Low-Power Design by Hazard Filtering

Vishwani D. Agrawal
Bell Labs, Lucent Technologies
Murray Hill, NJ 07974
va@research.bell-labs.com

Abstract – *Before signals of a digital circuit reach steady state, gates can have multiple transitions. Since the power is dissipated in a CMOS circuit mainly due to transitions, the extra transitions increase power consumption. These transitions are the hazard pulses generated by logic gates when signals arrive by paths of varying delays. The maximum width of a hazard pulse produced by a gate is the maximum difference between the delays of incident paths, which is generally much smaller than the clock period. We propose suppression of hazard pulses by increasing the delays of gates where hazards could have been generated. Thus, a hazard filtering gate has a delay which is at least as much as the differential delay of its input paths. We give examples to illustrate the novel technique and also indicate that the overall reduction in the circuit speed may not be too much with proper sizing of transistors, while there can be a significant reduction in power consumption.*

1 Introduction

Low power design of digital circuits is an important aspect of today's mobile electronic systems. Power dissipation in a CMOS circuit consists of four components [1, 2, 3]: (1) functional switching, (2) hazard or glitches, (3) short circuit and (4) leakage. Functional switching power is the *essential* component as it corresponds to the desired logic states. Hazard or glitch power is consumed by transients before a steady logic state is achieved. In general, combinational logic produces such transients and design techniques often eliminate their effect from the overall system function. Thus, hazards consume about 20-40% of the total power even though they do not effect the operation of the system ([1], p. 493). For a 16×16 bit multiplier with a logic depth of 30, hazards are found to consume as much as 67% of the total power ([3], p. 45). Elimination of these hazards is the topic of this paper.

Short circuit power is consumed at the time of actual switching of transistors of a logic gate. This happens due to a momentary short circuit path produced between the power supply and ground. In properly designed circuits this component is negligible. The leakage power is due to the very small current that continues to flow after a CMOS gate has reached steady state. It depends on the open circuit resistance of transistors that are in the open state. Unless some transistors are faulty, the leakage power is several orders smaller in magnitude than all other components of power.

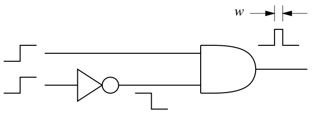
During charging and discharging of gate capacitances, power is consumed in the short circuit resistances of transistors. Since heat dissipation is proportional to the square of the voltage, low voltage operation is one of the most effective methods of reducing power. Any reduction in the operating speed can be compensated by pipelining or parallelizing the computation, which increases the hardware (and power dissipation) only in inverse proportion of the voltage. This technique is discussed in detail in the recent literature [1, 2, 3, 4].

Another method, *adiabatic computing*, dynamically varies power supply. It involves increased delay and additional hardware. However, the technique has been effectively implemented [5]. Both, low voltage and adiabatic computing methods, reduce the power consumed by all signal transitions (functional or hazard). A low power design would still consume about 20-40% of the total power in hazards. The number of hazard-related transitions grows as $O(N^2)$, N being the logic depth ([2], pp. 80-83).

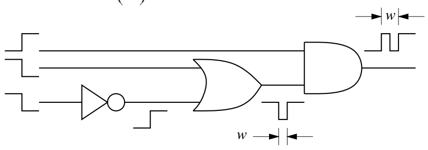
Special algorithms and logic design methods have been reported for low power systems. Some earlier multiplier architectures increased the computing speed by reducing logic activity [6, 7]. They reduce functional activity as well as hazards [3, 8]. A method of eliminating hazards is to balance delays of multipath signals arriving at a gate, either by using a tree-like logic structure or by inserting delay buffers in small delay paths. The balanced delay technique is sensitive to delay tolerances. It also adds power dissipation due to delay buffers. The *retiming* method, in which flip-flops are moved without changing the function of the circuit, is also used for hazard reduction ([3], p. 148). Flip-flops are moved to nodes with high hazard activity. However, the number of combinational nodes with potential hazards is typically much larger than the number of flip-flops. Also, the permitted movement of flip-flops is restricted by retiming rules. Another method is to use self-timed design in which a combinational element would not compute until all inputs have stabilized. This requires additional circuitry to generate the timing or *enable* signals for various stages of logic ([1], pp. 493-496), [9, 10, 11]. The added circuitry reduces the power saving.

2 Hazards

In the circuit of Fig. 1(a), each gate has 1 unit of delay and the input changes occur simultaneously. In general, gates can have any delays and input changes can be



(a) Static hazard



(b) Dynamic hazard

Figure 1: Hazard generation in logic circuits

staggered by assigning delays to primary inputs. In Fig. 1(a), the output returns to the logic 1 state after completion of a transient pulse of width w. This pulse is called a *static hazard.* Its width w equals the delay of the inverter, 1 unit of time in this case. In Fig. 1(b), the OR gate produces a static hazard of 1 unit. The output of the AND gate has a logic value change from 0 to 1. The transient consists of three edges, two rising and one falling. This is a *dynamic hazard* with a combined width (w) of 2 units.

The number of edges in transients at the output of a gate may equal the number of paths arriving at its inputs. The position of each edge in time corresponds to the delay of a path. Suppose, a gate has k inputs. Each input can have several arriving paths. Let Δ_{ip} be the delay of pth path arriving at input i. Then, the total width of hazard at the output of the gate is given by

$$w = \max_{p,q}(\Delta_{ip} - \Delta_{jq}),\ \ for\ all\ i,j = 1,2,...k,\ i \neq j \quad (1)$$

The delay w is defined as the *differential path delay* for the gate. For a single input gate, like inverter, w is 0 and the gate will never generate a hazard. It may just pass the hazard to its output if one arrives at the input. A *hazard producing gate* is a gate which can generate a hazard at its output when there is no hazard at any of its inputs. Two necessary conditions are: (1) gate should have two or more inputs and (2) the differential path delay (w) should be greater than 0.

The meaningful outputs of a logic circuit are the steady state signals. Synchronous systems use a clock signal to sample the output as shown in Fig. 2. Input changes and output sampling are synchronized with a periodic clock edge, whose period is chosen greater than the maximum delay of any path in the logic. The delay of a signal has two components: pure and inertial delays [12].

Pure delay simply shifts the waveform in time. It is caused by gate interconnects acting like transmission lines. Thus a narrow hazard pulse may be delayed by a large pure delay without any change in shape. *Inertial delay* is due to the finite switching speed of transistors. It is modeled as the time of charge or discharge for the output capacitance of a gate through the resistance of transistors in the current path. This delay can severely distort a hazard

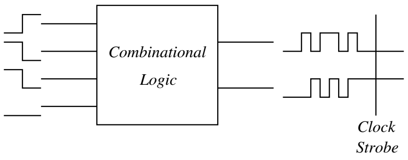


Figure 2: Hazard elimination by clocked sampling

pulse. For example, a narrow pulse at the input of a gate can be reduced in height and width, or even completely eliminated, at the output. The differential delay of a gate contains both components of delay.

Asynchronous systems contain no clock and hazards must be eliminated by special circuit design methods [13]. Of interest here is a technique of filtering hazards by adding inertial delays to outputs of a logic network. *Inertial delay* (or simply delay in the present discussion) is the time a device takes to switch the output after the cause for that change has occurred at its input. If the output delay is made greater than the width of hazards as given by (1), then the hazards will be filtered out. In an asynchronous system, we eliminate hazards only at the output of the combinational network. However, hazards on internal signals also consume power and should be eliminated. We will, therefore, filter a hazard as soon as it is generated [14]. Hazards can be prevented by using *precharge (domino) logic* ([3], p. 45). In that technology, signals change only once for each precharge. However, the precharge power more than compensates for any saving due to hazard elimination. The present technique is targeted for the static CMOS technology.

3 Energy Dissipation due to a Hazard

The energy consumed in a CMOS circuit due to a hazard pulse can be analyzed by examining the inverter circuit of Fig. 3. A pulse of width w is being transmitted through the inverter. Fig. 3(b) gives a two transistor implementation of the inverter. P is a PMOS transistor which shorts the supply V_{DD} to the output when the *gate* input (shown with a circle) is *low*. N, the NMOS transistor, shorts the output to ground when its *gate* input is *high*. Suppose, the short-circuit impedance of each transistor is R and the total capacitance of the output node is C. Figs. 3 (a) and (b) assume ideal transistor switches with $R = 0$.

In Fig. 3(c), transistor switches have non-zero resistances. This model depicts any CMOS gate where the single PMOS and NMOS transistors would be replaced by clusters of transistors of each type. Before the falling edge of the input pulse, N is shorted and P is open. We assume that the open-circuit resistance of a transistor is infinitely high. Thus no current flows from the supply and the output is *grounded.* C has no electric charge. At the falling edge ($t = 0$), P is shorted and N opened, causing a charging current $i(t)$ through the series path of R and C:

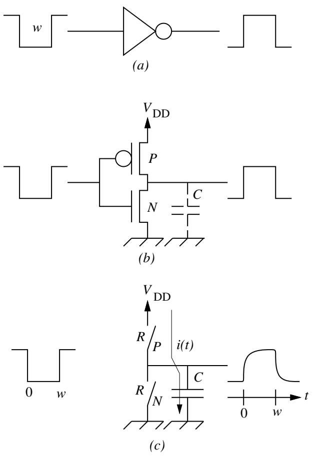


Figure 3: Energy dissipation in a CMOS circuit

$$i(t) = \frac{V_{DD}}{R} e^{-\frac{t}{RC}} \tag{2}$$

The charging of C raises the output voltage until the time $t = w$ when the rising edge occurs at the input. The energy drawn from the supply is [2]:

$$E(w) = \int_0^w V_{DD} i(t) dt = CV_{DD}^2 (1 - e^{-\frac{w}{RC}}) \tag{3}$$

Half of this energy is dissipated as heat in the resistance R. The other half is stored in C. *Thus, the energy dissipated by a gate is proportional to the number of signal transitions at its output.* If the pulse width w is much greater than the *time constant* RC, the output voltage rises to V_{DD} causing maximum energy consumption:

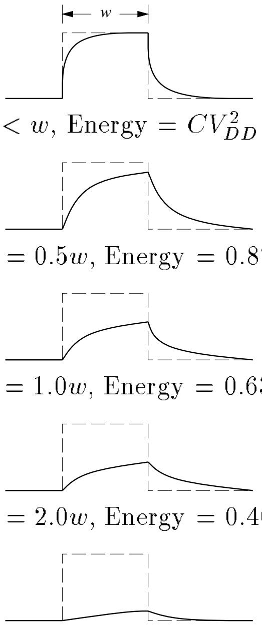
$$E_{max} = CV_{DD}^2, \ w \ >> \ RC \tag{4}$$

The rising edge occurs at $t = w$ and opens P while shorting N. Now the capacitor discharges and the stored energy is dissipated in the resistance R of N. Fig. 4 shows several cases of energy dissipation as the time constant RC is increased with respect to the pulse width w. When RC is about four times the pulse width, the output pulse is almost completely eliminated, saving 78% of energy.

The RC time constant in the above analysis is a measure of the switching delay of a CMOS gate. C consists of the parasitic capacitances of the gate structure and routing. R is the short circuit resistance of transistors and can be controlled by changing the width and length of channels in MOS devices. Thus, the delay of a gate can be increased by reducing the channel widths of transistors. Such variations are generally known as *transistor sizing* and are used for the optimization of chip area and timing [15].

4 Increasing Delay for Hazard Filtering

We start with given delays for all gates and interconnections in an initial version of the circuit. These are



(a) $RC << w$, Energy $= CV_{DD}^2 = E_{max}$

(b) $RC = 0.5w$, Energy $= 0.87E_{max}$

(c) $RC = 1.0w$, Energy $= 0.63E_{max}$

(d) $RC = 2.0w$, Energy $= 0.40E_{max}$

(e) $RC = 4.0w$, Energy $= 0.22E_{max}$

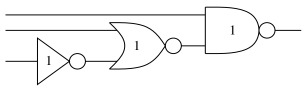
Figure 4: Effect of gate delay on energy dissipation

generally computed for simulation using a variety of methods discussed in the literature ([16], Chapter 4). For accurate data, one may use an initial layout of the circuit from which routing capacitances can be extracted. If time skews are specified for primary input changes, then the inputs can be assigned delays. Otherwise, all inputs are assumed to have zero delay. In general, gates may have different delays for rising and falling signals. For such a case we will consider two logical paths for each physical path. One of these paths will have the delay that a rising transition takes to propagate, and the other path has the delay of the falling transition.

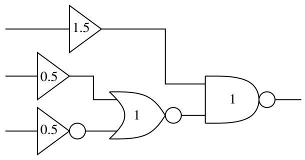
Differential path delay (w) is computed for each gate according to Equation (1). All gates with single input will have $w = 0$. Their delays can be optionally reduced to smallest permissible value. Once delays are changed, differential path delays are recomputed. Adjustment of delays for hazard suppression is done from inputs to outputs. If the delay of gate g is $d(g)$ and its differential path delay is $w(g)$, then we require,

$$d(g) = \alpha \times w(g), \ \alpha \ \geq \ 1 \tag{5}$$

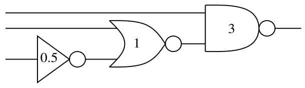
The ratio α is chosen any number equal to or greater than 1. The procedure of differential path delay computation and delay increase is iteratively applied. In one iteration many gates, whose delays do not affect each other's paths, can be simultaneously adjusted. Since the delay of a gate depends on the RC time constant, either R, or C, or both can be changed. The resistance R depends on the channel width and length of transistors and is easy to adjust. Adjustment of C is generally difficult as it is layout dependent.



(a) Original circuit



(b) Balanced delay design



(c) Hazard filter design

Figure 5: Examples of balanced delay and hazard filtering

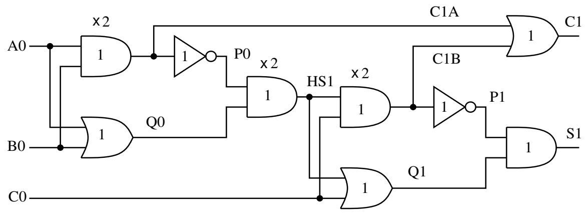
Fig. 5 illustrates the difference between our method and that of balancing delays. The original circuit of Fig. 5(a) has three gates, each with 1 unit of delay. To balance path delays, in Fig. 5(b) we have reduced the delay of the inverter to 0.5 unit and inserted two buffers with delays of 0.5 and 1.5 units. Maximum delay of the circuit is now 2.5 units. Our design, shown in Fig. 5(c), also has the inverter delay reduced to 0.5 unit. This can be done since the differential path delay of inverter is 0. Using $\alpha = 2$, we leave the OR gate unchanged and increase the delay of the NAND gate to 3 units. The overall delay is now 4.5 units. Apparently, the application must demand power reduction to be more critical than the speed. The balanced delay design has two buffers that add to power consumption.

5 A Simulated Example

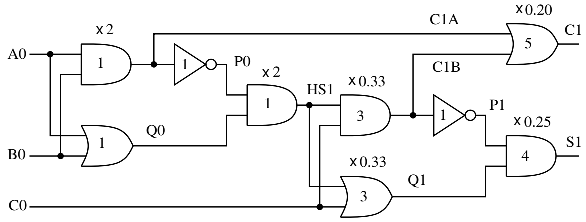
We simulated the full-adder circuit of Fig. 6(a) using the *Spice* simulator [17]. All gates are implemented in 1 micron CMOS technology. AND and OR gates are implemented as NAND and NOR, respectively, followed by inverters. To obtain a uniform delay for all gates, the inverters in the AND gates with double fanout have their transistor sizes doubled. In Fig. 6(a), all gates are shown with 1 unit delay (about 2ns).

The result of Spice simulation for three input transitions, applied at intervals of 30ns, is given in Fig. 7. No hazard is produced at HS1 because the multipath differential delay (w) at this gate is 1 unit, which equals its own delay. Thus, the pulse of unit width that could have been produced around 30ns, has been filtered. At 60ns, there are hazards at Q1 ($w = 3$ units) and S1 ($w = 4$ units). At 90ns, there are hazards at C1B ($w = 3$ units), P1 (propagated from C1B) and S1. The applied inputs do not produce any hazard at C1 ($w = 3$ units).

In Fig. 6(b), the full-adder is modified according to the procedure of Section 4. Assuming $\alpha = 1$, the delay of HS1 is not changed. Since the differential delay (w) at C1B is 3 units, the delay of that AND gate is increased to 3 units. This is done by sizing the transistors of the



(a) nominal delays



(b) delays increased for hazard filtering

Figure 6: Full-adder circuits used in Spice simulation

internal NAND to $\times 0.33$ and leaving the output inverter as $\times 2$ for driving the double fanout. Transistors in the NOR of Q1 are sized $\times 0.33$ since $w = 3$. For the NOR of C1, $w = 5$, and it is sized as $\times 0.2$. The output S1 has $w = 4$ and the NAND part of that AND gate is sized as $\times 0.25$. In the waveforms, obtained from Spice simulation, all hazards at 60ns and 90ns have been filtered out. An estimate of power consumption can be obtained from transitions. Ignoring the transitions on primary inputs (A0, B0 and C0), the circuit of Fig. 6(a) produces 24 transitions (Fig. 7). The circuit of Fig. 6(b) has only 14 transitions in Fig. 8. This 42% reduction in activity should give rise to a proportionate reduction in power consumption.

Fig. 9 shows the power supply current I_{DD} for two circuits. We notice that even at 30ns, where the two circuits have the same number of transitions, there is some reduction in current. The reduction is more significant at 60ns and at 90ns where hazards have been eliminated. Waveforms include short-circuit and quiescent currents.

6 Conclusion

We see two advantages of the present method: (1) The system need not generate evaluation signals (delayed clocks) requiring critical hardware which may impact the yield [10, 11]. (2) Manufacturing variations of delays, which affect the balanced delay method ([1], p. 493), can be accounted for by using tolerances in computing differential path delays. The technique adds delays in a graded fashion. As the signal propagates toward outputs, it encounters slower gates. The basic idea is to suppress hazards just as they are generated and before they have consumed any energy. Thus, single input gates like inverters and buffers can be made faster whenever possible. The cost of energy saving is the reduced speed. With smaller depth, the overall speed reduction will be less. Similarly, other methods of logic design, which eliminate certain types of hazards [13], will reduce the delay penalty. In general,

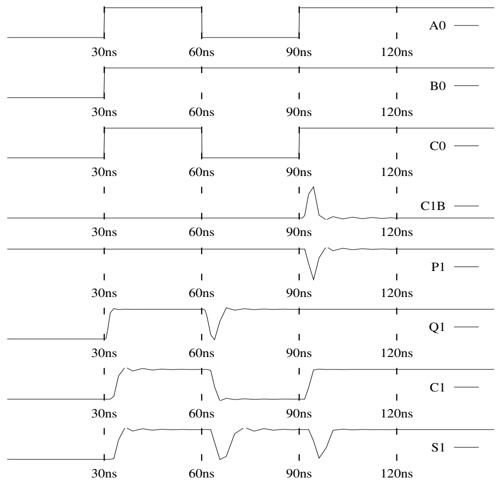


Figure 7: Spice simulation of nominal delay full-adder

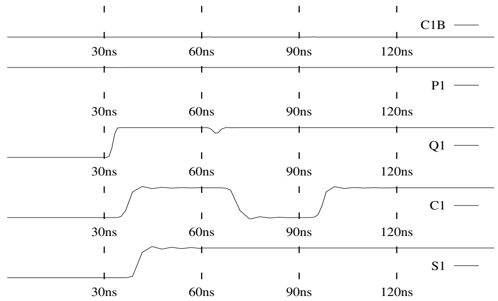


Figure 8: Spice simulation of full-adder with hazard-filter

the short-circuit power is small [1, 2, 3]. However, the influence of the increased switching time of transistors requires a careful study. The present method can be used in combination with the method of power reduction by voltage scaling [1, 2]. Our technique is suitable for both synchronous and asynchronous systems. In a conventional asynchronous circuit, one filters hazards only at the output of the combinational logic. That leaves power-consuming hazards on internal signals. Another advantage of hazard-free logic is in delay testing with *non-robust* tests [18, 19]. This is beneficial in self-test for delay faults [20]. A possibility, not explored in this paper, is the combination of balanced delay and hazard filtering techniques. We may first minimize the delay unbalance without adding any buffers and only by adjusting delays of gates. The circuit will then have reduced duration of hazards, which can be filtered by adding delays to reconvergent gates.

Acknowledgment: The author thanks M. Gharaybeh, R. Ramadoss and S. Sapatnekar for their help.

References

[1] A. Bellaouar and M. I. Elmasry, *Low-Power Digital VLSI Design: Circuits and Systems.* Boston: Kluwer Academic Publishers, 1995.

[2] A. P. Chandrakasan and R. W. Brodersen, *Low Power Digital CMOS Design.* Boston: Kluwer Academic Publishers, 1995.

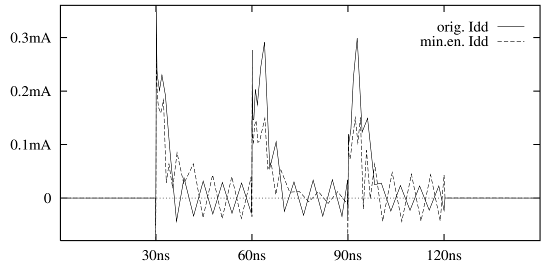


Figure 9: Transient power supply current computed by Spice simulator for the full-adder circuit with nominal delays (solid line) and with increased delays for hazard filtering (dashed line)

[3] J. M. Rabaey and M. Pedram, *Low Power Design Methodologies.* Boston: Kluwer Academic Publishers, 1995.

[4] J. Monteiro and S. Devadas, *Computer-Aided Design Techniques for Low Power Sequential Logic Circuits.* Boston: Kluwer Academic Publishers, 1997.

[5] T. Gabara, "Pulsed Low Power CMOS," *International Jour. High Speed Electronics and Syst.*, vol. 5, no. 2, pp. 159–177, 1994.

[6] L. Dadda, "Some Schemes for Parallel Multipliers," *Alta Frequenza*, vol. 34, pp. 349–356, 1994.

[7] C. S. Wallace, "A Suggestion for a Fast Multiplier," *IEEE Trans. Electronic Comput.*, vol. EC-13, pp. 14–17, 1964.

[8] M. Pedram, "Power Estimation and Optimization at the Logic Level," *International Jour. High Speed Electronics and Syst.*, vol. 5, no. 2, pp. 179–202, 1994.

[9] C. Lemonds and S. S. Shetti, "A Low Power 16 by 16 Multiplier Using Transition Reduction Circuitry," in *Proc. International Workshop on Low Power Design*, pp. 139–142, 1994.

[10] D. L. Raatz and G. E. Sobelman, "Digital Signal Processor with Delayed-Evaluation Array Multipliers and Low-Power Memory Addressing," *United States Patent No. 5,333,119*, July 1994.

[11] G. E. Sobelman and D. L. Raatz, "Low-Power Multiplier Design Using Delayed Evaluation," in *Proc. International Symp. Circ. and Syst.*, 1995.

[12] S. H. Unger, *The Essence of Logic Circuits.* Englewood Cliffs, NJ: Prentice Hall, 1989.

[13] S. H. Unger, *Asynchronous Sequential Switching Circuits.* New York: Wiley-Interscience, 1969.

[14] V. D. Agrawal, "Low-Power Design by Hazard Filtering," in *Proc. 10th International Conf. VLSI Design*, pp. 193–197, 1997.

[15] J. P. Fishburn and A. E. Dunlop, "TILOS: A Posynomial Programming Approach to Transistor Sizing," in *Proc. International Conf. CAD*, pp. 326–328, 1985.

[16] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design, A Systems Approach.* Reading, MA: Addison-Wesley, 1985.

[17] *Spice 3f.4.* Berkeley, CA: University of California, Dept. of EE and CS, July 1993.

[18] M. A. Gharaybeh, M. L. Bushnell, and V. D. Agrawal, "Classification and Test Generation for Path Delay Faults using Single Stuck-Fault Tests," in *Proc. International Test Conf.*, pp. 139–148, 1995.

[19] C. J. Lin and S. M. Reddy, "On Delay Fault Testing in Logic Circuits," *IEEE Trans. CAD*, vol. 6, pp. 694–703, September 1987.

[20] I. P. Shaik and M. L. Bushnell, "A Graph Approach to DFT Hardware Placement for Robust Delay Fault BIST," in *Proc. 8th International Conf. VLSI Design*, pp. 177–182, 1995.