Analysing the Robustness of Surfing Circuits

Suwen Yang and Mark R. Greenstreet*

Department of Computer Science
University of British Columbia
Vancouver, BC V6T 1Z4
Canada
{swyang,mrg}@cs.ubc.ca

1 Introduction

We consider the robustness of surfing circuits. Surfing is a technique for implementing high-speed digital pipelines that exploits the analog dynamics of the underlying circuits. Thus, verification must consider the analog behaviour of the design.

In this extended abstract, we give a description of surfing in section 2 to make the abstract self-contained. Section 3 presents a method for analysing the sensitivity of surfing circuits to small-signal disturbances by approximating the continuous time system as a sampled-time one and computing the eigenvalues of the appropriate sensitivity matrix. In section 4 we briefly sketch results from using our approach. We go on to show how the small-signal analysis provides a basis for exploring large-scale, non-linear behaviours by examining the noise margins of our circuits.

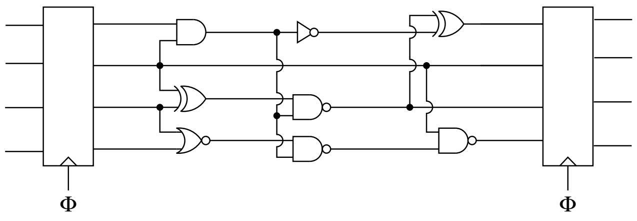


Fig. 1. A Synchronous Pipeline Stage

2 Surfing

Figure 1 shows a typical synchronous pipeline stage. For proper operation, the clock period must be longer than the maximum possible delay through the combinational logic. Let

* This research has been supported by grants from the BC Innovation Council, Intel, NSERC, and SUN Microsystems

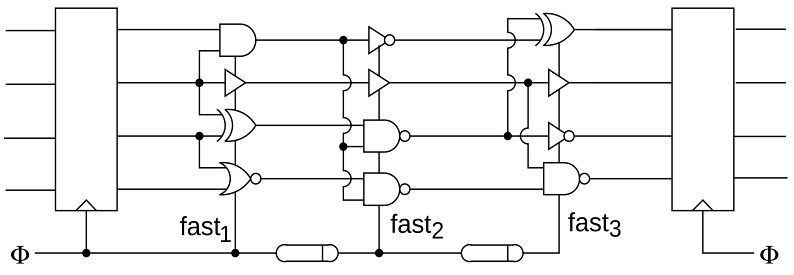


Fig. 2. A Three-Stage Surfing Pipeline

$\delta_{\max}$ be the maximum delay of the combinational logic and P be the clock period. The requirement for synchronous operation is

$$\delta_{\max} < P \tag{1}$$

We note that the purpose of the register is to delay values that arrive before the clock event. In other words, the register slows down the fast paths. Furthermore, the set-up and hold window and the propagation delay of the register slow down all paths.

Wave pipelining mitigates the overhead of registers by allowing multiple waves of computation to propagate through the combinational logic at the same time. Let $\delta_{\min}$ be the minimum delay through the combinational logic. If $\delta_{\min} > \delta_{\max}/2$ and a few technical side conditions are satisfied (see [BC$^+$98]), then the circuit can operate properly for any clock period with $\delta_{\max}/2 < P < \delta_{\min}$. Under these conditions, two waves propagate through the logic at the same time, and the overhead of inserting a latch in the middle of the logic is avoided. In theory, this approach can be extended to allow any number of waves, k, to propagate through the logic without latches as long as

$$\frac{\delta_{\max}}{k} < P < \frac{\delta_{\min}}{k-1} \tag{2}$$

Equation 2 implies $P > \delta_{\max} - \delta_{\min}$. If the total delay of the combinational logic is large, then the total uncertainty (i.e. $\delta_{\max} - \delta_{\min}$) is likely to be large as well, thus requiring a large clock period. In other words, while wave pipelining might seem most appropriate for use with logic with long worst-case paths, it is exactly in these situations where the large, cumulative timing uncertainty prevents using a large number of waves to achieve a short clock period. In practice, applications of wave pipelining have been largely limited to specialized structures (e.g. on-chip caches) and small numbers of waves (i.e. two or three).

Figure 2 depicts a surfing pipeline. Each logic element has an extra input labeled "fast". When fast is asserted, the delays of the logic elements are reduced. For the logic elements regulated by fast_i, let $\delta_{\mathsf{fast},\max,i}$ and $\delta_{\overline{\mathsf{fast}},\min,i}$ denote the maximum delay when fast is asserted and the minimum delay when fast is not asserted respectively. Let $\delta_{\mathrm{TC},\min,i}$ and $\delta_{\mathrm{TC},\max,i}$ be lower and upper bounds for the delay through the timing chain from fast_i to fast_{i+1} respectively. As described in [WG02,WG03], surfing occurs for each stage, timing pulse propagates faster than values in the data path when fast is not asserted, and slower

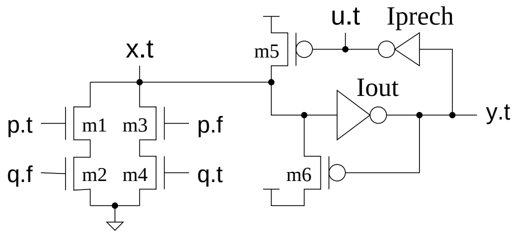


Fig. 3. A Self-Resetting Domino XOR-Gate (true half)

than values in the data path when fast is asserted. In other words, we require:

$$\forall i.\ \delta_{\mathsf{fast},\max,i} < \delta_{\mathrm{TC},\min,i} \leq \delta_{\mathrm{TC},\max,i} < \delta_{\overline{\mathsf{fast}},\min,i} \tag{3}$$

To see this, note that if inputs for a logic element arrive before the fast signal, then the delay of the logic element will be greater than that of the fast signal because $\delta_{\mathrm{TC},\max,i} \leq \delta_{\overline{\mathsf{fast}},\min,i}$. Events in the timing path will tend to catch up with events in the logic path. Conversely, if the inputs arrive when the fast signal is already high, then the delay of the logic element will be less than that of the fast signal because $\delta_{\mathsf{fast},\max,i} < \delta_{\mathrm{TC},\min,i}$, and events in the logic path will tend to catch up with those in the timing path. Thus, events in the logic path are attracted to the rising edge of the fast signal at each stage.

To implement surfing we need logic gates whose delay can be modulated by the fast signal. In this paper, we will use the pre-switching approach to surfing presented in [WG02]. We use this approach for its simplicity rather than its optimality. We intend to apply our methods to other surfing circuits in the near future.

The surfing circuits from [WG02] are based on self-resetting domino CMOS [CC+91]. Because domino gates are monotonic, dual-rail implementations are commonly used that compute separate “true” and “false” outputs for each logical signal. Figure 3 shows the true-half of a self-resetting domino XOR gate. Inputs p.t and p.f represent the “true” and “false” versions of p. Likewise for q.t, and q.f. Output y.t is the “true” version of y, and a similar circuit can be used to produce y.f. For example, if p is true and q is false, then p.t and q.f will go high. Node x.t will be pulled low by transistors m1, and m2, and inverter Iout will drive y.t high. The circuit is self-resetting: once y.t goes high, inverter Iprech drives node u.t low; transistor m5 restores x.t to a high level; and inverter Iout restores y.t to a low value. Transistor m6 is a “keeper” transistor. Between output pulses, node y.t is low and transistor m6 is conducting. This keeps node x.t high in spite of inevitable leakage currents.

Self-resetting domino is a pulse-mode logic. Logic values are communicated by the *rising* edge of pulses; thus, n-channel devices are used to implement the logic function. This reduces the capacitive load presented by the input of a logic gate and takes advantage of the higher transconductance of n-channel devices compared with p-channel ones. The self-resetting design eliminates the need for multi-phase clocks and allows “footless” designs to be used (as shown). These features allow self-resetting domino circuits to achieve very high performance. However, the pulses at the input to a gate must overlap enough to allow the dynamic nodes (e.g. x.t in the figure) to be pulled low. To avoid short-circuit currents, the pulses must be

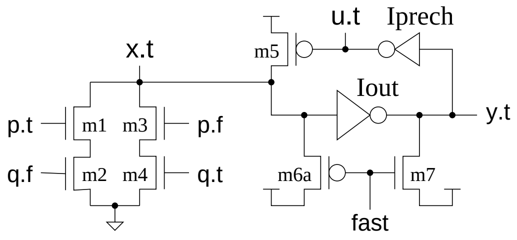


Fig. 4. A Self-Resetting Domino XOR-Gate (true half)

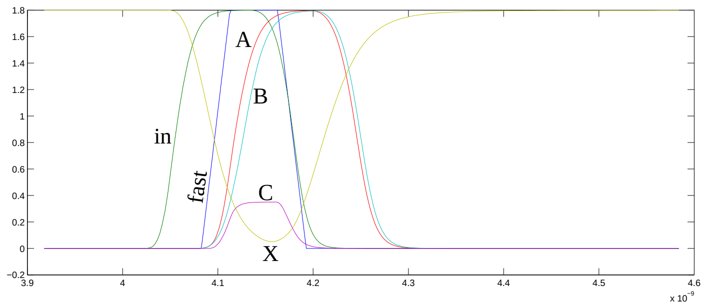


Fig. 5. Operation of a Surfing Gate

short enough to ensure that the inputs to a gate return to low values before precharging starts. These constraints make self-resetting design challenging [DY99].

Surfing mitigates the timing issues of self-resetting design. Figure 4 shows when fast is low, no input pulses are expected, and transistor m6a keeps node x.t high. During the high portion of a fast pulse, node x.t floats at its high level until appropriate input pulses arrive. Furthermore, current flowing through transistor m7 pulls node y.t slightly above ground. In particular, transistor m7 and the pull-down device in inverter Iout form a voltage divider. As these are both n-channel devices, their properties track closely over variations of fabrication parameters. We have found that by making m7 about 80% of the width of the pull-down in Iout, it pulls y.t to about $0.2V_{dd}$ when fast is asserted. This provides the speed-up required for surfing.

Figure 5 shows the operation of the surfing gate. Curves A, B, and C show the propagation of a surfing pulse, the propagation of the same input pulse with fast connected to ground, and the response of the gate when no input pulse is received. Comparing curves A and B reveals the speed-up that occurs when fast is asserted. Curve C shows the voltage shift on the output due to pre-switching independent of the input pulse. Curve "in" is the input for gates A and B, and X is the pre-charged node for gate A. This voltage shift reduces the voltage noise-margin of the surfing circuit. Thus, surfing trades voltage robustness for timing robustness. The remainder of this paper describes an analytical framework for quantifying this trade-off.

3 Analysing the Robustness of Surfing

The central idea in surfing is that the timing of events in the logic circuits converges to the timing of events in the timing chain. Let δ_{fast} be the time from the arrival of the rising edge of the fast pulse at stage k to its arrival at stage $k+1$. Intuitively, pulses at stage $k+1$ should look like the pulses that were at stage k δ_{fast} time units earlier. We consider the simple case where the surfing circuit is a chain of buffers. Let

$$v_k(t) \triangleq \text{voltage at output of stage } k \text{ at time } t \tag{4}$$

We will approximate the continuous time function $v_k(t)$ with samples at times $t_1 \ldots t_n$. We will then construct a sensitivity matrix, S with

$$S(i,j) \triangleq \frac{\partial v_{k+1}(t_i + \delta_{\mathsf{fast}})}{\partial v_k(t_j)} \tag{5}$$

$S(i,j)$ is the perturbation at the output of stage $k+1$ at time $t_i + \delta_{\mathsf{fast}}$ when a unit perturbation is applied to the output of stage k at time t_j and the output of stage k is then returned to its non-perturbed value at time t_{j+1}. Matrix S gives the response at the output of stage $k+1$ to a perturbation of the output of stage k.

If all of the eigenvalues of S are less than one, then we conclude that the chain is stable. Furthermore, the eigenvalues and eigenvectors of S provide insight into the response of the chain to disturbances. In the remainder of this section, we sketch our methods for constructing S. In the next section, we apply this to a chain of surfing buffers and describe the results that we obtained.

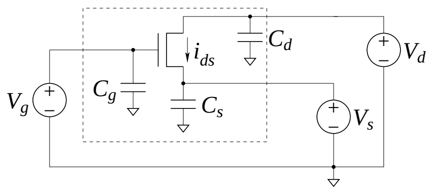
To calculate S, we first find the equilibrium relationship between pulses in the logic circuits and in the timing path for a long chain. We approximate this equilibrium solution by numerically integrating the ODE model for a moderately long surfing chain. We then consider perturbing the pulse in the logic path by an infinitesimal amount, and determine the corresponding disturbance at the outputs of subsequent stages. For all time points t_i and t_j considered by S with $t_i < t_j$, we first calculate the response at time t_j when a node is shifted at time t_i and the circuit state evolves from there. From this, we derive the response when the node is perturbed at the time t_i and restored to its original value at time t_{i+1}. This corresponds to the definition of S. The remainder of this section describes these calculations in detail.

We now consider the response of a stage that receives the fast pulse and a perturbed version of the equilibrium logic pulse as inputs. The original circuit gives rise to an ODE model:

$$\dot{v} = f(v) \tag{6}$$

Where f is derived according to the circuit models for the transistors and capacitors in the surfing chain (see section 4.1). To calculate the small-signal response of this circuit, we use the Jacobian operator: $\mathrm{Jac} f(v)$ is the matrix of partial derivatives of $\dot{v}$ with respect to v. This tells us how small perturbations of v perturb $\dot{v}$. By integrating the perturbations along with the original system, we obtain the small-signal response of the circuit. In particular, we augment the ODE from equation 6 with a matrix $\Delta(t)$ where

$$\dot{\Delta} = (\mathrm{Jac} f(v))\Delta \tag{7}$$



$$\begin{aligned} V_{th} &= \text{threshold voltage} \\ k &= \text{proportionality factor} \\ S &= \text{shape factor, } W/L \end{aligned}$$

$$\begin{aligned} V_{ge} &= V_g - V_s - V_{th} \\ V_{ds} &= V_d - V_s \end{aligned}$$

$$\begin{aligned} i_{ds} &= 0, && \text{if } V_{ge} \le 0 \\ &= \tfrac{kS}{2} V_{ge}^2, && \text{if } 0 \le V_{ge} \le V_{ds} \\ &= kSV_{ds}(V_{ge} - \tfrac{1}{2}V_{ds}), && \text{if } V_{ds} \le V_{ge} \end{aligned}$$

Fig. 6. A First-Order Transistor Model

where Jac is the Jacobian operator. Let Δ_{t_i} denote $\Delta(t_{i+1})$ with the initial condition that $\Delta(t_i) = I$. Thus, if $v(t) = v_i$ at time i leads to $v(t) = v_{i+1}$ at time $i+1$, then $v(t) = v_i + d$ at time i will lead to $v(t) \approx v_{i+1} + \Delta_{t_i} d$ at time j for sufficiently small d.

We now consider the response of the circuit when a node is perturbed at time t_i and restored to the value of the non-perturbed solution at time t_{i+1}. We define:

$$\begin{aligned} \Gamma_{t_i,t_{i+1}}(k,k) &= 0 \\ \Gamma_{t_i,t_{i+1}}(p,q) &= \Delta_{t_i}(p,q), \text{ if } p \neq k \text{ or } q \neq k \\ \Gamma_{t_i,t_{j+1}} &= \Delta_{t_j}\Gamma_{t_i,t_j}, \text{ if } j \geq i+1 \end{aligned} \tag{8}$$

In other words, $\Gamma_{t_i,t_{i+1}}$ "restores" node k to its original value at time t_{i+1} having been disturbed at time t_i, and Γ_{t_i,t_j} gives the effect of this disturbance at subsequent times. We now have

$$S(i,j) = \Gamma_{t_j,t_i+\delta_{\mathsf{fast}}}(k+1,k) \tag{9}$$

This gives us the desired procedure for analysing the stability of the surfing chain. In this procedure, S is calculated by using equations 8 and 9 with only one sweep of integrating equations 6 and 7 from t_1 to t_n, this results in a factor of n savings in compute time compared with a direct integration of equations 6 and 7 for each possible time point for input disturbances.

4 Results

We have implemented the stability calculation described in the previous section using Matlab. Given functions for computing f and Jac$f$ from equations 6 and 7, computing the Δ, Γ, and S matrices is straightforward. We used a simple, first-order transistor model that we summarize in section 4.1. We then describe the application of our procedure to the surfing chain.

4.1 The Circuit Model

For simplicity, we use a simple, first-order transistor model as indicated in figure 6 to calculate all drain→source currents. Likewise, we assume that all capacitances are fixed capacitances to ground and use the relationship that $i_C = C\dot{v}$. Satisfying Kirchoff's current law at each node yields:

$$\dot{v} = -C^{-1} i_{ds}(v) \tag{10}$$

This gives us the ODE model for our circuit. For our simple model, explicit calculation of the Jacobian operator is straight forward. Thus, we obtain the ODEs from equations 6 and 7 directly.

In principle, our approach could be extended to more realistic, industrial strength models for semiconductor circuits. Explicit calculation of the Jacobian operator would be much more tedious due to the extra complexity of the model, but we see no fundamental reason why our approach could not be applied in a more realistic setting.

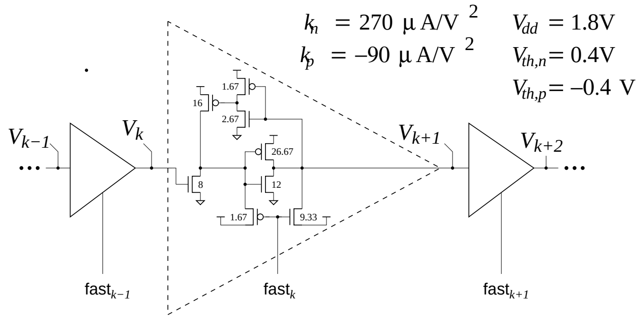


Fig. 7. A Chain of Surfing Buffers

4.2 Small Signal Stability

We implemented the stability analysis described in section 3 for the chain of surfing buffers shown in figure 7. Transistors are labeled with their shape factors, W/L. All transistors have a width of 0.18μ, and k_n, k_p, $V_{th,n}$ and $V_{th,p}$ roughly correspond to the TSMC 0.18μ bulk CMOS process. We estimate gate capacitance as $(0.97F/m^2) * W * L$, and drain capacitance as $(0.003F/m) * W$.

As described in section 3, we determine the equilibrium relationship for pulses in the logic and timing chain by simulation. We then choose an interval that contains the input and fast pulses for a chain with a moderate margin on both sides. For our surfing chain, this interval has a width of 324ps. We divide this into 2ps intervals and have 163 sample points for our analysis. We then compute the S matrix as described in section 3.

Analysing the surfing chain from figure 7, the five largest magnitude eigenvalues are: 0.507, 0.230, 0.0029. and $(1.26 \pm 7.90i) * 10^{-5}$. Given the monotonicity of our circuit models, we expect the sensitivity matrix, S, to be positive definite. We believe that the complex values for the small eigenvalues is most likely a side-effect of our approximation of continuous time with a set of discrete samples.

Figure 8 shows the eigenvectors for the three largest eigenvalues for the surfing chain. The eigenvector corresponding to the largest eigenvalue, 0.507 corresponds to a time shift of the pulse. The left, positive peak shifts the rising edge of the pulse earlier, and the right, negative peak, shifts the falling edge earlier. Because the gate is self-resetting, advancing the

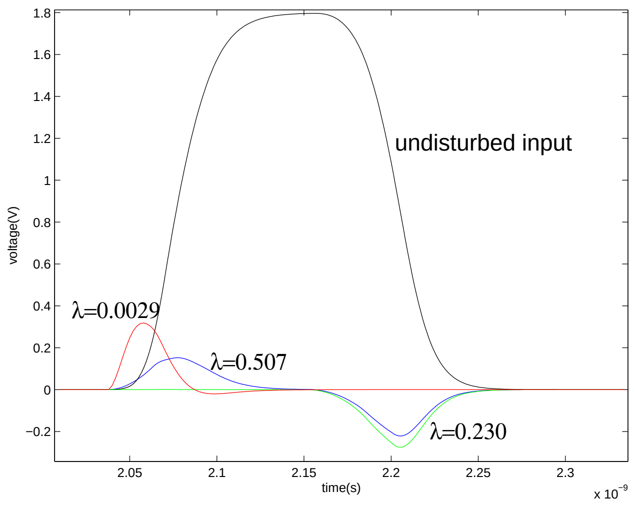


Fig. 8. Eigenvectors of the Sensitivity Matrix for the Surfing Chain

rising edge also advances the falling edge. Thus, the negative peak of the eigenvector has a bigger magnitude than the positive peak. The eigenvector for the second largest eigenvalue, 0.230 shifts the falling edge without disturbing the rising edge. The eigenvector for the third largest eigenvalue, 0.0029 shifts just the rising edge. We find it interesting that the main disturbance modes of the surfing gate can all be interpreted as time shifts of the input events.

All of the eigenvalues for the surfing gate have magnitudes that are significantly less than one. This shows the stability of the surfing gate. Intuitively, any small disturbance will decrease by at least a factor of nearly two from one stage to the next in the chain.

For the sake of comparison, we applied our method to a buffer chain built from static inverters without surfing. For such a chain, we expected that a time shift of an input edge should produce a corresponding shift of the output edge. Thus, the largest eigenvalues for the sensitivity matrix for the non-surfing buffer should be a pair with value 1. When we perform our analysis, this is what we see. The two largest eigenvalue are 0.9972 and 0.9934. Again, we attribute the difference between these values and the predicted value of 1 to artifacts of our time quantization. We also compared the eigenvectors for these two eigenvalues with the time-derivative vectors for the rising and falling edges. They agree to within less than 2%, confirming our understanding of the buffer. The next four eigenvalues are 0.0026, 0.0025, $1.25 * 10^{-5}$, and $1.24 * 10^{-5}$. These are all real and positive as expected.

Next, we explored the use of our method as a design aid. For surfing pipeline, performance is determined by the delay of the timing chain. The delay per stage, δ_{fast}, can be any value between $\delta_{\mathsf{fast},\max,i}$ and $\delta_{\overline{\mathsf{fast}},\min,i}$ (see equation 3); however, robustness is lost for extreme value in this range. First, we studied the trade-off between stage delay and the robustness of the surfing design as measured by the largest eigenvalue of the sensitivity matrix, S. Figure 9

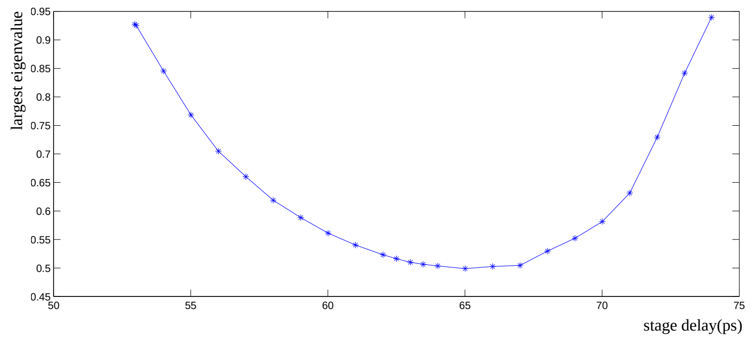


Fig. 9. The Effects of Varying the Stage Delay

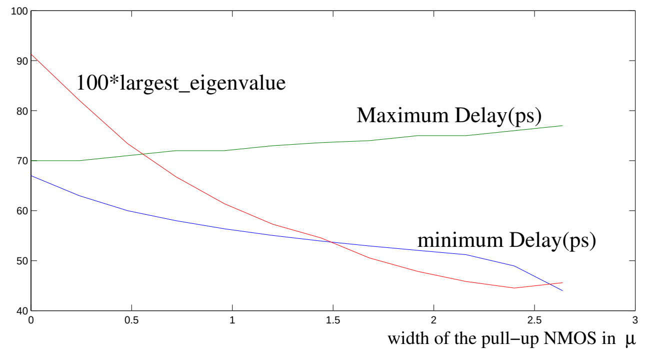


Fig. 10. The Effects of Varying the Width of the N-channel Pull-Up

shows this trade-off. Prior to this work, we have used the midpoint of the minimum and maximum delay as the target value for the stage delay. This plot shows that the maximum robustness actually occurs at a slightly larger delay.

Figure 10 shows the effect of varying the width of the n-channel pull-up transistor in the surfing gate. The plot shows the maximum delay of the gate (with fast low), the minimum delay of the gate (with fast high), and the value of the largest eigenvalue of the sensitivity matrix. Not surprisingly, the minimum delay decreases as the pull-up becomes stronger. There is a slight growth in the maximum delay due to the extra drain capacitance of the larger transistor. When the pull-up is eliminated (width = 0), there is still a small surfing effect contributed by the keeper transistor controlled by fast. Accordingly, the largest eigenvalue is slightly less than 1, namely 0.913. As the pull-up is made stronger, this eigenvalue decreases, quantifying the increase in the strength of the surfing effect. For very large pull-ups, the eigenvalue starts to increase again. This is because the surfing transistor pulls the output of the gate above the n-channel threshold voltage, and larger transistors result in a malfunctioning gate.

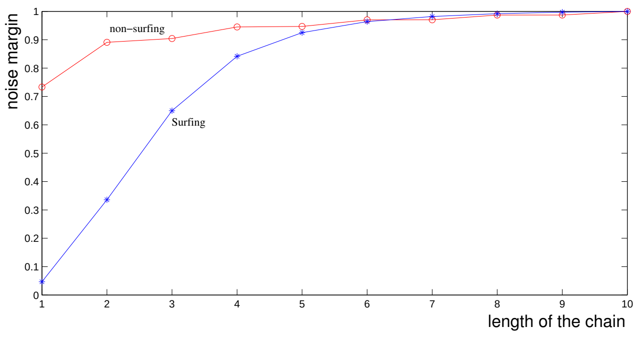


Fig. 11. Noise-Margin Estimates

4.3 Large Signal Stability

The previous results were based on a small-signal analysis of surfing chains. However, the main concern of practicing designers is the large-signal, non-linear, digital behaviour of the circuit. Here, we briefly examine the robustness of surfing circuits in a large-signal model. In particular, we look at the question:

> What is the smallest (by the l_2 metric[1]) disturbance ν at the input of a gate that creates a larger disturbance at the output?

The vector ν gives the disturbance of the input signal at each time point. We formulated this as a non-linear optimization problem using Matlab's fmincon function. Although this is not guaranteed to give us the globally smallest such disturbance, it does shed light on some of the differences between surfing and non-surfing circuits. We note that fmincon failed when using the integrator directly to evaluate the surfing circuit. By using the sensitivity matrix S in the optimization, we obtain successful convergence in relatively short times – about 20 minutes for the problems described here running on a 3GHz Pentium IV processor.

Figure 11 shows the noise margin as described above as a function of the length of the chain for both surfing buffer and non-surfing inverter chains. We have normalized the asymptotes in both cases to 1 because the actual values depend strongly on the load capacitance and we have not yet determined how to make a fair comparison. For both chains, the noise margin increases with the length of the chain. This is because the while the input and output disturbances have the same magnitude by the l_2 metric, they don't necessarily have the same shape. Thus, the disturbance that will pass through one stage will not necessarily propagate through a long chain. This effect is especially pronounced for the surfing chain. In fact, we note that the chain appears to have almost no noise margin when a single stage is considered, but is quite robust when a chain of four or five stages is analysed.

[1] The l_2 measure of a vector ν is the square-root of the sum of the squares of the components of ν.

5 Conclusions

We have presented a method for analysing the robustness of surfing circuits. Surfing creates event attractors whereby events in the data path are attracted to a fixed relationship to events in the timing path. We analyse these attractors by constructing the mapping from behaviours at one stage in the pipeline to those at the next with a time shift corresponding to the nominal stage delay. This analysis shows that surfing circuits are very robust (i.e. highly damped) with respect to disturbances of the input. We have presented preliminary results showing how this approach can be applied to the design of surfing pipelines and to noise-margin analysis.

References

[BC+98] Wayne P. Burleson, Maciej Ciesielski, et al. Wave-pipelining: a tutorial and research survey. *IEEE Transactions on VLSI Systems*, 6(3):464–474, September 1998.

[CC+91] Terry I. Chappell, Barbara A. Chappell, et al. A 2-ns cycle, 3.8-ns access 512-kb CMOS ECL SRAM with a fully pipelined architecture. *IEEE Journal of Solid-State Circuits*, 26(11):1577–1585, November 1991.

[DY99] Ayoob E. Dooply and Kenneth Y. Yun. Optimal clocking and enhanced testability for high-performance self-resetting domino pipelines. In *Proceedings of the Twentieth Anniversary Conference on Advanced Research in VLSI*, pages 220–214, March 1999.

[WG02] Brian D. Winters and Mark R. Greenstreet. A negative-overhead, self-timed pipeline. In *Proceedings of the Eigth International Symposium on Asynchronous Circuits and Systems*, pages 32–41, Manchester, UK, April 2002.

[WG03] Brian D. Winters and Mark R. Greenstreet. Surfing: A robust form of wave pipelining using self-timed circuit techniques. *Microprocessors and Microsystems*, 27(9):409–419, October 2003.