Designing robust asynchronous circuit components

S. Mohammadi, S. Furber and J. Garside

Abstract: Asynchronous circuits require components that display hazard-free operation under normal input conditions. In addition, quasi-delay-insensitive circuits are based on the assumption of isochronic forks, an assumption that can in practice be compromised by threshold variations due to the use of, for example, dynamic or pseudo-dynamic C-gate circuits. In the paper, the authors investigate the severity of these problems in practical circuits. It is shown that threshold variations are much less significant than has previously been assumed, but hazard-free operation is, by contrast, a much more significant problem. Gates with a stack of transistors in series can exhibit charge-sharing problems under specific input sequences that expose hazards that are not evident in the logic description. A design methodology is proposed which overcomes the charge-sharing problem, resulting in more robust circuits.

1 Introduction

Asynchronous circuits are classified by, *inter alia*, their delay model [1]. The most robust class of circuit is delay-insensitive (DI), which means that its functionality is independent of all gate and wire delays. It has been shown that the class of true DI circuits is very restrictive [2], and practical circuits require some relaxation of the DI assumptions. The minimum relaxation that supports the design of practical systems is the class of quasi-delay-insensitive (QDI) circuits, where the DI circuit model is extended to allow isochronic forks. An isochronic fork is a forked wire where each branch of the fork has the same delay. In practice, the requirement is that the difference between the branch delays is less than the minimum logic gate delay.

The isochronic fork assumption is affected by differences between the thresholds of the gates that are driven by the branches of the fork. This is because the wire will have non-zero rise and fall times, and any threshold difference will cause an effective delay variation in addition to the wire delay difference. Unless carefully managed, such variations could violate the QDI model. In an analysis of different circuit styles, van Berkel argued that some styles of gate circuit were prone to exhibit unusual thresholds and should therefore be avoided, at least where an input signal is from a branch of an isochronic fork [3].

Another issue that affects the robustness of asynchronous circuits is that all asynchronous control components are assumed to operate free from glitches and hazards when used correctly, and several synthesis tools exist to produce logically hazard-free circuits. However, many of the circuits produced by such tools require gates with a high fan-in, and these gates may be prone to glitching as a result of charge-sharing under specific input sequencing conditions.

© IEE, 2003

IEE Proceedings online no. 20030349

doi:10.1049/ip-cds:20030349

Paper first received 28th January and in revised form 1st October 2002

S. Mohammadi is with Cogency Semiconductor Inc., 144 Front Street West, Suite 600, Toronto, ON, Canada M5J 2L7

S. Furber and J. Garside are with the Department of Computer Science, The University of Manchester, Oxford Road, Manchester M13 9PL UK

In this paper we show that the threshold variation problem is less severe than was previously thought, but the charge-sharing problem cannot be ignored.

2 Muller C-element circuits

Many asynchronous control components are variants of the Muller C-gate. The basic 2-input Muller C-gate is a gate whose output will be high when both inputs are high, low when both inputs are low, and retains its previous level when one input is high and the other low. These gates contain state, and are used for a range of synchronisation purposes.

Three different C-gate designs are considered here, based on dynamic, pseudo-dynamic and static circuits. Fig. 1 shows a pseudo-dynamic C-gate circuit. The internal state is maintained by a weak feedback inverter across the output inverter. When the inputs A and B are the same, either the n-transistor stack or the p-transistor stack will conduct, driving the output to the required level (over-driving the weak feedback inverter where necessary). When A and B differ, neither stack conducts and the weak feedback inverter comes into play to retain the output state.

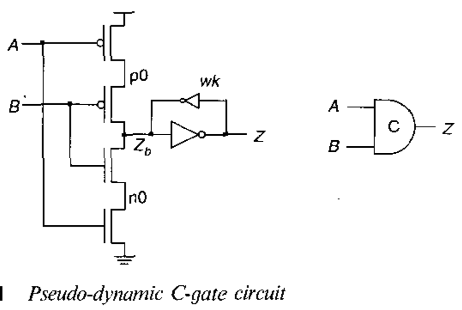

Fig. 1 *Pseudo-dynamic C-gate circuit*

The dynamic C-gate circuit differs from the pseudo-dynamic circuit only in that the weak feedback inverter is omitted. When A and B differ, the output state is retained as the charge on node Z_b. The charge is not retained indefinitely, and the leakage characteristics determine the maximum time that the gate operates correctly when A and B differ.

In both the dynamic and pseudo-dynamic gates the non-inverting output is used. The gate has a good output drive and the inputs pass through two inverting stages to reach the output.

A static C-gate circuit is shown in Fig. 2. Active feedback is used to ensure that the output is driven in all the input states, and no over-driving is required in any input transition. Both the inverting and the non-inverting outputs may be used, with the former representing only a single inverting stage from input to output. Note that only the leftmost transistor stacks are important when switching Z_b; the other stacks are used only for state holding.

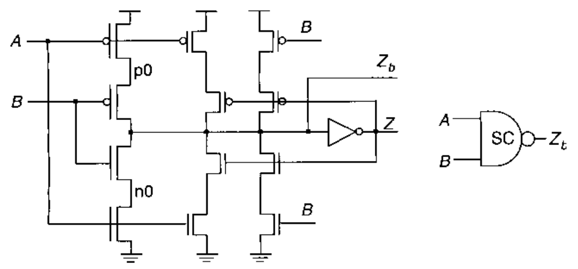

Fig. 2 *Static C-gate circuit*

This is not the only possible static 2-input C-gate circuit, and it is arguably not the best [4]. However, the 'symmetric' C-gate circuit recommended by Shams *et al.* does not extend to the more complex forms discussed in the remainder of this paper. It is also clearly possible to reduce the number of transistors in this circuit by sharing those in the feedback circuit closest to Z_b.

The choice of which C-gate style to use in a particular circumstance depends on several factors, including the input load (where the static gate presents a higher load as the inputs drive additional transistors), the output drive (where the non-inverting output from all gates has better drive) and the gate delay (where the inverting output from the static gate is best).

2.1 C-gate thresholds

An inspection of the circuits in Figs. 1 and 2 shows that they have fundamentally different input threshold characteristics:

- *The static circuit:* This has the usual property of a static ratioed CMOS circuit. At all stages during an input transition there is a balance between the n- and p-type stacks that results in a threshold close to 50% of the supply voltage, V_{dd}, with small threshold variations attributable to transistor sizing.
- *The dynamic circuit:* This has no such balance. On a rising edge where the other input has already become high, the input threshold is just the threshold of the n-transistor, V_{tn}. Similarly, the threshold on a falling edge is $V_{dd} - V_{tp}$. These thresholds are quite different from the static-gate threshold, and this has been used as a rationale for ruling out the use of dynamic gates on isochronic forks [3].
- *The pseudo-dynamic gate:* This has some balancing effects due to the weak feedback inverter that will move the threshold back towards that of the static gate, the extent of this effect depending on how weak the feedback inverter is.

In order to demonstrate the effects of these threshold differences, SPICE simulations of the three circuits were undertaken.

Fig. 3 shows the switching characteristics of node Z_b of the three different circuits using 0.35 μm technology. The input is a very slow ramp starting just below the gate threshold (V_t). This allows us to observe the behaviour of the gate at the threshold. We can see that the three circuits do, indeed, switch at different points on the input ramp. However, this is observable only with input ramps that are very slow indeed. For faster inputs the differing thresholds have very little effect, as shown in Fig. 4. Note that even here the edge speed is much slower than would normally be used, at 3 ns. Typical edge speeds are well below 1 ns in circuits based on this technology.

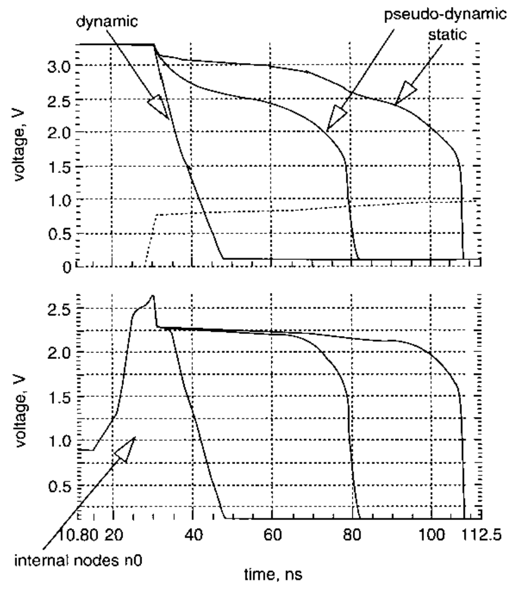

Fig. 3 *Switching characteristics of the static, pseudo-dynamic and dynamic C-gate circuits*

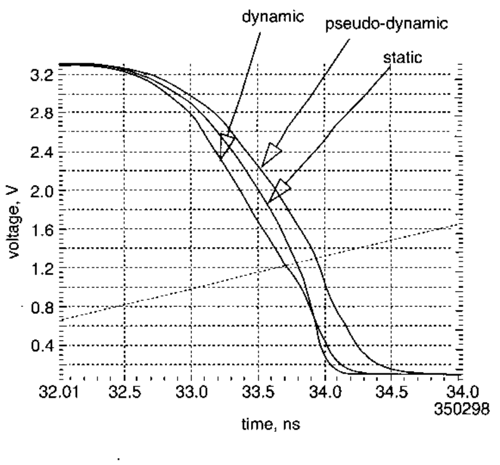

Fig. 4 *Switching characteristics with a faster edge speed*

We conclude therefore that, although the threshold differences can be displayed, this is only possible at switching speeds that would never be used in practical circuits, and the use of the dynamic and pseudo-dynamic circuits in isochronic forks should not be precluded on these grounds.

This conclusion is in direct contradiction to the conclusions of previously reported findings [3] and underlines the fact that the circuit-level behaviour of even quite simple transistor circuits is often subtle and counter-intuitive. The different switching thresholds of static, dynamic and pseudo-dynamic C-gates in practice lead to much smaller variations in their output timings than the threshold differences would suggest, provided that excessively slow input edges are avoided.

3 Charge sharing

Charge sharing refers to the redistribution of charge when two networks at different initial voltages are connected together, and it is a major source of noise in CMOS gates.

Two kinds of charge sharing have been identified [5]: pure charge sharing, where two non-driven subnets are connected through a switch; and charge sharing with a driven path, which is the same as pure charge sharing except that a resistive path to a driving source exists for one of the subnets.

The second form commonly arises in digital CMOS circuits where a switching transistor connects an internal node of a gate with a high-impedance fan-out network to the output whose capacitance is initially charged to a different voltage.

Charge sharing occurs in CMOS gates when either charge flows from the output to internal nodes that were previously discharged, or when charge flows from internal nodes previously charged to the output of the gate. For some particular combination of the inputs of some gates, charge sharing can generate glitches on the output which can then, if large enough, trigger following gates and change their states. The charge redistribution problem has been discussed for CMOS domino logic where it causes the output of the tree to switch falsely, placing the incorrect value on a signal and causing other parts of the logic to discharge falsely [6].

In static asynchronous circuits the glitches caused by charge sharing can have an equally destructive effect as many of the gates, such as the C-gate already described, are state-holding. A sufficiently large glitch can switch the state of the gate, causing erroneous operation and, possibly, deadlock; asynchronous control circuits are prone to complete seizure if an error causes an unwanted state change.

To illustrate this problem we will employ a rather more complex C-gate, the C2_4-gate (illustrated in Fig. 5). This is an example of an asymmetric C-gate, and is chosen because the long n-transistor stack renders the effects of charge sharing particularly severe. The function of the gate is to

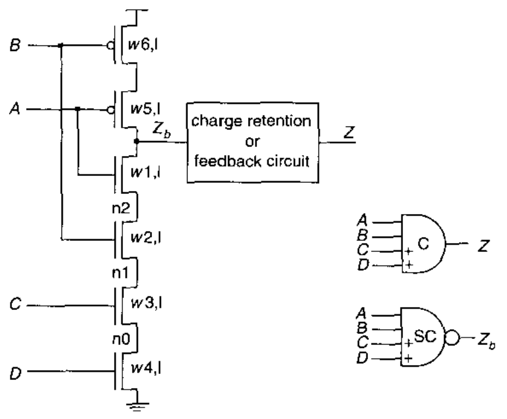

Fig. 5 *C2_4-gate circuit*

switch high when A, B, C and D are all high, but to switch low as soon as A and B are both low. The state-holding circuit can be dynamic, pseudo-dynamic or static as discussed earlier. The static variant is shown in Fig. 6.

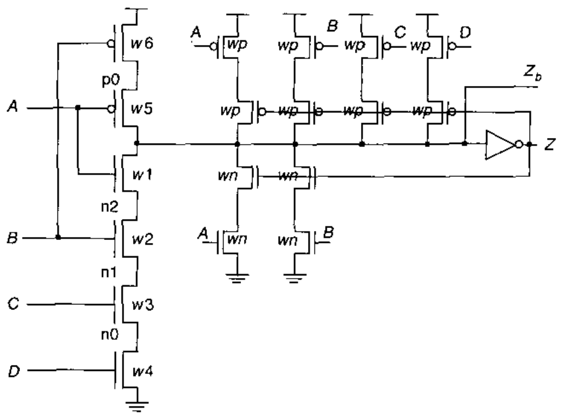

Fig. 6 *Static C-gate sc2_4*

The sc2_4 gate was simulated using SPICE [7] operating at worst-case conditions (V_{dd} = 3.3 V, V_{ss} = 0.1 V, slow-slow process corner, at 25 °C). The gate was laid out using 0.35 μm CMOS design rules. All capacitances were extracted from the layout and the output Z_b had a minimum load corresponding to an inverter. Simulations show that this gate has a severe charge sharing problem under certain input sequences. Fig. 7 shows the glitches which occur in the worst case for the input sequences below.

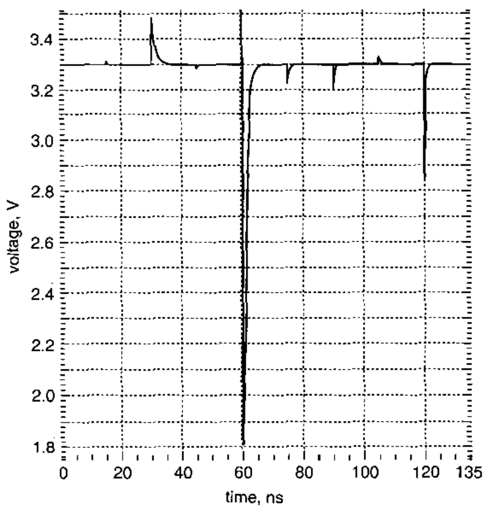

Fig. 7 *SPICE results for the sc2_4 gate*

The vectors for the first glitch (at 60 ns) shown in Fig. 7 are:

(A B C D)	
(0 0 0 1);	discharge n0
(0 0 1 1);	discharge n1
(0 1 1 1);	discharge n2
(0 1 1 0);	turn bottom n-FET off
(1 1 1 0);	turn top n-FET on

The vectors for the second glitch (at 120 ns) shown in Fig. 7 are:

(1 1 0 0); turn C n-FET off
(1 0 0 0); charge p0
(1 1 0 0); turn B p-FET off
(0 1 0 0); turn A p-FET on.

In the simulated circuit the transistors all have different widths, that is $w4 > w3 > w2 > w1$ for the n-transistor stack and $w6 > w5$ for the p-transistor stack. The diffusion capacitances accentuate the charge-sharing problem as the diffusion area is increased to accommodate the stepped widths (see Fig. 8), increasing the diffusion capacitance and the charge stored. The feedback transistors have minimum widths. The transistor sizes in λ are $w4$, $w3$, $w2$, $w1 = 47, 43, 35, 31$ and $w6$, $w5 = 45, 40$, wp, $wn = 5, 5$ and the transistor length is 2, where λ is 0.2 µm in the technology used for this work.

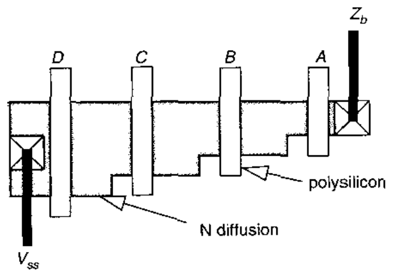

Fig. 8 *Stacked transistor layout*

The first negative glitch in Fig. 7 clearly goes far enough to perturb other gates that may be connected to its output, and the risk of it causing false gate switching in subsequent logic is high. In a similar gate with just one p-transistor in the pull-up stack, the sc1_4, the glitch is large enough to cause the state of the gate itself to switch erroneously (see Fig. 9). Here the glitch on Z_b is deep enough to cause Z to switch, whereafter the feedback circuit completes the transition on Z_b, leaving the gate in the wrong stable state at the end of the sequence of input transitions.

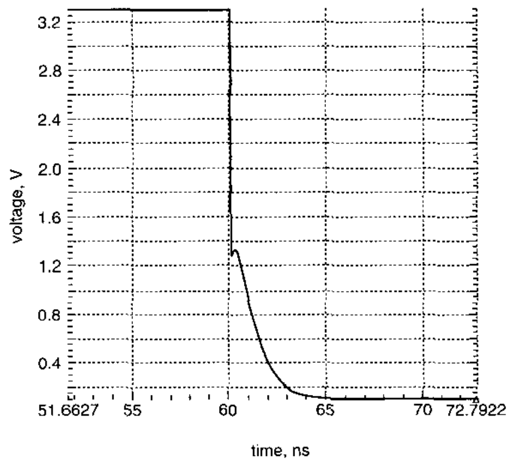

Fig. 9 *sc1_4 fails when charge sharing occurs*

As stated earlier, the simulation includes a minimum inverter (and no wiring) as the load on Z_b. This is almost the worst-case assumption for charge sharing, as any increase in load capacitance will reduce the size of the glitch. However, it is appropriate when designing library components to ensure that they operate correctly under all possible conditions of use.

3.1 Transient suppression

In static C-gates the inverting output directly propagates the transients generated in the input stage. Dynamic and pseudo-dynamic C-gates require the inverter buffer stage, and this filters out the internal glitch to leave a fairly clean output (provided that the glitch does not cause a state change).

Clearly, the non-inverting buffered output could be used with the static C-gate too, but this would negate the performance advantage of the static variant, and still leave open the risk of the internal glitch causing an erroneous state change.

4 Reducing the charge sharing problem

In this Section we examine the techniques used to alleviate the problem caused by charge redistribution. Different methods of reducing the charge sharing problem are considered.

First, since it is not possible to eliminate glitches completely from CMOS circuits, we must decide what glitch amplitude is acceptable and use this to determine when we can claim a gate is free from the problem. We suggest a conservative goal is to limit glitches to around 10% of the power supply voltage. For example, in the cases we have studied, where a 3.3 V power supply voltage is used, gates presenting glitches of less than 350 mV (negative glitches relative to the 3.3 V rail or positive glitches relative to the 0 V rail) are considered to be safe.

We have considered three approaches to overcoming the charge sharing problem:

(i) *Decrease the diffusion capacitance between stacked transistors:* Often in a gate with stacked transistors, the transistor sizes are stepped (as shown in Fig. 8). This forces a bigger diffusion area between adjacent gates (due to the design rule constraint on the minimum distance between the gate polysilicon and the diffusion step) and therefore a bigger capacitance. We can minimise these capacitances by using the same size for all the transistors in the stack.

(ii) *Change the sizes of the transistors in the feedback network:* These transistors are responsible for charge retention on the internal node. In the case of static C-gates, the feedback transistors are by default minimum size, but for a gate presenting glitches on its output, increasing their size can help fight charge sharing. This method has a disadvantage: it increases the output capacitance and so the propagation delay of the gate. Since charge sharing is almost instantaneous, changing the size of the feedback transistors only affects the widths of the glitches and not their heights, so it will not help us meet our acceptance criterion.

(iii) *Use a complex input stage structure:* This consists of duplicating a stack of n- and/or p-transistors, giving two parallel stacks where each transistor's width is reduced by half ($wi/2$, l) and in one of the stacks the transistors are arranged in the reverse order. Fig. 10 shows the modification of the C2_4 gate into a pair of n-transistor stacks. This method allows the diffusion capacitance to be reduced; nodes *nia* and *nib* are half as capacitive. So when, for instance, the sequence (0 1 1 0), (1 1 1 0) is applied to (A B C D) (considering internal nodes *nia* and *nib* initially to be discharged) the charges being dumped from Z_b to the

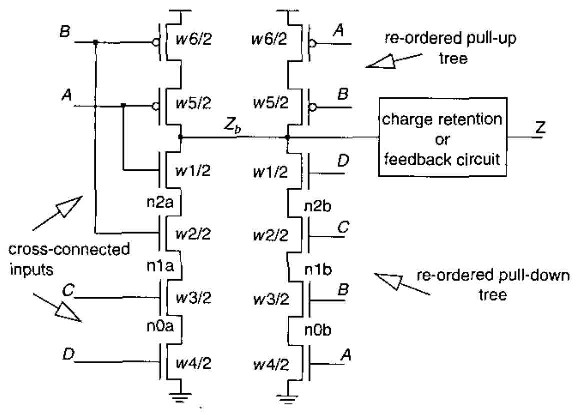

Fig. 10 *Gate sclc2_4 with a pair of stacks*

internal nodes are now half and therefore the glitch appearing on Z_b will be reduced by almost half. One has to bear in mind that now the input capacitance is slightly higher due to longer routing.

Simulations show that none of these methods is satisfactory on its own, but converting the structure of a gate to a pair of stacks, combined with one or both of the other methods, is a good way to deal with the charge sharing problem.

4.1 Results

The structure of the static C-gate sc2_4 was altered to yield the gate sc2_4_p, using parallel stacks for both the n- and the p-transistors. Each n- and p-stack now has half-width transistors. One of each pair of stacks has a reverse input order. Simulation shows that the amplitude of the glitches is still over 10 per cent of the power supply. To reduce the glitches further, the feedback transistors were made larger, which has the side-effect of increasing the capacitance on the output node. For this particular gate, the p-transistors in the feedback network are made three times and the n-transistors twice the minimum width. Consequently, the drive strength of the inverter also has to be increased. Fig. 11 shows the SPICE results from the modified gate sc2_4_p. The two major glitches are now reduced to below

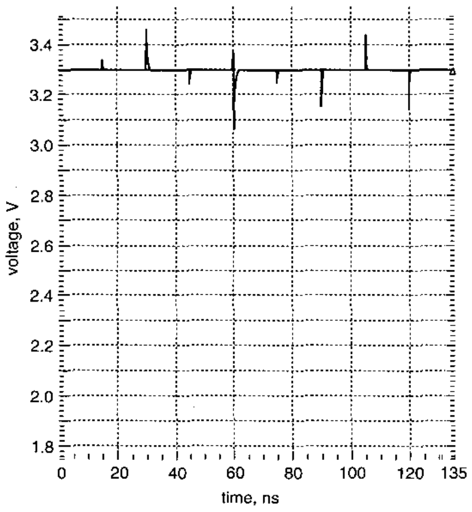

Fig. 11 *SPICE figure for sc2_4_p*

10% of the power supply voltage. The transistor sizes for the feedback circuit are now $wp = 20\lambda$ and $wn = 10\lambda$.

This method turns out to be very effective. In the case of the gate mentioned earlier that failed internally, the sc1_4 (Fig. 9), using a re-ordered pull-down tree with cross-connected inputs and stronger feedback reduced the amplitude of the internal glitch down to 350 mV, completely removing any risk of the state flipping.

This method was used to develop a complete library of asynchronous cells that formed the basis of the design of the Amulet3i asynchronous processing subsystem employed on the DRACO system-on-chip telecommunications controller [8].

5 Conclusion

We have discussed issues relating to the robustness of state-holding gates for use in asynchronous circuits, and shown that previous concerns over threshold variations between static and dynamic gate structures are not well founded. On the other hand, glitches caused by charge sharing are a significant source of potential circuit failure. In one case we have shown that an internal glitch can be large enough to flip the state of a gate erroneously.

Three different approaches to reducing charge sharing problems have been offered, based on minimising the capacitances of the charge-bearing nodes, duplicating and reordering transistor stacks and increasing feedback strengths. While no one of these is sufficient on its own to solve the problem, a combination of two or all of them has proved to work in all of the cases we have investigated.

We aim to have glitches that are less than 10% of the supply voltage. All of the simulations are based on gates with a minimum load (one inverter) on their outputs, so the worst-case glitch would be generated. In a real circuit, gates have higher output loads and so the glitches would be less than 10% of the supply voltage.

Using the approaches described here, it has been possible to develop a cell library for asynchronous circuits that is robust under many different conditions of use and has been used in the design of the Amulet3i large-scale asynchronous system-on-chip design. The silicon for this design has been tested and is highly functional and no errors attributable to the cell library have been found.

The techniques described here for reducing charge sharing were developed in the context of the development of a cell library to support asynchronous design; clearly they are equally applicable to cell libraries for conventional clocked design where the problems of charge sharing, although perhaps less severe, still apply.

6 Acknowledgments

The work described in this paper was carried out in the course of the development of Amulet3 within the EU-funded OMI-ATOM project, and the authors are grateful to the European Commission for their support for this work. The helpful comments of the reviewers of the first draft of the paper are also gratefully acknowledged.

7 References

1 Sparsø, J., and Furber, S.B.: 'Principles of asynchronous circuit design: a systems perspective' (Kluwer, 2001)
2 Martin, A.J.: 'The limitations to delay-insensitivity in asynchronous circuits'. Proceedings of 6th MIT Conference on Advanced research in VLSI, MIT Press, 1990, pp. 263–278
3 van Berkel, C.H.: 'Beware the isochronic fork', *Integr. VLSI J.*, 1992, **13**, (3), pp. 103–129

4 Shams, M., Ebergen, J.C., and Elmasry, M.I.: 'Modeling and comparing CMOS implementations of the C-element', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 1998, **6**, (4), pp. 563–567

5 Chu, C.Y., and Horowitz, M.A.: 'Charge-sharing models for switch-level simulation', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 1987, **6**, (6), pp. 1053–1061

6 Oklobdzija, V.G., and Montoye, R.K.: 'Design-performance trade-offs in CMOS-domino logic', *IEEE J. Solid-State Circuits*, 1986, **21**, (2), pp. 304–306

7 Nagel, L.W., and Pederson, D.O.: 'Simulation program with integrated circuit emphasis (SPICE)'. Report ERLM383, Electronics Research Laboratory, University of California, Berkeley, 1983

8 Garside, J.D., Furber, S.B., and Chung, S.-H.: 'AMULET3 revealed'. Proceedings of Async'99, Barcelona, Spain, 19–21 April 1999, pp. 51–59