

Modeling for Logic Level Minmax Simulation

William Paulsen
Viewlogic Systems, Inc.
293 Boston Post Road West
Malboro, Massachusetts 01752
bpaulsen@viewlogic.com

Zainalabedin Navabi
Electrical and Computer Engineering
Northeastern University
409 Dana Research Building
Boston, Massachusetts 02115
navabi@nuvlsi.coe.neu.edu

ABSTRACT

This paper discusses minmax simulation for logical circuits and suggests a VHDL implementation methodology for this simulation. The document starts with definition of minmax and how it relates to the timing analysis of digital circuits. This is followed by presentation of an approach for minmax simulation. We will then show the VHDL implementation of this approach. We will also suggest ways in which VHDL can be improved to help the implementation of the minmax models more efficient to simulate.

1. Introduction

Physical properties of the logic gates in a circuit cause each of these elements to have their unique timing values even if they are on the same chip. Temperature and load values also contribute to different gate delay values. Simulation with worst case, typical delay, or minimum delay values, while efficient, do not present a precise timing behavior of the circuit. Alternatively, a simulation that results in signal values that indicate a gray area during which the signal may change, presents a more realistic picture of the actual behavior of a circuit. We will refer to this method of simulation as *minmax* simulation. We will use other terminologies for distinguishing between various timing schemes used in simulation. A simulation that uses either minimum, maximum, or typical delay values will be referred to as *minimum*, *maximum*, or *typical* delay simulation respectively. A simulation that uses either of these three values will be referred to as *static* delay simulation. The *static* delay simulation

is in contrast with the *minmax* or *ambiguous* delay simulation.

In this paper we will first describe timing in logic simulation and define minmax simulation. Section 3 discusses how minmax timing can help timing checks. In Section 4 signal characteristics for minmax timing analysis will be discussed. Section 5 focuses on the application of minmax signals in timing analysis. In section 6 we will begin the presentation of the VHDL implementation of models for minmax simulation. Section 7 discusses alternative implementations and VHDL shortcomings in this implementation.

2. Logic Simulation Timing

Physical characteristic of logic gates cause functionally equivalent gates used in different parts of a circuit to have different timing properties. This is usually caused by inconsistent manufacturing processes that even vary between gates on the same waver or even the same chip. In addition to manufacturing issues, temperature and load values of the individual gates can significantly change their timing behavior. A method of overcoming these timing differences is to assume a worst case scenario and simulate an entire circuit based on the worst case gate delays. This method causes several functional problems such as those with detection of hazards. Other problems this may cause is the overly pessimistic view of the timing of the circuit that will be presented, which makes full speed capabilities of a circuit not to be utilized. Typical delay or minimum delay simulation also have similar problems. For example minimum delay can also report false hazard detection or let hazards go undetected.

In minmax simulation the minimum and the maximum gate delays are considered. Each signal has a logical output that includes a gray area with a width that is equal to the difference of the maximum and the minimum delay values. The gray areas will be part of the logic system used for the simulation and propagate and combine according to the gate functions. These gray areas will be used for timing checks such as pulse width, setup time and hold time. We will present several examples for illustrating discrepancies in simulation that these gray areas may cause. These examples will be used for justification of the analysis methods presented in the following sections. In these examples we label the gates as $g0, g1, \dots$, and use $d0, d1, \dots$ for the difference of max and min propagation delay values of gates labeled as such.

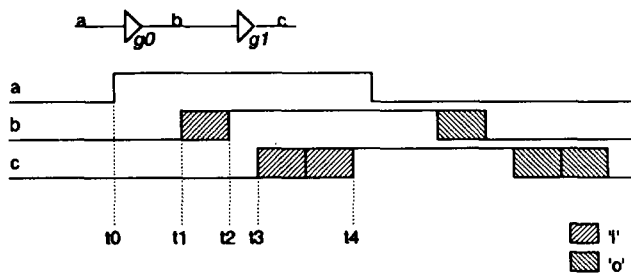


Figure 1. Edge minmax delay

Figure 1 shows a signal that passes through two buffers before it gets to the output node c . In static delay simulation, the output will change from 0 to 1 after a delay contributed by $g0$ and $g1$ gates. In minmax simulation, the output at node b accumulates a gray area contributed by $g0$ and the output at node c accumulates a shaded area contributed by both gates $g0$ and $g1$. The waveform shown at point c indicates that the output may change from 0 to 1 any time between $t3$ and $t4$. Note that the duration of the gray area on this signal is $d0+d1$.

Figure 2 shows a signal that fans out and re-converges back to an AND gate through two paths with different delay values. Because of the delays at nodes b and c , a hazard appears on the output of the AND gate at node d . Figure 2 shows that the duration of this hazard may have a maximum value of $t7-t2$ which is a total of high and low delay transitions of all gates in the input to output path. However, considering that gate $g1$ contributes to this delay from two separate paths, its contribution should only be taken into account once. Therefore, the duration of the hazard will only be $t7-t2-d1$ in a time window equal to $t7-t2$. Although the gray area on the d signal is still $t7-t2$, we

should have the mechanism to detect the maximum pulse width on this signal.

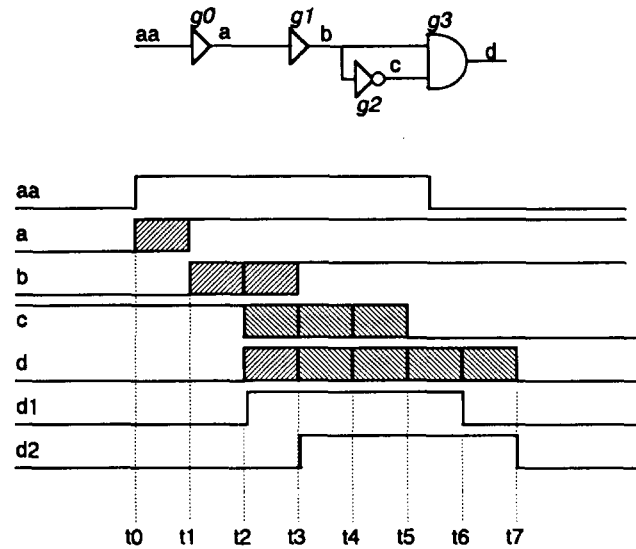


Figure 2. Delay in a re-convergent fanout path

The timing diagram shown in Figure 4 corresponds to the circuit of Figure 3. In this figure, if all worst cases are considered a required setup time of greater than 2 time units will appear to be violated at $t16$, where the distance between worst edge of x and best edge of c is only 2 time units. On the other hand, if we consider that the $g0$ gate has contribution to both x and c signals, we deduce that there is at least 4 time units between edge of x' at $t12$ and edge of c at $t16$.

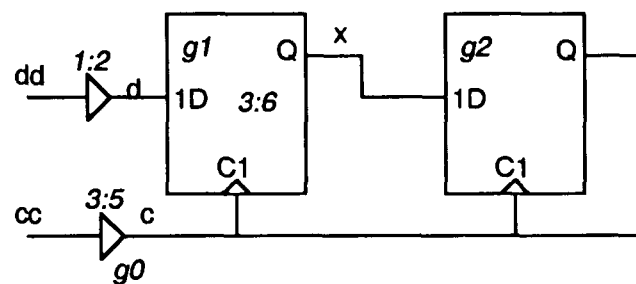


Figure 3. Clock signal timing contribution

Figure 3 shows a two bit shift register consisting of two flip-flops with a common clock signal. In checking the setup and hold times of the second flip-flop, it must be considered that the check in the second flip-flop should be against the same clock edge that generates signal x in the first flip-flop.

Therefore, the contribution of the gray area of gate $g0$ must only be considered once.

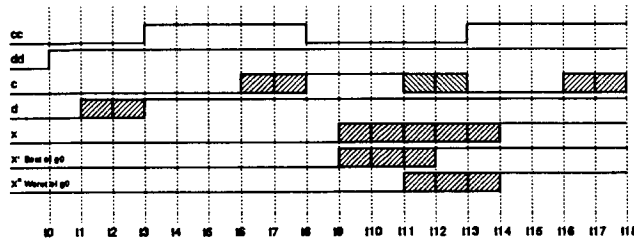


Figure 4. Setup and hold timing

3. Timing Verification

The discussion of the previous section indicates that there are two different timing verifications. In the first example, the concern was the time range in which the output signal makes a 0 to 1 or a 1 to 0 transition. This is a delay timing and is not in reference with any other transition on this or any other signal. We will refer to this timing as *absolute*. The timings referred to in the other examples of the previous section is *relative*.

In general delays are absolute values. A signal has a certain deal in making a transition. A gray area on this signal can specify the range of the delay.

In a relative timing, the time of an event on an edge is compared to the timing of another edge. The edges that are being compared together may or may not be on the same signal. For relative timings, gray areas of more than one edge must be considered into timing checks. Timing checks for minimum pulse width, maximum pulse width, setup time, and hold time are of the relative kind. In the minimum and maximum pulse width, gray areas of different edges of the same signal take role, while in checking the setup and hold time, gray areas of different signals are participated.

4. A Minmax Signal

For timing evaluation and appropriate timing checks, special characteristics must be given to signals that are used for interconnecting components. The previous sections suggest that gray area information as well as information as to the source of delay values must be present when needed. For this purpose we are defining a minmax signal that carries all such information.

As shown in Figure 5, a minmax signal consists of gray areas any time it makes a transition from a logical value to another. We will refer to the gray area corresponding to the 0 to 1 transitions as i ,

and the gray area corresponding to a 1 to 0 transition as o . Therefore, i signifies a time window during which the signal can turn from 0 to 1, and o signifies a time window in which signal is turning from 1 to 0.

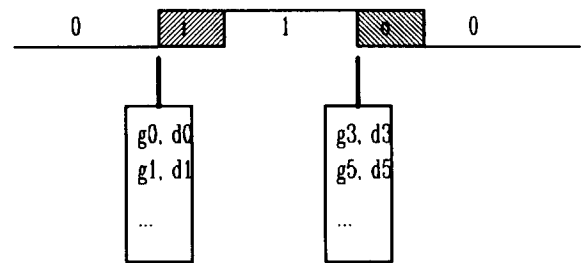


Figure 5. Minmax Signal

At the start of each gray area a record indicates all gates and their contribution to the gray area. We will refer to this as edge record. An edge record can have any number of elements. Each element has two fields, the first field is a gate identification and the second field is the delay contribution of that gate. The delay contribution of a gate is the difference between the maximum and the minimum delay values of the gate. It is assumed that all output signals are delayed by the minimum delay values of gates.

Figure 5 shows that an edge record appears at the beginning of a gray area. In cases where gray areas on various signals overlap, an edge record may also appear at the end of a gray area. Such cases usually signify a potential positive or negative hazard. In general, a signal defined as shown in Figure 5, can be used for proper minmax simulation and all absolute and relative timing checks. In a circuit for minmax simulation all signals must be of the type defined here.

5. Using Minmax Signals

Figures 2 and 3 show cases that the knowledge of the source of a delay is essential in performing timing checks such as pulse width and setup and hold time. Inclusion of the information of Figure 5 helps performing these timing checks. The delay (maximum - minimum) of a gate contributing to two edges must be considered only once.

6. VHDL Implementation of Minmax Models

The VHDL implementation of the models mainly consists of a package that includes the definitions and functions for minmax signals. In addition to this, using these fundamental issues in various models also deserves attention. In this section, we will first present a VHDL package that includes the appropriate minmax

utilities. This will be followed by sections presenting models that utilize the package of definitions.

6.1 MINMAX UTILITIES

Utilities for the minmax simulation include minmax delay scheme, minmax signal, and minmax overloading of logical functions.

6.1.1 Minmax Delay Scheme

Figure 6 shows the delay scheme utilities. Function *get_id* return a unique identification number hen it is called. This function is used by gates for identifying their delay value contribution to signals. The present implementation uses file IO for this function. Each time an id is returned the information in a text file is updated. Shared variables of VHDL'92 can be used for a more efficient implementation of this function.

```

FUNCTION get_id RETURN NATURAL;
TYPE lo_hi IS (lo, hi);
TYPE mm_delay IS (minmax, at_min, at_max);
TYPE delay_scheme IS
  ARRAY (lo_hi, mm_delay) OF TIME;
  
```

Figure 6. Delay Scheme Utilities

The *mm_delay* type defines three timing simulation schemes. This enables a gate model to operate at minmax, minimum, or maximum delay values. Type *lo_hi* defines an array index for the low and high delay values. Array type *delay_scheme* will be initialized to the minimum and maximum delay values of gates as shown in Figure 7.

		<i>mm_delay</i>		
		<i>minmax</i>	<i>at_min</i>	<i>at_max</i>
<i>lo_hi</i>	<i>lo</i>	minimum value	minimum value	maximum value
	<i>hi</i>	maximum value	minimum value	maximum value

Figure 7. Delay Schemes

6.1.2 Minmax Signal

The most important part of the VHDL implementation of the minmax simulation is the definition of the minmax signal. Figure 8 shows the declaration of *mm_signal* that corresponds to the waveform shown in Figure 5. A minmax signal is a record of wave and edge. The wave part of the signal can takes one of the values in the enumeration type *mm_wave*, which is '0',

'o', '1', or 'i', these enumeration elements represent values shown in Figure 5 and discussed in Section 4. The edge type is the *mm_edge* record that consists of a natural number for gate identification and a positive time value for the gate delay contribution. This number is the difference off a gates maximum and minimum delay values. A gate that sets a signal to a certain value, appends an *mm_edge* record to the already existing *mm_edge* record values. All gates contributing to the delay of an edge append such records to the *mm_edge* field of a signal. This implementation separates different contributions by one femtosecond. Therefore if an edge has delay contributions from *n* different gates, in the next *n* femtosecond after the *mm_signal* changes its values the *mm_edge* values of the *n* gates appear at one femtosecond intervals on the *mm_edge* field of the signal. A gate producing a signal generates such values, and a gate checking timings uses these values for relative timing checks. Edge values are carried through a signal and they are removed when the that they correspond to is gated out by logical operations or removed in a flip-flop clocking operation.

From the above discussion it is clear that edge values are multiplexed in time after the edge that they belong to. Multiplexing between delta values instead of femtoseconds was also considered as alternative. The implementation of this method was considered but was not done due to difficulties in transporting events that are delta time apart. In any case, either implementation requires complex scheduling and retrieving of edge record values.

```

SUBTYPE delta_time IS
  TIME RANGE 0 FS TO TIME'HIGH;
TYPE mm_wave IS ('0','o','1','i');
TYPE mm_edge IS RECORD
  gateid : NATURAL;
  offset : delta_time;
END RECORD;
TYPE mm_signal IS RECORD
  wave : mm_wave; edge : mm_edge;
END RECORD;
  
```

Figure 8. Minmax Signal Definition

Another method of implementation is to generate a shard variable linked list structure and associate pointers to the linked list with each edge of a signal. In this implementation the timing of the edge and the identification of a signal will be used to point to appropriate linked list. The details of this

implementation should be considered when shared variables are finalized in VHDL 93.

6.1.3 Function Overloading

Type definitions shown in Figure 9 are used for defining logical values of the new *mm_signal* type. The one dimensional type is used for the single input functions and the two dimensional type is used for two inputs functions such as AND, OR and NAND.

```
TYPE mm_wave_1d IS
  ARRAY (mm_wave) OF fast_slow;
TYPE mm_wave_2d IS
  ARRAY (mm_wave, mm_wave) OF fast_slow;
```

Figure 9. Array types for overloading

Logical functions are overloaded to include the new values 'i' and 'o'. An AND function is shown in Figure 10. ANDing two minmax values results in one signal for the fast case and one signal for the slow case. Recall that the 'i' value represents a value that is making a '0' to '1' transition, and 'o' is the value of a signal that is making a '1' to '0' transition. As an example, assume that one input of an AND gate is '1' and the other is 'i', then the best that can happen (fastest input) is for the output to become '1'. On the other hand, the worst that can happen (slowest input) is that the 'i' input still holds its '0' value and the output remains at '0'. For this reason, the value ('1', '0') appears in the intersection of '1' and 'i' in the AND table, to give values of fastest and slowest events. Note that these are values generated by the inputs and not delayed by the gate itself.

```
CONSTANT mm_and_table : mm_wave_2d :=
  ( -- '0',   'o',   '1',   'i'
    -----
    (('0','0'), ('0','0'), ('0','0'), ('0','0')), --'0'
    (('0','0'), ('0','1'), ('0','1'), ('0','0')), --'o'
    (('0','0'), ('0','1'), ('1','1'), ('1','0')), --'1'
    (('0','0'), ('0','0'), ('1','0'), ('1','0'))); --'i'
```

Figure 10. AND function in the minmax value system

Function declarations for basic logical operations are shown in Figure 11. This figure also shows the *combine* utility function. This function combines two signals having bit values into a minmax

wave signal. We will describe the usage of this function in the next section.

```
FUNCTION "AND" (i1, i2 : mm_wave)
  RETURN fast_slow;
FUNCTION "OR" (i1, i2 : mm_wave)
  RETURN fast_slow;
FUNCTION "NOT" (i1 : mm_wave)
  RETURN fast_slow;
FUNCTION pass (i1 : mm_wave)
  RETURN fast_slow;
FUNCTION combine (best, worst : mm_wave)
  RETURN mm_wave;
```

Figure 11. Basic logical functions

6.2 MINMAX MODELS

Using the utilities presented above, and using the techniques discussed in the earlier parts of this paper, we will illustrate the basic models for minmax simulation. Generation of wave and timing parts of output of an AND gate will be discussed.

```
waving : PROCESS (i1.wave, i2.wave)
BEGIN
  delayed_best <= "AND"(i1.wave, i2.wave).fast
    AFTER delay (lo, del);
  delayed_worst <= "AND"(i1.wave, i2.wave).slow
    AFTER delay (hi, del);
END PROCESS waving;
o1.wave <= combine (delayed_best, delayed_worst);
```

Figure 12. Wave process, generating the output wave

6.2.1 Waveform Generation

Figure 12 shows a process statement in an AND function for calculation of the minmax wave signal. A fast wave value and a slow wave value are looked up from the table of Figure 10. The fast signal is delayed by the least delay of the AND gate to form the best signal, and the slow signal is delayed by the most amount of delay to form the worst signal. These two signals are then combined to form the output of the AND gate. This implementation allows actual signal values to use the inertial delay model, while gray areas are transported to the output of a gate. The gray areas on the output appear after the main signal values have been delays, therefore, the inertial delay of the fast and slow signals do not apply to them.

```

edging : PROCESS
  VARIABLE new_out_f, new_out_s,
           old_out_f, old_out_s : mm_wave;
  VARIABLE
    self_considered : BOOLEAN := FALSE;
BEGIN
  IF i1'EVENT THEN
    ...
  END IF;
  IF i2'EVENT THEN
    ...
  END IF;
  WAIT ON i1.wave, i2.wave;
END PROCESS edging;

```

Figure 13. General structure of the edging process

```

IF i1'EVENT THEN
  new_out_f :=
    "AND"(i1.wave, i2.wave).fast;
  new_out_s :=
    "AND"(i1.wave, i2.wave).slow;
  old_out_f :=
    "AND"(i1.wave'DELAYED, i2.wave).fast;
  old_out_s :=
    "AND"(i1.wave'DELAYED, i2.wave).slow;
  IF (new_out_s /= old_out_s) OR
     (new_out_f /= old_out_f) THEN
    self_considered := FALSE;
    get_contributors : LOOP
      WAIT FOR 1 FS;
      EXIT get_contributors WHEN
        NOT i1.edge'EVENT;
      o1.edge <= TRANSPORT i1.edge AFTER min;
      IF i1.edge.gateid = gate_id THEN
        self_considered := TRUE;
      END IF;
    END LOOP get_contributors;
    IF NOT self_considered THEN
      o1.edge <= TRANSPORT (gate_id, max - min)
        AFTER min;
    END IF;
  END IF;
END IF;

```

Figure 14. Appending edge records

6.2.2 Signal Edge Record Generation

In addition to the waveform described above, every transition can have an edge information associated

with it. The edge information are caused by input changes, therefore, the process for generating such information is sensitive to all inputs of the primitive cell that is being modeled. Figure 13 shows such a process for a 2-input AND gate. For every input, the process includes a section that assigns edge events from a changed input to the output.

The dotted portions of the edging process that correspond to the changing of the *i1* input is shown in Figure 14. When an input changes, the new value and the old value of the output are calculated. If the input change has caused a change on the output, then edge records from the input are attached to the output signal. When done, the edge record of the AND gate, i.e., the gate number and its delay offset (max-min), is appended to the end of the other records. A variable, called *self_considered*, is used to keep track of the assignment of the edge record of the gate that is being processed (the AND gate in this case) to its output. If, because of feedback, the record of the present gate appears on one of its input edges, a new record for this gate will not be appended to its output.

For transporting edge records from a changed input to the output (the loop of Figure 14), the input edge *i* searched for events every 1 femtosecond. If an event is not found, it is concluded that no more edge records are present and the loop terminates. If a record is found, it is assigned to the output edge after the minimum gate delay using TRANSPORT delay model. For multi-input gates, the above code section is repeated for every input.

6.3 TIMING CHECKS

Using the information appended to each edge of a signal, timing checks can be performed. For each relative timing check, such as pulse width or setup and hold time, the edge information from the two edges being compared are extracted and compared. In this comparison contribution of the same gate to two edges will be considered in the calculation of timing. For example, in comparing setup time at *t16* in Figure 4, when the edge information of signal *x* indicates that it contains 2 time unit delays contributed by *g0*, and the same gate also appears in the edge information of the *c* signal, the setup time check procedures adds the contributed amount to the setup tolerance.

The VHDL code for this purpose implements strategies set forth in Section 5. A procedure collects all edge information. When relative timing checks are to be done, corresponding collected edge information are searched for equivalent gate numbers. Depending on the type of timing check, timing contributed by the same gates are either considered once or they are not considered in the timing check.

Conclusions

In this work, we have outlined the general strategies for including information with signals such that they can be used for minmax simulation. We have defined general timing procedures and have shown how gate models should transmit their timing values to their outputs. We have also shown VHDL models of basic gates for minmax simulation. Although the VHDL language was primarily used here, the techniques developed in this paper apply to general simulation of gate level models. In general, we are concluding that for minmax simulation, a record in the form of a linked list must be associated with edges of all signals. Because of the lack of support for shared variables in VHDL'87, we had to time multiplex this information into a signal. The inclusion of shared variables in VHDL'93 significantly improves the implementation of the techniques presented here.

REFERENCES

- Bowden, M. A., "Design Goals and Implementation Techniques for Time-Based Simulation and Hazard Detection," Digest of Papers 1982 International Test Conference, pp. 147-152, November 1982.
- Navabi, Z., "VHDL: Analysis and Modeling of Digital Systems," McGraw-Hill Publishing, Inc., New York, 1993.
- Abramovici, M., M. A. Breuer, A.D. Friedman, "Digital System Testing and Testable Design," Computer Science Press, New York, 1990.
- Rajsuman, Rochit, "Digital Hardware Testing: Transistor-Level Fault Modeling and Testing," Artech House, Inc. Boston, 1992.