Modeling Domino Logic for Static Timing Analysis

by

David Van Campenhout, Trevor Mudge and Karem Sakallah

CSE-TR-295-96

THE UNIVERSITY OF MICHIGAN

Computer Science and Engineering Division
Department of Electrical Engineering and Computer Science
Ann Arbor, Michigan 48109-2122
USA

Modeling Domino Logic for Static Timing Analysis

David Van Campenhout, Trevor Mudge and Karem Sakallah

Advanced Computer Architecture Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, Michigan 48109-2122

January 1, 1996.

Abstract

High performance microprocessors employ various advanced circuit techniques to achieve high clock frequencies. Critical sections of the design are often implemented in domino logic, a popular style of dynamic logic. This paper describes a new model for analyzing the temporal behavior of sequential circuits containing both static logic and domino logic. Our model integrates naturally with the SMO model for static timing analysis of sequential circuits.

1 Domino Logic

Domino logic is a popular dynamic logic family. A generic Domino gate is depicted in Figure 1. The operation of the circuit is as follows. When the clock ϕ is low, the internal node z is precharged, and the output node y is set to 0. The period in which ϕ is low is called the *precharge phase*. A rising transition on the clock permits to conditionally discharge the internal node y through the pulldown network. The values of the inputs determine whether the discharge actually takes place. This phase is called the *evaluate phase*. Once z is discharged, it will stay low for the rest of the evaluate phase no matter what values the inputs assume. Therefore, either the inputs have to settle to their stable value before the start of the evaluate phase, or they can settle to their stable value (a high value) by making a single rising transition during the evaluate phase. This condition, of inputs making at most one rising transition, during the evaluate phase is actually too strict, as will be illustrated later in Figure 5. The condition can be relaxed as follows. Inputs are allowed to make a single rising transition in the beginning of the evaluate phase. After this rising transition, they have to remain stable, for at least long enough for the dynamic gate to propagate the transition to its output. This minimum stable time dictates a hold constraints on inputs of the dynamic gate. After the transition has been propagated, the input signals are allowed to change. This effectively reduces the period in which the correct result is available at the output of the domino gate: the output might get set to 0 before the precharge phase starts.

The inverter at the output of the gate is included for several reasons. First, it is required for proper operation of a chain of domino gates. Second, the internal node z is a weak node. When the clock is low, the high value on that node is not driven. The inverting buffer separates that dynamic node from the rest of the circuit, thus alleviating charge-sharing problems and minimizing capacitive coupling. A consequence of the inverting buffer is that a domino gate can only implement a non-inverting function of its inputs. The dual circuit of that shown in Figure 1 is also a valid domino gate. However, in many technologies, and especially in CGaAs, the performance of such a dual gate is far inferior to that of the original gate due to the poor characteristics of the pFET. Among the reasons why zipper logic, a variant of domino logic is not very popular are that it uses these dual gates and that it does not use the buffering inverter. The following discussion will only be concerned with the type of gate shown in Figure 1.

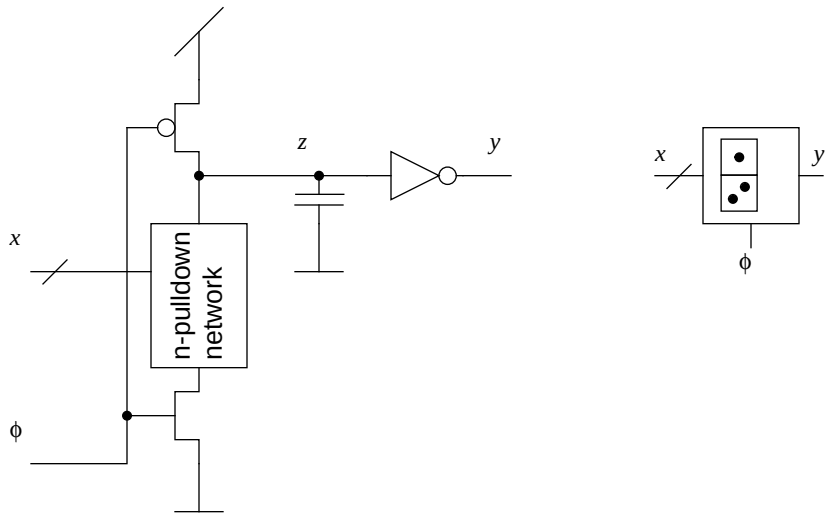

Figure 1: Generic domino gate: circuit (left), symbol (right)

2 Modeling the Waveform of a Signal Produced by a Domino Gate

Accurate verification of domino circuits requires a more sophisticated waveform model than the classical two-event model used in [1]. The output signal y, of a dynamic gate has characteristic properties. A waveform model can be derived by enumerating all different waveforms the y can exhibit. First consider the domino gate under the classical rules for proper operation, i.e. inputs have to remain stable during evaluate, or make at most a single rising transition during evaluate. Four cases can be distinguished on the basis of what value the gate evaluated to in the previous cycle, and what value the gate evaluates to in this cycle. These are the top four waveforms for y in Figure 2.

Now consider the implications of allowing signals corresponding to the next cycle to arrive before the end of the evaluate phase. Such an early arrival can only affect y if y is supposed to evaluate to 0 during this cycle. An early arrival of a rising transition corresponding to the next cycle, on an input signal, can then overwrite y with a 1. Shortly after this transition, the falling transition on the clock will force y to 0. The exact waveform depends on the exact arrival time of that early transition on the input, the delay from that input to y, and the delay from the clock to y. For our treatment it suffices to note that there is going to be an interval starting near the end of the evaluate phase of this cycle, and continuing until somewhere in the beginning of the precharge phase of the next cycle, in which y exhibits a rising transition followed by a falling transition. Furthermore, due to proximity effects the transitions may blend and the high value may never be reached at all. The effect described extends over two consecutive clock cycles. Hence, in a clock cycle the effect can occur both at the beginning of the precharge phase, and at the end of the evaluate phase. All cases that occur are enumerated in Figure 2.

All of the waveforms enumerated above can be summarized in a single waveform shown at the bottom of Figure 2. This waveform constitutes our waveform model. It is characterized by four events, instead of the usual two-event model used in [1]. The region marked *changing* could be denoted with a more specific attribute (at most 2 transitions), but this information is not used in the rest of the analysis. Essential of that region is that it separates the region in which the gate evaluates to its intended value from the region in which the gate is set to the precharge value.

Figure 3 shows the I/O waveforms of a dynamic gate using our model. We distinguish between inputs driven by static logic and input driven by dynamic logic. Our notation to denote the events defining the waveform is an extension of that used in [1]. Essentially we have refined the interval $[a,A]$ in which the signal is changing, to three intervals: during $[\lfloor a \rfloor,\lceil a \rceil]$ the waveform is changing; during $[\lceil a \rceil,\lfloor A \rfloor]$ the waveform is 0; during $[\lfloor A \rfloor,\lceil A \rceil]$ the waveform exhibits a single rising transition or no transition at all.

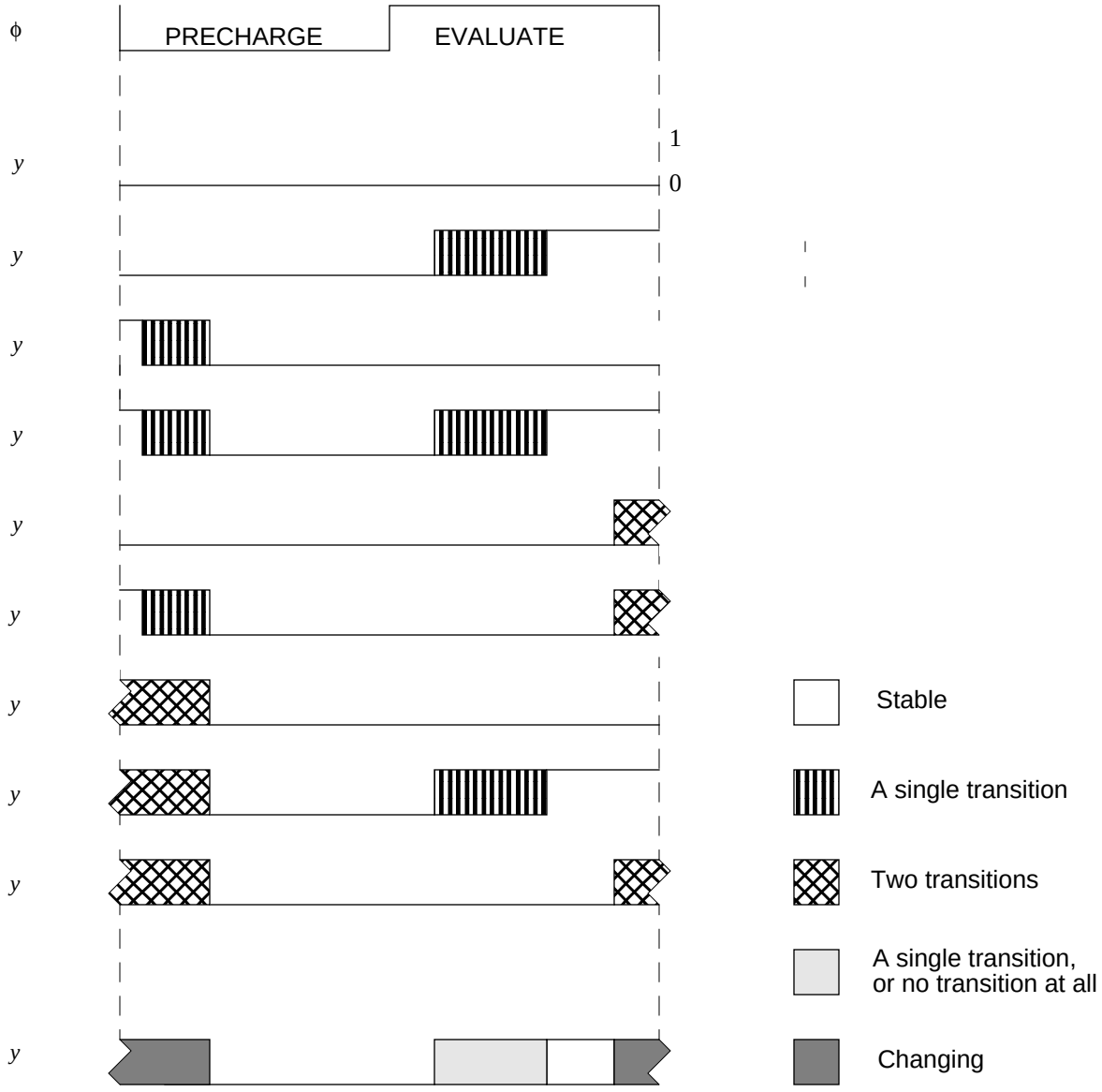

Figure 2: Waveforms of the output of a domino gate

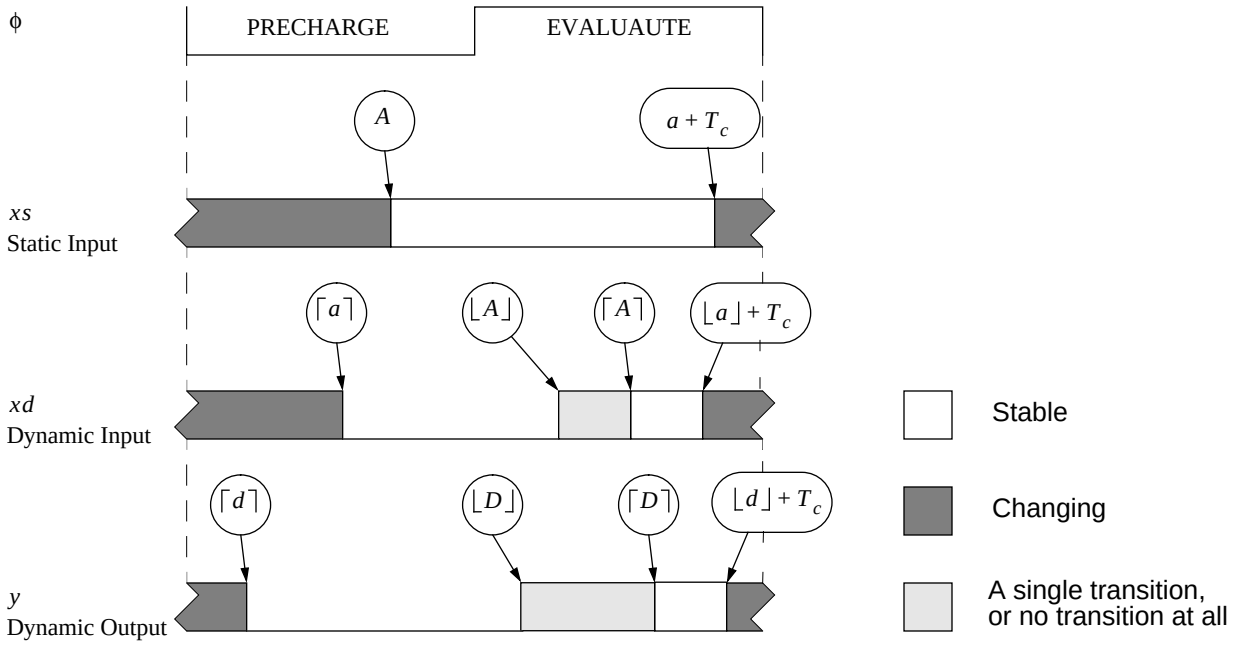

Figure 3: I/O Waveforms of a Domino gate

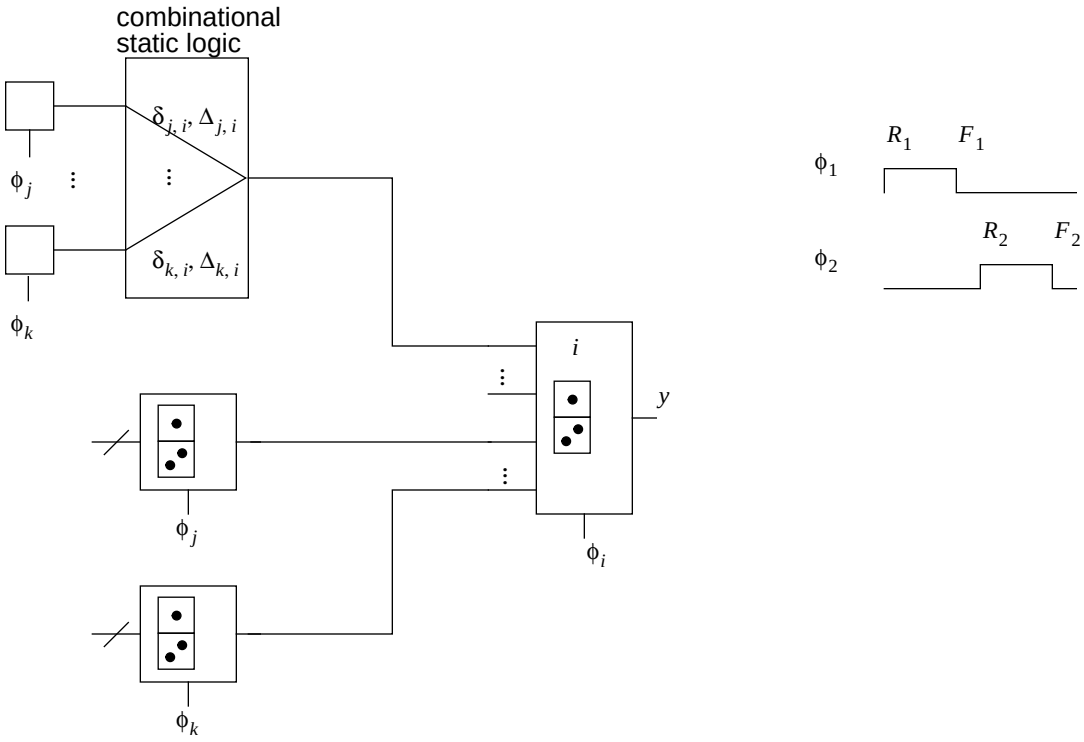

Figure 4: Timing checks for dynamic gates: circuit fragment

3 Static Timing Verification of Circuits Containing Domino Gates

In this section we extend the SMO static timing analysis model ([1] and [2]), dynamic logic. The discussion in the previous section provides enough material to formulate the timing constraints associated with dynamic gates. The configuration under investigation is shown in Figure 4. A multiphase clock generator produces clock phases with the same period T_c. Each phase p is characterized by its rising and falling transition R_p and F_p. Each latch i has an enabling and latching event time, E_i and L_i. These map onto R_p and F_p depending on whether the latch is sensitive to the positive or the negative level of the clock. Event times relative to a local time zone are denoted E'. The actual signal clocking a latch has a minimum and maximum skew q_i and Q_i with respect to the base phases. Combinational logic is characterized by its minimum and maximum delay $\delta_{j,i}$ and $\Delta_{j,i}$. Earliest and latest arrival times of signals at the input of synchronizers or dynamic gates are denoted by a and A respectively. Arrival times are measured relative to the time zone of the primary clock of the synchronizer or dynamic gate. Similarly, d and D denote the earliest and the latest departure time respectively. A more elaborate discussion of the base model can be found in [1] and [2].

We will now augment the SMO model to handle domino logic. The configuration not covered by the original SMO model is shown in Figure 4. A dynamic gate i clocked by ϕ_i. Some of its inputs are driven by static combinational logic. The other inputs are driven by other dynamic gates.

Dynamic gate macromodel

- Static inputs:

$$A_i \le E'_i - S_i \tag{1}$$

$$a_i + T_c \ge A_i + H_i \tag{2}$$

- Dynamic inputs:

$$\lceil A_i \rceil \le T_c - S_i \tag{3}$$

$$\lceil a_i \rceil \le E'_i - S_i \tag{4}$$

$$\lfloor a_i \rfloor + T_c \ge \lceil A_i \rceil + H_i \tag{5}$$

- For fanout to dynamic logic:

$$\lceil D_i \rceil = \max\{\lceil A_i \rceil + \Delta_{x \to y}, E'_i + \Delta_{\phi \to y} + Q_i\} \tag{6}$$

$$\lfloor D_i \rfloor = E'_i + \delta_{\phi \to y} + q_i \tag{7}$$

$$\lceil d_i \rceil = \Delta_{\phi \to y} + Q_i \tag{8}$$

$$\lfloor d_i \rfloor + T_c = \min\{a_i + T_c + \delta_{x \to y}, \lfloor a_i \rfloor + T_c + \delta_{x \to y}, E_i' + \delta_{\phi \to y} + q_i\} \tag{9}$$

- For fanout to static logic:

$$d_i = \lfloor d_i \rfloor \tag{10}$$

$$D_i = \lceil D_i \rceil \tag{11}$$

Combinational Propagation Constraints

- Static inputs:

$$A_i = \max\{D_j + \Delta_{ji} - \varnothing_{ji}\} \tag{12}$$

$$a_i = \min\{d_j + \delta_{ji} - \varnothing_{ji}\} \tag{13}$$

- Dynamic inputs:

$$\lceil A_i \rceil = \max\{\lceil D_j \rceil - \hat{\varnothing}_{ji}\} \tag{14}$$

$$\lfloor A_i \rfloor = \min\{\lfloor D_j \rfloor - \hat{\varnothing}_{ji}\} \tag{15}$$

$$\lceil a_i \rceil = \max\{\lceil d_j \rceil - \hat{\varnothing}_{ji}\} \tag{16}$$

$$\lfloor a_i \rfloor = \min\{\lfloor d_j \rfloor - \hat{\varnothing}_{ji}\} \tag{17}$$

Phase Shift Operators:

$$\varnothing_{ji} = T_c - (L_j - L_i) \bmod T_c \tag{18}$$

$$\hat{\varnothing}_{ji} = (T_c - (L_j - L_i)) \bmod T_c + c_{ji} T_c \tag{19}$$

The parameter c_{ji} in the expression for the phase shift involving dynamic gates has to be set according to what the design intent is.

3.1 Timing Verification

For timing verification, the constraints formulated above need to be checked. This requires the computation of the arrival times of the inputs to the domino gates. If all synchronizers in the circuit are edge triggered, all the arrival times can be computed by traversing the circuit in level-order, starting from the synchronizers.

If level-sensitive synchronizers are present, several iterations might be required to obtain the actual arrival times. A relaxation algorithm for solving these equations is described in.

4 Example

An example circuit is shown in Figure 5. The waveforms exhibited on each circuit node are shown in Figure 5. Causality is indicated by the arrows. The registers at the boundaries of the circuit are negative-edge triggered, and are clocked by the same phase. Thus one clock cycle is available for the signals to propagate between the registers. The logic in between the registers consists of a mixture of static and dynamic logic. The dynamic exor gate is of special interest. Both the true and the complement of the input signals are required. Input signals are allowed to make a rising transition during the evaluate phase. But, a rising transition on the uncomplemented input implies a falling transition on

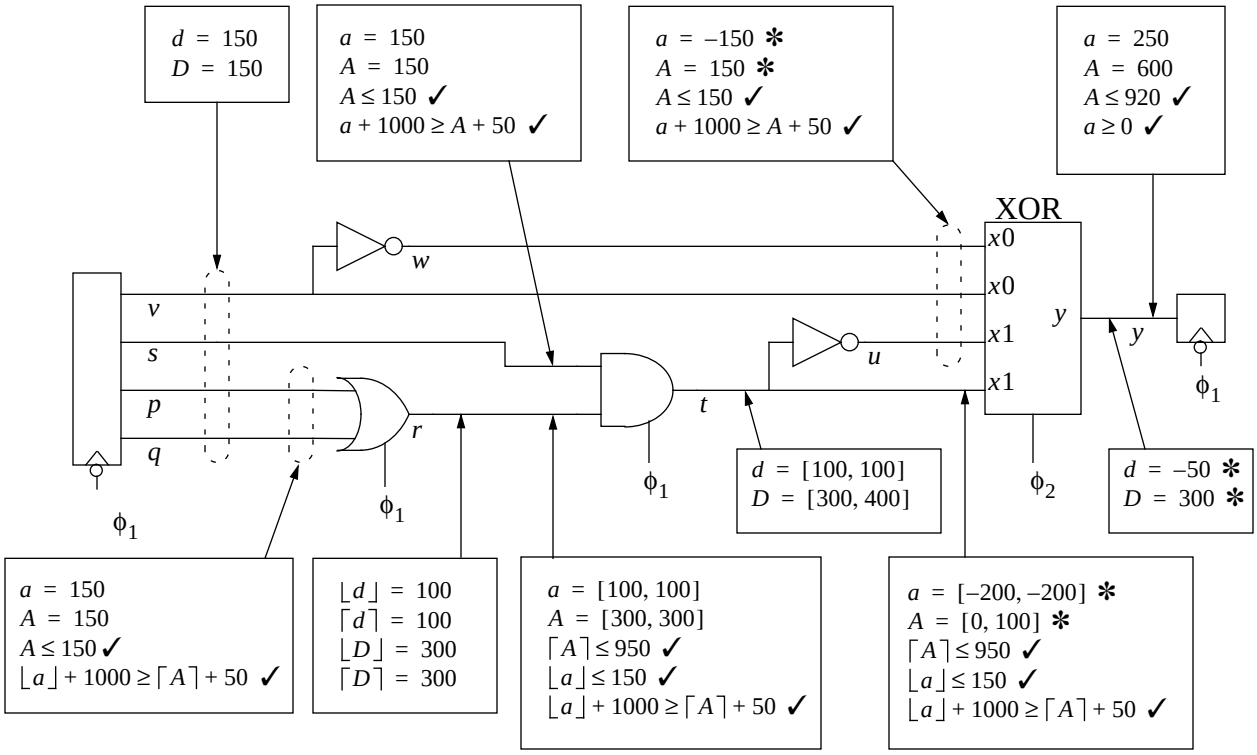

Table 1: Delays [ps]

	Dynamic (AND,OR, XOR)	static	
		INV	DFF
$\phi \rightarrow y$ (prech.)	100	N/A	N/A
$\phi \rightarrow y$	100	N/A	150
$x \rightarrow y$	100	50	N/A
Setup	50	N/A	80
Hold	50	50	0

N/A = not applicable

✻ These event times are relative to ϕ_2 .

Clock Schedule

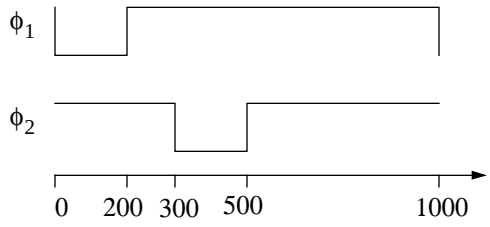

Figure 5: A multiphase example

the complemented signal. Thus if a dynamic exor-gate is appended to a chain of domino gates, and they all have the same clock, incorrect operation will result. There are two solutions to this problem. The first one is to produce the true and the complemented signals directly from dynamic logic. This might involve a significant area overhead. The second solution is to generate to complemented signal with a static inverter from the dynamic signal, and to ensure that the latest arrival of both true and complemented signals occurs during the precharge phase of the exor. This is achieved by using a separate clock phase to clock the exor-gate. This second solution is adopted in the example. The example also

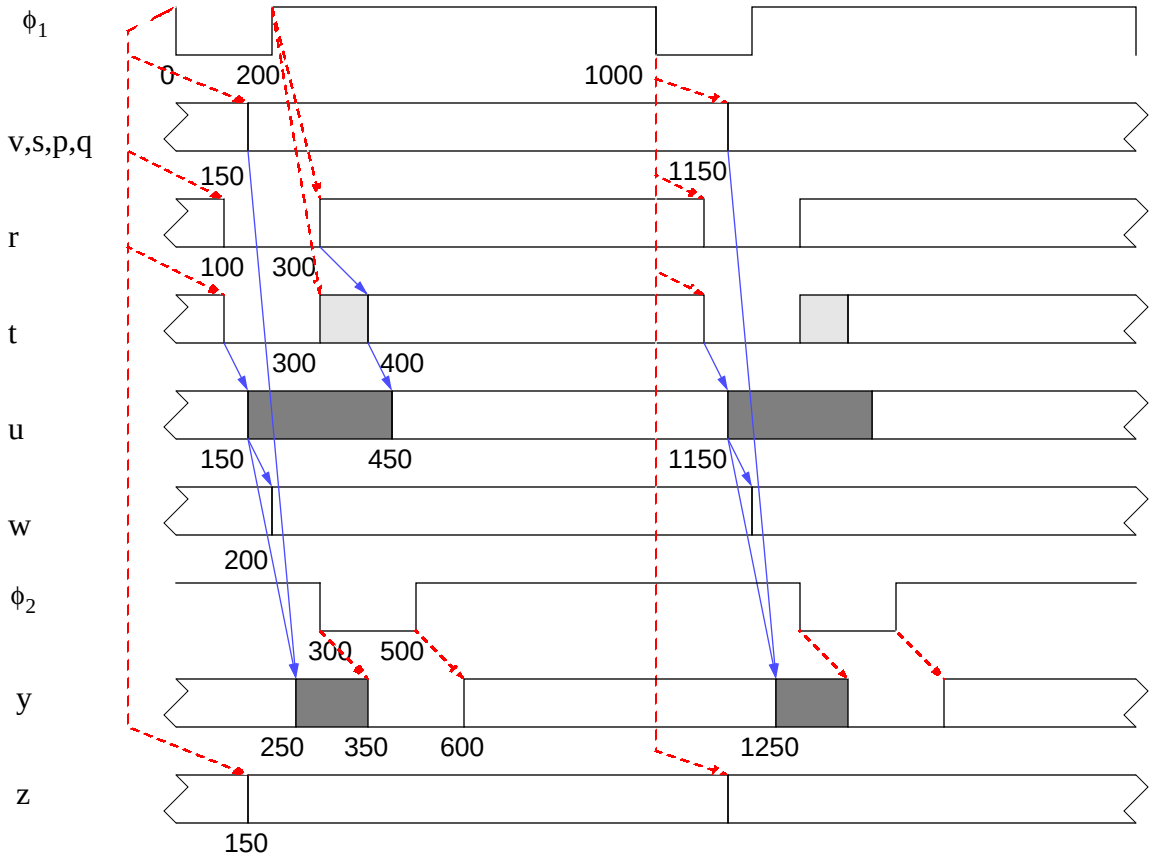

Figure 5: Waveforms for the example in Figure 5.

illustrates that early signal changes pertaining the next clock cycle, might arrive during the evaluate phase. The classical domino rules would not allow this behavior. The input signal v assumes its new value at $t = 1150$. At that time the clock steering the xor, ϕ_2, is still high. Hence, if the exor gate evaluated to a 0, the early arrival of v might cause the zero to be overwritten by a 1. The delay of the exor causes the output to be affected 100ps later. This is no problem provided that the intended value the exor evaluates to, propagates to the synchronizers. At $t = 1000$, the correct value of y, is captured by the output register. This is well before the early arrival of v corrupts y. According to the classical domino rules we would be forced to make the falling edge of ϕ_2 occur earlier so that the early arrival of v occurs during the precharge phase. This requires more complex circuitry. In our case ϕ_2 can simply be generated by delaying ϕ_1.

5 Conclusion

The timing of domino logic was analyzed. A waveform model for signals produces by domino gates was proposed. The waveform has four parameters instead of the usual two for waveforms produced by static logic. The new waveform model conveys enough information to allow for an accurate analysis of the subtle issues in the timing of domino gates. Our model integrates naturally with the SMO model for static timing analysis.

References

[1] K. Sakallah, T. Mudge, O. Olukotun, “checkTc and minTc: Timing Verification and Optimal Clocking of Synchronous Digital Circuits,” in *ICCAD-90 Digest of Technical Papers*, pp. 552-555, November 1990.

[2] T. Burks, K. Sakallah and T. Mudge, “Critical Paths in Circuits with Level-Sensitive Latches,” *IEEE Trans. VLSI Systems*, Vol. 3, No. 2, pp. 273-291, June 1995.