MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

Computation Structures Group Memo 91

Synchronizers and Arbiters

by

Suhas S. Patil

October 1973

Introduction

The problem of <u>synchronization</u> arises when independently operating synchronous units communicate with each other. Because of the possibility that the signals received at a unit may not be in proper synchronization with the internal clock of that unit, the signals must be synchronized before they are used in the internal logic of that unit. This task is performed by a device called synchronizer that delays the changes in signal levels in such a way that the changes take place at the time instances defined by the clock.

Synchronization is a problem because no way is known for realizing a synchronizer which works correctly every time; under some critical conditions such as when the incoming signal changes at the same time as the clock pulse, the synchronizer may fail to synchronize the signal [4, 15]. It is doubtful if a perfect synchronizer can be physically realized.

The problem of <u>arbitration</u> which is to properly control the access of a shared resource unit by several users is closely related to the synchronization problem. The source of difficulty with physical realization of both the arbiters and the synchronizers is the same: a phenomenon called "glitch" which every bistable element such as a flip-flop [2,3,4,6,8,10,16] experiences. The glitch arises out of an unwanted unstable equilibrium state which is always present in addition to the two stable states. Under certain critical conditions, this unstable equilibrium state, called a <u>meta-stable</u> <u>state</u>, leads to abnormal delays which can cause a synchronizer or an arbiter to malfunction.

Our first objective in this paper is to explain the problem of synchronization and arbitration and to show the relationship between them. The requirements of arbitration for synchronous and asynchronous systems are not exactly the same; the differences are crucial to the extent that even though it seems impossible to physically realize the arbiter needed for synchronous sytems, the type of arbiter needed

for asynchronous systems is easily realized with electronic gates. A new circuit for a perfect asynchronous arbiter which is perhaps the smallest circuit for such a device is presented in this paper.

The second objective of the paper is to explore new structures for the arbiters and synchronizers. We present an iterative structure for arbiters based on a new concept for realizing arbiters. The iterative structure can provide arbiters and synchronizers of arbitrarily low probability failure. The price paid of the reduced probability of failure is an increase in the delay of operation. One important property of the iterative structure is that when two arbiters or synchronizers are connected in tandem, the probability of failure for the new device is equal to the product of the probabilities of failure of individual devices. We examine the conditions under which multi-stage structures for synchronizers and arbiters are better than the conventional two-stage structures.

We suggest a set of parameters for comparing a flip-flop against another with regard to their performance in synchronizers and arbiters, and show how the performance of the multi-stage synchronzers and arbiters can be derived from these parameters of the quality of a flip-flop.

Synchronizers and Arbiters

The schematic diagram of a synchronizer is shown in Figure 1. A change in the level at the output of the synchronizer corresponds to a change in the level at the input, but the change at the output occurs only at the next clock pulse following the level change in the input signal. If the input changes at the same time as the clock pulse, the output may change either at that very clock pulse or at the next clock pulse, but not in between the clock pulses. The phrase "the output changes at a clock pulse" will mean that the output changes within a fixed time following the initiation (or the termination) of the clock pulse. To ensure that the output does not miss out on any input changes, the successive changes in the input must be separated by more than the time period of the clock with which the signal is being synchronized.

Consider the circuit shown in Figure 2. This circuit for the synchronizer works correctly except for the situation when the input signal changes at nearly the same time as the clock pulse. In this situation an arbitrary amount of additional delay up to a clock period may be encountered in the output change because of the problem arising out of the 'glitch' in the operation of the flip-flop [2,3,4,7,10].

Before we take up the study of the glitch phenomenon and its effect on synchronizers and arbiters, we shall describe an elementary arbiter and show the relationship between synchronizers and arbiters.

An arbiter is a two-input two-output device whose schematic diagram is shown in Figure 3 [1,7,11,13]. The dotted line connecting an input and the corresponding output are drawn in this particular illustration to emphasize that except for whatever delay that might be introduced by the arbiter as necessary, the signal levels at the outputs correspond to the levels at the associated inputs. If we begin from the initial condition, when the levels at all input and output are 0, and change one input level from 0 to 1, the corresponding output will change from 0 to 1, and
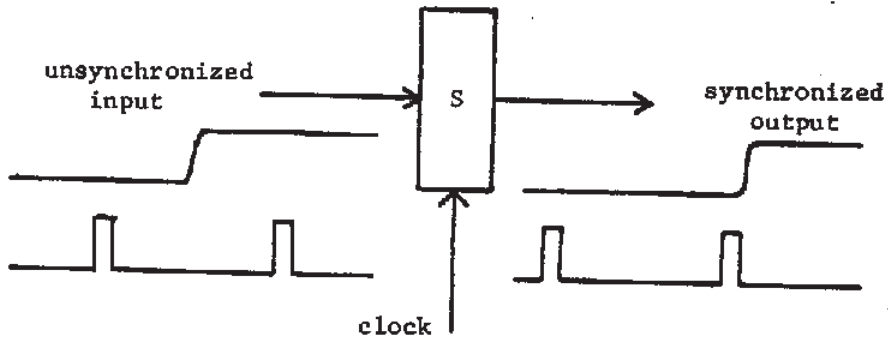
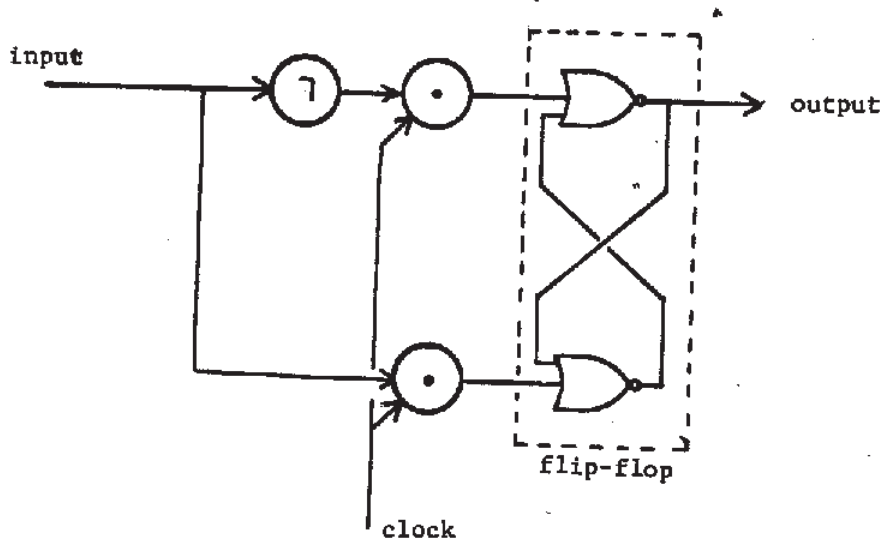Figure 1. A schematic diagram of a synchronizer.

Figure 2. A circuit for an (imperfect) synchronizer.

the arbiter will be said to be engaged. If the other input level is also now changed to 1, the corresponding output will not change to 1 so long as the arbiter remains engaged to the first input. When the input that has engaged the arbiter returns to 0, the corresponding output will change to 0 and the arbiter will be set free to be engaged by the input that might be waiting to get through the arbiter. Thus the function of the arbiter is merely to block some 0 to 1 changes in the input from reaching the associated output as long as is necessary to prevent both outputs from being at the level 1 at the same time. The critical operation of the arbiter arises when both inputs change to level 1 at nearly the same time. In this case the arbiter must choose which input should get through and which should get blocked; it is not important what that choice is, but it is important that the choice be made -- a task which seems impossible to always do in a fixed period of time.

We will identify two kinds of arbiters based on the needs of the synchronous systems and the asynchronous systems. The arbiter needed in the synchronous systems must operate within a fixed length of time, that is, if the arbiter is not engaged and any one or both inputs changes to 1, the arbiter must get engaged to one of the inputs and respond by changing the corresponding output to 1 within a fixed length of time. The arbiter is also required to respond within a fixed time when it is set free by changing the input to 0. An arbiter which meets these requirements will be called an <u>arbiter with a time bound</u>, or simply a <u>synchronous arbiter</u>. The asynchronous arbiter is not required to act within a time bound: an asynchronous arbiter which resolves conflicts quicker will be considered better but because of the asynchronous nature of the environment no bound on the time is necessary.

An illustration of the use of an arbiter in the sharing of a function unit is shown in Figure 4. The figure shows a function unit f which can be connected
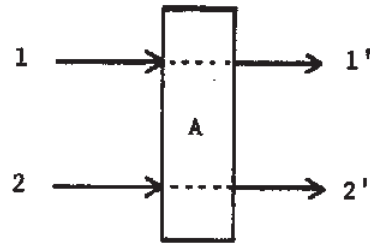
Figure 3. An arbiter.

either to the input registers U and V and the output W or to the input registers
X and Y and the output Z. The connection points, u, v and w provide the physical
means for connecting the function unit in the first arrangement and the connec-
tion points x, y and z for connecting the function unit in the second arrangement.
For correct operation the function unit may be connected in either arrangement but
not both at any one time. Let us say that the first arrangement, where the func-
tion unit is connected to U, V and W, corresponds to the use of the function
unit by user 1, and the alternate arrangement corresponds to user 2. To prevent
both users from using the function unit at the same time we introduce an arbiter
in the circuit as shown. Essentially this arrangement represents an asynchronous
system. To capture the function unit a user raises the level of its ready wire
to 1. If the arbiter is free, the arbiter is engaged to that user and the 0 to 1
change on the ready input passes through the arbiter. The level of 1 at the output
connects the function unit to the appropriate registers by activating the necessary
connection points. The 0 to 1 change in the output of the arbiter reaches the
user on the acknowledge wire after a fixed delay which, let us say, is equal to
the time the function unit needs to complete its operation. Upon receiving the
acknowledge signal, the user uses the results and brings the level of the ready
signal back to 0 to release the function unit. The arbiter then brings the level
of the corresponding output to 0 and becomes 'free' to be engaged to the other
user. If the other user is waiting, the level of the associated ready wire will
be 1, and the arbiter will become engaged to that user. Basically, the function
of the arbiter is to resolve the conflicts that may arise in the use of the shared
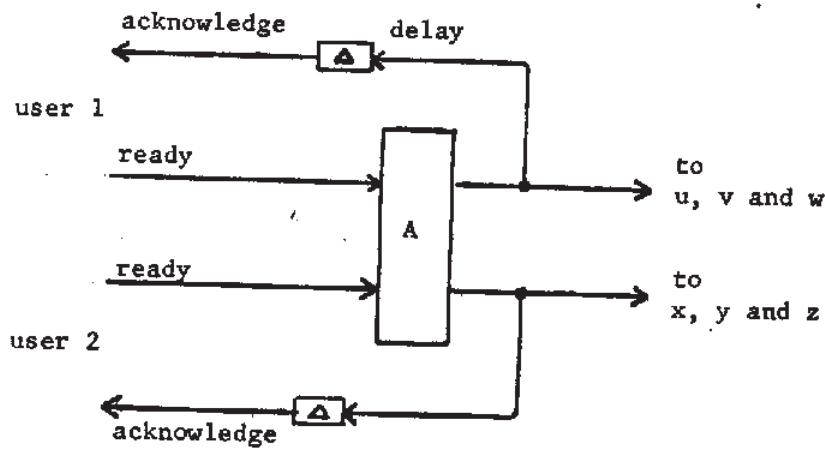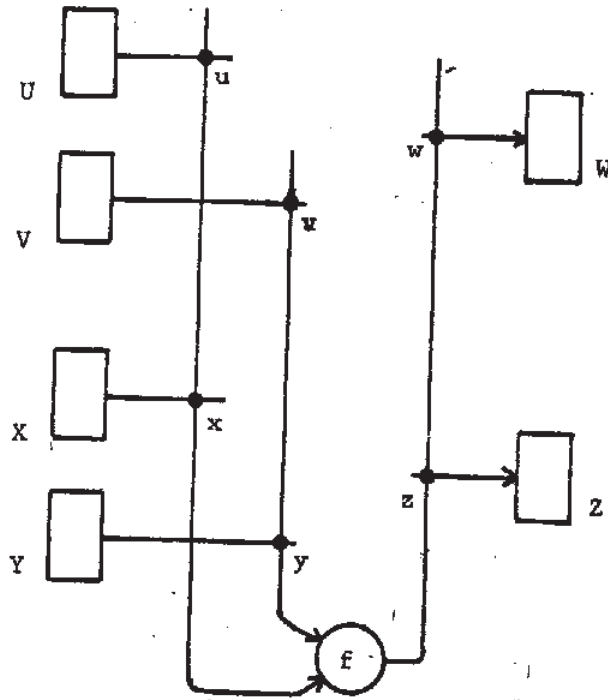function unit by the two users so as to maintain a consistent behavior.

Figure 4. Resource sharing with arbiter.

Relationship Between Synchronizers and Arbiters

In this section we show how an arbiter with a bounded delay can be realized from a synchronizer and the other way around. We also discuss single edge and double edge synchronizers. First we take up the realization of an arbiter with a bounded delay from a synchronizer.

Figure 5 shows how a two input arbiter with bounded delay can be realized with synchronizers and standard circuit elements such as flip-flops, AND gates and NOT gates. The first part of the circuit consists of synchronizers which synchronize both inputs to a clock so that the second part which is made of synchronous logic, correctly realizes the necessary logic of the arbiter. If the outputs of both synchronizers switch to 1 at the same clock pulses (as can happen when the inputs change at nearly the same time), the selection logic selects user 2 and blocks the input from user 1. The blocking circuitry is necessary to keep the other input blocked while the arbiter is engaged to one input. The circuit works correctly because we have assumed that synchronizers are perfect and that the gates have bounded delays.

We shall next examine how a synchronizer can be realized from the arbiters with bounded delays. This will be done by realizing a single edge synchronizer that synchronizes only the 0 to 1 transitions (the positive edges) and permits the 1 to 0 transitions (negative edges) to go through and reset the synchronizer into its initial condition without any obstruction. The double edge synchronizer, the synchronizer which synchronizes both the positive and negative edges can be realized by an arrangement shown in Figure 6 where the problem of synchronization is broken up into the problem of synchronizing the positive going edge and the problem of synchronizing the negative going edge, these tasks being performed by two separate parts which are combined at a C-element (Muller's C-element).
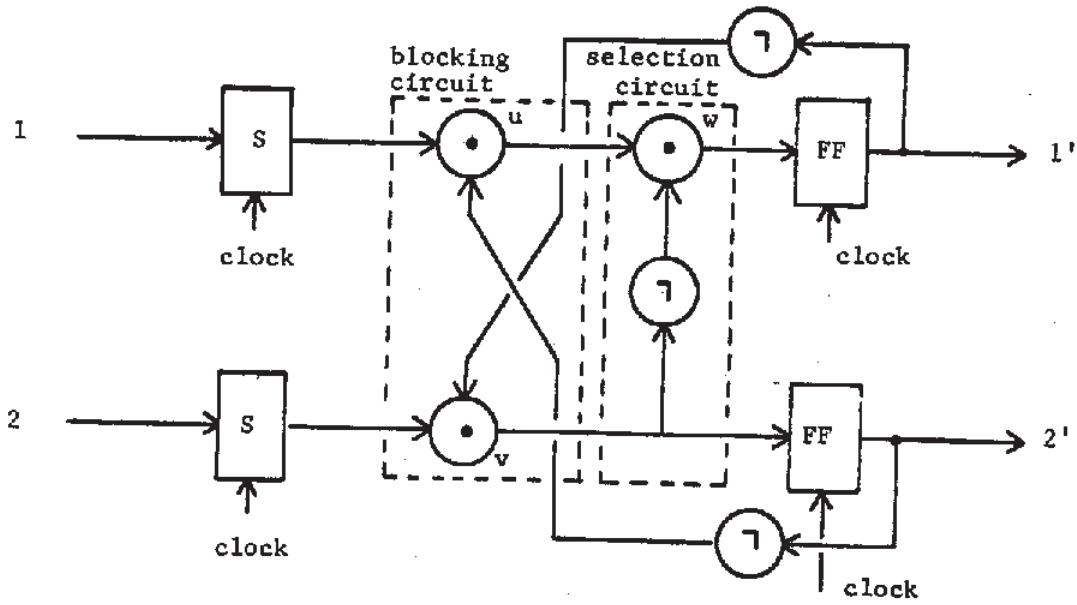
Figure 5. A bounded delay arbiter realized from synchronizers.
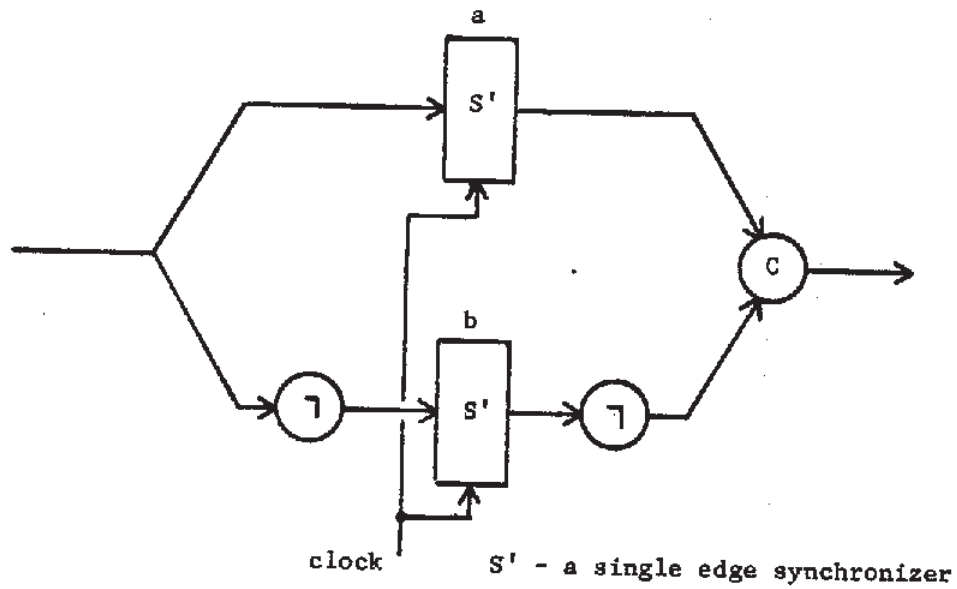

S' - a single edge synchronizer

Figure 6. A double edge synchronizer from single edge synchronizers.

The C-element prevents the output from changing until the synchronized signal is received from the appropriate synchronizing unit [12]. Basically, for the 0 to 1 transitions the C-element behaves like an AND gate and for the 1 to 0 transitions like an OR gate; the C-element output assumes the value of the input either when inputs are both 0 or both 1. In the remaining situations, in which both inputs are not the same, the C-element output maintains its current value. A transition table and a gate level implementation of the C element is shown in Figure 7.

A single edge synchronizer can be realized from the arbiter with a time bound as shown in Figure 8. This configuration synchronizes only the 0 to 1 transitions. The 1 to 0 transitions go through without any obstruction. Notice that an inverted clock signal is fed into one input of the arbiter so that the arbiter is free to be engaged by the signal to be synchronized only at the time defined by the inverted clock pulse. The second arbiter is required only if there is a possibility that some times the signals take less time to get through the arbiter than at other times. If the arbiter always takes the same amount of time, the second arbiter can be replaced by an appropriate delay.

The double edge synchronizer can be realized by connecting two of these synchronizers in an arrangement described earlier.

Above we have discussed the relationship between the synchronizers and the synchronous arbiters which are needed in the synthesis of synchronous systems. In asynchronous systems there is no need for synchronizing a signal to a clock,but instead synchronization takes the form: "Wait for a signal from system A and a signal from system B before initiating the action of system C." Synchronization of this kind corresponds to the primitive 'join' in parallel programming languages. Such synchronization can be easily realized with Muller's C element [12]. There
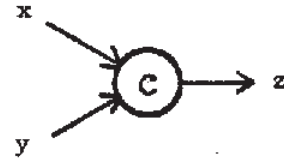
a) transition table
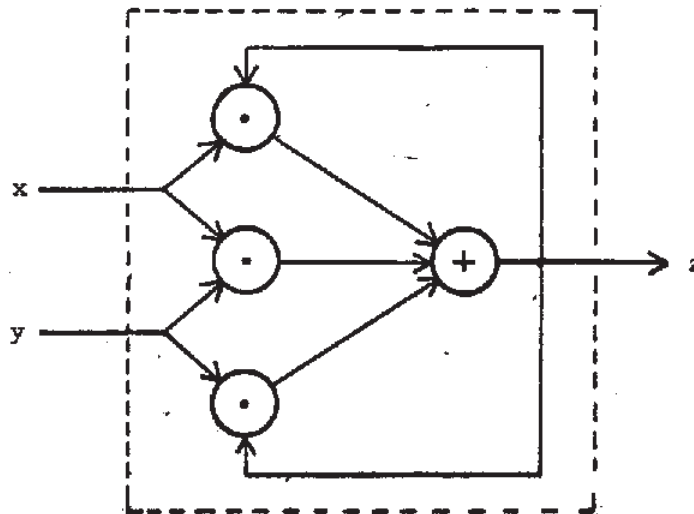
b) circuit realization
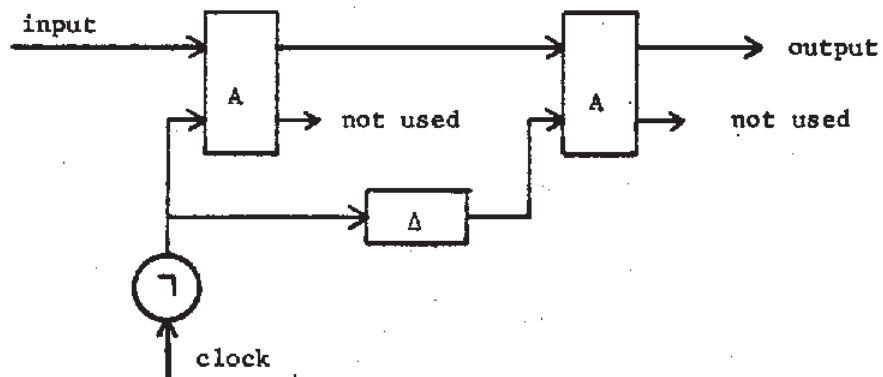
Figure 7. Muller's C element.

Figure 8. Realization of a positive edge synchronizer
from arbiters of bounded delays.

is no difficulty in realizing the C-element for use in synchronization because
there is no conflict involved as the synchronizer waits for both inputs before
sending out a signal. There is no direct relationship between the arbiters and
synchronizers in asynchronous systems: the synchronizers do not encounter any
conflict, but the arbiter must deal with conflicts.

The Meta-Stable State and Its Effects

All bistable devices have an unstable state in addition to the two stable
states of the device (see the example of a seesaw in Figure 9). The unstable state
of equilibrium is intermediate between the two stable states and unfortunately the
device must pass through it on the way from one stable state to the other stable
state. [2,4,7].

Consider a bistable electrical device such as a flip-flop which consists of
two NOR gates. When both the set and reset inputs are at level 0, the flip-flop
has two stable states, one corresponding to the output level being 0 and the other
corresponding to the output level being 1. To see that the flip-flop has an inter-
mediate state of equilibrium called the meta-stable state, we must treat the flip-flop
as an electrical device. The two gates of the flip-flop, each of which behaves
as an amplifier, are connected in a closed circuit (Figure 10). To determine
the points of equilibrium we could open the loop and examine the open loop
transfer function $y = f(x)$. The points of equilibrium are then given by the
solutions to the equations $y = f(x)$ and $y = x$, which correspond to the inter-
section of the transfer function and the $y = x$ line in Figure 11. In that fig-
ure, a and b are points of stable equilibrium and c is a point of unstable
equilibrium. In the unstable equilibrium the levels at the output of the flip-
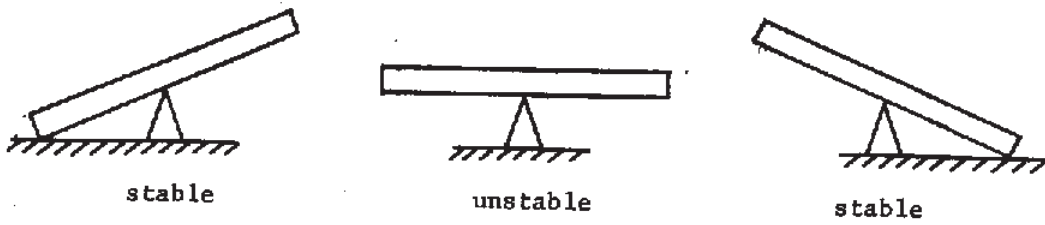
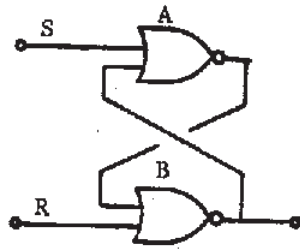Figure 9.   Three states of equilibrium of a seesaw.
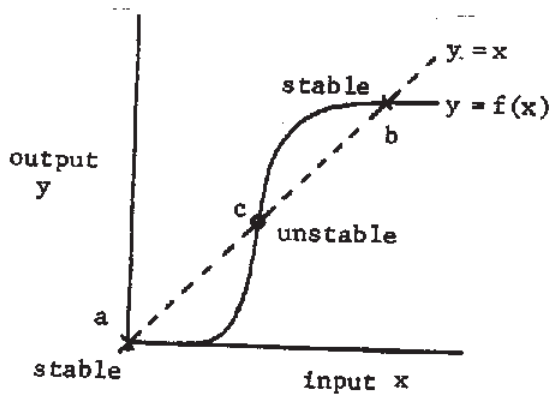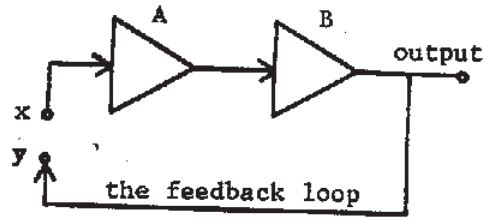

Figure 10.   A flip-flop.

Figure 11.   The unstable state.

flop are intermediate between the levels corresponding to logical 0 and 1.

The flip-flop in a synchronizer may enter into the meta-stable state if the input signal changes at the same time as the clock pulse because under this condition the pulse reaching the flip-flop (the set input of the flip-flop in Figure 2) may not have enough strength to completely flip the flip-flop, and depending on the relative timing of the signal and the clock pulse, the pulse may have just enough strength to bring the flip-flop into the meta-stable state and leave it there. In this case the output of the flip-flop rises to a value intermediate between 0 and 1 and remains there until the flip-flop leaves the meta-stable state and goes to one of the two stable states.

Both the duration for which the flip-flop stays in the meta-stable state and the stable state to which it goes from the meta-stable state are uncertain (Figure 12). Thus, under the critical operation, the synchronizer output goes to an intermediate value and later either goes to 1 or falls to 0 at a time which is not necessarily the time defined by the clock pulses (Figure 13). Both the intermediate value and the changes taking place anywhere between the clock pulses constitutes an incorrect operation for the synchronizer [2,4].

The intermediate level can be suppressed from the output if the synchronizer is modified by connecting a Schmitt trigger between the synchronizing flip-flop and the output (Figure 14). The Schmitt trigger acts as a threshold gate whose threshold is set high for the 0 to 1 transitions and low for the 1 to 0 transitions. In effect the Schmitt trigger prevents the output from changing until the synchronizing flip-flop comes out of the meta-stable state and definitively enters into the opposite stable state; if the flip-flop in the meta-stable state falls back to the original state, the output is not affected.

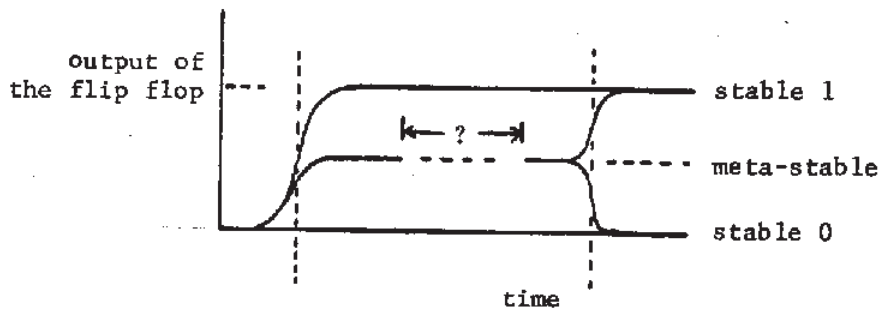Figure 12.   The possible output of the flip-flop.

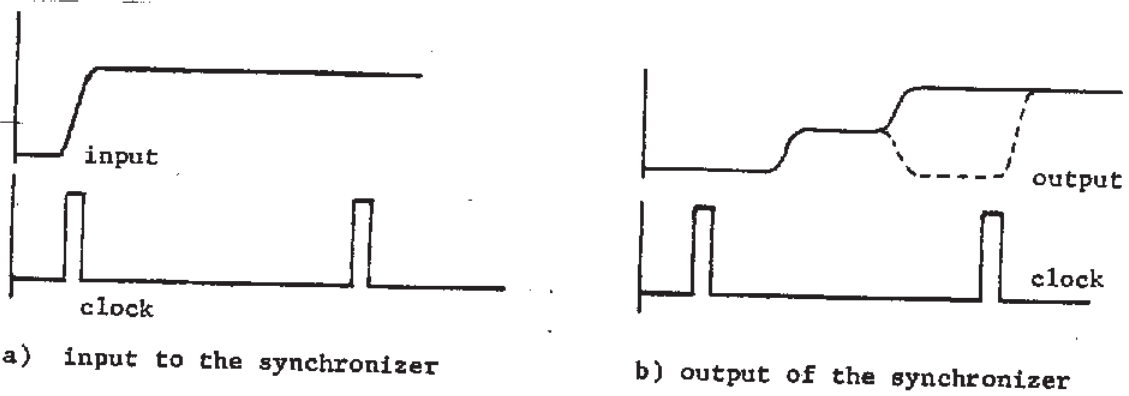a) input to the synchronizer

b) output of the synchronizer

Figure 13.   Possible output of the synchronizer under critical operation.
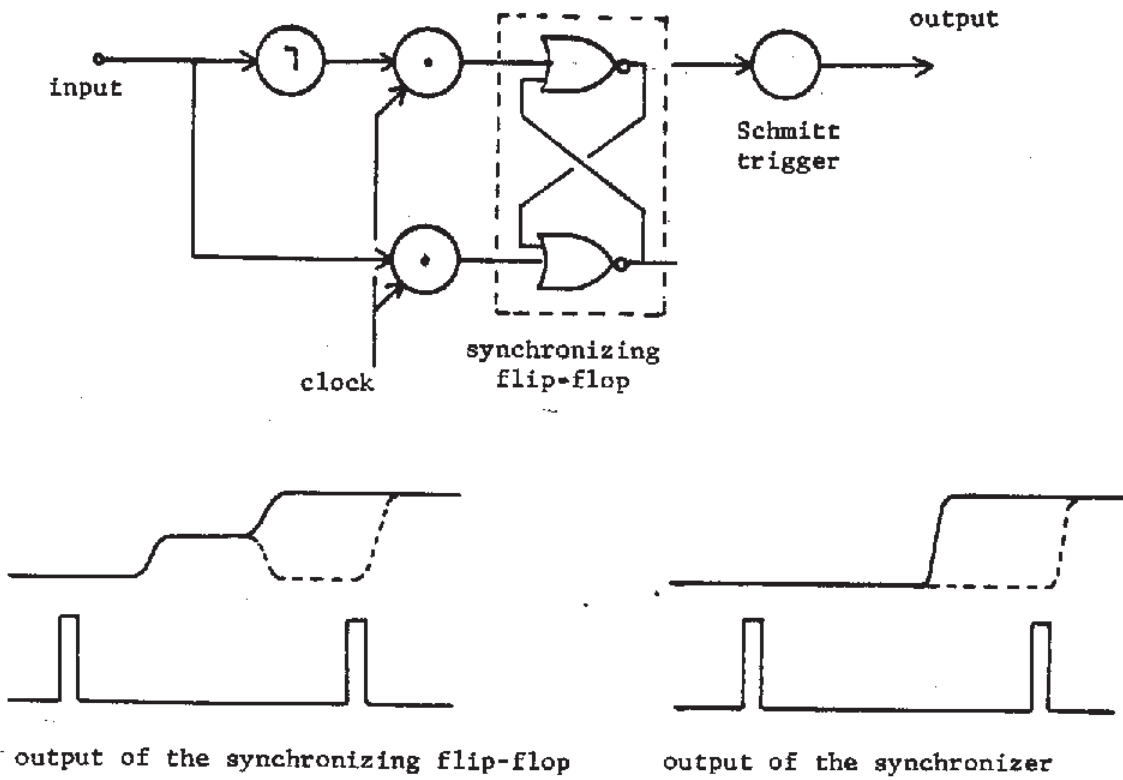
Figure 14. Masking the intermediate output of the synchronizer.

The above modification eliminates the problem of intermediate level at the output, but the synchronizer still has the defect that under the critical condition the output may not be synchronized with the clock.

Parameters of a Flip-Flop

We shall now focus our attention on the measures of the quality of a flip-flop, the critical element affecting the performance of the synchronizers and arbiters. The objective is to find parameters which will (i) permit comparison of one flip-flop against another and (ii) enable us to derive the measures of quality for synchronizers and arbiters. There are three parameters of a flip-flop which directly affect the synchronizers and arbiters: (i) an effective conflict window W_c which is a measure of how easily a flip-flop may enter the meta-stable state, (ii) a time constant τ which is a measure of how long a flip-flop may stay in the meta-stable state and (iii) Δ_0, the normal time the flip-flop takes to operate (i.e. the time of operation when there is no conflict).

We shall first explain the parameter τ. Laboratory experiments and theoretical studies have shown that the probability that a flip-flop will still be in the meta-stable state at time Δ given that it was in the meta-stable state at time 0 is given by an exponential distribution $P(\Delta) = e^{-\Delta/\tau}$ (Figure 15) [3,4,6,8,9,14]. The exponential distribution is completely characterized by a time constant τ which is a good measure of the quality of the flip-flop with regard to how easily it leaves the meta-stable state; smaller τ means that the flip-flop quickly leaves the meta-stable state.

If we were to perform an experiment in which a number of identical flip-flops are put into the meta-stable state at time 0 and observation is made
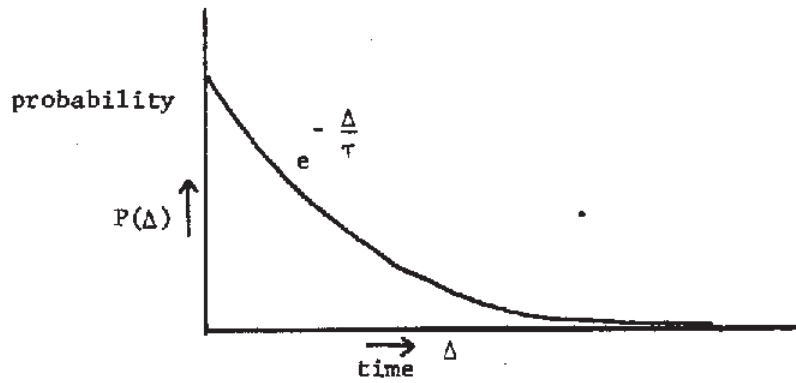
Figure 15. The probability of the flip-flop being in the meta-stable state.

W_c = effective conflict window

W_a = absolute conflict window

probability of a flip flop being in the meta-stable state at Δ_0 against the temporal separation between the signal and the clock
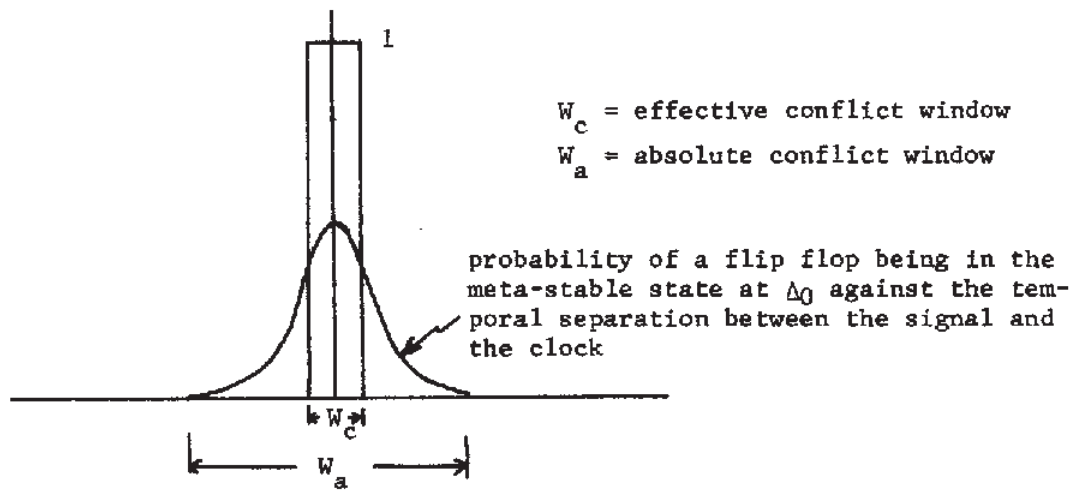
Figure 16. The conflict window.

of the number of them in meta-stable state at various times, we would also get
an exponential plot with the same time constant. That is, if at time 0 some
number k of flip-flops are in the meta-stable state then at time Δ, $k \times e^{-\Delta/\tau}$
number of them would be found in the meta-stable state. This distribution,
obtained through experiment, gives us a basis for measuring τ.

We next discuss a measure of how easily a flip-flop might enter the meta-
stable state. We expect that the probability of the flip-flop entering the meta-
state would be highest when the signal change and the clock pulse coincide, and
the probability would decrease as the temporal separation between the
clock and the change in the signal is increased. There are several questions we
must face at once. At what time after the clock pulse must we observe the flip-flop
to determine if it entered the meta-stable state; we cannot observe the flip-
flop immediately after the clock pulse because the flip-flop has some inherent
delay Δ_0 , the time for which the flip-flop does not respond in any case. We
cannot wait too long either because that might give the flip-flop a chance to
get out of the meta-stable state. Therefore a good suggestion is to observe
the flip-flop at Δ_0 units of time after the clock pulse where Δ_0 is the normal delay;
this is the time in which the flop-flop is expected to switch to the appropriate
stable state if there is no conflict. But even this suggestion has some difficulties
because observation of the meta-stable state at Δ_0 is obstructed by the fact
that as the temporal separation between the clock and the signal change
decreases, the flip-flop may take longer than Δ_0 to respond in any manner.
Therefore, the best we can do is to estimate the probability of the flip-flop
being in meta-stable state at Δ_0 by observing the behavior of the flip-flop at
a later time. This can be done by conducting an experiment in which
the temporal separation between the clock and the signal change is kept constant
and a plot similar to the one for measurement of τ is made.

This graph can be extrapolated backwards until it meets Δ_0 to give the estimated probability of the flip-flop being in meta-stable state at time Δ_0.

Comprehensive information about how easily a flip-flop gets into a meta-stable state would then be given by a plot of the probability of the flip-flop being in the meta-stable state at time Δ_0 against the temporal separation between the clock and the signal change. This much information is, however, not generally needed especially when the signals to be synchronized can be assumed to be completely asynchronous with respect to the clock which is generally the case. Under these circumstances a measure called the <u>effective conflict window</u> W_c which is equal to the area of the above plot of probability against time is sufficient for our purposes (Figure 16). The effective conflict window has a very useful intuitive interpretation: the conflict window W_c can be imagined as a window extending from $-\frac{1}{2} W_c$ to $+\frac{1}{2} W_c$ and, for the purposes of computing overall performance, we could say that the flip-flop always enters into the meta-stable state if the temporal separation between the clock and signal change falls within this window and does not enter the meta-stable state if it does not. Thus, the smaller the conflict window, fewer will be the times the flip-flop will enter into the undesired meta-stable state.

Both parameters τ and W_c can be determined by an experiment which involves repeated trials in which the temporal separation between the signal and the clock is uniformly varied over the range $-\xi/2$ to $\xi/2$ when ξ is larger than the absolute conflict window.

The outcome of the experiment is a plot of the number of observations of meta-stable state against time measured from the clock pulse. As we have said previously this plot will be exponential (Figure 17). The time constant τ that we desire is the time constant of this curve, and if the curve is described by

the expression $m \times e^{-(\Delta-\Delta_0)/\tau}$, the conflict window will be given by the expression $W_c = \frac{m}{n} \times \xi$ where n is the total number of trials. Therefore, the expression for the plot of probability can also be written as $\frac{n}{\xi} \times W_c \times e^{(\Delta-\Delta_0)/\tau}$.

We can use these parameters of the flip-flop to compute the measures of the quality of a synchronizer. The measures are very similar to the measures for a flip-flop; an effective error window and a delay. When a single flip-flop is used as a synchronizer, the error window is the same as the conflict window of the flip-flop. By cascading stages of synchronizer, however, the effective error window can be decreased to any desired degree but at the expense of increased delay. A two stage synchronizer is shown in Figure 18. For this synchronizer to fail both stages must fail. This happens only if the first stage fails and causes such delay that the signal reaches the second stage in its conflict window.

To estimate the error window of the combined synchronizer we can imagine an experiment in which n trials are performed varying the separation between the clock and the signal uniformly over $-\xi/2$ to $\xi/2$. The expected number of times the first stage will be in the meta-stable state at time Δ is given by the expression $\frac{n}{\xi} \times W_c \times e^{-(\Delta-\Delta_0)/\tau}$ (see the paragraph above). Therefore, the expected number of times the first stage will come out of the meta-stable state in the interval of time from $\Delta - (1/2)W_c$ to $\Delta + (1/2)W_c$, the critical interval, will be $\frac{n}{\xi} \times W_c \times e^{-(\Delta-(1/2)W_c-\Delta_0)/\tau} - \frac{n}{\xi} \times W_c \times e^{-(\Delta+(1/2)W_c-\Delta_0)/\tau}$, which is equal to $\frac{n}{\xi} \times W_c \times e^{-(\Delta-\Delta_0)/\tau} \times (e^{W_c/2\tau} - e^{-W_c/2\tau})$.

If $W_c/2\tau$ is small, this expression reduces to $\frac{n}{\xi} \times W_c \times e^{-(\Delta-\Delta_0)\tau} \times \frac{W_c}{\tau}$. This is the expected number of times the first synchronizer will leave the meta-stable state in the range $\Delta - (1/2)W_c$ to $\Delta + (1/2)W_c$. If we assume that the probability of the meta-stable state switching to state 1 is 1/2, the expected number

- total number of trials = n
- temporal separation between the clock and the signal is distributed uniformly over $-\xi/2$ to $\xi/2$

number of observations of the meta-stable state

$W_c = \dfrac{m}{n} \times \xi$

$m\,e^{-\frac{\Delta-\Delta_0}{\tau}}$

$\tau = \dfrac{\Delta_2 - \Delta_1}{\log_e m_1 - \log_e m_2}$

m, m_1, m_2

$\Delta_0 \quad \Delta_1 \quad \Delta_2$

units of time following the clock pulse
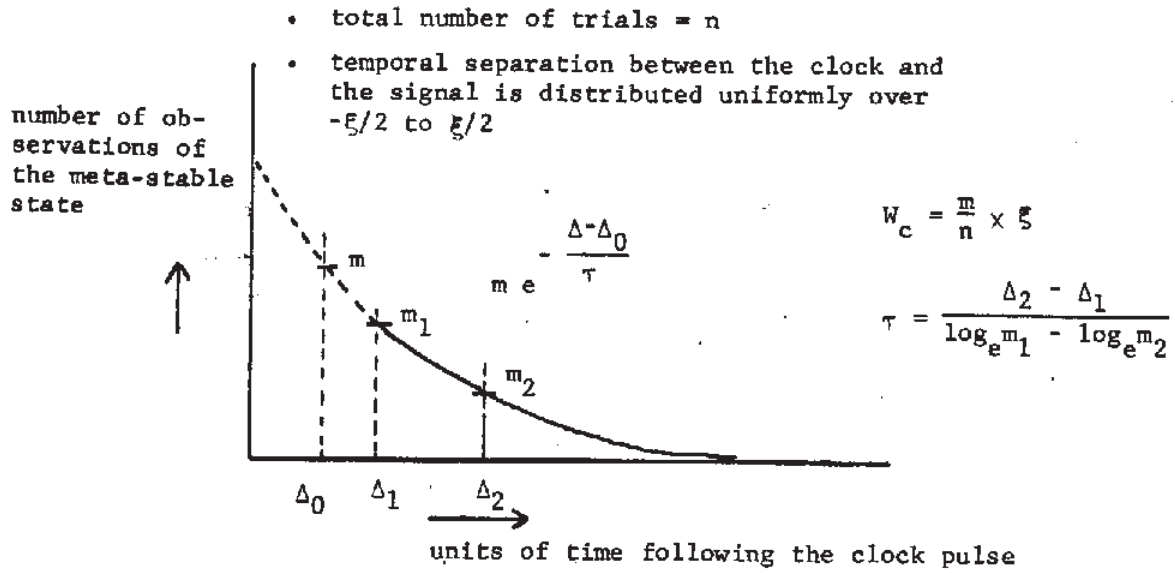
Figure 17. Determination of τ and W_c.

signal → S → S → output
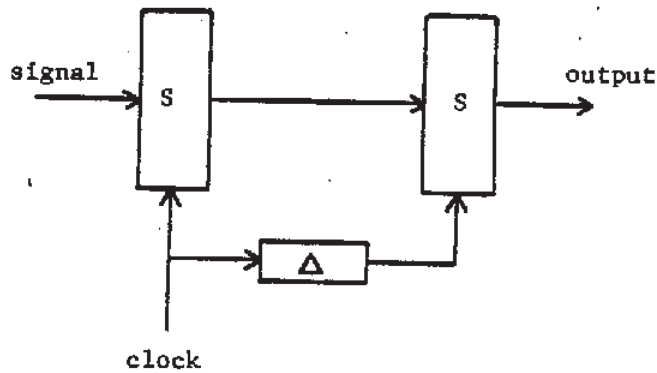
Δ

clock

Figure 18. A two-stage synchronizer.

of times the first stage will experience delay in the said range will be $\frac{1}{2} \times \frac{n}{\xi} \times W_c \times e^{-(\Delta-\Delta_0)/\tau} \times \frac{W_c}{\tau}$. This expression also represents the expected number of times both stages will fail because a delay in the range from $\Delta - (1/2)W_c$ to $\Delta + (1/2)W_c$ causes the signal from the first stage to arrive at the second stage in the conflict window. Therefore, the error window for the two-stage synchronizer is given by

$$W_e = W_c \times e^{-(\Delta-\Delta_0)/\tau} \times \frac{W_c}{2\tau}$$

The above expression could also be derived in the following manner very helpful for an intuitive perception of the problem. Because the n trials are distributed uniformly from $-\xi/2$ to $\xi/2$, and the conflict window is W_c wide, the expected number of times the first stage will fail is $\frac{n}{\xi} \times W_c$. Furthermore, given that the first stage has failed, the probability that the failure will lead to a delay in the conflict window of the next stage is $\frac{1}{2} \times (e^{-(\Delta-\Delta_0)/\tau} \times \frac{1}{\tau}) \times W_c$. In this expression, $e^{-(\Delta-\Delta_0)/\tau} \times \frac{1}{\tau}$ is the probability density distribution obtained by taking the negative of the derivative of $e^{-(\Delta-\Delta_0)/\tau}$ with respect to Δ. It will be recalled that $e^{-(\Delta-\Delta_0)/\tau}$ is the expression for the plot of the probability of the synchronizer being in the meta-stable state at time Δ given that it was in the meta-stable state at time Δ_0. The negative of the derivative of this expression gives the rate at which the synchronizer leaves the meta-stable state at time Δ. The term W_c comes from the fact that the conflict window is W_c wide. The product of W_c and the above stated rate of exit at Δ gives the expected number of times the synchronizer will leave the meta-stable state in the said critical interval. The term $\frac{1}{2}$ in the expression comes from the assumption that the synchronizer is equally likely to go to state 0 as is likely to go to state 1. The expected number of times both stages fail is therefore equal to

$$\frac{n}{\xi} \times W_c \times \frac{1}{2} \times e^{-(\Delta-\Delta_0)/\tau} \times \frac{1}{\tau} \times W_c$$

and from this we get the expression for error window

$$W_e = W_c \times \frac{W_c}{2\tau} \times e^{-(\Delta-\Delta_0)/\tau}$$

This expression shows that the error window of the two stage synchronizer decreases exponentially with the delay between stages. Furthermore, for any given Δ, the error window for the two stage synchronizer is narrower than the single stage synchronizer by a factor of $\frac{2\tau}{W_c} \times e^{(\Delta-\Delta_0)/\tau}$.
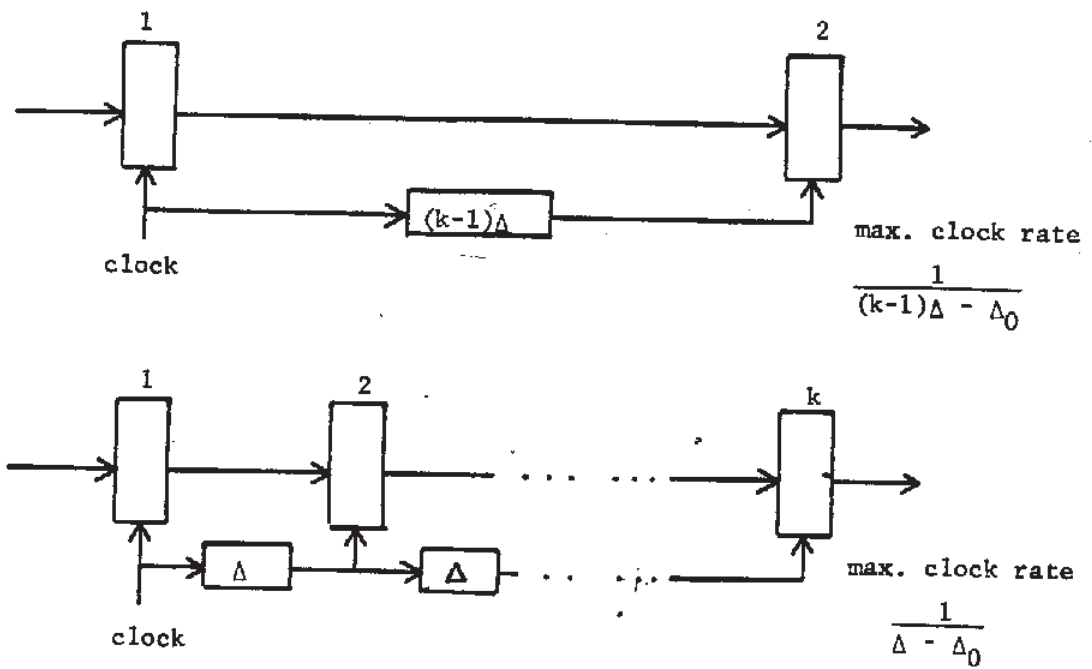
Iterative Structures for Synchronizers

The analysis of the two stage synchronizer has shown that the error window can be reduced exponentially with delay. It is, therefore, possible to reduce the probability of the synchronizer failure caused by the meta-stable state much below the probability of failure of the synchronizer due to the physical failure of the electronic components. But as the speed of the digital system is pushed to the limit, the synchronizer delay required to achieve this desired degree of reliability becomes unacceptably large. Since things are being pushed to the limit, it is reasonable to ask if it is possible to improve the reliability of the synchronizer beyond what is possible with the two-stage synchronizer. In particular we are interested in examining if a multi-stage iterative structure could give a synchronizer of lower probability of error than the two-stage synchronizer.

The expression for the effective error window of the two-stage synchronizer shows that the probability of failure goes down exponentially with the overall delay of the synchronizer. Is there any way for improving upon the time constant of this exponential function through cascading of synchronizers? The following analysis gives an affirmative answer to this question.

Consider a two-stage and a k-stage synchronizer which have the same overall delay (Figure 19); the k-stage synchronizer has Δ unit of delay between each stage and the two-stage synchronizer has a $(k-1) \times \Delta$ unit delay between the two stages. The k-stage synchronizer will fail only if all stages of the synchronizer fail to synchronize the signal. The probability that i^{th} stage will fail given that the stage $i-1$ has failed is $\frac{W_c}{2\tau} \times e^{-(\Delta-\Delta_0)/\tau}$. Therefore, the error window of the k-stage synchronizer is $W_c \times (W_c \times \frac{1}{2\tau} \times e^{\Delta_0/\tau})^{k-1} \times e^{-(k-1)\Delta/\tau}$. The error window for the two-stage synchronizer is $W_c \times (W_c \times \frac{1}{2\tau} \times e^{\Delta_0/\tau}) \times e^{-(k-1)\Delta/\tau}$. This shows that whether the error window of the k-stage synchronizer is smaller or larger than the two-stage synchronizer depends on whether the quantity $(2\tau/W_c)e^{-\Delta_0/\tau}$ is greater than or less than 1; if it is greater than 1 then the error window of the k-stage synchronizer will be smaller than the two stage synchronizer. Therefore, we call this quantity the __figure of merit__, μ, of the flip-flop. The error window of the two-stage synchronizer is μ^{k-2} times that of the k stage synchronizer.

If μ is less than 1, the two-stage synchronizer gives the smallest error window, but if μ is greater than 1, the best arrangement consists of as many stages as is possible (it is assumed that the permissible delay is large enough to accommodate two stages, otherwise the single stage synchronizer would be the only possibility and one would have to be satisfied with an error window equal to W_c). Since the overall delay of the synchronizer is fixed, the maximum number of stages is realized when Δ is made as small as possible. The delay Δ, which is the delay between two successive stages, can in no case be made smaller than Δ_0, the propagation delay of the synchronizer under normal conditions. In a practical circuit, Δ will have to be slightly larger than Δ_0 to avoid some undesired behavior of flip-flops

$$W_e^2 = W_c \times \left(\frac{W_c}{2\tau} \; e^{\frac{\Delta_0}{\tau}} \right) \times e^{-\;(k-1)\Delta/\tau}$$

$$W_e^k = W_c \times \left(\frac{W_c}{2\tau} \; e^{\frac{\Delta_0}{\tau}} \right)^{k-1} \times e^{-(k-1)\Delta/\tau}$$

$$W_e^k < W_e^2 \quad \text{iff} \quad \left(\frac{2\tau}{W_c} \times e^{-\frac{\Delta_0}{\tau}} \right) > 1 \qquad [k > 2]$$

Figure 19. A multi-stage synchronizer against the two-stage synchronizer.

at a delay in the vicinity of Δ_0. For example, it is observed experimentally that when the separation between the clock and the signal change is reduced, the normal delay of a flip-flop increases slightly before non-predictable effects due to the meta-stable state are encountered. If Δ is not greater than the maximum such delay, the stages leading to the last stage can all be made to fail in a manner so as to put the signal in the conflict region of the final stage by a suitable choice of separation between the input and the clock.

Even if μ is equal to 1, the k-stage synchronizer may have an advantage over the two-stage synchronizer when high clock rates are being used. The maximum clock rate with which the two-stage synchronizer can be used is $\dfrac{1}{(k-1)\Delta-\Delta_0}$ while the k-stage synchronizer can be used with clock rate $\dfrac{1}{\Delta-\Delta_0}$ which is at least (k-1) times that for the two-stage synchronizer. The ability to use higher clock rates without degrading the error window is important when the internal clock rate of the system to which a communication link is connected is so high that the time between clock signals is not sufficiently long to achieve synchronization with the desired reliability.

We will now study the expression for the error window further to show that the figure of merit of a flip-flop is a paramter of the same significance as the time constant τ. Suppose we keep the time delay between stages of a multi-stage synchronizer fixed at Δ and change the number of stages to obtain the desired low error window. Let μ, the figure of merit, be expressed as $e^{\Delta/\tau'}$ where Δ is the delay of each stage and τ' is a constant such that $\mu = e^{\Delta/\tau'}$. The expression for the error window of the multi-stage synchronizer now becomes

$$W_e = W_c \times e^{-(k-1)\Delta/\tau'} \times e^{-(k-1)\Delta/\tau}$$

where $(k-1)\Delta + \Delta_0$ is the overall delay of the synchronizer. If the total time is denoted by T, the expression assumes the form

$$W_e = W_c \times e^{-T/\tau'} \times e^{-T/\tau} \times e^{\Delta_0(1/\tau + 1/\tau')}$$

which shows that the error window decreases exponentially with the overall delay on account of two exponentials, one arising directly out of the time constant τ and the other out of the figure of merit of the flip-flop. Furthermore, because the figure of merit, the delay Δ and the time constant τ' are related by the expression $\mu = e^{-\Delta/\tau'}$, for a given figure of merit if the delay between the stages is reduced, the time constant τ' will be reduced proportionately, i.e. the smaller the delay, the smaller will be the time constant τ, which is very desirable. Thus, even this analysis suggests that Δ should be made as small as possible.

A Synchronizer Without a Time Bound

Even though synchronization of the signal within a fixed number of clock periods is desirable, this is not an absolute requirement; what is essential is that the signal transitions be properly synchronized to the clock as early as possible. A synchronizer which takes only a few (perhaps just a single) clock period to perform its task most of the times but takes an arbitrary number of clock periods some of the times should be acceptable. Surprisingly even such a synchronizer, which now does not have the strict requirement that it must operate in a fixed number of clock periods, cannot be physically realized.

This reason why such a synchronizer cannot be physically realized is that a single edge synchronizer with a bound can be realized from two single edge synchronizers which take a bounded number of clock periods when there is no conflict and take an unbounded

number of clock periods when there is a conflict as shown in Figure 20. The signal is sent directly to one synchronizer and through a delay to the other synchronizer. The delay is large enough so that at most one of the two synchronizers may experience critical operation. The one which experiences critical operation either operates at the same time as the other or is delayed in its action. The rising edge at the output, which is obtained by taking a logical OR of both synchronizers is defined by the synchronizer which responds first -- the one that does not encounter critical operation. The output thus responds to the input in just the time needed when there is no critical operation, which is bounded.

It may be recalled that a single edge synchronizer synchronizes only, say, 0 to 1 transitions; the 1 to 0 transitions go through unobstructed and initialize the synchronizer. It may further be recalled that a double edge synchronizer can be realized from two single edge synchronizers (Figure 6).

We chose to obtain a synchronizer with a bound from single edge synchronizers because the realization of a single edge bounded synchronizer from the unbounded single edge synchronizers is most easily accomplished. This is so because the 0 to 1 transitions (the ones which are not synchronized) reset the synchronizer. Resetting the synchronizer is necessary to prevent a component synchronizer that is still in the meta-stable state due to the previous action from affecting the next instance of synchronization. A double edge synchronizer without bounds that is provided with a reset input can also be used easily.

The above scheme for obtaining a bounded delay synchronizer can also use synchronizers that produce a synchronized signal in k or fewer clock instances when no conflict is involved and produce an unsynchronized output after $k+1$ clock instances (counting the one at which the signal arrives) when there is a conflict. The fact that the output of the synchronizer is guaranteed not to change in the interval between k^{th} and $k+1^{st}$ clock instances is a fact of crucial importance.

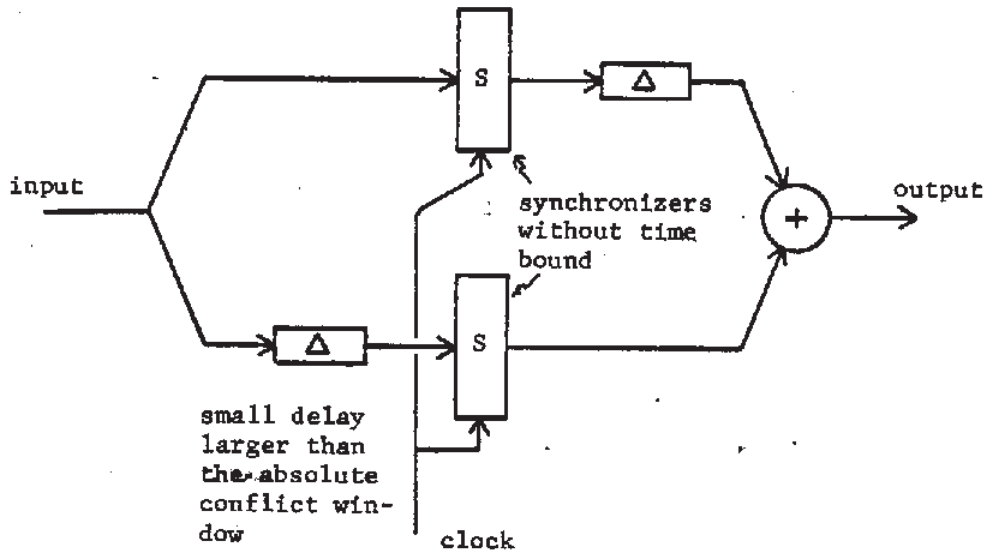Figure 20. A single-edge synchronizer with bound from single-edge synchronizer without time bound.

Iterative Structures for Arbiters

In this section we shall focus our attention on iterative structures for arbiters with a time bound, i.e. the arbiters needed in synchronous systems. We will define a single stage arbiter called a limited resolution arbiter and show how the desired arbiter can be obtained by cascading limited resolution arbiters. We will also analyze the error window of these arbiters and derive the condition under which a multi-stage arbiter is better than the two-stage arbiter.

In the construction of a limited resolution arbiter we need a blocking element, designated as β, that has a block input b, and a transmit input t. A circuit realization of the β element (shown in Figure 21) consists of a flip-flop constructed from two NAND gates and a threshold NOT gate which is connected to the output of one of the NAND gates. The threshold of the threshold gate is adjusted lower than the level of the meta-stable state of the flip-flop. The input of the NAND gate to which the threshold element is connected is designated the transmit input and the input of the other NAND gate is designated the block input. The operation of this circuit is explained below.

Initially both inputs are at level 0 and the output is also at level 0. If the transmit input is changed to 1 while the block input is held at 0, the output will change to 1. But if the block input is raised to 1 before the transmit input is changed to 1, the output will remain at 0. The situation in which both the block and the transmit inputs change to 1 at nearly the same time represents a conflict situation in which the flip-flop component of the circuit may enter into the meta-stable state causing uncertainty about when, if at all, the output will change to 1. Since the threshold of the NOT gate is adjusted lower than the level at the meta-stable state, the output changes only when the flip-flop comes out of the meta-stable state and settles into the state in which the flip-flop output connected to the NOT gate is low.
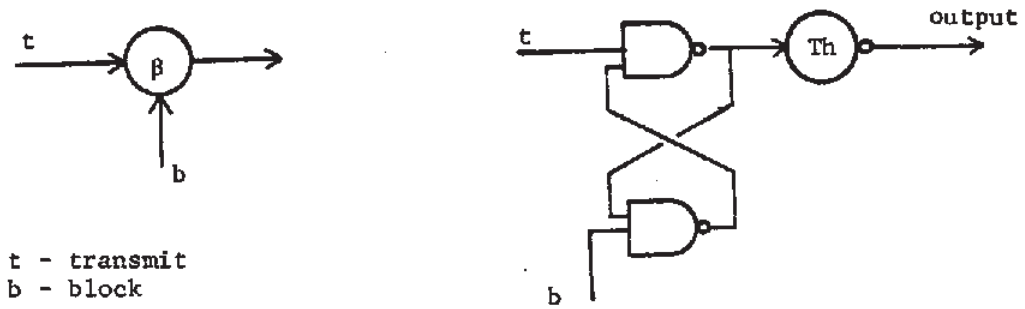
t - transmit
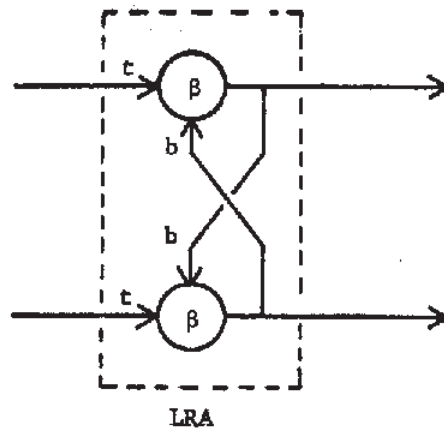b - block

Figure 21.   The blocking element.

Figure 22.   A limited resolution arbiter.

The limited resolution arbiter is composed of two blocking elements connected in the criss-cross arrangement shown in Figure 22. If one input of the limited resolution arbiter is changed to 1 while the other one is held at 0, the associated output will change to 1, and at the same time the blocking element associated with the other input will be put in the blocked condition. If the other input should now change to 1, this change will be blocked from reaching the output. The limited resolution arbiter, therefore, performs arbitration correctly if the signals are separated in time by a duration at least equal to the sum of the time that a signal at the transmit input of the block element takes to appear at the output under normal condition and the time a signal at the block input takes to (completely) block the blocking element. Twice this duration of time is called W_{ra}, the (absolute) resolution window of the limited resolution arbiter; the (absolute) resolution window extends from $-\frac{1}{2} W_{ra}$ to $+\frac{1}{2} W_{ra}$. In our discussion, the statement that the separation between two signals falls within the (absolute) resolution window will means that the temporal separation between the signals falls within the range $-\frac{1}{2} W_{ra}$ to $+\frac{1}{2} W_{ra}$. Thus we can say that the limited resolution arbiter performs correctly as long as the signals do not fall within the absolute resolution window of the arbiter.

If the two signals to the limited resolution arbiter arrive simultaneously, they will both manage to get through the arbiter because neither one will be able to block the other and the arbiter will have failed to perform arbitration. In this case neither of the signals that passed through the arbiter will experience any uncertain delays. The situation in which the signals marginally fall within the resolution window is the one in which uncertain delays may be encountered in the second signal because in this case the output from the first signal will try to block the second signal just as it arrives at the blocking element causing a conflict which

may give rise to uncertain delays. Note that only the second signal (the one that comes late) has the possibility of being delayed; the first signal experiences only the normal delay.

To understand how cascading can improve the resolution, consider for the moment an idealized limited resolution arbiter; if the signals do not fall within the resolution window it successfully blocks one input and if they fall within the resolution window then both signals pass through experiencing only the normal delay. We have thus assumed for the time being that conflict situation does not arise.

Two idealized limited resolution arbiters can be connected in tandem as shown in Figure 23 to realize a perfect (infinite resolution) arbiter. This structure exploits the fact that if both signals get through the idealized limited resolution arbiter, they will be within the resolution window at the output of the arbiter because this happens only when the input signals are within the resolution window, and the limited resolution arbiter being ideal, both signals experience only the normal delay. If one of the signals is now purposefully delayed by more than the resolution window, we can be certain that the signals will be outside the resolution window, and, therefore, the signals will be sure to be resolved at the second limited resolution arbiter. If the idealized limited resolution arbiters are replaced by the ordinary ones, the arrangement will not give a perfect arbiter, but it will be a much improved arbiter over the single limited resolution arbiter. In fact unlike the single limited resolution arbiter, the two-stage arbiter cannot be made to fail with certainty by choice of separation between signals. Therefore, it will be necessary to talk about the measure of the quality of arbiters in terms of an effective error window as in the case of the synchronizers.

Let us look into the details of how the two-stage arbiter may fail to perform correctly. Suppose signal 2 comes first and signal 1 comes just when signal 2 is blocking signal 1. The resulting conflict may cause signal 1 to be delayed by just the amount necessary for signal 1 and the deliberately delayed signal 2 to reach
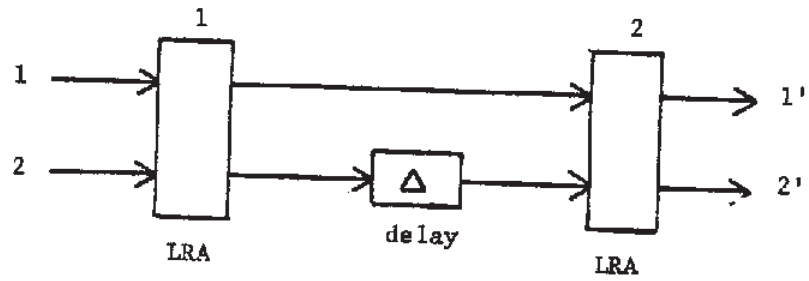
Figure 23.   A two-stage arbiter

the second stage nearly simultaneously (within the resolution window of the second limited resolution arbiter). In this case both of the signals will go through the second stage, and the arbiter will have failed to perform its stated task. Note that there is no possibility of incorrect operations when signal 1 comes first and signal 2 comes later because any conflict at the blocking element associated with signal 2 will only help increase the desired delay in the path of signal 2.

We will now obtain an expression for the effective error windows of the two-stage and the multi-stage arbiters in terms of the parameters of the blocking element β just as the effective error window of the multi-stage synchronizer was obtained in terms of the parameters of the flip-flop. The time constant τ and the conflict window W_c of the blocking element correspond to the parameters τ and W_c of the flip-flop in the circuit of the element and the propagation delay Δ_0 is equal to the sum of the propagation delay of the flip-flop and the delay of the NOT gate. The effective resolution window W_r of a limited resolution arbiter is a statistical parameter defined as follows: Consider an experiment in which n instances of arbitration are performed varying the temporal separation between the two signals uniformly from $-\xi/2$ to $\xi/2$ where ξ is larger than the absolute resolution window. If the number of times the arbiter fails to block a signal is m, the effective resolution window W_r of the limited resolution arbiter is equal to $\frac{m}{n} \times \xi$. The effective resolution window is approximately equal to $2\Delta_0$, twice the normal propagation delay of the β element.

The effective error window W_e of an arbiter (single as well as multi-stage arbiter) is estimated by imagining a similar experiment in which the temporal separation between the two signals to the arbiter is varied uniformly from $-\xi/2$ to $\xi/2$ units of time and n instances of arbitration are performed. If the estimated number of times the arbiter fails is m then the effective error window W_e is $\frac{m}{n} \times \xi$.

It may be noted that the error window of a single stage arbiter is equal to the resolution window W_r by definition. Again as in the case of the multi-stage synchronizer, the multi-stage arbiter will fail only if all stages of it fail. For this to happen all stages except the last two stages must experience a conflict which results in such delays as to cause the next stage to conflict, and the last but one stage should result in such delay as to cause both signals to arrive within the resolution window of the last stage.

With the aid of the timing diagram of Figure 24a, the expected error window for the two-stage arbiter can be calculated to be

$$W_e^2 = W_c \times (e^{-(\Delta + \Delta_0 + (1/2)W_r - \Delta_0)/\tau} \times \frac{1}{\tau} \times \frac{1}{2} \times W_r)$$

$$= W_c \times e^{-(\Delta + (1/2)W_r)/\tau} \times \frac{W_r}{2\tau} .$$

In the expression the first term comes from the width of the conflict window which is W_c, and the remaining terms give the probability that the first stage will delay input 1 by an amount that will cause both inputs to arrive in the resolution window of the second stage. A delay in the range $(\Delta + \Delta_0 + (1/2)W_r) \pm (1/2)W_r$ will cause the two signals to be in the resolution window. With the aid of the timing diagram of Figure 24b, the error window of the n stage arbiter can be calculated to be

$$W_e^n = W_c \times (e^{-(\Delta + \Delta_0 - \Delta_0)/\tau} \times \frac{1}{2\tau} \times W_c)^{n-2} \times e^{-(\Delta + \Delta_0 + (1/2)W_r - \Delta_0)/\tau} \times \frac{W_r}{2\tau}$$

$$= 2W_c \times (e^{-\Delta/\tau} \times \frac{W_c}{2\tau})^{n-2} \times e^{-(\Delta + (1/2)W_r)/\tau} \times \frac{W_r}{2\tau}$$

In order to examine whether a multi-stage arbiter is any better than a two-stage arbiter, we will now examine a two-stage arbiter and an n-stage arbiter of equal overall delay (see Figure 25). The expressions for the error windows for both of them are:
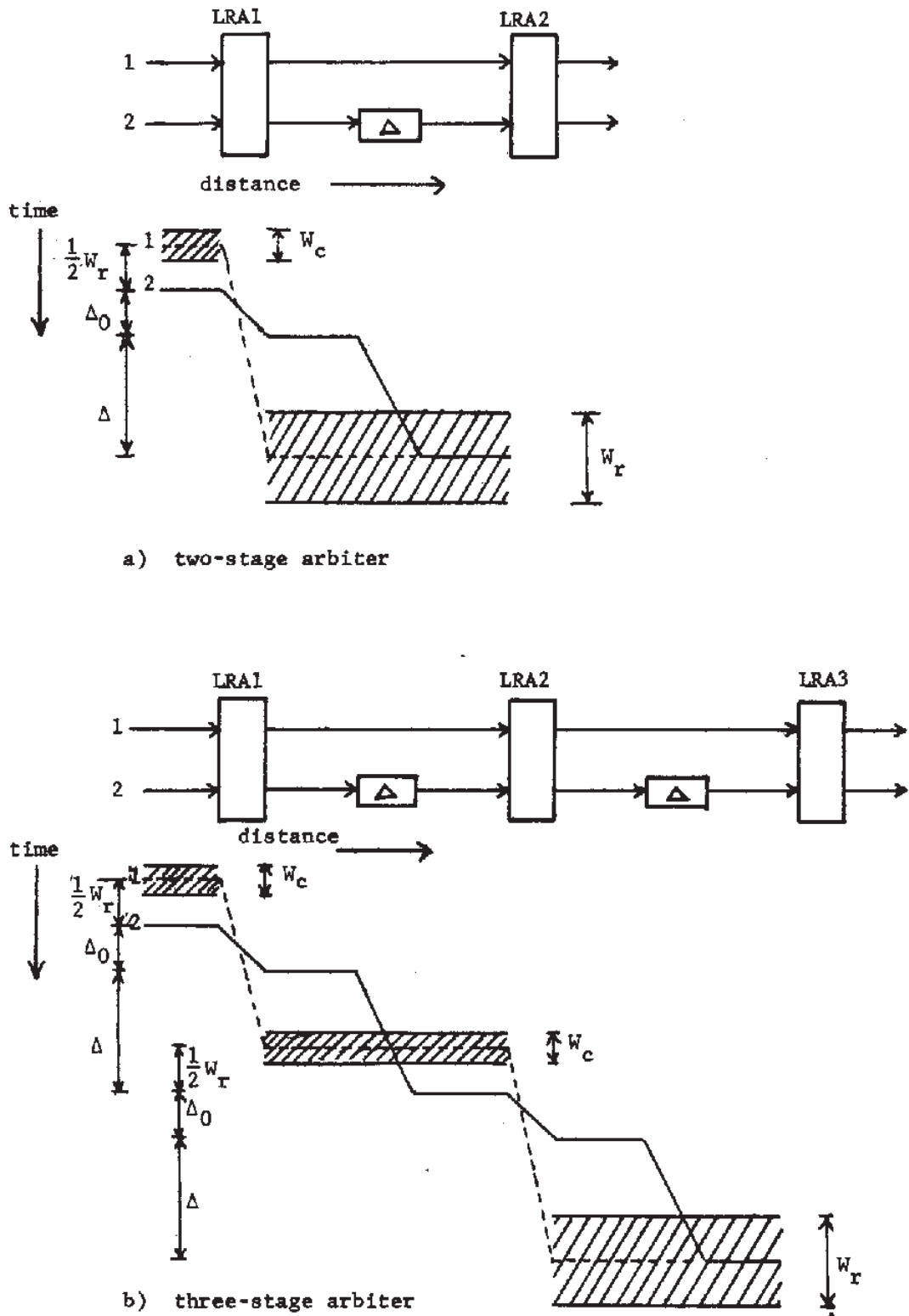
a) two-stage arbiter

b) three-stage arbiter

Figure 24. Timing diagram of critical operation.

$$W_e^n = W_c \times (e^{-\Delta/\tau} \times \frac{W_c}{2\tau})^{n-2} \times e^{-(\Delta+(1/2)W_r)/\tau} \times \frac{W_r}{2\tau}$$

$$W_e^2 = W_c \times e^{-((n-1)\Delta+(n-2)\Delta_0+(1/2)W_r)/\tau} \times \frac{W_r}{2\tau}$$

$$= (W_c \times (e^{-\Delta/\tau} \times \frac{W_c}{2\tau})^{n-2} \times e^{-(\Delta+(1/2)W_r)/\tau} \times \frac{W_r}{2\tau}) \times (e^{-\Delta_0/\tau} \times \frac{2\tau}{W_c})^{n-2}$$

The ratio $W_e^2/W_e^n = (\frac{2\tau}{W_c} \times e^{-\Delta_0/\tau})^{n-2} = \mu^{n-2}$. Therefore, the effective error window of the n stage arbiter where n is larger than 2 will be smaller than the two-stage arbiter if and only if the figure of merit μ is greater than 1. The multi-stage arbiter represents the best arrangement if the figure of merit of the blocking element is greater than 1, otherwise the two-stage arbiter is the best.

It should be noted that regardless of when the two signals arrive at the arbiters discussed above and regardless of what goes on inside the arbiters, at least one signal comes out of the arbiter within a fixed length of time. There is, however, the possibility that the arbiter may fail to stop the other input signal from going through the arbiter. The usefulness of this device depends on the ability to reduce the probability of such failures to an acceptable low value. We have already examined how this can be achieved. Next we turn our attention to arbiters which do not necessarily act in a fixed length of time but which never fail to block one of the inputs.
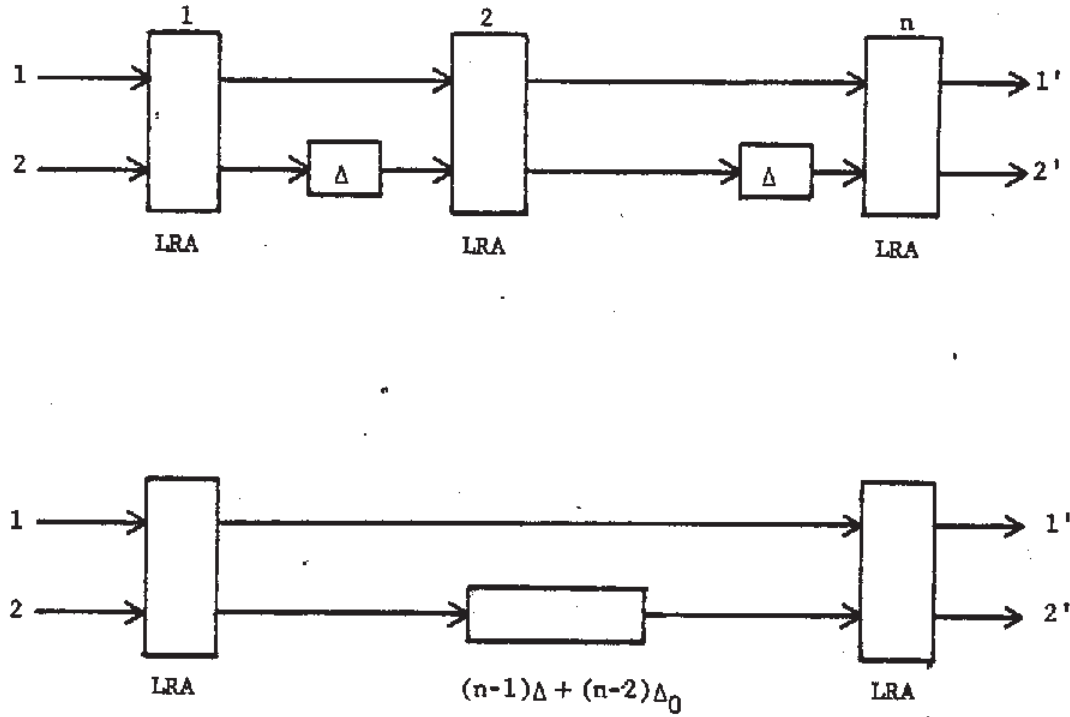
Figure 25.   Comparison of the multistage arbiter and the two-stage arbiter.

Arbiters Without Time Bounds

Asynchronous speed independent circuits can tolerate arbiters that may not necessarily act within a fixed length of time provided the arbitration is performed correctly, i.e. the arbiter does not fail to block one input signal. Since the ability to construct a synchronizer without a time bound is seriously in doubt, one might question whether a perfect arbiter without any time bound (asynchronous arbiter) can be realized. The situation here is, however, slightly different, and it is indeed possible to physically realize a perfect arbiter without a time bound. Such arbiters have been constructed by the researchers at the University of Washington at St. Louis [7,11,5]; the principles on which these constructions are based are similar to those explained in a patent on real time detection of flip-flop resolution by Adams [1]. These constructions employ high and low threshold elements to detect the meta-stable state and the outputs of the arbiter are prevented from changing so long as the meta-stable state persists.

A circuit for an asynchronous arbiter employing these principles is shown in Figure 26. This circuit has a set-reset flip-flop for each input to the arbiter. Initially, the admit signal is at level 1, and a signal of 1 on any of the inputs can set the associated flip-flop. If none of the flip-flops enter into the meta-stable state, the output associated with the input of the highest priority changes to 1. We will next discuss the operation of the arbiter under the critical condition.

Imagine a situation in which input 2 changes to 1 first and suppose that while admit signal is changing to 0 to block the input gates, input 1 also changes to 1. The resulting conflict may cause the flip-flop associated with input 1 to enter into the meta-stable state. The output gates remain blocked at this time because of the delay element in the path of $\overline{\text{wait}}$ signal which keeps $\overline{\text{wait}}$ at level 0 long enough for the meta-stable state detection circuitry consisting of threshold gates H and L
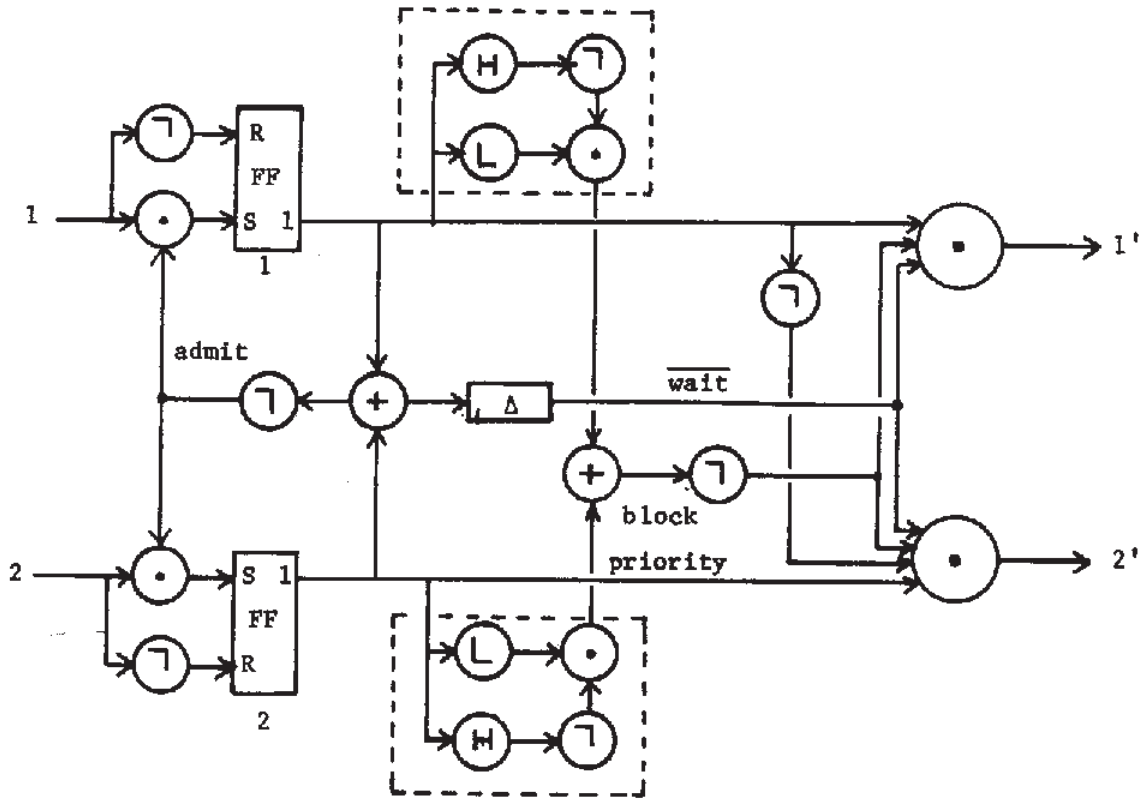
Figure 26. A circuit for asynchronous arbiter based on the detection of the meta-stable state.

to produce the block signal. The detection circuitry for flip-flop 1 produces output of 1 because the high and low threshold gates produce outputs 0 and 1, respectively because the level at the meta-stable state is in between the high and low thresholds. The signal from the detector keeps the output gates blocked as long as the meta-stable state persists. When the flip-flop leaves the meta-stable state there are two possible outcomes depending on whether the meta-stable changes to state 0 or state 1. In the first case, the block signal is removed and output 2' changes to 1, and in the other case the priority signal changes to 0, reflecting the fact that flip-flop 1 has higher priority, output 2' is blocked, and the level at output 1 changes to 1.

We now present another circuit realization of an asynchronous arbiter (Figure 27) which is considerably simpler. The circuit consists of one flip-flop and two threshold elements. The flip-flop is constructed of two NAND gates and the threshold elements are NOT gates whose threshold is adjusted lower than the level at the meta-stable state.

Briefly the operation of the circuit is explained as follows: initially both inputs are at level 0. In this condition the circuit consisting of the two NAND gates (the flip-flop) is not in its bistable condition as both outputs of the flip-flop circuit are forced to be at level 1. In the conventional use of a flip-flop this condition is avoided, but here it is used to a great advantage. When either signal at the input changes to 1, the circuit will become a bistable circuit. If only one input changes to 1, the circuit settles into the proper stable state without any hesitation. But if both inputs change simultaneously, the circuit may enter into the meta-sable state. In this case the levels at the output of both NAND gates of the flip-flop circuit come down to some intermediate value from 1 and stay there until the circuit comes out of the meta-stable state at which point one of them goes all the way to 0 and the other returns to 1. The threshold of one of the
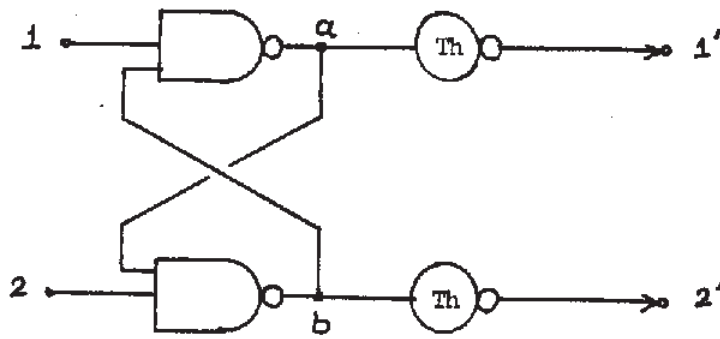
a)   threshold adjusted low
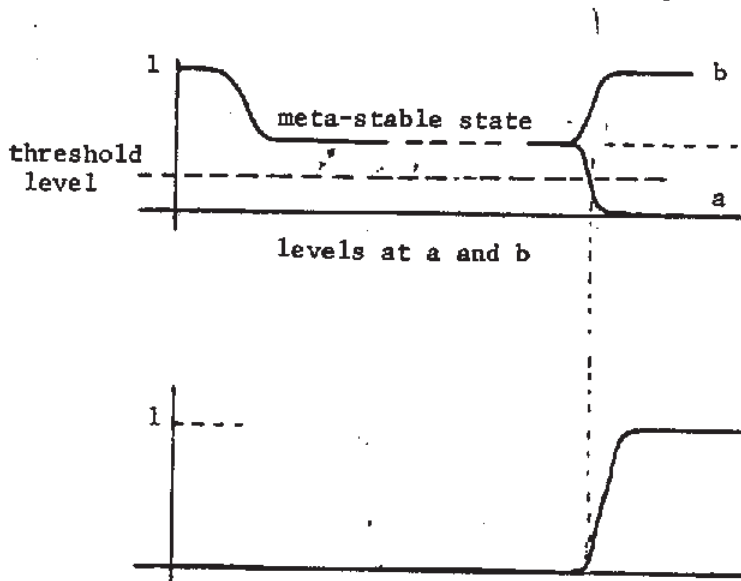
b)   the output

Figure 27. The realization of a perfect asynchronous arbiter.

NOT gates is then crossed, and the corresponding output of the arbiter becomes 1. The process of arbitration is thus completed.

The proper operation of the above circuit requires that the bistable circuit must not experience large oscillations either when it enters or when it leaves the meta-stable state. Otherwise the threshold elements will not be successful in performing their task of blocking the output. Some but not all practical circuits have been observed to have such smooth entry and exit into and out of the meta-stable state. Furthermore the choice of the threshold levels must allow for variations in the meta-stable state level arising out of such factors as variations in temperature and the device characteristics. These variations can in some cases be so large as to leave no margin for the selection of the fixed threshold level (Chaney [5]). The problem due to the variations in the meta-stable state level and the oscillations can be eliminated by a difference amplifier scheme which operates on the difference in the two output lives of the flip-flop instead of their absolute values [11, 3]. The above discussion equally applies to the circuits for the synchronizers and the blocking elements discussed earlier.

Figure 28 shows the modified circuit for the asynchronous arbiter using difference amplifiers which now play the role of the threshold NOT gates. Output is produced by a difference amplifier when the outputs of the flip-flop differ by an amount larger than the level set by the offset voltage. It may be noted that both in the initial condition and the meta-stable state the outputs of the flip-flop have the same level (except for a small variation due to the differences in the characteristics of the two NAND gates that constitute the flip-flop), but when the flip-flop comes out of the meta-stable state, the difference assumes its full value as one side goes to the low level and the other to the high level.
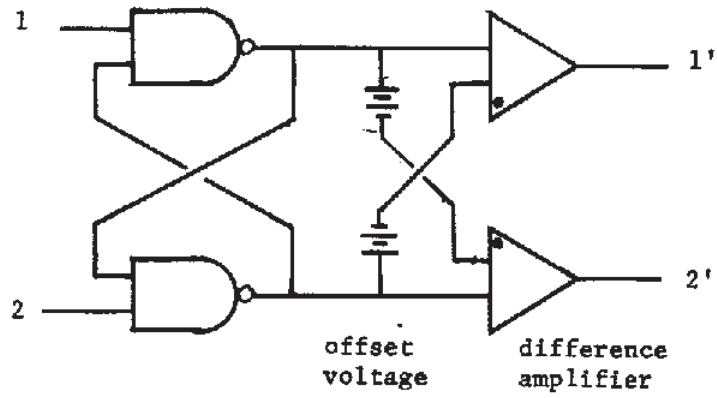
Figure 28. The modified circuit for asyn-
chronous arbiter.

Conclusion

In conclusion we note that the synchronizers and arbiters are closely related. The ability to perform synchronization implies the ability to realize an arbiter with a time bound. Furthermore, the ability to perform synchronization without the constraint that it must be performed within a bounded number of clock periods also implies the ability to realize an arbiter with a time bound because the synchronizer with a time bound can be realized from a synchronizer without a time bound. Since the ability to physically realize a (perfect) arbiter with a time bound is seriously in doubt, neither the bounded delay synchronizer nor the unbounded delay synchronizer seems to be physically realizable. On the other hand the unbounded delay arbiters can be easily realized with electronic circuits; the ability to realize an unbounded delay arbiter does not imply the ability to realize a bounded delay arbiter.

Both synchronizers and arbiters have interesting solutions in iterative structures. The iterative structures allow some additional characteristics of a flip-flop to contribute towards improving the performance of the synchronizers and arbiters in which they are used. The parameters of a flip-flop that have been identified permit a direct comparison of the quality of the flip-flops as regards their use in synchronizers and arbiters. The figure of merit, for example, completely determines whether a flip-flop is suitable for use in the iterative structure.

In LSI circuits, the constraint that all components must be on the same integral substrate and that all components must be realized from one integral process, could restrict the freedom to control some characteristics of a flip-flop and leave some characteristics free to be controlled as desired. Iterative structures could be very useful in this situation if some of the parameters which independently contribute towards the performance are free to be controlled.

It is hoped that this paper will give the readers a better understanding
of the synchronizers and arbiters and help the designers of digital systems in
realizing more reliable implementations of systems when the problems of syn-
chronization and arbitration must be faced.

References

1. Adams, R. L., D. R. Castaldo, and G. W. Kurz. Real-time detection of latch
   resolution using threshold means. U.S. patent 3,515,998, June 2, 1970.

2. Catt, I. Time loss through gating of asynchronous logic signal pulse.
   IEEE Trans. on Electronic Computers, Vol. EC-15, No. 1 (February 1966), pp 108-111.

3. Chaney, T. J., S. M. Ornstein, and W. M. Littlefield. Beware the synchronizer.
   Proceedings of the Sixth Annual IEEE Computer Society International Conference,
   San Francisco, California, September 1972.

4. Chaney, T. J., and C. E. Molnar. Anomalous behavior of synchronizer and arbiter
   circuits. IEEE Trans. on Computers, Vol. C-22, No. 4 (April 1973), pp 421-422.

5. Chaney, T. J. Private communication. Computer Systems Laboratory, Washington
   University, St. Louis, Missouri.

6. Couranz, G. R. An Analysis of Binary Circuits Under Marginal Triggering Con-
   dition. Technical Report 15, Computer Systems Laboratory, Washington University,
   St. Louis, Missouri, November 1969.

7. Couranz, G. R. An Unclocked Two Line Interlock. Technical Memorandum 55,
   Computer Systems Laboratory, Washington University, St. Louis, Missouri,
   February 1968.

8. Gray, H. Digital Computer Engineering. Prentice-Hall, Inc., Englewood Cliffs,
   New Jersey 1963, pp 198-201.

9. Kirk, C. Private communication. Lincoln Laboratory, Massachusetts Institute of
   Technology, Cambridge, Mass.

10. Littlefield, W. and T. J. Chaney. The Glitch Phenomenon. Technical Memorandum 10,
    Computer Systems Laboratory, Washington University, St. Louis Missouri,
    December 1966.

11. Littlefield, W. M. _Interlocken_. Technical Memorandum 26, Computer Systems Laboratory, Washington University, St. Louis, Missouri, June 1967.

12. Patil, S. S., and J. B. Dennis. Description and realization of digital systems. _Proceedings of the Sixth Annual IEEE Computer Society International Conference_, San Francisco, California, September 1972.

13. Plummer, W. W. Asynchronous arbiters. _IEEE Trans. on Computers_, Vol. C-21, _No_. 1 (January 1972), pp 37-42.

14. Seitz, C. Private communication. University of Utah, Salt Lake City, Utah.

15. Science and the citizen. _Scientific American_, April 1973, pp 43-44.

16. Ware, W. H. _Digital Computer Technology and Design_, Vol. II. John Wiley, New York 1963, pp 13-18.