Unit 5

Synthesis for Low Power

Low Power Clock Distribution

Power Dissipation in Clock Distribution:

The clock frequencies of CMOS digital systems are approaching gigahertz range as a result of the submicron technology. At the same time, power dissipation is becoming a limiting factor in integrating more transistors on a single chip and achieving the desired performance.

Layout or physical design has traditionally been a key stage in optimizing the design to achieve performance and cost goals. As the demand for minimizing power increases, low-power driven physical design has attracted more and more attention.

Generally, low power considerations were given in the floor-plan stage of chip layout while placement and routing techniques for minimizing power consumption.

In this section, physical design of clock distribution is considered.

The synchronization of a digital system requires one or more reference signals to coordinate and insure correct sequences of operations. Fully synchronous operation with a common clock has been the dominant design approach for digital systems. A clock tree is usually constructed to globally distribute the clock signal to all modules throughout the system. The transitions of clock signal provide the reference time for each module to latch in data, trigger operations and transmit outputs. Clock has been recognized as the most important signal in determining system performance and total power consumption.

The amount of power dissipated by a CMOS circuit consists mainly of two parts:

Dynamic power: due to charging and discharging capacitive loads at every clock cycle.

Short-circuit power: due to the short-circuit current through PMOS and NMOS transistors during the switching interval.

Clock is a major source of dynamic power dissipation in a digital system. For example, the DEC Alpha chip uses a clock driver which drives 3250 pF capacitive load. When operating at 200 MHz, with a 3.3 V supply voltage, the dynamic power dissipation alone on the clock is 7.08W, which amounts to 30% of the chip's total power dissipation.

The dynamic power dissipated by switching the clock can be given by:

$$P_{clk} = f V_{dd}^{2} (C_L + C_d)$$

where *CL is the total load on the clock* and *Cd is the clock driver capacitance.*

Given the total number of clock terminals *N*, *the nominal input capacitance at* each terminal, cg , the unit length wire capacitance, cw and the chip dimension, *D*, assuming an H-tree based global clock routing of *h levels* , *CL can be given by:*

$$C_L = N c_g + 1.5 (2^h - 1) D c_w + \alpha \sqrt{N 4^h} c_w$$

where the second and third terms are the global and local wiring capacitance respectively, α is an estimation factor depending on the algorithm used for local clock routing

From the above equations it is clear that the dynamic power dissipated by clock increases as the number of clocked devices and the chip dimensions increase. A global clock may account for up to 40% of the total system power dissipation .

For low power clock distribution, measures have to be taken to reduce the clock terminal load, the routing capacitance and the driver capacitance.

Clock skews are the variations of delays from clock source to clock terminals. To achieve the desired performance, clock skews have to be controlled within very small or tolerable values . Clock phase delay, the longest delay from source to sinks, also has to be controlled in order to maximize system throughput .

WKT amount of capacitive load carried by the clock was largely depends on loading capacitance of clock terminals.

However, as technology advances, device sizes are shrinking rapidly which reduces the clock terminal capacitances.

Also high frequency requirement, results performance driven clock tree construction methods such as adjusting wire lengths or widths to reduce clock skew increase the interconnect capacitance to a more dominant part of the total load capacitance on clock.

Reducing the interconnect capacitance may significantly reduce the overall system power consumption.

The low power systems with reduced supply voltages require increasing the device sizes(width of tr) to maintain the necessary speed. This results both the dynamic and short-circuit power dissipated by clock. Therefore, clock distribution is a multi-objective design problem.

Minimizing clock power consumption has to be considered together with meeting the constraints of clock skew and phase delay.

In the following sections, we address some of the issues and propose some solutions in low power clock distribution.

Single Driver v/s Distributed Buffers

Clock Driving Schemes

To ensure fast clock transitions, buffers have to be used to drive the large load capacitance on a clock. There are two common clock driving schemes:

In the *single driver scheme* *as shown in* *Figure a*, *a* *chain of cascaded* buffers with a very large second buffer is used at the clock source, no buffers are used elsewhere;

In the distributed buffers scheme *as shown in Figure b*, intermediate buffers are inserted in various parts of the clock tree. At each insertion location, one or more buffers are cascaded.

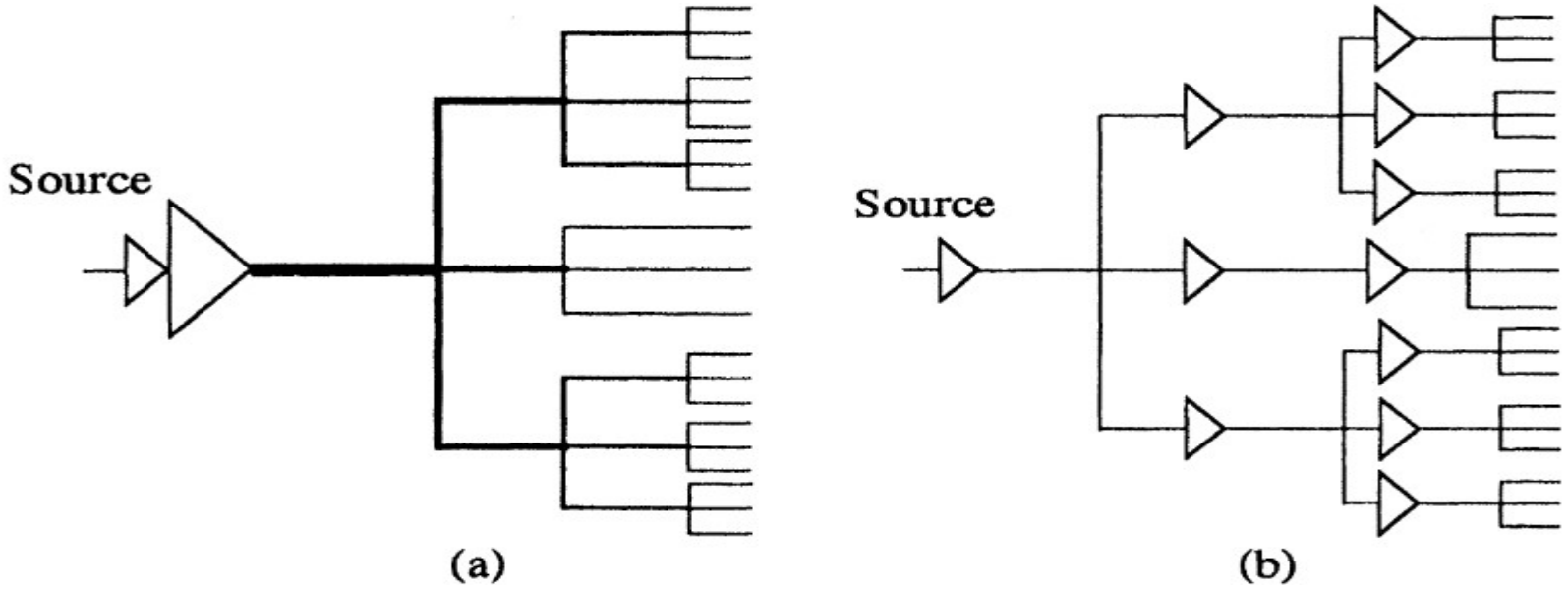
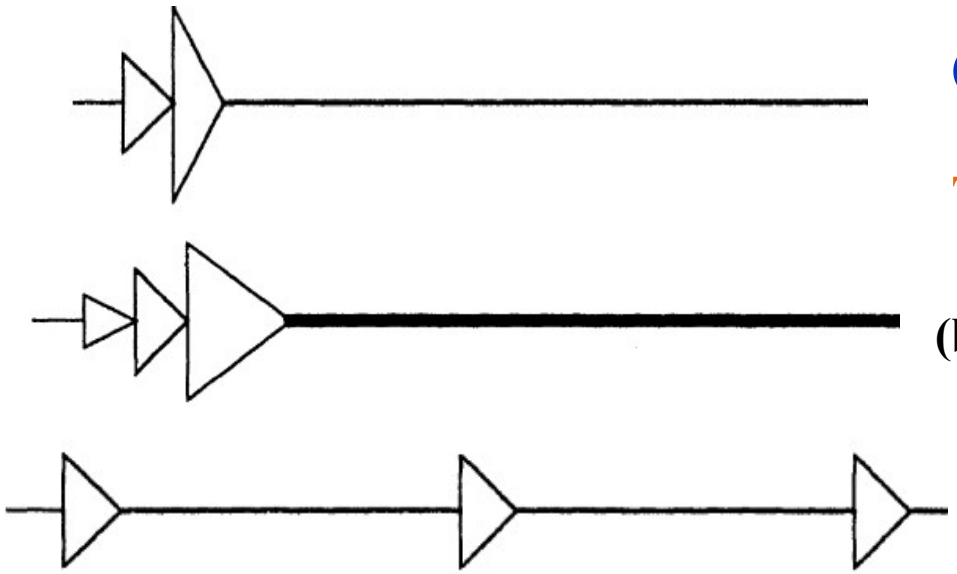


Figure: Two clock tree driving schemes: (a) Single driver scheme where drivers are at clock source and no buffers else where; (b) Distributed buffers scheme where intermediate buffers are distributed in the clock tree.

The single driver scheme *has the advantage of avoiding the adjustment of* intermediate buffer delays. Here the wire sizing is used to reduce the clock phase delay. Widening the branches that are close to the clock source can also reduce skew caused by asymmetric clock tree loads and wire width deviations.

The distributed buffers scheme *has been recognized to have the advantage* that relatively small buffers are used and they can be flexibly placed across the chip to save layout area. Also, for a large clock tree with long path-length, the intermediate buffers (or repeaters) can be used to reduce clock phase delay.



(a) A long clock line can be considered as a distributed RC line.

Two ways to reduce delay of the long clock line:

(b) by wire widening;

(c) by intermediate buffers insertion.

Figure illustrates the effects of wire widening and intermediate buffer insertion on delay reduction. In high speed design, a long clock path as shown in Figure a can be treated as a distributed RC delay line. Widening the line as shown in Figure b will make it a capacitive line with smaller line resistance. By adjusting the sizes of buffers at the source, the line delay will be reduced. The other way to reduce line delay is to insert intermediate buffers along the line as shown in Figure c.

The intermediate buffers partition the line into short segments each of which has small line resistance. This makes the delay of the line more linear with the length. While widening wires requires increasing the sizes of buffers at the source and hence the short circuit power dissipation, the small intermediate buffers used to drive the short wire segments impose little penalty on power dissipation.

For power minimization in a clock tree, the distributed buffers scheme *is* preferred over the single driver scheme.

We consider a simple example of an equal path length tree and its delay model as shown in below Figure, where *l1=l2* .

The lengths and widths of the tree branches are *l0, l1, l2 and w0, w1, w2. The load* capacitances at sinks, sl and *s2 are Cl1* and *Cl2* respectively, The skew in between *can be derived as:*

*(*The spatial variation in arrival time of a clock transition on an integrated circuit is commonly referred to as *clock skew)*

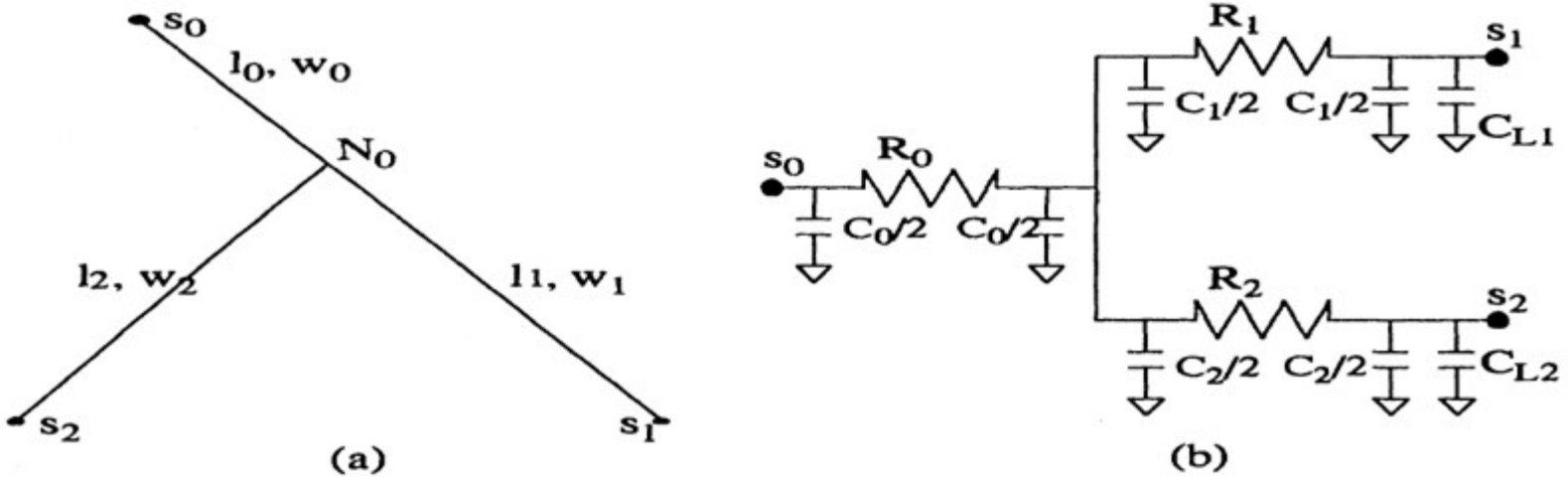


Figure (a) An equal path-length clock tree; (b) The delay model.

$$t_s = \frac{rl_1}{w_1}C_{L1} - \frac{rl_2}{w_2}C_{L2}$$

where *r is the sheet resistance of the wire*.

***The skew variation in terms of wire* width variations can be stated as:**

$$\Delta t_s = \frac{\partial t_1}{\partial w_1}\Delta w_1 + \frac{\partial t_2}{\partial w_2}\Delta w_2 = -\frac{rl_1C_{L1}}{w_1^2}\Delta w_1 + \frac{rl_2C_{L2}}{w_2^2}\Delta w_2$$

Assuming the maximum width variations *;$\Delta w = \pm 0.15w$, the worst case*

additional skew is:

$$\Delta t_s = 0.15\left(\frac{rl_1C_{L1}}{w_1} + \frac{rl_2C_{L2}}{w_2}\right)$$

Above equations indicate that without wire width variations, skew is a linear function of path length. However, with wire width variations, the additional skew is a function of the product of path length and total load capacitance.

Increasing the wire widths will reduce skew but result in larger capacitance and power dissipation. Reducing both the path length and load capacitance also reduces skew and minimum wire width can be used such that wiring capacitance is kept at minimum. Therefore, if buffers are inserted to partition a large clock tree into sub-trees with sufficiently short path-length and small loads, the skew caused by asymmetric loads and wire width variations becomes very small.

Additionally, if the chip is partitioned in such a way that clock distributed to subsystems can be disabled (or powered down) according to the subsystem functions, significant amount of power can be saved. This can be done by replacing the buffers with logic gates.

Clock buffers dissipate short-circuit power during clock transitions. The intermediate buffers also add gate and drain capacitance to the clock tree. But compared to the *single driver scheme which uses large buffers at the clock source,* intermediate buffers does not introduce more short-circuit power dissipation if buffers are not excessively inserted and overly sized.

Buffer Insertion in Clock Tree

Different buffer delays cause phase delay variations on different source-to-sink paths.

The tolerable skew of a buffered clock tree *ts* *is divided into two components:* *the* *tolerable skew for buffer delays,* t_s^b *and* the skew allowed for asymmetric loads and wire width deviations after buffer insertion t_s^w,

$$t_s = t_s^b + t_s^w$$

To balance the source-to-sink delays in a clock tree, we start with an equal path-length clock tree. An equal path-length tree, *T consists of a source* S0 and a set of sinks S = {s1, S2, . . , S*m* }. *The buffer insertion problem is to find the* locations on the clock tree to insert intermediate buffers. We call these locations *Buffer Insertion Points(BIPs).*

To meet the skew constraint, the buffer insertion scheme should try to balance the buffer delays on source-to-sink paths independent of the clock tree topology.

The resulting clock tree has the following properties:

Each source-to-sink path has the same number of buffers(levels); all sub-trees rooted at a given level are equal path-length trees.

We select the cut-lines so as to form *iso-radius levels. An iso-radius level of the clock* tree is a circle centered at the clock source such that all the paths from the source to the BIPs are of the same length.

For a clock tree with path length of *L, we* choose the radius r of the first level cut-line (nearest to the clock source) as r = *L/(Φ+1) where Φ is the designated number of levels of buffers. The radius is r for* the first iso-radius level, 2r for the second iso-radius level, and so on... ,

Φr for the *Φ*th iso-radius levels. We determine the minimum number of buffer levels *Φ* that satisfies the skew constraint by iteratively evaluating the worst case skew of the clock tree with *Φ* = 1,2, ... until we find some number of levels *Φ*. *It is* paradigm is depicted in Figure

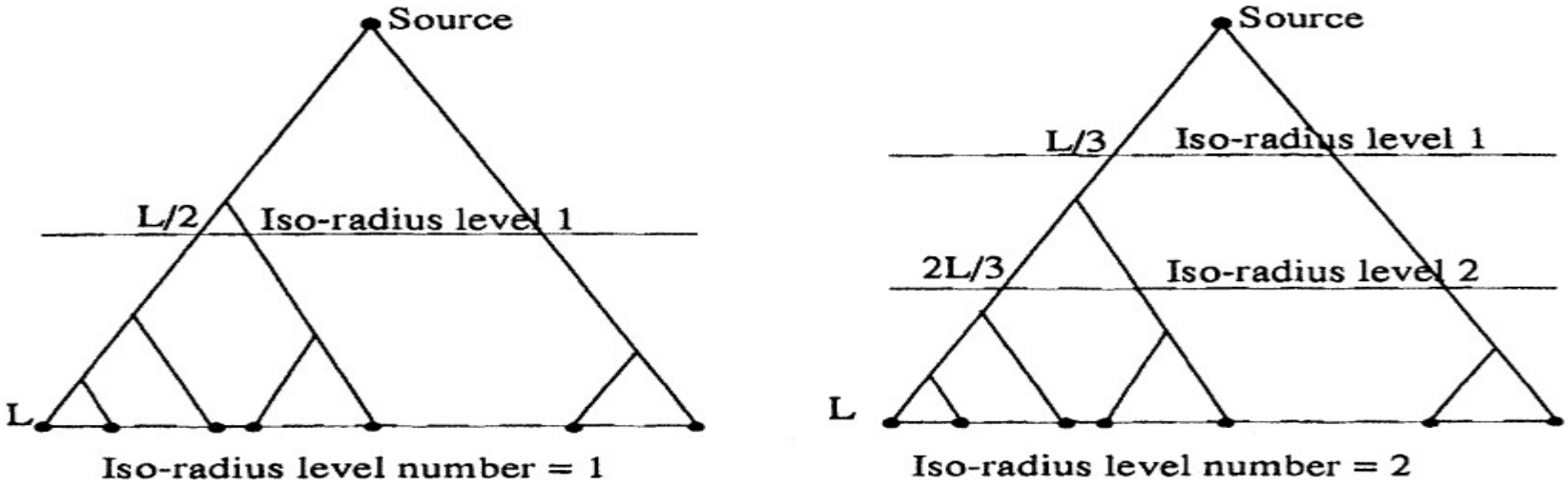


Figure : Buffer levels are increased until the tolerable skew bound is satisfied.

An example of the buffer insertion scheme is shown in Figure (a).

The conventional level-by-level at the branch split points of the clock tree is as shown in Figure (b). This works well in a full binary tree where all sinks have the same number of levels.

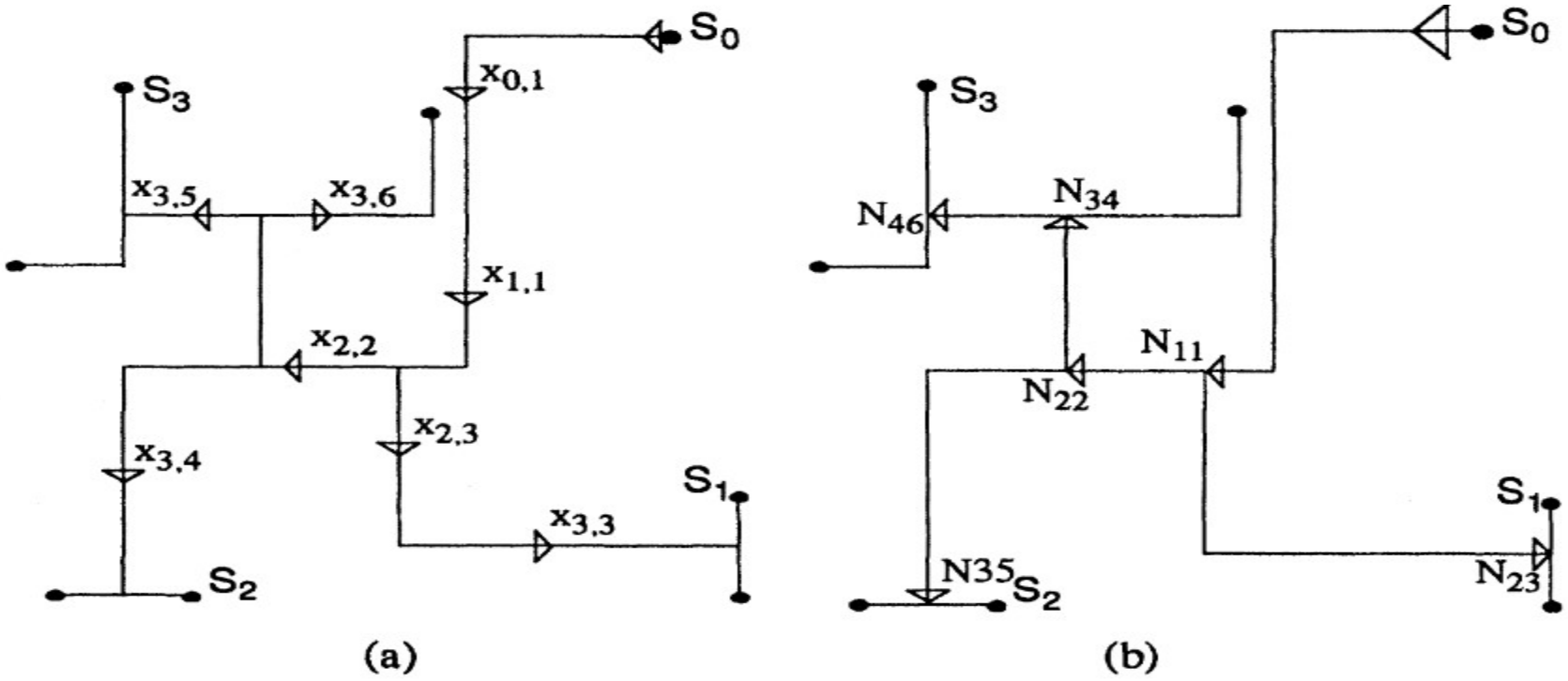


Figure : An example of buffer insertion in an equal path-length clock tree:
(a) using the balanced buffer insertion method;
(b) using the level-by-level method.

In the case of a general **equal path-length** tree, such as the case in Figure , different numbers of buffers are inserted on different source to sink paths. **Depending on the clock tree topology, some large sub-trees may still require wire widening to reduce skew.**

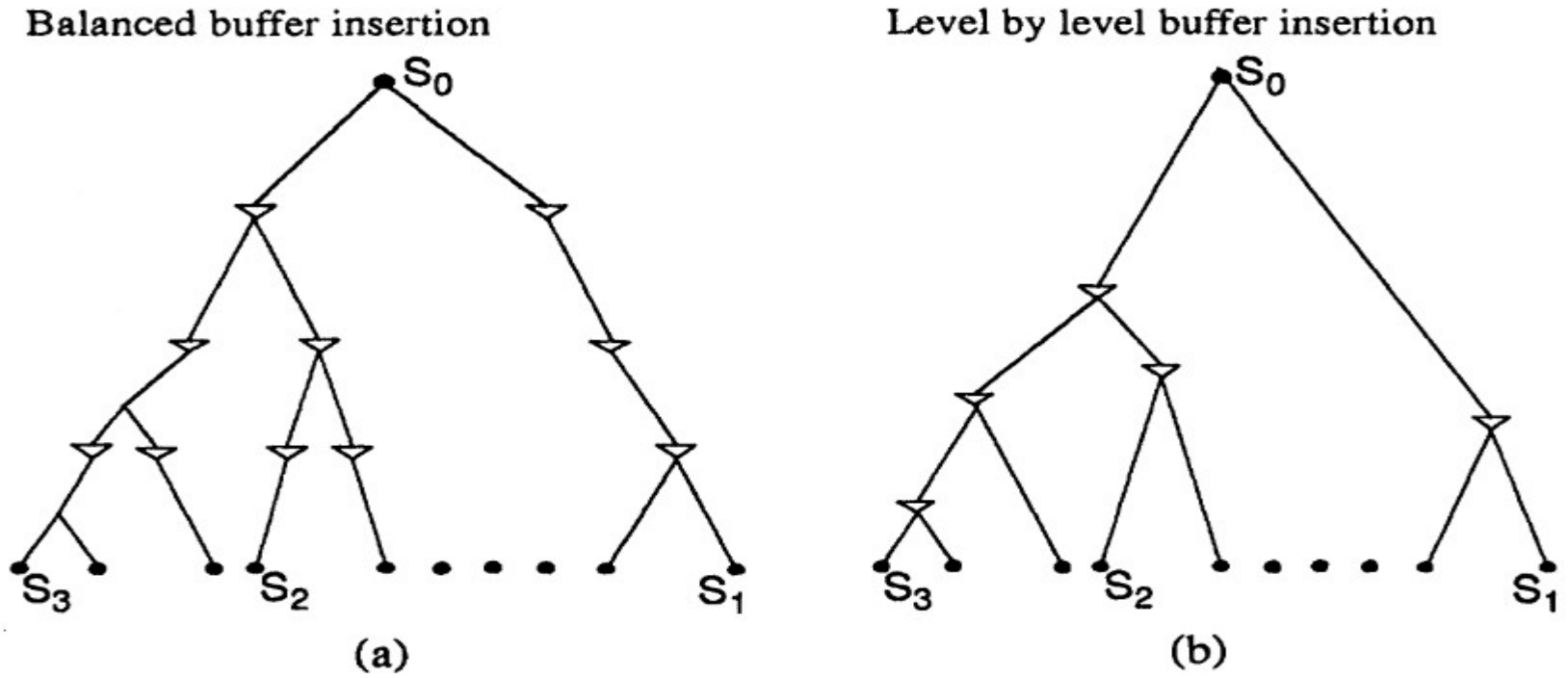


Figure : Comparison of buffer insertion in a general equal path-length clock tree using: (a) the balanced buffer insertion method; (b) the level-by-level method.

Zero Skew vs. Tolerable Skew

Most techniques used for skew minimization are based on adjusting the interconnect lengths and widths: the *length adjustment technique moves the balance* points or elongates the interconnect length to achieve zero skew the *width sizing technique achieves zero skew by adjusting the widths of wires in* the clock tree .

Due to the attempt to achieve the minimum skew for all clock sinks, wire lengths or widths of the clock tree may be increased substantially resulting in increased power dissipation.

Here, we discuss an alternative approach by taking advantage of tolerable skews. Tolerable skews are the maximum values of clock skew between each pair of clock sinks with which the system can function correctly at the desired frequency.

To illustrate the concept of tolerable skew, the operation of a synchronous digital system is shown in figure. The Figure shows the sequential operation of the registers with one single-phase clock, assuming edge-triggered flip-flops are used.

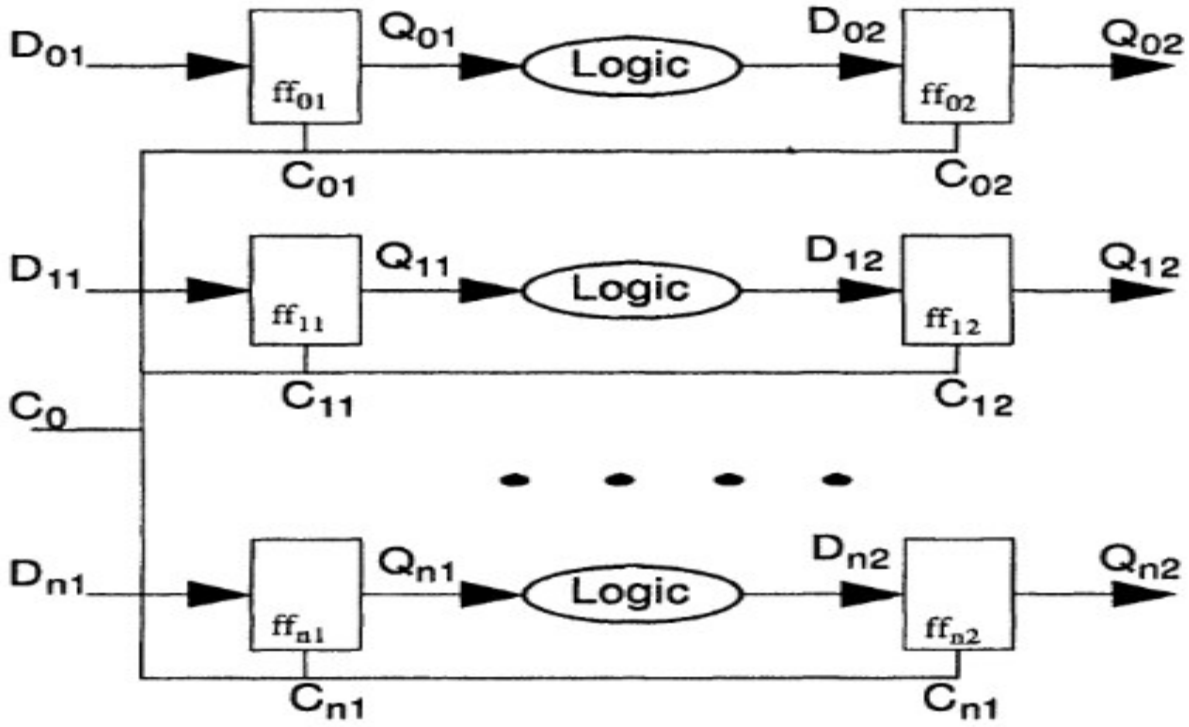


Figure : The basic building block in a paralle/pipelined synchronous system.

The flip-flops are characterized with setup time, d_{Setup}, hold time, d_{Hold} and flip-flop delay d_{FF}. The signal D_{01} is the input to the flip-flop ff_{01} and D_{02} the input of flip-flop ff_{02}, which is the output of the combinational logic. There are some phase delays due to interconnection delays between the clock source C_0 and C_{01} and C_{02} which are clock terminals of ff_{01} and ff_{02} respectively. Let d_{01} and d_{02} denote the arrival time of C_{01} and C_{02}. The combinational logic block is characterized with maximum delay $MAX(d_{logic})$ and minimum delay, $MIN(d_{logic})$, due to the variations of input values and input to output path delays through the combinational logic.

Figure 5.9 illustrates two cases of correct synchronous operations with tolerable skews. In Figure 5.9(a), the clock arrives at C_{02} later than the previous stage clock terminal C_{01} resulting in a negative clock skew. In Figure 5.9(b), the clock arrives at C_{02} earlier than C_{01} causing a positive clock skew. In both cases, the synchronous operation is correct since the correct data is latched in the next-stage flip-flop, ff_{02}. Thus in both cases the skews are tolerable.

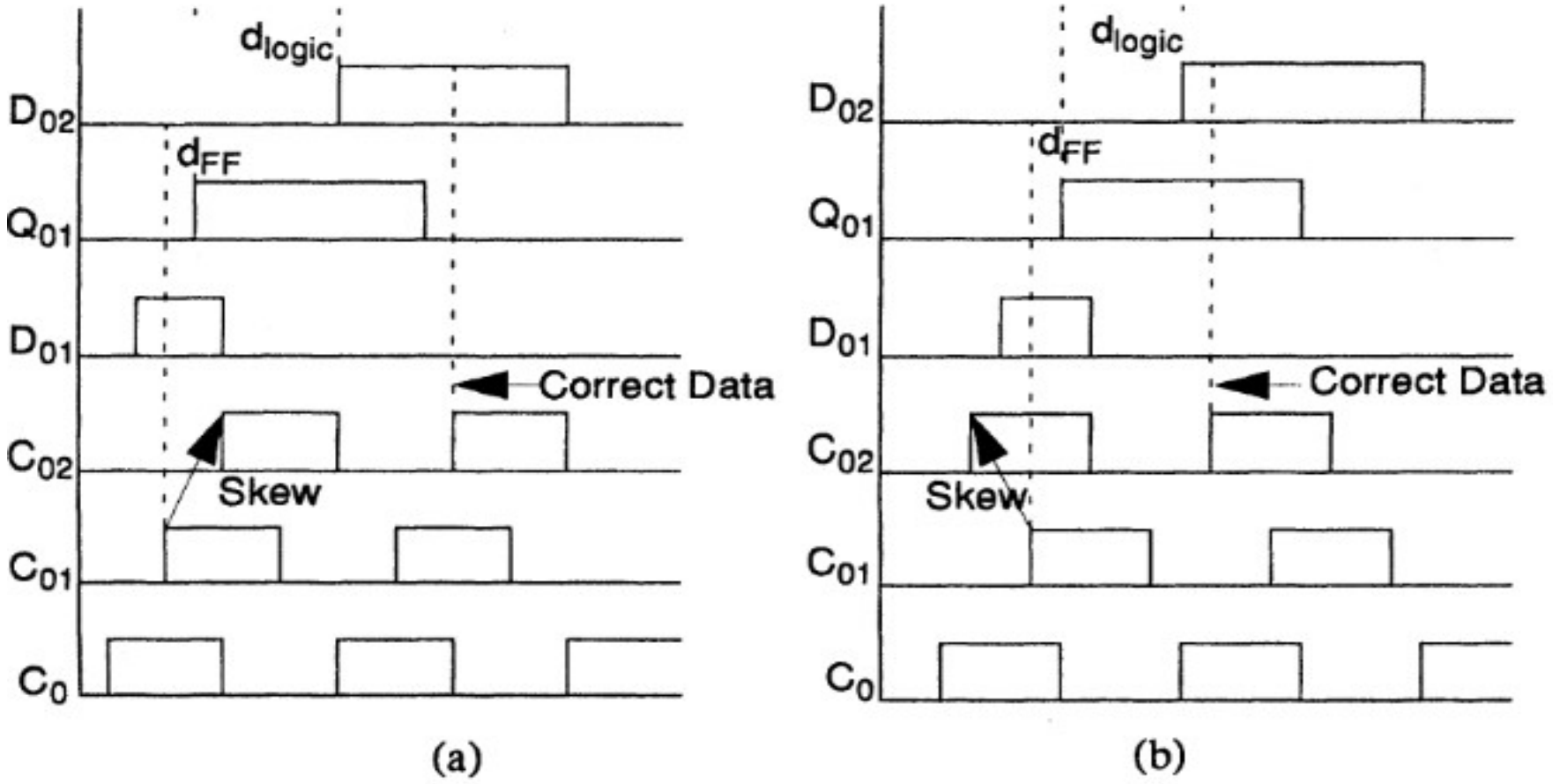


Figure : Correct synchronous operations with tolerable skews: (a) with negative clock skew; (b) with positive tolerable skew.

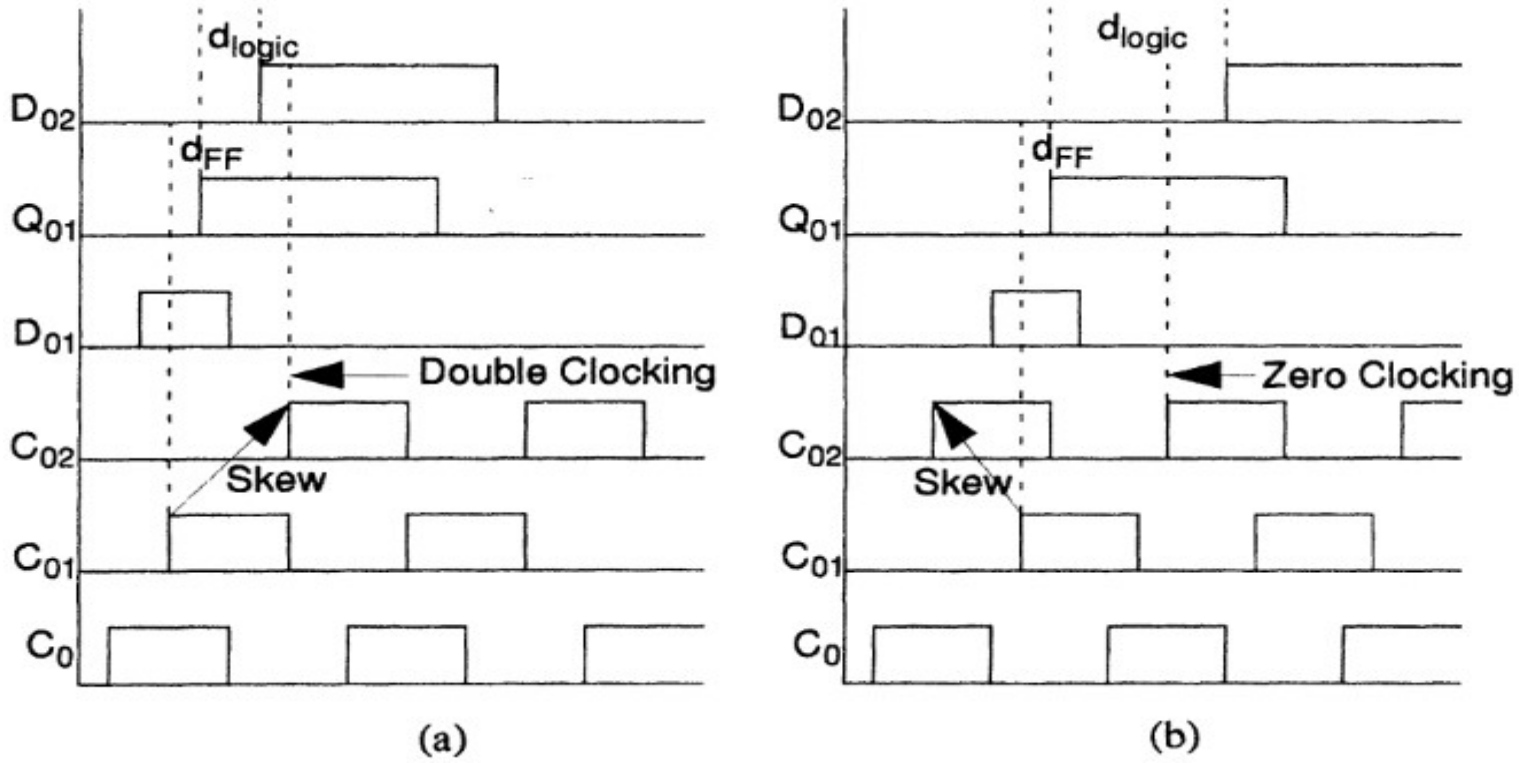


Figure : Incorrect synchronous operations with excessive clock skews:
(a) Double-clocking with negative skew;
(b) Zero-clocking with positive skew.

However, with excessive clock skew, incorrect operations may occur as shown in Figure .

However, with excessive clock skew, incorrect operations may occur as shown in Figure 5.10. In the case shown in Figure 5.10(a), a phenomenon called *double-clocking* is caused by the late arrival of the clock signal at C_{02}. At the first clock transition, data is latched in by the first flip-flop, ff_{01}. The second stage flip-flop, ff_{02} latches in the data produced by the combinational logic at the same clock transition while missing the data at the next clock transition. In the case shown in Figure 5.10(b), a phenomenon called *zero-clocking* occurs when the clock arrives at C_{02} too early and the second stage flip-flop, ff_{02} is unable to latch in the data produced by the combinational logic at the next clock transition. To avoid these clock hazards under a given clock cycle time, P, the clock timing has to satisfy the following constraints [8],[15].

To avoid *double-clocking:* $$d_{01} + d_{FF} + MIN(d_{logic}) \geq d_{02} + d_{Hold}$$

To avoid *zero-clocking:* $$d_{01} + d_{FF} + MAX(d_{logic}) + d_{Setup} \leq d_{02} + P$$

d01 and d02 indicates clock arrival time of C01 and c02, P➔Clock cycle time