26

VHDL Power Simulator: Power Analysis at Gate-Level

L. Kruse, D. Rabe, W. Nebel
FB 10 - Department of Computer Science
OFFIS and University of Oldenburg
D - 26111 Oldenburg, Germany
Tel.: 0049 - 441 - 798 2154
Fax: 0049 - 441 - 798 2145
E-Mail: Kruse@OFFIS.Uni-Oldenburg.DE

Abstract

Power consumption of integrated circuits becomes more and more an important issue in the design phase. In this paper a new application of VHDL for gate-level power analysis and accurate timing verification is presented. Our VHDL Power Simulator (VPS) is able to accurately estimate the mean power consumption of a static CMOS standard cell design described in VHDL at gate-level. Additionally VPS increases the timing accuracy of logic level simulation in case of glitches, defined here as pairs of incomplete transitions. This is achieved by modifying the VHDL event handling and propagating ramps instead of infinite slope events. Our tool, implemented as an add-on to Cadence's Leapfrog VHDL simulator, allows to employ a Monte Carlo simulation for mean power consumption analysis while taking rise/fall times and glitches into account. The VHDL descriptions of the cells have to be VITAL Level 1 compliant and only slight modifications of these descriptions have to be done. The library adaptation can be automatically performed. First simulation results show that the accuracy of VPS is within 10% of circuit-level simulation even in case of circuits with high glitch activity.

Keywords

VHDL, Power Estimation, Timing

1 INTRODUCTION

In the last five to ten years power consumption has become another dimension in the design space of integrated circuits besides area, throughput and testability (Chandrakasan, 1992; Cole, 1993). The designer has to make design decisions concerning power consumption and has to check if he meets given power constraints. For example, the consumer market demands for portable applications with a long time of battery operation. Therefore integrated circuits within these applications should manage with a low power budget (Manners, 1991). Packaging cost is another reason for low power design. Chips which consume less than about two watts can be mounted in cheap plastic packages without any heat problems. Otherwise expensive ceramic packages, possibly with additional cooling devices, must be used (Pivin, 1994).

Power analysis tools allow the designer to check implementations against power constraints and guide the design process for low power as a cost function. Since power consumption is not a static property of a circuit but strongly depends on the application input patterns a simulation for power analysis has to be done. An analog simulator like Spice is able to calculate the power consumption at circuit-level with high accuracy. But this kind of simulators cannot handle practical size circuits and a sufficient number of patterns because of memory and computational complexity (Nagel, 1975).

Additionally in deep submicron electronics the interconnect RC-delay plays a dominating role in timing behaviour. Furthermore intrinsic delays need to be modelled input slope dependent. Hence for the timing analysis of circuits, refined timing models are needed. As we will show in this contribution, the delay of gate responses to incomplete transitions depends on the voltage peak of the glitch which itself is a function of the input skew, slope and output load. The model and the simulator presented here take the delay impact on incomplete transitions into account.

A number of proposals for glitch modelling at the gate-level have been made (Eisele, 1995; Melcher, 1991; Metra, 1995). A comparison of our model (Rabe, 1996c) with these models show that our model is more accurate with respect to glitch peak voltage prediction and timing (Rabe, 1996b).

We present a power analysis tool, the VHDL Power Simulator (VPS), which enables the designer to estimate the power consumption of static CMOS designs at gate-level. VPS uses the two-step paradigm of a single library characterization and logic simulation. The tool can be easily used within a VHDL based design flow because VPS uses a VITAL (VHDL Initiative Towards ASIC Libraries) Level 1 compliant (Schulz, 1994) VHDL description as input which is normally obtained after logic synthesis. The VITAL standard is a modelling guideline of standard cell descriptions in VHDL. The standard facilitates fast library development and enables VHDL simulators to accelerate the simulation of the cell descriptions. VPS, implemented as an add-on to Cadence's Leapfrog VHDL simulator, is based on a Monte Carlo approach (Burch, 1993) and takes slopes and glitches into account. A glitch is defined as a set of pairs of incomplete signal transitions. Glitches are generated by colliding events caused by at least two input transitions as depicted in Figure 1 (Rabe, 1996b; Rabe, 1996c). In most cases glitches cause less power consumption than their corresponding complete transitions and should therefore be handled differently (Favalli, 1995).

Simulations of a 4-bit ripple adder with 256 input patterns (cf. Figure 2) indicate that in this case 14.9% of the total number of transitions are glitching transitions. Some gate outputs and their glitching behaviour are listed in Table 1. Note that this behaviour depends on the set of

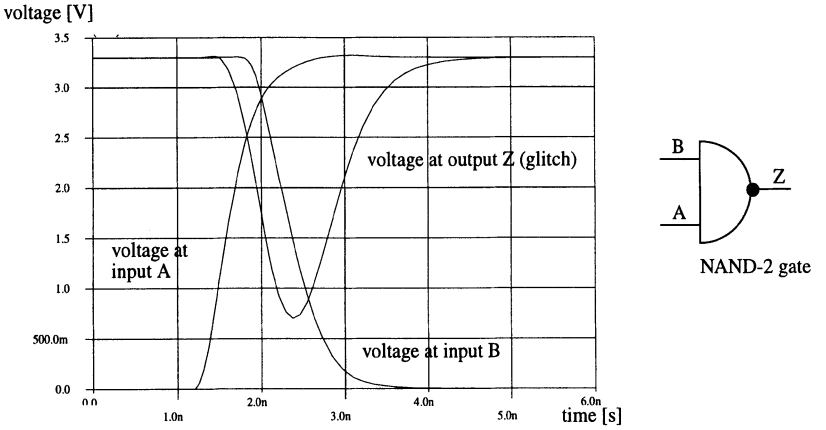

Figure 1 Two colliding output transitions at output Z forming a glitch.

input patterns as well as on the architecture of the design.

This paper is organized as follows. Section 2 briefly describes the models used by VPS. Section 3 gives some details about the implementation of the tool. Some simulation results are presented in section 4. Finally in section 5 conclusions are drawn and an outline of our future work concerning VPS is given.

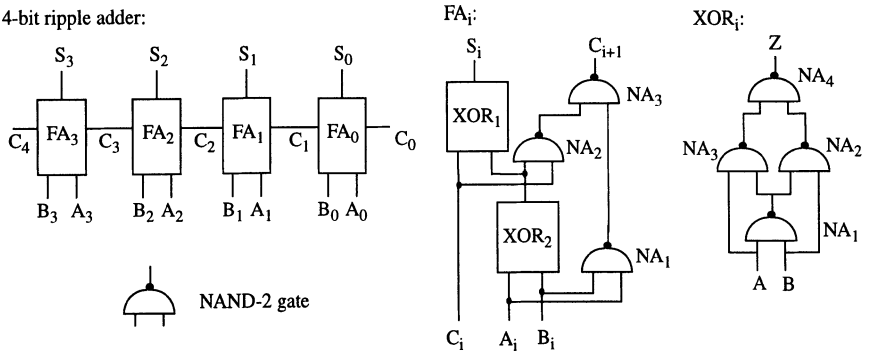

Figure 2 4-bit ripple adder.

Table 1 Occurrence of glitches within a ripple adder

output of gate:	total number of transitions	total number of glitching transitions	% glitching transitions of total transitions
:FA3:NA3	198	22	11.1%
:FA3:XOR2:NA3	197	76	38.6%
:FA3:XOR1:NA3:Z	165	0	0.0%
:FA3:XOR2:NA2:Z	225	14	6.2%

2 POWER MODELLING

In the next two subsections the power models for complete and incomplete transitions respectively used by VPS are described.

2.1 Complete Transitions

The power dissipation of a circuit can be calculated by summing up the power consumed by each gate. The power consumption of a gate within a static CMOS design can be divided into three parts:

$$P_{Gate} = P_{Cap} + P_{Short-circuit} + P_{Leakage} \tag{1}$$

The third component $P_{Leakage}$ is due to static leakage currents and in typical designs very small so that it can be neglected. In static CMOS the capacitive and the short-circuit power are consumed only in case of signal activity caused by a gate output. Hence in our model the power consumption is derived from gate output activity and fan-out information.

$$P_{Short-circuit} = \overline{I_{Short-circuit}} \cdot V_{DD} \tag{2}$$

is due to short circuit currents from the V_{DD} to the ground pin while both a N-MOS as well as a P-MOS block of the gate are conducting.

The capacitive power without glitches

$$P_{Cap} = N(T) \cdot \frac{1}{2 \cdot T} \cdot C_L \cdot V_{DD}^2, \tag{3}$$

where $N(T)$ is the number of complete transitions in the time interval T to be considered, is consumed due to charging resp. discharging gate output, wire and connected input capacitances C_L of the fan-out of the switching gate output. P_{Cap} depends on the output load and the number of output transitions while $P_{Short\text{-}circuit}$ in addition depends on the rise resp. fall time (*rftime*) of the causing input slope. The *rftime* of a transition is defined by the time of the slope

between crossing 10% and 90% V_{DD}. Figure 2 shows the charge Q_{VDD} consumed by a NAND-2 gate due to a rising input transition at input A over the rise time of the input slope.

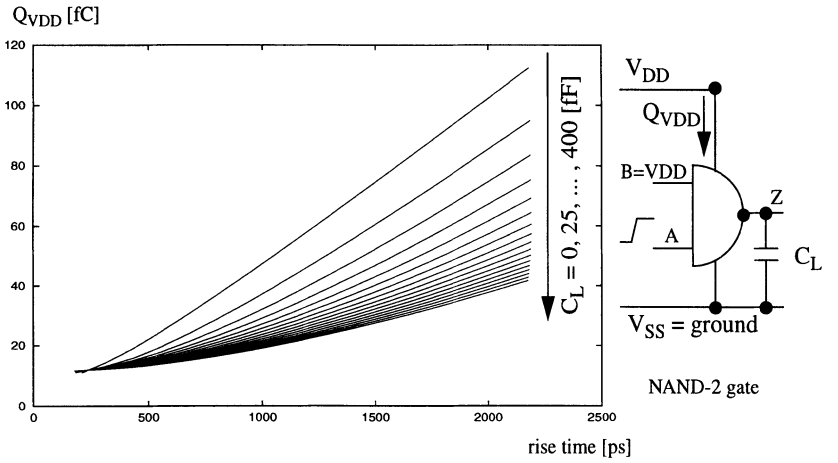

Figure 3 Dependency of the power consumption on the input rise (fall) times.

Within VPS the power consumption P of a gate G because of a transition T_O on output O caused by an input slope T_I on input I is modelled as a function of the output load C_L, the *rftime* of T_I and the direction of the slopes:

$$P = \frac{V_{DD}}{T} \cdot p_{G,O,I}(C_L, rftime(T_I)) \tag{4}$$

In the implementation p returns the charge Q_{VDD} drawn from the power supply so that the capacitive as well as the short-circuit portion is included. The total power consumption is then given by $P = \frac{\sum_i^N Q_{VDD}(i)}{T} \cdot V_{DD}$, where N is the number of transitions within time interval T and $Q_{VDD}(i)$ is the power supply's contribution due to transition i. The *rftime* of the output slope is a function of the same parameters as p:

$$rftime(T_O) = f_{G,O,I}(C_L, rftime(T_I)) \tag{5}$$

p and f are determined during library characterization. As described in section 3 VPS is able to associate each event in a VHDL simulation with the *rftime* of the corresponding transition so that p and f can be dynamically evaluated. This allows to model voltage waveforms by ramps with the same *rftimes*. VPS allows the user to specify voltage values on the input and the output slopes from which the delay is measured, for example 40% - 60% V_{DD} (rise to fall delay) as depicted in Figure 4. Apart from that VPS uses the pin-to-pin delay model as defined in the VITAL standard.

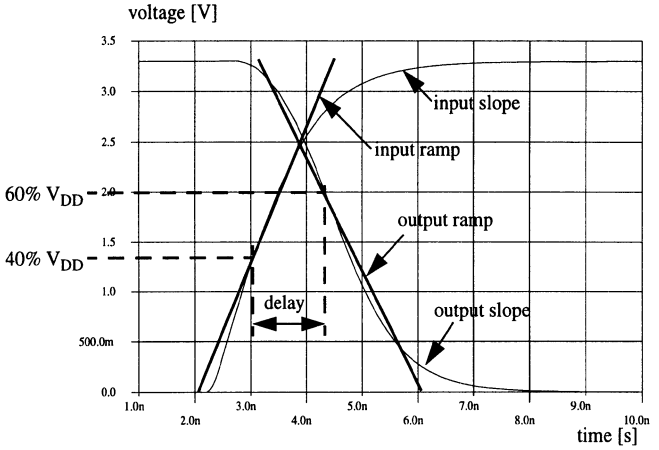

Figure 4 Delay definition.

2.2 Incomplete transitions (glitches)

If a glitch occurs at an output of a gate its fan-out capacitances are not completely charged and discharged. For this reason, equation (3) has to be generalized for glitches:

$$P_{Cap} = \frac{1}{2 \cdot T} \cdot C_L \cdot V_{DD} \cdot \sum_{j=1}^{M(T)} \Delta V_j, \tag{6}$$

where $M(T)$ is the number of glitching transitions in the time interval T and ΔV_j is the peak voltage (measured from the initial voltage value) of the glitch transition j is involved in.

VPS models a glitching output slope T_j by its corresponding complete transition T_i. The power P_{T_j} consumed due to T_j is calculated by

$$P_{T_j} = \frac{\Delta V}{V_{DD}} \cdot P_{T_i}, \tag{7}$$

where P_{T_i} is the power that would be dissipated if only the complete transition T_i took place and ΔV is the peak voltage of the glitch. Equation (7) models exactly the capacitive power, however, the estimation of the short-circuit part is only moderate (Rabe, 1996a). The peak voltage ΔV is estimated as described below.

Besides the glitch power consumption VPS takes into account the timing behaviour of glitching slopes. Figure 5 shows a glitch at a gate's output and the corresponding complete setting resp. resetting single slopes if the other one doesn't occur. The setting slope is the one which drives the output-waveform away from the initial state while the resetting slope is the one with the opposite direction. Because the output capacitances aren't completely discharged the delay of the resetting glitching transition is smaller compared to the delay of the single resetting one.

Ignoring this decrease in delay can result in large errors while calculating the power con-

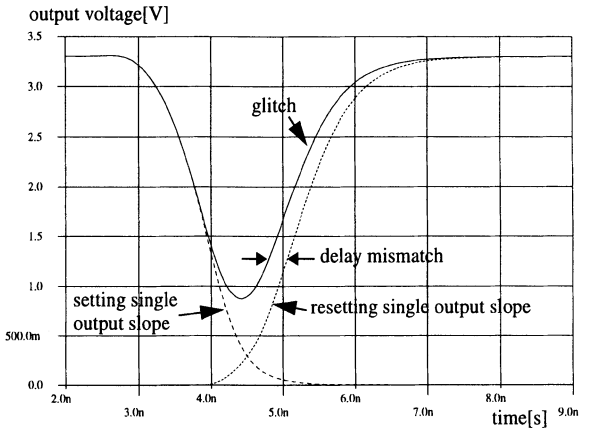

Figure 5 timing behaviour of a glitch.

sumption of propagated glitches through successive gates as well as in the timing behaviour of the fan-out cone of the glitching output (Rabe, 1996b). VPS models glitches with their underlying complete ramps and shifts the resetting ramp such that the point of intersection of both ramps is at the glitch peak time t_g as depicted in Figure 6.

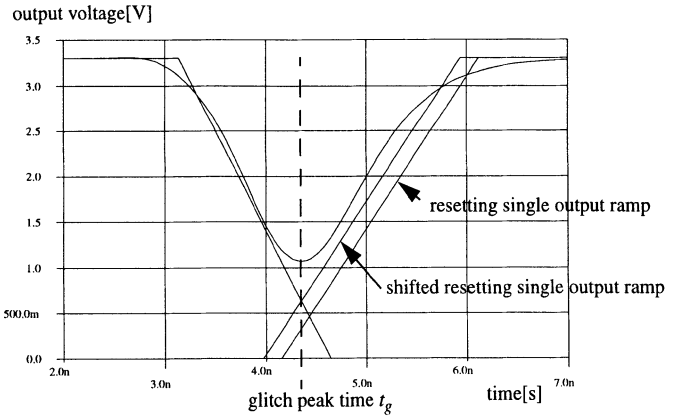

Figure 6 Shift of resetting output slope.

In order to enable VPS to estimate the glitch peak voltage ΔV and the glitch peak time t_g only four voltage values for each input pin to output pin combination of a gate have to be determined during characterization of the standard cell library. Figure 7 shows the determination of ΔV and t_g for a glitch with a rising resetting output slope. Within the model the peak voltage ΔV equals the voltage value of the setting output ramp at the instant when the resetting input ramp

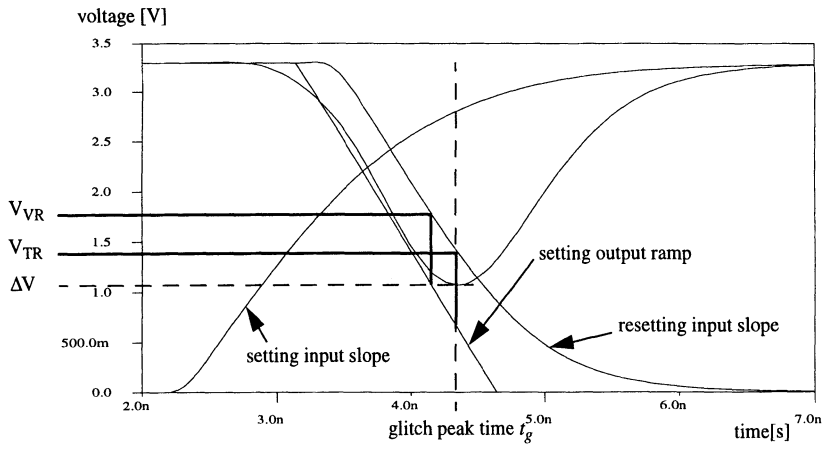

Figure 7 Determination of the peak voltage and the peak time.

crosses V_{VR} (peak Voltage, Rising output transition). t_g equals the time point when the resetting input ramp crosses V_{TR} (peak Time, Rising output transition). The voltage values for a falling resetting output transition are called V_{VF} and V_{TF}.

The proposed model is only applicable to a single CMOS stage. Complex gates have to be subdivided into their stages. A detailed description of the glitch model and the physical background is given in (Rabe, 1996b; Rabe, 1996c).

3 IMPLEMENTATION

The Leapfrog VHDL simulator from Cadence Design Systems possesses a programming interface that allows the user to replace a VHDL function or procedure by a C function and to access parts of the data structure of a VHDL model. A lot of C functions are defined which can be called from user functions, for example to generate an event on a VHDL signal. The C code of the user can be statically or dynamically linked to the simulator. If a C function replaces a VHDL procedure the function will have the same arguments as the VHDL procedure and an additional argument that points to the scope in which the procedure is defined.

3.1 VITAL library modelling

The application of VPS is only feasible on structural VHDL descriptions that use cells whose architectures are VITAL Level 1 compliant and restricted to the commonly used pin-to-pin delay model in combination with the wire delay model (Schulz, 1994). In Figure 8 a graphical representation of a VHDL description of a cell is shown. For each input port a concurrent procedure call to `VitalPropagateWireDelay()` is done in order to model wire delays on the net connected to the input. The incoming delayed events are scheduled on the internal sig-

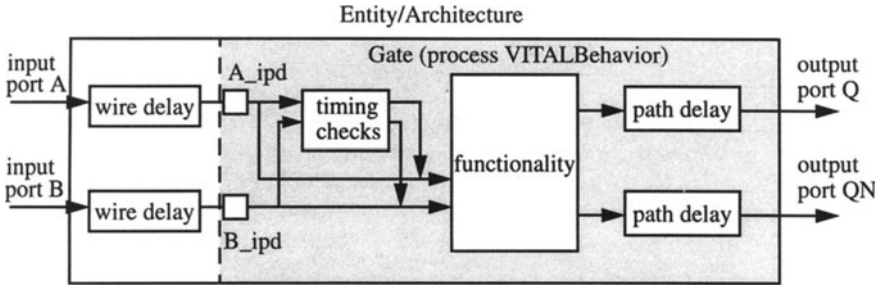

Figure 8 Gate description according to VITAL Level 1, pin-to-pin delay model.

nals with suffix `_ipd`. After optionally performing some timing checks, for example checking hold time violations within flip-flops, the functionality of the cell is calculated. If new events have to be generated on an output port, the procedure `VitalPropagatePathDelay()` performs VITAL glitch handling, for example by generating an `'X'` if a glitch is detected, and propagates the new output value on the port with the corresponding pin-to-pin delay. The delay values are initially read in from a Standard Delay File (SDF). The timing checks, the calculation of the functionality and the calls to `VitalPropagatePathDelay()` are written within one VHDL process with the consequence that each time an event arrives on an internal signal the procedure `VitalPropagatePathDelay()` is called for each output port.

Because VPS has to propagate ramps and has to apply the proposed glitch model, we have replaced the VHDL procedures of the wire and the path delay part by our own C functions using the programming interface of Leapfrog. The main library modifications to be done are restricted to the VHDL descriptions of the cells. The attribute `VITAL_LEVEL_1` of the architectures of the cells must be set to `false` and the attributes `Foreign` of the procedures `VitalPropagateWireDelay()` and `VitalPropagatePathDelay()` have to be set to the name of their C-functions. This can be done automatically by a script and the new cell descriptions should preferably be stored in another library. In order to use the modified library the relevant library clauses within configurations have to be set to the new library.

3.2 Modified event handling

In order to correctly propagate and generate incomplete signal transitions in VHDL, the VHDL event handling needs to be modified. In particular the VHDL events with zero rise and fall time have to be replaced by signal ramps. In consequence in VPS the event queues are substituted by a data structure to store the rise/fall times of the ramps. Each net of the design has a list of ramps in the data structure. The ramps are sorted by their starting points. A VHDL simulation starts with the activation of all processes so that for each output the C function `path()`, which replaces `VitalPropagatePathDelay()`, is called. In this phase the data structure is formed and a callback function is installed for each primary input. The callback function is automatically called if an event is generated on the primary inputs by the test bench and then produces ramps for these events. The user sets the *rftimes* of the primary input ramps. These

rftimes and other parameters must be specified within an option file of VPS. At the end of this phase wire capacitances are read in and stored in the data structure. The input capacitances of the gates have to be determined during characterization and are made available to the tool through a characterization file which is also read in at this time.

Figure 9 depicts the timing relationship between VHDL events and their ramps. The events of complete transitions and of setting transitions in case of glitches are generated when the ramp starts. Events of resetting transitions are propagated at the glitch peak time because this is the time the resetting ramp describes the output voltage waveform.

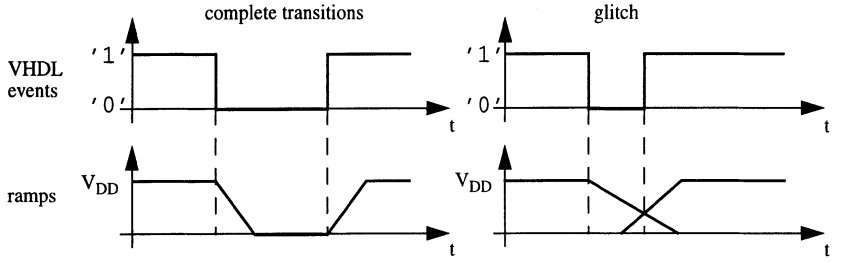

Figure 9 Timing relationship between events and ramps.

The main algorithm of the `path()` function to propagate ramps can be summarized as follows:

```
(1)  if ( output O changes its value )
(2)     determine the input transition T_I which caused the value
        change;
(3)     get the delay of the new event;
(4)     calculate p_{G,O,I}(C_L, rftime(T_I)) and f_{G,O,I}(C_L, rftime(T_I)) of the
        new ramp;
(5)     create output ramp T_O;
(6)     schedule(O, T_O, T_I);
(7)     generate an event on the output port;
(8)  end if;
```

The first two steps (lines (2) and (3)) are implemented in such a way that `path()` behaves like `VitalPropagatePathDelay()`, for example if two ramps arrive on different inputs at the same time the one with the smallest delay to the output will be the causing input ramp. p and f are implemented as look-up tables.

The function `schedule()` updates the list of ramps of the output net and does the glitch handling. The main part of `schedule()` is shown next:

```
(1)  schedule(O, T_O, T_I)
(2)     delete ramps which end in the past from the list of the
        net connected to output O;
(3)     if ( list of ramps is empty )
(4)         insert T_O into the list;
(5)     else
(6)         determine the ramp T_P that will be the predecessor
            of T_O in the list;
(7)         delete all ramps after T_P; /* like the transport
            delay model */
(8)         if ( T_O and T_P have opposite directions )
(9)             if ( T_I crosses V_V(R/F) during T_P )
(10)                /* glitch detected */
(11)                determine the glitch peak time using V_T(R/F);
(12)                reduce the delay of T_O as described in
                    section 2;
(13)                insert T_O after T_P into the list;
(14)            elseif ( T_I crosses V_V(R/F) before T_P begins )
(15)                /* glitch does not propagate */
(16)                delete T_P;
(17)            else
(18)                /* no glitch, two complete transitions */
(19)                insert T_O after T_P into the list;
(20)            end if;
(21)        end if;
(22)    end if;
(23) end schedule;
```

A glitch is detected if the causing input ramp T_I crosses $V_{V(R/F)}$ between the beginning and the end of the predecessor output ramp T_P (line 9), where $V_{V(R/F)}$ means V_{VR} if T_P is a rising transition and V_{VF} if T_P is a falling one. In this case the power consumed because of T_P and T_O is adapted according to equation (7). If T_I crosses $V_{V(R/F)}$ before T_P begins, that means the peak voltage would equal the initial voltage of T_P, a potential glitch is filtered out and neither T_O nor T_P is propagated. The algorithm is even able to handle multiple glitches with more than two colliding ramps.

VPS sums up the charges Q_k due to output transitions T_k and periodically calculates the mean current $I(t_l)$ drawn from the power supply by the circuit at simulation times t_l:

$$I(t_l) = \frac{\sum_k Q_k}{t_l} \tag{8}$$

If the deviation of the mean current $I(t_j)$ from the last value $I(t_{j-1})$ is lower than ε N times in a

row, i.e. $\left|\frac{I(t_{l-1}) - I(t_l)}{I(t_{l-1})}\right| < \varepsilon$, $I(t_{l-1}) \neq 0$, it is assumed that the mean current has converged and equals the mean current consumption of the circuit. The mean power consumption is then given by $P = I(t_l) \cdot V_{DD}$. N and ε are under user control. If the stopping criterion above is fulfilled VPS is able to propagate a `'1'` on a signal used as a flag. The flag can be tested, for example, within the test bench by a concurrent assertion statement which stops the simulation:

```
architecture behaviour of testbench is
...
signal stop : bit := '0';

begin
    Assert stop = '0'
        Report "Current has converged!"
        Severity failure;
    ... -- component instantiation
    ... -- process which generates the stimuli
end behaviour;
```

The name of the flag (`stop` in the example above) has to be defined by the user in the option file. The test bench has to produce stimulis which are equal or similar (similar means with the same statistical properties (Radetzki, 1996)) to stimulis generated by an application of the circuit in order to guarantee that the estimated power consumption is realistic.

4 RESULTS

4.1 Glitches

First of all we want to illustrate the quality of the glitch model used by VPS. Figure 10 depicts a small benchmark-circuit with a V_{DD} of 3.3 volts which has been used to generate and propagate glitches. At the inputs of the first NAND-2 gate different glitches can be initiated on output Z_1 by varying the skew between the rising slope on input A and the falling one on input B. Glitches may propagate through the second NAND-2 gate to Z_2. The *rftimes* of the input slopes and the capacitors C_{L1} and C_{L2}, which are inserted to model wire capacitances, have typical values. The pin-to-pin delays are read from a SDF-file (Standard Delay File). This file is usually generated using a separate delay calculator. In our case, however, due to the unavailability of a delay calculator, we used Spice simulations of single slopes to generate the SDF-file. VHDL as well as Spice simulations have been performed and compared in Table 2. For different skews the estimation of the peak voltage ΔV and the glitch peak time t is reported. Index V denotes the VHDL simulation while index S designates the results of the Spice simulation. $\Delta_V = |\Delta V_V - \Delta V_S|$ is the absolute error of the peak voltage estimation and its mean value is below 5% V_{DD} for glitch generation and about 6% V_{DD} for glitch propagation. Because the

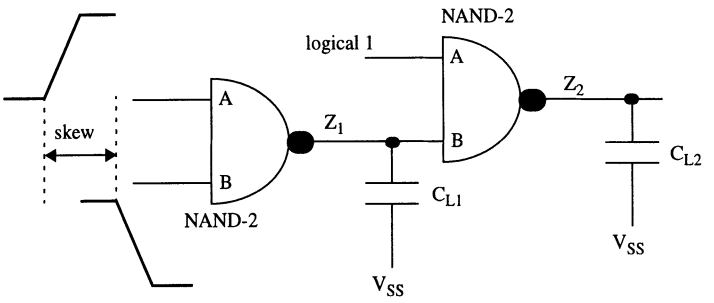

Figure 10 Benchmark-circuit for glitch generation and propagation.

calculation of the glitch peak times returns absolute values an error $\Delta_T = |t_V - t_S|$ can only be reported if both types of simulation have encountered a glitch. VPS isn't able to predict glitches at a skew of zero ps or below that value because in these cases no event will be generated on the output port.

Table 2 Peak voltage and glitch peak time computation

skew [ps]	peak voltage [V]						glitch peak time [ns]					
	generation (Z_1)			propagation (Z_2)			generation (Z_1)			propagation (Z_2)		
	ΔV_V	ΔV_S	Δ_V	ΔV_V	ΔV_S	Δ_V	t_V	t_S	Δ_t	t_V	t_S	Δ_t
0	0,00	0,16	0,16	0,00	0,00	0,00	-	2,37	-	-	-	-
200	0,33	0,41	0,08	0,00	0,00	0,00	2,56	2,53	0,03	-	-	-
400	0,92	0,82	0,10	0,00	0,00	0,00	2,76	2,67	0,09	-	-	-
600	1,52	1,42	0,10	0,00	0,19	0,19	2,96	2,84	0,12	-	3,01	-
800	2,11	2,07	0,04	1,12	0,67	0,45	3,16	3,02	0,14	3,49	3,26	0,23
1000	2,71	2,60	0,11	2,38	2,13	0,25	3,32	3,20	0,12	3,89	3,61	0,28
1200	3,30	2,89	0,41	3,00	2,89	0,11	-	3,35	-	4,09	3,85	0,24
1400	3,30	3,10	0,20	3,30	3,14	0,16	-	3,51	-	-	4,00	-
1600	3,30	3,19	0,11	3,30	3,24	0,06	-	3,66	-	-	4,20	-
mean value:			0,15			0,20			0,10			0,25

4.2 Mean power consumption

Next we want to show first simulation results concerning the power consumption of a circuit. Instead of presenting power values we report the charge flown from the V_{DD} to the ground pin. The power consumption can be calculated by $P = \frac{Q}{T} \cdot V_{DD}$. The benchmark-circuit we used is a 4-bit ripple adder (cf. Figure 2) whose four full adders consist of eleven NAND-2 gates with a maximum logic depth of six. The ripple adder is simulated with different input data streams (low and high activity) and wire capacitances which influence the glitching behaviour of the circuit. Table 3 shows the results. VPS has a switch that allows to turn off the glitch model, i.e. all transitions are complete ramps. The forth column depicts the errors of VPS with glitch handling (third col.) compared to Spice (second column) while the last column shows the errors of VPS with glitch handling (fifth column) switched off. The results show that the error of VPS is below 10% except for one case with an error of 10.35%. If glitches are ignored the error is in most cases larger than 10%.

Table 3 Flown charge computation of a 4-bit ripple adder

input data stream	load cap. [fF]	Charge [pC]				
		Spice	VPS w/ glitches	error	VPS w/o glitches	error
low activity	40-100	640.33	647.50	1.11%	766.08	16.41%
	140-200	1169	1271.17	8.04%	1487.30	21.40%
	240-300	1709	1906.22	10.35%	2227.97	23.29%
high activity	40-100	1016	972.94	-4.24%	1135.92	10.56%
	140-200	1877	1970.02	4.72%	2261.02	16.98%
	240-300	2754	2973.32	7.38%	3404.30	19.10%

VPS is also able to output the ramps and the mean current curve for a graphical representation. Figure 11 depicts the mean current consumption of the ripple adder with a high activity data stream and 140 - 200fF wire capacitances.

In Figure 12 some ramps and their corresponding Spice-waveforms are shown. It can be seen that all six glitches on *sum(1)* and *sum(2)* in the depicted time interval are detected by VPS.

VPS can give some statistical information like number of glitches, number of useless transitions or mean current consumption, all per net or hierarchy-level. This information can be used to detect parts of the design with high power consumption which are worth being optimized.

The actual implementation of VPS is about 30 times slower compared to a normal VHDL simulation but still two to three orders of magnitude faster than Spice. Because of the fact that

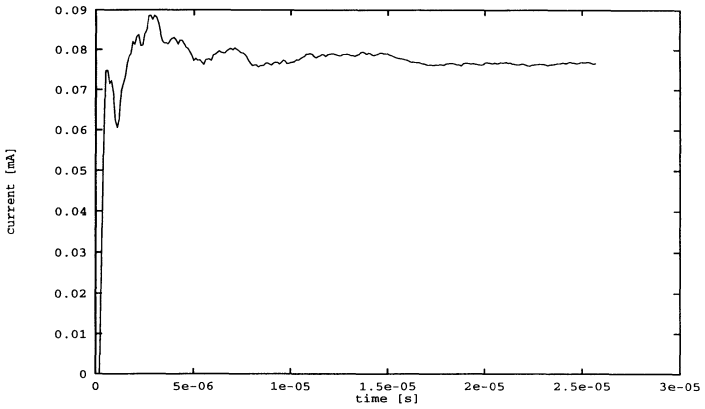

Figure 11 Graphical presentation of a mean current curve.

some VITAL procedures are replaced VPS has to sacrifice the VITAL acceleration. However, time measurements have shown that the main bottleneck which reduces the performance is the programming interface of Leapfrog.

5 CONCLUSION AND FUTURE PLANS

We presented our VHDL Power Simulator which is capable of analysing the mean power consumption of a design described in VHDL at gate-level. The simulator takes rise/fall times into account and accurately handles glitches. Simulations of small benchmark-circuits show that the results of VPS are within about 10% of Spice and that glitches cannot always be neglected. Because of its glitch model VPS can also be used for accurate timing verification.

A drawback of VPS is that delays are read in from an SDF-file and therefore a dependency on the input slopes rise/fall times isn't modelled. We are going to extend the characterization-step of the cell library in order to model the delay as a function of the rise/fall times of the input ramps and the output load. The delay function will be implemented as a look-up table, too. Next we will finish the characterization of our library so that we will be able to simulate circuits from well known benchmark suites.

The Open Verilog International (OVI) organization plans to standardize a file format of characterization data like rise/fall times, delays and power values (OVI, 1996; Cottrell, 1996). An interface to this format, which allows data exchange between different tools (simulators and characterization tools), could be easily added to VPS.

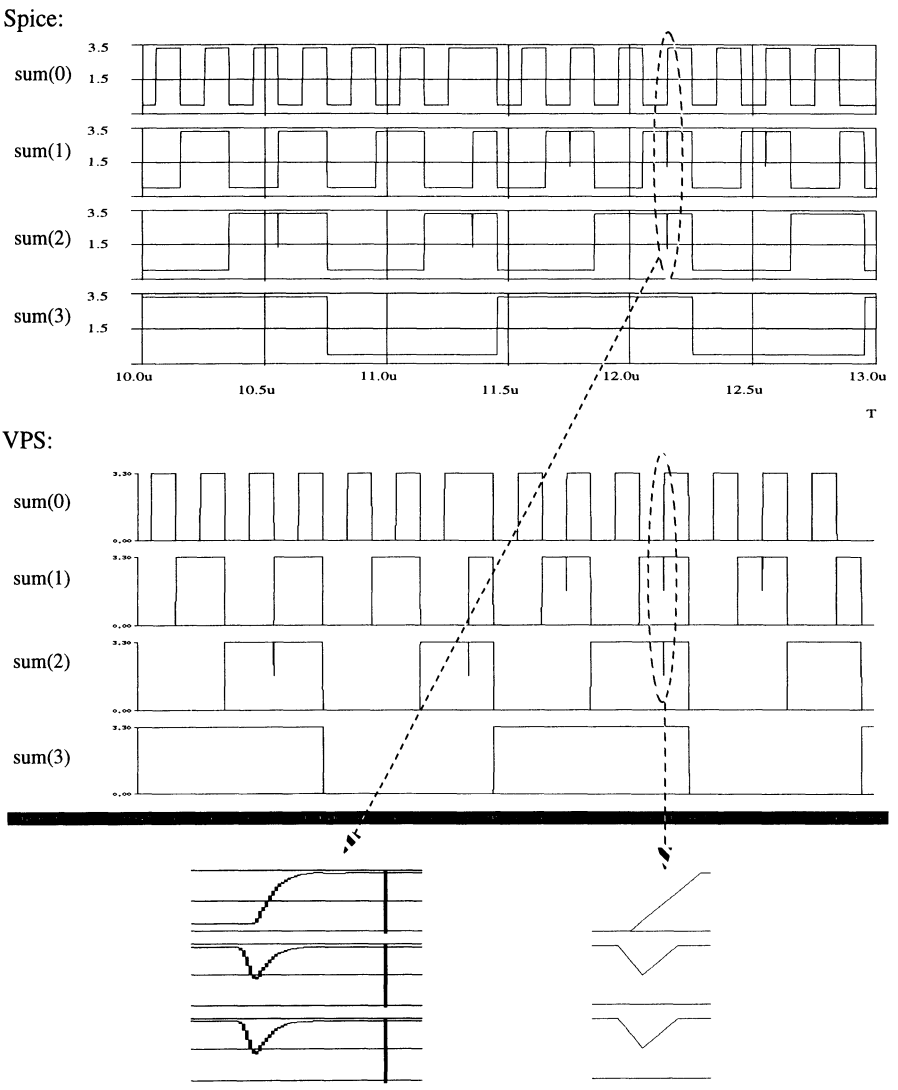

Figure 12 Ramps and their corresponding Spice-waveforms.

6 REFERENCES

Burch, R., Najm, F.N., Yang, P. and Trick, T.N. (1993) A Monte Carlo Approach for Power Estimation. *IEEE Transactions on very Large Scale Integration (VLSI) Systems*, **1**, No. 1, 63 - 71

Chandrakasan, A., Sheng, T. and Brodersen, R.W. (1992) Low Power CMOS Digital Design. *Journal of Solid State Circuits*, 473-484

Cole, B. (1993) AT CICC: Designers Face Low-Power Future. *Electronic Engineering Times*. **745**, 1-2

Eisele, M. and Berthold, J. (1995) Dynamic Gate Delay Modeling for Accurate Estimation of Glitch Power at Logic Level. *Proceedings of PATMOS*, 190 - 201

Favalli, M. and Benini, L. (1995) Analysis of Glitch Power Dissipation in CMOS IC's. *Proceedings of the International Symposium on Low Power Design*, 123 - 128

Manners, D. (1991) Portables Prompt Low-Power Chips. *Electronics Weekly*, **1574**, 22

Melcher, E., Dana, M. and Jutand, F. (1991) PATMOS: Report on Cell Assembly and Multi-Level Modelling. *Second Periodic Progress Report*, **3237**, Appendix 1, 1 - 21

Metra, C., Favalli, M. and Ricco, B. (1995) Glitch Power Dissipation Model. *Proceedings of PATMOS*, 175 - 189

Nagel, L.W. (1975) SPICE2: A Computer Program to Simulate Semiconductor Circuits. *Technical Report ERL-M520*, University of California, Berkeley

OVI ftp server (1996) , ftp.metasw.com/pub/OVI/PS-TSC (09/24/1996)

Cottrell, D., Beatty, J. and Chang, S.-S. (1996) ASIC Delay Calculation and DCL. *documentation tutorial 6*, EURO-DAC

Pivin, D. (1994) Pick the Right Package for Your Next ASIC Design. *EDN*, **39**, No. 3, 91-108

Rabe, D. and Nebel, W. (1996a) Short Circuit Power Consumption of Glitches. *Proceedings of the International Symposium on Low Power Electronic and Design*, 125 - 128

Rabe, D., Fiuczynski, B., Kruse, L., Welslau, A. and Nebel, W. (1996b) Comparison of Different Gate Level Glitch Models. *Proceedings of PATMOS* , 167 - 176

Rabe, D. and Nebel, W. (1996c) New Approach for Glitch-Modelling on Gate Level. *Proceedings of EURO-DAC*, 66 - 71

Radetzki, M., Timmermann, B., Rabe, D. and Nebel, W. (1996) Generation of Binary Patterns with Given Spatiotemporal Correlations. *Proceedings of PATMOS*, 199 - 208

Schulz, S.E. (1994) VITAL - VHDL Initiative Toward ASIC Libraries - Model Development Specification, Version 2.2b

7 BIOGRAPHY

Lars Kruse finished his master thesis in computer science at the University of Oldenburg, Germany in 1996. Since June 1996 he is working as a research assistant towards his PhD at the research institute OFFIS, Oldenburg. His main fields of interest are low power design and power estimation.