Wave-Pipelining: A Tutorial and Research Survey

Wayne P. Burleson, *Member, IEEE,* Maciej Ciesielski, *Senior Member, IEEE,* Fabian Klass, *Associate Member, IEEE,* and Wentai Liu, *Senior Member, IEEE*

***Abstract*— Wave-pipelining is a method of high-performance circuit design which implements pipelining in logic without the use of intermediate latches or registers. The combination of high-performance integrated circuit (IC) technologies, pipelined architectures, and sophisticated computer-aided design (CAD) tools has converted wave-pipelining from a theoretical oddity into a realistic, although challenging, VLSI design method. This paper presents a tutorial of the principles of wave-pipelining and a survey of wave-pipelined VLSI chips and CAD tools for the synthesis and analysis of wave-pipelined circuits.**

***Index Terms*—Performance optimization, VLSI circuits, wave-pipelining.**

I. Introduction

WAVE-PIPELINING is an example of one of the many methods currently being used in sophisticated VLSI designs. As an alternative to pipelining, it provides a method for significantly reducing clock loads and the associated area, power and latency while retaining the external functionality and timing of a synchronous circuit. It is of particular interest today because it involves design and analysis across a variety of levels (process, layout, circuit, logic, timing, and architecture) which characterize VLSI design. However it also questions some of the fundamental tenets of simplified VLSI design as popularized in the early 1980's.

The idea of wave-pipelining was originally introduced by Cotten [6], who named it *maximum rate pipelining*. Cotten observed that the rate at which logic can propagate through the circuit depends not on the longest path delay but on the difference between the longest and the shortest path delays. As a result, several computation "waves," i.e., logic signals related to different clock cycles, can propagate through the logic simultaneously. One can also view the wave-pipelining as a virtual pipelining, in which each gate serves as a virtual storage element.

In an attempt to understand the wave-pipelining phenomenon and turn it into a useful and reliable computer technology, research focused on the following aspects: 1) developing correct timing models and analyzing the problem mathematically [36], [8], [9], [17], [11], [12], [26], 2) developing logic synthesis techniques and computer-aided design (CAD) tools for wave-pipelined circuits [41], [43], [16], [20], [21], [37], 3) developing new circuit techniques specifically devoted to wave-pipelining [27], and 4) testing the wave-pipelining ideas by building VLSI chips [27], [42], [22]. A comparative study of existing methods in wave-pipelining can be found in [23].

Wave-pipelining has suffered from a number of myths and the problems which need to be solved in a wave-pipelined design are not widely understood. This paper presents a tutorial with examples to demonstrate the type of design problems which arise in wave-pipelining. In Section II timing constraints are derived which ensure the proper operation of a wave-pipelined circuit. Section III discusses the sources of delay variation which affect the timing constraints and presents methods for minimizing their impact. Section IV reviews CAD tools available for synthesizing wave-pipelined circuits and demonstrates their use with two example circuits. Section V reviews a number of industrial and academic designs which employ wave-pipelining. Section VI poses some open research problems in wave-pipelining and related areas.

The impact of the paper is broader than only wave-pipelining as the methods of analysis and synthesis apply to many other aggressive timing and circuit techniques being used in industry today. We anticipate that in the future this type of VLSI design techniques will be required to maintain the steady increases in performance improvement to which we have become accustomed. VLSI designers, CAD developers and system designers need to be aware of these trends and their future impact on design, manufacture, testing and education in microelectronics.

II. Clocking of Wave-Pipelined Circuits

The timing requirements of wave-pipelined circuits will be defined for a single combinational logic block with registers attached to its inputs and outputs. In the case of multistage pipelines, the timing constraints must hold for all stages. In the sequel, the term *conventional pipelining* will be used to refer to a pipeline where only a single set of data propagates between registers at any given time. The term *constructive clock-skew* will be used to refer to a clock-skew that is intentionally created between two clock signals and that can be adjusted with predictable effects. This is in contrast to an *uncontrolled clock-skew* that exists in the circuit due to delay differences along the clock lines.

Parameters listed below will be used in the derivation of the timing constraints. These parameters are defined under worst case conditions, including manufacturing tolerances, data-dependent delays, and environmental changes.

Manuscript received February 12, 1996; revised October 30, 1997. This work was supported by the National Science Foundation under Grant MIP-9208267.

W. P. Burleson and M. Ciesielski are with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003 USA.

F. Klass is with the SUN Microsystems Inc., Sunnyvale, CA 90095 USA.

W. Liu is with the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh NC 27607 USA.

Publisher Item Identifier S 1063-8210(98)05977-0.

1063–8210/98$10.00 © 1998 IEEE

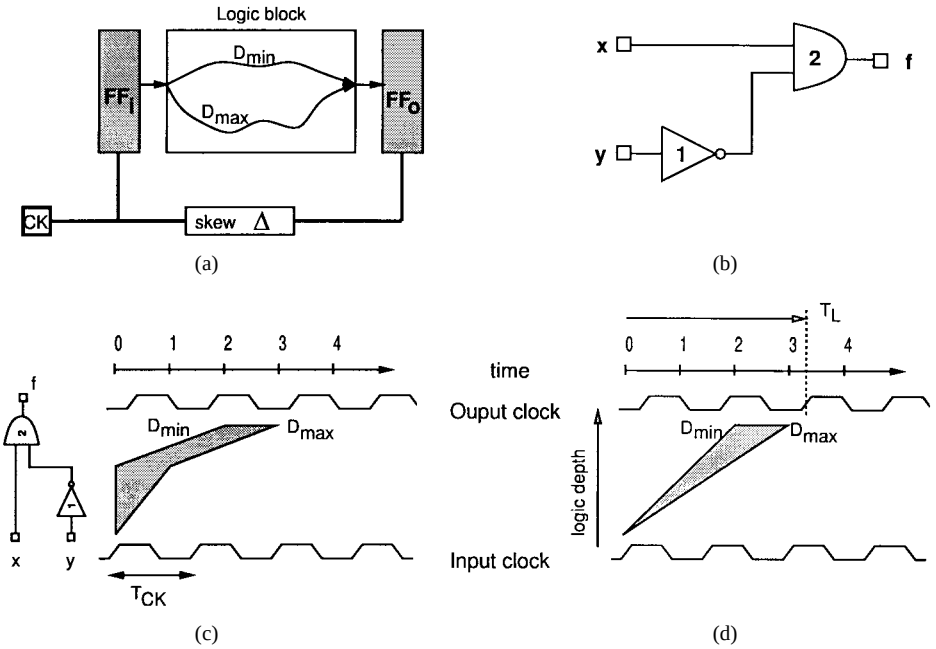


Fig. 1. Model of wave-pipelined circuit.

- $D_{\mathrm{MIN}}, D_{\mathrm{MAX}}$ Minimum and maximum propagation delays in the combinational logic block.
- T_{CK} Clock-period.
- T_S, T_H Register setup and hold times.
- D_R Propagation delay of a register.
- Δ Constructive clock skew between the output and input registers.
- Δ_{CK} Worst case uncontrolled clock skew at a register.

Our analysis assumes that all the input and output registers have the same setup time T_S, hold time T_H, and propagation delay D_R. For simplicity and without much loss of generality, we shall only consider edge-triggered registers. Timing constraints for other types of registers, including transparent latches, can be similarly derived.

A. Circuit and Timing Models

The phenomenon of wave-pipelining can be illustrated using the timing model presented in [12], shown in Fig. 1. A synchronous logic block is clocked by a set of input and output registers, with the constructive clock skew equal to Δ, Fig. 1(a). Associated with each block is a pair of minimum and maximum propagation delays, D_{MIN}, D_{MAX}. These parameters are the delays of the earliest and the latest signals propagating through the logic along its shortest and longest paths, respectively.

The spread of signals traveling along the longest and the shortest paths through the logic block can be visualized as *delay contours*. The shaded area between the contour lines represents an unstable region when signals switch, i.e., when the computation is being performed by the logic block. For a simple logic network in Fig. 1(b) the delay contours are shown in Fig. 1(c). The exact contours depend on the logic and the delay characteristics of its elements. Following [12], we shall simplify the delay contours by linearizing the delays along the logic block to obtain delay "cones," as shown in Fig. 1(d).

Fig. 2 depicts a combined temporal and spatial diagram for a wave-pipelined circuit. The horizontal axis represents time, while the vertical axis represents the *logic depth* of the circuit. A point on the logic depth axis represents a node (gate) of the circuit. The shaded regions denote the time intervals when computation is being performed and data is unstable, while the space between two adjacent computation regions corresponds to stable data. The computation regions are labeled with the clock cycle during which the data has been latched at the input register. It can be observed that at any given reference time, t_{ref}, several "waves" of computation can present in the logic simultaneously.

For correct clocking, the output data must be sampled at time T_L at which data is stable, i.e., T_L must fall in the nonshaded area. Furthermore, at any internal node of the network, there must be a minimum separation between the arrival time of the latest signal of a given wave, and the earliest signal of the next wave. The temporal separation between the waves at an internal node x is represented in the figure by T_{SX}.

B. Timing Constraints

For a wave-pipelined system to operate correctly, the system clocking must be such that the output data is clocked *after* the latest data has arrived at the outputs and *before* the earliest data from the *next* clock cycle arrives at the outputs. We shall first derive the conditions for clocking of the latest and the earliest data propagating in the circuit, referred to as the *register constraints*. We introduce parameter N, which represents the number of clock cycles needed for a signal to propagate through the logic block before being latched by the output register. This parameter serves as an intuitive measure of the *degree of wave-pipelining*. The data should be clocked

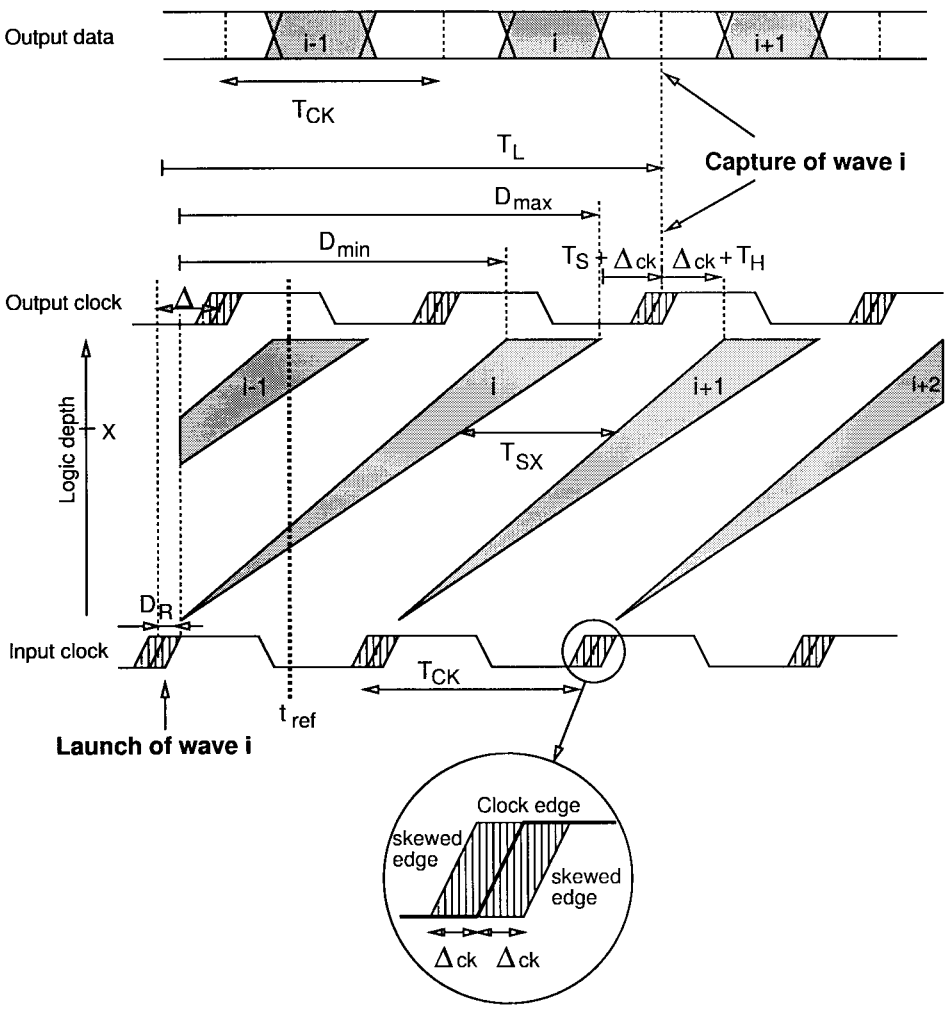


Fig. 2. Temporal/spatial diagram for wave-pipelining system.

at time T_L by the rising edge of the output register N clock cycles after it has been clocked by the input register. Due to possible constructive skew Δ (of arbitrary value) between the output and the input registers, this time can be expressed as

$$T_L = NT_{\mathrm{CK}} + \Delta. \tag{1}$$

1) Register Constraints:

a) Clocking of the latest data: This constraint requires that the latest possible signal arrives early enough to be clocked by the output register during Nth clock cycle. Therefore, the lower bound on T_L, which denotes the time at which the output wave is captured, is given by

$$T_L > D_R + D_{\mathrm{MAX}} + T_S + \Delta_{\mathrm{CK}}. \tag{2}$$

b) Clocking of the earliest data: This condition requires that the arrival of the *next* wave must not interfere with the clocking of the current wave. That is, the earliest possible signal of wave $i+1$ must arrive later than the clocking of the ith wave at the output register. This condition is similar to the *race-through* constraint in conventional pipelining. Notice that the earliest arrival of wave $i+1$ is given by $T_{\mathrm{CK}}+D_R+D_{\mathrm{MIN}}$. After the clock pulse has been applied to the output register, additional hold time T_H must be allowed for the data to remain steady. In addition, one must account for an uncontrolled clock-skew Δ_{CK} at the output register. As a result, the T_L is bounded above as follows:

$$T_L < T_{\mathrm{CK}} + D_R + D_{\mathrm{MIN}} - (\Delta_{\mathrm{CK}} + T_H). \tag{3}$$

Combining constraints (2) and (3) gives us the well-known *maximum rate pipelining* condition of Cotten

$$T_{\mathrm{CK}} > (D_{\mathrm{MAX}} - D_{\mathrm{MIN}}) + T_S + T_H + 2\Delta_{\mathrm{CK}}. \tag{4}$$

The minimum clock period is limited by the difference in path delays $(D_{\mathrm{MAX}} - D_{\mathrm{MIN}})$, plus the *clocking overhead* $(T_S + T_H + 2\Delta_{\mathrm{CK}})$ resulting from the insertion of clocked registers.

2) Internal Node Constraints: Constraint (4) guarantees that waves do not collide, or overlap, at the output of the logic block. Additional constraints need to be imposed at the internal nodes of the combinational block to prevent wave collision at the individual logic gates. These signal separation constraints can be informally expressed as follows:

> The next earliest possible wave should not arrive at a node until the latest possible wave has propagated through.

These constraints, also known as *internal node constraints*, were derived, in slightly different forms, by Ekroot [8], Wong *et al.* [43], Joy *et al.* [17], and Gray *et al.* [12].

Let x be an internal node (output of a gate) of the logic network, a point on the logic depth axis in Fig. 2. To help derive the internal node constraint we define $d_{\rm MAX}(x)$ and $d_{\rm MIN}(x)$ as the longest and the shortest propagation delays from the primary inputs to node x. The following internal node constraint that must be satisfied at each node x of the circuit:

$$T_{\rm CK} \geq d_{\rm MAX}(x) - d_{\rm MIN}(x) + T_{SX} + \Delta_{\rm CK} \tag{5}$$

where T_{SX} is the minimum time that node x must be stable to correctly propagate a signal through the gate, and $\Delta_{\rm CK}$ is the worst case uncontrolled clock skew at the input register. Notice the similarity between constraint (5) and (4). While $(d_{\rm MAX}(x) - d_{\rm MIN}(x))$ is equivalent to $(D_{\rm MAX} - D_{\rm MIN})$, the term T_{SX} is equivalent to $T_S + T_H$ (the minimum sampling time for the data). The factor of two in the clock-skew term is not present here since the signals are not stored (clocked) at the internal nodes.

C. Intervals of Valid Clocking

Based on the conditions presented in the previous section, a linear program (LP) can be readily formulated to find a minimum value of clock period for a given value of N (number of waves). The LP approach minimizes $T_{\rm CK}$ subject to both register constraints (2), (3) and the internal node constraints (5) for all nodes in the circuit. This approach, taken by Fishburn [9] and Joy *et al.* [17], seems to suggest that the circuit will operate correctly at any value of clock cycle above the computed minimum, that is, that the region of valid clock period is only bounded below. However, as independently demonstrated by Lam *et al.* [26] and by Gray *et al.* [12], the feasibility region of the valid clock period $T_{\rm CK}$ is not continuous but is composed of a finite set of disjoint regions. Furthermore, the degree of wave-pipelining, N, does not change monotonically with the increase in clock frequency.

To see why this is the case we shall examine again register constraints (2) and (3) to obtain a two-sided constraint on the clock period. Recalling that T_L, the latching time of the data at the output register can be expressed as $T_L = NT_{\rm CK} + \Delta$, we obtain the following inequality (refer to Fig. 2):

$$\begin{aligned} D_R + D_{\rm MAX} + T_S + \Delta_{\rm CK} < NT_{\rm CK} + \Delta < T_{\rm CK} \\ + D_R + D_{\rm MIN} - (\Delta_{\rm CK} + T_H). \end{aligned} \tag{6}$$

To simplify the interpretation of the above relation let us introduce two parameters $T_{\rm MAX}$ and $T_{\rm MIN}$

$$T_{\rm MAX} = D_R + D_{\rm MAX} + T_S + \Delta_{\rm CK} - \Delta \tag{7}$$

represents the maximum delay through the logic, including clocking overhead and clock skews, while

$$T_{\rm MIN} = D_R + D_{\rm MIN} - \Delta_{\rm CK} - T_H - \Delta \tag{8}$$

represents minimum delay through the logic. With this, (6) can be expressed as follows:

$$\frac{T_{\rm MAX}}{N} < T_{\rm CK} < \frac{T_{\rm MIN}}{N-1}. \tag{9}$$

Analyzing the above result reveals that the feasibility region of clock period $T_{\rm CK}$ may not be continuous. For $N = 1$ the clock period is only bounded below by $T_{\rm MAX}$, as in the conventional pipelining. For larger values of N, however, it shows that the period is also bounded above by $\frac{T_{\rm MIN}}{N-1}$. In addition, if $T_{\rm MAX} > T_{\rm MIN}$, then any two successive intervals $[\frac{T_{\rm MAX}}{N+1}, \frac{T_{\rm MIN}}{N}]$ and $[\frac{T_{\rm MAX}}{N}, \frac{T_{\rm MIN}}{N-1}]$ are disjoint. For a given value of N the size of the interval which represents valid clock period grows with the increase of $T_{\rm MIN}$ and the decrease of $T_{\rm MAX}$.

D. Timed Boolean Function Approach

An alternative approach to clocking analysis of wave-pipelined systems, based on *timed Boolean function* representation, was proposed by Lam *et al.* [26]. A Timed Boolean Function (TBF) is a Boolean function whose variables are functions of time. Consider a TBF for output y_j of the circuit in terms of its input variables $x(t)$

$$y_j(t) = f(\cdots, x_i(t - d_{ij}), \cdots) \tag{10}$$

where d_{ij} is a logic delay from input x_i to output y_j. The value of this function at time $t = kT_{\rm CK}$, normalized w.r. to, and sampled with the clock period $T_{\rm CK}$ is

$$\begin{aligned} y_j(kT_{\rm CK}) &= f\left(\ldots, x_i\left(k - \left\lceil \frac{d_{ij}}{T_{\rm CK}} \right\rceil\right), \ldots\right) \\ &= f(\cdots, x_i[k - N_{ij}], \cdots) \end{aligned} \tag{11}$$

where $N_{ij} = \lceil \frac{d_{ij}}{T_{\rm CK}} \rceil$ is the number of clock cycles needed to propagate a signal from input i to output j. Now the condition for valid computation that satisfies internal node constraints can be expressed as follows [26]: computation is valid if and only if all variables x_i in the support of y_j, for all j, have the same time argument $[k - N]$, i.e., if $f(kT_{\rm CK}) = f[k - N]$. In this case the latency of the circuit, measured in the number of clock cycles is the same as the degree of wave-pipelining, N. Combining the above constraint with (9), results in the following theorem for valid clock period [26]:

> The clock period $T_{\rm CK}$ in a circuit is valid if it satisfies (9) for some positive integer N, and the function computed at $t = kT_{\rm CK}$ evaluates to $f[k - N]$.

TBF approach provides a convenient way to express the condition for safe signal separation both at the internal nodes of the circuit and at the output register. The internal node constraints are simply accounted for by adding the requirement that all variables in the support of the logic function have the same discrete argument $[k - N]$. Lam *et al.* [26] describe a procedure to compute all valid clock intervals for a circuit with given delay parameters and tolerances. The reader is referred to their paper for details.

III. Sources of Delay Variations

As discussed in Section II, critical speed-limiting factors in wave-pipelining are the uncontrolled clock-skew, the sampling time of registers, and the worst case transition time at the logic outputs. While the minimization of these factors has

been a major challenge in the design of conventional high-speed pipelined systems as well, the equalization of path delays comes as a new challenge for the design of wave-pipelined systems. While in theory the path-delay equalization problem has been solved, the real challenge is to accomplish it in the presence of a variety of static and dynamic delay tolerances, some of which are listed below.

1) *Gate-Delay Data-Dependence:* Gate-delay independence on the input pattern, i.e., constant gate delay, is not guaranteed in general. It depends on the particular technology and the structure of logic gates. Some input patterns may cause significant delay variations.
2) *Coupling Capacitance Effects:* The effective capacitance seen by a gate depends on the capacitive coupling between adjacent wires. This may introduce significant changes in the gate delay, especially in advanced multilevel metal processes.
3) *Power-Supply Induced Noise:* Power supply noise is attributed mainly to IR drops, capacitive coupling between the interconnection wires and the power supply lines, and the inductance of bonding wires, package trace, and pins. Noise in the power-supply voltage can cause accumulative delay dispersions as waves propagate through several logic layers.
4) *Process Parameter Variations:* Variations of process parameters during manufacture may result in substantial gate delay variations. Equivalent circuits from different fabrication runs may have different propagation delays. This delay dispersion must be accounted for to establish the minimum separation between waves.
5) *Temperature-Induced Delay Changes:* Some process parameters, such as carrier surface mobility of metal–oxide–semiconductor (MOS) transistors, are highly thermally sensitive. Changes in the operational temperature result in changes in the gate-delay.

A. Data Dependencies

One of the major obstacles initially found in static complementary metal–oxide–semiconductor (CMOS) is the strong dependence of the gate-delay on the input data. Consider for example a static two-input CMOS NAND gate. Depending on whether one or both of the parallel PMOS devices are switching on, the delay of the gate can vary by a factor of two. For gates with several inputs, this factor is even larger. Clearly, this feature is undesired for wave-pipelining; constant logic gate delay is a requirement for the equalization of path-delays. To overcome this problem the use of biased CMOS gates, also known as pseudo-NMOS gates, has been proposed. In those devices parallel pullup devices are replaced by a single device whose gate is connected to a bias voltage [28]. While this approach reduces the delay dependence problem and makes CMOS better suited for wave-pipelining, it is achieved at the expense of increased power dissipation.

B. Process and Environmental Delay Variations

Changes in temperature, supply levels, and process parameters can have a substantial effect on the delay of a CMOS circuit. How such changes affect the operation of wave-pipelining is discussed below. Although the following analysis is for temperature, it applies to process and power supply changes as well.

As shown in Section II-C, the clock-period T_{CK} and the number of waves N in a wave-pipelined circuit are related by (9) which can be rearranged as

$$T_{\mathrm{MAX}} < NT_{\mathrm{CK}} < T_{\mathrm{MIN}} + T_{\mathrm{CK}} \tag{12}$$

where $T_{\mathrm{MAX}}, T_{\mathrm{MIN}}$ are specified at nominal temperature.

We assume that the temperature across the die area is uniform and that every path delay in the circuit is affected in the same way by changes in temperature. If, for a temperature above the nominal, T_{MAX} and T_{MIN} are increased by a factor $\beta_S > 1$, constraint (12) becomes

$$\beta_S T_{\mathrm{MAX}} < NT_{\mathrm{CK}} < \beta_S T_{\mathrm{MIN}} + T_{\mathrm{CK}}. \tag{13}$$

If, for a temperature below the nominal, T_{MAX} and T_{MIN} are reduced by a factor $\beta_F < 1$, we have

$$\beta_F T_{\mathrm{MAX}} < NT_{\mathrm{CK}} < \beta_F T_{\mathrm{MIN}} + T_{\mathrm{CK}}. \tag{14}$$

In order to find a valid clock-period that satisfies constraints (12), (13), and (14), it is required that

$$\beta_S T_{\mathrm{MAX}} < NT_{\mathrm{CK}} < \beta_F T_{\mathrm{MIN}} + T_{\mathrm{CK}} \tag{15}$$

or equivalently as

$$T_{\mathrm{CK}} > \beta_S T_{\mathrm{MAX}} - \beta_F T_{\mathrm{MIN}}. \tag{16}$$

Equation (16) can be interpreted as follows. The clock period is limited by the difference between T_{MAX} at maximum temperature $(\beta_S T_{\mathrm{MAX}})$ and T_{MIN} at minimum temperature $(\beta_F T_{\mathrm{MIN}})$. This is the minimum clock period that assures correct operation in the entire range between the two temperatures. (Notice that the use of a fixed zero constructive clock skew still results in invalid intervals for T_{CK}.)

In conclusion, in addition to the limit imposed by the maximum difference $T_{\mathrm{MAX}} - T_{\mathrm{MIN}}$, the process- and environment-induced delay changes impose tighter constraints on the minimum clock-period (or the maximum number of waves) for wave-pipelining. Notice that even in the ideal case of $T_{\mathrm{MAX}} = T_{\mathrm{MIN}}$ (i.e., under the perfect delay equalization and zero clocking overhead assumptions), the clock-period is still limited by $T_{\mathrm{CK}} > (\beta_S - \beta_F) T_{\mathrm{MAX}}$. While in conventional pipelining the minimum clock-period can be specified based on the worst case delay accounting for temperature, voltage, and process, in wave-pipelining the best case delay must also be considered. The consequence is a more substantial performance degradation for wave pipelined designs.

IV. CAD Tools for Wave-Pipelining

Equations (4) and (9) derived in Section II suggest that the minimum value of clock period as well as the range of valid clock interval can be adjusted by modifying certain parameters of the system. The following parameters can be controlled by the designer: maximum and minimum delays $(D_{\mathrm{MAX}}, D_{\mathrm{MIN}})$

of the combinational logic block, and the constructive clock skew, Δ, between the output and input registers. Two techniques are generally adopted to control these parameters. One, referred to as a *logic balancing*, attempts to minimize the difference $D_{\mathrm{MAX}}-D_{\mathrm{MIN}}$. It involves logic restructuring, delay buffer insertion along short paths, and device sizing. The other technique, *clock buffer insertion*, aims at adjusting the value of constructive clock skew Δ by inserting precisely controlled delays along the clock paths.

This section briefly reviews some of the most popular techniques and practical CAD tools for the analysis, optimization, and synthesis of wave-pipeline systems.

A. Timing Analysis and Optimization

In addition to the theoretical work on modeling and analysis of wave-pipelining presented in Section II, a number of timing analysis and verification tools for synchronous pipelined and wave-pipelined systems have been developed. A notable example of such a tool is a pipeline scheduler, *pipe*T_C, developed by Sakallah *et al.* [36]. This work provides a unified formalism for describing the timing in pipelines, accounts for multiphase synchronous clocking, and handles both short and long path delays. The tool generates correct minimum cycle-time clock schedules and signal waveforms from a multiphase pipeline specification. However, wave-pipelining and clock skews are not taken into account explicitly.

A number of timing optimization tools were developed to minimize clock period by adjusting constructive clock skews. These techniques are based on the LP approach proposed by Fishburn [9] and Joy *et al.* [16], mentioned in Section II-C.

B. Synthesis Tools

Most of the work in synthesis for wave-pipelining explores the idea of minimizing the difference in logic path delays by means of *logic balancing*. The following approaches are typically used to achieve logic balancing: logic restructuring, insertion of delay buffers or latches, device sizing, and controlled placement and routing of both the circuit components and clock distribution trees.

Wong *et al.* [41], [43] developed a method and a CAD tool to minimize path delays variance in ECL circuits by inserting delay buffers followed by device fine-tuning. The delay of each gate is fine-tuned by controlling its tail current, a feature unique to ECL circuits. This method was tested on circuits with regular structures, such as adders and multipliers, achieving a factor of 2.5 increase in throughput at a cost from 10 to 50% increase in area [42]. Shenoy *et al.* [37] expressed the logic balancing problem as an optimization procedure with additional short path constraints. This technique was implemented as an experimental CAD tool that interfaces with the SIS logic synthesis system [38].

Both of these approaches aim at obtaining the maximum rate pipelining with the use of minimum number of delay buffers. However, for multiple-fan-out gates the delay buffers are inserted individually for each fan-out. As a result a separate step is typically required to merge the common buffers for area recovery. While this is a simple task for ECL logic (delay of an ECL gate does not depend on the fan-out), the fan-out load of CMOS gates significantly affects the delay, and its impact on timing cannot be ignored. For this reason, Kim *et al.* [21] developed a delay buffer insertion method using chains of buffers, thus eliminating the need for merging common buffers. The remainder of this section briefly describes logic balancing for CMOS circuits based on logic restructuring and buffer insertion of Kim [21]. The algorithms have been implemented in the SIS environment [38].

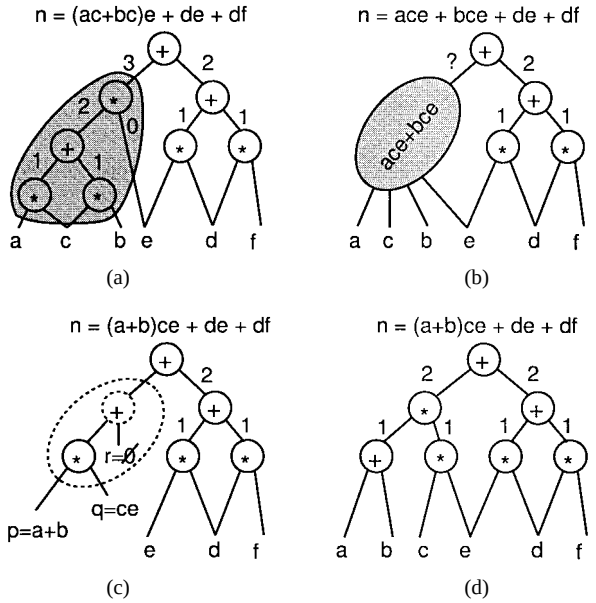


Fig. 3. Node collapsing and decomposition.

1) Logic Restructuring: The goal of logic restructuring is to equalize signal arrival times at the inputs of each gate. To facilitate delay estimation during logic restructuring the circuit is first decomposed into "canonical" form composed of two-input gates and inverters. It is then followed by selective *node collapsing*, and recursive *decomposition*. Node collapsing is a standard logic transformation technique which combines several nodes of a logic network into a single node. It facilitates the subsequent transformations, such as recursive decomposition, which will transform the function into a network with a desired timing property. This process is illustrated in Fig. 3(a) and (b) where node with delay of three units is collapsed into its fan-in nodes, creating a single node with four inputs. The subsequent decomposition transformation of the node creates a more balanced structure.

The *decomposition* of the collapsed nodes is accomplished using *kernel division* technique employed in SIS [38]. The goal is to find a decomposition that minimizes the difference between latest arrival times at the inputs to the collapsed node. This is accomplished by computing, for a given expression n, all its multiple-cube subexpressions (*kernels*) and selecting the one which leads to the most balanced structure. By estimating the delay of the kernel p, and the resulting quotient q and the remainder r, the kernel which gives the best balancing of the expression is selected. The kernel-based decomposition is applied recursively to nodes p, q, and r until no further improvement is possible. Fig. 3(c) and (d) shows the result of

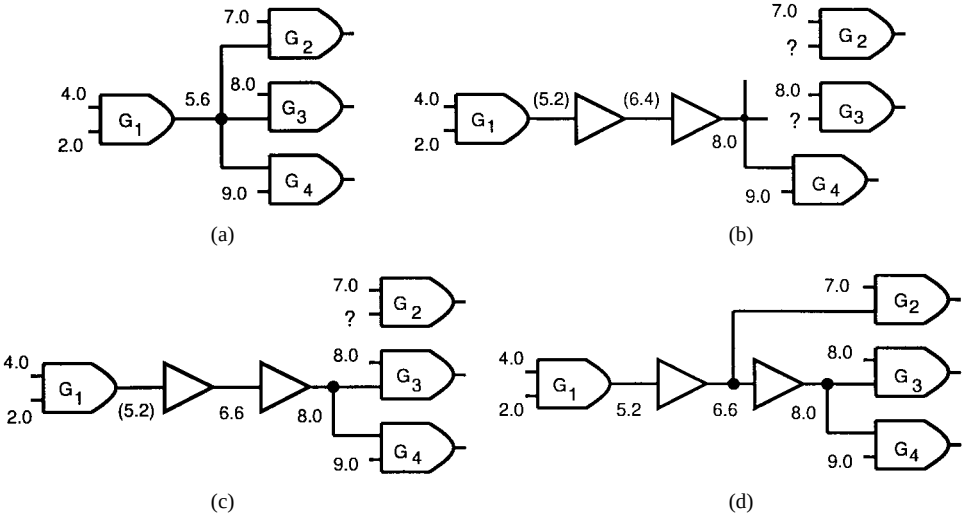


Fig. 4. Example of delay buffer insertion.

such a decomposition into two-input gates. The details of the procedure can be found in [20].

2) Delay Buffer Insertion: Once the circuit is restructured, additional balancing can be achieved by means of buffer insertion. Delay buffers are inserted in appropriate places so as to further minimize the difference in signal arrival times at the gate inputs.

The underlying requirement imposed on buffer insertion is that it must not affect the latency of the circuit. For this reason the buffers are inserted only along fast paths, without affecting the slow ones. The amount of the delay to be inserted at an input to a gate is chosen so that the arrival time at that input matches the arrival time of the slowest input, but does not exceed it. As a result, the latest arrival time at the output of the gate never increases as a result of buffer insertion. This approach allows to localize the buffer insertion problem to that of inserting a single chain of buffers at the output of each gate, wherever needed. The desired delays at the fan-outs of the gate are then obtained by providing taps at different stages of the buffer chain. It is assumed that only one type of buffer, with a fixed delay value, is available.

Fig. 4 illustrates the process of buffer insertion for the three gates fanning out of G_1. For simplicity it is assumed that each buffer contributes 1 unit delay, and the load at each fan-out contributes 0.2 unit delay. The numbers given at the inputs to the gates represent the latest signal arrival times. First, a chain of two buffers is created to achieve the input delay of 8.0 units at G_4. The input to G_3 is then tapped off the end of the chain to match the arrival time (8.0) of its other input, while the input to G_2 is tapped of the first stage of the chain, which provides the delay of 6.6. Notice that the arrival time at the end of the chain (8.0) is independent of how the inputs to G_2 and G_3 are connected to the chain. The details of the procedure, including the proper ordering of the gates for buffer chain construction, can be found in [21].

C. Synthesis Examples

To validate the logic synthesis approach described in the previous sections a number of combinational circuits were synthesized and tested. These included both regular arithmetic circuits and a set of random logic circuits from the MCNC benchmark set. This section presents two examples of circuits synthesized with those tools.

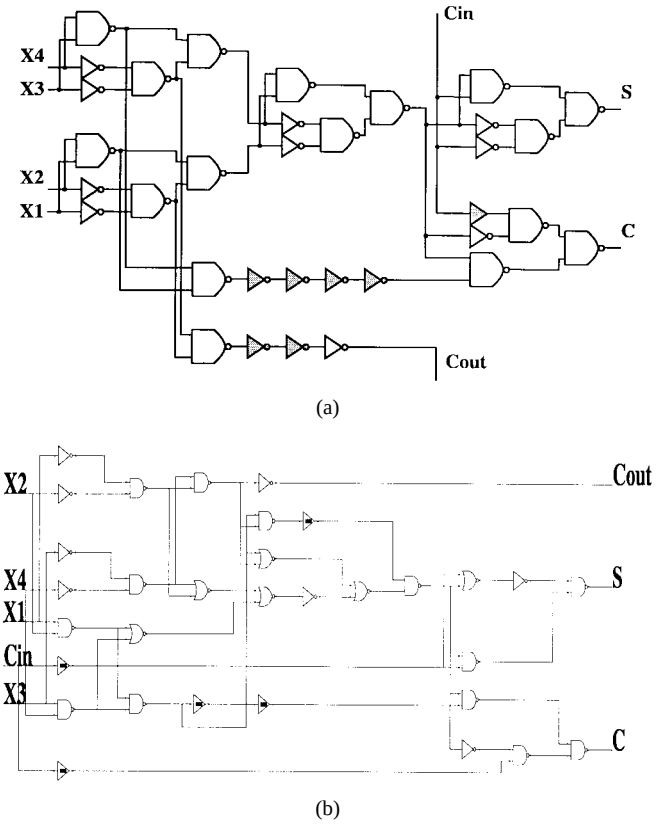


Fig. 5. (4,2) compressor circuit.

1) (4,2) Compressor: Fig. 5(a) shows a manual design of a (4,2) compressor circuit, used as a basic multiplier cell in [24]. Fig. 5(b) has a synthesized version of the same function, obtained with logic balancing tools described in Section IV-B.

A fair comparison of the two circuits is difficult to make since each was designed with a different goal in mind, used different processing technology and its performance was mea-

sured differently. Circuit (a) is part of a 16 × 16 multiplier, which was designed, balanced and tuned manually in order to obtain the fastest running circuit; it was fabricated using commercial 1 μm CMOS technology; and its path delays were measured by exhaustive HSPICE simulation. The maximum and minimum reported delays were 1.46 and 1.19 ns, respectively, resulting in a maximum delay variation of less than 20%. Circuit (b) was automatically synthesized to allow for degree of wave-pipelining equal to three; it was targeted for a generic 2 μm MOSIS cell library; finally, its delays were computed using a simple unit fan-out delay model. The circuit has delay of 2.18 ns and a latency of 6.6 ns (three waves). Notice its balanced logic structure and the inserted buffers.

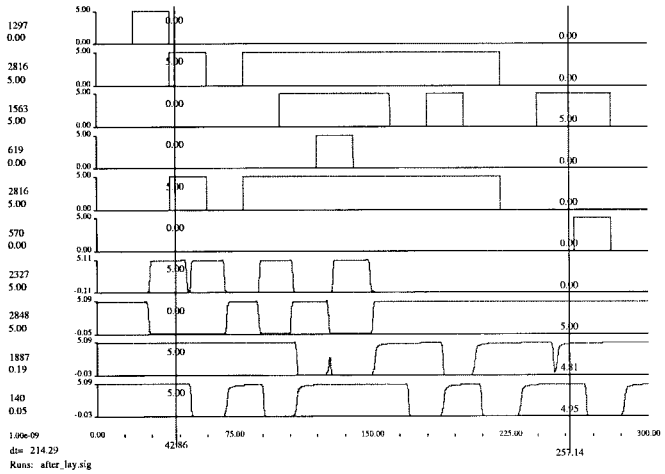
A simple-minded comparison of circuit areas based on gate count, and of latencies, based on unit fan-out delay, shows that synthesis tools can be used effectively to produce designs comparable with the manually tuned circuits.

2) Random Logic Example: The random logic circuits are characterized by a very irregular structure, and as such are not well suited for logic balancing. The example shown below (circuit *b9*, taken from the MCNC benchmark set) demonstrates that even for those circuits logic restructuring followed by buffer insertion can result in a significant improvement in circuit performance by means of wave-pipelining.

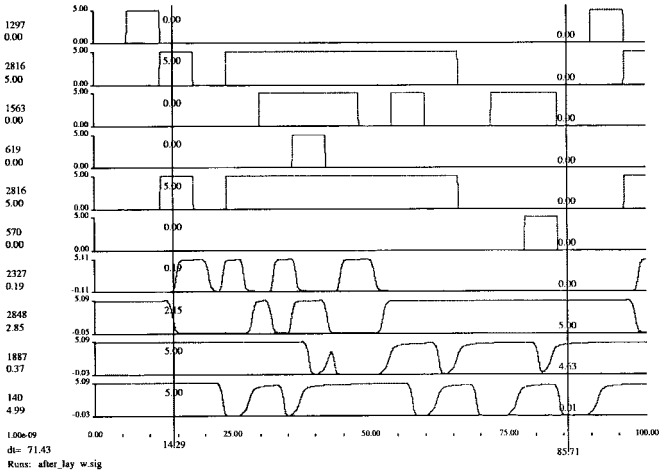
The circuit was automatically synthesized using the logic balancing procedures described in Section IV-B. Using basic two-input gates and inverters, and two types of delay buffers from the MSU standard cell library, gate delay parameters were constructed for the SIS library and the MOSIS 2μ p-well process. The circuit was synthesized to allow for three waves and laid out using standard cell design methodology. The simulated value of the minimum clock period of the circuit was 2.78 ns with latency 7.37 ns.

The latency of the circuit extracted from the layout was 11.39 ns with the degree of wave-pipelining equal to two. These differences were due to the unaccounted for physical effects of placement and routing. Considering the fact that the layout synthesis was performed without any consideration for timing optimization, it can be argued that obtaining the physical circuit satisfying the target constraints is possible with more sophisticated timing-driven layout design tools.

Fig. 6 shows the plot of CAzM [7] simulation result for circuit *b9*, with inputs applied (a) every 20 ns and (b) every 6 ns. Notice how the two waveforms, scaled accordingly to account for different clock period, match closely.



(a)



(b)

Fig. 6. CAzM simulation results for circuit *b9* for two different values of clock period.

V. Real Designs with Wave-Pipelining

Many attempts have been made at using wave-pipelining since Cotten proposed the technique in 1969 [6]. Prior to 1990, most designs of wave-pipelining systems targeted the bipolar ECL technology. These included a floating point unit [1], an experimental computer [35], and a population counter [42]. Since then, several university research groups have demonstrated the feasibility of CMOS implementations. In addition, industry has begun to apply wave-pipelining to RAM designs in both BiCMOS and CMOS technology. Table I lists selected publications in wave-pipelining. The research spans theoretical formulation, CAD tools research, and design projects that include dedicated processors and static/dynamic RAM's.

A. Dedicated Processors

The main hurdle for designing a wave-pipelined system is the two-sided constraints on the timing. In addition, designers must address the various sources of delay imbalance in the design at the gate level, the circuit level, and at the detailed layout level. The delay imbalance must also be minimized for environmental variations such as voltage and temperature fluctuations. Also, care must be given in the design of an on-chip test structure so that high-speed testing can be done without the requirement for high-speed input–output. This unique feature is employed by most of the university prototypes presented in this section. Researchers have devised various circuits to overcome the imbalances in successful implementations of wave-pipelined systems. Examples include wave-domino gates, biased-CMOS gates, static CMOS gates, and multiplexer-based gates.

TABLE I
ACTIVITIES RELATED TO WAVE-PIPELINING

Chiao-Tung U.	CMOS wave-pipelined multiplier [18]
Delft Univ	CMOS gates [22] and a 16 × 16 multiplier with Stanford [24] (CMOS, 3 waves)
DEC	L3 Cache Memory for AXP 21164 [2]
GTE	SONET-OC12 switch [5]. (CMOS, 4 waves)
HP	Cache RAM, HP Snake Workstation [25] (CMOS, 1.16 waves)
IBM	Bubbled Pipelined RAM [4] (CMOS, 4 waves)
IBM	Crosspoint switch [39] (ECL, 2 waves)
NEC	SRAM chip [31] (BiCMOS, 2 waves)
Hitachi	SRAM (CMOS, 330MHz and 2.6ns) [15], [40]
Hyundai	DRAM (CMOS, 150MHz, 256Mb) [44]
NCSU	WP theory for Multistages with feedback [12]
	16-bit adder (2 μm CMOS and 9 waves) [28]
	Digital sampler (1.2 μm CMOS, 25ps res.) [13]
	multiplier (1.2 μm CMOS, 4 waves) [32]
	(1.2 μm, 625 Mb/s) [19]
	Pattern generator and A/D converter [30]
Stanford	63-bit population counter [42] (ECL, 2.3 waves)
	16 × 16 multiplier (1.0 μm CMOS, 3.5 waves)
	vector unit (1 μm CMOS, 300 MHz) [33]
TI-India	8-bit multiplier (CMOS) [10]
Taiwan Univ.	Testing of wave-pipelined circuit [18]
U. Colorado	Optical computer design w/ wave-pipelining [34]
UC Berkeley	Single stage wave-pipelining theory [26]
U. Mass.	Wave-pipelining in CMOS domino logic [27]
	Wave-pipelining Theory [17]
	Wave-pipelining multiply-accumulator [3]
	Wave-pipelined multiplier and parity generator (2 μm CMOS) [27]
U. Michigan	Multistage wave-pipelining theory [36]
U. Washington	Wave-pipelining theory [29]

In order to demonstrate the high-speed capabilities of the wave-pipelining techniques using a conservative CMOS process, most university prototypes share some common aspects in architecture, circuit and layout level. First, they are large circuits, where the effect of circuit design as well as the properties of the interconnect affect the circuit performance. Second, they can be implemented by regular structures. Therefore, wave-pipelining technique can be more easily applied. Third, a major portion of a prototype, such as the carry tree, usually has a circuit structure where each cell has regular fan-out. Therefore, the cell output capacitances are mostly dominated by the length of the interconnection wires and next input gates, whose lengths can be predicted due to the circuit regularity. Last, due to a lack of commercial tools that are directly applicable to designs using wave-pipelining, each group has more or less developed in-house design analysis and optimization tools which enable VLSI design using wave-pipelining.

Wong *et al.* at Stanford University, Stanford, CA, have designed a 63 bit population counter in ECL technology [42]. The counter achieves 2.5 waves and is the first bipolar LSI chip that uses wave-pipelining. Liu *et al.* at North Carolina State University have reported an architecture, circuit, layout, and testing techniques for achieving high speedup in a CMOS parallel adder using wave-pipelining [28]. The adder achieves nine waves at 250 MHz and is implemented by biased-CMOS gates with 2 μm CMOS technology. Klass *et al.* at Stanford University have designed and tested a 16 × 16 wave-pipelined multiplier and implemented it using static CMOS 1.0 μm technology. It achieves 3.7 waves at the clock rates between 330 and 350 MHz despite the substantial effects of data-dependent delay [24]. Circuit level tools were important for this design. Using 2 μm CMOS domino gates, Lien and Burleson at the University of Massachusetts, have designed a 4 × 4 Wallace tree multiplier that achieves two waves in the multiplier circuit [27]. Nowka and Flynn of Stanford University have designed a CMOS VLSI vector unit which includes a vector register file, an adder, and a multiplier [33]. The vector unit is implemented with 1 μm CMOS technology and simulated at 300 MHz. In this design, an adaptive supply method is used to counteract the effects of process variation.

B. Dynamic and Static RAM

As the speed of microprocessors increases, so does the performance gap between DRAM and the microprocessor. Consequently, high-speed S/DRAM technology is necessary to boost the overall system performance. Wave-pipelining offers an efficient way to reduce access and cycle time without incorporating additional registers, required by conventional pipelining, and without the associated clocking costs. Along with the reduced access time, wave-pipelining provides the additional advantages of latency and power dissipation. Several successful RAM designs using wave-pipelining have been reported [4], [25], [2], [31], [44], [15].

It is fair to say that the design of wave-pipelined S/DRAM has become a trend in the industry. Chappel *et al.* [4] designed a "bubble pipelined" RAM chip with 2 ns access time. Wave-pipelining is used in the cache RAM access in the HP Snake workstation [25]. NEC has designed a 220-MHz, 16-Mbit BiCMOS wave-pipelined SRAM [31]. Hitachi [15] has designed a 300-MHz, 4-Mbit wave-pipelined CMOS SRAM. Moreover Hitachi proposed the concept of a dual sensing latch circuit in order to achieve a shorter cycle time at 2.6 ns. Hyundai [44] has designed a 150-MHz, 8-bank, 256-Mbit synchronous DRAM. All of these designs can sustain three to four waves.

VI. CONCLUSIONS AND OPEN PROBLEMS

Despite recent advancement in wave-pipelining research and other aggressive timing approaches, a number of open problems still exist. Solutions to these problems may have broader application than just wave-pipelining.

1) *Testing:* Wave-pipelining presents additional challenges in delay testing due to the difficulty in observing internal points in a circuit, which looks like a combinational circuit but behaves like a sequential one. Testing for the interaction of long and short paths and the sequential false paths warrant further study.
2) *Low-Power:* Despite reduced clock loading, it appears that wave-pipelining is not a low-power technique, since at low-power supply levels delay variations tend to worsen. However, alternative low power methods, such as dynamic power supply adjustment and power-down techniques could favor wave-pipelining.

3) *High-Level Synthesis:* High-level synthesis tools have only recently been able to exploit internally pipelined modules. Wave-pipelined modules can present an additional degree of freedom to such tools, however the methods for modeling and verifying the timing properties unique to wave-pipelining for use at higher levels still need to be developed.
4) *Dynamic Power Supply and Clock Tuning:* One approach to process and environmental variation is the dynamic tuning of both power supplies and clock rates. This has been explored in [33], but is still far from being a widely used technique.
5) *Physical Design Issues:* Deep submicron technologies will continue to complicate abstractions of physical design. As interconnect delays begin to dominate gate delays, timing optimization will have to be done at the physical level.
6) *Parameter Variation:* Advanced technologies tend to prioritize density and speed at the cost of wider parameter variation. This will make clock distribution and design centering more of a challenge and could severely limit a designer's choice of timing schemes. One solution is the use of asynchronous circuit techniques. However, as the ever-increasing density and speed surpass the capabilities of designers to exploit their use, it may be that a truly "advanced" technology of the future will prioritize parameter variation, much like analog processes of today.

REFERENCES

[1] S. Anderson, J. Earle, R. Goldschmidt, and D. Powers, "The IBM system/360 model 91 floating point execution unit," *IBM J. Res. Develop.*, Jan. 1967.
[2] P. Bannon and A. Jan, "Today's microprocessor aim for flexibility," *EE Times*, pp. 68–70, 1994.
[3] W. Burleson, E. Tan, and C. Lee, "A 150 MHz wave-pipelined adaptive digital filter in 2μ CMOS," *VLSI Signal Processing Workshop*, 1994.
[4] T. I. Chappell, B. A. Chappell, S. E. Schuster, J. W. Allan, S. P. Klepner, R. V. Joshi, and R. L. Franch, "A 2-ns cycle, 3.8 ns access 512-kb CMOS ECL SRAM with a fully pipelined architecture," *IEEE J. Solid-State Circuits*, pp. 1577–1585, Nov. 1991.
[5] M. Cooperman and P. Andrade, "CMOS gigabit-per-second switching," *IEEE J. Solid-State Circuits*, June 1993.
[6] L. Cotten, "Maximum rate pipelined systems," in *Proc. AFIPS Spring Joint Comput. Conf.*, 1969.
[7] W. M. Jr. Coughran, E. Grosse, and D. J. Rose, "CAzM: A circuit analyzer with macromodeling," *IEEE Trans. Electron Devices*, vol. ED-30, pp. 1207–1213, Apr. 1983.
[8] B. Ekroot, "Optimization of pipelined processors by insertion of combinational logic delay," Ph.D. dissertation, Stanford Univ., Stanford, CA, 1987.
[9] J. Fishburn,"Clock skew optimization," *IEEE Trans. Comput.*, 1990.
[10] D. Ghosh and S. Nandy, "Design and realization of high-performance wave-pipelined 8×8 b multiplier in CMOS technology," *IEEE Trans. VLSI Syst.*, vol. 3, pp. 37–48, 1995.
[11] C. T. Gray, T. Hughes, S. Arora, W. Liu, and R. Cavin III, "Theoretical and practical issues in CMOS wave pipelining," in *Proc. VLSI'91*, 1991.
[12] C. T. Gray, W. Liu, and R. Cavin III, "Timing constraints fro wave-pipelined systems," *IEEE Trans. Computer-Aided Design*, vol. 13, pp. 987–1004, Aug. 1994.
[13] C. Gray, W. Liu, W. van Noije, T. Hughes, and R. Cavin, "A sampling technique and its CMOS implementation with 1 Gb/s bandwidth and 25 ps resolution," *IEEE J. Solid-State Circuits*, pp. 340–349, Mar. 1994.
[14] S. Gupta, private communication, Nov. 1994.
[15] K. Ishibashi *et al.*, "A 300 MHz 4-mb wave-pipelined CMOS SRAM using a multi-phase PLL," in *Proc. ISSCC'95*, 1995, pp. 308–309.
[16] D. A. Joy and M. J. Ciesielski, "Placement for clock period minimization with multiple wave propagation," in *Proc. 28th Design Automation Conf.*, 1991.
[17] D. A. Joy and M. J. Ciesielski, "Clock period minimization with wave pipelining," in *IEEE Trans. Computer-Aided Design*, Apr. 1993.
[18] S. T. Ju and C. W. Jen, "A high speed multiplier design using wave pipelining technique," in *Proc. IEEE APCCAS*, Australia, 1992, pp. 502–506.
[19] J. Kang, W. Liu, and R. Cavin, "A monolithic 625 mb/s data recovery circuit in 1.2 μm CMOS," in *Proc. Custom Integrated Circuit Conf.*, 1993, pp. 625–628.
[20] T. S. Kim, W. Burleson, and M. Ciesielski, "Logic restructuring for wave-pipelined circuits," in *Proc. Int. Workshop Logic Synthesis*, 1993.
[21] T.-S. Kim, W. Burleson, and M. Ciesielski, "Delay buffer insertion for wave-pipelined circuits," in *Notes of IFIP Int. Workshop Logic Architecture Synthesis*, Grenoble, France, Dec. 1993.
[22] F. Klass, M. J. Flynn, and A. J. van de Goor, *Pushing the Limits of CMOS Technology: A Wave-Pipelined Multiplier Hot Chips*, 1992.
[23] F. Klass and M. J. Flynn, "Comparative studies of pipelined circuits," Stanford University, Tech. Rep. CSL-TR-93-579, July 1993.
[24] F. Klass, M. J. Flynn, and A. J. van de Goor, "Fast multiplication in VLSI using wave-pipelining," *J. VLSI Signal Processing*, 1994.
[25] C. Kohlhardt, "PA-RISC processor for "Snake" work-stations," in *Hot Chips Symp.*, 1991, pp. 1.20–1.31.
[26] W. K. C. Lam, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Valid clocking in wavepipelined circuits," in *Proc. Int. Conf. Computer-Aided Design*, 1992.
[27] W.-H. Lein and W. Burleson, "Wave domino logic: Theory and applications," in *Proc. Int. Symp. Circuits Syst.*, 1992.
[28] W. Liu, C. Gray, D. Fan, T. Hughes, W. Farlow, and R. Cavin "A 250-MHz wave pipelined adder in 2-μm CMOS," *IEEE J. Solid-State Circuits*, pp. 1117–1128, Sept. 1994.
[29] B. Lockyear and C. Ebeling, "The practical application of retiming to the design of high-performance systems," in *Proc. ICCAD'93*, 1993, pp. 288–295.
[30] G. Moyer, M. Clements, W. Liu, T. Schaffer, and R. Cavin, "A technique for high-speed, fine-resolution pattern generation and its CMOS implementation," in *Proc. 16th Conf. Advanced Res. VLSI*, 1995, pp. 131–145.
[31] K. Nakamura *et al.*, "A 220-MHz pipelined 16-mb BiCMOS SRAM with PLL proportional self-timing generator," *IEEE J. Solid-State Circuits*, pp. 1317–1322, Nov. 1994.
[32] V. Nguyen, W. Liu, C. T. Gray, and R. K. Cavin, "A CMOS multiplier using wave pipelining," in *Proc. Custom Integrated Circuits Conf.*, San Diego, CA, May 1993, pp. 12.3.1–12.3.4.
[33] K. Nowka and M. Flynn, "System design using wave-pipelining: A CMOS VLSI vector unit," in *Proc. ISCAS'95*, 1995, pp. 2301–2304.
[34] J. Pratt and V. Heuring, "Delay synchronization in time-of-flight optical systems," *Appl. Opt.*, vol. 31, no. 14, pp. 2430–2437, 1992.
[35] L. Qi and X. Peisu, "The design and implementation of a very fast experimental pipelining computer," *J. Comput. Sci. Technol.*, vol. 2, no. 1, pp. 1–6, 1988.
[36] K. A. Sakallah, T. N. Mudge, T. M. Burks, and E. S. Davidson, "Synchronization of pipelines," in *IEEE Trans. Computer-Aided Design*, vol. 12, 1993.
[37] N. V. Shenoy, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Minimum padding to satisfy short path constraints," in *Proc. Int. Wokshop Logic Synthesis*, 1993.
[38] E. M. Sentovich, K. J. Singh, C. Moon, H. Savoy, and R. K. Brayton, "Sequential circuit design using synthesis and optimization," in *Proc. Int. Conf. Comput. Design*, 1992.
[39] H. Shin, J. Warnock, M. Immediato, K. Chin, C.-T. Chuang, M. Cribb, D. Heidel, Y.-C. Sun, N. Mazzeo, and S. Brodskyi, "A 5 Gb/s 16 $\times$ 16 Si-bipolar crosspoint switch," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 1992, pp. 228–229.
[40] S. Tachibana *et al.*, "A 2.6 ns wavepipelined CMOS SRAM with duealsensing-latch circuits," *IEEE J. Solid-State Circuits*, pp. 487–490, Apr. 1995.
[41] D. Wong, G. De Micheli, and M. Flynn, "Inserting active delay elements to achieve wave pipelining," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1989, pp. 270–273.
[42] D. Wong, G. De Micheli, M. Flynn, and R. Huston, "A bipolar population counter using wave pipelining to achieve 2.5$\times$ normal clock frequency," *IEEE J. Solid-State Circuits*, vol. 27, May 1992.
[43] D. Wong, G. De Micheli, and M. Flynn, "Designing high performance digital circuits using wave pipelining: Algorithms and practical experiences," *IEEE Trans. Computer-Aided Design*, vol. 12, Jan. 1993.
[44] H. Yoo *et al.*, "A 150 MHz 8-banks 256 m synchronous DRAM with wave pipelining methods," in *Proc. ISSCC'95*, 1995, pp. 250–251.

Wayne P. Burleson (S'87–M'94) received the B.S. and M.S. degrees in electrical engineering from the Massachusetts Institute of Technology (M.I.T.), Cambridge, in 1983. He received the Ph.D. degree in electrical engineering from the University of Colorado, Boulder, in 1989.

From 1983 to 1986, he worked at VLSI Technology, Inc., as a Custom Chip Designer of CMOS VLSI for DSP (fax modems). He is currently an Associate Professor of Electrical and Computer Engineering at the University of Massachusetts at Amherst. He directs research in VLSI signal processing, in particular, VLSI methods and CAD tools for low power and high speed. The research is being applied in chips and systems for real-time robotics, wireless LAN's, and remote sensing.

Dr. Burleson was a Chair of the 1998 VLSI Signal Processing Workshop. He has been the Guest Editor two special issues of IEEE journals, has been an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II, and is on the Editorial Board for the *Journal of VLSI Signal Processing*.

Maciej Ciesielski (M'85–SM'95) received the M.S. degree in electrical engineering from Warsaw Technical University, Poland, in 1974, and the Ph.D. degree in electrical engineering from the University of Rochester, Rochester, NY, in 1983.

From 1983 to 1986, he was a Senior Member of Technical Staff at GTE Laboratories, MA, involved in a silicon compilation project. Currently, he is Associate Professor in the Department of Electrical and Computer Engineering at the University of Massachusetts, Amherst. His research interests include CAD for VLSI systems, high-level and logic synthesis, layout synthesis and physical design automation, performance optimization of integrated circuits, and mathematical optimization methods.

Fabian Klass (A'95) received the B.S.E.E. degree from Universidad Nacional de Tucuman, Argentina, in 1985, the M.S.E.E. degree from Technion—Israel Institute of Technology, Haifa, Israel, in 1989, and the Ph.D. degree in electrical engineering from Delft University, The Netherlands, in 1994.

From 1992 to 1994, he was a Visiting Scholar at Stanford University, Stanford, CA, where he joined the Computer Systems Laboratory, working in the area of high-speed CMOS digital circuits (wave-pipelining). In 1994, he joined Sun Microsystems Inc., Sunnyvale, CA, where he is currently a Circuit Design Leader for the UltraSparc III microprocessor. His current areas of interest include high-speed CMOS design, clocking, and computer organization. He has published ten technical papers on high-speed circuit techniques, holds one patent, and has ten pending.

Wentai Liu (S'78–M'81–SM'93) received the B.S.E.E. degree from National Chiao-Tung University, Taiwan, the M.S.E.E. degree from National Taiwan University, Taipei, and the Ph.D. degree from the University of Michigan, Ann Arbor.

Since 1983, he has been on the Faculty of North Carolina State University, Raleigh, where he is currently a Professor of electrical and computer engineering. Currently, he leads the widely reported retinal prosthesis project. His research interests include visual prosthesis, VLSI design/CAD, and sensor design. He holds three U.S. patents and has coauthored two books.

Dr. Liu received an IEEE Outstanding Paper Award in 1986. He has served as an IEEE-CAS representative and is currently an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II.