A CALCULUS FOR THE DERIVATION OF C-MOS SWITCHING CIRCUITS

C.A.R. Hoare

Draft - April 1988

Summary. This paper presents a method for correct design of C-mos switching circuits by simple calculation checked by occasional proof. It meets many of the criteria proposed in an earlier less successful paper [Hoare and Gordon]. Indeed, it provides a model which might be emulated in the search for new design methods in other branches of engineering, for example, software. However, no claim is made for its practical utility in hardware design.

1. Introduction

In the design of a combinational switching circuit, we let the wire names a, b, c stand for the signals 0 or 1 presented on the input wires before each cycle of operation; and similarly x, y, z stand for the signals readable on these output wires at the end of the cycle. The cycle ends when the circuit reaches a stable state, in which there is no current passing through any of its transistors. In addition to variables, the constant 1 (true) stands for the power wire, and 0 (false) stands for the ground, which always take these named values.

The stable states of a single transistor are described by a propositional formula relating the signals observable on its gate (g), its source (s) and its drain (d). When the signal on the gate of P-transistor is 0, a conducting path is set up between its source and its drain. If the signals on the source and drain are different, a current will flow. So in any stable state, the signals on source and drain are equal. We therefore define the stable states as those satisfying the formula

$$P(g, s, d) \;\hat{=}\; \neg g \Rightarrow (s \equiv d).$$

An N-transistor is similar, except that it conducts when its gate is 1

$$N(g, s, d) \;\hat{=}\; g \Rightarrow (s \equiv d).$$

The connections of a transistor within a circuit are determined by wire names written as arguments to these defining formulae. For example, a negation circuit with input a and output z is described by the pair of transistors

$$P(a, 1, z), \quad N(a, 0, z).$$

A circuit stabilises exactly when all its transistors have stabilised. So the stable states of a whole circuit are described by the conjunction of the formulae describing the stable states of its individual transistors. In the example above, this is

$$(\neg a \Rightarrow (1 \equiv x)) \wedge (a \Rightarrow (0 \equiv z)).$$

This simplifies to

$$z \equiv \neg a$$

which is exactly the specification of a negation circuit.

The design method described in this paper will ensure that every circuit will be equivalent to its specification. From the specification it deduces, one by one, the transistors needed to implement it. A proof at the end of the design checks completeness. As an example, we will design a circuit to compute the exclusive *or* function ∇

$$z = a \nabla b \qquad\qquad \ldots\text{Post}$$

We will assume the availability of a third input c, which carries the negation of the input signal b. This assumption is formalised in a precondition

$$c = \neg b \qquad\qquad \ldots\text{Pre}$$

Completion of the specification phase of the design is checked by a proof of its feasibility

Theorem 1. $\qquad$ Pre $\Rightarrow \exists z.$ Post.

The implementation proceeds by assuming Pre $\wedge$ Post, and deducing the following six transistors, one by one

$$\left.\begin{array}{rl}
P(a,b,z): & \neg a \Rightarrow (b \equiv z)\\
N(a,c,z): & a \Rightarrow (c \equiv z)\\
P(b,a,z): & \neg b \Rightarrow (a \equiv z)\\
N(c,a,z): & c \Rightarrow (a \equiv z)\\
P(z,a,b): & \neg z \Rightarrow (a \equiv b)\\
N(z,a,c): & z \Rightarrow (a \equiv c)
\end{array}\right\} \ldots\text{Imp}$$

By calculation we have proved

Theorem 2: $\qquad$ Pre $\wedge$ Spec $\Rightarrow$ Imp

As a check on the completeness of the design, we also need to prove

Theorem 3: $\qquad$ Pre $\wedge$ Imp $\Rightarrow \exists 1 z.$ Imp.

By a general theorem of the predicate calculus [Hoare], it follows that

$$\text{Pre} \wedge \text{Imp} \Rightarrow \text{Spec}.$$

It is easy to imagine a computer program that will verify the correctness of each transistor as it is added to the design, and even give notice when the design is complete. The only proof method needed is tautology.

Technical note

Assuming the precondition, Theorem 1 has established that the postcondition is satisfiable. From theorem 2, the implementation is also satisfiable. Consequently, there exists an allocation of values 0 or 1 to the wires of the circuit such that the circuit can stabilise. This means that it is always possible that the circuit will converge to a consistent valuation in which no current flows. Some protection against short circuits is thereby achieved. But actual convergence of the hardware to a stable state will depend on further considerations treated in the following section.

2. Signal drive

In the design of switching circuits, uniqueness of the output signal is not a sufficient condition for successful operation of the circuit. In the example shown above, the output z is uniquely defined by the last two transistors alone. But in practice a circuit built from just these two transistors will not work. This is because the hardware of a C-mos transistor is not able to adjust the value of its gate to prevent current flowing between its source and drain; instead, it just fails to stabilise, and current continues to flow. A circuit consisting only of the first two transistors would also be unsatisfactory, because in practice a C-mos P-transistor will attenuate a 0-signal passing between source and drain; and an N-transistor will attenuate a 1-signal.

In order to prove that a circuit will adequately drive its output wires, we need to introduce further propositional variables to describe this property. For any wire name w, let δw be true if wire w is strongly driven, either to 0 or to 1. Such variables are called δ-variables. For an output wire z, the truth of δz requires that z be strongly connected to some other wire which is also strongly driven, for example an input wire, or one of the constant wires 0 or 1; of course these are always strongly driven, i.e. $\delta 0 = \delta 1 = 1$.

A strong connection is defined as one that is capable of propagating strong drive. Such a connection is made by a P-transistor if and only if its gate wire is driven to 0, and also the value of the source and drain are 1. This is expressed by the introduction of a new formula δP, which is true of every P-transistor in the circuit:

$$\delta P(g, s, d) \;\widehat{=}\; \neg g \wedge s \wedge d \wedge \delta g \;\Rightarrow\; (\delta s \equiv \delta d)$$

The corresponding definition of an N-transistor is

$$\delta N(g, s, d) \;\widehat{=}\; g \wedge \neg s \wedge \neg d \wedge \delta g \Rightarrow (\delta s \equiv \delta d).$$

These formulae are called δ-formulae.

Technical note

The formulae δP and δN can never be used to prove that the gate of a transistor is driven (δg is true), no matter what combination of values is taken by its source and drain ($s, d, \delta s, \delta d$). The only way a gate wire can be driven is by connection to source or drain of some other conducting transistor. This correctly reflects asymmetry in the behaviour of C-mos hardware.

The introduction of the δ-variables permits the precondition of the design to state which variables are to be input variables, by postulating that they will be driven at the beginning of each cycle of operation. So the precondition of the EXOR circuit should be strengthened to include

$$\delta a \wedge \delta b \wedge \delta c \qquad \ldots \delta\text{Pre1}$$

which states effectively that a, b, and c are input variables. We can even specify a bidirectional circuit: on any cycle of operation either a or z may be used for input and the other one for output

$$\delta b \wedge \delta c \wedge (\delta a \vee \delta z) \qquad \ldots \delta\text{Pre2.}$$

The δ-Precondition is allowed to contain only positive occurrences of δ-variables.

Similarly the way in which a circuit is intended to drive its output can be specified as a δ-Post condition, for example

$$\delta a \wedge \delta z \qquad \ldots\delta\text{Post}$$

However δPost *must not* be assumed true during the design of an implementation. Indeed, it is the task of the implementor to prove that the whole implementation, together with its associated δ-formula, actually implies δPost. So each time a transistor is added to the circuit, the corresponding δP or δN formula may be assumed without proof; it will be used eventually to prove δPost, and thereby indicate completion of the design.

Returning to the example design of the exclusive or circuit, the relevant δ-formulae are

$$\left.\begin{array}{ll}\delta P(a,b,z): & \neg a \wedge b \wedge z \wedge \delta a \Rightarrow (\delta b \equiv \delta z)\\ \delta N(a,c,z): & a \wedge \neg c \wedge \neg z \wedge \delta a \Rightarrow (\delta c \equiv \delta z)\\ \delta P(b,a,z): & \neg b \wedge a \wedge z \wedge \delta b \Rightarrow (\delta a \equiv \delta z)\\ \delta N(c,a,z): & c \wedge \neg a \wedge \neg z \wedge \delta c \Rightarrow (\delta a \equiv \delta z)\\ \delta P(z,a,b): & \neg z \wedge a \wedge b \wedge \delta z \Rightarrow (\delta a \equiv \delta b)\\ \delta N(z,a,c): & z \wedge \neg a \wedge \neg c \wedge \delta z \Rightarrow (\delta a \equiv \delta c)\end{array}\right\} \ldots\delta\text{Imp}$$

It can be verified that

Theorem 4. $\qquad \delta\text{Pre2} \wedge \text{Imp} \wedge \delta\text{Imp} \Rightarrow \delta\text{Post}$

In the case of the stronger precondition δPre1, it would be permissible to truncate the design after the first four transistors. The four-transistor design of the exclusive *or* circuit has been described in a standard text (Mukherjee p.75) as "...... simply 'tricky', and does not follow from any systematic design method." This claim seems to be no longer true.

Technical Notes

(1) The prevention of short circuits requires that δw be proved for all wires w connected to the gate of any transistor. In most sensibly designed circuits, this will be needed anyway to prove that the outputs are driven.

(2) In certain design styles, a single pass-transistor is allowed to make a driven connection for both 0 and 1; but such transistors may not be connected in series. Our design method could be adapted for this design style; or more simply, a pass transistor may be regarded as a last-minute peep-hole optimisation of the more cumbersome transmission gate.

3. Procedures

In order to distinguish clearly between what must be proved and what may be assumed, we write the former in the left hand column of a tabular display, and the latter in the right hand column. At the top of the table are written the three parts of the specification. For a circuit that is declared as a procedure intended for repeated use with differing arguments, we write at the head of the the name of the procedure, followed by its formal parameters. Above the line in the table we write the specification, split into its four parts: Pre, Post and δPre are written on the right, because they may be assumed; and δPost is written on the left, because it must be proved. A true precondition may be omitted. As an example, Table 1 summarises the design of the familar negation circuit.

NOT (a,z) $\delta z \quad \ldots \delta\text{Post}$	$z = \neg a \ldots \text{Post}$ $\delta a \ldots \delta\text{Pre}$
$\exists z.\ \text{Post}$ $P(a,1,z) : \neg a \Rightarrow (1 \equiv z)$ $N(a,0,z) : a \Rightarrow (z \equiv 0)$ $\delta z \quad$ (in both cases)	$\neg a \wedge z \wedge 1 \wedge \delta a \Rightarrow (\delta 1 \equiv \delta z)$ $a \wedge \neg z \wedge \neg 0 \wedge \delta a \Rightarrow (\delta z \equiv \delta 0)$

Table 1. Negation

In more complex and useful procedures, in addition to input and output wires there will be some local wires to carry signals between components of the circuit; but these cannot be detected or influenced by any environment in which the current is embedded. A designer at discretion is permitted to introduce the name of a new local wire w, by writing

$$\text{Let } w = P,$$

where P is a Boolean formula defining the intended value of w. The explicit declaration of a value P for w is needed, so that we can prove the correctness of the transistors to which w is connected. The choice of P has no relevance to the eventual behaviour of the circuit in operation; its role in design is similar to that of an invariant in the design of a program loop.

Technical Note

If the wire w is connected to the gate of a transistor, then δw should be added to the proof obligations of the designer. In other cases, it is permitted that the wire w may sometimes be indeterminate, and the circuit will still work.

Whenever a procedure is called, its specification (Pre $\wedge$ Post) must be proved in the left hand column of the tabular design; and then its δSpec (δPre $\Rightarrow$ δPost) can be assumed in the right hand column. Thus the call of a procedure defining a complex circuit follows the same rules as those of the primitive transistors from which everything is built. For example, Table 2 describes the design of a simple amplifier, whose logical specification is trivial; it also gives an example of the declaration of a local wire, which is connected to a gate.

AMP(a,z)	
$\delta z \quad \delta\text{Post}$	$z = a \ldots \text{Post}$ $\delta a \ldots \delta\text{Pre}$
$\exists z.\ \text{Post}$ Let $w = \neg a$ NOT(a,w) : $w = \neg a$ NOT(w,z) : $z = \neg w$ $\delta w \wedge \delta z$	 $\delta a \Rightarrow \delta w$ $\delta w \Rightarrow \delta z$

Table 2. Amplifier

As a more substantial example, we derive the design of the familiar NAND circuit (Table 3). The precondition states that the output will be driven when either of its inputs is driven false, even if the other one is undefined! This requires the hardware to compute a non-sequential function of a kind that cannot be implemented in the conventional λ-calculus. In hardware it is easily implemented; later we will see that it is also useful.

Technical Note

I am not sure if I have the right rules for procedure call.

$\text{NAND}(a,b,z)$ $\delta z \quad \ldots\delta\text{Post}$	$z = \neg(a \wedge b) \quad \ldots\text{Post}$ $(\neg a \wedge \delta a) \vee (\neg b \wedge \delta b) \vee (\delta a \wedge \delta b)\ldots\delta\text{Pre}$
$\exists z.\ \text{Post}$	
$P(a,1,z) : \neg a \Rightarrow z$	$\neg z \wedge \delta a \Rightarrow \delta z$
$P(b,1,z) : \neg b \Rightarrow z$	$\neg b \wedge \delta b \Rightarrow \delta z$
Let $\quad w = \neg b$	
$N(b,w,0) : b \Rightarrow (w \equiv 0)$	$b \wedge \delta b \Rightarrow \delta w$
$N(a,z,w) : a \Rightarrow (z \equiv w)$	$a \wedge \delta a \wedge \neg z \Rightarrow (\delta z \equiv \delta w)$
δz	

Table 3. NAND

4. Sequential circuits

In a sequential circuit, the output of each cycle may depend not only on the values of the inputs at the beginning of the same cycle, but also on the values they take at earlier cycles. We therefore need to reinterpret each wire name w as an infinite sequence of signal values; the subscripted variable w_i denotes the value taken by wire w on cycle number i. Similarly δw_i states whether w is driven on cycle i. The constant wires always take the same value, according to the convention

$$0_i = 0,\ 1_i = 1,\ \delta 0_i = \delta 1_i = 1 \qquad\qquad \text{for all } i$$

Two further constant wires $\phi 0$ and $\phi 1$ (phase 0 and phase 1) will be useful as clock signals. They are defined by

$$\phi 0_{2i} = 1,\quad \phi 0_{2i+1} = 0,\quad \delta\phi 0 = 1$$
$$\phi 1_{2i} = 0,\quad \phi 1_{2i+1} = 1,\quad \delta\phi 1 = 1\ .$$

We continue to use familiar Boolean connectives between wire names, on the understanding that they apply componentwise between variables of the same index. For example, we could have defined $\phi 1$ by the single equation

$$\phi 1 = \neg\phi 0,$$

which means

$$\phi 1_i = \neg\phi 0_i \qquad\qquad \text{for all cycles } i.$$

The reason for this convention is that we can preserve unchanged all definitions and design methods used so far; we merely reinterpret the formulae to apply to sequences. Similarly, if P is any forumula, P_i stands for the result of adding a subscript i to all free wire names and constants occurring in P.

To avoid excessive use of subscripts, we introduce a notation w^-, which stands for the previous value of wire w. It is the sequence obtained by shifting w one place up. At cycle $i+1$, it has the same value that w had on cycle i

$$w^-_{i+1} \;\hat{=}\; w_i \qquad\qquad \text{for all } i$$

(Note that if i ranges over natural numbers only, $(w^-)_0$ is left undetermined). We also allow $^-$ to be applied to arbitrary Boolean formulae, as in the trivial theorem

$$\phi 0 = (\neg\phi 0)^-$$

In the design of a sequential circuit, we take advantage of the fact that a physical wire (especially if connected to a transistor gate) can retain as charge the same value that it had on the previous cycle, and that this charge can drive transistor gates. Let Q be any propositional formula describing the conditions under which the designer decides that the wire w is to retain the value it had on the previous cycle. The validity of this decision must be checked by proving

$$Q \Rightarrow (w \equiv w^-) \qquad\qquad \ldots \quad C(w)$$

This ensures that whenever Q is true, the charge on w will not be overridden by conduction through a transistor. The proposition Q serves merely as annotation; proof of $C(w)$ guarantees that it is at least consistent with the behaviour of the circuit; but it has no influence on the circuit design as printed onto silicon, or on circuit behaviour.

Now for each wire name w, we introduce a new variable γw, which means that wire w is properly defined, either by retaining its previously driven charge, or by strong connection on this cycle to power or to ground. So γw is defined by

$$\gamma w = \delta w \vee (Q \wedge \delta w^-) \qquad \ldots \quad \gamma C(w)$$

where Q is the formula declared by $C(w)$ as described in the previous paragraph. $\gamma C(w)$ may be introduced as an assumption (in the right column of a tabular design) whenever $C(w)$ has been proved (in the left column).

Technical Note

The formula $\gamma C(w)$ given above makes retention of charge possible only on wires which on the previous cycle have been strongly connected (δw^-) to power or to ground. This is a safe or conservative design rule. In practice the charge may last for many more cycles; to take advantage of this, the formula γC above can be weakened to

$$\gamma w = \delta w \vee (Q \wedge \gamma w^-) \quad \ldots \gamma C'(w)$$

But charge does not last forever, so the designer has the additional responsibility to show that Q goes false at regular intervals, say every 16 clock cycles. This design style has the disadvantage that it places a lower bound on clock speed; and the clock must not be slowed down or stopped during such a group of consecutive cycles (often called a major cycle).

The reason for retaining charge is to open and close gates of the transistors of the circuit. We therefore need to strengthen the δ-formulae describing the propagation of drive by transistors:

$$\begin{array}{ll} \neg g \wedge s \wedge d \wedge \gamma g \Rightarrow (\delta s \equiv \delta d) & \ldots \gamma P(g,s,d) \\ g \wedge \neg s \wedge \neg d \wedge \gamma g \Rightarrow (\delta s \equiv \delta d) & \ldots \gamma N(g,s,d) \end{array}$$

The distinction between γ and δ is still necessary, because charge can never propagate successfully, even over a strong connection, between the source and drain of a transistor.

Indeed, the very possibility of such a propagation leads to a serious problem known as charge sharing. This occurs when charge on one wire is lost by dissipation to another wire through a transistor whose gate is not driven shut. The simplest way to avoid the problem is to include γw among the proof obligations for all wires w, including local wires not connected to gates.

Technical Note

This does not prevent the charge sharing that may occur before the stability, while the transistors of the circuit are switching. It is therefore a good idea to ensure that signals which open gates do not arrive significantly before signals that close them. Proof of this will require more complex methods, which deal with timing.

A simple example of the use of these techniques is afforded by the design of a precharged NAND circuit (Table 4). The value of the output z is needed only in phase 1; this is specified by

$$\phi 1 \Rightarrow (z \equiv \neg(a \wedge b)). \quad \ldots \text{Post}$$

But this specification does not uniquely determine the value of z. Since all implementations uniquely define their outputs, it will be impossible to find one equivalent to this specification. It is therefore necessary to take an early design decision about the value of z in phase 0:

$$\phi 0 \Rightarrow z$$

The precondition

$$\phi 0 \Rightarrow a \wedge \neg b$$

is needed to assist in the precharging phase. Finally, the γ-spec allows the input to be presented as charge, and similarly the output is given only as charge.

Technical Note

In this example, we cannot prove absence of charge sharing on the wire w. In practice, this does not matter, because w is probably a short wire, which will not drain much charge; whereas z is connected to the gate of a transistor in some other circuit, and therefore stores so much charge that some can be spared for sharing with w. If this form of charge sharing is acceptable, there is no need to prove γw except for wires connected to gates. And in this example, the Precondition can be relaxed to

$$\phi 0 \Rightarrow \neg(a \wedge b)$$

$NANDC(a,b,z)$		$\left.\begin{array}{r}\phi 1 \Rightarrow (z \equiv \neg(a \wedge b)) \\ \phi 0 \Rightarrow z\end{array}\right\}$	... Post
		$(\phi 0 \Rightarrow a \wedge \neg b)$	...Pre
γz	...γPost	$\gamma a \wedge \gamma b$	...γPre
Pre $\Rightarrow \exists z$.Post			
$P(\phi 1, 1, z) : \neg \phi 1 \Rightarrow z$		$\neg \phi 1 \Rightarrow \delta z$	(1)
Let $w = a \wedge \neg b$			
$N(b, w, 0) : b \Rightarrow \neg w$		$b \Rightarrow \delta w$	(2)
$N(a, z, w) : a \Rightarrow (z \equiv w)$		$a \wedge b \Rightarrow (\delta z \equiv \delta w)$	(3)
$C(z) : \phi 1 \wedge z \Rightarrow (z \equiv z^-)$		$\gamma z = \delta z \vee (\phi 1 \wedge z \wedge \delta z^-)$	(4)
$\phi 0 \Rightarrow \delta z$	by (1)		
$\phi 1 \Rightarrow \delta z^-$			
$\phi 1 \wedge \neg(a \wedge b) \Rightarrow \gamma z$	by (4)		
$a \wedge b \Rightarrow \gamma z$	by (2)(3)(4)		
$\phi 0 \vee \phi 1$			
γz	by cases		

Table 4. Precharged NAND

A more substantial example is afforded by the design of an $R-S$ flip-flop (Table 5). Here the output wires y and z always take different values. Their values remain unchanged for as long as both the input wires r and s remain at 1. The precondition states that these never go to 0 simultaneously. But when r goes to 0, y goes to 1; and when s goes to 0, z goes to 1. Thus y and z retain a record of which of r and s most recently took value 0. The precondition states that initially (on cycle numbered 0) one of r and s is 0. This is ensures proper definition and drive for the initial values of y and z.

The design consists of two declarations of capacitance, and two cross-coupled NAND-gates; in these, the input wires are connected only to gates, so the proof of NAND in Table 3 still works when δa and δb in the precondition are replaced by γa and γb. The main novelty of this proof is that it needs induction. Proof of the base step in the left column permits assumption of the induction hypothesis in the right column. The assumption is discharged when the induction step is proved.

Proof of the δ-Spec in this case is quite subtle. Its deductive structure reflects the actual history of signals propagating in the wires. Consider the case

$$r = s = 1 \quad \text{and} \quad y = 0.$$

Then the truth of γg ensures the truth of δz, which (through the other NAND gate) ensures δy as well. A similar chain reaction occurs when $z = 0$. Fortunately one of these two cases always occurs, so both outputs are always driven.

Our method does not seem strong enough to deal with the case when r and s are both 0 and on the next cycle they both of them go to 1. In C-mos hardware, the signals on y and z will (after

$RS(r,s,y,z)$			
		$z \not\equiv y \ldots (1)$	Post
		$r \wedge s \Rightarrow z = z^- \wedge y = y^- \ldots (2)$	Post
		$(\neg r \Rightarrow y) \wedge (\neg s \Rightarrow z) \ldots (3)$	Post
		$r \vee s \ldots (4)$	Pre
		$\neg r_0 \vee \neg s_0 \ldots (5)$	Pre
$\delta y \wedge \delta z$	$\ldots \delta$Post	$\delta r \wedge \delta s \ldots \delta$Pre	
Pre $\Rightarrow \exists y, z.$ Post			
$C(y) : r \wedge s \Rightarrow y = y^-$	by (2)	$\gamma y = \delta y \vee (r \wedge s \wedge \delta y^-) \ldots (6)$	
$C(z) : r \wedge s \Rightarrow z = z'$	by (2)	$\gamma z = \delta z \vee (r \wedge s \wedge \delta z^-) \ldots (7)$	
NAND $(r,z,y) : y = \neg r \vee \neg z$	by (1)(3)	$(\neg r \wedge \delta r) \vee (\neg z \wedge \gamma z) \vee (\delta r \wedge \gamma z) \Rightarrow \delta y \ldots (8)$	
NAND$(y,s,z) : z = \neg y \vee \neg s$	by (1)(3)	$(\neg y \wedge \gamma y) \vee (\neg s \wedge \delta s) \vee (\gamma y \wedge \delta s) \Rightarrow \delta z \ldots (9)$	
$\neg r \Rightarrow \delta y$	by (8) δPre		
$\delta y \Rightarrow \delta z$	by (6) (9) δPre		
$\neg r \Rightarrow \delta z \wedge \delta y$	$\ldots (10)$		
also $\neg s \Rightarrow \delta y \wedge \delta z$	$\ldots (11)$		
$\delta y_0 \wedge \delta z_0$	by (5)	$\delta y^- \wedge \delta z^- \ldots$ hyp	
$r \wedge s \Rightarrow \gamma y \wedge \gamma z$	by (6)(7)		
$\Rightarrow \delta y \wedge \delta z$	by (8)(9)		
$\delta y \wedge \delta z$	by (10)(11)		

Table 5. RS Flip Flop

a short but in principle unpredictable delay) take on different driven values. Which of them takes the value 1 is non-deterministic, and may depend on thermal effects in the wires. This behaviour of C-mos is useful in deciding which of two simultaneous asynchronous interrupt signals should attract the attention of a computer. For design of such asynchronous circuitry, an alternative calculus [Martin] is available.

Acknowledgements

For inspiration, encouragement and helpful comments to Jonathan Bowen, Geoff Brown, Mani Chandy, Paul Gardiner, Mike Gordon, Mohamed Gouda, Roy Jenevein, Geraint Jones, Chris Lengauer, Jay Misra, Ian Page, David Shepherd, David Wheeler. The research of the first author was supported in 1986/87 by the B.R. Inman Chair of Computing Science at the University of Texas at Austin.