GATE TRIGGERING: A NEW FRAMEWORK FOR MINIMIZING GLITCH POWER DISSIPATION IN STATIC CMOS ICs AND ITS ILP-BASED OPTIMIZATION[*]

NIHAR R. MAHAPATRA RAJAGOPALAN JANAKIRAMAN
{mahapatr, rj1}@cse.buffalo.edu
Dept. of Computer Sc. & Eng., State University of New York at Buffalo, Buffalo, NY 14260, USA

Submission Category: Regular Paper in *Solid State Circuits* Track

Abstract

Glitches are an important source of power dissipation in static CMOS ICs that can contribute to as much as 70% of total power dissipation in certain cases (e.g., arithmetic modules). Although research into various aspects of glitch power dissipation has been undertaken in the past, most approaches to addressing it are ad hoc and limited in their applicability. In this paper, we propose a new framework, gate triggering, for systematically minimizing glitch power dissipation in static CMOS ICs. The framework is based on the idea that glitches can be effectively minimized by triggering logic evaluation at a gate only when all of its inputs have stabilized. For this purpose, to every potentially glitchy gate is added a small amount of control logic, which, when enabled, triggers logic evaluation at the gate. A clocked delay chain is employed to generate enable signals at the proper times for all gates to be triggered. We present an integer linear programming (ILP) formulation to minimize the overheads (viz. delay element, control logic, and extra wiring) of our approach subject to a critical-path delay constraint. Application of the new approach to test circuits (such as ripple carry adder and array multiplier) in 1.2μ *technology yields 95% or more elimination of glitch power dissipation with negligible area and timing overheads after optimization. An added advantage of the approach is that short-circuit power dissipation at all triggered gates is also minimized—short-circuit power dissipation in current standard-cell based designs can exceed 50% of the total power dissipation.*

1 Introduction

Increasing levels of device integration, die size, and operating frequency, combined with a burgeoning portable computing and communications market, have made power dissipation a major concern in VLSI design [12, 13]. Among VLSI technologies, digital CMOS is by far the dominant one and consumes relatively less power. Complementary static CMOS is a popular digital CMOS logic style that is being increasingly employed because of its robustness, which lends itself to design automation, and because of its amenability to voltage scaling for low power [13]. In this logic style, power is primarily dissipated during logic transitions when gate load capacitances charge and discharge.

While some logic transitions are necessary and are dictated by circuit functionality, others, such as glitches, are not. *Glitches* are *spurious transitions* that occur before a gate output reaches a stable value and are caused by unequal propagation delays of input signals to the gate. This is depicted in the synchronous sequential circuit, a popular and structured design style [11], of Fig. 1(c). Also, evident in Fig. 1(c) is that glitches multiply as they propagate through the combinational logic block. Glitch power is typically significant and can be as high as 70% of total power dissipation in some cases [12]. As we go into deeper submicron technologies, interconnect delays become more predominant, which leads to differential delays and more glitching [16].

Glitch minimization is important not only for low power, but also because of other reasons. Power estimation is a difficult problem because glitch power dissipation is significant and is hard to estimate accurately [11, 12]. Thus minimizing it will improve the accuracy of power estimation. Asynchronous systems need to be glitch-free to operate correctly [17], and so also does the clock-generation circuitry in a synchronous system [4]. Finally, glitches are also important to minimize in high-performance digital-to-analog converters [19].

Short-circuit power dissipation, which occurs because of the DC current that flows from V_{DD} to V_{SS} during switching when both N pull-down and P pull-up conduct simultaneously, is another source of concern. This is especially so in very high-performance circuits and in standard-cell based semicustom design, where it can be as high as 50% of total power dissipation [12, 13]. However, it should be noted that short-circuit currents virtually disappear when $V_{DD} \sim V_{Tn} + V_{Tp}$, which is somewhat true for digital signal processors, but not so for microprocessors, where V_{DD}'s are relatively higher.

In this paper, we present a new framework called *gate triggering* for minimizing glitch power dissipation in complementary static CMOS ICs which we discuss in the next section. An added advantage of our approach is that short-circuit power dissipation at gates that are controlled is also minimized. Next, in Sec. 3, we discuss approaches to minimizing logic and wiring overheads in our framework. Sec. 4 presents an integer linear programming (ILP) formulation to optimize overheads subject to a critical-path delay constraint. Application of the new approach to test circuits (such as ripple carry adder and array multiplier) in 1.2μ

[*]N.R. Mahapatra was supported by startup funds from the State University of New York at Buffalo.

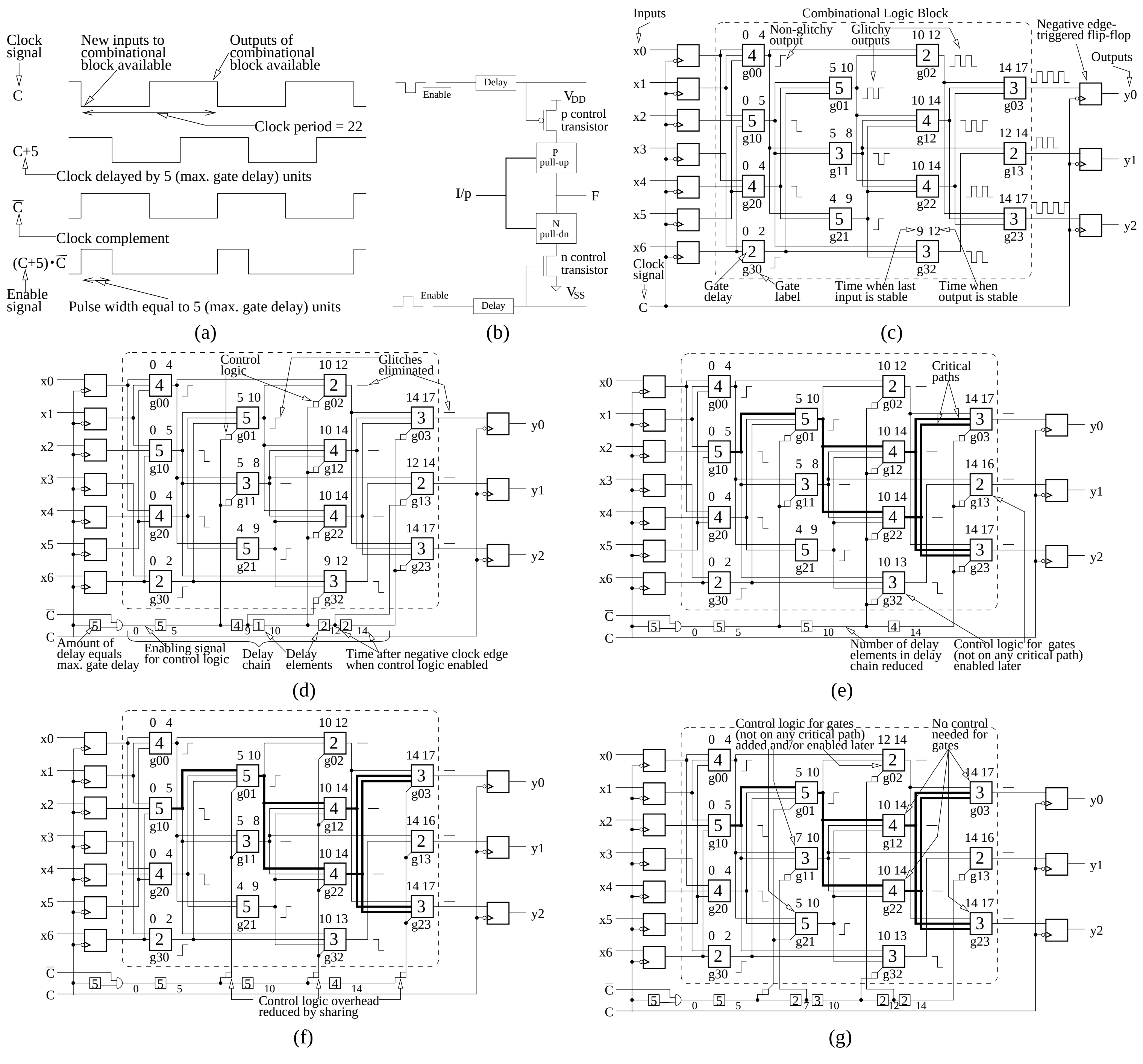
Clock signal
C
New inputs to combinational block available
Outputs of combinational block available
Clock period = 22
C+5
Clock delayed by 5 (max. gate delay) units
C̄
Clock complement
(C+5)•C̄
Enable signal
Pulse width equal to 5 (max. gate delay) units
(a)
Enable
Delay
V_DD
p control transistor
P pull-up
I/p
F
N pull-dn
n control transistor
V_SS
(b)
Inputs
Combinational Logic Block
Non-glitchy output
Glitchy outputs
Negative edge-triggered flip-flop
Outputs
Gate delay
Gate label
Time when last input is stable
Time when output is stable
Clock signal
(c)
Control logic
Glitches eliminated
Amount of delay equals max. gate delay
Enabling signal for control logic
Delay chain
Delay elements
Time after negative clock edge when control logic enabled
(d)
Critical paths
Number of delay elements in delay chain reduced
Control logic for gates (not on any critical path) enabled later
(e)
Control logic overhead reduced by sharing
(f)
Control logic for gates (not on any critical path) added and/or enabled later
No control needed for gates
(g)

Figure 1: Minimizing glitch power dissipation in a synchronous sequential circuit design and various optimizations: (a) Timing waveform for the clock signal and derivation of the enable signal for control logic from it. (b) Control logic comprising n and p control transistors connected to V_{SS} and V_{DD}, respectively, is enabled when the last input to the gate arrives. This prevents unnecessary charging and discharging to V_{DD} and V_{SS}, respectively, of output capacitance and internal capacitances in P pull-up and N pull-down and also prevents short-circuit current during the time when input signals are unsteady. Note that for this particular control logic, two delay chains are needed, one for the p and the other for the n control transistor. However, control logic types that control connection to only V_{DD} or only V_{SS} need a single delay chain, but provide a little less effective glitch power reduction [8]. (c) Glitches occurring in a combinational logic block at the outputs of gates that have multiple inputs changing asynchronously. (d) Glitches minimized by adding control logic to every potentially glitchy gate and enabling it through a delay chain when the last input to the gate stabilizes. (e) Control logic enabled later for certain gates not on any critical path so as to have synchronous evaluation of a number of gates and thereby reduce the number of delay elements required. (f) Control logic overhead reduced by sharing control logic across multiple synchronously evaluating gates. (g) Control logic for certain gates not on any critical path added and/or enabled later so as to make arrival times of inputs to a fanout gate(s) equal, thereby eliminating delay element, control logic, and wiring overhead for the latter gate.

technology yields 95% or more elimination of glitch power dissipation with negligible area and timing overheads after optimization. Then in Sec. 5 we briefly discuss related previous work. Conclusions are in Sec. 6.

2 Proposed Methodology: Gate Triggering

The key idea we employ to minimize glitches is to trigger logic evaluation at a gate only after all of its inputs have stabilized. For this purpose, to every potentially glitchy gate, we add some small control logic, which, when enabled, triggers logic evaluation at the gate (see Fig. 1(d)). Essentially, this logic controls gate connection to V_{DD} and/or V_{SS}. Fig. 1(b) shows one possible type of control logic. Various types of control logic and their analyses and associated simulation results are the subject of another paper [8]. In order to enable the control logic for different gates in the combinational logic block of Fig. 1(d) at the proper times (i.e., when the last input to the individual gates has stabilized), we first perform a timing simulation of the combinational block. Timing simulation is an essential step in the design flow of a VLSI chip [18] (e.g., to determine the critical-path delay in a combinational block, which in turn determines the clock period). Hence it does not represent an extra step in the application of our method. From this timing simulation, we obtain the delays of different gates and also the latest times by which the various inputs of a gate will have stabilized. For instance, in Fig. 1(d), gate `g01` has a delay of five time units, and its top, middle, and bottom inputs will have steady-state values latest by five, four, and two time units, respectively.

In order to prevent glitches at the output of gate `g01`, its control logic can be enabled at time five and should remain enabled for at least five time units, which is the delay of the gate, so that the gate logic may evaluate completely. Therefore we use an enable signal for the combinational block with a high period equal to the maximum delay for any gate in the block (five units for the combinational block of Fig. 1(d)). As shown in Figs. 1(a) and (d), the enable signal is generated by ANDing clock complement with the clock signal delayed by this maximum delay. This initial enable signal is then delayed by various amounts using a delay chain comprising delay elements as in Fig. 1(d). The output of a delay element in this chain provides an appropriately delayed version of the initial enable signal that can be used to trigger a gate(s). For example, in Fig. 1(d), gates `g01` and `g11`, for both of which the last input stabilizes by time five, are controlled by the initial enable signal delayed by five time units. Similarly, gate `g32` is enabled at time nine, since that is when its last input (the bottom input) stabilizes. In contrast, gate `g21` does not need any control logic or enable signal since both of its inputs have equal delays.

Using the above approach, all potentially glitchy gates are triggered by the enable signal when the last input to them stabilizes, thus ideally preventing all glitches in the combinational block. In practice, however, minor or partial glitches may occur due to the nonideal behavior of transistors. It should also be noted that short-circuit power dissipation in all triggered gates can be minimized by triggering them after the last input has stabilized, since before triggering, gate connection to V_{DD} and/or V_{SS} is cut off by the control logic. However, in most cases, it will not be cost-effective to control all gates in this manner to minimize short-circuit power dissipation because of the overheads it will entail. Rather, it will be best to control few select gates where potential for glitch and short-circuit power savings is maximum.

The main overheads of our approach are logic (delay element and control logic) overhead, wiring overhead for generating and routing the enable signal for potentially glitchy gates, and a delay overhead because of an increased delay for the combinational block. The logic overhead for generating the initial enable signal using an AND gate and a delay element is minimal. We have observed in our simulations that the delay overhead is negligible. Note that reducing the number of delay elements or the amount of control logic should lead to lower wiring overhead, since each delay element corresponds to a distinct enable signal to be routed and each control logic corresponds to an enable signal to be routed to control it. In the next section, we provide some ways by which logic overhead, and thus wiring overhead, can be minimized.

3. Logic and Wiring Overhead Optimization

3.1 Delay Element Optimization

The delay element overhead depends primarily upon the total delay provided by all delay elements and the number of delay elements. The number of delay elements in turn depends upon the number of delayed versions of the enable signal needed from the delay chain. Therefore, the number of delay elements can be reduced by synchronously triggering with a common enable signal as many gates as possible after their last inputs have stabilized. For example, in Fig. 1(e), the set of gates `g01` and `g11`, the set of gates `g02`, `g12`, `g22`, and `g32`, and the set of gates `g03`, `g13`, and `g23` are all triggered synchronously by enable signals delayed by five, ten, and fourteen time units, respectively. Note that to synchronize, some gates are triggered later than normal (such as `g32`, and `g13` in Fig. 1(e) compared to Fig. 1(d)). Also, note that the gates selected for late triggering are not on any critical path (shown with bold lines in Fig. 1(e)) so as not to increase the overall delay of the combinational block. The application of this optimization thus results in a much reduced number of delay elements in the delay chain (three in Fig. 1(e) compared to five in Fig. 1(d)) Note that a smaller number of delay elements also means a smaller wiring overhead, since a fewer number of distinct enable signals need to be routed. The delay element we chose is a transmission gate with appropriately-sized transistors to provide the required amount of delay. We selected a transmission gate because it requires less area and consumes very little power. A detailed discussion and comparison of delay elements motivating our choice is the subject of another paper ([9]).

3.2 Control Logic Optimization

There are two ways in which control logic may be optimized. First, after applying the technique to reduce the number of delay elements discussed above, we can use the same control logic to control all gates that are to be triggered simultaneously. For instance, in Fig. 1(e), the set of gates `g01` and `g11` can be controlled by the same control logic as shown in Fig. 1(f). However, it should be noted that sharing the control logic in this way may mean that the transistors of the control logic will need to be sized up (compared to when no sharing is done) to avoid increase in delay of the combinational block.

Another way to reduce the amount of control logic is to schedule the triggering of earlier gates so that inputs to later gates are synchronized. For instance, in Fig. 1(g), gates `g11` and `g02` are triggered later than normal and control logic is added to gate `g21` (compare Fig. 1(g) to Fig. 1(f)) so that all inputs to gates `g12`, `g22`, `g03`, and `g23` are synchronized, thereby obviating the need for controlling these gates using control logic. Note again that the gates selected for late triggering are those not on any critical path in order not to increase the delay of the combinational block. This results in less control logic, and possibly less delay element and wiring overhead, since some enable signals no longer need to be generated and routed.

The various optimizations discussed above are not exclusive, but may be used in conjunction to various degrees depending upon the combinational logic block under consideration to minimize the total overhead. In the next section, we formulate this overhead minimization problem as an integer linear program (ILP) subject to a critical-path delay constraint.

4 ILP Formulation for Overhead Minimization

The overhead minimization problem can be stated as follows:

Problem 1 *Minimize a weighted sum of the total amount of delay and the number of delay elements in the delay chain, and the number of gates triggered (which corresponds to the amount of control logic and wiring required), such that: (1) there are no glitches, and (2) the critical-path delay of the circuit does not exceed a specified upper bound.*

Clearly, the total amount of delay corresponds to the latest triggering time over all gates, while the number of delay elements to the number of distinct gate triggering times (see Fig. 1(g)). No glitching requires that every gate with asynchronous inputs must be triggered no earlier than the latest input arrival time for that gate; obviously, a gate with synchronous inputs will not glitch and hence should not be triggered. The problem of glitch *minimization*, in which the amount of glitching is part of the objective to be minimized, as opposed to glitch *elimination* being considered here, seems to be more difficult, and will be considered in future research. The second constraint in Problem 1 implies the following theorem.

Theorem 1 *There exists a finite set T_u of triggering time instants for every gate u in the circuit such that the optimal solution to Problem 1 is not affected by restricting u's triggering time to T_u.*

Proof Outline: The latest input arrival time for a gate in the original circuit (before applying the glitch-minimization technique) and the upper bound on critical-path delay set lower and upper bounds, respectively, on the triggering time of the gate. Triggering a gate later than the lower-bound time, rather than triggering it at the lower-bound time, can lead to lower overhead only if it satisfies one of the following two conditions. (1) The gate triggering time synchronizes with the triggering time of at least one other gate, so that a common control signal can be used for both gates, thereby saving a delay element (see Fig. 1(f)). (2) The gate triggering time is such that the gate output synchronizes with the arrival of other inputs to some fan-out gate, thereby saving a control element at the fan-out gate, which does not need to be controlled (see Fig. 1(g)). There are only a finite set of triggering time instants that will lead to one of these two synchronizations. □

Space constraints do not permit us to specify and prove what the exact finite set of triggering times implied by Theorem 1 is for a gate.

The constants, variables, expressions, objective (corresponding to Problem 1), and constraints for the ILP and their descriptions are given for easy reference in Table 1. Only two points need to be explained, and after that the rest of the ILP is easily understood by inspection of the detailed descriptions in the table. First, we need to understand for each variable the constraints that enforce the values indicated for it in the table. The value of variable $I_{u,i}^{max}$ is enforced by constraints R_1, R_2, and R_3, that of $I_{u,i}^{min}$ by R_4, R_5, and R_6, that of A_u^{trig} by R_7, and that of $A_{u,i}$ by R_8 and R_9. When gate u is triggered, the value of $B_{u,i}$ is enforced by constraints R_{10} and R_{11}, otherwise, by constraints R_{10}, R_{12}, and R_{13}. The value of A_i is enforced by constraints R_{14} and R_{15}, and finally, that of X_i by R_{16}, R_{17}, and R_{18}.

The second point to be understood is the correspondence between the two principal constraints of Problem 1 (i.e., no glitches and bounded critical-path delay increase) and the constraints of the table. Constraints R_1 through R_7 ensure that a gate is triggered when its inputs are asynchronous. Constraints R_8 and R_9 ensure that whenever a gate is triggered, its triggering time is no earlier than the latest input arrival time for the gate, so that all glitches are eliminated. We note that the upper bound on the triggering time for every gate automatically enforces the constraint on the increase in the critical-path delay. The objective function in the table directly corresponds to the objective in Problem 1.

5 Related Work

Glitch and short-circuit power dissipation are discussed in [12, 13]. Glitch estimation, modeling, and propagation issues are covered in [3, 11, 16]. The importance of glitch minimization for various applications is considered in [4, 17, 19]. Designing two-level glitch-free circuits using logic redesign, assuming only one input changes at a time, is addressed in [6]. Glitch removal through path balancing obtained via, say, transistor sizing or layout changes, is discussed in [10, 12, 13]; this can be cumbersome and involves trial and error. Furthermore, in deep submicron technologies, transistor sizing will not be very effective for path balancing since logic delays become relatively smaller compared to interconnect delays. Retiming and buffer placement approaches to filter or reduce glitches and glitch propagation are described in [2, 7]; these approaches, although somewhat effective, entail appreciable area overheads for flipflops and buffers. Glitch reduction at the RTL level in control flow intensive designs is given in [15].

Therefore, current methods for glitch reduction are either (i) not applicable in all contexts, or (ii) can not be automated and are ad hoc, or (iii) are not very effective, or (iv) have high area/delay overheads, or (v) restrict the manner in which logic is transformed to a gate realization. There is no methodical, generally applicable approach to minimizing glitch power dissipation. Our proposed gate triggering approach in this paper attempts to overcome all the above limitations of current approaches.

6 Conclusions

Although research into various aspects of glitch power dissipation has been undertaken in the past, most approaches to addressing it are *ad hoc* and limited in their applicability. This paper presented a new framework called gate triggering for systematically minimizing glitch power dissipation in static CMOS ICs. The logic and wiring overheads of our approach were analyzed and an ILP formulation was given to minimize these overheads subject to a critical-path delay constraint. Application of the new approach to test circuits (such as ripple carry adder and array multiplier) yields 95% or more elimination of glitch and, in gates to which applied, short-circuit power dissipation with very little to negligible area and timing overheads after optimization.

References

[1] A.P. Chandrakasan and R.W. Broderson, "Minimizing power consumption in digital CMOS circuits," *Proceedings of the IEEE*, Vol. 83, No. 4, pp. 498-523, Apr. 1995.

[2] M. Favalli and C. Metra, "The effect of glitches on CMOS buffer optimization," *Proc. PATMOS*, Oct. 1995 in pp. 202-212, Oldenberg, Germany, Oct. 1995.

[3] M. Favalli and L. Benini, "Analysis of glitch power dissipation in CMOS IC's," *Proc. of ISLPED*, pp. 123-128, 1995.

[4] "Gated clocks and hazards," http://erebor.cudenver.edu/courses/ee3651/hazards/hazard.html.

[5] F.S. Hillier and G.J. Lieberman, "Introduction to operations research," *McGraw-Hill*, 1995.

[6] R.H. Katz, "Contemporary logic design," *Addison Wesley Publications*, 1993.

[7] J. Leijten, et al., "Analysis and reduction of glitches in synchronous networks," *Proc. EDAC*, pp. 398-403, 1995.

[8] N.R. Mahapatra, S.V. Garimella, and A. Tareen, "Efficient techniques for designing static CMOS ICs with very low glitch power dissipation," submitted to *ISCAS 2000*.

[9] N.R. Mahapatra, A. Tareen, and S.V. Garimella, "An experimental and analytical comparison of delay elements," Technical Report, Dept. of Computer Sc. & Eng., SUNY-Buffalo, Buffalo, NY, 1999.

[10] E.J. McCluskey, "Logic design principles with emphasis on testable semicustom circuits," *Prentice Hall*, Englewood Cliffs, NJ, 1986.

[11] F.N. Najm, "A survey of of power estimation techniques in VLSI circuits," *IEEE T-VLSI*, Vol. 2, No. 4, Dec. 1994.

[12] M. Pedram, "Power minimization in IC design: Principles and applications," *ACM TODAES*, Vol. 1, No. 1, pp. 3-56, Jan. 1996.

[13] J.M. Rabaey, "Digital Integrate Circuits," *Prentice Hall*, 1996.

[14] D. Rabe, et al., "Comparison of different gate level glitch models," *Proc. PATMOS'96*, Bologna, Italy, 1996.

[15] A. Raghunathan, S. Dey, and N.K. Jha, "Glitch analysis and reduction in register transfer level power optimization," *DAC*, pp. 331-336, Jun. '96.

[16] S.J. Abou-Samra and A. Guyot, "Glitch threshold," *FTFC'97*.

[17] P.S.K. Siegel, "Automatic technology mapping for asynchronous designs," Tech. Rep. CSL-TR-95-663, Stanford Univ.

[18] N.H.E. Weste and K. Eshraghian, "Principles of CMOS VLSI design," *Addison-Wesley*, 1993.

[19] T.-Y. Wu, et al., "A low glitch 10-bit 75-MHz CMOS video D/A converter," *IEEE JSSC*, Vol. 30, No. 1, Jan. 1995.

Label	Name	Description	Label	Name	Description
Constants					
	V	Set of all gates in the combinational gate network.		$D_u, u \in V$	Delay for gate u.
	$T = (T_0, T_1, \ldots, T_{m-1})$	Set of all possible gate triggering times.		$T_u = (T_{u,0}, T_{u,1}, \ldots, T_{u,m_u-1})$	Set of all possible triggering times for gate u.
	$T_u^{in} = (T_{u,0}^{in}, T_{u,1}^{in}, \ldots, T_{u,n_u-1}^{in})$, $u \in V$	Set of all possible input arrival times for gate u.		$F_u^{in}, u \in V$	Set of fan-in gates for gate u.
Variables					
	$I_{u,i}^{max}, 0 \leq i < \lvert T_u^{in}\rvert, u \in V$	$= 1$, if the latest input arrival time for gate u is $T_{u,i}^{in}$, else $= 0$.		$I_{u,i}^{min}, 0 \leq i < \lvert T_u^{in}\rvert, u \in V$	$= 1$, if the earliest input arrival time for gate u is $T_{u,i}^{in}$, else $= 0$.
	$A_u^{trig}, u \in V$	$= 1$, if the inputs to gate u are asynchronous (i.e., if gate u needs to be triggered).		$A_{u,i}, 0 \leq i < \lvert T_u\rvert, u \in V$	$= 1$, if gate u is triggered at time $T_{u,i}$, else $= 0$.
	$B_{u,i}, 0 \leq i < \lvert T_u\rvert, u \in V$	$= 1$, if the output of gate u becomes stable at $T_{u,i} + D_u$, else $= 0$.		$A_i, 0 \leq i < \lvert T\rvert$	$= 1$, if at least one gate is triggered at time T_i, else $= 0$.
	$X_i, 0 \leq i \leq \lvert T\rvert$	$= 1$, if the latest triggering time over all gates is T_i, else $= 0$.			
Expressions					
$E_1(u)$	$\Sigma_{0 \leq i < \lvert T_u^{in}\rvert}(I_{u,i}^{max} \cdot T_{u,i}^{in})$, $u \in V$	Latest input arrival time for gate u.	$E_2(u)$	$\Sigma_{0 \leq i < \lvert T_u^{in}\rvert}(I_{u,i}^{min} \cdot T_{u,i}^{in})$, $u \in V$	Earliest input arrival time for gate u.
$E_3(u)$	$\Sigma_{0 \leq i < \lvert T_u^{in}\rvert} I_{u,i}^{max}, u \in V$	Number of latest input arrival times for gate u.	$E_4(u)$	$\Sigma_{0 \leq i < \lvert T_u^{in}\rvert} I_{u,i}^{min}, u \in V$	Number of earliest input arrival times for gate u.
$E_5(u)$	$\Sigma_{0 \leq i < \lvert T_u\rvert}(B_{u,i} \cdot T_{u,i}) + D_u$, $u \in V$	Time at which ouput of gate u stabilizes.	$E_6(u,i)$	$\Sigma_{v \in F_u^{in}} B_{v,j}$, $j \vert T_{v,j} + D_v = T_{u,i}^{in}$, $0 \leq i < \lvert T_u^{in}\rvert, u \in V$	Number of inputs that arrive at time $T_{u,i}^{in}$ for gate u.
$E_7(u)$	$\Sigma_{0 \leq i < \lvert T_u\rvert} A_{u,i}, u \in V$	$= 1$, if gate u is triggered, else $= 0$.	$E_8(u)$	$\Sigma_{0 \leq i < \lvert T_u\rvert}(A_{u,i} \cdot T_{u,i})$, $u \in V$	Time at which gate u is triggered.
$E_9(u)$	$\Sigma_{0 \leq i < \lvert T_u\rvert} B_{u,i}, u \in V$	Number of times output of gate u becomes stable.	$E_{10}(u)$	$\Sigma_{0 \leq i < \lvert T_u\rvert}(B_{u,i} \cdot T_{u,i})$, $u \in V$	Time at which output of gate u stabilizes is given by $E_{10}(u) + D_u$.
$E_{11}(i)$	$\Sigma_{u \in V} A_{u,j},\ j \vert T_{u,j} = T_i$, $0 \leq i < \lvert T\rvert$	Number of gates triggered at time T_i.	$E_{12}(u,i)$	$A_{u,j}, j \vert T_{u,j} = T_i, u \in V$, $0 \leq i < \lvert T\rvert$	$= 1$, if gate u is triggered at time T_i, else $= 0$.
$E_{13}(i)$	$A_i \cdot T_i, 0 \leq i < \lvert T\rvert$	$= T_i$, if at least one gate is triggered at time T_i, else $= 0$.	E_{14}	$\Sigma_{0 \leq i < \lvert T\rvert}(X_i \cdot T_i)$	Latest triggering time over all gates.
E_{15}	$\Sigma_{0 \leq i < \lvert T\rvert} A_i$	Number of distinct triggering times in the circuit.	E_{16}	$\Sigma_{u \in V} \Sigma_{0 \leq i < \lvert T_u\rvert} A_{u,i}$	Number of gates triggered.
E_{17}	$\Sigma_{0 \leq i < \lvert T\rvert} X_i$	Number of latest triggering times.			
Objective Function					
	Objective $= C_{tot\_delay} \cdot E_{14} + C_{num\_delay} \cdot E_{15} + C_{trig} \cdot E_{16}$				Objective is to minimize total overhead comprising total amount of delay in the delay chain, number of delay elements, and the number of gates triggered/controlled; C_{tot_delay}, C_{num_delay}, and C_{trig} are the associated weights.
Constraints					
R_1	$E_3(u) = 1$	There is only one latest input arrival time for gate u.	R_2	$E_1(u) \geq E_5(v), v \in F_u^{in}$, $u \in V$	The latest input arrival time for gate u can be no earlier than any of its individual input arrival times.
R_3	$I_{u,i}^{max} \leq E_6(u,i)$, $0 \leq i < \lvert T_u^{in}\rvert, u \in V$	The latest input arrival time for gate u has to be one of its input arrival times.	R_4	$E_4(u) = 1$	There is only one earliest input arrival time for gate u.
R_5	$E_2(u) \leq E_5(v), v \in F_u^{in}$, $u \in V$	The earliest input arrival time for gate u can be no later than any of its input arrival times.	R_6	$I_{u,i}^{min} \leq E_6(u,i)$, $0 \leq i < \lvert T_u^{in}\rvert, u \in V$	The earliest input arrival time for gate u has to one of its input arrival times.
R_7	$M \cdot A_u^{trig} \geq E_1(u) - E_2(u)$, $u \in V$, M is a sufficiently large number.	Gate u is triggered if its inputs are asynchronous (i.e., if its latest and earliest input arrival times are unequal).	R_8	$E_7(u) \leq 1, u \in V$	Gate u cannot be triggered more than once.
R_9	$E_8(u) \geq E_5(v) - M(1 - A_u^{trig}), v \in F_u^{in}$, $u \in V$	If gate u is triggered, its triggering time can be no earlier than any of its individual input arrival times.	R_{10}	$E_9(u) = 1, u \in V$	Output of gate u can stabilize exactly once.
R_{11}	$B_{u,i} \geq A_{u,i}, 0 \leq i < \lvert T_u\rvert$, $u \in V$	If gate u is triggered at time $T_{u,i}$, then its ouput stabilizes at time $T_{u,i} + D_u$.	R_{12}	$E_{10}(u) \leq E_1(u) + M \cdot E_7(u)$, M is a sufficiently large number, $u \in V$	If gate u is not triggered (i.e., all its inputs are synchronous or its earliest and latest input arrival times are equal), the time at which its output stabilizes cannot be any later than the sum of its latest input arrival time and its gate delay.
R_{13}	$E_{10}(u) \geq E_2(u) - M \cdot E_7(u)$, M is a sufficiently large number, $u \in V$	If gate u is not triggered (i.e., all its inputs are synchronous or its earliest and latest input arrival times are equal), the time at which its output stabilizes cannot be earlier than the sum of its earliest input arrival time and its gate delay.	R_{14}	$A_i \leq E_{11}(i), 0 \leq i < \lvert T\rvert$	Triggering time T_i is not chosen (i.e., $A_i = 0$) if no gate is triggered at time T_i.
R_{15}	$A_i \geq E_{12}(u,i), u \in V$, $0 \leq i < \lvert T\rvert$	Triggering time T_i is chosen (i.e., $A_i = 1$) if at least one gate is triggered at time T_i.	R_{16}	$E_{17} \leq 1$	The number of latest triggering times can be no greater than 1.
R_{17}	$E_{14} \geq E_{13}(i), 0 \leq i < \lvert T\rvert$	The latest triggering time can be no earlier than any of the gate triggering times.	R_{18}	$X_i \leq A_i, 0 \leq i < \lvert T\rvert$	The latest triggering time can only be chosen from one of the gate triggering times.

Table 1: ILP formulation for minimizing delay element, control logic, and wiring overheads when applying the glitch-minimization technique of Sec. 2 to a combinational logic block so that no glitches occur and the increase in critical-path delay of the block does not exceed a specified upper bound.