generator is triggered by one or more of the pulsed signals in the macro. Triggering options include simple OR-ing of dual-rail signals, quorum or majority circuits, and interlocking of signals from multiple paths. In any case, the reset generator must be triggered whenever the macro is activated. A delay chain, started by the trigger circuit, generates the required reset signals. One of these signals resets the trigger, so that the reset signals are self-terminating in normal operation. The reset generator may also include static inputs to force all reset signals on or off for test modes [5].

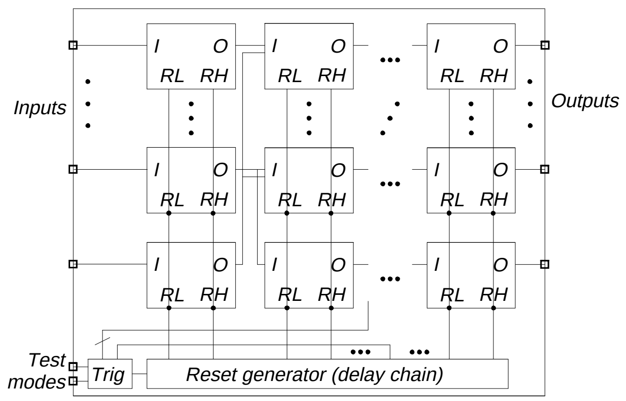


Figure 2: SRCMOS Macro

In summary, the macro as a whole is triggered by the leading edges of its input data pulses, some of the gates evaluate in sequence (as in domino logic), some of the outputs produce pulses, and the whole macro returns to the standby state until new pulses arrive the following cycle. Note that this process takes place without direct dependence on a global clock signal. In SRCMOS logic, a single edge of a single lightly-loaded clock launches data from the registers; unlike domino logic, the reset time is hidden in the cycle by the internal operation of the logic macros.

There are other approaches to improving cycle time and reducing clock loading through local reset generation. For example, in self-timed rings [10] reset signals are returned from later gates in a hand-shaking scheme. In delayed-reset logic [8], [9], reset and logic signals are generated together within the logic gates, and both propagate forwards.

Our design methodology for custom SRCMOS macros uses careful planning and tuning of logic and reset delays within a macro. Of all the gate-level timing constraints one could write for a macro, nearly all are satisfied by this plan. The intra-macro timing plan and a small set of external timing constraints are developed together; macro verification is performed within the bounds of the external constraints. Static timing analysis at the macro-to-macro level then verifies the consistency of the external constraints.

3 STATIC TIMING ANALYSIS FOR SRCMOS

Typically, static timing analysis programs propagate arrival times of each logic level independently, from separate timing equations for rising and falling transition. Also, timing checks (usually called *tests*) are typically performed only at latch boundaries and primary outputs. Most timing analyzers allow them to be specified at other points, but are not optimized to check a large number of points throughout the circuit. The tests ensure that static CMOS signals settle to the correct logic level before getting latched.

For SRCMOS, it is important to check the timing characteristics at many different points of interest. Usually, the tests are specified at all macro boundaries. The tests cannot be specified independently for each transition; they are usually based on pulse-width related constraints. Although the tests are specified at the macro boundaries, they are designed to support the fundamental gate-level constraints. These gate level constraints ensure that a) pulses occur when required, and b) no unexpected pulse occurs. The gate-level timing constraints are discussed in detail in Section 3.1. In standard domino logic [7], analysis of constraints at the gate level is avoided by connecting a global clock phase to reset and ground-interrupt devices in every gate. As a result, the trailing edges of all pulses are tied to global timing [11], and analysis of timing constraints is pushed into cycle-time analysis.

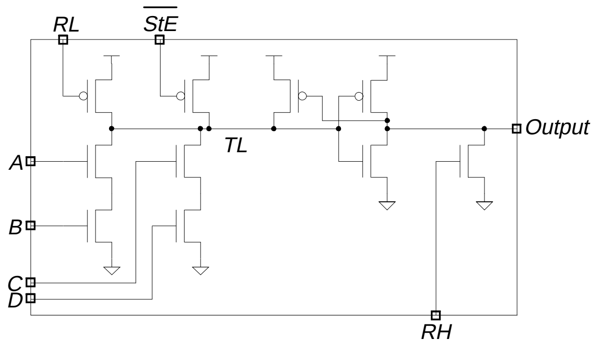
3.1 Timing Constraints For SRCMOS Gates

The design philosophy we follow for SRCMOS circuits is that the pulse widths within the macro are tuned based on macro requirements, while the output pulse widths are tuned based on global interconnect requirements. In this scenario, it is not feasible to design receivers for a wide variation in pulse width and arrival times without a significant degradation in performance. Since dynamic circuits are used in situations demanding high performance, it is preferrable to enforce a timing discipline at the macro boundaries than to design receivers for a wide variation in pulse widths.

Following this circuit design approach, we must make sure that several categories of timing constraints are satisfied at each SRCMOS gate. We then specify tests at macro boundaries that ensure that the gate level constraints are satisfied. Broadly, we may categorize these constraints into a) Pulse overlap constraints, b) Pulse width constraints and c) Collision avoidance constraints. Each of these categories is dealt with in more detail below.

3.1.1 Pulse Overlap Constraints

For pulse-based circuits to work correctly, the pulses arriving at different inputs must be active for a specified common period of time. This is illustrated by the schematic for the AND-OR gate shown in Figure 3. The gate is activated by pulses on some combination of inputs to form a path from the node *TL* to ground, for some minimum time. Looking at the timing diagram in Figure 3, in cycle 1 the output is produced by inputs A and B. In cycle 2, there is no output, because A and B do not have enough overlap. In cycle 3, the output is produced by inputs C and D; A need not overlap with C or D. We would therefore define two pulse overlap constraints, one on A and B, the other on C and D. To avoid an explosion of constraints for a complicated gate, we could define instead one conservative constraint per gate, requiring that all pulses that do arrive overlap by a given amount.



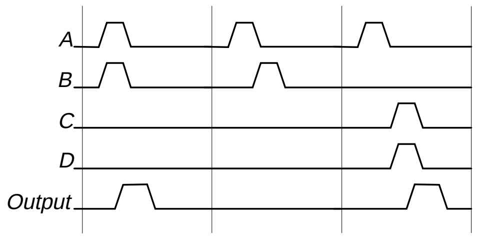


Figure 3: SRCMOS AND-OR Gate

In practice, timing tests are specified for macros, not gates. We may use some pulse overlap constraints at a macro's inputs to ensure pulse overlap at its internal gates, without exposing that internal structure to timing analysis. Within a macro, however, we may avoid static timing analysis in favor of a pattern-sensitive approach. In that case, it may be quite feasible to check the overlap of just relevant inputs and allow non-overlap for others without an explosion of explicit gate-level rules. For example, a simulation-based verification of the gate in Figure 3 should not flag cycle 3 in the timing diagram as an error.

Deriving the pulse overlap constraints for a macro's inputs is part of our macro-design methodology. If the constraints are getting unwieldy or impossible to satisfy, two techniques can help ensure overlap at gates while allowing simple or even no overlap constraints at the macro inputs. First, the assignment of reset signals to gates within a macro provides some levelization and synchronization to the logic pulses' trailing edges. Using that levelization in planning the timing of the leading edges can help ensure solid overlaps at each gate. Second, transparent latches at the macro inputs can stretch early inputs and can be used to avoid any external overlap constraints.

3.1.2 Pulse Width Constraints

For correct operation of circuits, input signals must have a certain minimum pulse width. Pulse width constraints are generally necessary when a signal is the only input to a gate (*i.e.*, a buffer or pulse stretcher). If pulse overlap constraints are present, they create implicit pulse width constraints. If not, or if wider pulses are required, then explicit pulse width constraints must be given.

3.1.3 Collision-Avoidance Constraints

As described earlier, an SRCMOS gate must be reset after it has evaluated. During this reset phase, the input signals must remain quiet so that they do not collide with the reset process. This can be illustrated by looking at the example in Figure 1.

If the RL pulse were to overlap or *collide* with the input pulses, a low-resistance path from would be created from Vdd to $Ground$, and the gate's average power dissipation could increase dramatically. Such collisions can result from wide or late input pulses as in cycle 1 in Figure 4, or from early input pulses colliding with the previous cycle's reset, as in cycle 3 in the figure. A ground-interrupt device between the nfet tree and ground, with its gate connected to RL, as in domino, would prevent overlaps from creating undue power dissipation. Ground-interrupt devices have drawbacks, however: increased forward delay, area, and reset loading. Even with a ground-interrupt device, the input pulse must still end before RL does, to avoid spurious double pulses at the output of the gate.

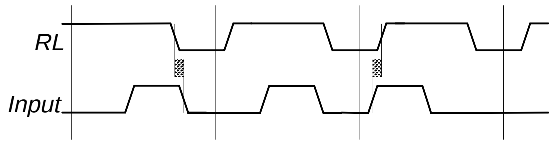


Figure 4: Pulse Collisions between Reset and Input

Within a macro, tuning the delays of the reset chain and the reset devices within gates prevents most potential collisions. Reset delays may be de-tuned relative to forward delays, to reduce area, reset power, and loading on the forward path. In such cases, signal pulse-widths grow from stage to stage; judicious use of ground-interrupt devices in some logic stages can counteract this pulse growth.

Judicious use of ground-interrupts in an early stage, together with advancing or delaying early resets, can also provide extra margin against collisions between leading edge of the input and the preceding reset signal, or the trailing edge of the input and the following reset signal.

3.2 Component Timing Models

Timing analyzers use *timing* or *delay* models to estimate the delay through different components, based on the electrical environment in which the component is instantiated. These timing models are further parametrized for variations in fabrication processes and operating conditions. Timing models for SRCMOS components require sufficient information to ensure that a) no unwanted pulses occur, and b) all required pulses do occur.

First, we need to chose the granularity of the leaf level components for delay modeling. The timing equations which describe the leaf level components are derived through extensive simulations followed by curve fitting. Clearly, some accuracy is lost for each component that is thus modeled. These inaccuracies can add up for all the components that are on a path. In most semi-custom approaches, individual cells which implement primitive functions (such as latches, AND gates, OR gates etc..) are the leaf level components. This could produce an error of 20% or more between the estimated and actual timing results.

To estimate the worst case delay for a given component, each component is modeled for its typical delay under worst case patterns; in typical operation, it is unlikely that all the components are stimulated by their worst case pattern in the same cyle. In general, accurate analysis of critical paths can't be achieved by static timing analysis at the end of the design loop, but requires careful planning, design, control of parasitics, *and* good analysis.

Gate delay models for standard cell libraries are usually characterized using a single input switch model. Depending on the nature of the gate, the slow path or fast path delays are significantly affected by simultaneous switching at the inputs. For SRCMOS circuits, the component models must account for simultaneous switching to some degree. Simultaneously switching inputs can affect the delay from a gate's critical input to its output by $\pm 20\%$ in one-stage static gates, or $\pm 10\%$ in two-stage dynamic gates.

There can be significant delay variations across any given chip. These variations arise from two different sources; across-chip variations in L_{eff}, and across-chip variations in operating conditions. For example, in a 2.5 V technology, transient across-chip supply-voltage variations of $\pm.25$ V are likely. This variation in supply voltage can lead to a corresponding change of about 10% in gate delay.

One also needs to decide how the trailing edge of the pulse is propagated. It is not desirable for the trailing edge does depend on the trailing edge of the input pulses; variations in pulse width would accumulate over several stages, and would soon lead to problems with pulse intersection or collision in later stages.

Based on the above considerations, we made the following choices for our timing models.

- In order to improve accuracy, we chose to model the components at the granularity of large macros. For example, the incrementer described in [4] would be characterized through extensive simulation of cross sections, and then delay models would be generated for that macro from the simulation results.

- Separate timing equations are used for fast (*early*) path and slow (*late*) path analysis. Both slow and fast timing models are generated with typical, worst-case and best-case process conditions. Since the same analysis is done at each process condition, we assume the typical process in the rest of this paper.

- The slow delay (D_s) equation coefficients are generated assuming the slowest operating conditions for a typical process with the slowest excitation pattern. The determination of the slowest excitation pattern is the responsibility of the designer.
- The fast delay (D_f) equation coefficients are generated assuming the fastest operating conditions for a typical process with the fastest excitation pattern. Again, the determination of the fastest excitation pattern is the responsibility of the designer.
- The basic pulse width is defined with a sensitivity to the load capacitance, but independent of the input pulse width. Then the minimum and maximum pulse widths (PW_{min} and PW_{max}) values at the receiving end are computed based on leading edge transition times and RC delays. The output pulse width is independent of the input pulse width. The macro design is constrained to guarantee that the sensitivity of the output pulse width to input pulse width can be expressed with this model.

The timing models use equations of the form

$$\begin{aligned} delay = D + A \times (TR - TR_d) + \\ B \times (CL - CL_d) + X \times (CL - CL_d)^2 + \\ Y \times (TR - TR_d)^2 \end{aligned}$$

where D is a constant delay, TR is the transition time of the rising edge, TR_d is the design value of the leading edge transition time, CL is the load capacitance and CL_d is the design value of the load capacitance. Note that this is just the well known k-factor [6] approach. However, we have centered the equation on the design values — in addition to being more intuitively obvious to the designer, this forces some planning in the design for expected values.

In addition to the model described above, two new delay modeling concepts are used. These are the notions of *delta delay* and *tracking*.

- **Delta Delay (DD):** In a typical SRCMOS macro, groups of output pins are often related. Also, groups of input pins usually use the same signal to derive the reset, and hence have related constraints. So, all the delay equations are typically specified for an entire group of signals. When a group of signals is modeled together, there can be some variation in the delay between the different signals; for example, if a bundle of signals is modeled with a single delay equation, there will be some spread between the strands of the bundle due to geometric considerations. The worst case spread that can occur between the signals is modeled by the parameter DD called delta-delay.
- **Tracking:** As mentioned earlier, across-chip variations can create significant changes in the delay behavior of a given macro. When we analyze the pulses at the inputs of a macro, we may not know the operating conditions at the points where those signals are created. This uncertainly implies that we have to be conservative in performing various checks. Hence, we have to assume that the maximum possible spread will occur between these signals. In certain cases, we can infer that the signals are all derived from sources operating under similar conditions. For example, they may all be coming from the same macro. In this case, in addition to having similar operating conditions, they may also have the same pattern dependence (*i.e.*, we can assume that those signals are in the same mode, early or late). Such a group of signals is said to *track*.

The significance of the above two concepts will become clearer when we discuss modes of analysis in Section 3.3 and timing specifications in Section 4.

3.3 Early and Late Mode Analysis

As mentioned earlier, timing analysis is performed in two modes, early and late. Early mode analysis is used to detect problems with fast paths in the design. Fast path conditions could lead to insufficient hold times and to race condition failures. In this analysis, we use the fast delay equations, D_f to model the different components.

Late mode analysis is used to detect problems with slow paths, *i.e.*, slow paths typically lead to setup violations at latch points. In this case, we use the slow delay equations, D_s to model the components.

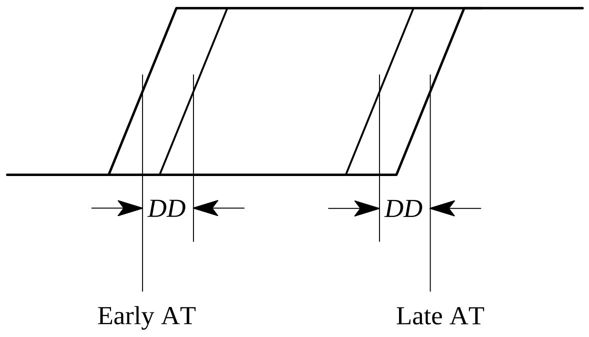


Figure 5: Effect of DD

The *delta-delay* specification, as defined in the previous section, changes the fast and slow delay values. We define the delta delay in such a way that it adds to the fast delay and subtracts from the slow delay values. The delay ranges are defined by the following inequality:

$$D_f \leq (D_f + DD) \leq (D_s - DD) \leq D_s$$

This is pictorially depicted in Figure 5.

Minimum Pulse Width	WA_{min}
Maximum Pulse Width	WA_{max}
Minimum Hold Time	DH_{min}
Maximum Hold Time	DH_{max}
Minimum Quiet Time	DQ
Minimum Setup Time	DS
Maximum Target Time	T

Table 1: Available timing tests

4 TIMING TESTS FOR SRCMOS

In this section, we describe the various tests that we use to verify the timing constraints for SRCMOS. An attempt is made throughout to use conventional static timing analysis terminology.

As described earlier, our timing model uses extensive grouping to minimize the amount of data that must be entered by the designer. The two most commonly used types of groups are *setup* groups and *pulse zone* groups.

A setup group may have one or more *reference* signals and one or more *data* signals. Basically, the reference signals are used for timing information, and the timing of the data signals must be checked with respect to the reference signal. In this case, the reference signals are analogous to clock signals.

A pulse zone group consists of only data signals. Here, the timing information is derived completely from the data signals. All signals in the group must obey timing tests with respect to the latest arriving signal in the group.

The timing tests that can be specified are shown in Table 1. However, not all of these tests apply to both types of groups described above. These tests have been described only for the cases where pulse signals interact with each other. In practice, several tests between pulse and static signals are also required, since every design uses a mixture of static and dynamic circuits.

Notation: The notation used in the rest of this section is summarized in Table 2.

Pulse Width Computation

Our timing models specify the pulse width at the output of the driver as a function of the load capacitance; to obtain the pulse width at the receiver, we need to adjust this based on RC effects. Since we do not propagate the trailing edge of the pulse through the RC network, we approximate the degradation of the trailing edge based on the RC delay the leading edge (RC_{LT}). In general, this approach tends to err on the conservative side. The computation is described by the following equations:

$$\begin{aligned} PW_{min}(rcv) &= PW_{min}(drv) + \\ &\quad 0.5 \times (TT_{drv} + RC_{LT}) \\ PW_{max}(rcv) &= PW_{max}(drv) + \\ &\quad 0.5 \times (TT_{drv} + RC_{LT}) \end{aligned}$$

Description	*Notation*
Set of pulse zone group signals	PZ
Set of data signals in setup group	$DATA$
Set of reference signals in setup group	REF
Early Mode Arrival Time	AT_{min}
Late Mode Arrival Time	AT_{max}
Leading Edge Transition Time	LT
Trailing Edge Transition Time	TT
RC Delay	RC
Minimum Pulse Width	PW_{min}
Maximum Pulse Width	PW_{max}
Minimum Hold Time	DH_{min}
Maximum Hold Time	DH_{max}
Minimum Quiet Time	DQ
Minimum Setup Time	DS
Maximum Target Time	T
Receiver of net	rcv
Driver of a net	drv

Table 2: Notation

4.1 Minimum and Maximum Pulse Width

The minimum and maximum pulse width tests are defined only for pulse zone groups. Typically, minimum pulse widths are defined for macros which use pulse stretchers (transparent latches) at their inputs. In these cases, the input pulse must be a) long enough for the pulse stretcher to fire, and b) short enough that it does not interfere with the resetting of the pulse stretcher. However, there are no tests between different signals in the group. The pulse width tests are checked against the actual pulse widths at all the macro inputs in a pulse zone group. The tests can be described by the following equations:

$$\forall p \in PZ, \quad PW_{min}(p) \geq WA_{min}$$

$$\forall p \in PZ, \quad PW_{max}(p) \leq WA_{max}$$

4.2 Minimum and Maximum Hold Time

The hold time tests are applied to data signals in setup groups and to all signals in pulse zone groups. The signals are checked relative to the latest arriving signal in the group. The objective of the minimum hold time test is to ensure that pulse intersection constraints are satisfied for the circuit under test.

In the default case, we have to make the worst case assumption that the reference signal might arrive in late mode, while the data signals might arrive in early mode. For a setup group, the minimum hold time test can be described by the following inequality:

$$\begin{aligned} &\min_{d \in DATA} \{AT_{early}(d) + PW_{min}(d)\} \\ &\quad - \max_{r \in REF} \{AT_{late}(r)\} \geq DH_{min} \end{aligned}$$

If the reference and data signals belong to the same tracking group, then, the early mode and late mode values are checked separately using the following formulae:

$$\min_{d \in DATA}\{AT_{early}(d) + PW_{min}(d)\} - \max_{r \in REF}\{AT_{early}(r) + DD\} \geq DH_{min}$$

$$\min_{d \in DATA}\{AT_{late}(d) - DD + PW_{min}(d)\} - \max_{r \in REF}\{AT_{late}(r)\} \geq DH_{min}$$

Note that DD affects the reference signals in early mode and the data signals in late mode.

The minimum hold time test is analogously applied to the pulse zone groups, with the main difference that it is applied between the latest arriving signal and all other signals (*i.e.*, there is no distinction between data reference signals). The general test for pulse zone groups is expressed by equation 1 while the test for the case where all the signals track is expressed by equation 2 and equation 3.

$$\min_{p \in PZ}\{AT_{early}(p) + PW_{min}(p)\} - \max_{p \in PZ}\{AT_{late}(p)\} \geq DH_{min} \tag{1}$$

$$\min_{p \in PZ}\{AT_{early}(p) + PW_{min}(p)\} - \max_{p \in PZ}\{AT_{early}(p) + DD\} \geq DH_{min} \tag{2}$$

$$\min_{p \in PZ}\{AT_{late}(p) + PW_{min}(p) - DD\} - \max_{p \in PZ}\{AT_{late}(p)\} \geq DH_{min} \tag{3}$$

The maximum hold time test arises from the requirement to have inputs inactive before the reset process starts. The expressions for checking maximum hold time can be constructed in a similar manner to those for minimum hold time; they are ommitted here for brevity.

4.3 Minimum Setup Time

Minimum setup time test is used for setup groups. The objective of the setup time test is to ensure that pulse intersection requirements are met; it works jointly with the minimum hold time test to ensure this.

In the general case, the setup time test is checked between the arrival times of the reference signals in early mode and the arrival time in late mode of the data signals. This is expressed in equation 4.

$$\min_{r \in REF}\{AT_{early}(r)\} - \max_{d \in DATA}\{AT_{late}(d)\} \geq DS \tag{4}$$

For the cases where the reference and data signals track, the tests can be done using the DD value. This is shown in equation 5 and equation 6.

$$\min_{r \in REF}\{AT_{early}(r)\} - \max_{d \in DATA}\{AT_{early}(d) + DD\} \geq DS \tag{5}$$

$$\min_{r \in REF}\{AT_{late}(r) - DD\} - \max_{d \in DATA}\{AT_{late}(d)\} \geq DS \tag{6}$$

4.4 Minimum Target Time

The minimum target time test is used only for pulse zone groups. The minimum target time test specifies that all signals within a pulse zone group must arrive within a certain interval of each other. This test works with the minimum hold time test to ensure that sufficient pulse intersection is achieved. The general case test is expressed by equation 7. The case where the signals track is expressed by equation 8 and equation 9.

$$\max_{p \in PZ}\{AT_{late}(p)\} - \min_{p \in PZ}\{AT_{early}(p)\} < T \tag{7}$$

$$\max_{p \in PZ}\{AT_{late}(p)\} - \min_{p \in PZ}\{AT_{late}(p) - DD\} < T \tag{8}$$

$$\max_{p \in PZ}\{AT_{early}(p) + DD\} - \min_{p \in PZ}\{AT_{early}(p)\} < T \tag{9}$$

4.5 Minimum Quiet Time

The quiet time test is used to ensure that pulse collision is avoided. The quiet time test is specified against the leading edge of the appropriate signal. This test specifies that a *new* pulse cannot appear on the given signal for DQ time units from the given reference point. Since this test applies across cycle time boundaries, tracking has no effect on this test.

This test can be specified for both pulse zone groups and setup groups as described by equation 10 and equation 11. In these equations, CT refers to the target cycle time used for the static timing analysis[1].

$$\min_{p \in PZ}\{AT_{early}(p)\} + CT - \max_{p \in PZ}\{AT_{late}(p)\} \geq DQ \tag{10}$$

$$\min_{d \in DATA}\{AT_{early}(d)\} + CT - \max_{r \in REF}\{AT_{late}(r)\} \geq DQ \tag{11}$$

5 IMPLEMENTATION

The timing analysis system for SRCMOS has been implemented using an existing static timing analysis tool *STEP* [11] as the basic engine. The small-gate delay model was adapted to handle large macros. In our methodology, the analysis of each macro was done

[1] This is typically different from the actual target cycle time, since adjustments have to be made for clock skew and other effects.

by the designer, using pattern-dependant simulation. The various coefficients for the delay and pulse width equations are extracted from the simulation results. The values for the timing tests are specified by the designer. A *timing rule* is then created for each macro, combining the delay equation coefficients and the timing tests. The timing analysis program reads this rule.

The leading edge of the pulse is propagated using the existing timing analyzer. The timing analyzer creates arrival times, required arrival times and slacks for at all nodes for the leading edge of the signal[1]. After this operation is complete, the SRCMOS timing analyzer is dynamically loaded and executed.

The overall flow of the SRCMOS timing analyzer is as follows:

```
Create logic network data model
Read special SRCMOS timing tests
Read arrival time values from prior
analysis
Run SRCMOS Timing Analyzer:
    Compute pulse widths at outputs of macros
    Compute pulse widths at inputs based on
    driving pulse width and RC
    Identify which tests are to be performed
    under tracking conditions
    Perform timing tests defined for input groups
    and pulse zone groups
Create error messages for failing tests
```

6 CONCLUSIONS

A method to extend static timing analysis to SRCMOS circuits has been described. Such techniques will become critical as dynamic circuit use becomes more widespread. Even though many of the constraints that arise out of such circuit techniques may be new, it is very important to base new tools on existing tools and methodologies.

A program based on the described approach has been implemented, and exercised on unit-level models containing moderate number of macros. A full chip exercise is still pending.

7 Acknowledgements

Many people at IBM Yorktown Heights and IBM Austin contributed to this work in different ways. Scott Neely participated extensively in the implementation of the program. Ruud Haring, Terry Chappell, Joel Silberman, Ed Seewann, Marty Schmookler and Don Mikan contributed to the development of the timing constraints and tests. Gary Nusbaum, Clint Miller and Bob Swanson helped in integrating the timing analysis tool into the design methodology.

References

[1] R.B. Hitchcock, G. L. Smith, and D. D. Cheng, "Timing analysis of computer hardware," *IBM J. of Research and Development. 26*, pp. 100–105, January 1982.

[2] K.A. Sakallah,T.N. Mudge, and O.A. Olukotun, "checkTc and minTc: Timing Verification and Optimal Clocking of Synchronous Digital Circuits", *ICCAD 1990*, pp. 552–555, November 1990.

[3] T. I. Chappell, B. A. Chappell, S. E. Schuster, J. W. Allen, S. P. Klepner, R. V. Joshi, R. L. Franch, "A 2-ns cycle, 3.8ns access 512-kb CMOS ECL SRAM with a fully pipelined architecture," *IEEE JSSC*, vol. 26, no. 11, pp. 1577–1585, Nov 1991.

[4] R.A. Haring, M.S. Milshtein, T.I. Chappell, S.H. Dhong, B.A. Chappell, "SelfResetting Logic Register and Incrementer," *To appear - Proceedings of the 1996 Symposium on VLSI Circuits*, June 1996.

[5] T.I. Chappell, R.A. Haring, T.K. Jaber, E. Seewann, M.P. Beakes, B.A. Chappell, B. M. Fleischer, "High Performance Self Resetting Circuits with Enhanced Testability," *IBM Research Report RC20321*, January 1996.

[6] Weste and Eshraghian, "Principle of CMOS VLSI Design, 2nd Edition", pp. 220-225, Addison-Wesley, 1992.

[7] Weste and Eshraghian, "Principle of CMOS VLSI Design, 2nd Edition", pp. 308-310, Addison-Wesley, 1992.

[8] L. A. Lev *et al*, "A 64-b microprocessor with multimedia support," *IEEE J. Solid-State Circuits*, vol. 30, no. 11, pp. 1227–1238, Nov 1995.

[9] D. Wendell, "Reset logic circuit and method," U.S. Patent 5,438,283.

[10] T. Williams, "Performance of iterative computation in self-timed rings," *J. VLSI Signal Processing*, Kulwer Publishers, vol. 7, pp. 17–31, Feb 1994.

[11] R.E. Mains, T.A. Mosher, L.P.P.P. van Ginneken, R.F. Damiano, "Timing Verification and Optimization for the PowerPC Processor Family," *IEEE Internation Conference on Computer Design*, pp. 390–393, 1994.