Redundant adder for reduced output transitions *

T. Lang

Dept. of Electrical and Computer Eng.
University of California at Irvine
Irvine, CA 92717

E. Musoll and J. Cortadella

Dept. of Computer Architecture
Universitat Politècnica de Catalunya
08071 Barcelona, Spain

Abstract

Flip-flops are one of the most energy-consuming components of digital circuits. In CMOS circuits the energy consumption depends on the signal-transition count. In this paper, alternative designs for arithmetic structures that include carry-save adders are proposed. These new adders present less output activity than the carry-save adder and, therefore, reduce the energy of the flip-flops that are usually placed at their outputs to store the results. It is shown that the best adder depends on the application and a method is developed to choose this adder.

The effectiveness of the approach is illustrated in a sequential multiplier and an accumulator. In the latter, the activity at the flip-flop outputs is reduced by 40% with respect to the use of a carry-save adder, which results in about 15% energy reduction in the flip-flops and 20% overall. This reduction in energy is significantly lower than the reduction in output activity since a flip-flop dissipates energy even when its output does not change; this implies that the method described here should be combined with a reduction of the energy consumed by the flip-flop when its output does not change.

1 Introduction

In the last few years there has been an increasing demand for low-energy digital systems. This demand has triggered many research activities aiming at finding design techniques and circuit structures that consume less energy [2, 11]. In CMOS technologies, most of the energy consumed is proportional to the activity of the signals. Thus, techniques at the logic level are often targeted to reducing this activity [7, 8, 9].

A major part of the activity of data-dominated applications, such as DSP, is concentrated in the data-path of the system in which most data transformations involve arithmetic operations. Many arithmetic structures use redundant representations of the operands to include carry-free adders and benefit from the short delays and the small number of output transitions of this type of adders [4]. The latter is specially important when the output of the adder is loaded into flip-flops, which consume a large amount of energy. For example, for the radix-4 multiplier and accumulator with carry-save adder described in this paper, the energy consumed in the flip-flops (implemented in static CMOS) is 75% of the total. Since flip-flops also consume energy when there are no transitions, the reduction of the transition count has to be combined with techniques such as gated clocks as discussed in [1, 5].

In this paper we focus on reducing the energy of the flip-flops at the outputs of arithmetic structures and we show that a reduction of output activity can be achieved by choosing the best coding for the values of a 2-bit redundant representation. We also show that the best coding depends on the transition probabilities between the adder output values, so that a different coding might be optimal for different applications. Depending on the transition probabilities we identify three codings and design the corresponding adders, which have an area complexity and delay similar to those of a full adder.

We apply the technique to two examples, a radix-4 multiplier and an accumulator, and we conclude that the selection of the optimal adder results in a large reduction of the output activity. The circuits have been implemented in a Sea-of-Gates design style [3] and the energy consumption (at the flip-flops and for the overall application) has been determined using a switch-level simulator [12] based on an RC model. We have observed that the energies of the conventional carry-save and the proposed adders are roughly the same. Moreover, there is a significant energy reduction at the flip-flops, but less than the reduction in output activity. This is due to the energy consumed by flip-flops even when the output does not change.

2 Redundant adder for reduced output transitions

We consider the addition of two operands, one of them in redundant representation and one in conventional representation and produce a redundant result (in the same representation as the operand). This type of adder is typical in the implementation of many operations, such as multiplication, accumulation, and division [4].

The typical radix-2 redundant representations are carry-save, with digit set $\{0, 1, 2\}$, and signed-digit (also called sometimes borrow-save), with digit set $\{-1, 0, 1\}$. We concentrate here on the first of

*This work has been partially supported by CICYT TIC95-0419 and Dept. d'Ensenyament de la Generalitat de Catalunya.

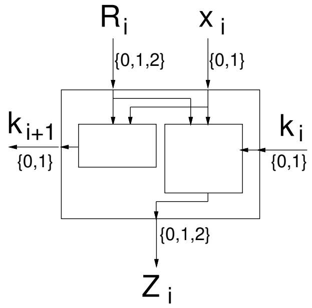


Figure 1: Schematic of the redundant addition algorithm.

these digit sets, although similar results should be possible for the second.

Calling R_i, x_i, and Z_i the i-th digit of the redundant and of the conventional operands and of the result, respectively, the addition algorithm is

$$k_{i+1} = \begin{cases} 1 & \text{if } R_i + x_i \geq 2 \\ 0 & \text{otherwise} \end{cases}$$

$$Z_i = (R_i + x_i) \ mod\ 2 + k_i$$

See Figure 1 for a schematic. Note that k_{i+1} does not depend on k_i and therefore there is no carry propagation.

Our objective is to select a coding for the redundant operand to reduce the average output transition count. This count is affected by the following two factors:

- The transition probabilities between the values of the output digit. This depends on the distribution of the input values, and therefore on the application.
- The coding of the digit on binary variables. Since a digit has three values, a minimum of two bits is required for this coding. We consider only codings with this minimum number of bits.

If we call a and b the two bits representing a digit and $P(a)$ and $P(b)$ the corresponding transition probabilities, we want to minimize

$$P = P(a) + P(b)$$

To determine a coding that produces the minimum we observe the following:

1. To code the three digits it is possible to select a code which produces one bit transition for two of the digit transitions and two bit transitions for the third.
2. The minimum value of P is obtained if the digit transition that produces the two bit transitions is the one that has lowest probability.

digit	(a,b)	(a,b)	(a,b)
0	00	00	00
1	11	01	01
2	10	10	11
	$0 \leftrightarrow 1$ (C01)	$1 \leftrightarrow 2$ (C12)	$0 \leftrightarrow 2$ (C02)

Table 1: One coding of each symmetry with the transition that should have minimal probability.

Adder	Delay (ns)	Area (transistors)
gh	0.68	36
pq	0.66	34
ef	0.58	32
cs	0.66	36

Table 2: Area and delay comparison among the new redundant adders and the carry-save adder (cs).

3. The previous considerations indicate that it is better to use a code that utilizes only three of the four possible combinations of the two bits (that is, one of the combinations is a don't care) than to use all four, as the carry-save coding does.

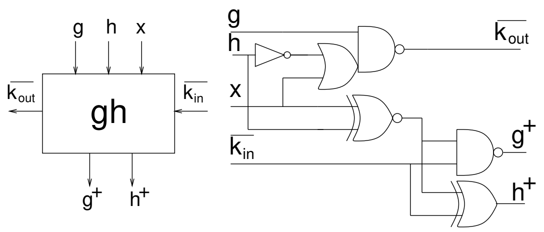
Table 1 presents the possible codings, showing only one of each symmetry class, since the others can be obtained by swapping and/or inverting the bits. We also indicate the transition that should have minimal probability to make the corresponding coding optimal. Henceforth, we will refer to these codings as C01 (for transition $0 \leftrightarrow 1$ producing two bit changes), C02 (for transition $0 \leftrightarrow 2$) and C12 (for transition $1 \leftrightarrow 2$).

To cover all possible applications, we should design three adders, one for each coding class, so that the corresponding adder is used depending on the application. In addition to producing the minimum number of output transitions, the adder should be small, fast and present a low internal energy. We have designed the adders shown in Figure 2 and simulated them using gates from [6]. The delays have been evaluated with the SIS package [10] assuming unit output loads and inputs with drive of one inverter and the same arrival time. As shown in Table 2, the new adders present a similar delay to the carry-save adder and similar or smaller area.

3 Applications

We now illustrate the use of the alternative redundant adders in three applications: a 16-bit sequential radix-4 multiplier, a 16-bit accumulator with uniformly distributed operands and the same accumulator with the operands distributed according to a normal distribution.

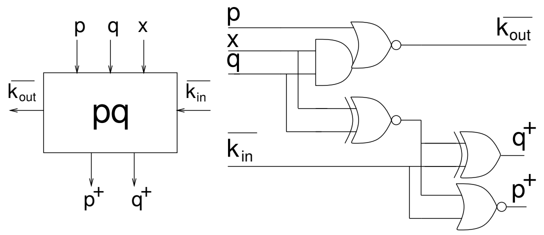
In each case we evaluate the design with carry-save adder and the best redundant adder so that the reduction in output activity is obtained. The ratio of the output activity of the carry-save and



value	gh
0	00
1	11
2	10

$$\begin{aligned} k_{out} &= g(\overline{h}+x) \\ v &= h \oplus x \\ g^+ &= v + k_{in} \\ h^+ &= v \oplus k_{in} \end{aligned}$$

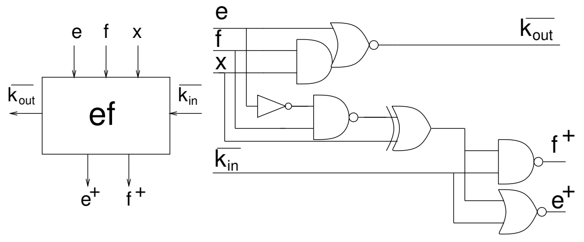
(a) for code C01



value	pq
0	00
1	01
2	10

$$\begin{aligned} k_{out} &= p + qx \\ v &= q \oplus x \\ p^+ &= v k_{in} \\ q^+ &= v \oplus k_{in} \end{aligned}$$

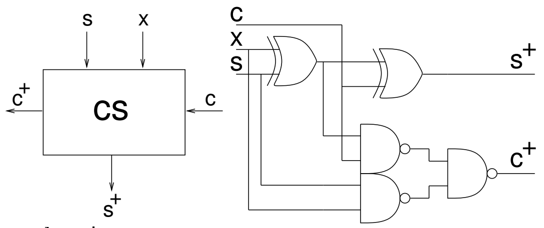
(b) for code C12



value	ef
0	00
1	01
2	11

$$\begin{aligned} k_{out} &= e + fx \\ v &= \overline{e}\, f \oplus x \\ f^+ &= v + k_{in} \\ e^+ &= v k_{in} \end{aligned}$$

(c) for code C02



value	cs
0	00
1	01
	10
2	11

$$\begin{aligned} v &= s \oplus x \\ s^+ &= v \oplus c \\ c^+ &= sx + vc \end{aligned}$$

(d) Carry-save adder

Figure 2: **(a-c)** Proposed adders (**(a)** for code C01; **(b)** for code C12 and **(c)** for code C02) and **(d)** carry-save adder.

the best adder is reported. Moreover, we have implemented the required combinational network and the flip-flops in a Sea-of-Gates design style [3] and report the ratio of the the energies of the flip-flops and of the overall application. Figure 3 shows the implementation used for the D flip-flops of the applications.

For each application, energy has been estimated by means of switch-level simulations [12] of a realization of the circuit. To compute the average, we have performed a suitable number of simulations with inputs obtained from the specific distributions.

3.1 Radix-4 multiplier

We consider a serial-parallel implementation of a radix-4 multiplier with recoding of multiplier digits into {-2,-1,0,1,2} and using redundant addition. For the n-bit multiplicand x and z_j the radix-4 digit of the multiplier, the recurrence is

$$p[j+1] = 2^{-2}(p[j] + 2^n x z_j)$$

with $p[0] = 0$ and $p[n]$ the product. An implementation is shown in Figure 4. We keep only the most-significant half of the product (rounded); to achieve this, guard bits are used (we do not discuss this further since it has a negligible effect on the energy). To have a fast addition, a redundant representation of $p[j]$ is used.

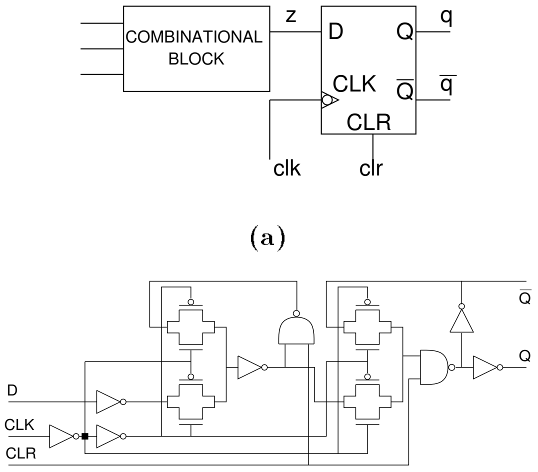
Carry-save conventional implementation

The most typical representations are carry-save and signed digit: the characteristics are similar so we only consider the carry-save case. We do not include the final conversion to non-redundant representation, since this is not necessary in some applications. Moreover, the module for conversion (a carry-propagate adder) operates only at the end of the operation, so that its effect on the overall energy is small.

Although it might be possible to determine analytically the transition probabilities of the output bits (s and c), this is complicated because of the following:

1. The initial value in the registers is 0, so that the carry vector after the first iteration is also 0

2. The radix-4 recoding produces a 0 digit with probability 1/4. Consequently, the probability of value 1 of a bit of xz_j is 0.375, instead of 0.5



(a) **(b)**

Figure 3: **(a)** Trailing-edge D flip-flop and **(b)** a possible implementation [6].

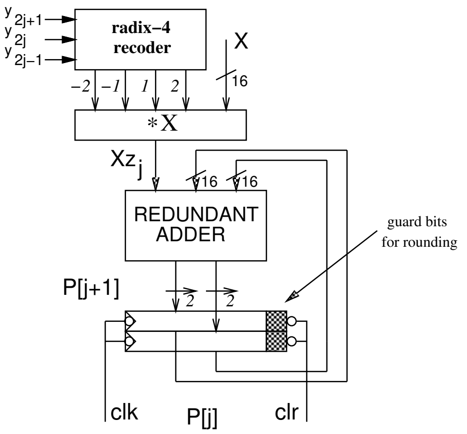


Figure 4: Serial-parallel implementation of a radix-4 multiplier with recoding of the multiplier.

Because of these facts the analysis of the transition probabilities is complex. Therefore, we have measured (with switch-level simulations) these probabilities. We have obtained the following:

- For a bit of s, $P = 0.49$
- For a bit of c, $P = 0.35$

Consequently, for this implementation the transition per digit (two flip-flops) is

$$P(s) + P(c) = 0.84$$

Using the best coding

To determine the best coding we need the transition probabilities between pairs of values of the digits. We have measured

$P(0 \leftrightarrow 1) = 0.33$ $P(1 \leftrightarrow 2) = 0.12$ $P(0 \leftrightarrow 2) = 0.10$

Consequently, the best coding is C02 and we use the *ef* adder. The transition probabilities for outputs e and f have been measured, obtaining:

- For bit e, $P = 0.21$
- For bit f, $P = 0.47$

Now the transition per digit is

$$P(e) + P(f) = 0.68$$

Comparisons

The ratio of the output activity of the carry-save and the best adder is $0.68/0.84 = 0.81$.

Both multiplier designs have been implemented obtaining by switch-level simulations the following ratios for the energy consumed by the flip-flops and for the overall energy of the multiplier: $r_{ffs} = 0.93$ and $r_{overall} = 0.93$.

The reduction in energy is significantly lower than the reduction in output activity because of the energy dissipated by the flip-flops when its output does not change; this implies that larger reductions could be obtained by disabling the clock when the flip-flops must not change. This has been addressed in [5] and the following ratios are obtained: $r_{ffs} = 0.67$ and $r_{overall} = 0.81$.

Radix-16 multiplier

A similar approach could be used to reduce the output transitions for a radix-16 multiplier in which two radix-4 digits are used each cycle and the conventional implementation uses a 4-2 carry-save adder. Using the C02 coding, an adder that performs the addition of a redundant operand and two conventional operands is needed (see Figure 5 for the boolean expressions and a design of this adder). Both radix-16 multiplier designs have also been implemented obtaining the following values:

	P
carry-save representation (s, c)	(0.49,0.34)
(e, f) representation	(0.20,0.50)

and the measured ratios are: $r_{ffs} = 0.92$ and $r_{overall} = 0.94$.

3.2 Accumulator with uniform distribution of operand

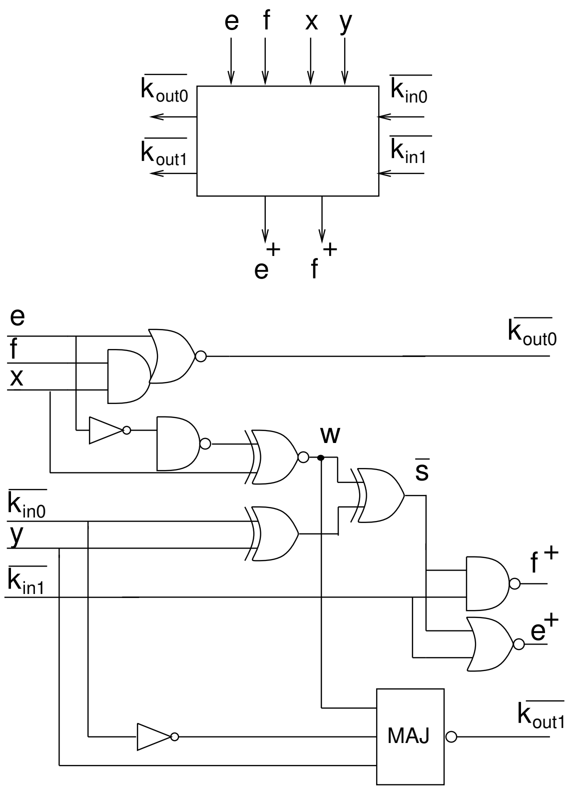
The operation is described by the recurrence:

$$z[j+1] = z[j] + y[j]$$

and the value of $y[j]$ is selected from a uniform distribution in the range 0 to $2^{16} - 1$. An implementation using redundant addition is shown in Figure 6.

Carry-save conventional implementation

The following values have been measured for the transition probabilities:



$$
\begin{aligned}
k_{out0} &= e + fx \\
w &= \overline{e}f \oplus x \\
s &= w \oplus (k_{in0} \oplus y) \\
k_{out1} &= Maj(w, y, k_{in0}) \\
e^{+} &= k_{in1} s \\
f^{+} &= k_{in1} + s
\end{aligned}
$$

Figure 5: *ef* adder used in the radix-16 multiplier.

- For bit s, $P = 0.50$
- For bit c, $P = 0.66$

resulting in the transition per digit

$$P(s) + P(c) = 1.16$$

Using the best coding

This application has the following digit transition probabilities:

$P(0 \leftrightarrow 1) = 0.28$ $P(1 \leftrightarrow 2) = 0.22$ $P(0 \leftrightarrow 2) = 0.16$

Consequently, the best coding is C02 and we use the *ef* adder. The transition probabilities for outputs e and f have been measured, obtaining:

- For bit e, $P = 0.37$
- For bit f, $P = 0.43$

resulting in the transition per digit

$$P(e) + P(f) = 0.80$$

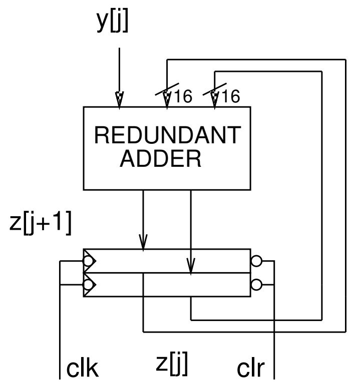


Figure 6: Accumulator with redundant addition.

Comparisons

The ratio of the output activity of the carry-save and the best adder is $0.80/1.16 = 0.70$.

Both accumulator designs have been implemented and the following ratios for the energy consumed by the flip-flops and for the overall energy for the accumulator have been obtained: $r_{ffs} = 0.88$ and $r_{overall} = 0.83$.

3.3 Accumulator with operand with normal distribution

To illustrate that the distribution of the operands has an effect on the best adder, we now consider a 16-bit accumulator in which operands are signed, represented in the 2's complement system, and operands with small module are more frequent than those with large module. As a specific distribution we consider a normal distribution with standard deviation of 100.

Carry-save conventional implementation

The following values have been measured for the transition probabilities:

- For bit s, $P = 0.50$
- For bit c, $P = 0.60$

resulting in the following transition per digit:

$$P(s) + P(c) = 1.1$$

Using the best coding

In the previous examples, all digits of the adder perform in a similar manner. However, because of the distribution, this is not the case now. Consequently, we have measured the transition probabilities per digit (shown in Table 3) and conclude that for digits 0 to 7 and 13 to 15 the best adder is the *ef* adder whereas for digits 8 to 12 the best is the *gh* adder. Since the communication between digits of the adder is the variable k_i, there is no problem in having different bits with different codings (this

transition	digit 0	digit 1	digit 2	digit 3	digit 4	digit 5	digit 6	digit 7
$0 \leftrightarrow 1$	0.51	0.38	0.32	0.28	0.27	0.26	0.26	0.26
$0 \leftrightarrow 2$	0.00	0.12	0.15	0.16	0.16	0.17	0.16	0.17
$1 \leftrightarrow 2$	0.00	0.13	0.18	0.23	0.24	0.24	0.24	0.23
transition	digit 8	digit 9	digit 10	digit 11	digit 12	digit 13	dig it 14	digit 15
$0 \leftrightarrow 1$	0.17	0.13	0.12	0.12	0.11	0.11	0.10	0.05
$0 \leftrightarrow 2$	0.20	0.21	0.22	0.24	0.22	0.14	0.05	0.02
$1 \leftrightarrow 2$	0.17	0.13	0.12	0.13	0.16	0.15	0.10	0.05

Table 3: Probability of transition between digit values for each digit (0 is LSD).

would not be the case for a multiplier, for example, because of the shifting).

The average transition probabilities are

	$0 \leftrightarrow 1$	$0 \leftrightarrow 2$	$1 \leftrightarrow 2$
for digits 0-7 and 13-15	0.25	0.12	0.16
for digits 8-12	0.13	0.22	0.14

and the following values have been measured for the transition probabilities:

	P
for bits (e, f)	(0.27,0.35)
for bits (g, h)	(0.35,0.29)

Therefore, the average transition per digit is

$$\frac{11\,(P(e)+P(f))+5\,(P(g)+P(h))}{16} = 0.63$$

Comparisons

The ratio of the output activity of the carry-save and the best adder is $0.63/1.1 = 0.57$.

Both accumulator designs have been implemented obtaining the following ratios for the energy consumed by the flip-flops and for the overall energy: $r_{ffs} = 0.83$ and $r_{overall} = 0.79$.

4 Conclusions

The energy consumption of flip-flops accounts for a large part of the total consumption of digital circuits. In this paper we have concentrated on reducing the activity at the inputs of the flip-flops to decrease their energy consumption. This technique should be combined with a reduction of the energy when the flip-flop outputs do not have to change to obtain larger energy savings.

New coding strategies for redundant adders have been explored. Rather than minimizing area or delay, codings for energy minimization have been sought.

We believe that this is only an initial effort towards this direction. So far, no much effort has been invested in studying arithmetic circuits from the perspective of low energy. We foresee novel techniques to appear in the future for a wider range of arithmetic circuits.

References

[1] L. Benini and G. De Micheli. Transformation and synthesis of FSMs for low-power gated-clock implementation. In *Int. Symp. on Low Power Design*, pages 21–26, April 1995.

[2] A. Chandrakasan, S. Sheng, and R. Brodersen. Low power CMOS digital design. *IEEE Trans. on SSC*, 27(4):473–483, April 1992.

[3] P. Groeneveld and P. Stravens. Ocean: The Sea-of-Gates design system. Technical report, Delft Univ. of Tech., 1993.

[4] I. Koren. *Computer Arithmetic Algorithms*. Prentice-Hall, 1993.

[5] T. Lang, E. Musoll, and J. Cortadella. Reducing energy consumption of flip-flops. Technical Report UPC-DAC-1996-4, UPC/DAC, February 1996.

[6] LSI LOGIC. *LCA500K Preliminary Design Manual*, November 1994.

[7] V.G. Moshnyaga and K. Tamary. A comparative study of switching activity reduction techniques for the design of low-power multipliers. In *Proc. Int. Symp. on Circuits and Syst.*, pages 1560–1563, April 1995.

[8] R. Murgai, R.K. Brayton, and A. Sangiovanni-Vincentelli. Decomposition of logic functions for minimum transition activity. In *Proc. Int. Workshop on Low Power Design*, pages 33–38, April 1994.

[9] K. Roy and S.C. Prasad. Circuit activity based logic synthesis for low power reliable operations. *IEEE Trans. on VLSI Syst.*, 1(4):503–513, December 1993.

[10] E.M. Sentovich, K.J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P.R. Stephan, R.K. Brayton, and A. Sangiovanni-Vincentelli. SIS: A system for sequential circuit synthesis. Technical Report Memorandum No. UCB/ERL M92/41, University of California, Berkeley, 1992.

[11] D. Singh, J.M. Rabaey, M. Pedram, F. Catthoor, S. Rajgopal, N. Sehgal, and T.J. Mozdzen. Power conscious CAD tools and methodologies: A perspective. *Proceedings of the IEEE*, 83(4):570–594, April 1995.

[12] A.J. van Gerenden. SLS: An efficient switch-level timing simulator using min-max voltage waveforms. In *Proc. VLSI 89 Conf.*, pages 79–88, August 1989.