Bounded and Unbounded Delay Synchronizers and Arbiters

Computation Structures Group Memo 103
June 1974

Suhas S. Patil

This research was supported in part by the National Science Foundation under research grant GJ-34671.

Abstract

Synchronizer and arbiter malfunction can cause serious unpredictable malfunction of digital systems. It is believed that a perfect synchronizer cannot be physically realized. This view is reenforced by the close relationship among the synchronizers of bounded delay, synchronizers of unbounded delay and arbiter of bounded delay shown in this paper. If either the synchronizer or the bounded delay arbiter can be realized, the other can also be realized. Surprisingly the ability to realize the unbounded delay synchronizer implies the ability to realize the bounded delay synchronizer. Synchronizers and arbiters operate under fundamentally different conditions compared to the other elements because they encounter conflicts while other elements in a digital system can be assumed not to encounter any conflicts. Conflicts can lead to uncertain delays which can cause malfunction of the system. The performance of synchronizers and arbiters can be measured by two parameters: an effective error window and a time delay, both of which can be derived from the parameters characterizing the behavior of a flip-flop. Even though there is difficulty with realization of synchronizers, the asynchronous arbiter can be easily realized. A very efficient realization of asynchronous arbiters is also presented in this paper.

Introduction

The problem of synchronization arises when independently operating synchronous systems communicate with each other. The possibility that the signals received at a system may not be in proper synchronization with the internal clock of that system creates serious difficulties in ensuring correct operation of the system. Unless the input is synchronized to the clock before it is used in the system, the input as received by the system may be quite different from what was sent to it. This difficulty, explained in greater detail in the following section, implies that the means to properly synchronize signals is essential to correct realization of synchronous digital systems.

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 possible to realize a synchronizer of arbitrarily low probability of malfunction, but it is doubtful if a perfect synchronizer can be physically realized. Moreover, the lower probability of failure comes at the expense of increased delay of the device.

The problem of arbitration 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 of a flip-flop which is always present in addition to the two stable states. Under certain critical conditions, this unstable equilibrium state, called a meta-stable state, leads to abnormal delays which can cause a synchronizer or an arbiter to malfunction.

The objective of this paper is to explain the problems of synchronization and arbitration and to show how they relate to each other and to the behavior of a flip-flop under critical operation. Parameters that characterize synchronizers and arbiters are identified together with their relationships to the parameters of a flip-flop under critical operation.

The requirements of arbitration for synchronous and asynchronous systems are not exactly the same. The differences are critical to the extent that even though it seems impossible to physically realize the arbiter for synchronous systems, the type of arbiter needed in asynchronous systems can be easily realized with electronic gates. A new very simple and efficient circuit for asynchronous arbiters is presented in this paper.

Necessity of Synchronization

We will now explain how the lack of synchronization seriously affects the ability of a synchronous system in receiving input from outside. Figure 1 illustrates a situation in which system S_2 receives data from system S_1 over an input/output interface which consists of a set of data wires and a control wire. System S_1 sends data to system S_2 by sending a signal on the control wire after it has placed the data on the data wires.

Since the two systems are independent, the signal from S_1 may reach system S_2 at the same time as the clock pulse, in which case the gating pulse to the flip-flops of the register which is to record the input may be a degenerate pulse. Since the flip-flops may not have the same threshold for switching, the degenerate pulse will succeed in setting some flip-flops and not others, and thus cause a datum to be placed in the register which is quite different from what was sent to the system.

One might suggest that a careful control of the threshold of the flip-flops might solve this problem. But, as we shall see, even if the thresholds are the same, a degenerate pulse may cause the flip-flop to enter into a meta-stable state, and because there is uncertainty about which final state the flip-flop will go from the meta-stable state, we see that any equalization of threshold will not solve the problem. Therefore, for correct communication between synchronous systems it is essential to synchronize the control signal to the time frame (clock) of the system receiving the input.

Synchronizers

The schematic diagram of a synchronizer is shown in Figure 2. 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 3. 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.

Arbiters

An arbiter is a two-input two-output device whose schematic diagram is shown in Figure 4 [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 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 _arbiter with a time bound_, or simply a _synchronous arbiter_. 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 5. The figure shows a function unit f which can be connected 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 connection 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 function 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.

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 connection 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 function 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.

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 6 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 7 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).

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 8.

A single edge synchronizer can be realized from the arbiter with a time bound as shown in Figure 9. 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 intervals defined by the inverted clock pulse. The second arbiter is required only if there is a possibility that on some occasions signals may take less time to get through the arbiter than at other occasions. If the arbiter always takes the same amount of time, the second arbiter can be replaced by a delay element of appropriate value.

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 is no difficulty in realizing the C-element used in such synchronization because the synchronizer does not encounter any conflict as it 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 10). 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 intermediate 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 11). 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 intersection of the transfer function and the $y = x$ line in Figure 12. In that figure, 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-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 3) 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 13). 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 14). 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 15). 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.

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 16)[2,4,6,8,9,14]. 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 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</u> <u>conflict</u> <u>window</u> W_c which is equal to the area of the above plot of probability against time is sufficient for our purposes (Figure 17). 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 18). 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 $\frac{n}{\xi} \times W_c \times e^{(\Delta-\Delta_0)/\tau}$.

Performance of Synchronizers

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 19. 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 Δ - $(1/2)W_c$ 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 Δ - $(1/2)W_c$ 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 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 $\frac{2\tau}{W_c} \times e^{(\Delta-\Delta_0)/\tau}$.

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 if the bounded delay synchronizer cannot be physically realized.

The fact that a synchronizer without a time bound may take arbitrary number of clock periods when there is a conflict but only a bounded number of clock periods when there is no conflict is used in realizing the bounded delay synchronizer. We need only show that a single edge bounded delay synchronizer can be realized from the unbounded delay synchronizer because the double edge synchronizer can be realized from single edge synchronizers as shown earlier in Figure 7. The arrangement to realize a bounded delay synchronizer from two unbounded delay single edge synchronizers which synchronize the 0 to 1 transitions and reset themselves on 1 to 0 transitions is illustrated in Figure 20. The signal to be synchronized is sent directly to one synchronizer and through a delay larger than the absolute conflict window to the other so that at most one of the two synchronizers may experience conflict. The one which does not experience any conflict operates in normal time and the one which experiences a conflict 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.

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 1 to 0 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.

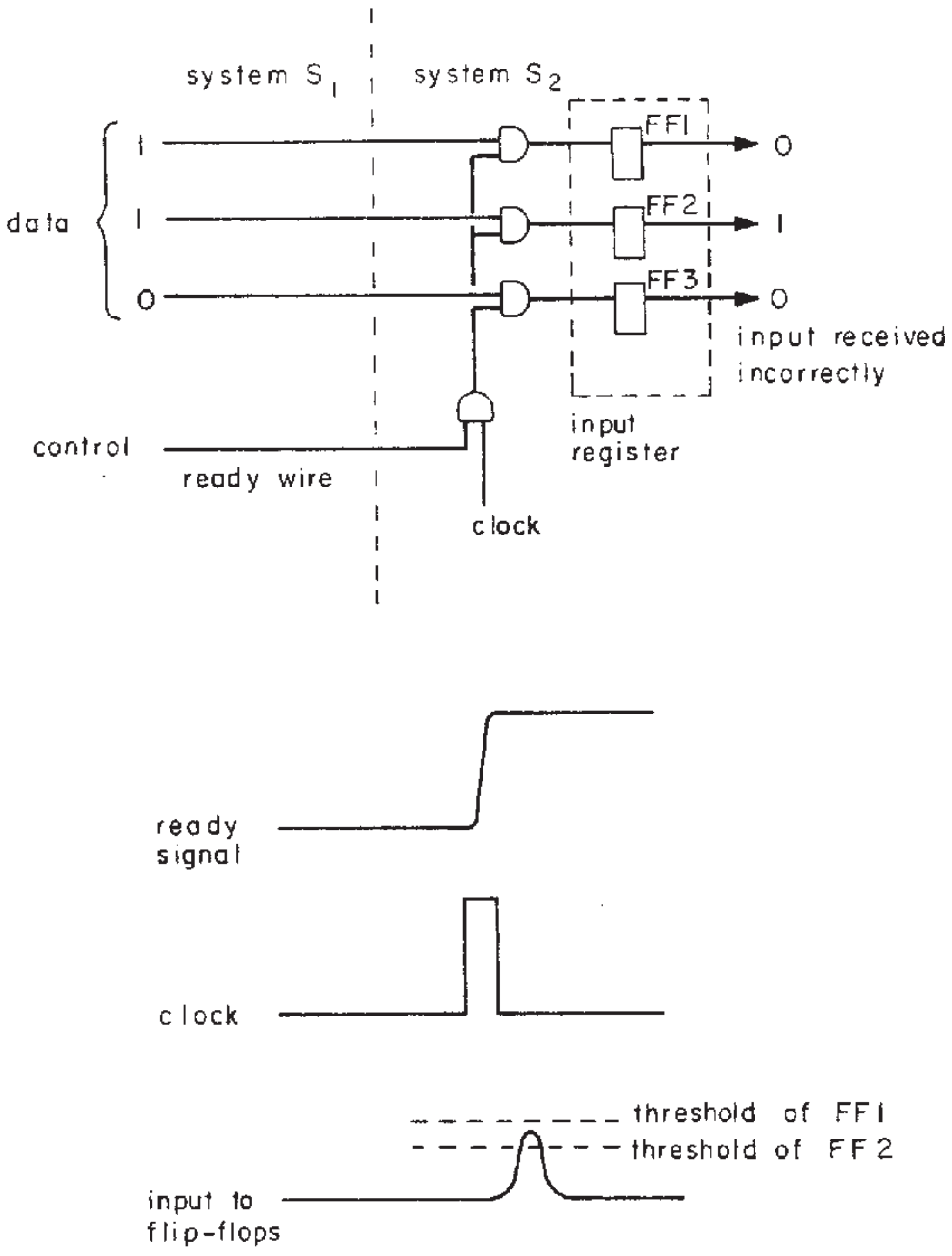


Figure 1. Lack of synchronization causes faulty communication.

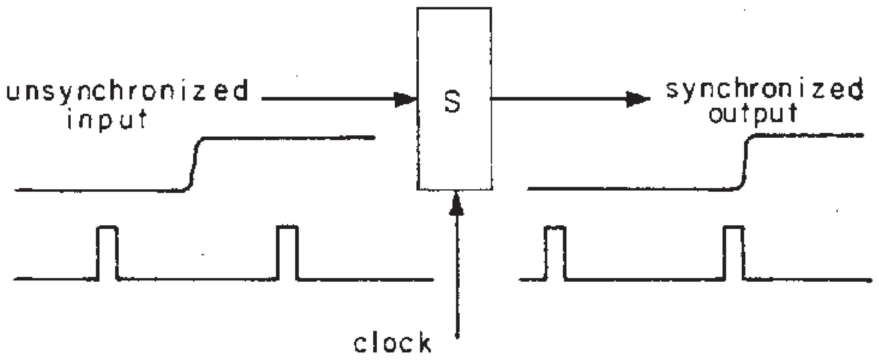


Figure 2. A schematic diagram of a synchronizer.

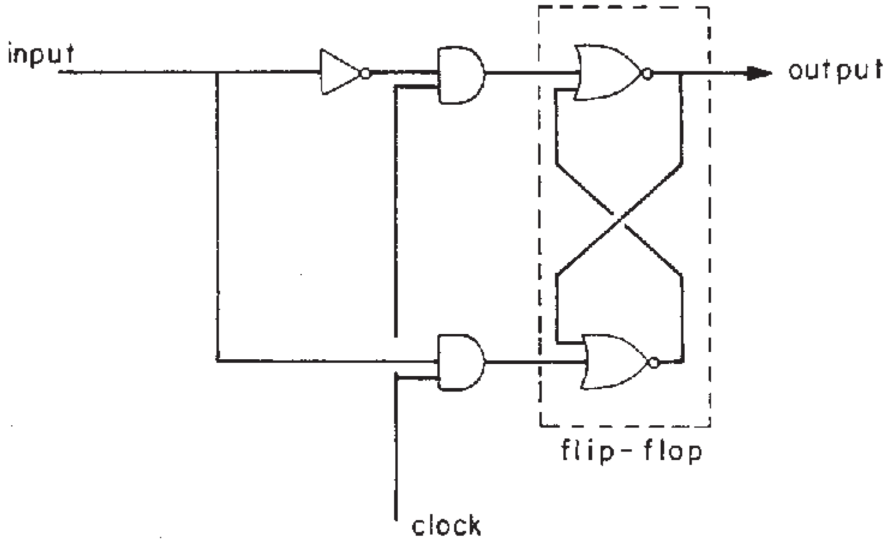


Figure 3. A circuit for an (imperfect) synchronizer.

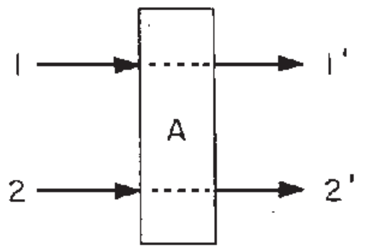


Figure 4. An arbiter.

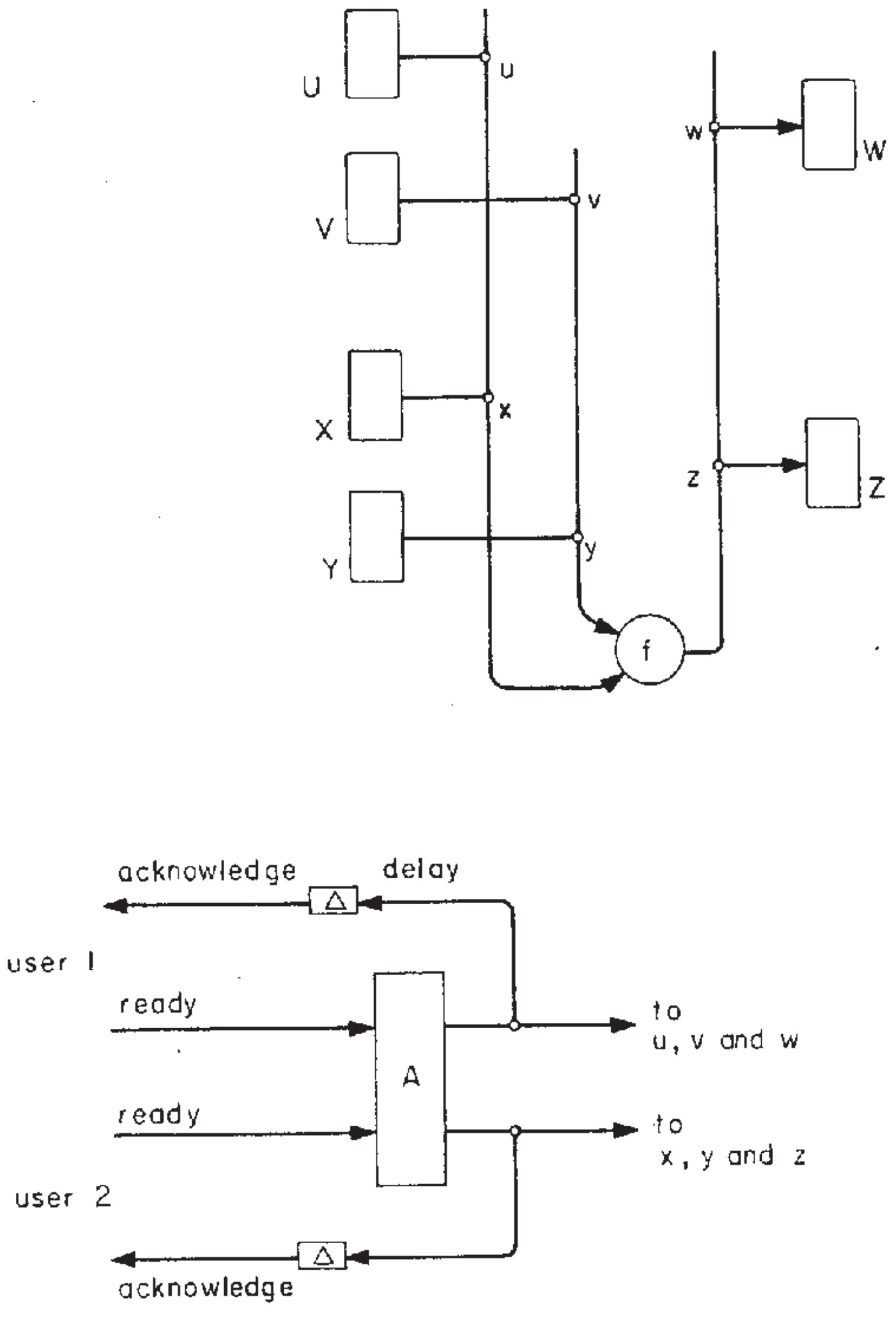


Figure 5. Resource sharing with arbiter.

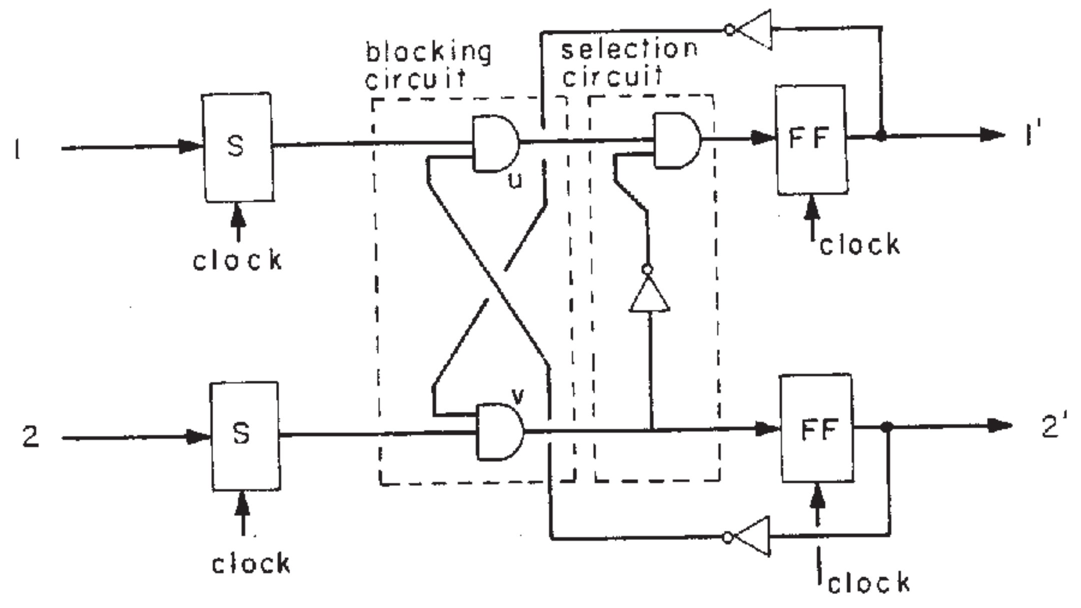


Figure 6. A bounded delay arbiter realized from synchronizers.

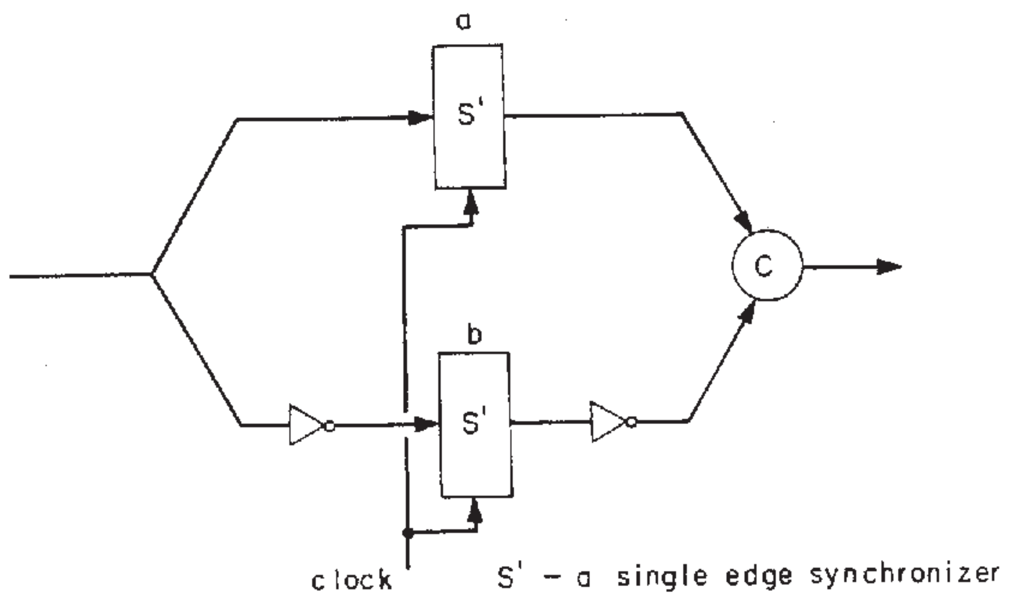
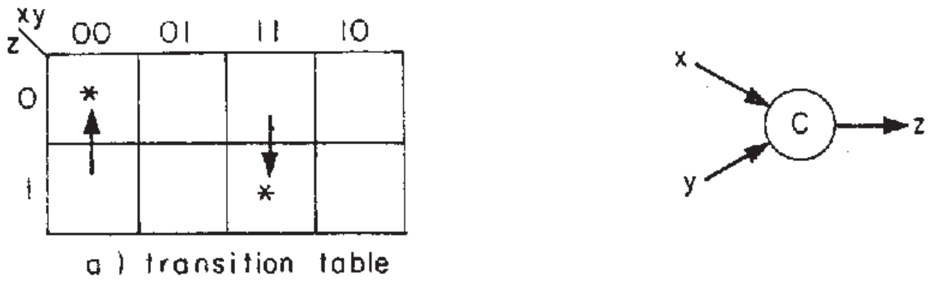


Figure 7. A double edge synchronizer from single edge synchronizers.



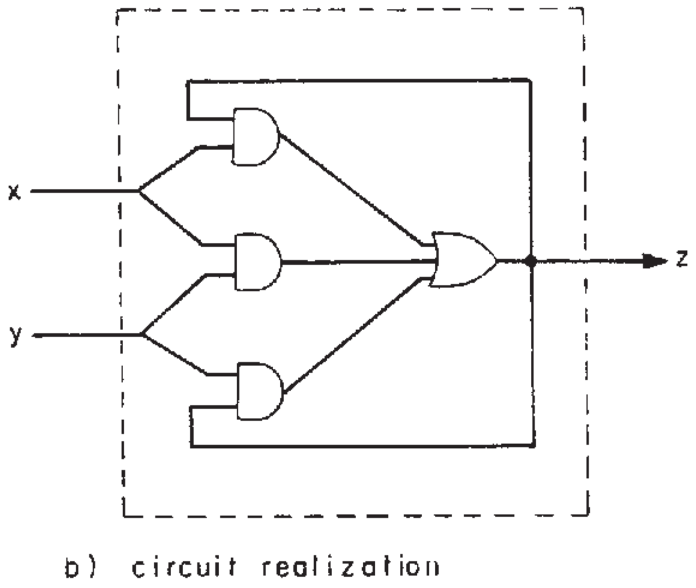


Figure 8. Muller's C element.

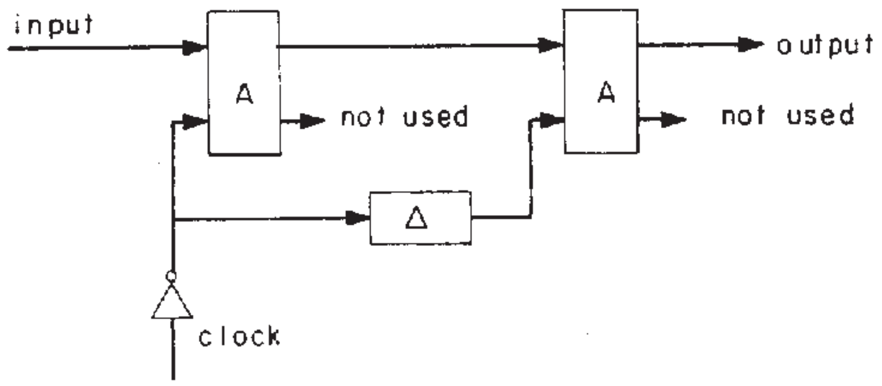


Figure 9. Realization of a positive edge synchronizer from arbiters of bounded delays.

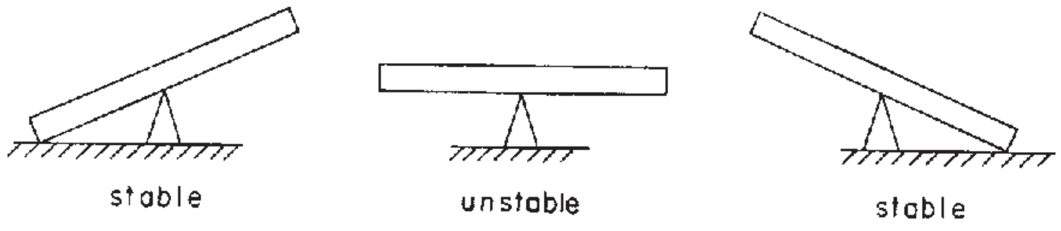


Figure 10. Three states of equilibrium of a seesaw.

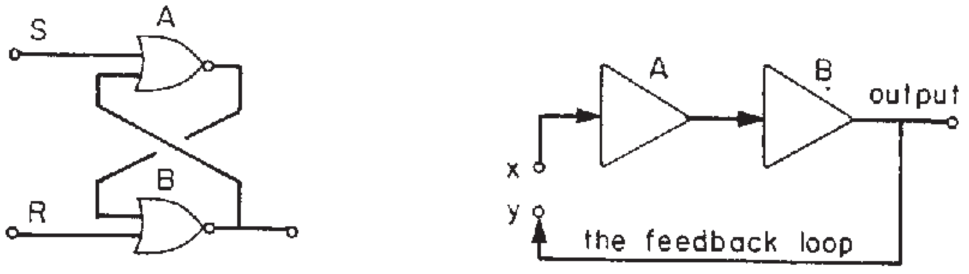


Figure 11. A flip-flop.

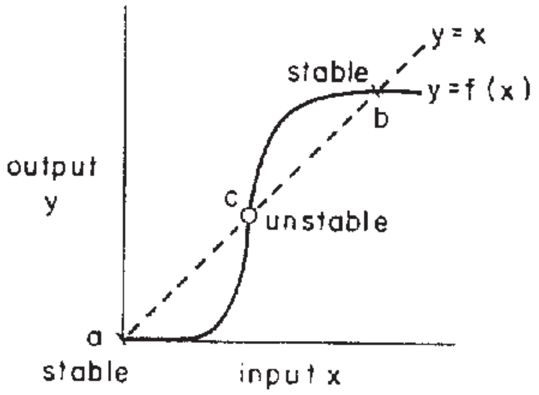


Figure 12. The unstable state.

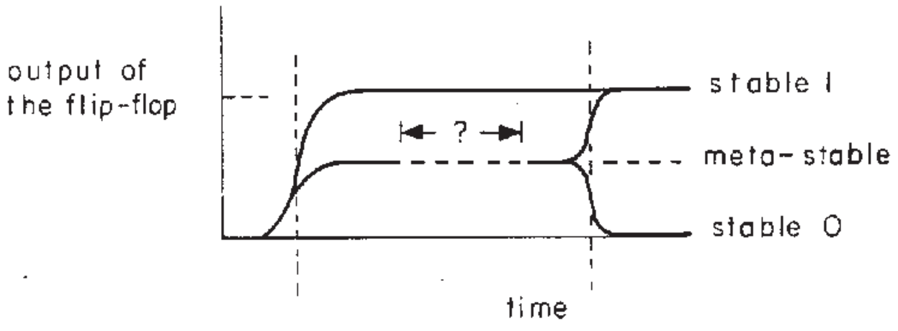


Figure 13. The possible output of the flip-flop.

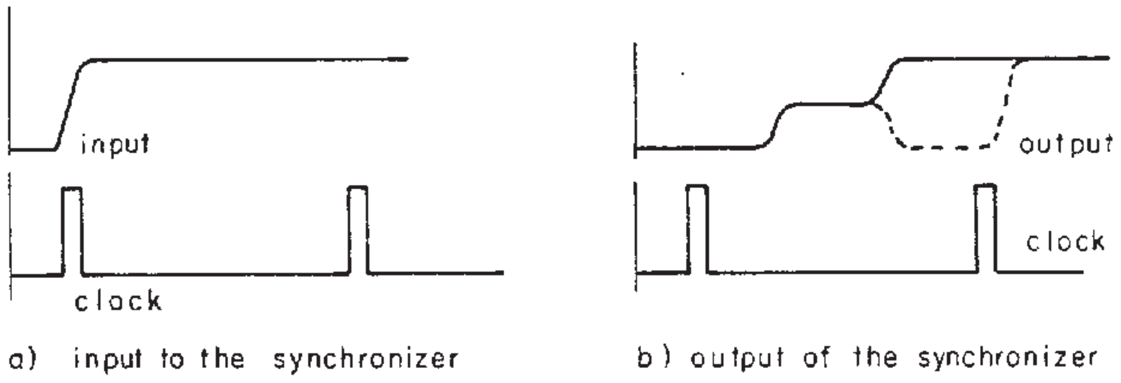


Figure 14. Possible output of the synchronizer under critical operation.

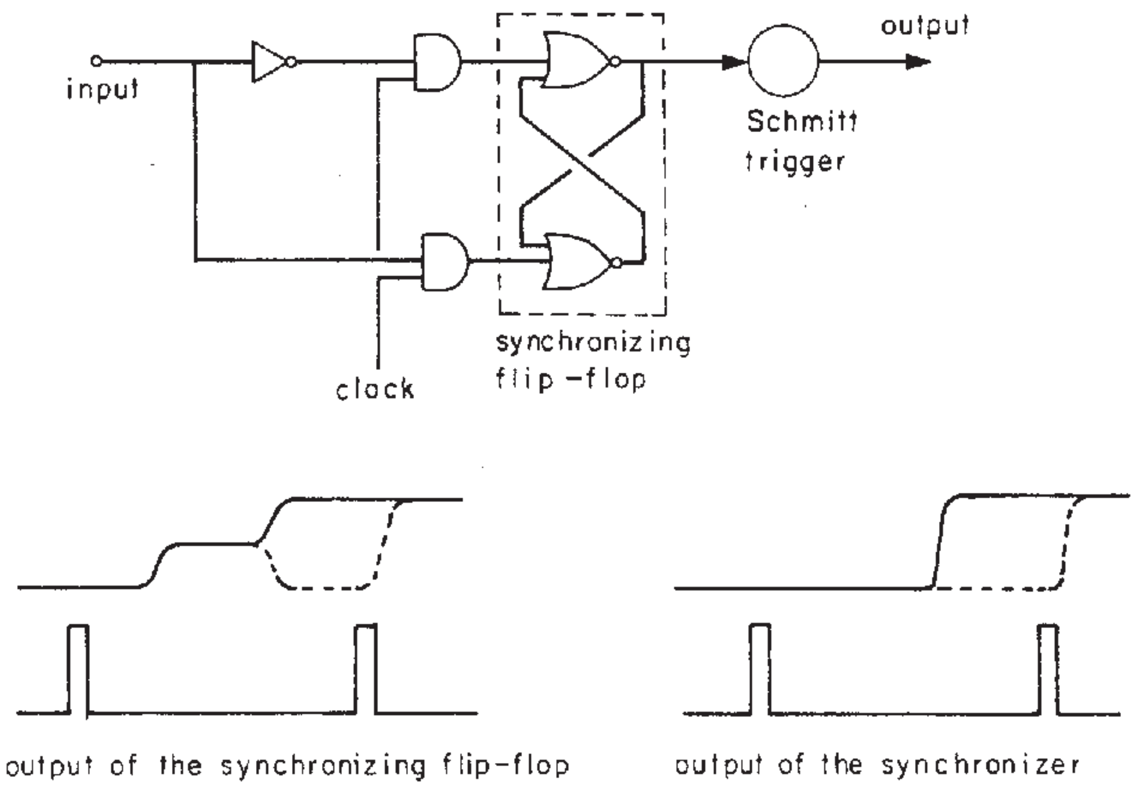

input
output
Schmitt trigger
synchronizing flip-flop
clock
output of the synchronizing flip-flop
output of the synchronizer

Figure 15. Masking the intermediate output of the synchronizer.

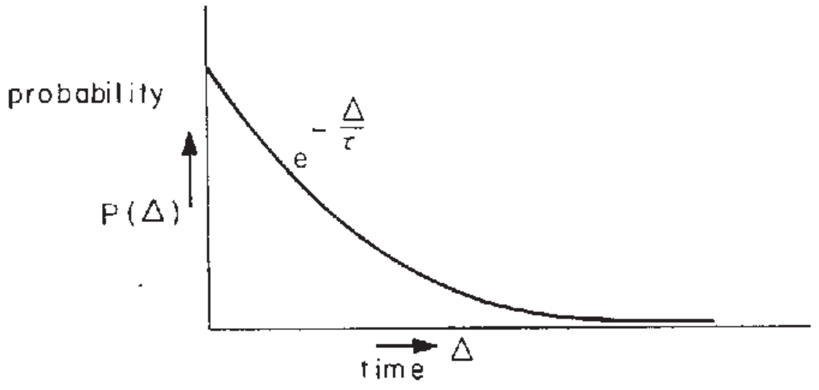


Figure 16. The probability of the flip-flop being in the meta-stable state.

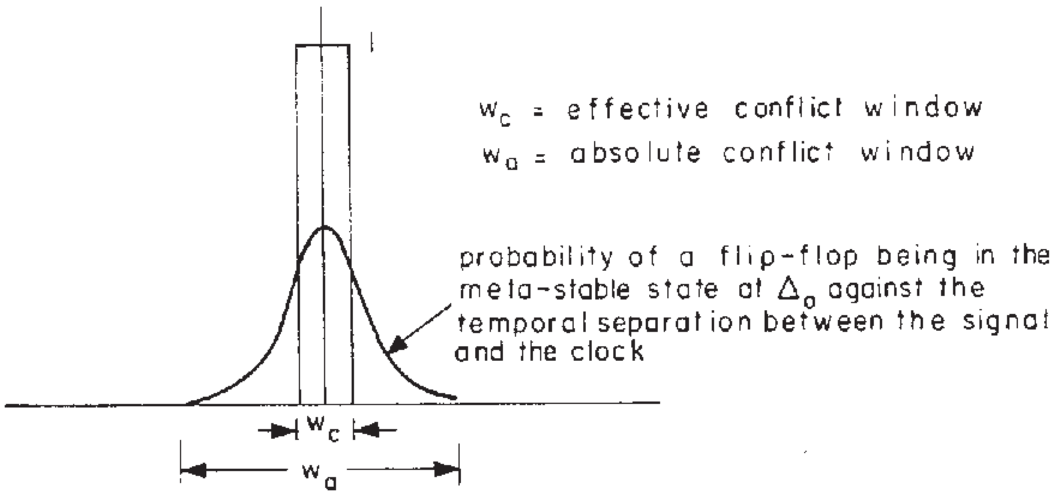

w_c = effective conflict window
w_a = absolute conflict window
probability of a flip-flop being in the meta-stable state at Δ_a against the temporal separation between the signal and the clock
w_c
w_a

Figure 17. The conflict window.

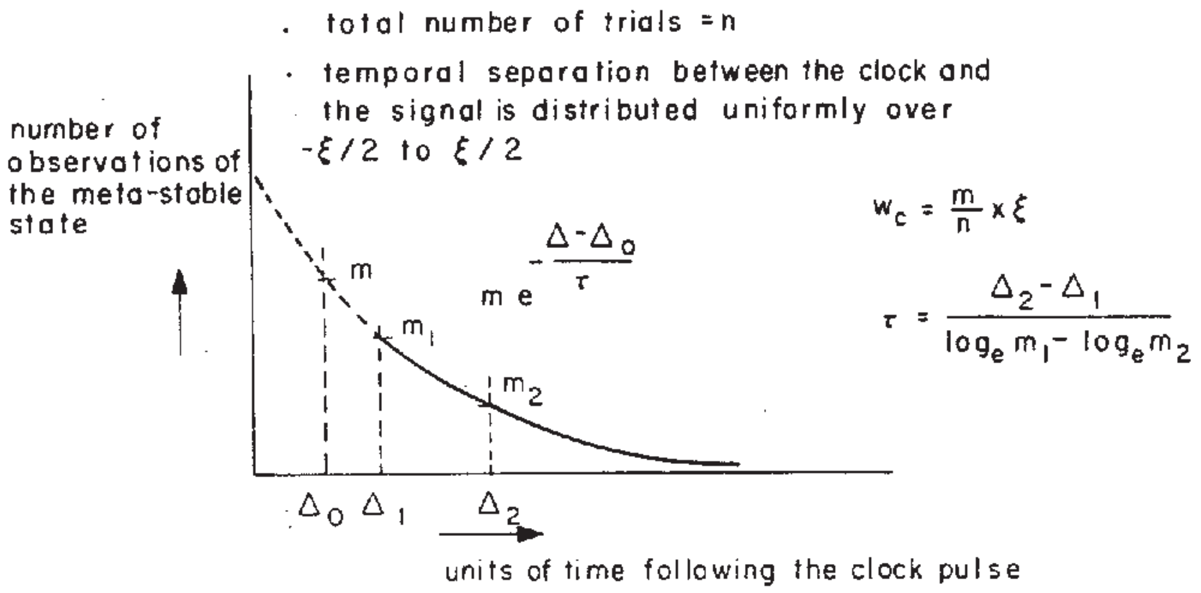


Figure 18. Determination of τ and W_c.

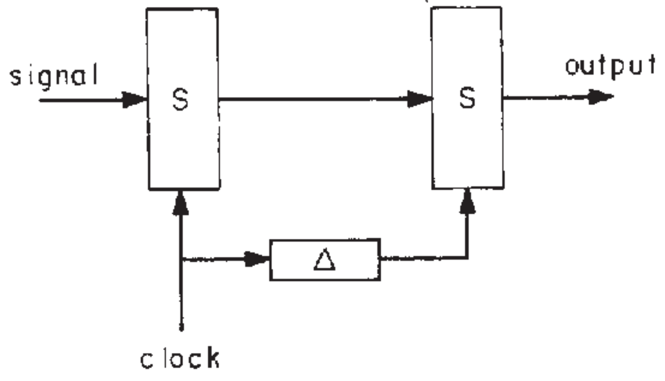


Figure 19. A two-stage synchronizer.

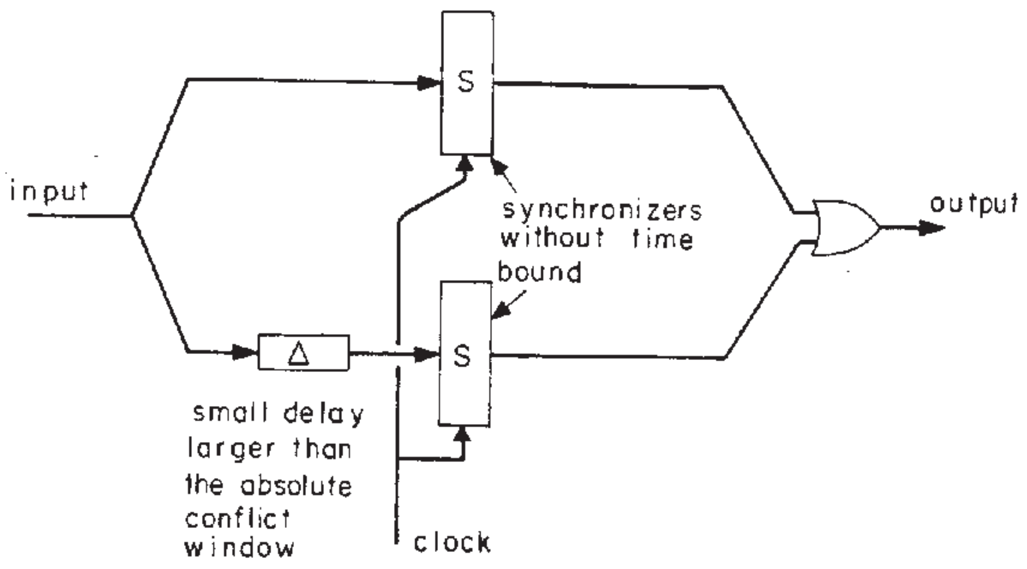

input
S
S
synchronizers without time bound
Δ
small delay larger than the absolute conflict window
clock
output

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

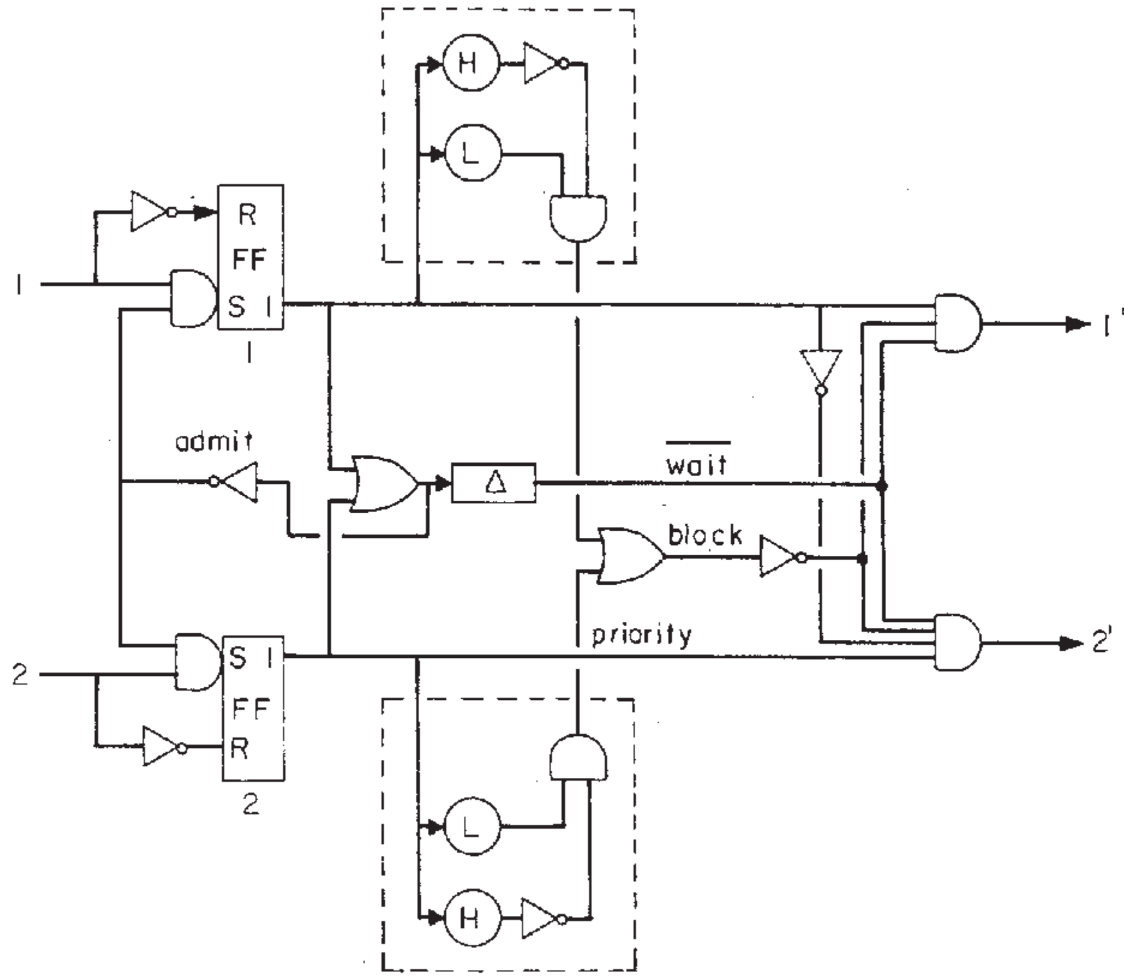
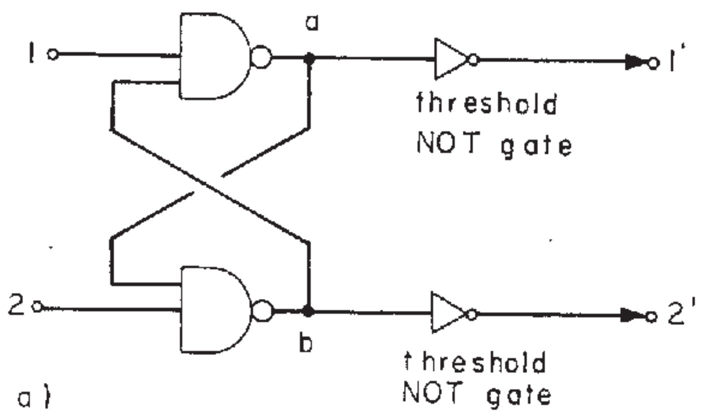


Figure 21. A circuit for asynchronous arbiter based on the detection of the meta-stable state.



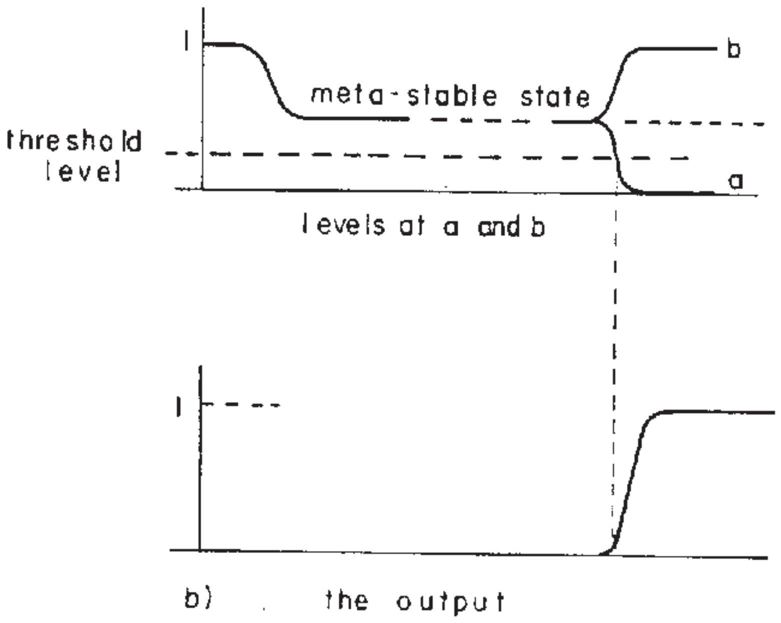


Figure 22. The realization of a perfect asynchronous arbiter.

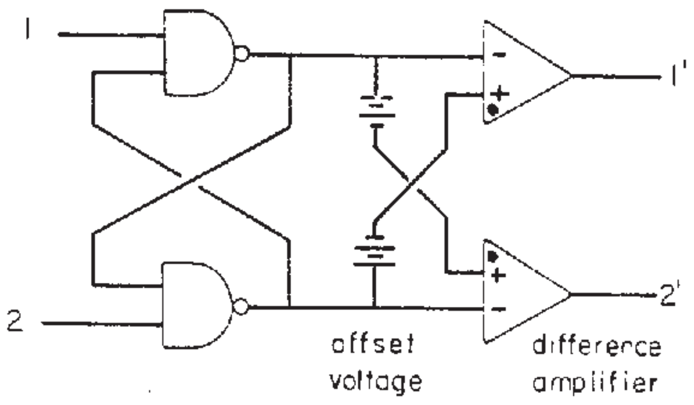


Figure 23. The modified circuit for asynchronous arbiter.

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 21. 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 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 22) 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

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.

Figure 23 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.

Conclusion

In conclusion we note that the synchronizers and arbiters are very closely related. The ability to perform synchronization implies the ability to realize an arbiter of bounded delay, and the ability to perform arbitration in a bounded length of time implies the ability to realize a perfect synchronizer. Surprisingly, even the ability to perform synchronization without the constraint that it must be performed in a bounded number of clock periods also implies the ability to realize the synchronizer with bounded delay and the arbiter with bounded delay. All this reinforces the belief that neither synchronization can be performed correctly nor can arbitration be performed in a bounded length of time. On the other hand, the unbounded delay arbiter can be easily realized with electronic gates.

The performance of synchronizers and arbiters can be characterized by two measures of performance: an effective conflict window and the propagation delay. These measures provide a means to compare a synchronizer or an arbiter against another with regard to performance in a system. These measures of performance can be derived from three parameters of a flip-flop: the effective conflict window and a time constant, which characterize the behavior of the flip-flop under critical operation, and the delay of the flip-flop measured under normal operation.

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 synchronization and arbitration must be faced.

References

1. R. L. Adams, 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. I. Catt, "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. T. J. Chaney, 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. T. J. Chaney, 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. T. J. Chaney. Private communication. Computer Systems Laboratory, Washington University, St. Louis, Missouri.
6. G. R. Couranz, "An Analysis of Binary Circuits Under Marginal Triggering Condition," Technical Report 15, Computer Systems Laboratory, Washington University, St. Louis, Missouri, November 1969.
7. G. R. Couranz, "An Unclocked Two Line Interlock," Technical Memorandum 55, Computer Systems Laboratory, Washington University, St. Louis,Missouri, February 1968.
8. H. Gray, Digital Computer Engineering. Prentice-Hall, Inc., Englewood Cliffs, New Jersey 1963, pp 198-201.
9. C. Kirk, Private communication. Lincoln Laboratory, Massachusetts Institute of Technology, Cambridge, Mass.
10. W. Littlefield and T. J. Chaney, "The Glitch Phenomenon," Technical Memorandum 10, Computer Systems Laboratory, Washington University, St. Louis Missouri, December 1966.
11. W. M. Littlefield, "Interlocken," Technical Memorandum 26, Computer Systems Laboratory, Washington University, St. Louis, Missouri, June 1967.
12. S. S. Patil 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. W. W. Plummer, "Asynchronous arbiters," IEEE Trans. on Computers, Vol. C-21, No. 1 (January 1972), pp 37-42.

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

15. Science and the citizen, Scientific American, April 1973, pp 43-44.

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