Tools For Validating Asynchronous Digital Circuits

Aaron Ashkinazy[1], Doug Edwards[2], Craig Farnsworth[2], Gary Gendel[1], and Shiv Sikand[2]

Abstract

Asynchronous design methodologies can yield designs that are smaller, and/or consume less power, than their synchronous counterparts. Traditional tools, oriented toward synchronous designs, may miss critical asynchronous design problems. This paper describes the modeling methodology and hazard analysis of the SIMIC logic simulator that address asynchronous designs. It also describes tools and a methodology for generating accurate timing models from SPICE simulations and for analyzing and viewing dynamic power consumption. Finally, it presents a case study illustrating the use of these tools in a leading-edge asynchronous design.

1. Simulation overview

Since the ultimate goal of the design verification process is to manufacture working parts, the simulation phase must, in addition to modeling a design's functionality and timing as accurately as possible, also predict its operation if event timing could vary somewhat (e.g., due to changes in processing parameters or supply voltage). In general, this may require multiple simulations with different component delay distributions. However, since today's designs could easily contain over 100,000 devices, manual analysis of the (voluminous) simulation results by itself cannot be expected to uncover every timing problem. Thus, the simulator must at least be capable of directing the designer to problematic sections of the logic.

Since circuit-level simulators such as SPICE, whose domain is voltages and currents, provide the greatest modeling accuracy and coupling to manufacturing parameters, they have been the mainstay for verifying and characterizing cell libraries for both synchronous and asynchronous applications, and for analyzing delays along critical paths *once these paths have been determined.* Simulating the entire design at this level is prohibitive, however, even with the speedup provided by "timing simulators", which exploit latency in MOS circuits and utilize relaxation-based iteration and/or lookup tables rather than analytical models. Even if the barrier of execution time didn't exist, it is very difficult to isolate and identify, at this level of abstraction, many types of hazard conditions (e.g., setup and hold time violations, essential hazards, functional hazards) that could cause different operation with minor changes in event timing. Also, there is no way to represent indeterminate values, that is, the values of signals that are unknown during simulation but will either be logical 0 or logical 1 in the actual circuit (e.g., the values of state variables at power up, the state of a flip-flop after a setup-time violation occurs, etc.).

Switch-level simulators abstract analog node voltages to logical levels, which can increase throughput considerably. They also solve the initial state problem by adding an uninitialized logic state. However, current switch-level simulators have problems with timing accuracy and, significantly, with reconvergent fanout of an unknown state. For example, a switch level simulator will incorrectly set the output of the multiplexer of Figure 1 to X (unknown) when the two data inputs have the same value and the control input is unknown.

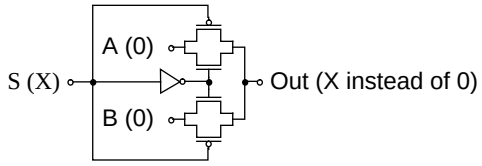


FIGURE 1 Switch model of 2-input multiplexer

Gate-level and systems-level simulators can provide fast throughput by utilizing higher-level primitive elements and behavioral models. They can, therefore, eliminate many switch-level problems by encapsulating reconvergent fanout within a single model (e.g., a functional model of a 2-input multiplexer). Traditionally, these simulators have focused on ease of modeling and/or simulator throughput, neglecting some fundamental circuit properties (for example, nonlinear delay vs. loading characteristics, input-slew-dependent delays, merging the out-

1. Genashor Corp, 9 Piney Woods Drive, Belle Mead, NJ 08502, USA (908) 281-0164, genashor@pluto.njcc.com
2. AMULET Group, Department of Computer Science, The University, Oxford Road, Manchester M13 9PL, UK, farnswoc@cs.man.ac.uk

put characteristics of wire-tied drivers) that may cause gross errors in timing. Errors of 400% have been seen in many "good" models at points over the expected operating range of the cells.

Modern gate-level simulators support various features to detect and locate timing problems. Almost all support spike propagation (generation of an X-pulse when a "glitch" occurs, i.e., when an element's inputs change too quickly for its output to respond) and timing checks. However, these timing checks are primarily oriented toward synchronous designs, where proper clocking (glitch-free clocks with adequate pulse-widths and periods) and adherence to gross safety margins for setup and hold times can usually avert any unexpected problems. For this reason, designers of asynchronous circuits do not have a high degree of confidence in current generation systems- and/or gate-level simulators.

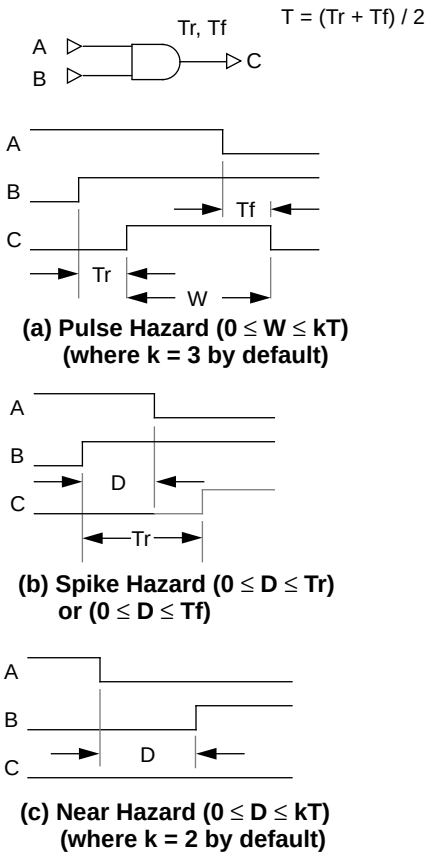
Some simulators utilize a bounded delay methodology, allowing element delays to range between their minimum and maximum extremes. Ambiguity region ("min-max") analysis propagates rising and falling levels within bounded regions and utilize element-specific rules to set outputs to the unknown state when the transition regions of the element's inputs overlap. This approach tends to be overly-pessimistic, even when "correlated delay" and reconvergent-fanout analysis is used to reduce the overlap of related inputs. Monte Carlo simulation is arguably the best bounded delay approach; some judgement is necessary, however, in determining the number of simulations required to reach a comfortable level of confidence.

2. Hazard analysis for asynchronous designs

The SIMIC logic simulator provides the benefits of gate-level simulation with accuracy typically within 20% of SPICE timing. It supports a robust set of hazard checks to trap potential problems in both asynchronous and synchronous designs. Additionally, SIMIC can be run as an interactive debugging tool to quickly locate and correct timing problems.

In addition to the flip-flop setup-time, hold-time, and clock/set/reset pulse-width checks, SIMIC supports checks for wire-tie conflicts, oscillations (excessive activity in response to a primary input event), combinational hazards, plus a number of other useful checks. The combinational hazard checks can direct the designer not only to the origin of a manifest timing problem, but also to sections of the logic that are dangerously close to malfunctioning, *even though circuit operation is correct for the delay distribution being simulated*.

The hazard categories are illustrated in Figure 2 for an AND gate, where T_r and T_f are respectively the output's rise and fall delays, and T is the output's average propagation delay:



(a) Pulse Hazard (0 ≤ W ≤ kT) (where k = 3 by default)

(b) Spike Hazard (0 ≤ D ≤ Tr) or (0 ≤ D ≤ Tf)

(c) Near Hazard (0 ≤ D ≤ kT) (where k = 2 by default)

FIGURE 2 Combinational Timing Hazards

(1) Pulse hazard – a "narrow" pulse on a signal whose width is comparable to the signal's average propagation delay. The user can define "narrow"; by default, it is 3× the average propagation delay. In general, narrow pulses are unplanned, and could grow wider or disappear with perturbation of delays due to variations in processing parameters and supply voltage. Pulse hazard checks are supported for all element types.

(2) Spike hazard – a pair of events at an element's inputs in which the second event arrives before an element output can respond to the first event. This is the classic "glitch". In general, the operation of the actual circuit is indeterminate. However, to avoid overly-pessimistic results when the second input event closely follows the first (so that the output signal could not have begun to respond to the first event), the user can define a threshold interval on a per-signal basis that filters out "innocuous" transients. Spike hazard checks are supported for all element types.

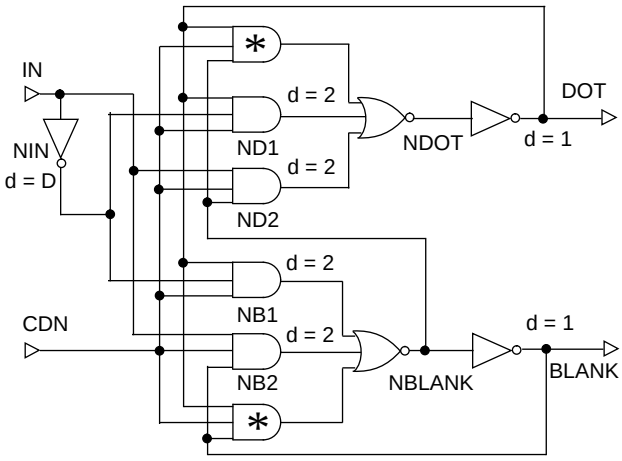
(3) Near hazard – a sequence of input events within an interval in which an element's output response would have been different had the events occurred in a different order. For example, in Figure 2(c), instead of remaining at a constant logical 0, signal C would pulse or spike if the order of the transitions at inputs A and B were reversed. The user can define the analysis interval; by default, this interval is twice the output's average propagation delay. Near hazard checks are supported for all combinational primitives.

The user can independently enable or disable each of the combinational hazard checks on a per-signal basis. When a hazard is detected at an enabled signal, SIMIC will issue a warning message describing the hazard. The user can also (and independently) direct SIMIC to temporarily set a signal to the unknown value (for the duration of the transient) when a spike hazard or a near hazard has been detected. This X value will be then propagate along the signal's fanout cone, and if the timing problem can cause a steady-state error, the X value will ultimately reach, and latch into, the subset of state variables that could be affected.

These timing and hazard checks can be very effective for detecting, locating, and correcting design problems in asynchronous circuits:

- If a near hazard occurs at a signal along a sensitized path to a state variable, the circuit may be close to malfunctioning, though it may be operating correctly with the simulated delay distribution.
- If a spike hazard is sensitized to a state variable, the final state of the circuit may be uncertain.
- If a pulse hazard is sensitized to a state variable that should, at most, execute a single transition in response to a single input event (i.e., there are no transient states), the circuit may already be operating incorrectly.

Consider, for example, the circuit shown in Figure 3, which is a gate-level representation of a toggle (T) flip-flop in one cell library used at the University of Manchester [Yant92]. For simplicity, the rise and fall delays of each element are assumed to be equal; for example, the propagation delay of each multiplexor AND gate is 2 time-units. The two gates labeled with an asterisk are redundant consensus terms generated to eliminate switching transients in the model, and are not in the physical cell (which utilizes charge storage in a functional block to achieve the same result). This model is a slightly simplified representation of the model generated by the SPICE-to-gate-level extractor, SP2LOG, described later.



	CDN IN: 00	01	11	10	BLANK	DOT
1	(1)	(1)	2 ←	(1)	0	0
2	1	1	(2)	3	0	1
3	1	1	4	(3)	1	1
4	1	1	(4)	1	1	0

FIGURE 3 Toggle cell and its flow table

This cell's flow table exhibits (in fact, is the classic example of) an *essential hazard*, which is defined as [Ung69]: "For some initial total state and input variable *x*, three consecutive changes in *x* take the system to a state that is different from (and not equivalent to) the state reached after a single *x*-change". Typically, unless the delay distribution is such that an essential hazard manifests itself to produce the wrong final state, it is difficult (if not impossible) for simulation to uncover its existence without the "what-if" hazard checks described above.

In this case, the essential hazard manifests itself when the toggle cell outputs DOT and BLANK are lightly loaded, so their propagation delays are small, and input IN is heavily loaded, so the slew-dependent delay at the output of inverter NIN, labeled D in the figure, is relatively large. In particular, if the circuit is initially stable in State 1 with the inputs CDN = 1 and IN = 0, and if the output delays at DOT and BLANK are 1 time-unit, as shown in the figure, then when input IN executes a 0→1 transition (see arrow in flow table), the final state will be State 4 instead of State 2 when $D \geq 5$. Furthermore, if simulation is performed with $D < 5$, the final state will be State 2, and there will be no indication that the circuit is close to malfunctioning *unless* the hazard checks are enabled.

```
C=                CI N ND NB
C=                DN I DO BL
C=                N  N OT LA
C=                     T  AN
C=                        NK
C=                        K
```

time *test*

```
  0 T     1: 00 X XX XX
  2 T     1: 00 1 1X 1X
  3 T     1: 00 1 10 10
 50 T     2: 10 1 10 10
100 T     3: 11 1 10 10
102 T     3: 11 0 00 10
103 W     3> NEAR NB1 (AND)
103 W     3> NEAR ND1 (AND)
103 T     3: 11 0 01 10
```

(a) SIMIC Terminal output for D = 2

```
C=                CI N ND NB
C=                DN I DO BL
C=                N  N OT LA
C=                     T  AN
C=                        NK
C=                        K

  0 T     1: 00 X XX XX
  2 T     1: 00 X 1X 1X
  3 T     1: 00 X 10 10
  4 T     1: 00 1 10 10
 50 T     2: 10 1 10 10
100 T     3: 11 1 10 10
102 T     3: 11 1 00 10
103 T     3: 11 1 01 10
104 W     3> SPIKE(0->1->0) NB1
104 W     3> SPIKE(0->1->0) ND1
104 T     3: 11 0 01 10
```

(b) SIMIC Terminal output for D = 4

```
C=                CI N ND NB
C=                DN I DO BL
C=                N  N OT LA
C=                     T  AN
C=                        NK
C=                        K

  0 T     1: 00 X XX XX
  2 T     1: 00 X 1X 1X
  3 T     1: 00 X 10 10
  5 T     1: 00 1 10 10
 50 T     2: 10 1 10 10
100 T     3: 11 1 10 10
102 T     3: 11 1 00 10
103 T     3: 11 1 01 10
105 W     3> NEAR NB1 (AND)
105 W     3> NEAR ND1 (AND)
105 W     3> NEAR NDOT (NOR)
105 T     3: 11 0 01 00
106 T     3: 11 0 01 01
107 W     3> PULSE(0->1->0 Width=2 Widh/Delay=1.00) NB1
107 W     3> PULSE(0->1->0 Width=2 Width/Delay=1.00) ND1
107 W     3> PULSE(0->1->0 Width=5 Width/Delay=2.50) ND2
107 T     3: 11 0 11 01
108 T     3: 11 0 10 01
```

(c) SIMIC Terminal output for D = 5 exhibiting the essential hazard

FIGURE 4 SIMIC simulations of the Toggle Cell for different values of input delay

Three SIMIC simulations were performed for the toggle circuit, as shown in Figure 4. Each applies the same three tests (distinct primary input states); the first test, at time 0, sets CDN to 0 to initialize the circuit to State 1, and the second test, at time 50, sets CDN to 1 to bring the circuit to the desired total state. The third test, at time 100, is the 0→1 transition at IN. Warning messages for near and spike hazards were enabled, and X propagation for these hazards were disabled. With D = 2, the circuit correctly enters State 2, as shown in Figure 4(a). However, the near hazard message for signal NB1 (which occurs for all values of D < 4) indicates that if the order of its input transitions had been different (i.e., had the value of D been greater), the simulation results might have been different. (There is also a near hazard reported at signal ND1, which happens to be the same logical function, but this hazard is non-critical since ND2 is 1 at the time, so ND1 is not sensitized at DOT.) The simulation results for D = 4 are shown in Figure 4(b); the circuit still enters State 2, but now, the 1→0 transition at NIN occurs *after* the 0→1 transition at DOT, resulting in a spike hazard at NB1. This indicates that the final state might actually be indeterminate. With D = 5, NB1 actually executes a 0→1 transition, forcing NBLANK low, and ultimately causing the circuit to enter State 4, as shown in Figure 4(c). Had X propagation been enabled for near and spike hazards in these simulations, both DOT and BLANK would have been forced to X, indicating that circuit operation is really uncertain.

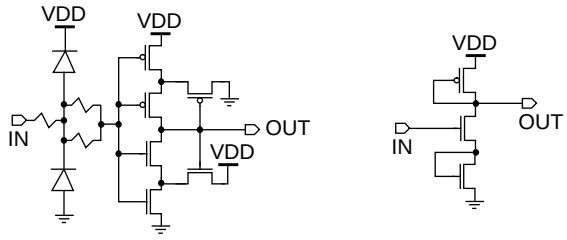
3. Model Accuracy

Since different gate-level models of a given cell can exhibit different transient responses, the importance of selecting a model that captures the nuances of the cell's physical implementation cannot be overemphasized. Without a one-to-one correspondence between the physical design and its logical representation, the accuracy of the delay distribution and consequent hazard analysis may be

seriously compromised. To guarantee a good model, we developed a circuit-to-logic translator called SP2LOG.

SP2LOG translates a SPICE description of a circuit, consisting of transistors and capacitors, to a gate-level description. It attempts to retain all important functional nodes of the SPICE description in the final model. In addition, SP2LOG can either calculate delay characteristics at nodes from user-defined equations, or provide symbolic references into delay tables supplied by alternate means. SP2LOG utilizes boolean analysis of the pullup and pulldown path functions for each node, coupled with sophisticated conflict resolution analysis. After the analysis, the resulting expressions are then examined to determine the best gate-level representation. Since the resulting model accurately represents the physical implementation, the SP2LOG models are applicable to fault analysis as well as fault-free analysis.

Even in "pure digital" designs, there are several subcircuits whose behavior is sufficiently non-digital to confuse a switch-level simulator. Typically, but not always, these circuits are located in the pad circuitry; Schmitt trigger input pads and TTL level shifters are good examples, as illustrated in Figure 5. If the effective ON resistance of the P and N devices are comparable, and if either circuit is simulated at the switch-level as show, its output will always remain in an unknown, X, state. However, SP2LOG recognizes the fact that the effective resistance of a p-type transistor used as a pulldown, or of an n-type transistor used as a pullup, is significantly greater than the resistance of the transistors in their normal configurations. It also recognizes the fact that when a pullup or pulldown transistor has its gate connected to its drain, it effectively functions as a large resistance. Thus, the SP2LOG model for the circuits of Figure 5 would be an inverter.



(a) Schmitt Trigger input pad **(b) Level Shifter**

FIGURE 5 Quasi-digital circuits

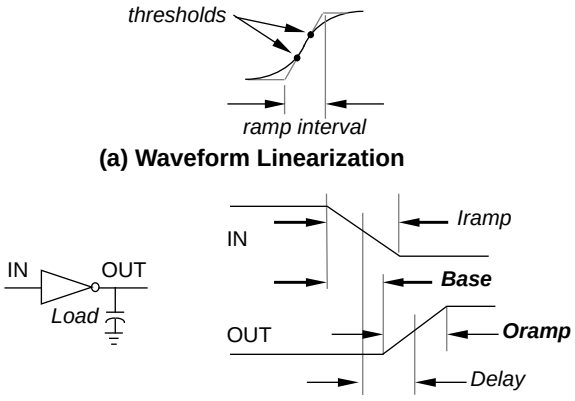
4. Timing Accuracy

For many current technologies, the effects of non-zero input rise and fall times (so-called "slew-rates") must be incorporated in order to achieve a high degree of accuracy in modeling propagation delays and performing hazard analysis. We quickly learned that the dependence of propagation delays on output loading and input slew rate is *not* linear over the expected operating range, and that using a single, simple linear relationship introduces unacceptable errors. SIMIC therefore allows delays to be described by two-dimensional piecewise-linear tables to represent the non-linear relationships. SIMIC will warn if the loading or input-slew rates exceed the measured data limits but will extrapolate from the nearest segment in the table.

As illustrated in Figure 6(a), the complex rise or fall waveshape obtained from circuit-level simulation is approximated by a linear ramp. To generate this ramp from the original waveshape, we project the central, linear portion of the rise (fall) waveshape to the rails. This is usually accomplished by constructing a line that passes through two threshold points that bracket the 50% point, sufficiently far enough away to reduce numerical errors, but close enough so the non-linear "tails" of the curves do not significantly affect the results. In the technologies that we have worked with, we have found the 35% and 65% points to be satisfactory.

Delays are characterized by two output waveform parameters, as shown in Figure 6(b); the base delay, *Base*, which is the time interval from the start of the input ramp to the start of the output ramp, and (2) the output ramp interval, *Oramp*. Each piecewise-linear entry in the delay table contains the values six coefficients, *a* through *f*, that relate the two delay parameters to output loading (*Load*) and input ramp interval (*Iramp*) using the following equations:

$$Base(a,b,c) = a + (b \times Load) + (c \times Iramp)$$
$$Oramp(d,e,f) = d + (e \times Load) + (f \times Iramp)$$



(a) Waveform Linearization

(b) Delay Model Parameters

FIGURE 6 Slew-Rate Delay Model

The propagation delay used by SIMIC is the time interval between the 50% points of the input and output waveforms:

$$Delay = \frac{Oramp - Iramp}{2} + Base$$

5. Parameter Fitting

To utilize the delay model described above, we required a method of automatically capturing the delay coefficient sets (*a* through *f*) for the delay tables. Without an automated means, the task of generating accurate models would be too complex and time consuming. We tried using the Simplex algorithm commonly used to fit delays in the model, based upon measured path delays, and also a simultaneous equation solver, but both methods had problems accurately distributing the delays due to the under-constrained nature of the problem. Exhaustively characterizing every node of every cell used in a design over the entire range of operating conditions would be a computer resource and/or manpower intensive task, and was ruled out as a viable alternative.

We discovered that we could identify a number of basic circuit configurations, or *atoms*, that could be used to build accurate delay models; by characterizing just these atoms, we could cover most of the configurations found in the cell library. In this way, we could collect data on a small number of circuits, and relate them to the rest of the cells with a high degree of delay accuracy.

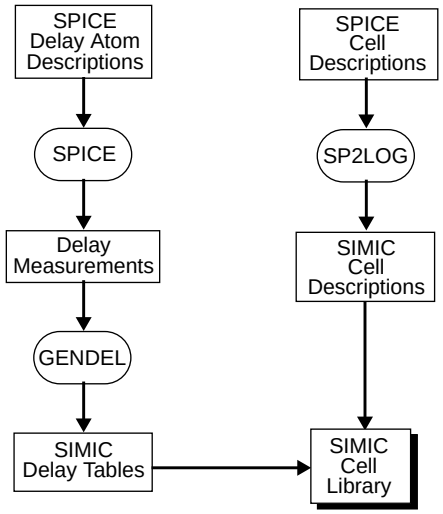


FIGURE 7 Process for automatic model generation

The atoms consist of P-N stacks with each stack ranging from 1 to *n* high, where *n* is the maximum stack size expected in the library of cells. Each possible stack/transistor size configuration is then fully characterized. SP2LOG can be instructed to recognize the atom configurations in the circuit, and reference the proper delay information for that configuration. In this manner, a library can be characterized in a fraction of the time it usually takes, while providing superior models.

A parameter fitting program, GENDEL, was developed to automate the creation of SIMIC delay tables from these measurements. With this tool, we have completed all the tools necessary to totally automate the model generation process, as illustrated in Figure 7.

6. Dynamic Power Consumption Analysis

Since one of the potential benefits of asynchronous design is low power consumption, a tool was needed to assist in the analysis of power consumption. The tool developed, XPOWER, analyzes dynamic power consumption, and displays the results in a number of graphical and textual representations. Windowing on test ranges, and/or node selection, allows the user to focus on any area of interest in great detail.

Dynamic power consumption is calculated from the capacitive energy stored and retrieved at each node whenever a signal transition occurs. This energy is calculated using the formula $E = \frac{1}{2} \times C \times (\Delta V)^2$, where C is the nodal capacitance, and ΔV is the high to low (or low to high) change in voltage, which is usually the supply voltage, Vdd. XPOWER does not currently take into account power consumption arising from leakage currents, shorting currents during transient switching states (through ON transistors), spikes and other transient phenomena, and currents through pullup or pulldown resistors.

XPOWER automatically filters out the power dissipated by external sources at primary input and bidirectional pads. In addition, through a command file, the user can indicate other signal transitions that need to be discounted. The power consumption reports are normalized to mw/Mhz, allowing the results to be easily scaled for any input frequency.

XPOWER can also generate a tailored report in PostScript (with graphs) or ASCII (with tables).

Figure 8 illustrates a Dissipation Graph displaying dynamic power consumption along the vertical axis vs. test vector number along the horizontal axis. This display is typically used to identify areas of interest for more detailed analysis. It clearly illustrates the effect of exercising various circuit functions on overall power consumption of power. One of the interesting side benefits of this display has been the identification of certain timing problems that show up as unusually high power consumption spikes in this graph.

A second display is the summary statistics, shown in Figure 9. This lists summary information for the displayed Dissipation Graph, such as the number of tests, the number of transitions, and the minimum and maximum points in the graph.

A third display is the Distribution Graph, illustrated in Figure 10, with the number of vectors (in which a given level of power consumption occurs) on the vertical axis

FIGURE 8 Dissipation Display from XPOWER

FIGURE 9 Summary Statistics Display

FIGURE 10 XPOWER Display of Distributed Power

and power on the other. This clearly illustrates the range of power consumption and how this consumption is distributed.

XPOWER allows groups of nodes to be excluded from analysis. Thus, the power consumption in selected areas of the design can be determined in dedicated XPOWER sessions. Reports can be generated that include any of the graphs or other information during the session.

7. The University of Manchester, UK, A Case Study

An asynchronous serial-in-parallel-out register (SIPO) with a 2-phase-bundled-data interface, consists of a ring counter and a set of latches (Figure 11). Upon reception of an event on *Rin* the enabled stage of the ring counter sends an event to its associated latch stage. The event is converted to a pulse so the latch can be made transparent and then returned to opaque, storing the data on *Sin* as a consequence.

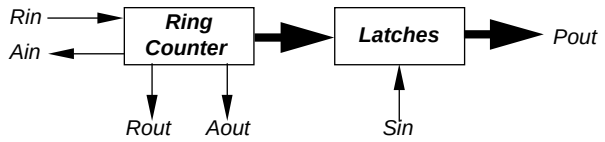


FIGURE 11 SIPO Architecture

A ring counter consists of one event input and *n* event outputs. Each output makes one output transition, in turn, during a cycle. This operation is best demonstrated with a three bit ring counter, see Figure 12(a). Assume the circuit is initialised to the all-zero state. The top Muller C-element will fire in response to an event on *Rin* causing *d0* to fire, enabling the second Muller C-element. Two more events on *Rin* cause *d1* and *d2* to fire respectively, returning the ring counter to its initial state, albeit with inverted absolute levels.

When a ring counter has an even number of bits, see Figure 12(b), the circuit operation is slightly more complicated. On completion of a cycle the ring counter should return to its initial state in which the top Muller C-element is enabled. However, an even number of events has been received, since there are an even number of stages. As a consequence, *Rin* has the wrong absolute value to fire the top Muller C-element, hence the introduction of an additional XOR gate and Muller C-element. The XOR gate inverts *Rin* (*Rin2*) when *d3* is fired and the Muller C-element delays the enabling of the top Muller C-element until the inversion of R*in* is complete so as to avoid a false transition on *d0*.

The SIPO can return bit level completion signal in two ways. The first method delays *Rin* sufficiently. The second method, shown in Figure 12(c), derives completion signals

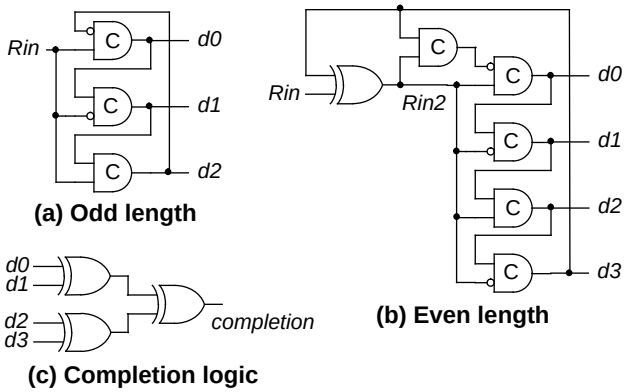


FIGURE 12 Ring counters

from the latching signals using an XOR tree, and is clearly less effective than the first method, particularly as the number of bits in the SIPO increases.

These ring counters rely on the enabled Muller C-element firing after the next Muller C-element has been deactivated by *Rin* so the event on *Rin* does not race through two stages of the ring counter. This situation arises when the edge speed is greater than 51 nS, according to a HSPICE simulation using typical models for both n and p type transistors on the 1 micron ES2 process. Unfortunately, many switch-level simulators lose the information which is required to detect this condition during characterisation. For instance, prior to using SIMIC we relied on characterising the standard cell library using one input slew rate for the HSPICE simulation, since in Verilog the ES2 cells are represented as simple module path delays (gate produces output change a delay after the input event causing the change arrives). The delay is composed of two parts:

$$t_d = t_{int} + k \times C_L$$

where:

t_{int} = internal propagation delay,
C_L = Load Capacitance and
k = constant scaling factor in nSec/pF.

However, since a symbolic table of results with input edge speed, stack size and capacitive loading as variables, is built for the process, many of the inaccuracies introduced during conversion from the continuous to the discrete domain are removed in SIMIC. As a consequence, the race condition in the ring counter is detected with an input slew rate of 50.1 nS.

In bounded-delay designs such as micropipelines, the modelling of delays can be critical to the performance of the circuit. A circuit-level simulator such as HSPICE, can be used to determine all match paths; however, the design time must be compromised as a consequence. Using SIMIC, a similar amount of confidence can be gained in a gate-level simulation environment. For instance, consider

the latches used in the SIPO (Figure 13). To store data, each latch is first pulsed transparent momentarily, then returned to opaque. To achieve this a delay is inserted before each XOR gate of the SIPO, so that a request from its associated ring counter stage causes data to be latched. A SIPO with this implementation removes all redundant transitions on data outputs, since the stage only samples the input data when required.

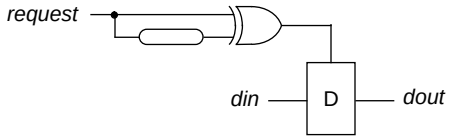


FIGURE 13 SIPO latch

This particular design has been used in the design of an asynchronous interface to two serial synchronous buses, namely the I^2C-Bus and I^2S-Bus [Farn94]. The objective of the I^2C-Bus I/O Expander design was to determine how effective micropipeline design techniques are for low power consumption in comparison to synchronous design. Therefore a synchronous device was also developed. XPOWER enabled us to compare the power consumption of the device in all modes of operation at great accuracy. Furthermore, since signals, subcircuits and test vectors can be removed interactively in an X window environment, the actual source of the power consumption was determined immediately. For instance, in the synchronous design the power attributed to the clock could simply be evaluated by removing the clocking signals which ended up consuming 60% of the entire power consumption in this design. In the micropipeline design the power consumption takes a slightly different form since the control is localised. Therefore the effectiveness of a subcircuit in a given simulation was evaluated by either removing the subcircuit and associated outputs or by only considering the subcircuit and its associated outputs. From these simulations the control component of the device appeared to use a high proportion of the power consumption. For example a node on an input to a select block with 0.5% of the simulations transitions (also available interactively) consumes 1.5% of the power consumption. As a consequence, we have re-evaluated our design strategy with respect to standard cells. In this design example the exclusion of signals from the simulation was particularly important since the I^2C-Bus consists of two bidirectional wire-ANDed signals. All transitions not caused by the device could therefore be suppressed. The resulting simulation task was therefore simplified since it could be achieved within a tailored system environment. In addition, access is given to all the system nodes (produced by SIMIC at simulation run-time), allowing the nodal capacitances to be altered. In the design of the I^2C-Bus I/O Expander this allowed the application of extra capacitance to its external outputs so as to determine its effectiveness in a system. The same test vectors were simply run again so a direct comparison could be made. As a consequence of the resulting power simulation, a design inefficiency was discovered in the micropipeline solution, which consumed over half of the power consumption in the device.

At Manchester, we believe there is a potential for synthesis methods that can be effectively applied to high performance architectures and have been developing such a system based on VLSI programming from Philips Research. [vBer88]

The first generation of tools from PRL were used to build dual rail devices on a custom “standard” cell library from a CSP based specification language, Tangram. The second generation tools are designed to make use of conventional standard cells and therefore overcome the problems posed by portability and foundry migration.

VLSI programming allows the designer considerable feedback on the chosen specification in regard to area, timing and power and allows fast re-iteration of the design to eliminate errors and improve performance. Thus, the goals at Manchester are to:

- Encompass the VLSI programming discipline
- Extend the scope of the system to allow the addition of custom components
- Maintain the portability and migration advantages of standard cell design.

To this end, a complete design system is being integrated into a commercial CAD environment, namely Cadence Design Systems Framework II. The first stage involves translating the output from the Tangram compiler, known as Handshake Circuit Language into a Verilog description so that the resulting description can then be used for two purposes:

(1) Stand alone behavioral simulation. This allows the designer to confirm the functional behavior of his program at an early stage and get very coarse timing and power estimates.

(2) Integration into the Cadence Design Framework.

With the traditional approach to cell development, a library of fixed cells is created, but if cell variations are needed, redesign must take place. Variations might include changing the number of inputs, cell function, design rules, or drive capability. Redesigning always involves more than just moving a polygon; it means re-simulating, re-characterizing, rebuilding simulation models, and re-documenting as necessary.

With the module generator approach, the generator developer puts the design effort into creating a module generator that is a parameterized design. The generator is

then delivered to an end user, who can use the generator to produce many variations of a cell.

From one generator and one set of input parameters, all cell library types can be generated including the schematic symbol, schematic, simulation models, logic synthesis models, symbolic layout, polygon layout, place and route constraint information, etc.

The automatic generation of all these views ensures consistency and simplifies cell library management. In order to maintain portability, the custom cells are built using symbolic layout and fed to a compactor to enable DRC correct cells to be constructed with ease. The designers merely specify the topology of the cell rather than concerning themselves with geometrical constraints which thus eliminates low level errors, often a major problem when dealing with custom circuits.

With existing simulators, it is often very difficult to use layout generators since the re-characterisation task is typically very time consuming and the end user is not granted the flexibility to tune his circuit with respect to performance in terms of speed, power or area.

The GENDEL/SP2LOG path facilitates this approach, since process characterisation is now only CPU and not labour intensive. The HSPICE interface has also been extended to allow automatic generation of SP2LOG models from HSPICE verified schematics and extracted layouts.

Dynamic resizing of transistors in a design is often useful to avoid the need to insert buffers in critical parts of the circuit. SIMIC information can be used to identify slow edges as well as circuits that may be consuming too much power. This information can then be used to resize the cell or block, which will then automatically be re-compacted and re-characterised to allow the circuit to be re-simulated very quickly.

A complete graphical interface has also been developed using the Open Simulation system. The netlister module produces hierarchical netlist descriptions from both Composer and Virtuoso. STL can be used to generate the stimulus files transparently and full color cross probing is available with Genashor's waveform analysis and display program, XWAVE. Both XWAVE and XPOWER support auto update features which allows the new simulation data to be automatically updated at high speed. The Genashor tools were easy to incorporate in this framework (a strength of both Framework and Genashor products) and allows us to utilize all of the available features in a fully integrated environment.

8. Future Work

We are continuing to enhance SP2LOG/SIMIC modeling capabilities, and are exploring a Monte Carlo algorithm. A prototype version of SIMIC supports pin to pin delay specification for each element. This will further reduce differences in delays reported by SIMIC and SPICE. The prototype fully supports the hazard analysis of the original version. In addition, the computed delays remain accurate even when multiple inputs change within the gate's propagation interval. No other known gate-level simulator has demonstrated this ability.

We are also investigating the use of a methodology, similar to the one described here for delay modeling, to characterize power consumption, due to shorting currents and other power consumption phenomena currently not addressed by XPOWER, with accuracy close to SPICE.

9. Conclusions

This paper has described several tools that have proven to be very effective for debugging, modeling, and analyzing the operation of asynchronous designs. The tools run stand-alone interactively or in batch, and are easily integrated in a custom design environment. The SIMIC logic simulator performs numerous timing and hazard checks, often enabling the designer to locate potential timing problems even when the simulation results appear to be correct. The SP2LOG gate-level extractor generates simulation models that accurately reflect the physical implementation. Both programs utilize piecewise-linear characterization of output delay as a function of output loading and input slew rate that achieve agreement with SPICE to within 20%. Generation of these delay tables from SPICE simulations of P and N transistor stacks is completely automated with GENDEL. The XPOWER program has proven to be extremely useful for analyzing dynamic power consumption in asynchronous designs.

10. References

[Farn94] Farnsworth C., "Low Power Implementation of an I2C I/O Expander" Master's Thesis.

[Suth89] Sutherland I.E., "Micropipelines", Communications of the ACM, p 720-738, 1989.

[Ung69] Unger, S. H. "Asynchronous Sequential Switching Circuits", Wiley Interscience, 1969, page 143

[vBer88] van Berkel C.H., Rem M., Saeijs R.W.J.J., "VLSI Programming", Proceedings of ICCD, IEEE, p 152-156, 1988.

[Yant92] Yantchev J., An S-I Toggle design. Private Fax communications, 1992.