

IV. CONCLUSION

The SLS and RLS estimation methods proposed for average power estimation are linear, unbiased, and do not assume any distribution for the input sample data. The experimental results indicate that the least square estimation algorithms converge up to 24 times faster than the Monte Carlo and DIPE for random inputs.

ACKNOWLEDGMENT

The authors would like to thank the reviewers and the associate editor for their comments and suggestions.

REFERENCES

- [1] E. M. Sentovich *et al.*, "SIS: A system for sequential circuit synthesis," in *Electronics Research Laboratory*, Berkeley, CA, May 1992.
- [2] S. M. Kay, *Fundamentals of Statistical Signal Processing—Estimation Theory*. Englewood Cliffs, NJ: Prentice Hall, 1993, ch. 7 and 8.
- [3] P. S. R. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*. Norwell, MA: Kluwer, 1997.
- [4] R. Burch, F. N. Najm, P. Yang, and T. Trick, "A Monte-Carlo approach for power estimation," *IEEE Trans. VLSI Syst.*, vol. 1, pp. 63–71, Mar. 1993.
- [5] L. P. Yuan, C. C. Teng, and S. M. Kang, "Statistical estimation of average power dissipation in sequential circuits," in *Proc. Design Automation Conf.*, 1997, pp. 377–382.
- [6] C. S. Ding, Q. Wu, C. T. Hsieh, and M. Pedram, "Statistical estimation of the cumulative distribution function for power dissipation in VLSI circuits," in *Proc. Design Automation Conf.*, 1997, pp. 371–376.
- [7] M. G. Xakellis and F. N. Najm, "Statistical estimation of the switching activity in digital circuits," in *Proc. Design Automation Conf.*, 1994, pp. 728–733.
- [8] L. P. Yuan, C. C. Teng, and S. M. Kang, "Statistical estimation of average power dissipation using nonparametric techniques," *IEEE Trans. VLSI Syst.*, vol. 6, pp. 65–73, Feb. 1998.
- [9] C.-S. Ding, Q. Wu, C.-T. Hsieh, and M. Pedram, "Stratified random sampling for power estimation," *IEEE Trans. VLSI Syst.*, vol. 17, pp. 465–471, June 1998.
- [10] A. K. Murugavel, N. Ranganathan, R. Chandramouli, and S. Chavali, "Average power in digital CMOS circuits using least square estimation," in *Proc. Int. Conf. VLSI Design*, 2001, pp. 215–220.
- [11] R. Burch, F. N. Najm, P. Yang, and T. Trick, "McPOWER: A Monte Carlo approach to power estimation," in *Proc. Int. Conf. Computer-Aided Design*, 1992, pp. 90–97.
- [12] T. L. Chou and K. Roy, "Accurate power estimation of CMOS sequential circuits," *IEEE Trans. VLSI Syst.*, vol. 4, pp. 369–380, Sept. 1996.
- [13] C. Tsui, M. Pedram, and A. M. Despaigne, "Efficient estimation of dynamic power consumption under a real delay model," in *Proc. Int. Conf. Computer-Aided Design*, 1993, pp. 224–228.
- [14] F. N. Najm, "Transition density: A new measure of activity in digital circuits," *IEEE Trans. Computer-Aided Design*, vol. 12, pp. 310–323, Feb. 1993.
- [15] S. Bhanja and N. Ranganathan, "Dependency preserving probabilistic modeling of switching activity using Bayesian networks," in *Proc. Design Automation Conf.*, 2001, pp. 209–214.
- [16] S. Gupta and F. N. Najm, "Power modeling for high-level estimation," *IEEE Trans. VLSI Syst.*, vol. 8, pp. 18–29, Feb. 2000.
- [17] N. Shanbhag, "Lower bounds on power dissipation for DSP algorithms," in *Proc. Int. Symp. Low Power Design*, 1996, pp. 43–48.
- [18] F. N. Najm and M. Y. Zhang, "Extreme delay sensitive and the worst case switching activity in VLSI circuits," in *Proc. Design Automation Conf.*, 1995, pp. 623–627.
- [19] D. Marculescu, R. Marculescu, and M. Pedram, "Information theoretic measures for power analysis," *IEEE Trans. Computer-Aided Design*, vol. 16, pp. 599–610, June 1996.
- [20] V. Krishna, R. Chandramouli, and N. Ranganathan, "Computation of lower and upper bounds for switching activity using decision theory," *IEEE Trans. VLSI Syst.*, vol. 7, pp. 125–129, Mar. 1999.

Locally Clocked Pipelines and Dynamic Logic

Gregg N. Hoyer, Gin Yee, and Carl Sechen

Abstract—Micropipelines and most of its variants use a delay-insensitive controller to moderate a pipeline. In search of improved performance, we depart from the delay-insensitive model in favor of a bounded-delay model for the controller. In particular, we demonstrate how a general delay-insensitive controller for level-sensitive pipelines can be improved by assuming a bounded-delay model and taking advantage of delay information to make the controller faster and more efficient. The new control scheme is referred to as locally clocked (LC) control. A highly pipelined logic technique called LC dynamic logic is presented that combines the bounded-delay controller with a latching dynamic logic gate design. Simulations comparing LC control with its delay-insensitive counterpart are presented. Also, an 8×8 bit multiplier with a maximum frequency of 715 MHz for a $1 \mu\text{m}$ CMOS process that uses LC dynamic logic is presented.

Index Terms—Asynchronous pipelines, digital complementary metal-oxide-semiconductor (CMOS), dynamic-logic circuit, high performance, locally clocked (LC) control, multiplier design.

I. INTRODUCTION

Asynchronous techniques avoid many of the difficulties associated with synchronous designs by using some type of internal mechanism to determine the system states [1]. One such asynchronous technique is micropipelines, which were introduced in Ivan Sutherland's Turing Award paper [2]. Micropipelines combine a delay-insensitive control structure and a bounded-delay pipelined datapath. In Section II, we provide an introduction to micropipelines and discuss a basic micropipeline scheme for controlling "traditional" linear pipelines that use level-sensitive storage elements. In Section III, the general micropipeline structure is modified, producing a new control scheme called locally clocked (LC) control. LC control abandons a strict delay-insensitive handshake protocol in favor of a bounded-delay model that enables the controller to achieve a higher level of performance and become more efficient. Section IV discusses a high-speed dynamic gate design that combines logic functionality and latching capabilities into one gate. The LC control scheme can be used in conjunction with these gates to provide a low-latency high-throughput circuit solution referred to as LC dynamic logic. Section V contains simulation results comparing the performance of the LC controller to its delay-insensitive counterpart. Finally, an 8×8 bit multiplier using LC dynamic logic that has a maximum frequency of 715 MHz for a $1\text{-}\mu\text{m}$ CMOS process is presented.

II. MICROPIPELINES

Micropipelines are event-driven elastic pipeline structures that use a bundled data interface to communicate and transfer data between adjacent blocks. They are created by cascading stages that all adhere to the same handshake protocol. Sutherland's micropipeline [2] uses a delay-insensitive handshake protocol for the control circuitry that moderates the bounded-delay datapath. Since their introduction, many other micropipeline variants have been developed [3]–[5]. Fig. 1 shows

Manuscript received May 12, 1999; revised October 25, 1999. This work was supported by the Semiconductor Research Corporation, CDADIC, and Sun Microsystems.

The authors are with the Department of Electrical Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: gnhoyer@ee.washington.edu; gsyee@ee.washington.edu; sechen@ee.washington.edu).

Publisher Item Identifier S 1063-8210(02)00793-X.

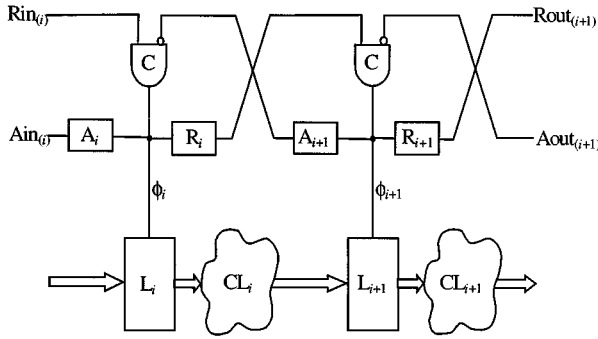


Fig. 1. General micropipeline structure.

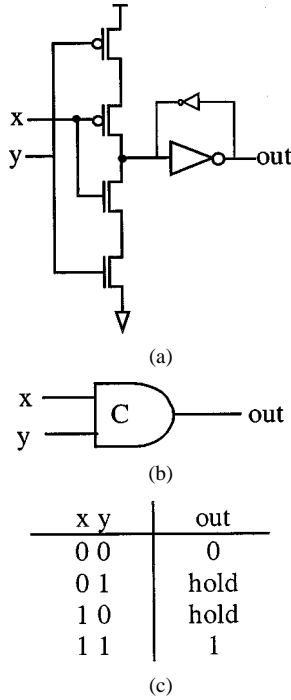


Fig. 2. Muller C-element. (a) Schematic. (b) Symbol. (c) Truth table.

two stages of a general micropipeline structure that is controlling a linear-pipelined datapath.

Each stage of the control circuitry consists of a Muller C-element and two delay elements R_i and A_i . The datapath consists of storage elements and a block of combinational logic. The control circuitry generates stage-clock signals (ϕ_i) that dictate the movement of data tokens through the stages. The two delay elements in each stage are used to meet the timing constraints of the bounded-delay datapath. The Muller C-element associated with each stage is used to coordinate the states of neighboring stages and determine the state of the current stage. Fig. 2 details a Muller C-element and its properties.

In the discussion to follow, we will present a scheme for using the micropipeline control circuitry of Fig. 1 to control a bounded-delay level-sensitive linear pipeline. First, assume the storage elements are latches that are transparent when the clock is high and opaque when the clock is low. Second, assume that the C-elements have no delay associated with them. Third, for this analysis assume that the rise and fall delay of the delay elements can be adjusted separately. The rise and fall delays of the delay element in the request path of stage i will be referred to as $R_{u(i)}$ and $R_{d(i)}$, respectively. Likewise, the rise and fall delays of the delay element in the acknowledge path of stage i will be referred to as $A_{u(i)}$ and $A_{d(i)}$, respectively.

One possible scheme for controlling the data token movement is to make sure the following two timing constraints are enforced by the micropipeline control circuitry.

- 1) Data must be valid at latch inputs prior to the latches going transparent (setup constraint).
- 2) Data must not change at latch inputs until the latches have gone opaque (hold constraint).

In this control methodology, a new data token arrives at the latch inputs of a stage when the latches are opaque. After the data token has arrived, it gets accepted into the stage when the latches go transparent. The latches must then return opaque prior to the arrival of the next data token. This cycle is repeated for every data token.

Rule 1 is a max-delay constraint that specifies when data tokens must be available to a stage and is enforced by the following constraint on the request delay element:

$$R_{u(i)} \geq T_{\text{latch}(i)} + T_{\text{CLmax}(i)} + T_{\text{setup}(i+1)}. \quad (1)$$

This constraint states that the rise delay of the delay element R at stage i needs to be equal to or greater than the sum of the latch delay ($T_{\text{latch}(i)}$), combinational logic critical delay ($T_{\text{CLmax}(i)}$) for stage i and setup time for the latches of the successor stage ($T_{\text{setup}(i+1)}$).

After a stage clock ϕ_i has gone high to make the latches L_i transparent, its predecessor stage is allowed to return opaque by setting the rise time of the feedback to be equal to or greater than zero

$$A_{u(i)} \geq 0. \quad (2)$$

Once the latches for a stage i have returned opaque, a new data token can be placed on the inputs of the latches L_i . To guarantee that the latches will be completely opaque by the time the new data token arrives at the inputs of the latches (Rule 2) the following constraint is placed on the acknowledge delay element:

$$A_{d(i)} \geq T_{\text{hold}(i)} - (T_{\text{latch}(i-1)} + T_{\text{CLmin}(i-1)}). \quad (3)$$

This constraint states, that the fall delay of the acknowledge feedback delay element at stage i , needs to be equal to or greater than the hold time of the latches for stage i ($T_{\text{hold}(i)}$) minus the sum of the latch delay of stage $i-1$ latches and the minimum delay of the combinational logic in stage $i-1$ ($T_{\text{CLmin}(i-1)}$). The right-hand side of (3) is typically negative and in such cases the constraint can be reduced to

$$A_{d(i)} \geq 0. \quad (4)$$

Finally, in order to complete the full handshake protocol so the control circuitry operates correctly without the potential for race conditions, the following constraint is imposed:

$$R_{d(i)} \geq 0. \quad (5)$$

III. LC CONTROL

Each stage of the general micropipeline control circuitry shown in Fig. 1 has four delays that can be tuned to satisfy the timing constraints and the delay-insensitive handshaking protocol. By employing a bounded-delay model for both the datapath and the control circuitry, we can take advantage of the known delays to improve the performance and efficiency of the controller. Specifically, the fall delay of the delay element in the request path ($R_{d(i)}$) is used solely to complete the full handshake protocol, making the controller delay insensitive. By using a bounded-delay model, the devices associated with $R_{d(i)}$ can be removed and replaced with a timing constraint. This is done by replacing the full Muller C-elements in Fig. 1 with asymmetrical C-elements [6], shown in Fig. 3. The *request* from the predecessor stage is connected

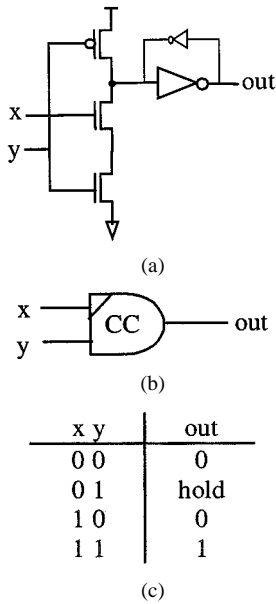


Fig. 3. CC element. (a) Schematic. (b) Symbol. (c) Truth table.

to the x input. The asymmetrical C-element used in LC control will be referred to as a clock controller (CC) element. Prior uses of general C-elements have concentrated on delay-insensitive designs.

The signal transition graph (STG) [7] for the LC controller is shown in Fig. 4. The rise and fall delays of the delay element in the request path are shown on their corresponding dependency arrow. Both the rise and fall delays of the acknowledge delay element are assumed to be zero.

The LC control circuitry contains a race condition that can potentially cause an incorrect sequence of signal transitions. Namely, if Path #2 is faster than Path #1, the active request (R_{out+}) from the predecessor stage $i - 1$ will produce multiple $\phi_i +$ transitions, which is an incorrect sequence of signal transitions. The correct sequence of transitions can be guaranteed by using a bounded-delay model for the controller and making sure that Path #1 is faster than Path #2. This can be accomplished by imposing the following constraint on fall delay of the delay element in the request paths:

$$R_{d(i)} \leq R_{u(i+1)} + R_{u(i+2)}. \quad (6)$$

The maximum sustainable frequency of a LC linear pipeline is governed by how quickly the slowest stage can process data tokens, which is given by (7)

$$T_{\text{cycle}} = \left[\left\{ R_{u(i)} + \delta_u(\text{CC}_{(i+1)}) \right\} + \delta_d(\text{CC}_{(i)}) \right. \\ \left. + \delta_u(\text{CC}_{(i-1)}) + \left\{ R_{u(i-1)} + \delta_u(\text{CC}_{(i)}) \right\} \right]. \quad (7)$$

$\delta_u(\text{CC}_{(i)})$ and $\delta_d(\text{CC}_{(i)})$ are the rise and fall delays of the CC element at stage i . Previous timing analyses assumed that the C-elements and CC elements did not have any delay. In reality, these elements do have a finite delay that needs to be taken into consideration. The two terms inside the curly braces are required in order to meet the timing constraints of the datapath. However, the two delays of the CC elements not in the curly braces are overhead associated with the control circuitry. The more efficient CC element in LC control reduces this overhead delay compared to its delay insensitive counterpart allowing the controller to achieve a higher level of performance. This is especially important in very fine-grained pipelines where the overhead terms are a significant portion of the cycle time. The highly pipelined dynamic

logic technique that will be presented in the next section exploits this fact.

LC control uses delay matching to enforce the timing constraints. Because of process and environmental variations it is necessary to add a safety margin (T_{margin}) to the delay elements to ensure that all of the timing constraints are satisfied by the matched delays. Typical margins used in industry for delay matching are 10–20%.

LC pipelines maintain the desirable properties associated with event-driven micropipelines. Namely, they reduce unneeded switching activity and eliminate latency bottlenecks. Furthermore, they behave like a first-in first-out (FIFO) queue that can process the data at each stage because they are elastic.

IV. LC DYNAMIC LOGIC

Several asynchronous dynamic logic pipeline schemes have used differential logic techniques that make use of the dual rail coded data to generate completion signals [3], [5], [8], [9]. LC dynamic logic uses single rail logic and relies upon delay matching in the controller to produce the correct handshaking signals. In doing so, LC dynamic logic eliminates the overhead associated with the completion detection circuitry. It also allows for high fanin OR/NOR structures that avoid large stack heights by not requiring a dual network.

LC dynamic logic gates are comprised of a dynamic logic portion followed by a latching section, as shown in Fig. 5. The logic of the gate is performed by the NMOS pull-down network in the dynamic logic portion of the gate when the gate is in evaluation (ϕ high). The latching section of the gate is used to hold the evaluation result produced by the dynamic logic section when the gate precharges (ϕ low). These gates pass a new data token when they are in evaluation, therefore, they can be thought of as being transparent. Likewise, they prohibit the propagation of data tokens when in precharge, so they are opaque.

The basic structure of LC dynamic logic gates is similar to that of true single-phase clock gates [10], however, they are controlled in a different manner. Designs that use LC dynamic logic must be fully pipelined with only one gate level per stage. A stage (i) can enter the evaluation state when the following three conditions are met:

- 1) inputs to stage i must be at their final values;
- 2) successor stage $i + 1$ must have consumed the previous data token;
- 3) all of the gates in stage i must be fully precharged.

There is only one condition that must be met before a stage (i) can enter the precharge state: the successor stage $i+1$ has started to evaluate the new data token.

Rule 1 is a setup constraint that allows the LC dynamic logic gates to perform both positive and negative logic functions. The concept of delaying the evaluation edge of the clock until the inputs are stable is similar in spirit to clock delayed (CD) domino [11]. Rule 2 eliminates the potential of data races and Rule 3 is necessary for the dynamic logic section to work correctly. Rule 4 guarantees that the data token has been accepted by the successor stage.

The LC control scheme can be used to generate stage clocks that enforce the rules required for LC dynamic logic. The constraints on $R_{u(i)}$, $A_{d(i)}$, and $A_{u(i)}$ will enforce LC dynamic logic Rules 1, 2, and 4, respectively. Also, each stage (i) is guaranteed to remain in precharge for at least as long as its successor stage ($i + 1$) is in evaluation. Therefore, by designing the gates in stage i such that they precharge faster than the gates in stage $i + 1$ evaluate, LC dynamic logic Rule 3 will be satisfied. This is accomplished by carefully sizing the precharge devices and noting that the precharge time does not include any propagation through the latching section of the LC dynamic logic gate.

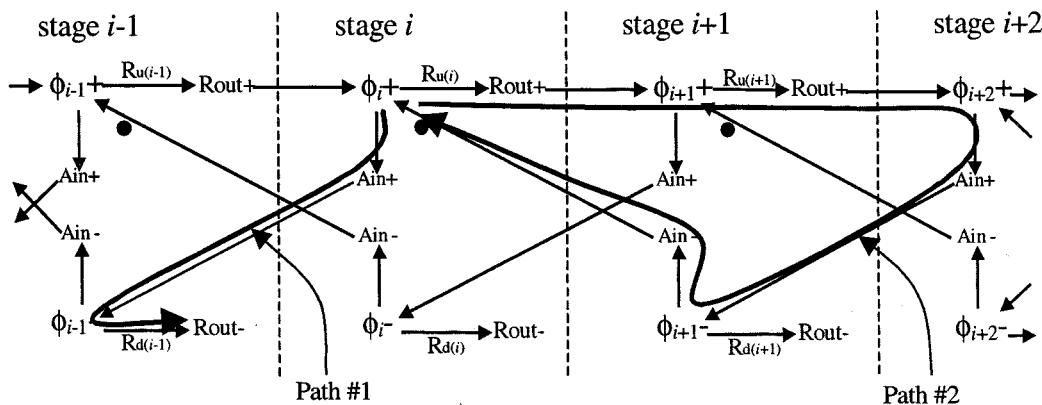


Fig. 4. STG for LC control.

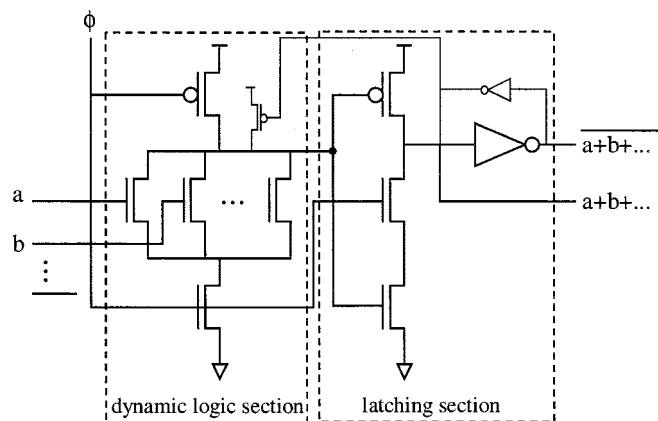


Fig. 5. LC dynamic logic gate structure.

V. RESULTS

The LC control scheme is capable of a higher performance level and is more energy efficient than the delay-insensitive full C-element implementation. This is because the CC element has better drive and input capacitance characteristics compared to a full Muller C-element. This fact can be seen by observing that the logical effort [12] of the full C-element is 2 on both inputs.¹ On the other hand, the logical effort of the CC element is 2/3 on input *x* and 4/3 on input *y*. The LC and delay-insensitive controllers were optimized for maximum frequency by eliminating all of the delay elements and simply using the CC element and the full C-element for the required delays. Both designs had a load of 25 minimum-sized inverters on each stage clock; the stage clocks were not buffered and were simulated in a 1-μm CMOS process.² Table I shows that the LC control circuitry has a maximum obtainable throughput that is 54% faster, a minimum stage latency that is 44% faster and consumes 32% less energy per cycle than the full C-element version.

In coarse-grained pipelines the timing constraints imposed by the bounded-delay datapath primarily determine the maximum performance, with the overhead of the controller having little effect. However, in very fine-grained pipelines such as LC dynamic logic designs and FIFOs, the overhead from the control circuitry is significant and the maximum performance of the controller can become the limiting factor. As shown in Table I, LC control extends the point at which the controller becomes the limiting factor. The multiplier that

¹Assuming a PMOS/NMOS width ratio of two for a symmetrical inverter design.

²This process has a fanout-of-four inverter delay of 264 ps under the conditions used in the simulations.

TABLE I
CONTROLLER COMPARISON

Controller	Throughput (MHz)	Latency (ps)	Energy per cycle (pJ)
Locally-Clocked	910	300	97
Full C-element	590	540	142

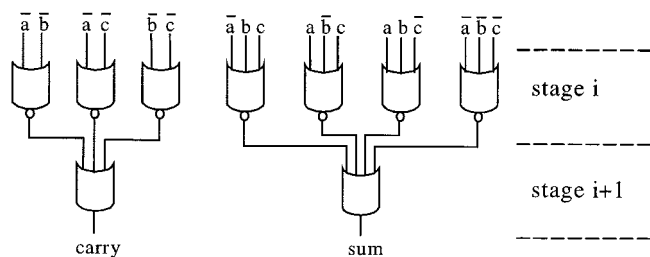


Fig. 6. Two-stage NOR/OR full adder implementation.

will be presented next is an example of where LC control extends the maximum performance of a design beyond that which is obtainable with the delay-insensitive controller.

An 8 × 8 bit multiplier using a carry-save array architecture [13] was designed to examine the performance of the LC control scheme when used with LC dynamic logic gates. The full adder cells used in the carry-save array were implemented using two stages of OR and NOR gates as shown in Fig. 6. The two stage OR/NOR implementation takes advantage of LC dynamic logic’s ability to implement very efficient wide OR/NOR structures. This minimized the delay of the slowest gate at each level to 475 ps, which maximized the sustainable throughput.

Hspice simulations of extracted results from the layout of the 8 × 8 bit multiplier show that the design is capable of operating at up to 715 MHz when a 10% margin is added to the matching delay constraints. If this pipelined multiplier was controlled by the full C-element controller the maximum sustainable frequency would be limited by the controller to 590 MHz. To the best of our knowledge, the highest frequency multiplier previously published was 625 Mhz [14]. It also used a carry-save array architecture that was pipelined at the half bit level and was implemented in a faster 0.8μ process, but the fanout-of-four inverter delay for this process was not provided. Also, the fall-through latency of the 8 × 8 bit LC dynamic logic multiplier is 15.3 ns and the block used 3.4 mm² of silicon.

VI. CONCLUSION

This paper presented a new control scheme, called LC control that can be used to moderate a level-sensitive linear pipeline. In a quest

for improved performance, LC control departs from a delay-insensitive full handshake protocol in favor of a bounded-delay model. Simulations show that the LC controller consumes 32% less energy and has a maximum frequency 54% greater than a delay-insensitive implementation of the same scheme. A new highly pipelined dynamic logic technique called LC dynamic logic was presented. It uses latching dynamic logic gates in conjunction with the LC control scheme to provide a fine grained, high throughput, low latency circuit solution. LC dynamic logic was used to implement an 8×8 bit multiplier in a $1\text{-}\mu\text{m}$ process that has a maximum sustainable throughput of 715 MHz.

ACKNOWLEDGMENT

The authors would like to thank D. Harris and other members of the asynchronous group at Sun Laboratories for sharing their knowledge.

REFERENCES

- [1] S. Hauck, "Asynchronous design methodologies: An overview," *Proc. IEEE*, vol. 83, pp. 69–93, Jan. 1995.
- [2] I. Sutherland, "Micropipelines," *Commun. ACM*, vol. 32, pp. 720–738, June 1989.
- [3] S.-L. Lu, "Implementation of micropipelines in enable/disable CMOS differential logic," *IEEE Trans. VLSI Syst.*, vol. 3, pp. 338–341, June 1995.
- [4] S. B. Furber and P. Day, "Four-phase micropipeline latch control circuits," *IEEE Trans. VLSI Syst.*, vol. 4, pp. 247–253, June 1996.
- [5] T. Meng and R. W. Broderson, "A fully asynchronous design for programmable digital signal processors," *IEEE Trans. Signal Processing*, vol. 39, pp. 939–952, Apr. 1991.
- [6] S. M. Burns, "Performance analysis and optimization of asynchronous circuits," Ph.D dissertation, California Inst. Technol., Pasadena, CA, Dec. 1990.
- [7] T. Chu, "Synthesis of self-timed VLSI circuits from graph-theoretic specifications," in *Proc. Int. Conf. Comput. Design*, Rye Brook, NY, Oct. 1987, pp. 220–223.
- [8] M. Renaudin, B. Hassan, and A. Guyot, "A new asynchronous scheme: Application to the design of a self-timed divider ring," *IEEE J. Solid-State Circuits*, vol. 31, pp. 1001–1013, July 1996.
- [9] T. Williams and M. Horowitz, "A zero-overhead self-timed 160-ns 54-b CMOS divider," *IEEE J. Solid-State Circuits*, vol. 26, pp. 1651–1661, Nov. 1991.
- [10] J. Yuan and C. Svensson, "High-speed CMOS circuit technique," *IEEE J. Solid-State Circuits*, vol. 24, pp. 62–70, Feb. 1989.
- [11] G. Yee and C. Sechen, "Clock-delayed domino for adder and combinational logic design," in *Proc. Int. Conf. Computer Design*, Austin, TX, Oct. 1996, pp. 332–337.
- [12] I. Sutherland and R. Sproul, "Logical effort: Designing for speed on the back of an envelope," in *Proc. Conf. Advanced Research in Very Large Scale Integration Systems*, Santa Cruz, CA, Mar. 1991.
- [13] G. Hoyer, G. Yee, and C. Sechen, "Locally-clocked dynamic logic," in *Proc. IEEE Midwest Symp. Circuits Syst.*, Notre Dame, IN, Aug. 1998, pp. 18–21.
- [14] D. Gosh and S. K. Nandy, "Design and realization of high performance wave-pipelined 8×8 b multiplier in CMOS technology," *IEEE Trans. VLSI Syst.*, vol. 3, pp. 36–48, Mar. 1995.