Event-driven gate-level logic simulation using a timing wheel data structure

ECE-470 Digital Design II

Imagine how the circuit in Fig.1 can be simulated. In other words, the job of the gate-level simulator is to apply an input vector at the "abc" primary inputs (PIs) and compute the response values at the "g" primary output (PO), considering the gate and wire delays. Gate delays are shown inside the gate symbols while wire delays are assumed to be zero in this example. The simplest way to think of it is to process all gates in the circuit, going from PIs toward POs. The gates will be evaluated in the right order according to their logic depth/level. The logic value at the output of a given gate will be set and propagated according to the gate delay. A logic gate or logic cell (AND, OR, and so on) is treated as a black box modeled by a function whose variables are the gate-input signals. Setting all the delays to unit value is the equivalent of functional simulation. In the example from Fig.1, for an input abc=111 the gate level simulator should be able to compute the output response g=1 at time t=8. This process can be repeated for any other input vector applied at the PIs. However, this process can be simplified by observing that not all gates in the circuit change their output values for a subsequent input vector. For example, if in Fig.1, only "a" changes from 1→0 while "b" and "c" remain 1, then only gates "e" and "g" will changes their output values. Therefore, the simulation of the input vector abc=011 needs to only evaluate two gates (i.e., two nodes) instead of all four gates, in order to be able to compute the PO "g". This speeds up the runtime of the simulator.

The signals changing values are called active nodes or signals. The ratio between the number of active signals and the total number of signals in the circuit is referred to as activity. In general, the activity of a digital circuit is between 1-5%. This fact forms the basis of **activity-directed simulation**, also known as **event-driven simulation**.

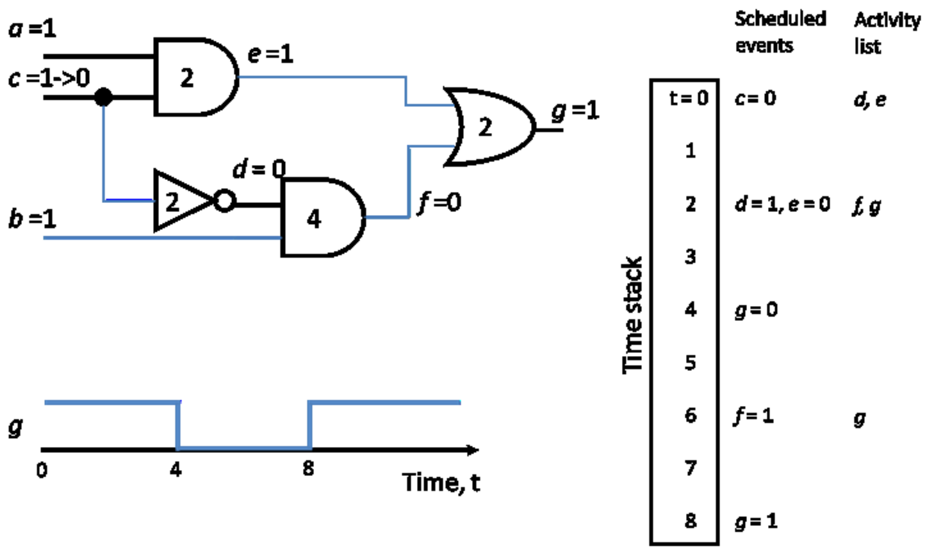

Fig.1 Circuit example. Illustration of Scheduled events (event-list) and Activity list.

Event-driven simulation is the most common type of digital simulation. An event driven simulator uses a structural model of a circuit to propagate events. When a circuit node (i.e., signal) changes in value, the time, the node, and the new value are collectively known as an **event**. The event is scheduled by placing it in an **event queue** or **event list**. For example, in Fig.1, the event list will contain the events under "Scheduled events" column. When the specified time is reached, the logic value of the node is changed. The change affects logic cells that have this node as an input. All of the affected logic cells must be **evaluated**, which may add more events to the event list. The simulator keeps track of the current time, the current **time step**, and the event list that holds future events. For each circuit node the simulator keeps a record of the logic state. When a node changes its logic state, whether as an input or as an output of a logic cell, this causes an event.

The event list keeps track of logic cells whose outputs are changing and the new values for each output. The **activity (evaluation) list** keeps track of logic cells whose inputs have changed. Using separate event and evaluation lists avoids any dependence on the order in which events are processed, since the evaluations occur only after all nodes have been updated. The sequence of event-list processing followed by the evaluation-list processing is called a **simulation cycle**, or an event-evaluation cycle.
For example, in Fig.1, the first simulation cycle at time t=0 processes the event c=0 (the result of which is setting c to zero logic), which had been initially scheduled for this time t=0 (t=0 is now the current **simulated time**) due to the new input vector abc=011 being applied at PIs. Also part of this simulation cycle is the processing of elements "d" and "e" from the activity/evaluation-list (these elements had been inserted due to the event c=0 from the event-list), which has as a result the insertion in the event-list of the new events d=1, e=0, scheduled at time in the future t=2 (relative to the current simulated time).

One way to implement the event list ("Activity list" in Fig.1) is to create an array with entries indexed by time. Each entry in this array could be a linked list whose entries would store the scheduled events for the corresponding time (index in the "Time stack' of Fig.1). An elegant way of implementing such an array is the **time wheel**. Delays are tracked using a time wheel divided into ticks or slots, with each slot representing a unit of time. See Fig.2 for a pictorial representation. A software pointer marks the current time on the timing wheel. As simulation progresses, the pointer moves forward by one slot for each time step. The event list tracks the events pending and, as the pointer moves, the simulator processes the event list for the current time. Note that the number of slots in the time wheel has to be large enough so that the largest delay in the circuit can be accommodated. For example, the minimum size of the timing wheel necessary to simulate the circuit in Fig.1 is 5, so that events that are four time units ahead in time can be scheduled (four is the largest delay among all the gates in the circuit netlist).

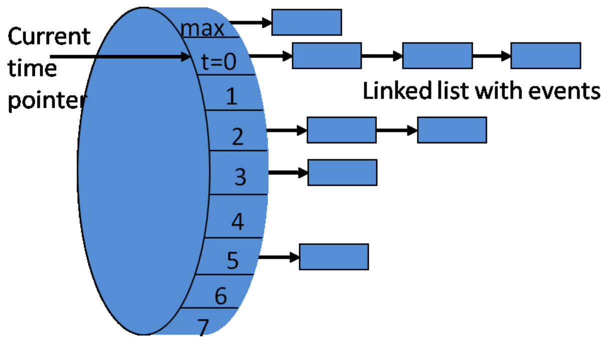

Fig.2 Time wheel representation.

References

[1] Chapter 3 of Digital Systems Testing and Testable Design, Miron Abramovici, Melvin A. Breuer, Arthur D. Friedman, Wiley-IEEE Press, Revised edition, 1994.
[2] Section 5.4.2 of Essentials of Electronic Testing for Digital Memory and Mixed-Signal VLSI Circuits, M. Bushnell, V. Agrawal, Springer 2000.