Clocking and Latches ©

by

Prof. Vojin G. Oklobdzija

Notes for EEC180B*

Spring 1999

University of California

Davis

1. CLOCK SUB-SYSTEM

Proper timing and clock system design are one of the most critical components in digital system. This has been summarized in the words of Prof. Steven Unger:

> "*Despite the deceptively simple outward appearance of the clocking system, it is often a source of considerable trouble in actual systems.*"

This chapter is dedicated to the subject of clocking in digital systems.

The function of the clock in the digital system can be compared to the function of a metronome in the music. It designates a beginning of a music score, exact moment when certain notes are to be played by particular instruments in orchestra, designates the end of the part or section, i.e. provides synchronization for various instruments in the orchestra during various parts and periods of the score that is being performed. Similarly, in digital system the clock designates the exact moment when the signal is to change as well as its final value is to be captured, when the logic is active or inactive. Finally, all the logic operations have to finish before the tick of the clock and the final values of the signals are being captured at the tick of the clock. Therefore, the clock provides the time reference point which determines the movement of data in the digital system. This definition fits the description of the synchronous systems, which will be the subject of this chapter.

Asynchronous and self-timed systems are not covered here, though, they have been capturing attention in the research and academia for quite some time. The reason why self-timed systems are drawing attention is an increasing difficulty to control the clock

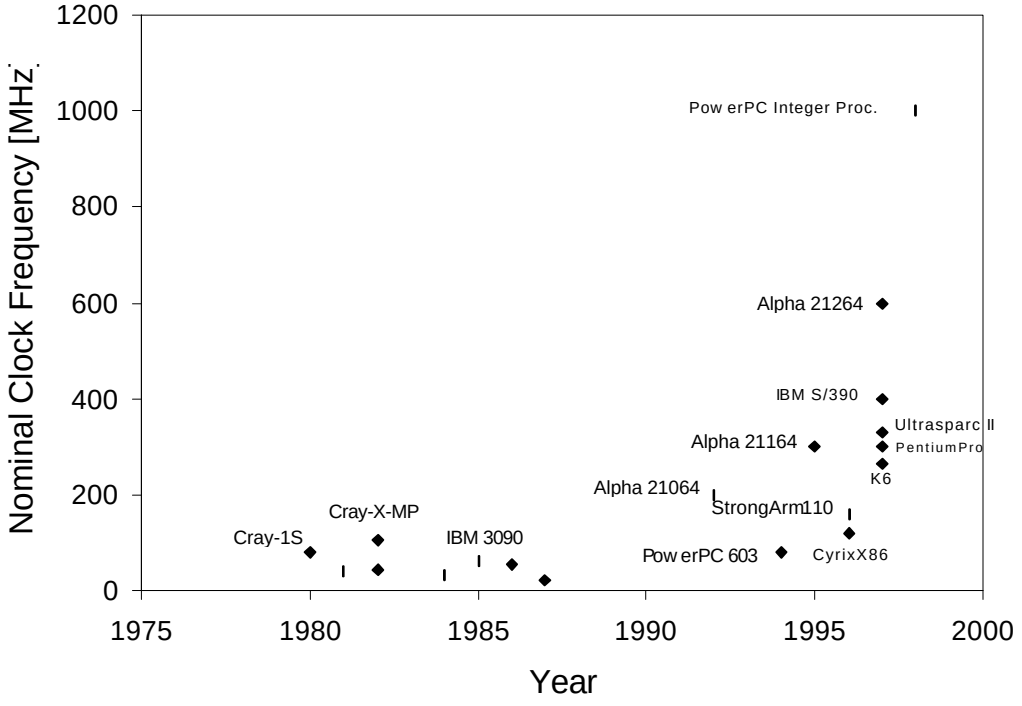


Fig. 1.1. *The trend in Clock frequency over the years*

skew and distribution of the clock, as the operating frequency of today's systems keeps increasing, reaching 600MHz or even 1GHz and beyond, for the next generation of the systems that are currently under development. The trend in clocking speed is shown in Fig.1.1, indicating the exponential growth in clock frequency, over the years.

1.1. Clock Signals

Clocks are defined as pulsed, synchronizing signals that provide the time reference for the movement of data in the synchronous digital system. The clocking in a digital system can be either single phase, multi-phase (usually two-phase) or edge-triggered, as illustrated in Fig.1.2.

The dark rectangles in the figure represent the interval during which the bi-stable element samples its data input. Fig. 4.2 shows the possible types of clocking techniques and corresponding general finite-state machine structures:

a) single-phase clocking and single-phase latch machine, Fig.1.2 (a)

b) edge-triggered clocking and flip-flop machine, Fig.1.2 (b)

c) two-phase clocking and two-phase latch machine with single latch Fig.1.2 (c)

d) two-phase clocking and two-phase latch machine with double latch Fig.1.2 (d)

In Fig.1.2 (c) and (d), W_j is the pulse width of the phase j and g_{ij} is the inter-phase gap from phase i to phase j; if $g_{ij} > 0 \Rightarrow$ two-phase, non-overlapping, if $g_{ij} < 0 \Rightarrow$ two-phase, overlapping clocking scheme.

The multi-phase design typically extends to three, but not more than four non-overlapping phases.

Multi-phase clocking has been used in the early dynamic MOS circuits at the very beginning of the VLSI, as well as in the systems where this is dictated by the nature of the computation. In the main-frame computer systems such as IBM's, the two phase clocking was almost exclusively used and it was incorporated in the LSSD (Level-Sensitive Scan Design) discipline. The reason for that is that the two non-overlapping clocks provide most reliable and robust clocking system that fits well into the design for testability methodology that was incorporated in LSSD.

Single phase clock was used in simpler systems with less stringent requirements such as those found in LSSD (some of the high-performance systems used single-phase clocking, such as CRAY 1). However, due to the increased demand for performance and increasing difficulties with handling clock-skews, single-phase clock has seen its resurgence. This is due to the increasing penalties from the clock-skew, which does not scale down proportionally as the technology makes for ever faster transistor. What this means is that the percentage of the cycle which is lost due to the clock-skew has increased to the point where it can not be tolerated. Therefore, some sacrifice in reliability has to be made as a compromise to performance. Given that machine cycle may take one or several clock cycles, as illustrated in Fig.1.3 (especially in the micro-programmed systems), the

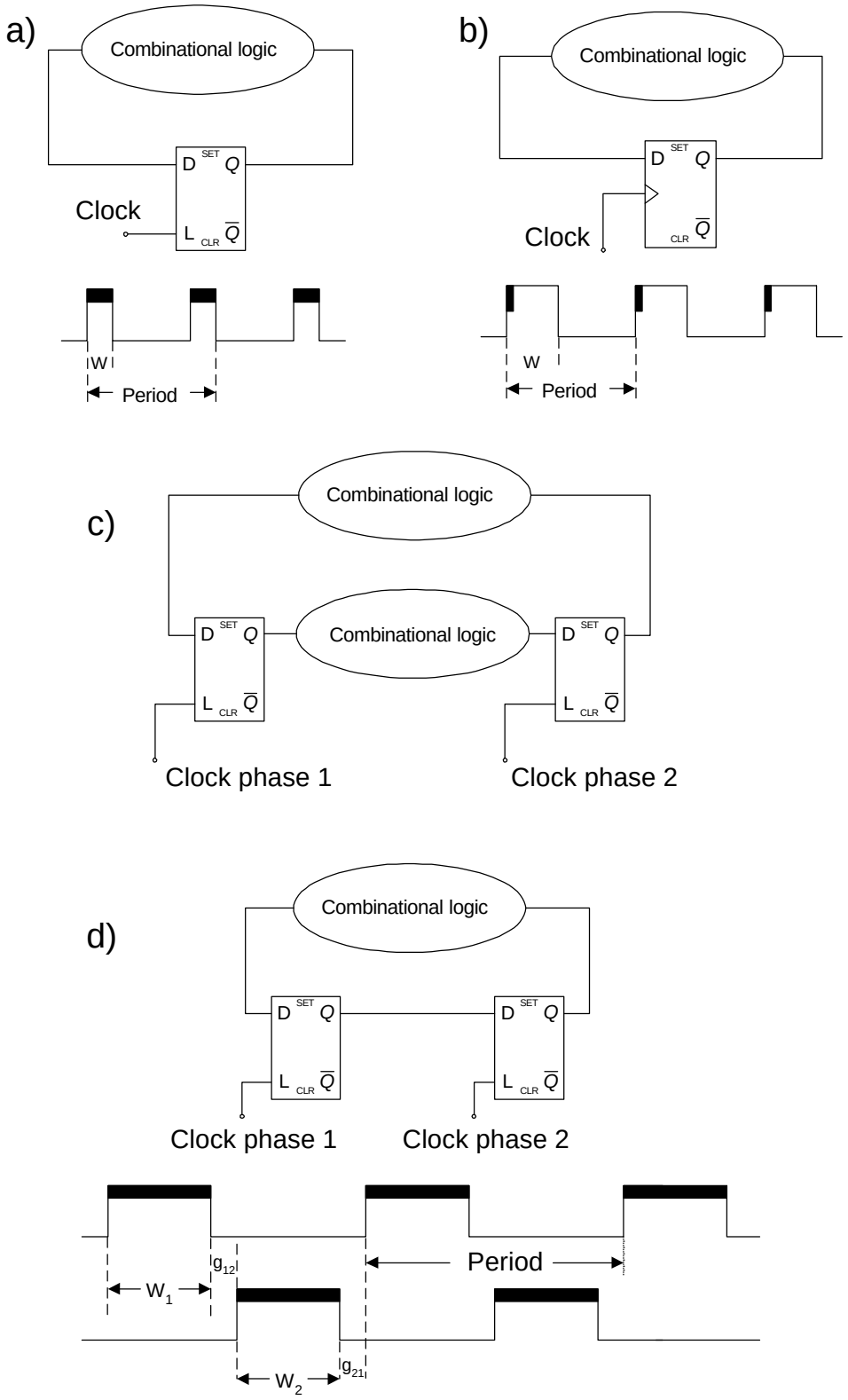


Fig. 1.2. *System clocking waveforms and general finite-state machine structures*

overhead for the clock-skew directly influences the machine cycle and therefore the performance of the overall system.

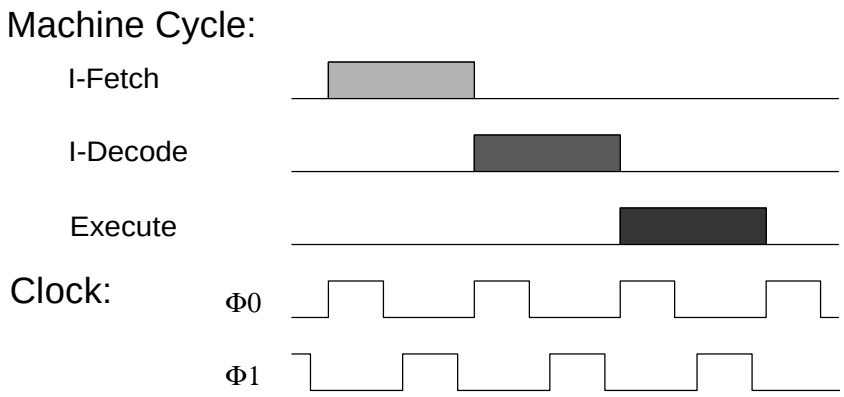


Fig. 1.3. *Relationship between the Clock and the Machine cycle*.

Timing analysis strongly depends on the type of clocking used in the system. Single-phase systems and multi-phase overlapping systems require more extensive timing analysis than multi-phase non-overlapping and edge-triggered systems. The single-phase and multi-phase overlapping timing requirements are bounded by both *short* and *long paths*. This constrain is illustrated in Fig.1.4, for the simplified case where *setup* and *hold* times are set to zero. The advantage of these systems over their non-overlapping counterparts is that they have, so called, *cycle stealing* feature, which effectively reduces the clock cycle and boosts the performance, at the price of more difficult timing analysis and less robust operation.

Fig.1.4 (a) shows the timing constraints for a single-phase system:

1. LS data available at t_1
2. LS data must arrive at LD after t_2 (or be latched up in Cycle $1 \Rightarrow$ *short path*)
3. LS data arrives at LD by t_3 (or reduces the path length available in Cycle 2)
4. LS data must arrive at LD before t_4 (or be latched up in Cycle $3 \Rightarrow$ *long path*)

Fig.1.4 (b) presents the timing constraints for two-phase overlapping systems:

1. L_2S data available at t_1
2. L_2S data must arrive at L_2D after t_2 (or be latched up in Cycle $1 \Rightarrow$ *short path*)
3. L_2S data must arrive at L_2D by t_4 (or violate system cycle time requirement)
4. L_2S data must arrive at L_1D before t_5 (or be latched up in Cycle $3 \Rightarrow$ *long path*)

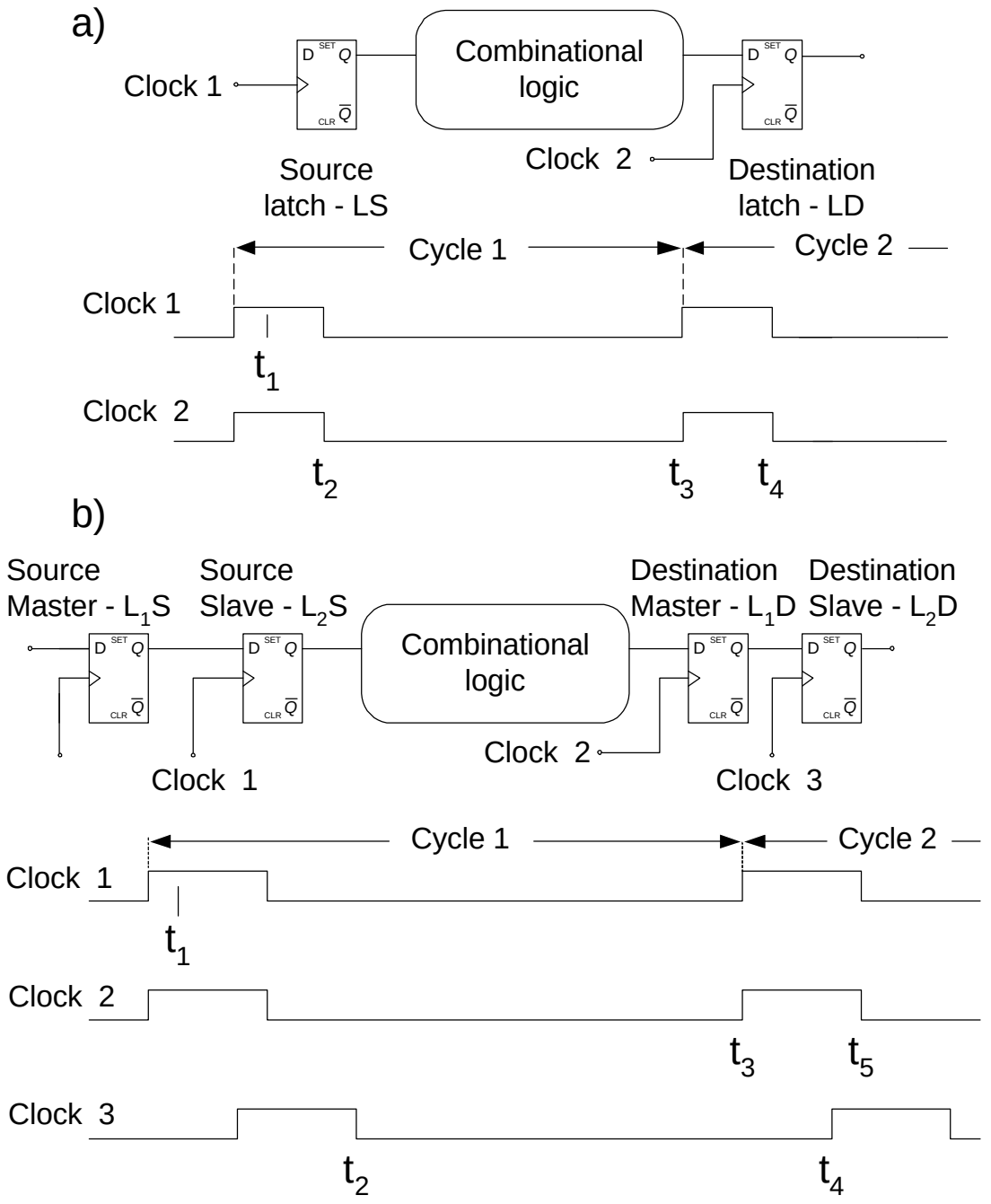


Fig. 1.4. *Timing constraints in single-phase and two-phase overlapping clocking techniques*

1.2. Bi-stable Elements

In order to define the clocking of the system properly, the nature and behavior of the bi-stable element, often referred as a "latch" or "Flip-Flop" needs to be specified precisely. It is quite common to find both terms "Flip-Flop" and a "Latch" to be used indiscriminately for the bi-stable element used in the synchronous systems. In the further text we would like to make a distinction between the "Flip-Flop" and a "Latch".

"Latch":

is a device capable of storing the value of the input D in conjunction with the clock C and providing it at its output Q. The latch has a following relationship between the input D and the clock C: While C=0 the output Q remains constant regardless of the value of D (the latch is "opaque"). While C=1 the output Q=D and it reflects all the changes of D (the latch is *"transparent"*). Often we describe such a behavior of a latch as *"level-sensitive"* (the behavior of the latch is dependent on the value of the clock - not the changes). Behavior of an ideal Latch is illustrated in Fig.1.5.

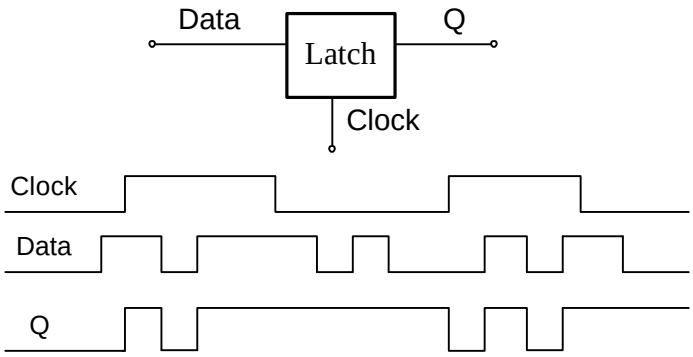


Fig. 1.5. *Signal relationship of an ideal Latch*

"Flip-Flop":

is defined as a bi-stable memory element with the same inputs and outputs as a "latch", Fig.1.6. However, the output Q responds to the changes of D only at the moment the clock C is making transitions. We define this as being *"edge triggered"*. The internal mechanisms of the "flip-flop" and that of a "latch" are entirely different. We further define a "Flip-Flop" as a *"leading edge triggered"* if the output Q assumes a value of the input D as a result of the transition of the clock C from 0-to-1. Conversely in a *"negative edge triggered"* Flip-Flop the output Q assumes a value of the input D as a result of the transition of the clock C from 1-to-0. It is also possible to build *a "double-edge triggered"* flip-flop that responds to both: *leading* and *trailing* edge of the clock C. Such

flip-flop implementations, first published in 1981 by Unger are starting to gain attention recently, given the increasing demand for performance.

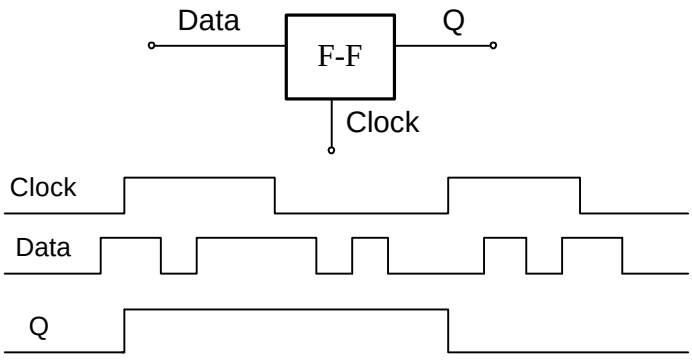


Fig. 1.6. *Signal relationship of an ideal leading-edge triggered Flip-Flop*

1.2.1. Difference between a "Master-Slave Latch" and a "Flip-Flop":

The difference between a *"Master-Slave Latch"* and a *"Flip-Flop"* is often not understood and there is a wide spread misconception in which a *"Master-Slave Latch"* combination is referred to as a *"Flip-Flop"*. It is important to understand a fundamental difference between a *"Master-Slave Latch"* and a *"Flip-Flop"* which exists in their latching (triggering) mechanisms. While in a *"Master-Slave Latch"* a latching mechanism occurs as result of the clock reaching a logic value of one (level) allowing the data to be locked, this mechanism in a *"Flip-Flop"* is fundamentally different. What causes the data to be latched in a *"Flip-Flop"* is not the value (level) of the clock but the transition from 0 to 1 (or 1 to 0 - *"negative edge triggered Flip-Flop"*). The rate of this transition is what causes the latching to occur in a *"Flip-Flop"*. This dependency of the transition rate, in order to make a reliable latching, is precisely the reason why the use of *"Flip-Flops"* represents a hazard and for the very same reason the use of *"Flip-Flop"* has been forbidden in some design disciplines (such as IBM's LSSD).

In order to picture the "latching" mechanism in flip-flop structures, we took the SN7474 flip-flop as an example. A schematic diagram of a D-type *"Flip-Flop"* (SN 7474) is presented in Fig.1.7 and the analysis of the "latching" mechanism in Fig.1.8 and Fig.1.9. It is a useful exercise to analyze what happened in this *"Flip-Flop"* when the clock changes from 0 to 1.

The whole scheme has two functional segments. The left part of the scheme (with output nodes $\overline{S}$ and $\overline{R}$, in Fig.1.8 and Fig.1.9) is multiplexing one of the two clocked NAND gates depending on the state of D. The state of D before the rising edge of the clock determines which clocked NAND gate is to be isolated. The output of the isolated NAND

gate remains high. The output of the selected clocked NAND gate goes low at the rising edge of the clock causing the regenerative process in S-R latch.

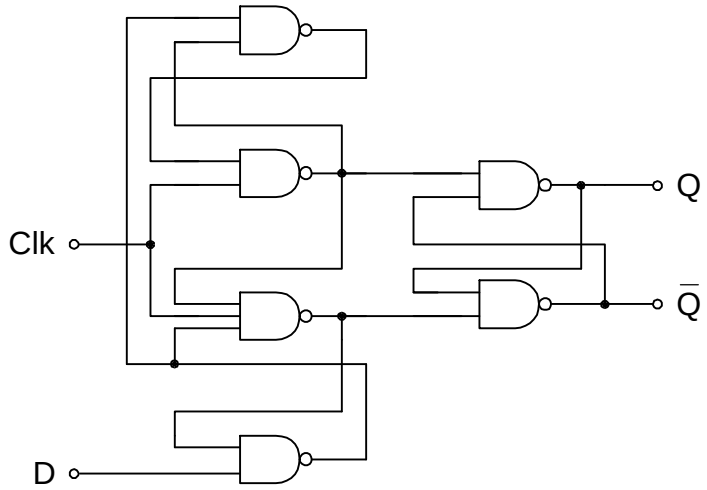


Fig. 1.7. *SN 7474, leading-edge triggered Flip-Flop*

On the leading edge of the clock, depending of the state of the input D, a flow of signals occurs and causes the latching to a new state. If we assume that the delay of all the NAND gates is the same, τ, and that leading edge of the clock occurs at 0 time point, we will have changes in different nodes of the circuit in moments of τ, 2τ, and 3τ, as the signal progresses through the circuit. That is illustrated in Fig.1.8 for 0 to 1 transition of Q and on Fig.1.9 for 1 to 0 transition.

For the sake of clarity only the first part of the clock cycle is shown in the diagrams. In the remaining part of the cycle, low level on Clk node forces both nodes $\overline{S}$ and $\overline{R}$ to high level, making them ready for another cycle.

The key point in the edge-triggered behavior of this flip-flop is that once the clock has made a transition from 0 to 1, one of the nodes $\overline{S}$ or $\overline{R}$ (depending of the state of D before the leading-edge of the clock) makes a $1 \rightarrow 0$ transition thus disabling the further impact of D on the value of the nodes $\overline{S}$ and $\overline{R}$.

In Fig.1.8 node $\overline{S}$ makes a transition $1 \rightarrow 0$ thus disabling the further changes of the nodes A and $\overline{R}$. In this way input D is isolated and can not influence the state of nodes $\overline{S}$ and $\overline{R}$ while clock is 1.

In Fig.1.9 node $\overline{R}$ makes a transition $1 \rightarrow 0$ thus disabling the further changes of the nodes B and $\overline{S}$. In this way input D is isolated and can not influence the state of nodes $\overline{S}$ and $\overline{R}$ while clock is 1.

It is worth to mention the issue of the races in this structure. Races are caused only if Data input changes in the window either around the leading edge of the Clock (for Data:

0 to 1) or around one gate delay before the leading edge of the Clock (for Data: 1 to 0). The width of the critical window is determined by the parameters of the NAND gates in the first (multiplexing) stage of a Flip-Flop.

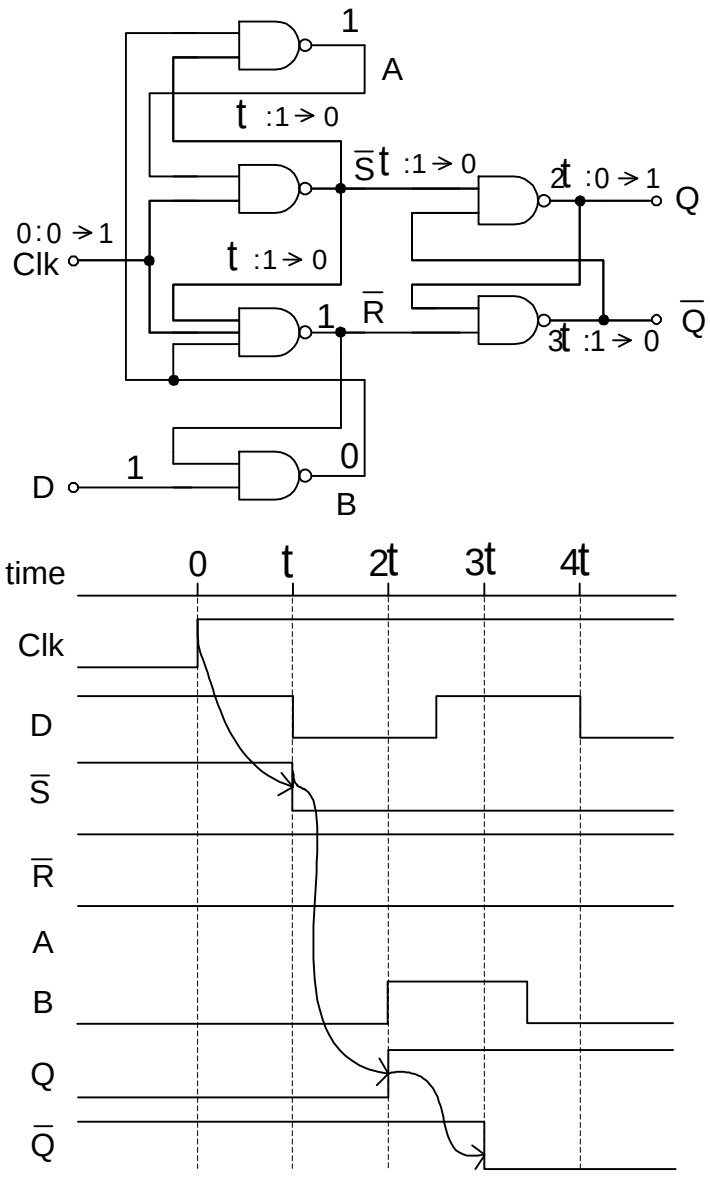


Fig. 1.8. *SN 7474, Signal flow for **0** to **1** transition of Q*

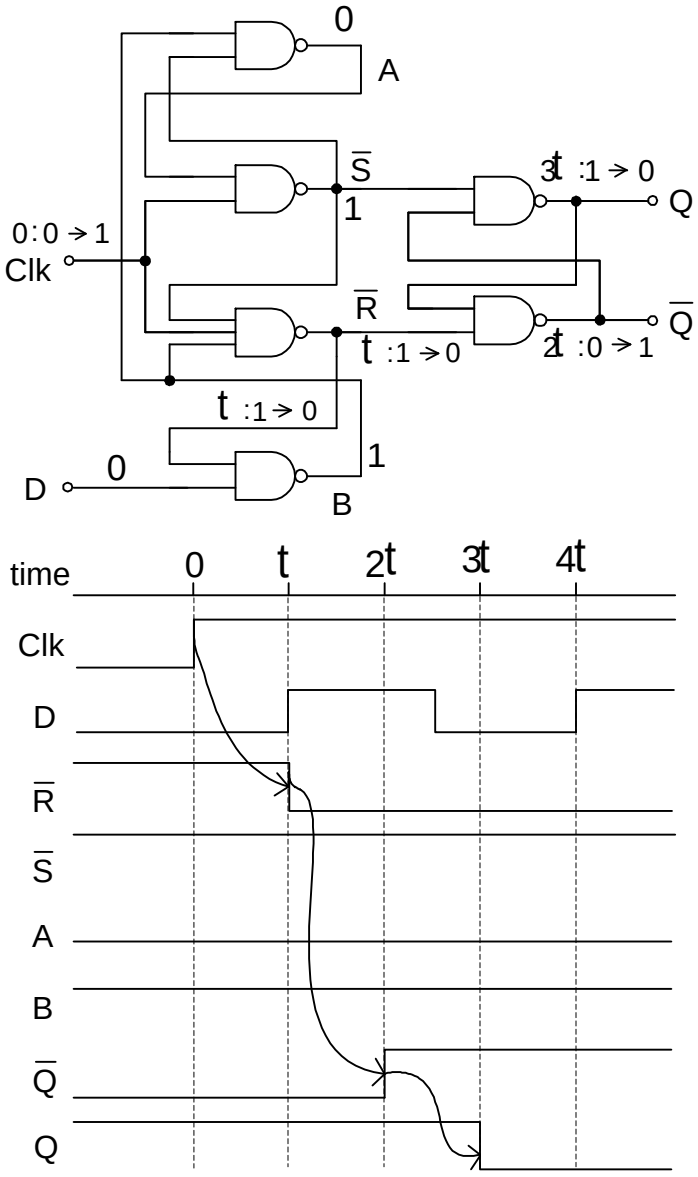
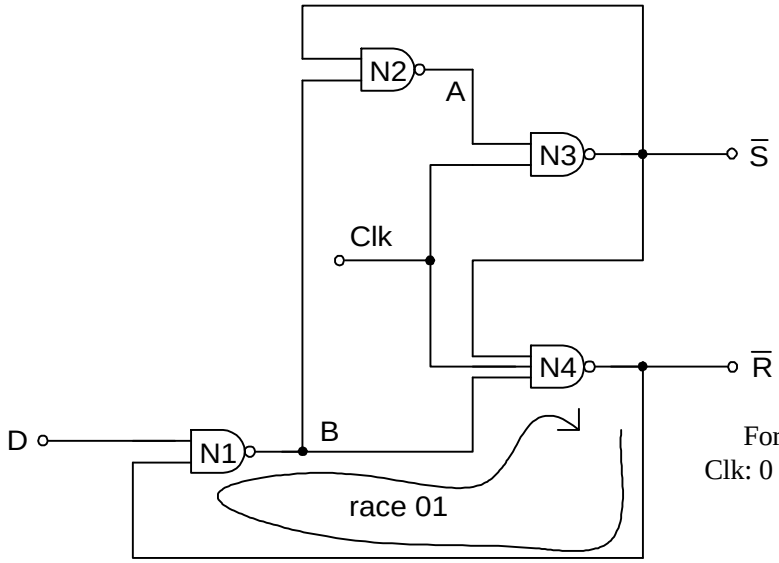


Fig. 1.9. *SN 7474, Signal flow for **1** to **0** transition of Q*

The discussion of the races is presented in Fig.1.10 and Fig.1.11. If Data changes in the first part of the critical window, the change will have a chance to win the race. If Data changes in the second part of the window, the change will cause the race, but it will not win, and will not be remembered. In ideal case of the equal delays of the NAND gates N1&N4 for race 01 and N2&N3 for race 10, the first part and the last part of the window will be equal. Any change in parameters of the NAND gates will cause one or the other part of the window to be bigger. The races are very hard to present on the logic level and

in the way of changes in discrete time intervals of NAND gate delays because they actually occur between these discrete points. This is why we have chosen to present the signals in the fashion of windows.



For D: 0 to 1 in a window around
Clk: 0 to 1 there is a race in N1-N4 loop

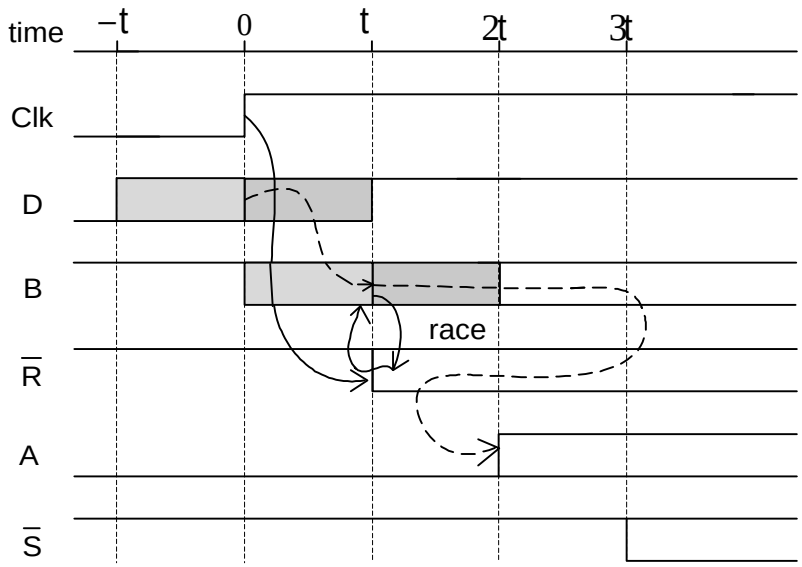


Fig. 1.10. *SN7474, The race for 0 to 1 transition of D*

For the case in Fig.1.10 the race occurs in the N1-N4 loop between the signals in the nodes B and $\overline{R}$. Both nodes are initially high. If both D and Clock go high in the time window of two gate delays, the positive-loop feedback is enabled and the race begins. The race ends when either B or $\overline{R}$ goes low, depending on the relative position of Data and Clock transition within the window.

Similar situation occurs in the N2-N3 loop for the transition of D from 1 to 0 within the critical window. The difference is that the window is shifted one delay earlier because signals B and Clk actually enable the positive-feedback loop. The race is between the signals on node A and $\overline{S}$.

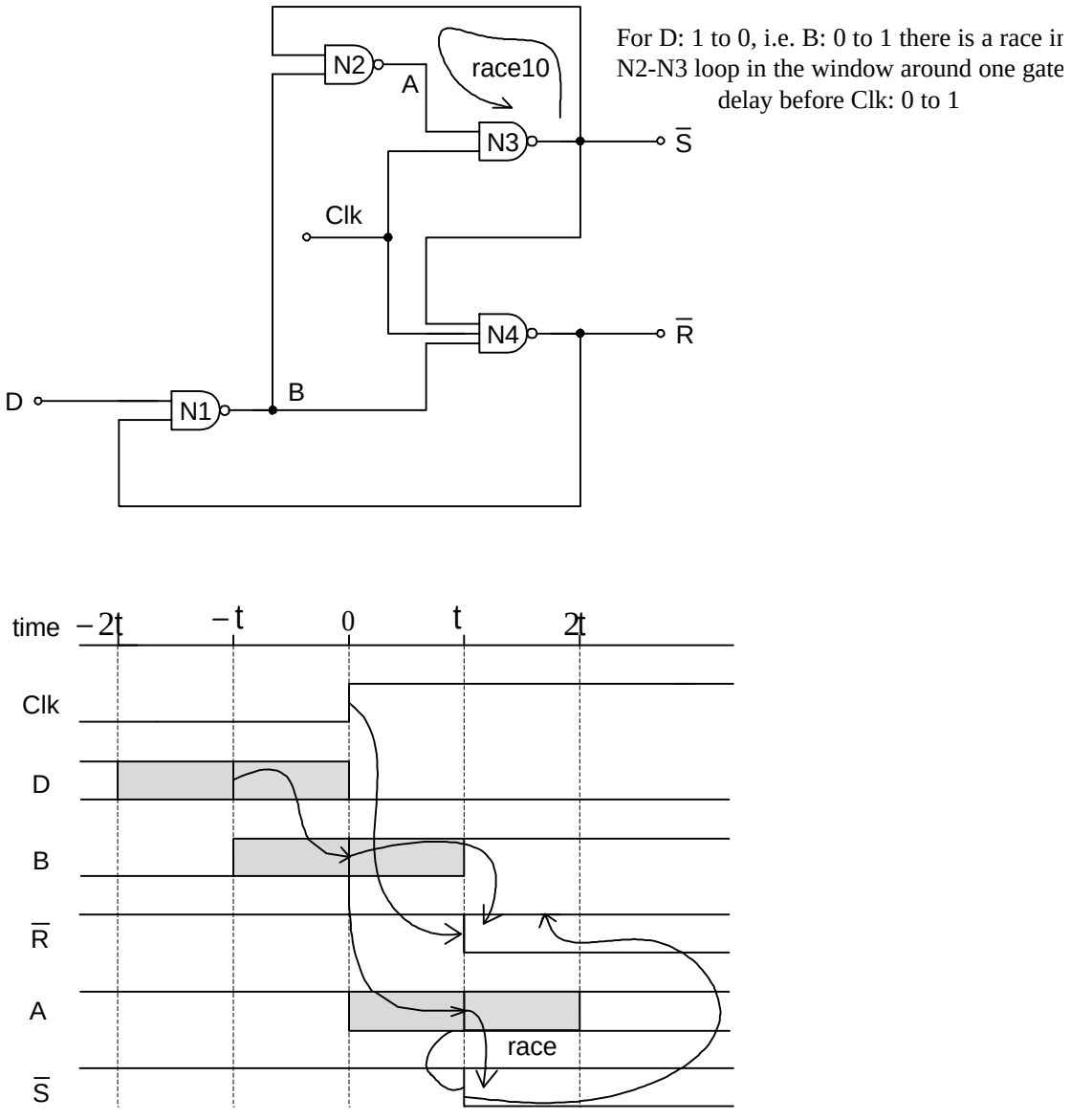


Fig. 1.11. *The race for the 1 to 0 transition of D*

1.2.2. Parameters of Bi-Stable Elements:

It is not possible to establish any requirements on the clock without the intimate knowledge of the bi-stable element used in the system. The clocking of a digital system and the choice and parameters of a bi-stable elements are very closely interrelated. The clocking strategy often implies and determines the choice of the bi-stable element used in the system and vice versa. For the start we should establish some basic parameters:

Parameters presented in this chapter define the conventionally accepted timing parameters of bi-stable elements, Fig.1.12. The inadequacies of this approach will be discussed later.

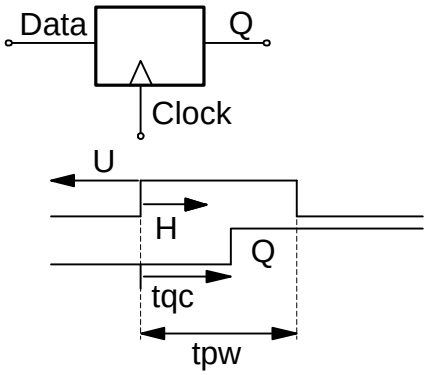


Fig. 1.12. *Parameters of a bi-stable element*

Setup time U: is defined as the minimum amount of time that the input signal D need to be stable before the latching mechanism takes place in order to assure reliable latching.

Hold time H: is defined as a minimal amount of time data D needs to remain stable after the latching mechanism has been occurred.

In addition we define delay of the bi-stable element which is added to the signal:

In a case of Flip-Flop this delay D_{CQ} is defined as a time difference between the time the triggering event C takes place (leading or trailing edge of the clock) and the time the value of the input D propagates to the output Q. (As stated by Unger and Tan in [3], we can make this concept more precise by requiring that the change in D occurs sufficiently early so that making it appear any earlier would have no effect on when Q changes).

In case of a Latch, there are two delays to be defined:

1) *delay D_{CQ}:* from the time clock C changes from 0 to 1, until the time the output Q reflects the value of the signal at the input D.
2) delay D_{DQ}: from the time the input D changes until the time the output Q reflects this change (while the clock C=1 the latch being in *"transparent"* mode)

The *latch delay* is usually taken as worse of the two: D_{CQ} or D_{DQ}, whichever happens to be greater.

1.3. Design of High-Performance Latch:

Availability of a Latch that can support requirements for high clock rates and a shallow pipeline is an essential factor in high-performance systems. A high-performance latch has to exhibit design flexibility, immunity to noise and immunity to race-through. Given that the latch delay is directly subtracted from the time available to perform useful computation in the clock period, the latch has to introduce a minimal amount of delay in the critical path. Various latch designs were developed to satisfy this last requirement. One of the significant advancements made was so called *"Earle's Latch"* developed by Earle in 1965 and Halin and Flynn in 1972, during the course of design of IBM 360/91 mainframe computer. The *"Earl's Latch"* is shown in Fig.1.13.
One can observe that *Earl's Latch* represents a logical AND-OR combination, which is a Sum-of-Products (SOP) form providing a general expression for representation of any logic function that can be implemented with this latch. The delay of *Earl's Latch* is two gates, which is one gate delay less compared to the commonly used cross-coupled gate combination.

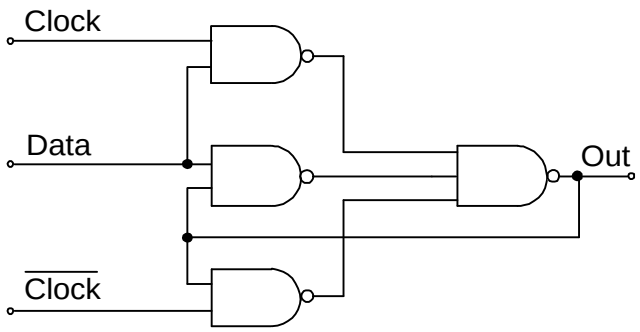


Fig. 1.13. *Earl's Latch*

The objective of minimizing latch delay has been the subject of work by Yuan and Svensson and later Afghahi and Svensson [7,8]. They developed very fast Latch and Flip-Flops better known as "True-Single-Phase-Clock" (TSPC) Latch(as shown in Fig. 1.14.). The same latch (with some improvements) has been used in the first implementation of Digital's "Alpha" processor achieving 200MHz clock rate [10], as shown in Fig.1.15. An interesting development of a Hybrid Latch Flip-Flop element is presented in [9]. Development and modifications of the clocking strategy including the selection and development of an appropriate latch structure is best illustrated in the later stages of Digital's "Alpha" processor development described in [11-13].

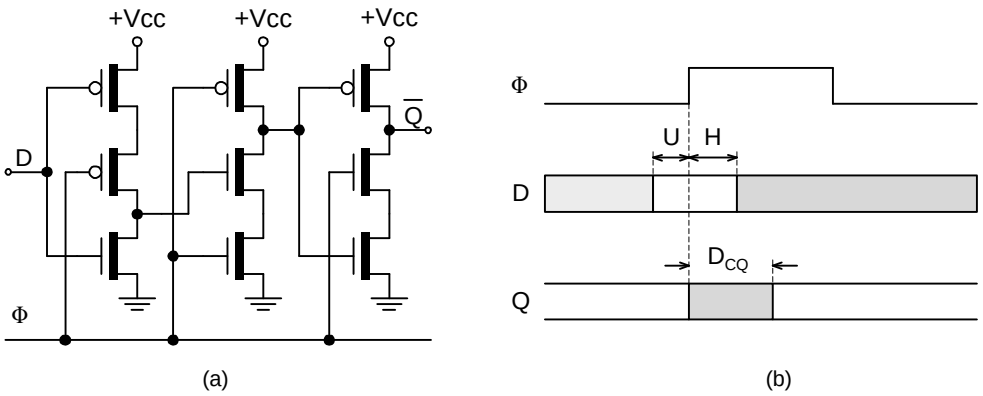


Fig. 1.14. *True-Single-Phase-Clock (TSPC) Flip-Flop: (a) schematic diagram (b) timing*

An attempt to reduce power dissipated by the clock sub-system is described in [5,6] in which the clock signal swing is reduced. In [6] a differential sense-amplifier structure is used for latching the signal. A Flip-Flop that was developed by Toshiba was later used in Digtal's 21264 "Alpha" processor which runs at 600MHz clock rate (shown in Fig. 1.16).

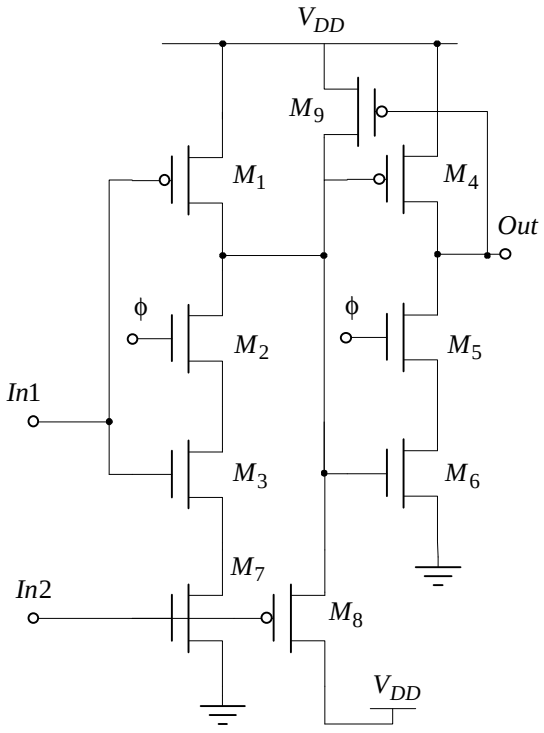


Fig. 1.15. *Modified TSPC Flip-Flop as used in the first generation 21064 "Alpha" processor from Digital [10].*

Finally the synthesis of the clock tree in order to reduce the clock skews is described in [4].

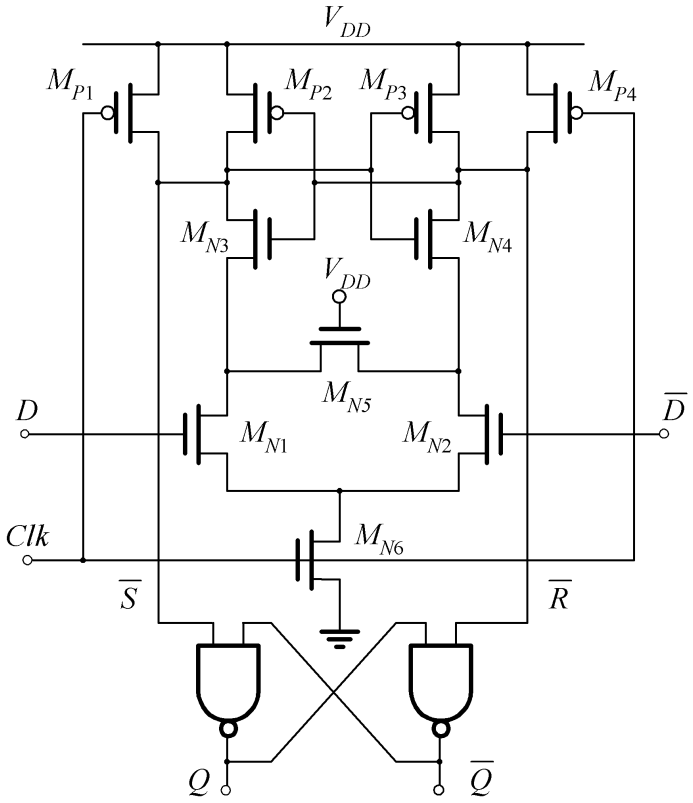


Fig. 1.16. *Toshiba Flip-Flop [6] used in the third generation 212064"Alpha" processor from Digital 21264. The Flip-Flop is differential.*

Further Reading:

1. Eby G. Friedman, *"Clock Distribution Networks in VLSI Circuits and Systems"*, IEEE Press, 1995.
2. Wagner, *"Clock System Design"*, IEEE Design & Test of Computers, October 1988.
3. S.H. Unger, C. Tan, *"Clocking Schemes for High-Speed Digital Systems"*, IEEE Transactions on Computers, Vol. C-35, No 10, October 1986.
4. Minami, M. Takano, *"Clock Tree Synthesis Based on RC Delay Balancing"*, Proceedings of IEEE Custom Integrated Circuits Conference, p. 28.3.1-28.3.4, May 1992.
5. Kojima, S. Tanaka, K. Sasaki, *"Half-Swing Clocking Scheme for 75% Power Saving in Clocking Circuitry"*, IEEE Journal of Solid-State Circuits, Vol. 30, No 4, April 1995.
6. H. Kawaguchi, T. Sakurai, *"A Reduced Clock-Swing Flip-Flop (RCSFF) for 63% Power Reduction"*, IEEE Journal of Solid-State Circuits, Vol. 33, No 5. May 1998.
7. M. Afghahi, C. Svensson, *"A Unified Single-Phase Clocking Scheme for VLSI Systems"*, IEEE Journal of Solid-State Circuits, Vol. 25, No 1. February 1990.
8. J. Yuan, C. Svensson, *"High-Speed CMOS Circuit Technique"*, IEEE Journal of Solid-State Circuits, Vol. 24, No1, February 1989.
9. H. Partovi et al, *"Flow-Through Latch and Edge-Triggered Flip-Flop Hybrid Elements"*, Proceedings of 1996 IEEE International Solid-State Circutis Conference, San Francisco, California February 1996.
10. D. Dobberpuhl et al, *"A 200MHz 64-b Dual-Issue CMOS Microprocessor"*, IEEE Journal of Solid-State Circuits, Vol. 27, No 11. November 1992.
11. B. J. Benschneider, et al, *"A 300-MHz 64-b Quad-Issue CMOS RISC Microprocessor"*, IEEE Journal of Solid-State Circuits, Vol. 30, No 11. November 1995.
12. B. Gieske, et al, *"A 600MHz Superscalar RISC Microprocessor with Out-of-Order Execution"*, *1997 ISSCC Dig. Tech. Papers*, p. 176-177, February 7, 1997.
13. P.E. Gronowski et al., *"High-Performance Microprocessor Design"*, IEEE Journal of Solid-State Circuits, Vol. 33, No 5. May 1998.