Clocking and clocked storage elements in a multi-gigahertz environment

V. G. Oklobdzija

Clocking considerations and the design of clocked storage elements are discussed in this paper. We present a systematic approach for deriving a clocked storage element suitable for "time borrowing" and absorption of clock uncertainties. We explain how to compare different clocked storage elements with each other, and discuss issues related to power consumption and low-power designs. Finally, results of comparisons among representative designs are presented.

Introduction

Deciding on the clocking strategy is one of the single most important decisions when designing a digital system. If the wrong strategy is employed, system bring-up and diagnostics can be very costly, and system operation will remain unreliable throughout its lifetime. The importance of clocking is gaining recognition as clock speeds rapidly increase, traditionally doubling every three years—and lately, every two years.

As clock speeds increase, the number of logic levels in the *critical path* diminishes. In today's high-speed processors, instructions are executed in one cycle, driven by a single-phase clock. In addition, the number of pipeline stages has increased to 15 or 20 to accommodate the increase in clock speed. Today, ten levels of logic in the critical path are common, and, as shown in **Figure 1** [1], this number is expected to decrease further. The diminishing amount of logic placed between two pipeline stages is responsible in large part for the recent and rapid increase in clock frequency, an increase that has surpassed the traditional trend in technology scaling. This decrease in the amount of logic between two pipeline stages is occurring at about half the rate at which clock frequency is increasing, bringing the number of pipeline stages to roughly one half every six years. However, this trend cannot be expected to continue much longer because a minimal amount of logic (at least two stages) is necessary to make the pipeline stage meaningful. With deeper pipelines, any overhead associated with the clock system and clocking mechanism that directly and adversely affects machine performance is critically important.

At today's frequencies, the ability to absorb clock skew and use faster clocked storage elements (CSEs) results in a direct performance improvement comparable to those obtained through difficult implementations of architectural or microarchitectural techniques.

As the clock frequency reaches 5–10 GHz, traditional clocking techniques will be stretched to their limits, because three to five gates per stage would be barely useful. Beyond that frequency, traditional CSEs would be using as much logic as the pipeline stage. With power continuing to grow, requirements for low power would necessitate more efficient clocking solutions. Thus, new ideas and new ways of designing digital systems are required.

Clocking considerations in sequential systems

Clock distribution

The two most important timing parameters that affect the clock signal are clock skew and clock jitter.

Clock skew is a spatial variation of the clock signal as distributed through the system. It is caused by the various resistive/capacitive (RC) characteristics of the clock paths to the various points in the system and the different loading of the clock signal at different points on the chip. Further, we can distinguish *global clock skew* and *local*

0018-8646/03/$5.00 © 2003 IBM

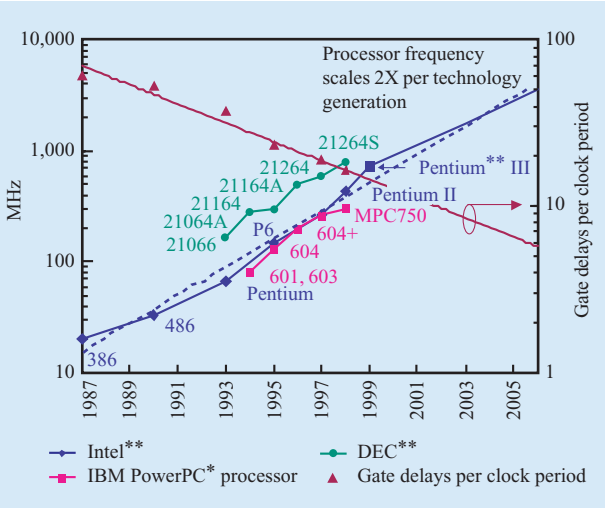

Figure 1

Increase in the clock frequency and decrease in the number of logic levels in the pipeline. Reprinted from [1] with permission; © 1999 IEEE.

clock skew, which are equally important in the design of high-performance systems.

Clock jitter is a temporal variation of the clock signal with regard to the reference transition (reference edge) of the clock signal. Clock jitter represents edge-to-edge variation of the clock signal in time. As such, clock jitter can also be classified as either of two types: *long-term jitter* or *edge-to-edge clock jitter*, which is defined as clock-signal variation between two consecutive clock edges. In high-speed logic design, we are more concerned about the edge-to-edge clock jitter, because it affects the time available for the logic operation.

Typically, the clock signal has to be distributed to several hundreds of thousands of the CSEs. Therefore, the clock signal has the largest fanout of any node in the design and requires several levels of amplification. As a consequence, the clock system by itself can consume up to 40–50% of the power of the entire VLSI chip [2]. We must also ensure that every CSE receives the clock signal precisely at the same moment in time.

There are several methods of distributing the on-chip clock signal while minimizing clock skew and limiting the power dissipated by the clock system [3, 4]. Two typical cases are an RC-matched tree and a grid [5].

If given superior computer-aided design tools, a perfect and uniform process, and the ability to route wires and balance loads with a high degree of flexibility, an RC-matched delay clock distribution tree would be preferable to a grid. However, we do not have a perfect and uniform process and a high degree of flexibility in routing and balancing loads. As a result, a grid is used when clock distribution on the chip has to be controlled very precisely, as is the case with high-performance systems. However, because the clock consumes more power when using a grid arrangement, and because local variations in device geometry and supply voltage are important components of the clock skew, it is necessary to use more sophisticated clock distribution than simple RC-matched or grid-based schemes. Active schemes with adaptive digital de-skewing typically reduce the clock skew of simple passive clock networks by an order of magnitude, allowing tighter control of the clock period and higher clock rates [6].

Synchronous systems

A traditional view of the finite state machine is represented by the Huffman model, which consists of a combinational logic element and CSE. In this model, the next state, which is determined by the present state and the input, is stored into the CSE by the triggering mechanism of the clock (edge or level). Following this model, we are used to thinking that the purpose of the CSE is to "hold" or "memorize" the state. This view is further supported by the level-sensitive scan design (LSSD) methodology, which uses storage elements to "scan out" the state of the machine during the test and debug mode.

In this paper, however, we offer a different view. In order to ensure proper operation, the purpose of the CSE is to prevent the corruption of the next state [**Figure 2(a)**]. Though memorizing the state is a needed function for the architected registers, it is not a necessary function for every CSE in the machine.

This view is broader and can represent wave pipelining [7], for example. In the case of wave pipelining, the signal is blocked from corrupting the present state, S_n, by the sheer delay of the wire. The signal simply cannot arrive in time; therefore, no blocking is necessary. However, this model also reveals the problems of wave pipelining. Ideally, all signals should arrive at the same moment in time, which is not possible. Therefore, the fast-path problem becomes more difficult to control, and it is necessary to impose much more stringent requirements on the fast-paths. Because this too is not possible, the system runs the danger of corrupting the state in the course of several cycles. The case of skew-tolerant domino logic [8, 9], shown in **Figure 2(b)**, conforms to the model presented in Figure 2(a).

Blocking of the signal is accomplished by the pre-charge phase of the clock. For example, while Clock Φ_2 is *low* (pre-charge), data from Stage 1 cannot be passed to Stage 2. Only after the pre-charge phase has elapsed and Clock Φ_2 has returned to its *high* value can data from Stage 1 be passed to Stage 2. This transfer has to be completed while

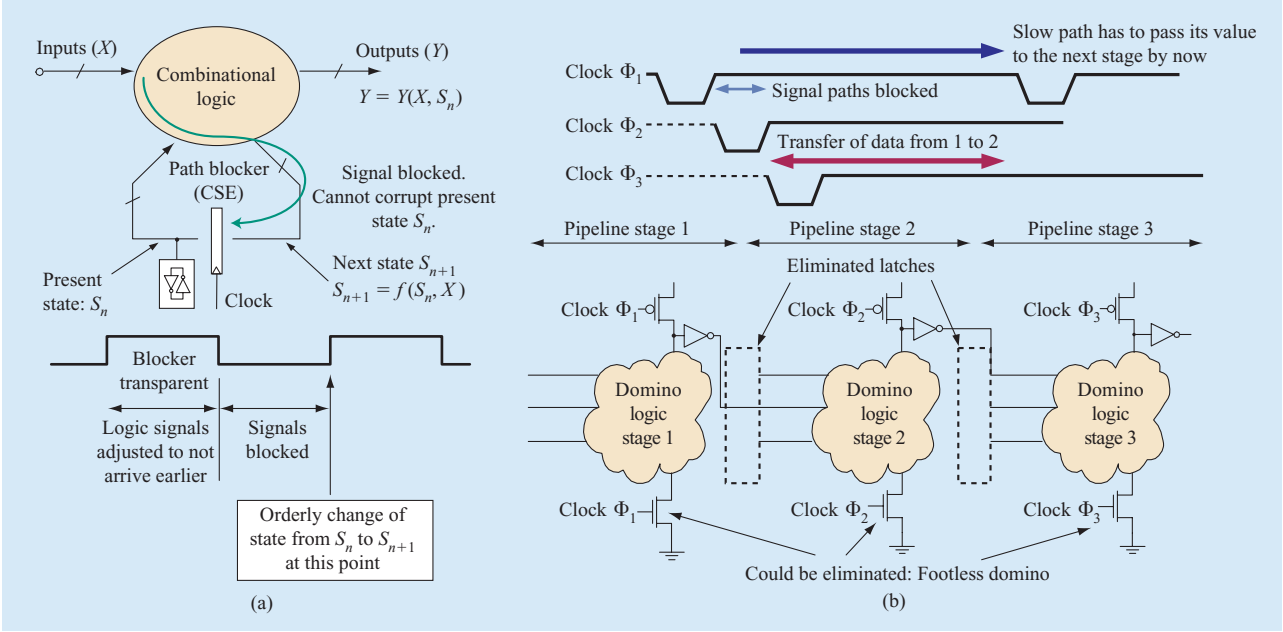

Figure 2

(a) A different view of the finite state machine (FSM). (b) Skew-tolerant domino logic: no explicit latches.

Clock Φ_1 is *high*. Obviously, the speed of this logic is determined by the precise matching of the clocks. This is accomplished by having the clock signal travel along the datapath, while the local clocks are generated by delaying the clock for the amount of time needed in the logic stage. In a way, this is similar to the clocking used in early mainframe computers [10]. In general, the overlap between negative phases of the Clock Φ_i–Φ_{i+1} is not necessary; however, this overlap is needed if we are to eliminate the bottom ("footer") transistors, as shown in Figure 2(b). The need for an overlap results from the requirement that all paths-to-ground be deactivated before the gate begins pre-charge. When footer transistors are removed, it is necessary to ensure that at least one transistor in the series stack be off during pre-charge to avoid contention [9].

Asynchronous systems

With the increase in clock frequency, synchronous systems are facing serious problems—the inability to precisely control the clock, nonscaling clock uncertainties, wire delays, and the simple fact that the signal may require one or more clock cycles to reach its destination. Thus, asynchronous system design has been revisited.

The overhead imposed on the synchronous system by clock uncertainties and CSE properties is simply traded for the overhead imposed by the handshake signaling in the asynchronous system (**Figure 3**). Thus, the question really is this: *Which system—synchronous or asynchronous—can be designed so that it imposes lesser penalties on the data transfer as the speed of the logic continues to rapidly increase?* Today, it makes logical sense to use synchronous design in local domains, which can be clocked synchronously without considerable difficulties. Data transfers lasting several clock cycles could be accomplished using asynchronous communication. This opinion is supported by the fact that at 10 GHz or more, it would take several clock cycles to cross from one chip edge to another, as well as the fact that an entire processor in a one-billion-transistor chip would occupy only a small portion of the chip.

Clocked storage elements

The function of a CSE, flip-flop, or latch is to block the signal path, thus preventing it from corrupting the present state. In addition, it may be used to capture the state information and preserve it as long as it is needed by the digital system. It is not possible to define a storage element without defining its relationship to the clock.

Master–slave latch

To avoid the transparency of a single latch, two latches are clocked back-to-back with two non-overlapping phases of the clock. In such an arrangement, the first latch serves as the *master* by receiving the values from the data input

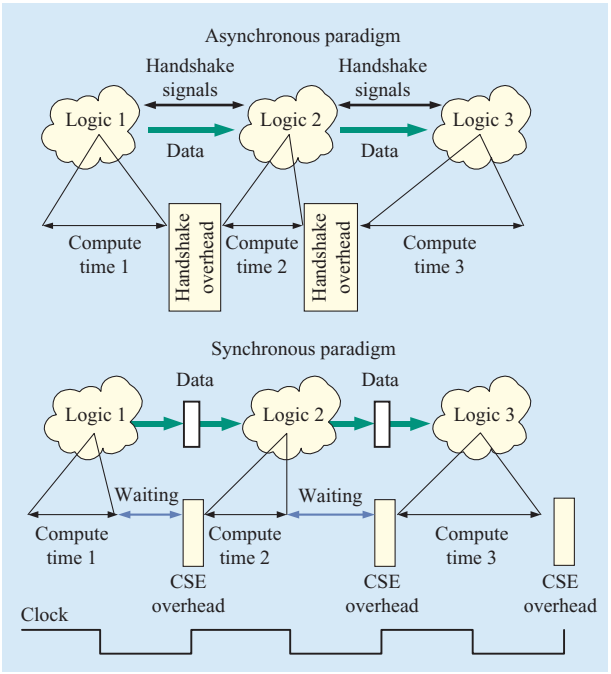

Figure 3

Comparison of data transfer in an asynchronous system and a synchronous system.

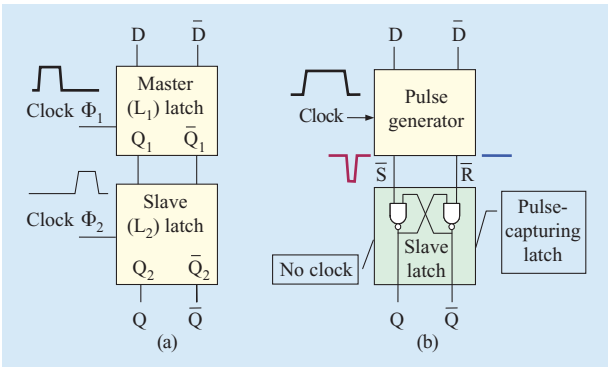

Figure 4

General structure of (a) master–slave (M–S) latch and (b) a flip-flop.

and passing them to the *slave* latch, which simply follows the master. This is known as a master–slave (M–S) or L1–L2 latch, as shown in **Figure 4(a)**. This configuration should not be confused with the flip-flop [**Figure 4(b)**]. In this paper, we stress the fundamental differences between the flip-flop and M–S latches.

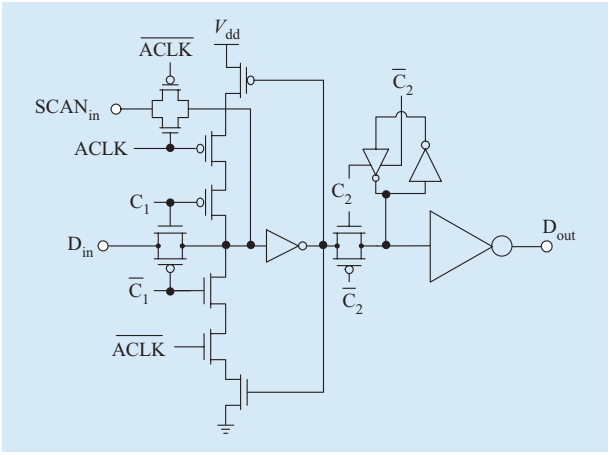

Figure 5

Master–slave latch with LSSD capability as used in the IBM PowerPC 603 processor. Reprinted from [12] with permission; © 1994 IEEE.

In an M–S latch, the slave latch can have two or more masters acting as an internal multiplexer with storage capabilities. The first master is used to capture the data input, while the second master can be used as scan-input for testing and is generally clocked with a separate clock, as it is done in IBM LSSD [11]. M–S latch design provides robustness and low-power characteristics when data activity is low. One example of an LSSD-compatible M–S latch, as used in the IBM PowerPC 603* processor, is shown in **Figure 5** [12].

Flip-flop

Flip-flops and latches operate on different principles. While a latch is *level-sensitive*, meaning that it is acting on the level (logical value) of the clock signal, a flip-flop is *edge-sensitive*, which means that the mechanism of capturing the data value on its input is related to the changes of the clock. The two are designed for a different set of requirements and thus consist of inherently different circuit topology.

The general structure of the flip-flop is shown in Figure 4(b). Notice the difference between the flip-flop and the M–S latch. A flip-flop consists of two stages: a pulse generator (PG) and a pulse-capturing latch (PCL). The PG generates a negative pulse on either $\bar{S}$ or $\bar{R}$ lines, which are normally held at logic 1 level. This pulse is a function of data (D) and clock signals and should be of sufficient duration to be captured in the PCL. The duration of that pulse can be as long as half of the clock period, or it can be as short as one inverter delay. On the contrary, the M–S latch generally consists of two identical clocked latches,

and its nontransparency feature is achieved by non-overlapping clocks Φ_1 and Φ_2, clocking master latch L_1, and slave latch L_2.

The flip-flop PG block shown in Figure 4(b) should provide the pulse on S_{n+1} (the value of the signal S after the rising edge of the clock) such that: *The next state of a flip-flop will be set to 1 only at the time the clock becomes 1 (rising edge of the clock), the data at the input is 1, and the flip-flop is in the "steady state" (both S_n and R_n are 0). The moment the flip-flop is set ($S_{n+1} = 1$, $R_{n+1} = 0$), no further change in data input can affect the flip-flop state; data input will be "locked" by ($D + S_{n+1}$) = 1, and reset R_{n+1} would be disabled (by $S_{n+1} = 1$).*

This ensures the *edge sensitivity*—i.e., after the transition of the clock and setting of the S or R signal to its desired state, the flip-flop is *locked* for receiving new data. It is possible to derive these equations from the functional specifications on a Karnaugh map, as shown in **Figure 6(a)**.

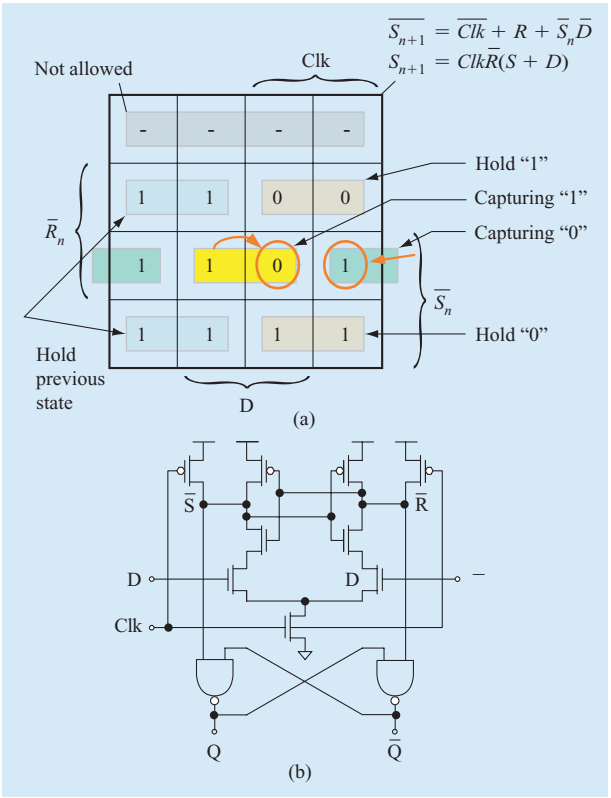

Figure 6

(a) Karnaugh map showing derivation of the pulse-generating stage of a flip-flop (only the S_n signal is shown). (b) A sense amplifier flip-flop (SAFF) used in the DEC 21264 Alpha processor. Reprinted from [2] with permission; © 1998 IEEE.

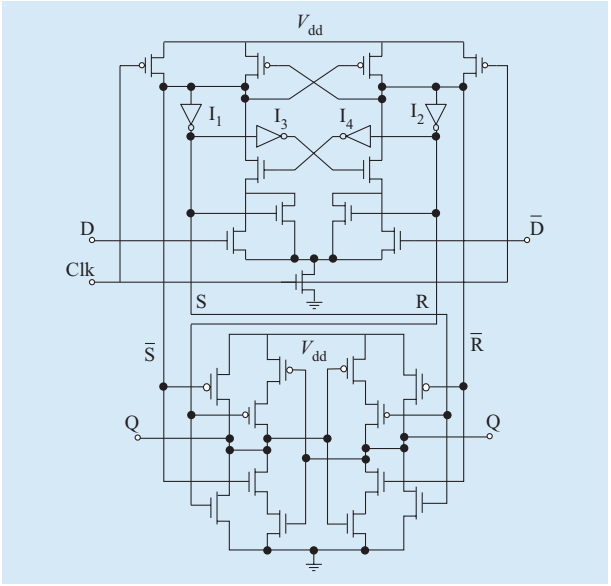

Figure 7

Improvement by proper design of the pulse generator stage of the sense amplifier flip-flop. First stage [12]; second stage, [13].

In a flip-flop, the relationship of the S and R signals with respect to the data (D) and clock (Clk) signals is expressed as

$$S_{n+1} = Clk\overline{R_n}(D + S_n) \tag{1a}$$

and

$$R_{n+1} = Clk\overline{S_n}(\bar{D} + R_n). \tag{1b}$$

The subscript $n+1$ refers to the time after the rising edge of the clock, while the subscript n refers to the time before it. Equations (1a) and (1b) form a basis for a derivation of the flip-flop PG stage shown in **Figure 6(b)** [2].

However, it took engineers several attempts to arrive at the right circuit topology of the particular flip-flop shown in Figure 6(b). This flip-flop—as used in the third generation of the DEC 600-MHz Alpha [2] and ARM [13] processors—is a version of the flip-flop introduced by Madden and Bowhill and based on the static memory cell design [14]. Thus, it is also known as a sense amplifier flip-flop (SAFF). Further development of the PG block of this flip-flop is illustrated in **Figure 7** [15, 16]. A doubling in speed while maintaining the same energy of operation is achieved by the modification of the second stage by Stojanovic and Oklobdzija [16], shown in Figure 7. Contrary to the impression given by the increase in the number of transistors in Figure 7 over the SAFF shown in Figure 6(b), the area increase is actually relatively small—roughly 7%, and the layout size is comparable to those of

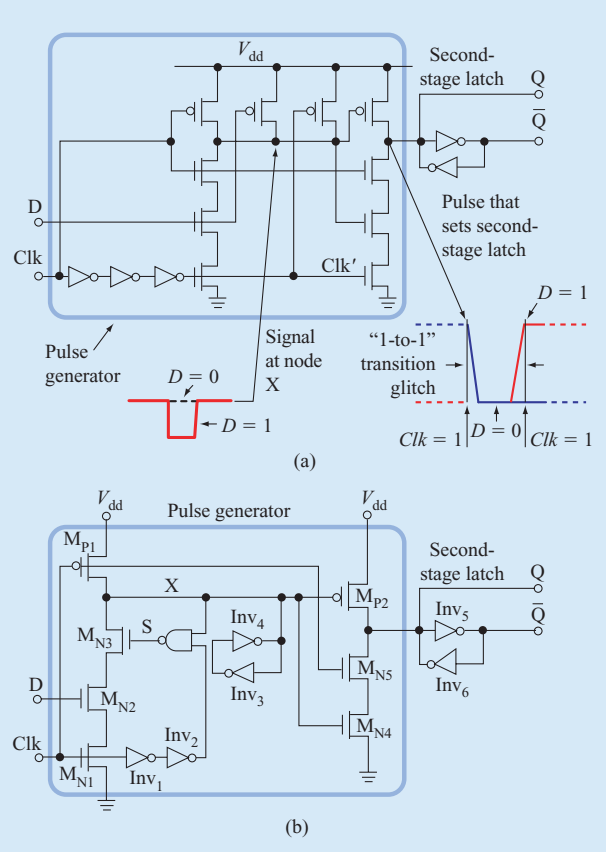

Figure 8

(a) A hybrid-latch flip-flop (HLFF) introduced by Partovi. Reprinted from [19] with permission; © 1996 IEEE. (b) Semi-dynamic flip-flop (SDFF). Reprinted from [20] with permission; © 1998 IEEE.

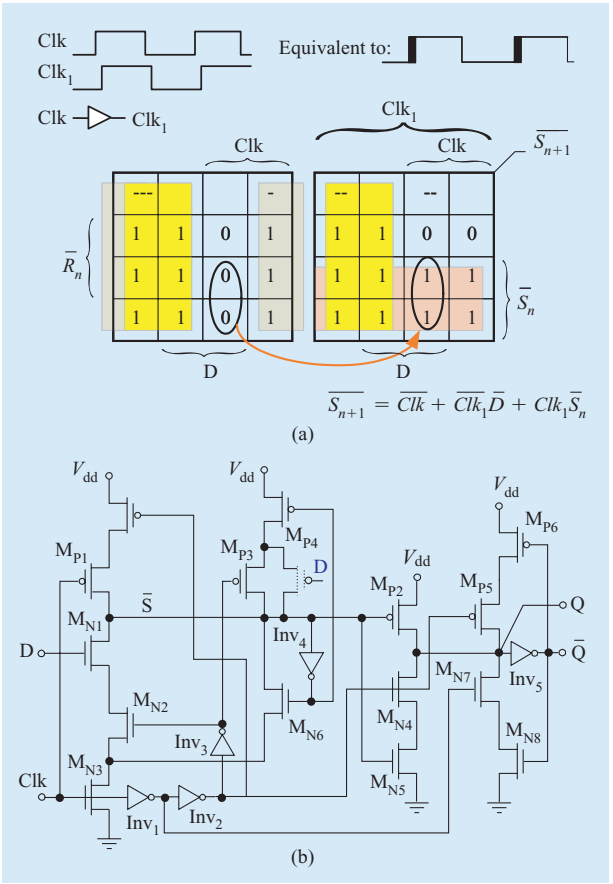

Figure 9

(a) Systematic derivation of time-window-based flip-flop. (b) Systematically derived single-ended flip-flop. Reprinted from [21] with permission; © 2000 IEEE.

other representative latches and flip-flops [17]. Substantial power reduction of the second stage and elimination of the floating nodes while using minimal-size transistors were key to achieving higher performance at the same energy level, as reported in [15, 17, 18].

Time-window-based flip-flops

Digital circuits are based on discrete time events. The time reference is a clock signal and/or finite delay through one or more logic elements. To generate a needed time reference, a pulse created by the property of reconvergent fan-outs is commonly used. This method is illustrated in **Figure 8(a)** on a hybrid-latch flip-flop (HLFF) introduced by Partovi et al. [19]. The trailing edge of this pulse is used as a time reference for shutting the flip-flop off. Thus, a short *time window* is created during which the flip-flop is accepting data (which is how "edge" is created in the digital world). However, analysis of an HLFF shows that the design is incomplete, resulting in an output glitch during the *1-to-1 transition*, as shown in Figure 8(a).

A flip-flop based on a similar principle—the semi-dynamic flip-flop (SDFF)—was introduced by Klass [20] [**Figure 8(b)**]. It uses a NAND gate to inhibit any further changes and allows a later arrival of D, thus widening the transparency window and decreasing the setup time of the flip-flop. The SDFF is characterized by one of the highest performances, but it suffers the same output glitch problem as the HLFF. The problem is in the floating output node, which is susceptible to glitches and even the slightest mismatch of clock signals.

A systematic approach to derive a time-window-based single-ended flip-flop that eliminates the output glitch problem is shown in **Figure 9(a)**. The time window is created by using two reference points: clock signals Clk and Clk_1, where Clk_1 is the time reference created when the Clk signal is delayed by passing it through a buffer.

The equation for the S_{n+1} signal is given in Figure 9(a). This flip-flop does not suffer from reliability problems.

In **Figure 9(b)**, we show the structure of a flip-flop [21] similarly derived in the described fashion. This flip-flop has three time reference points: the clock signal, Clk, the clock signal passed through three inverters, Clk_3, and the clock passed through two inverters, Clk_2. The equation describing the pulse generator stage of this flip-flop is given by

$$\bar{S}_{n+1} = X = \overline{(Clk + Clk_2)(D\, Clk_3 + S_n)}\,. \tag{2}$$

The n-MOS transistor section is a full realization of this equation. For performance reasons, the p-MOS section is somewhat abbreviated to

$$\bar{S}_{n+1} = \overline{(Clk + Clk_2)(Clk_3 + S_n)}\,. \tag{3}$$

The second stage (capturing latch) is implemented as

$$Q_{n+1} = \overline{\bar{S}_n(Clk_2 + \bar{Q}_n)}\,. \tag{4}$$

This systematically derived flip-flop [21] does not have hazards in the output stage and is outperforming HLFF [19] and SDFF flip-flops [20].

Pulsed latches

To decrease the time overhead imposed by the CSE, designers may resort to using a single latch. To narrow the transparency window of the latch, the latch is clocked with short pulses generated locally from the global clock signal. Thus, the possibility of hold time violation and *races* (short paths) is not entirely eliminated, but it is traded for the convenience of a single latch and lower pipeline overhead. Given that the clock pulse is short, the hazard could be reduced by *padding* the logic—adding inverters in the fast-paths to eliminate the problem. It is reasonable to expect that when comparing the power of the pulsed latches, the portion of the power that goes to padding should be accounted for.

The clock produced by the local clock generator must be wide enough to enable the latch to capture its data. At the same time, it must be short enough to minimize the possibility of critical race. By reducing the robustness and reliability of such a design, these conflicting requirements make it hazardous to use such single-latch designs. Nevertheless, such a design has been used in response to the critical need to reduce the cycle overhead imposed by the CSEs. An Intel** version of a pulsed latch is shown in **Figure 10** [22]. An additional benefit of this design is its low power consumption as a result of the common clock signal generator and the simple structure of the latch. This power can be traded for speed. The pulse generator used in the Intel pulsed latch uses the principle of reconvergent fanout with nonequal parity of inversion to obtain the desired short clock pulse. However, further

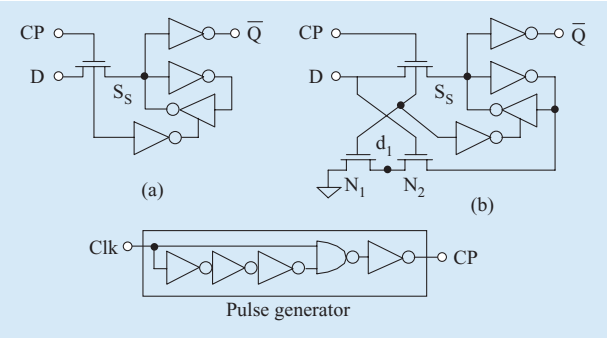

Figure 10

The Intel explicit pulsed latch as an example of a pulse latch. Reprinted from [22] with permission; © 2001 IEEE.

analysis shows that as the technology is scaled down and less and less logic is placed into the pipeline stage, the timing constraints imposed by the pulsed latch may be more difficult to meet than it seems.

Analysis of the pulsed-latch timing conditions

The conditions for reliable operation of a system using a single latch are described in a paper by Unger and Tan [23] and given by

$$P_m = P \geq D_{LM} + D_{CQM} + T_L + T_T + U - W, \tag{5}$$

$$P \geq D_{LM} + D_{DQM}\,, \tag{6}$$

and

$$D_{Lm} > D_{LmB} \geq W + T_T + T_L + H - D_{CQm}\,, \tag{7}$$

where

P_m is the minimum period at which the system can operate—the inverse of the maximum achievable frequency of operation,
U and H are setup and hold times,
W is the duration of the active portion of the clock signal,
T_L and T_T are the clock uncertainties of the leading and trailing edge of the clocks,
D_{CQM} is the longest clock-to-output Q delay of the latch,
D_{LM} is the longest path in the logic,
D_{Lm} is the minimum delay of the logic required to avoid races, and
D_{CQm} is the smallest clock-to-output Q delay of the latch.

From Equation (5), it can be seen that the increase of the clock width W is beneficial for speed, but it also increases the minimal bound for the fast-paths [Equation (7)]. The maximum useful value for W is obtained when the period, P, is minimal [Equation (6)]. Substituting P

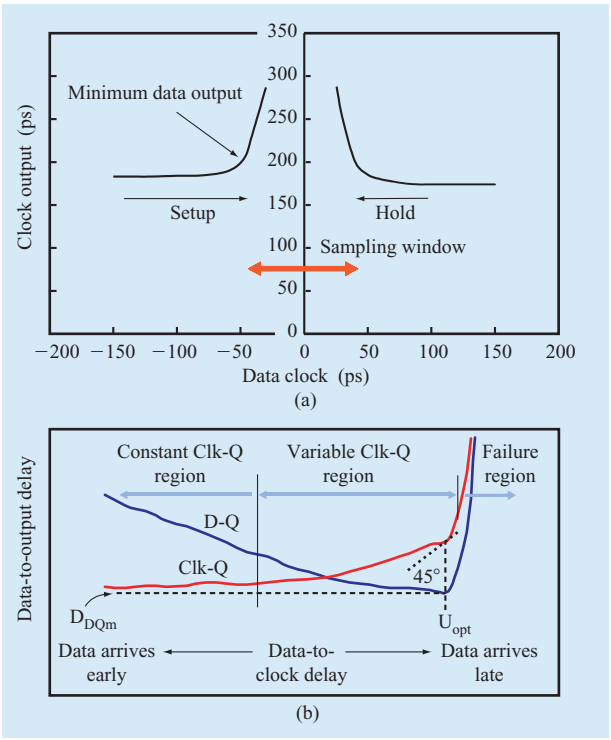

Figure 11

(a) Setup and hold time behavior as a function of clock-to-output delay. (b) Setup and hold time behavior as a function of data-to-output delay.

from Equation (6) into Equation (5) yields the optimal value of W:

$$W_{opt} = T_L + T_T + U + D_{CQM} - D_{DQM}\,. \tag{8}$$

If we substitute the value of the optimal clock width, W_{opt}, into Equation (5), we obtain the values for both the maximum speed [Equation (6)] and the minimal signal delay in the logic that has to be maintained to satisfy the conditions for optimal single-latch system clocking:

$$D_{LmB} = 2(T_T + T_L) + H + U + D_{CQM} - D_{CQm} - D_{DQM}\,. \tag{9}$$

Equation (6) tells us that in a single-latch system, it is possible to make the clock period, P_m, as small as the sum of the delays—latch delay and the critical path delay in the logic block—in the signal path. This can be achieved by adjusting the clock width, W, and ensuring that all of the fast-paths in the logic are longer in their duration than some minimal time, D_{LmB}. In the pulse latch, data arrives during the transparency period of a latch and very close to the optimal setup time, U, which is, in fact, negative with respect to the clock rising edge. Thus, we can write $U + D_{CQM} = D_{DQM}$, and approximating $T_L = T_T \approx T_{unc}$, we can simplify Equations (8) and (9) to

$$W_{opt} \approx 2T_{unc} \text{ and } D_{LmB} \approx 4T_{unc} + H - D_{CQm}\,. \tag{10}$$

Equation (9) tells us that, under ideal conditions, if there are no clock skews and no process variations, the fastest path through the logic has to be greater than the sampling window of the latch ($H + U$) minus the time the signal spends traveling through the latch. If the travel time through the latch, D_{DQM}, is equal to or greater than the sampling window, we do not have to worry about fast-paths. This also assumes that we can produce a very short pulse.

However, in practice, this is not true. The optimal clock width, W_{opt}, that can be produced depends on the generation method and, compared with the clock period, P, it is not short. We may illustrate this with some typical delay numbers for 100-nm CMOS technology:

- FO4 delay = ~25–40 pS.
- Latch delay = ~80 pS.
- T_{unc} = ~25–35 pS.
- P_m = 250–400 pS (f_{max} = ~2.5–4 GHz).
- W_{opt} = ~$2T_{unc}$ = ~50–70 pS.
- D_{Lm} = ~$4T_{unc} + H - D_{CQm}$ = ~100–60 pS (close to one third to one half of a cycle).

The optimal clock pulse, W_{opt}, is close to what can be expected from Figure 10. However, the fast-paths must be longer than one third to one half of the period, P, which represents a significant constraint.

Timing parameters

The data and clock inputs of a CSE must satisfy basic timing restrictions to ensure correct operation of the flip-flop. Fundamental timing constraints between data and clock inputs are quantified with setup and hold times, as illustrated in **Figure 11(a)**. Setup and hold times define time intervals during which input has to be stable to ensure correct flip-flop operation. The sum of the setup and hold times defines the sampling window of the CSE.

Setup and hold-time properties

Failure of the CSE due to setup and hold-time violations is not an abrupt process. This failing behavior is shown in **Figure 11(b)**. Considering how close to the locking event data should be allowed to change, we encounter two opposing requirements:

- It should be kept farther from the failing region for the purpose of design reliability.
- It should be as close to the clock as possible to increase the time available for the logic operation.

This is an obvious dilemma. Some vendors specify setup and hold times as points in time when the Clk-Q (t_{CQ}) delay raises for an arbitrary number of 5–20%. We do not find this reasoning to be valid.

A redrawn picture [Figure 11(b)], in which the D-Q (t_{DQ}) delay is plotted (instead of Clk-Q), provides more insight. From this graph we see that in spite of the increase in the Clk-Q delay, we are still benefiting because the D-Q delay (representing the time taken from the cycle) is reduced.

Time borrowing and absorption of clock uncertainties

Even if data arrives close to the reference edge of the clock or passes the clock edge, the delay contribution of the storage element is still smaller than the amount of delay passed on to the next cycle, allowing more useful time for logic operation. This is known as *time borrowing*, *cycle stealing*, or *slack passing*. To understand the full effect of delayed data arrival, we have to consider a pipelined design in which the data captured in the first clock cycle is used as input in the next clock cycle (**Figure 12**).

The sampling window is the time period in which the CSE is sampling, and thus data is not allowed to change. The length of time for which T_{CR1} was stretched does not come without cost. It is simply taken away (*stolen* or *borrowed*), leaving less time in the next cycle (Cycle 2) for T_{CR2}. Thus, a boundary between pipeline stages is somewhat flexible. If we move around the clock reference edge, giving it some flexibility, this feature helps in absorbing the clock skew and jitter uncertainties. Thus, time borrowing is one of the most important characteristics of today's high-speed digital systems. Absorption of clock jitter is shown in **Figure 13(a)** [19], and the effect on data arrival in the following cycle is illustrated in **Figure 13(b)**. We observe how moderate amounts of clock uncertainties can be effectively absorbed, while the absorption property diminishes as clock uncertainties become excessive.

The benefits of the flat data-to-output characteristic are obvious, and we create them by expanding the time window during which the storage element is transparent (*transparency window*). Widening of the transparency window is equivalent to increasing the separation in time between the two reference events—one that opens the

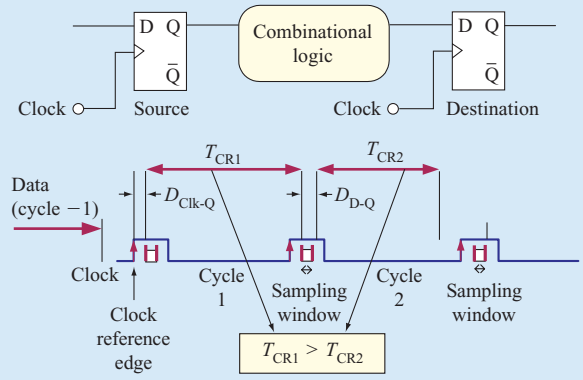

Figure 12

Time borrowing in a pipelined design. The setup time, U, is negative with respect to the rising edge of the clock.

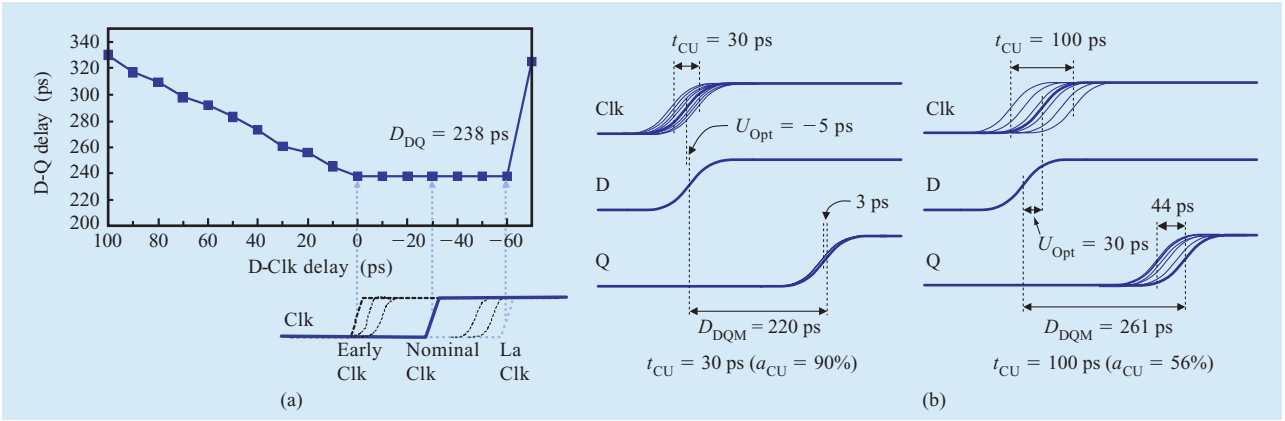

Figure 13

(a) Clock skew absorption property: data-to-output delay as a function of clock arrival time. (b) Effects of clock uncertainties on data arrival in the next cycle.

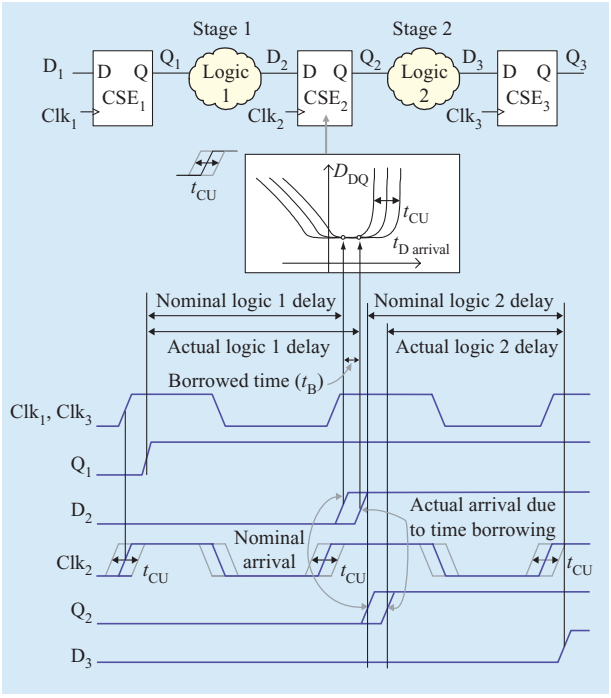

Figure 14

Time borrowing with uncertainty-absorbing clocked storage elements.

CSE and the other that closes it. In effect, the storage element behaves as a transparent latch for the short time after the active clock edge. The wider the transparency window, the wider the flat region of the data-to-output characteristic. Widening the transparency window can be done by intentionally creating wider capturing pulses of flip-flops and pulsed latches or by overlapping the master and slave clocks of M–S latches.

A consequence of increasing the transparency window is that the failure region of the data-to-output characteristic is moved away from the nominal clock edge. This results in the decrease of the setup time (larger negative values) and the increase of the hold time of the storage element. While decreasing setup time has no significant effect on system timing as long as the data-to-output delay is constant, a large hold time makes the fast-path requirement harder to meet. Thus, the design for the absorption of clock uncertainty is often traded for longer hold time. In many cases, however, these two requirements are not contradictory, because different types of storage elements are used in fast and slow paths. The maximal clock skew that a system can tolerate is determined by the CSEs. If the clock-to-output delay of a CSE is shorter than the hold time required, and there is no logic between two storage elements, a race condition can occur. A *minimum delay restriction* on the clock-to-output delay is given by

$$t_{\mathrm{CLK-Q}} \geq t_{\mathrm{hold}} + t_{\mathrm{skew}}. \tag{11}$$

If this relation is satisfied, the system is immune to hold-time violations. Otherwise, it is necessary to check that all of the timing paths have some minimal delay, which ensures that there is no hold-time violation.

The clock uncertainty absorption property shows how the propagation delay of a CSE changes if the arrival of the reference clock is uncertain. Applying clock uncertainty to a CSE is equivalent to keeping the reference-clock arrival fixed while allowing the data arrival to change.

More generally, uncertainty absorption could be treated as *data-to-output* delay degradation resulting from the changes in *data-to-clock* delay. As such, it can be used for time borrowing in exactly the same way it is used for clock uncertainty absorption. Therefore, a soft clock edge designates a property of a storage element whose output follows both early and late arrivals of the input, thus allowing slower stages to borrow time from subsequent faster stages.

The time-borrowing capability and clock uncertainty absorption are not mutually exclusive; they can be traded off for one another. **Figure 14** illustrates a case in which a wide transparency window, denoted as a flat data-to-output characteristic, is used both to absorb the clock uncertainties, t_{CU}, and to borrow time, t_{B}, from the surrounding stages. The combinational logic of Stage 1 takes more time than nominally assigned, and it borrows a portion of the cycle time from Stage 2. In general, the storage element may not be completely transparent (i.e., the data-to-output characteristic is not completely flat). The combination of clock uncertainty, t_{CU}, and time borrowing, t_{B}, causes an increase in the data-to-output delay of the flip-flop, ΔD_{DQ}.

The delay increase, ΔD_{DQ}, is the same both in the case when the clock uncertainty is $t_{\mathrm{B}} + t_{\mathrm{CU}}$ with no time borrowing and in the case when the borrowed time between stages is $t_{\mathrm{B}} + t_{\mathrm{CU}}$ and there is no clock uncertainty. It should be noted that the practical values of the total borrowed time are about the width of the transparency window of the storage element and, in any event, shorter than the hold time. Better absorption and time-borrowing capability can be obtained by widening the transparency window. However, if the transparency window is widened, the hold time increases, and the short-path requirements become harder to meet. Therefore, use of a wide transparency window is a tradeoff between the time borrowing and uncertainty absorption on the one side and the hold time on the other side. In cases in which sufficient minimum delay in the logic path can be ensured, widening of this window may be beneficial.

Characterization

Energy

It is important to consider the sources of energy consumed in the CSE and the correct setup for the characterization and comparison. Energy consumed by a CSE comes from various sources, not solely from the power supply, V_{dd}. Using V_{dd} as a point for measuring energy use can be misleading. Some CSEs, characterized by low internal energy consumption, represent a considerable load on the clock distribution network, thus taking a considerable amount of energy from the clock. Energy can be drawn from the data input as well. Therefore, the total power, E_{tot}, should account for all possible energy sources supplying the CSE [24]:

$$E_{tot} = E_{internal} + \sum_{inputs(D,CLK)} E_{driver}. \quad (12)$$

Delay

In characterizing delay, it is therefore appropriate to take into account the amount of time taken from the cycle, T, due to the insertion of the CSE. This represents D − Q delay, t_{DQ}, as was discussed. The question is whether the delay characterizing CSE should be measured as D − Q, D − $\bar{Q}$, or the worse of the two? It is argued in this paper that it is most appropriate to characterize the CSE with the worse of the two delays, because the critical path in a design may impose that scenario.

When simulating CSE, their output load should represent a worst-case scenario, but only the scenario that can realistically be expected in an actual design. Therefore, the load is applied to the output that has the longer delay. This is justified by the fact that delay of the critical path can always be improved by duplicating the CSE, thus reducing the load on the output that is not in the critical path. Therefore, the scenario of loading both outputs of the CSE when simulating for the worst-case delay was not applied. It is reasonable to expect this approach from both the skilled designer and the well-engineered synthesis tool.

In our reported measurements, we use 14 minimal-size inverters as a representative load. However, one could insert a properly sized inverter between the output and the load if this would produce a lesser delay. The theory of logical effort helps to determine this as well as to determine proper transistor sizes.

The general simulation setup is illustrated in **Figure 15** [25]. To provide fair comparison, the size of the data input is fixed for all CSEs. Also, the slope of the data signal is set to that of an FO4 inverter. This setting is typical for energy–delay-balanced designs. In the high-speed design methodology of Intel, Sun Microsystems**, and the former Digital Equipment Corporation**, the

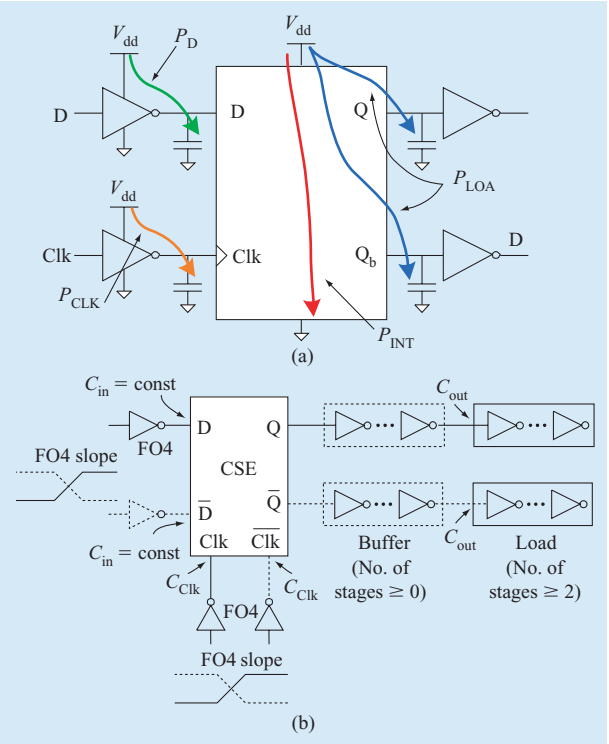

Figure 15

(a) Components of power consumption in a CSE. (b) General simulation setup. Reprinted from [25] with permission; © 2003 John Wiley & Sons, Inc.

FO3 inverter metric is more commonly used due to a more aggressive design style.

Figure of merit

It is well known that power can be traded for speed and that superior speed can always be obtained at the expense of higher power consumption. Thus, it is difficult to compare CSEs with each other. Various figures of merit have been used in the past. One misleading but commonly used factor is the power–delay product (PDP). It has been proven that the PDP as a figure of merit favors slower design, given that the energy consumed depends on the clock speed as well. Therefore, a more appropriate figure of merit is the energy–delay product (EDP) [26]. However, some recent results argue that ED^2P is even more appropriate, at least in high-performance systems.[1] The notion of hardware intensity was introduced in [27], which is the most comprehensive and detailed treatment of this subject. In our measurements, we use the EDP as a good optimization target for latches.

[1] K. Nowka and P. Hofstee, private communication, IBM Research Division, Austin, Texas, 2000.

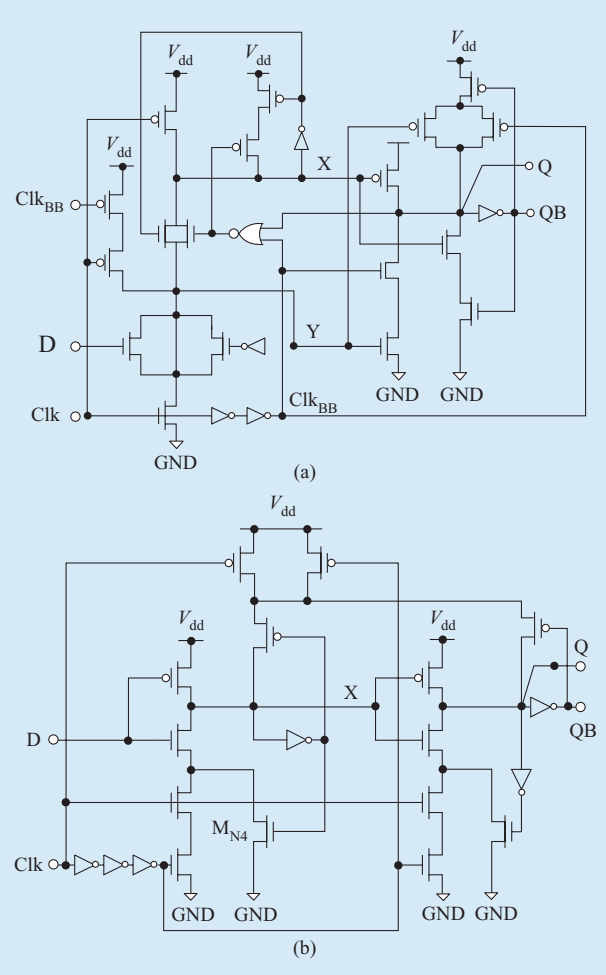

Figure 16

(a) Conditional capture flip-flop and (b) conditional pre-charge flip-flop. Reprinted from [30] with permission; © 2001 IEEE.

Designs for low power

An approximation of the energy consumed in a clocked storage element is given by

$$E_{\text{switching}} = \sum_{i=1}^{N} \alpha_{0\text{–}1}(i)\, C_i\, V_{\text{swing}}(i)\, V_{\text{dd}}\,, \qquad (13)$$

where N is the number of nodes in a CSE, C_i is the node capacitance, $\alpha_{0\text{–}1}(i)$ is the probability that transition occurs at node i, and V_{swing} is the voltage swing of node i. Starting from Equation (7), several commonly used techniques to minimize energy consumption can be derived:

- Reducing the number of active nodes and ensuring that when they are switching the capacitance is minimized.
- Reducing the voltage swing of the switching node.
- Reducing the voltage (technology scaling).
- Reducing the activity of the node.

These four approaches result in several known techniques used in low-power applications [4]. One of the most common is *clock gating*, which ensures that the storage elements in an inactive part of the processor are not switching. A thorough review of the common techniques for low power can be found in [28]. In this paper, we describe only briefly some recent techniques applicable to low-power design of CSEs.

Conditional-capture flip-flop

The motivation behind the conditional-capture technique is the observation that a considerable portion of power is consumed for driving internal nodes, even when the value of the output is not changed (low input activity). The conditional-capture technique attempts to minimize unnecessary switching of the CSE. By disabling redundant internal transitions, this technique achieves power reduction at little or no delay penalty. This makes it particularly attractive for use in high-performance VLSI systems. Conditional-capture flip-flop (CCFF) [29] operates on the principle of the J–K flip-flop; data can affect the flip-flop only if it will change the output. An improved version of a CCFF [30] reduces the overall EDP by up to 14% for 50% data activity when conditional capture is enabled. The total power saving of this flip-flop is more than 50% when there is no input activity (quiet inputs) [**Figure 16(a)**]. CSEs equipped with conditional features have advantageous properties in conditions of low data activity. In the implementation shown in Figure 16(a), conditional capture is achieved by direct sampling of the (inverted) input during the transparency window in a single-ended CCFF. However, this approach has drawbacks, the most important of which is increased setup time.

Conditional pre-charge flip-flop

The conditional pre-charge technique is a way of saving unnecessary expenditures of power in the flip-flop. It eliminates the power-consuming pre-charge operation in dynamic flip-flops when pre-charge is not required. A conditional pre-charge flip-flop (CPFF) [30] is shown in **Figure 16(b)**.

In CPFF, the pre-charge of the internal node is conditioned by the state of the output. With the assumption that the internal node, X, is pre-charged (to logic 1) when the clock is in the 0 state, the evaluation of node X happens during the flip-flop transparency window. If the input D is 1, X is discharged to 0, which is used to set the output Q to 1. Node X remains at logic 0 as long as both input D and output Q are at the logic 1 level. This allows savings in the power consumed on

unnecessary consecutive evaluations and pre-charges for D = 1. The logic 1-to-0 transition of the output is achieved by sampling high level on X in the transparency window. A conditional keeping function is applied at the output to avoid contention with the output keeper; the output is kept at logic 0 as long as X is 1 and, similarly, it is kept high outside the transparency window. Like a CCFF, this flip-flop has higher setup time for a 1-to-0 transition.

Dual-edge triggering

An approach suitable for high-performance, low-power applications is the use of dual-edge-triggered (DET) CSEs. Substantial power savings in the clock distribution network can be achieved by reducing the clock frequency by one half. This can be done if every clock transition is used as a time reference point instead of using only one transition of the clock (leading edge or trailing edge). The main advantage of this approach is that the system operates at half the frequency of a conventional single-edge clocking design style while obtaining the same data throughput [31]. Consequently, the power consumed by the clock generation and distribution system is roughly halved for the same clock load. In addition, less aggressive clock subsystems can be built which further reduce power consumption and clock uncertainties.

Dual-edge clocking is based on dual-edge-triggered storage elements (DETSE) capable of capturing data on both the rising and falling edge of the clock. The use of a dual-edge clocking strategy requires precise control of the arrival of both clock edges. This can be satisfied with reasonably low hardware overhead. In addition, the clock uncertainty resulting from the variation of the duty cycle can be partially absorbed by the storage element [32]. The two fundamental ways of building dual-edge CSEs are the latch-mux and flip-flop, as shown in **Figures 17(a)** and **17(b)**.

An example of a DET flip-flop (DETFF) design [33] is shown in **Figure 17(c)**. The circuit has a narrow data transparency window and a clockless output multiplexing scheme. Stage 1 is symmetric and consists of two PG latches. It creates the data-conditioned clock pulse on each edge of the clock. The clock pulse is created at node SX on the leading edge and node SY on the trailing edge of the clock. Stage 2 is a two-input NAND gate. It effectively serves as a multiplexer, relying on the fact that nodes SX and SY alternate in being pre-charged *high*, thus allowing the passage of the signal from the active stage (X or Y) alternatively. This type of output multiplexing is very convenient because it does not require clock control. The clock energy is mainly dissipated for pulse generation in the first stage. The clock load of this flip-flop is comparable to that of a single-edge-triggered flip-flop, thus making possible up to 50% power savings. This

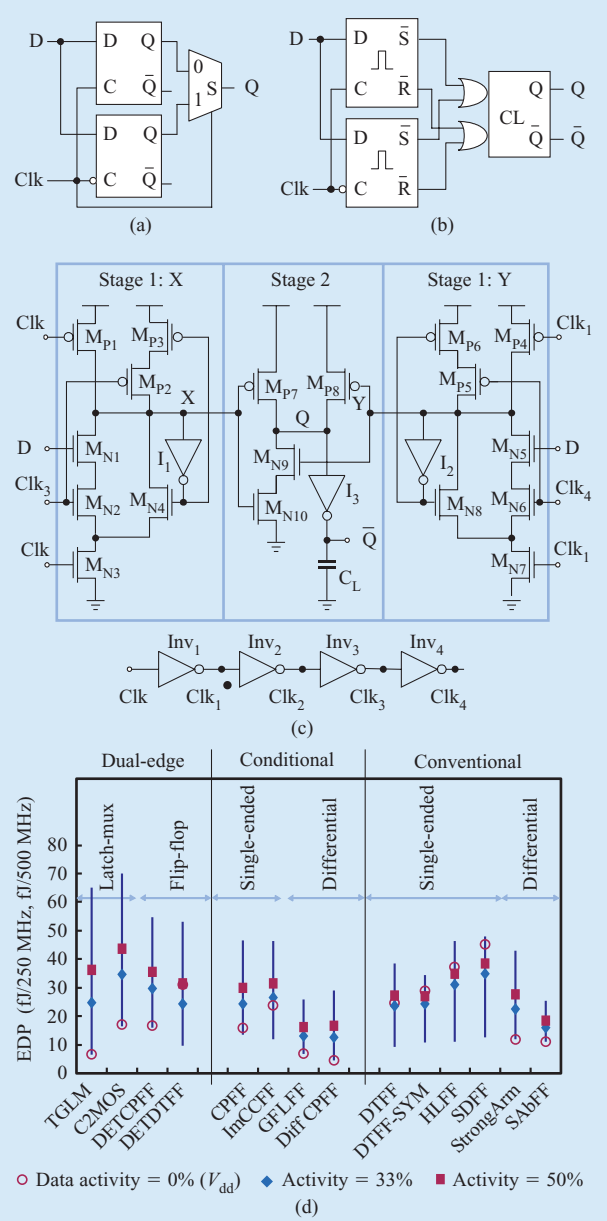

Figure 17

Dual-edge-triggered (DET) CSEs: (a) Latch-mux. (b) Flip-flop topology. (c) Dual-edge-triggered flip-flop (DETFF). Reprinted from [33] with permission; © 2002 IEEE. (d) Energy–delay product for different input activities of representative CSEs.

makes a DETFF a viable option for both high-performance and low-power systems. This statement is supported by the comparison results taken against a sample of conventional and conditional CSEs, as shown in **Figure 17(d)**. Stimulation conditions are listed in the results section.

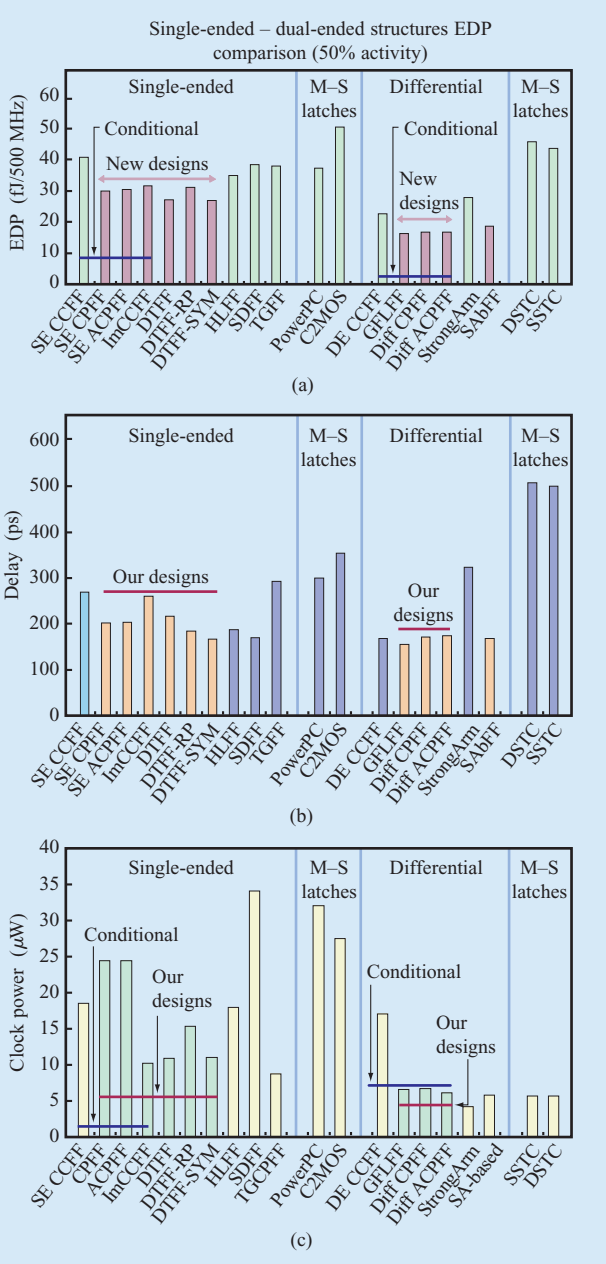

Figure 18

Results and comparison: (a) Energy–delay product comparison for various CSEs. (b) Speed comparison for various representative CSEs. (c) Component of energy taken from the clock distribution tree (power at 500 MHz).

Results

When comparing different CSEs with one another, several factors of importance for reliable design should be considered. Given that energy (power) and speed can be traded for one another, the two should always be related by taking the energy–delay product into consideration [**Figure 18(a)**]. Ideally it would be desirable to have a family of curves produced for each CSE on the energy–delay graph. Such a graph would provide CSE behavior in design space. However, the maximal achievable speed should also be considered separately, given that achieving maximal performance is dependent on the maximal speed achievable in the critical path. Speed comparisons for various CSEs are shown in **Figure 18(b)**.[2] The simulation conditions that are used to generate the comparisons shown in Figure 18 are given in **Table 1**, while an explanation of the initializations and a brief description of the simulated CSEs are given in **Table 2**. Both tables appear in the Appendix.

The power budget allocated to clocking and the clock distribution tree is very dependent on the clock load imposed by the CSE. Therefore, examining the energy distribution for various loads (the clock load in particular) is an important factor, as shown in **Figure 18(c)**.

Finally, the energy consumption as a function of data activity is of great importance (**Figure 19**). In low-power systems, it is particularly important to reduce the energy to as close to zero as possible when the system is not active. For zero data activity, two types of data activities must be considered—the CSE receiving a 0 and the CSE receiving a 1—because the two events can vary substantially.

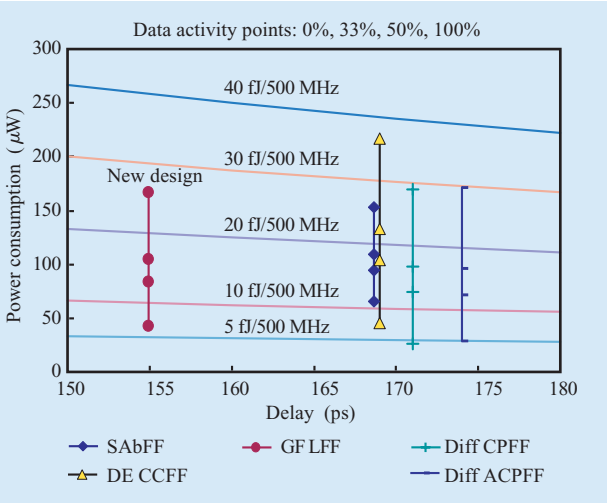

Figure 19

Energy consumption as a function of data activity for representative differential CSE.

[2] A database of comparative results exists at *www.ece.ucdavis.edu/acsel/*.

Conclusion

Clocking techniques and CSEs for high-performance and low-power systems are discussed. Given the rapid increase in clock frequency, not only in high-performance systems but in portable and low-power systems as well, it is important to consider clocking as the system reaches multiple-gigahertz speeds. For a complete analysis of representative CSEs, please see [25]. We expect that current clocking techniques will serve adequately as long as wire delay continues to scale. In the deep-submicron domain, this may not be sustainable much longer. At that point, the pipeline boundaries start to blur, and synchronous design will be possible only in limited domains on the chip. A mix of synchronous and asynchronous design may become necessary. This could represent the next design challenge, when more complex chips containing multiple processing systems begin to emerge.

Acknowledgment

We are grateful for the contributions to this paper made by Nikola Nedovic, Vladimir Stojanovic, and Dejan Markovic.

References

1. S. Borkar, "Design Challenges of Technology Scaling," *IEEE Micro* **19,** No. 4, 23–29 (August 1999).
2. P. E. Gronowski, W. J. Bowhill, R. P. Preston, M. K. Gowan, and R. L. Allmon, "High-Performance Microprocessor Design," *IEEE J. Solid-State Circuits* **33,** No. 5, 676–686 (May 1998).
3. E. G. Friedman, Ed., *Clock Distribution Networks in VLSI Circuits and Systems*, IEEE Press, Piscataway, NJ, 1995.
4. V. G. Oklobdzija, Ed., *High-Performance System Design: Circuits and Logic*, IEEE Press, Piscataway, NJ, 1999.
5. D. W. Bailey and B. J. Benschneider, "Clocking Design and Analysis for a 600-MHz Alpha Microprocessor," *IEEE J. Solid-State Circuits* **33,** No. 11, 1627–1633 (November 1998).
6. J. Schutz and R. Wallace, "A 450MHz IA32 P6 Family Microprocessor," *Proceedings of the IEEE International Solid-State Circuits Conference* (*ISSCC*), *Digest of Technical Papers*, February 1998, pp. 236–237.
7. W. P. Burleson, M. Ciesielski, F. Klass, and W. Liu, "Wave-Pipelining: A Tutorial and Research Survey," *IEEE Trans. Very Large Scale Integr.* (*VLSI*) *Syst.* **6,** No. 3, 464–474 (September 1998).
8. D. Harris and M. A. Horowitz, "Skew-Tolerant Domino Circuits," *IEEE J. Solid-State Circuits* **32,** No. 11, 1702–1711 (November 1997).
9. D. Harris, *Skew-Tolerant Circuit Design*, Morgan Kaufmann Publishers, San Francisco, 2001.
10. L. W. Cotten, "Circuit Implementation of High-Speed Pipeline Systems," *Proceedings of the Fall Joint Computer Conference* (*AFIPS*), 1965, pp. 489–504.
11. IBM Corporation, *LSSD Rules and Applications*, Manual 3531, Release 59.0, March 29, 1985.
12. G. Gerosa, S. Gary, C. Dietz, D. Pham, K. Hoover, J. Alvarez, H. Sanchez, P. Ippolito, T. Ngo, S. Litch, J. Eno, J. Golab, N. Vanderschaaf, and J. Kahle, "A 2.2 W, 80 MHz Superscalar RISC Microprocessor," *IEEE J. Solid-State Circuits* **29,** No. 12, 1440–1452 (December 1994).
13. D. W. Dobberpuhl, "Circuits and Technology for Digital's StrongARM and ALPHA Microprocessors," *Proceedings of the 17th Conference on Advanced Research in VLSI*, 1997, pp. 2–11.
14. W. C. Madden and W. J. Bowhill, "High Input Impedance, Strobed Sense-Amplifier," U.S. Patent 4,910,713, March 1990.
15. B. Nikolic and V. G. Oklobdzija, "Design and Optimization of Sense-Amplifier-Based Flip-Flops," *Proceedings of the 25th European Solid-State Circuits Conference* (*ESSCIRC'99*), 1999, pp. 410–413.
16. V. Stojanovic and V. G. Oklobdzija, "Flip-Flop," U.S. Patent 6,232,810, May 2001.
17. B. Nikolic, V. G. Oklobdzija, V. Stojanovic, W. Jia, J. Chiu, and M. Leung, "Improved Sense-Amplifier-Based Flip-Flop: Design and Measurements," *IEEE J. Solid-State Circuits* **35,** No. 6, 876–884 (June 2000).
18. B. Nikolic, V. Stojanovic, V. G. Oklobdzija, W. Jia, J. Chiu, and M. Leung, "Sense Amplifier Based Flip-Flop," *Proceedings of the IEEE International Solid-State Circuits Conference* (*ISSCC*), *Digest of Technical Papers*, 1999, pp. 282–283.
19. H. Partovi, R. Burd, U. Salim, F. Weber, L. DiGregorio, and D. Draper, "Flow-Through Latch and Edge-Triggered Flip-Flop Hybrid Elements," *Proceedings of the IEEE International Solid-State Circuits Conference* (*ISSCC*), *Digest of Technical Papers*, 1996, pp. 138–139.
20. F. Klass, "Semi-Dynamic and Dynamic Flip-Flops with Embedded Logic," *Proceedings of the Symposium on VLSI Circuits*, *Digest of Technical Papers*, 1998, pp. 108–109.
21. N. Nedovic and V. G. Oklobdzija, "Dynamic Flip-Flop with Improved Power," *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors* (*ICCD'00*), 2000, pp. 323–326.
22. J. Tschanz James, S. Narendra, Z. Chen, S. Borkar, M. Sachdev, and V. De, "Comparative Delay and Energy of Single Edge-Triggered and Dual Edge-Triggered Pulsed Flip-Flops for High-Performance Microprocessors," *Proceedings of the International Symposium on Low Power Electronics and Design*, 2001, pp. 147–152.
23. S. H. Unger and C.-J. Tan, "Clocking Schemes for High-Speed Digital Systems," *IEEE Trans. Computers* **C-35,** No. 10, 880–895 (October 1986).
24. V. Stojanovic and V. G. Oklobdzija, "Comparative Analysis of Master–Slave Latches and Flip-Flops for High-Performance and Low-Power VLSI Systems," *IEEE J. Solid-State Circuits* **34,** No. 4, 536–548 (April 1999).
25. V. G. Oklobdzija, V. Stojanovic, D. Markovic, and N. Nedovic, *Digital System Clocking: High-Performance and Low-Power Aspects*, John Wiley & Sons, Inc., Hoboken, NJ, 2003.
26. M. Horowitz, T. Indermaur, and R. Gonzalez, "Low-Power Digital Design," *Proceedings of the IEEE Symposium on Low-Power Electronics*, 1994, pp. 8–11.
27. V. Zyuban and P. Strenski, "Unified Methodology for Resolving Power–Performance Tradeoffs at the Microarchitectural and Circuit Level," *Proceedings of the International Symposium on Low-Power Electronics and Design*, 2002, pp. 166–171.
28. T. Kuroda and T. Sakurai, "Overview of Low-Power ULSI Circuit Techniques," *IEICE Trans. Electron.* **E78-C,** No. 4, 334–344 (April 1995).
29. B. S. Kong, S. S. Kim, and Y. H. Jun, "Conditional Capture Flip-Flop Technique for Statistical Power Reduction," *Proceedings of the IEEE International Solid-*

State Circuits Conference (*ISSCC*), *Digest of Technical Papers*, 2000, pp. 290–291.
30. N. Nedovic, M. Aleksic, and V. G. Oklobdzija, "Conditional Techniques for Small Power Consumption Flip-Flops," *Proceedings of the 8th IEEE International Conference on Electronics, Circuits and Systems*, 2001, pp. 803–806.
31. N. Nedovic, M. Aleksic, and V. G. Oklobdzija, "Conditional Pre-Charge Techniques for Power-Efficient Dual-Edge Clocking," *Proceedings of the International Symposium on Low-Power Electronics and Design*, 2002, pp. 56–59.
32. M. Saint-Laurent, V. G. Oklobdzija, S. S. Singh, and M. Swaminathan, "Optimal Sequencing Energy Allocation for CMOS Integrated Systems," *Proceedings of the International Symposium on Quality Electronic Design*, 2002, pp. 94–99.
33. N. Nedovic, W. W. Walker, V. G. Oklobdzija, and M. Aleksic, "A Low Power Symmetrically Pulsed Dual Edge-Triggered Flip-Flop," *Proceedings of the European Solid-State Circuits Conference* (*ESSCIRC'02*), 2002, pp. 399–402.
34. N. Nedovic and V. G. Oklobdzija, "Hybrid Latch Flip-Flop with Improved Power Efficiency," *Proceedings of the 13th Symposium on Integrated Circuits and Systems Design* (*SBCCI'00*), 2000, pp. 211–215.
35. Y. Suzuki, K. Odagawa, and T. Abe, "Clocked CMOS Calculator Circuitry," *IEEE J. Solid-State Circuits* **8,** No. 6, 462–469 (December 1973).
36. J. Yuan and C. Svensson, "New Single-Clock CMOS Latches and Flipflops with Improved Speed and Power Savings," *IEEE J. Solid-State Circuits* **32,** No. 1, 62–69 (January 1997).
37. F. Klass, C. Amir, A. Das, K. Aingaran, C. Truong, R. Wang, A. Mehta, R. Heald, and G. Yee, "A New Family of Semidynamic and Dynamic Flip-Flops with Embedded Logic for High-Performance Processors," *IEEE J. Solid-State Circuits* **34,** No. 5, 712–716 (May 1999).
38. J. Montanaro, R. T. Witek, K. Anne, A. J. Black, E. M. Cooper, D. W. Dobberpuhl, P. M. Donahue, J. Eno, W. Hoeppner, D. Kruckemyer, T. H. Lee, P. C. M. Lin, L. Madden, D. Murray, M. H. Pearce, S. Santhanam, K. J. Snyder, R. Stehpany, and S. C. Thierauf, "A 160-MHz, 32-b, 0.5-W CMOS RISC Microprocessor," *IEEE J. Solid-State Circuits* **31,** No. 11, 1703–1714 (November 1996).
39. A. G. M. Strollo, E. Napoli, and C. Cimino, "Low Power Double Edge-Triggered Flip-Flop Using One Latch," *IEE Electron. Lett.* **35,** No. 3, 187–188 (February 1999).

Received December 4, 2002; accepted for publication April 21, 2003

Vojin G. Oklobdzija *Department of Electrical and Computer Engineering, University of California at Davis, 3007 Kemper Hall, Davis, California 95616 (vojin@ece.ucdavis.edu).*
Dr. Oklobdzija is an IEEE Fellow and Distinguished Lecturer of the IEEE Solid-State Circuits Society. He received a Dipl. Ing. degree from the Electrical Engineering Department of the University of Belgrade in 1971, and his Ph.D. from the University of California at Los Angeles in 1982. From 1982 to 1991 he was at the IBM Thomas J. Watson Research Center, where he made contributions to the development of RISC processors and supercomputer design. In the course of this work, he obtained several patents, the most notable one on register renaming, with John Cocke and Greg Grohosky, which enabled a new generation of computers. From 1988 to 1990 he was an IBM visiting faculty member at the University of California at Berkeley. Since 1991, he has been a professor at the University of California and has served as a consultant to many companies, including Sun Microsystems, Bell Laboratories, Hitachi, Fujitsu, SONY, Intel, and Siemens Corporation, where he was the principal architect for the TriCore processor. He holds 11 U.S. patents and seven international patents, with six patents currently pending. Dr. Oklobdzija serves as associate editor for the *IEEE Transactions on Computers*, *IEEE Transactions on VLSI Systems*, *Journal of VLSI Signal Processing*, and *IEEE Micro*. He served for six years on the ISSCC program committee, among numerous other conference committees. He was a General Chairman of the 13th Symposium on Computer Arithmetic. Dr. Oklobdzija has published more than 140 papers, three books, and dozens of book chapters in the areas of circuits and technology, computer arithmetic, and computer architecture. He has given more than 150 invited talks and short courses in the United States, Europe, Latin America, Australia, China, and Japan.

Appendix: Tables 1 and 2

Table 1 Simulation conditions.

V_{dd} (V)	T (°C)	*Technology* (μm)	*FO4 delay* (ps)	*Transparency width* (μm)	*Load*	*Clock* (MHz)	*Data/Clk slopes* (ps)
1.8 nominal	27	0.18	75	Min. 0.36, Max. 10	14 min. invert	500 (250 DEFF)	Ideal signal 100

Table 2 Characteristics of the latches and flip-flops shown in Figures 18(a)–(c) and Figure 19.

Symbol	*Description (reference)*	*Features*
ACPFF	Single-ended conditional-precharge flip-flop alternate version [34]	Same as single-ended CPFF; modified circuit design to reduce power due to input glitching.
C2MOS	C2MOS latch [35]	Early version of M–S latch that uses popular C2MOS circuit technique; good power-consumption characteristics; however, it suffers from large delay.
CPFF	Single-ended conditional-pre-charge flip-flop [34]	Reduced internal switching activity—recharge—based on input switching.
DE CCFF	Dual-ended (differential) conditional-capture flip-flop [29]	Reduced internal switching based on input switching activity; differential circuit implementation.
Diff ACPFF	Differential alternate conditional-pre-charge flip-flop	Differential implementation of ACPFF.
Diff CPFF	Differential conditional-pre-charge flip-flop	Differential implementation of conditional pre-charge flip-flop.
DSTC	Dynamic single-transistor-clocked (M–S latch) [36]	Dynamic implementation of "single-transistor-clocked" M–S latch; small clock delay and power, but large delay.
DTFF	Dynamic flip-flop with improved power (improved SDFF) [21]	Systematic derivation of single-ended dynamic (pre-charge-evaluate) flip-flop.
DTFF-RP	Modified DTFF with improved pre-charge [21]	Modification of DTFF to improve reliability.
DTFF-SYM	Modification of DTFF with symmetric output [21]	Modification of DTFF with push-pull latch for faster operation.
GFLFF	Improved circuit implementation of differential CCFF	CCFF does not use explicit "transparency window."
HLFF	Hybrid-latch flip-flop based on transparency window [19]	First flip-flop based on "transparency window" principle.
ImCCFF	Improved conditional-capture flip-flop [30]	Improved circuit design of CCFF; same principle of operation as CCFF.
PowerPC*	Master–slave latch [12]	Basic implementation of M–S latch with transmission gates; excellent power-consumption characteristics with moderate delay.
SAbFF	Improved SAFF [11, 15]	Differential sense-amplifier flip-flop with symmetric push-pull latch to improve speed.
SDFF	Semi-dynamic flip-flop [37]	Same principle as HLFF; dynamic circuit design to improve speed.
SE CCFF	Single-ended conditional-capture flip-flop [29]	Reduced internal switching (evaluation of the pulse generator) based on input switching activity; single-ended circuit implementation.
SSTC	Static single-transistor-clocked (M–S latch) [36]	Static implementation of "single-transistor-clocked" M–S latch; small clock delay and power, but large delay.
StrongArm	StrongARM flip-flop (SAFF) [38]	Differential sense-amplifier flip-flop with *ad hoc* static circuit implementation; small clock load.
TGCPFF, TGFF	Transmission gate clock-pulsed flip-flop [39]	Straightforward implementation of transmission-gate pulsed latch.