Australian Telecommunications Research Institute (ATRI)
and
Co-operative Research Centre for Broadband Telecommunications
and Networking (CRC-BTN)

Frame Synchronization Techniques and Jitter Generation: Analysis, Modelling and Enhancement

Jacqueline Walker

This thesis is presented as part of the requirements for
the award of the Degree of Doctor of Philosophy
of the
Curtin University of Technology

March 1997

Acknowledgements

Firstly, I would like to thank Professor Tony Cantoni, my supervisor. Although an extremely busy person, Tony still manages to find the time to assist students. I also appreciate his understanding towards my particular needs as a new mother.

Secondly, I would like to thank Giles for his encouragement and practical support and my sister, Amanda, for helping with Oisin.

Abstract

Synchronization means the aligning of the significant instants of one signal to the significant instants of another. In digital systems, where timing transfer between systems is required, synchronization is an important function. In this thesis new results on the performance and design of synchronization processes are presented.

An inescapable consequence of the synchronization of external autonomous inputs in digital systems is the possibility of failure of digital devices used to capture the external signal. The anomalous behaviour of these devices is referred to as metastability. The most commonly used approach to controlling the problem of metastability is the use of synchronizers. A synchronizer can be designed to reduce the probability of metastable failure but cannot eliminate it altogether. New high performance synchronizer designs are presented and analysed in this thesis.

Another consequence of synchronization is the resulting disturbance of the significant epochs of timing signals. This disturbance is referred to as jitter. The characterization of jitter produced in synchronization processes is important in the design of digital systems. In this thesis, jitter characteristics are derived for two important applications that arise in digital communications systems. The characterization provides new insight into the dependence of the jitter on system parameters.

Contents

Figures

Tables

Chapter 1
Introduction

The term synchronization refers to the aligning of the timing instants of one signal to the timing instants of another signal. In synchronous digital systems, for example, all operations within the system are synchronized to the edges of the system clock. The synchronization to the system clock provides a known constraint on the timing of events within the system which greatly simplifies the design, test and functioning of the system. The thesis presents original work on various aspects of synchronization.

In digital communications networks, recovery of the transmitted information is dependent upon accurate timing linked to the data in order to decode correctly the stream of bits. Due to the large distances frequently encountered in communications systems, the issue of synchronization becomes even more important.

Digital systems are rarely useful if they operate in total isolation. They therefore must be able to accept inputs from external autonomous sources. These external inputs will be asynchronous to the digital system clock, that is, there is no relationship between the frequency and phase of the input and the system clock. For the digital system to be able to use the input, it must be synchronized into the timing domain of the system clock.

Bistable devices are frequently used to synchronize the inputs in digital systems. The presence of an asynchronous input means that the constraints on input timing of a bistable device cannot always be observed [15], [34], [42]. As a result, a state may result in which the output of the device becomes logically undefined, for a time that is long compared to the normal switching delay of the device [16], [17], [31]. This state is referred to as a metastable state. Metastable failure of bistable devices is important not only due to the excess settling delay but also because of the possibility of propagation of an undefined logic level within the system and inconsistent interpretation of the device output by other devices in the system [34], [55].

To control the problem of metastable failure within systems, synchronizers have been adopted as an interface between the asynchronous input and the synchronous system. The synchronizer samples the asynchronous input typically at the rate of the system clock. The output of the synchronizer is thus synchronous with the rest of the system. The synchronizer is designed to reduce the probability of metastable failure to an acceptable level, but it cannot eliminate it altogether.

Digital systems, including communications systems, are susceptible to errors caused by the misidentification of the significant instants of a signal. Jitter is said to affect the signal when the significant instants of a signal vary from their ideal positions in time. Jitter may cause bit errors, unwanted phase modulation after conversion back to an analog signal [8], or corruption of data if data buffers overflow or underflow. In digital communications systems there are many potential sources of jitter: for example, jitter arising in clock recovery in repeaters, wander (low frequency jitter) arising due to variations in transmission link delays, and jitter arising as result of synchronization processes in multiplexers [8], [13].

1.1 Motivation and Applications

1.1.1 External Timing Source Injection for an ATM LAN

The Co-operative Research Centre for Broadband Telecommunications Networks (CRC-BTN) has been involved in the development of an ATM LAN (Asynchronous Transfer Mode Local Area Network), based on extensions of the DQDB (Distributed Queueing Dual Bus) protocol which is the basis of the 802.6 MAN (Metropolitan Area Network) Standard [4]. The author participated in the project and studied aspects of network timing distribution for the ATM LAN.

A key feature of the DQDB network used in the ATM LAN is the use of autonomous nodes with independent clocks [4] which prevents the build-up of bit-level jitter due to link level clock recovery [52]. The clocks in such a network will be plesiochronous, that is with different frequencies within a certain tolerance of a nominal rate. Without synchronization, the average rate of data input to a buffer within a node will not be the same as the average data output rate leading to corruption of data due to buffer overflow or underflow. The same problem arises, for example in the asynchronous time division multiplexing used in the Plesiochronous Digital Hierarchy (PDH). In the PDH, the required synchronization of the plesiochronous lower rate tributaries is accomplished locally at the telecommunications node using a timing adjustment mechanism known as pulse stuffing or justification [7], [47]. The stuffing algorithm used in the ATM LAN is based on the constrained stuffing algorithm used in the DQDB [63] and can lead to a build-up of frame level jitter.

Although the nodes in the ATM LAN have autonomous clocks, a common timing reference is still required to enable the provision of certain services such as constant bit rate (CBR) services. The timing reference is carried by the framing signal. However, the build-up of frame-level jitter, containing undesirable low frequency components, makes provision of a high quality reference difficult.

The timing reference is usually derived from an external timing source (XTS) to generate a clean frame. If the reference is supplied by a public network, there is the additional advantage that information exchange with the public network becomes possible.

To inject the XTS, the frame timing must be locked to the XTS. Conventionally, the synchronization of the frame timing would be accomplished using a justification or stuffing approach. The conventional approach is complex, requiring a buffer and control blocks to monitor stuffing. As the XTS is asynchronous to the local clock of the node, a synchronizer must still be used in the conventional implementation. A simpler method of XTS injection combines the generation of a clean frame and synchronization in a single operation. In this thesis, new high performance synchronizer designs are presented and analysed [60], [61]. The need for managing metastability is a more widespread problem and is inherent in all synchronization processes as will be evident when timing transfer over an ATM network is considered.

The first investigations of the jitter generated by synchronization processes followed the introduction of pulse stuffing synchronization for digital multiplexing in the PDH. In pulse stuffing, certain locations within the data stream carry data bits or dummy 'stuff' bits as required to vary the tributary rate [47]. An important parameter is the stuff ratio which expresses the average rate of stuffing as a proportion of the maximum rate of stuffing, usually the frame rate. The most problematic jitter component produced by pulse stuffing synchronization is waiting time jitter which can be at arbitrarily low frequency and thus cannot easily be removed by a desynchronizer [22], [47].

Kuroyanagi et al. [35] used the value of the stuff ratio in the PDH to find the dominant frequency and peak-to-peak amplitude of the waiting time jitter. They also plotted the

so-called classical characteristic of the waiting time jitter, plotting the stuff ratio vs. the peak-to-peak jitter amplitude.

The spectrum of waiting time jitter was first derived by Duttweiler [22] by applying random process theory to waiting time jitter waveforms. Duttweiler's approach made simplifying assumptions about the operation of the phase detector [19], [47]. Some more detailed analysis [19], [20] suggests that the actual jitter generated in PDH multiplexing depends on the precise frame format, the buffer size and the phase detector implementation. The resulting filtered waiting time jitter amplitude may be less than that predicted by the ideal case. The effect of phase detector implementation was also investigated experimentally by Cock et al. [21]. Other methods of deriving the spectrum exist, for example, decomposition of the waiting time jitter waveform [56] and analysis of the phase waveforms at the buffer [10], [48]. The spectrum of the jitter generated by the constrained stuffing algorithm used in the 802.6 DQDB network [4] was derived by Warner [63].

In this thesis, the jitter generation inherent in XTS injection is characterized. Two different methods of accomplishing the XTS injection, are analysed using a new time domain approach developed by the author which yields exact expressions for frame-level jitter [62].

1.1.2 Synchronous Residual Time Stamp (SRTS) Timing Transfer

The Synchronous Digital Hierarchy (SDH) and SONET (Synchronous Optical Network), will be the underlying transmission network of the Broadband Integrated Services Digital Network (B-ISDN) [11], [37]. ATM is the preferred protocol for the transport of all kinds of traffic in the new B-ISDN [44]. The AAL1 (ATM Adaptation Layer 1) performs the functions necessary to allow CBR services to be transported in ATM cells, including encoding and transporting the CBR service clock. One of the methods accepted for end-to-end timing transfer in the AAL 1 is the Synchronous Residual Time Stamp (SRTS) [5] which makes use of the presence of a common network clock at the origin and destination.

It has been suggested that the jitter which is an inherent characteristic of the SRTS method of frequency encoding, is analogous to the waiting time jitter generated by

conventional pulse stuffing [37], [46], [57]. Lau and Fleischer [37] and Uematsu and Ueda [57] use the approach of [35] to deduce the jitter amplitude. [46] uses a piecewise linear approximation to the resulting time-varying frequency of the reconstructed timing signal at the destination to find an approximation to the spectrum. However, a formal derivation of the jitter spectrum associated with SRTS method timing transfer has not been provided.

In applications where waiting time jitter occurs, the stuff ratio is chosen so that the peak-to-peak amplitude of the waiting time jitter is small, for example, the telecommunications multiplexing hierarchies have been designed with values of the stuff ratio in the range (0.41, 0.44) [7].

In this thesis, the jitter characteristics of the SRTS method are derived. Original analysis by the author is used to determine whether it is possible to specify parameters of the SRTS method so as to reduce the amplitude of the waiting time jitter.

1.2 Contributions

Within the fields of synchronization, jitter analysis and metastability, the author has made the following contributions which are reported in this thesis:

(i) The development and analysis of three new synchronizer designs and the determination of their metastable failure performance.

(ii) A new time domain approach to jitter analysis, used to find exact expressions for frame-level jitter in a conventional approach to external timing source injection.

(iii)The analysis of a new method of external timing source injection showing that the jitter spectrum is equivalent to waiting time jitter. The new time domain approach to jitter analysis has also been applied to the SRTS method establishing formally that waiting time jitter is an inherent characteristic of this method of timing transfer.

(iv)A new, more direct and easily verifiable approach to the regeneration at the destination of the SRTS timing signal has been developed, analysed and simulated.

(v) It has been shown that it is not possible, in general, to choose the equivalent of the stuff ratio in the SRTS method in order to minimise the amplitude of the waiting time jitter generated in this method.

(vi)Extension of the new time domain approach to include analysis of the jitter spectrum in the presence of input jitter.

(vii)The jitter model developed has been experimentally verified, showing that the jitter spectrum conforms to the predictions, both with and without input jitter.

Much of the above work has also been reported by the author [60]-[62] as listed below:

J. Walker and A. Cantoni, "A New Design for a Frame Sampling Synchronizer", in *Proc. IEEE Int. Symp. Circuits and Systems*, London, vol. 3, pp. 97-100, June 1994.

J. Walker and A. Cantoni, "A New Synchronizer Design" *IEEE Trans. Comput.*, vol. 45, no. 11, pp.1308-1311, Nov. 1996.

J. Walker and A. Cantoni, "Jitter Analysis for Two Methods of Synchronization for External Timing Injection", *IEEE Trans. Commun.*, vol. 44, no. 2, pp. 269-276, Feb. 1996.

1.3 Outline of the Thesis

In Chapter 2, a brief overview of the problem of metastable failure is given including a summary of some useful models and metastable failure probability equations. Synchronizer design is the main focus of this chapter. Three new synchronizer designs are presented and their performance compared with that of existing synchronizer designs.

In Chapter 3, an introduction to jitter is first given, covering a range of topics in jitter modelling and analysis which are of relevance to later chapters. The analysis of the jitter equation of the synchronization process is then presented and techniques for Fourier analysis of jitter are applied to derive the jitter spectrum of the synchronization process. Finally, the analysis of the jitter equation of the

synchronization process is extended to determine the jitter spectrum in the presence of input jitter.

One application of synchronization is external timing source injection. Chapter 4, begins with an introduction to timing adjustment, jitter and the use of external timing sources in networks. Two different approaches to external timing source injection are then studied and the spectrum of the jitter generated by each is determined. Finally, the results of the jitter analyses for the two approaches are compared and shown to be equivalent to waiting time jitter. Issues relating to waiting time jitter and how the choice of the stuff ratio can minimise the waiting time jitter amplitude are then reviewed.

A second application of synchronization is the Synchronous Residual Time Stamp (SRTS) method of end-to-end timing transfer. Chapter 5 begins with a detailed introduction to the SRTS method, including presentation of a new method for destination timing signal recovery. A model of the SRTS method is developed and the spectrum of the jitter inherent to the SRTS method is derived, showing that it is equivalent to waiting time jitter. Finally, the question of whether it is possible to specify parameters of the SRTS method so as to reduce the amplitude of the waiting time jitter is studied in detail.

In Chapter 6, an experimental investigation into the jitter spectrum of the synchronization process is described. A brief description of the experimental approach is given and then the experimental results are compared with the theoretical predictions of the jitter spectrum both with and without input jitter. Finally, the question of whether sinusoidal input jitter might be useful in the suppression of unwanted waiting time jitter is briefly examined.

In Chapter 7, suggestions for further work are made.

Each chapter is followed by a number of appendices, the first of which contains a Glossary relevant to that chapter. Subsequent appendices contain the derivations of the key results found in the chapter.

Chapter 2
Metastability

2.1 Introduction

Systems which have asynchronous inputs will inevitably suffer from problems due to the phenomenon of metastability [15], [34], [42]. Any design for a synchronizer should take into account the possibility of metastability and be designed to cap the probability of metastable failure to an acceptable level.

This chapter presents new synchronizer designs and compares their performance to existing synchronizers. In Section 2.2, a brief overview of the problem of metastable failure is given including a summary of some useful models and metastable failure probability equations. In Section 2.3, three new synchronizer designs are presented [60], [61] which are shown to have significant advantages over previous designs.

2.2 The Problem of Metastable Failure

Bistable devices are frequently used to synchronize the inputs in digital systems. For correct operation, a bistable device such as a latch or a flip-flop has certain constraints on the timing of its inputs. For example, the input of a D-type flip-flop must be established for a certain minimum time before the clock edge (setup time) and must be held for a certain minimum time after the clock edge (hold time). Similar constraints on minimum pulse width and minimum pulse separation apply to the signals applied to the R and S inputs of a RS flip-flop.

If the timing constraints are not observed, the output of the device may become logically undefined, lingering between the two stable defined states of the device. When the output of a bistable device exhibits such behaviour, the device is said to be in a metastable state [16], [17], [31]. The invalid logic level may persist for a time that is long compared to the normal switching delay of the device [16], [17]. In certain device types, the metastable state may be characterized by oscillatory behaviour [17], [31], [50] which lasts for a time long in comparison to normal logic delays.

Metastable failure of bistable devices is an important issue in digital system design, not only because of the excess settling delay. When metastable failure occurs in a device, other components of the system which use the output of the failed device may propagate an undefined level in the system or may interpret the output inconsistently which can then cause malfunctions within the system [34], [55].

It has been shown mathematically that metastability cannot be avoided when bistable digital systems have inputs which are asynchronous to each other [33], [40]. The fundamental equivalence of the different kinds of circuits which operate with asynchronous inputs such as the synchronizer, the arbiter, the latch and the inertial delay has also been demonstrated [12].

With metastability acknowledged as an unavoidable problem, the focus of research has turned to other aspects: characterization and modelling of the metastable state [26], [32], [34], [50], [59]; detailed analysis and simulation of metastable behaviour in different kinds of bistable devices, such as the CMOS RS flip-flop [29], [30], the CMOS D latch [28], CMOS ASIC flip-flops [27]; design methods for reducing the probability of metastable failure either at the component [27], [28], [51] or the system level [34], [39], [54], [55]; and to ways of measuring the failure rate of bistables [23], [27], [51] or calculating the probability of system failure due to its occurrence [26], [31], [34], [59].

Several different approaches have been used to model the metastable behaviour of bistable devices: theoretical models where the device is modelled as a simple regenerative loop [49], [59]; experimental studies [23], [51] and simulation studies [26], [32]. The resulting model is of a two stage process. The first phase, known as the sampling phase, is brought to an end as the regenerative loop closes. The device is triggered into metastability when the separation between clock and data edges at the end of the sampling phase falls within the metastable window. Once metastability is triggered, the resolution of the device outputs to one or other valid stable state during the regeneration phase follows an exponential path. Some details of the metastable window and the resulting probability expressions are briefly summarised in Appendix 2B.

To control the problem of metastable failure within systems, synchronizers have been adopted as an interface between the asynchronous input and the synchronous system. The synchronizer samples the asynchronous input at the rate of the system clock. The output of the synchronizer is thus synchronous with the rest of the system. The synchronizer is designed to provide an increased settling time before the sampled asynchronous input is presented to the rest of the system. However, the use of a synchronizer does not eliminate the possibility of metastable failure. When the

synchronizer fails, the problems associated with metastability will occur and can affect the system. It is therefore important both to develop synchronizer designs which reduce the probability of metastable failure to an acceptable level and to assess the performance of the designs using metastable failure probability models.

2.3 Synchronizer Designs and their Assessment

In this section, three new synchronizer designs first reported by the author in [60], [61] are described and their performance compared with that of existing synchronizer designs. The section begins with a description of the framework in which the synchronizers will be compared. Existing synchronizer designs are then briefly described. Then, the new synchronizer designs are presented and analysed. The section concludes with a comparison of the performance of the different designs.

2.3.1 A Framework for Assessment of Synchronizer Performance

The four performance indices used to compare the synchronizer designs are: the MTBF (mean time between failure), the data delay time, the data sampling rate and the complexity. The MTBF for the desired clock period and flip-flop characteristics gives the reliability of the design and is a function of the settling time, T_s, allowed by the synchronizer. Failure of a synchronizer will be defined to occur when the output flip-flop samples a metastable state at its input, where the output flip-flop is the flip-flop which interfaces directly with the system where the synchronizer is used.

The total data delay time, T_d, gives a measure of the efficiency of the synchronizer by indicating how much delay is required as a trade-off for the MTBF. The data sampling rate depends on the system clock rate and the design of the synchronizer. The complexity of the synchronizer design is represented by the number of flip-flops in the design.

The current model of metastable failure for bistable devices, as summarised in Appendix 2B, allows the calculation of metastable failure probabilities and MTBF. The model is developed on the basis of a single loop only, however, experimental work [27], [51] suggests that the equations still apply asymptotically to the more complicated circuits present in real devices. When attempting to predict the

performance of synchronizers made of many flip-flops, the model can still be used but it is necessary to model how metastability is transferred between devices.

There are at least two approaches to modelling the transfer of metastability from one device to the next [34], [49]. In the performance assessment of the synchronizers in the following sections, the sampling model [34] will be used. In the sampling model, it is assumed that the transfer of metastability occurs by simple sampling because, while a metastable state is resolving, the device is in its linear region of operation. In this first-order linear model, the metastable state is assumed to resolve progressively as it passes from device to device. The model is simple but requires the assumption that the devices have identical parameters and that the device behaviour while resolving is monotonic. It also does not allow for further metastable triggering in the second and subsequent devices.

2.3.2 Existing Synchronizer Designs

The shift register synchronizer [34], which uses a chain of N+1 flip-flops, is illustrated in Figure 2.1. The timing diagram for the shift register synchronizer is in Figure 2.2 and shows that it provides a settling time, T_s, of

$$T_s = NT_c - (N-1)t_{pd} \tag{2-1}$$

where T_c is the system clock and t_{pd} is the propagation delay of a flip-flop. The propagation delays of the internal N-1 flip-flops are deducted from the settling time, because it is only in these flip-flops that the simple sampling transfer of the resolving metastable state is assumed to occur.

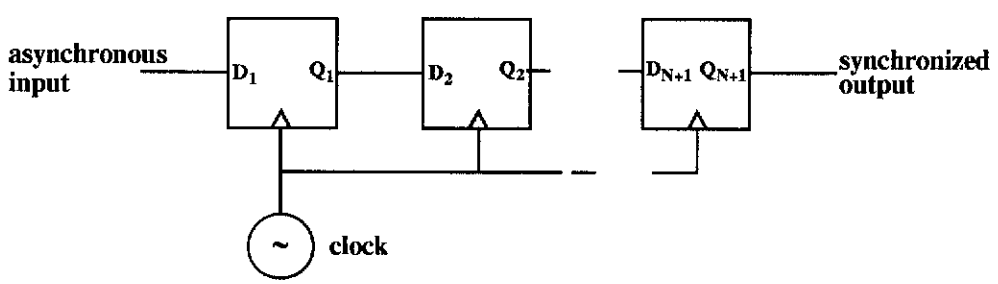


Figure 2.1: A shift register synchronizer.

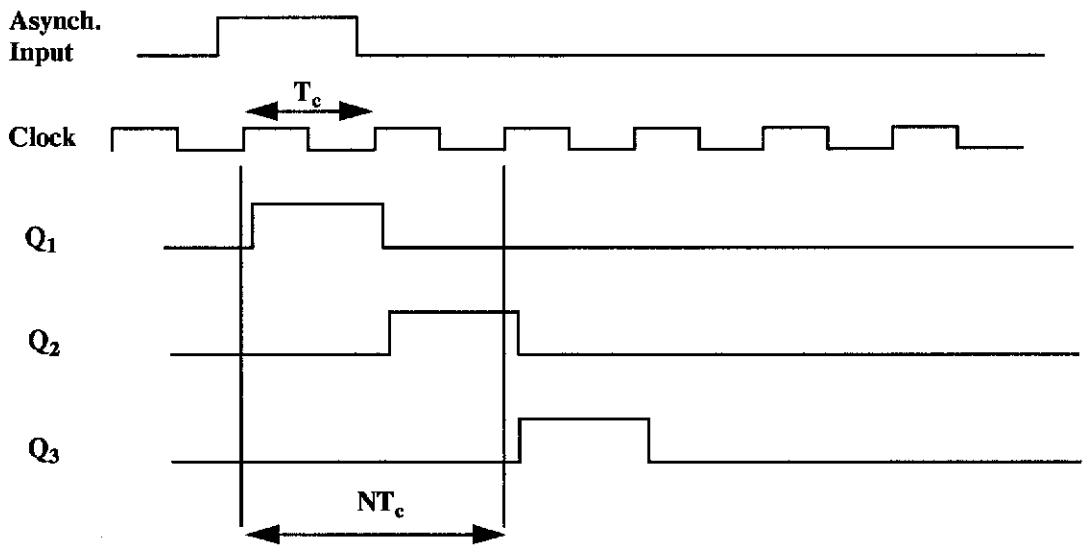


Figure 2.2: Timing diagram for the shift register synchronizer when N=2.

The MTBF is given by [34]

$$\frac{T_c e^{\frac{NT_c-(N-1)t_{pd}}{\tau}}}{\lambda \Delta t_{max}} \tag{2-2}$$

The data delay time of the shift register synchronizer is

$$T_d = NT_c \tag{2-3}$$

The second design to be discussed in this section is the divided clock synchronizer [34], illustrated in Figure 2.3, which uses three flip-flops and a divided clock. A timing diagram for the divided clock synchronizer is in Figure 2.4.

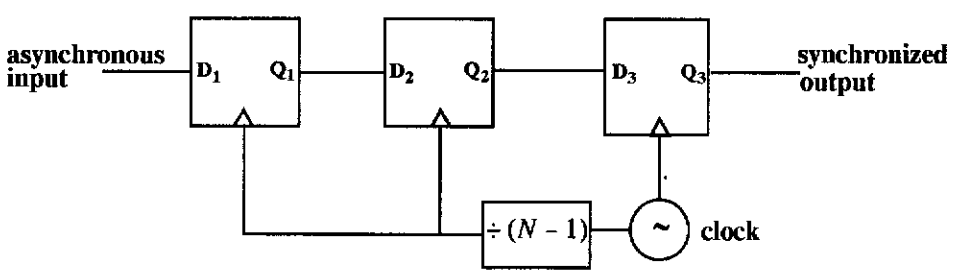


Figure 2.3: A divided clock synchronizer.

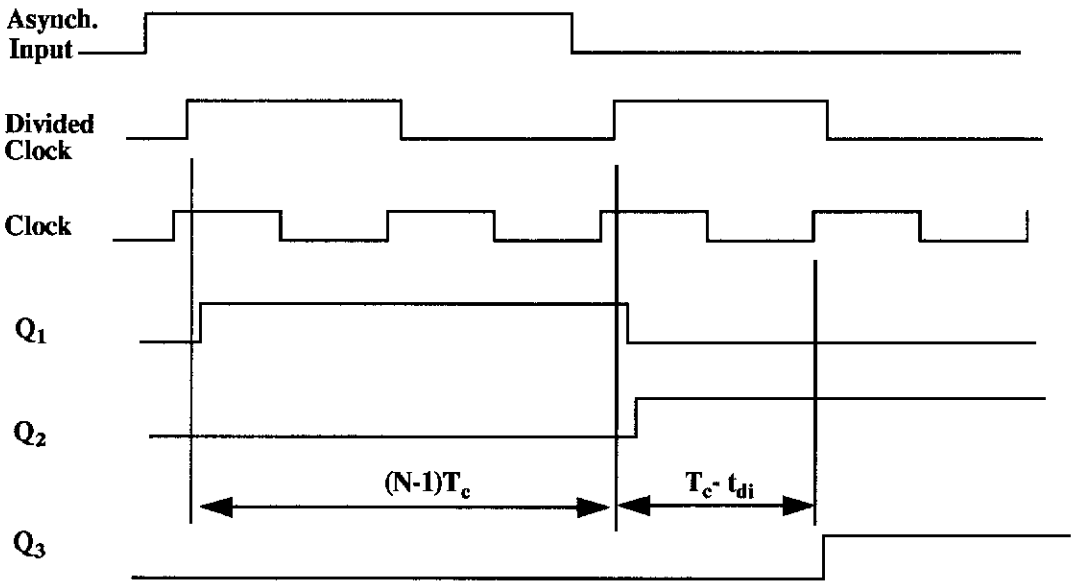


Figure 2.4: Timing diagram for the divided clock synchronizer for N=3.

The use of the divided clock in this synchronizer design means that most of the settling time is between the first and second flip-flops thus reducing the need for a long chain of flip-flops in order to provide a long settling time [34]. The settling time, T_s is,

$$T_s = NT_c - t_{pd} - t_{di} \tag{2-4}$$

where t_d is the delay in the clock divider circuit. Note that only the propagation delay of the second flip-flop, where the resolving metastable state is assumed to be transferred by simple sampling, is deducted from the settling time.

The MTBF is given by [34]

$$\frac{(N-1)T_c e^{\frac{NT_c - t_{pd} - t_{di}}{\tau}}}{\lambda \Delta t_{max}} \tag{2-5}$$

and the data delay time is

$$T_d = NT_c - t_{di} \tag{2-6}$$

2.3.3 Three New Synchronizer Designs

This section presents three new synchronizer designs, the first two of which were described by the author in [60], [61], and which represent distinct implementations of a basic synchronizer structure. The first has an asynchronous front end and is termed the AFE synchronizer. The second has a synchronous front end and is termed the SFE synchronizer. The third uses a new flip-flop design in a synchronous front end and is termed the SFE-NFF synchronizer.

2.3.3.1 The Asynchronous Front End Synchronizer

Figure 2.5 shows the AFE synchronizer. The design consists of four blocks: a sequential sampling register, a multiplexer, an output flip-flop and a timing generator. The sequential sampling register has N+1 independent flip-flops which all see the same asynchronous input, but are clocked cyclically. Each flip-flop within the sequential sampling register is clocked at $1/(N+1)$ the rate of the clock. Viewing the sequential sampling register as a whole, samples are taken at the clock frequency, but each successive sample is taken by a different flip-flop.

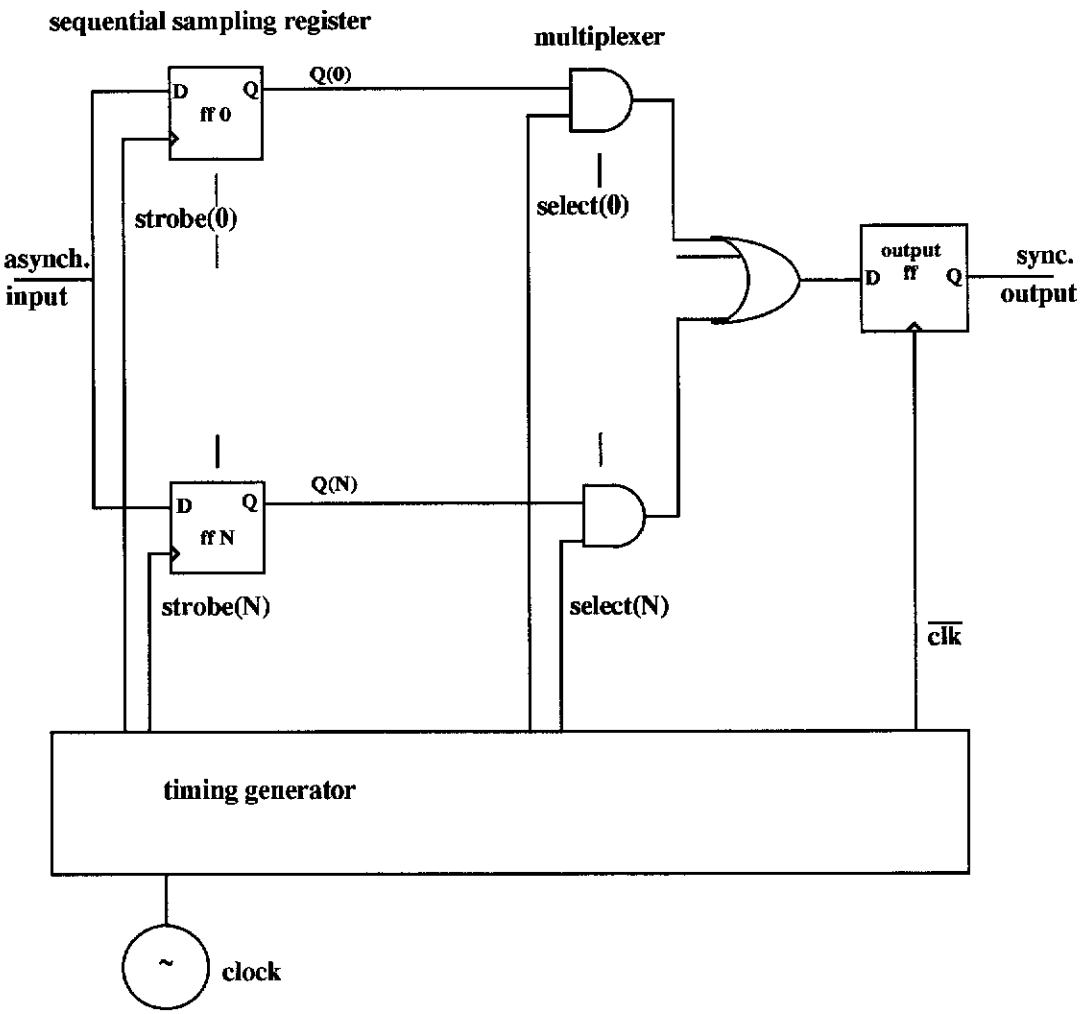
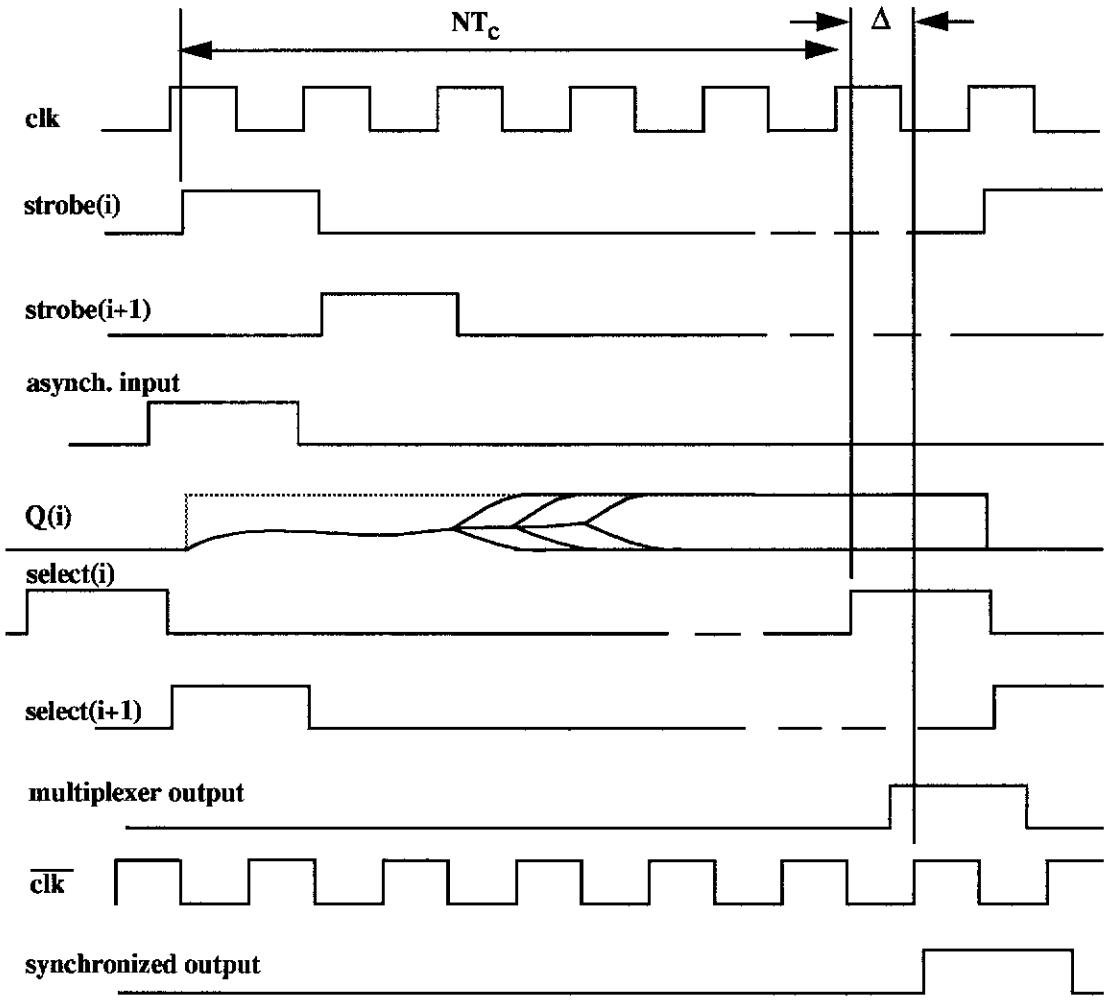


Figure 2.5: The new AFE synchronizer.

The N+1 outputs of the flip-flops are connected to the multiplexer together with N+1 corresponding select signals. The select signals are phased with respect to the flip-flop strobe by the timing generator, so that a given flip-flop's output is not selected until it has had N clock periods to settle. In this respect the design resembles the divided clock synchronizer. However, the multiplexer presents the delayed samples to the output flip-flop at the same rate as the samples are taken. This is a key difference with respect to the divided clock synchronizer which only samples the input every N-1 clock periods. The output flip-flop is clocked synchronously with the clocking of the flip-flops in the sequential sampling register. Figure 2.6 shows the timing diagram for the AFE synchronizer shown in Figure 2.5.



$$\Delta = \frac{T_c}{2} + t_{pi} - t_{ptg}$$

Figure 2.6: Timing diagram for AFE synchronizer.

Figure 2.7 shows the detail of a possible implementation of the timing generator. The shift register and nand gate combination is self-starting and circulates a zero through the shift register thus generating the cycle of N+1 successive pulses for the strobe and select signals.

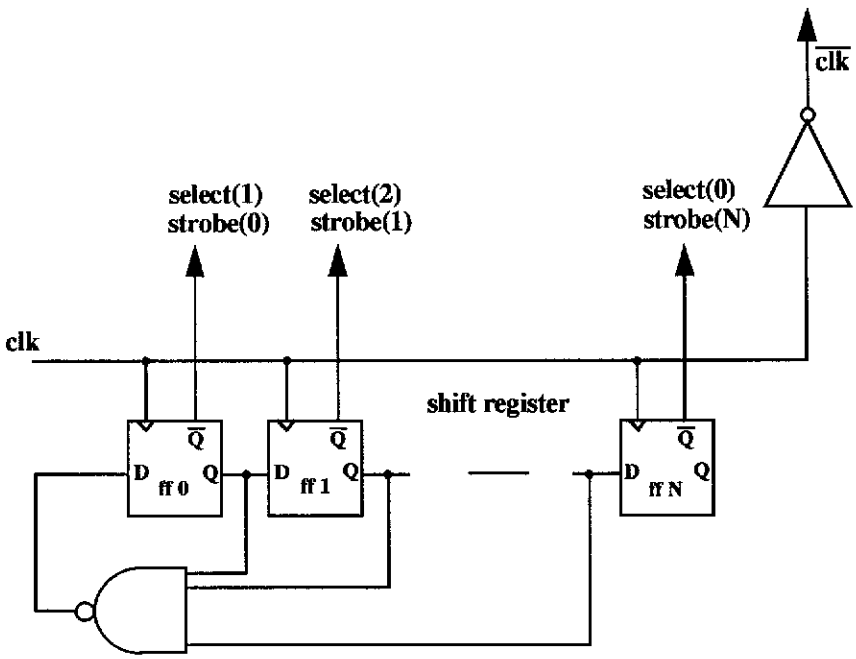


Figure 2.7: Possible implementation of the timing generator.

For the ith pathway through the AFE synchronizer, the settling time is calculated beginning from the clock edge at the ith input flip-flop, that is the ith strobe, to the clock edge at the output flip-flop, that is the inverted clock, as shown in the timing diagram of Figure 2.6. Hence, the settling time for the AFE synchronizer is given by

$$T_s = NT_c + \frac{T_c}{2} + t_{pi} - t_{ptg} - t_{pmux} \tag{2-7}$$

where t_{pi} is the propagation delay of the inverter which produces the inverted clock, t_{ptg} is the propagation delay of the timing generator in producing the strobe and select signals (assumed to be identical in all cases) and t_{pmux} is the propagation delay in the multiplexer, which is assumed not to be part of the settling time in a similar way that the propagation delay through the internal flip-flops was excluded in the previous designs.

The MTBF for the AFE synchronizer is given by

$$\frac{T_c e^{\frac{NT_c + T_c/2 + t_{pi} - t_{ptg} - t_{pmux}}{\tau}}}{\lambda \Delta t_{max}} \tag{2-8}$$

For the ith pathway through the synchronizer, the data delay time is calculated beginning from the clock edge at the ith input flip-flop, that is the ith strobe, to the clock edge at the output flip-flop, that is the inverted clock, as shown in the timing diagram of Figure 2.6. Hence, the data delay time for the AFE synchronizer design is given by

$$T_d = NT_c + \frac{T_c}{2} + t_{pi} - t_{ptg} \tag{2-9}$$

2.3.3.2 The Synchronous Front End Synchronizer Designs

In VLSI implementation, for ease of testing, it is preferable to use purely synchronous designs, for example, this simplifies the use of the scan path methodology. Two implementations of the synchronous front end synchronizer are presented in this section. The basic structure of the synchronizer remains the same as described for the AFE synchronizer in Section 2.3.3.1. A possible alternative and synchronous implementation of the synchronizer front end, as seen in the SFE synchronizer, is in Figure 2.8 and the corresponding timing diagram is shown in Figure 2.9.

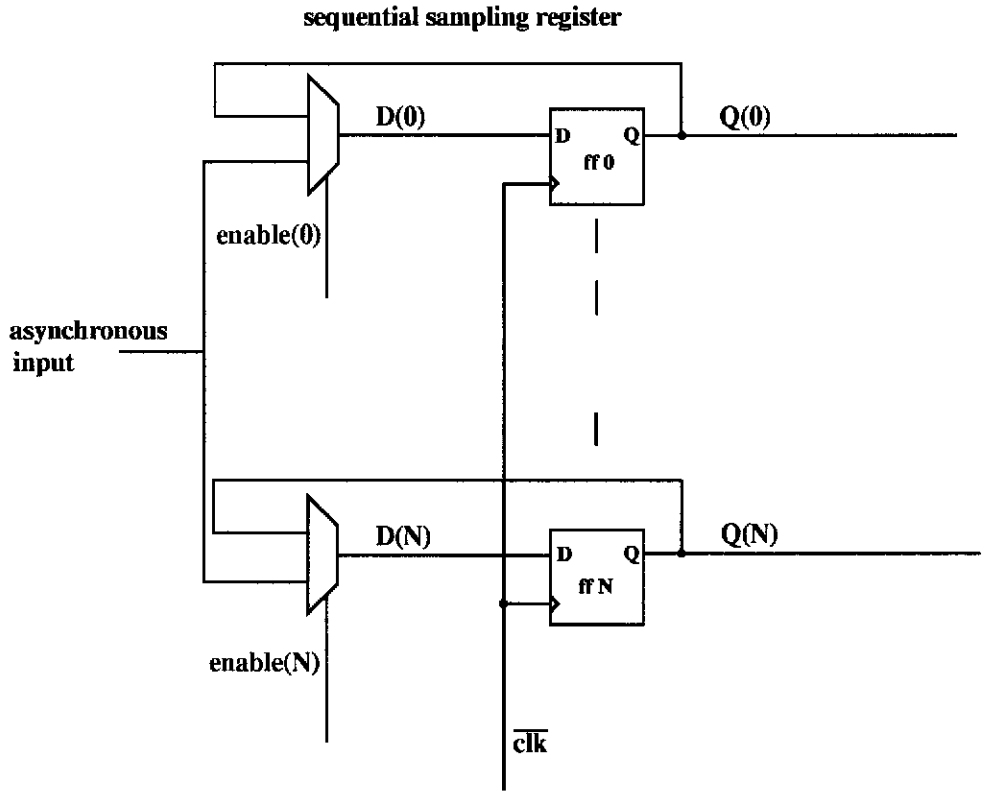


Figure 2.8: The front end of the new SFE synchronizer design.

In the SFE synchronizer, the enable takes the place of the strobe and is activated cyclically at $1/(N+1)$ the clock rate, routing the asynchronous input to the input of each flip-flop in turn. The flip-flops are clocked at the clock rate, thus although each flip-flop only samples the asynchronous input once every N+1 clock cycles, a sample is taken of the asynchronous input on every clock edge. When the enable input of a particular flip-flop is not activated, the output of that flip-flop is routed to its input. Thus, at every clock edge while its enable is not active, the flip-flop resamples its own output. For testing purposes the front end can be converted to a scan register. The timing of the data input can be forced to satisfy the necessary set-up and hold time requirements during testing.

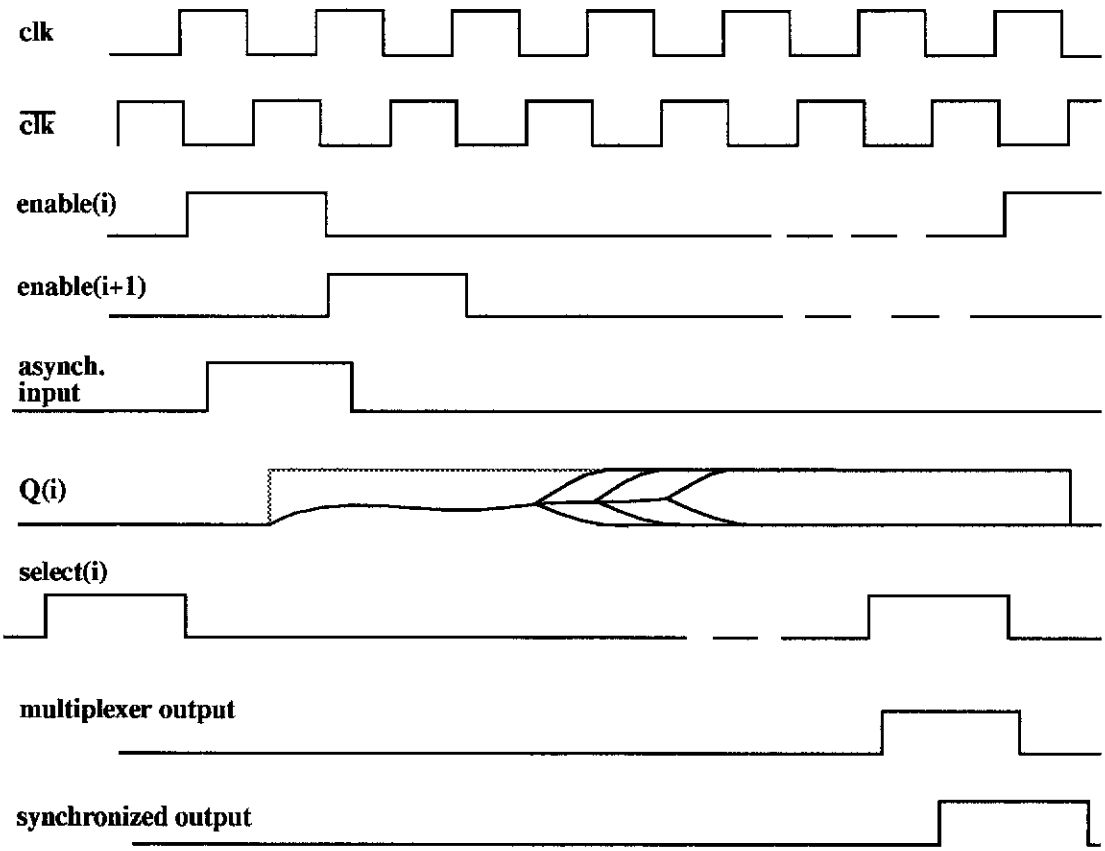


Figure 2.9: Timing diagram for SFE synchronizer.

The SFE synchronizer design is analogous to a shift register synchronizer with the first N flip-flops in the chain being exactly the same flip-flop. The settling time is calculated from the clock edge at the ith input flip-flop to the clock edge at the output flip-flop. As all the flip-flops are clocked by the same clock edge, the settling time is simply N clock periods minus the propagation times through the multiplexer and the internal flip-flops, which in this design is caused by the repeated clocking of a single input flip-flop when the input is not enabled. Hence, T_s is given by

$$T_s = NT_c - (N-1)t_{pd} - t_{pmux} \tag{2-10}$$

The MTBF for the SFE synchronizer is given by

$$\frac{T_c e^{\frac{NT_c - (N-1)t_{pd} - t_{pmux}}{\tau}}}{\lambda \Delta t_{max}} \tag{2-11}$$

The data delay time is calculated from the clock edge at the ith input flip-flop to the clock edge at the output flip-flop. As all the flip-flops are clocked by the same clock edge, the data delay time is simply N clock periods.

$$T_d = NT_c \tag{2-12}$$

A further interesting question about the implementation of the synchronous front end is the use of the flip-flop with an enable. In the analysis above it has been assumed that the implementation involves a simple multiplexer arrangement where the enable selects between the flip-flop input and the fed back flip-flop output as shown in Figure 2.8. Another possibility is shown in Figure 2.10, where the flip-flop uses the R-S latch design [14]. The SFE-NFF synchronizer is the synchronous front end implementation of the new synchronizer built with this new enable flip-flop.

Metastability may be triggered in the output latch of this flip-flop when the clock and data inputs occur too close together resulting in either the R or the S input being of a critical width [49]. Once the enable is low and the input is no longer selected, further access to the resolving latches during subsequent clock pulses is blocked. Neither the input nor the output of the flip-flop nor the clock can affect the resolving output latch of the flip-flop. For the output flip-flop to sample metastability, the input flip-flop must remain metastable for NT_c. Hence the MTBF for the SFE-NFF synchronizer is

$$\frac{T_c e^{\frac{NT_c}{\tau}}}{\lambda \Delta t_{max}} \tag{2-13}$$

where the data delay time is given by (2-12).

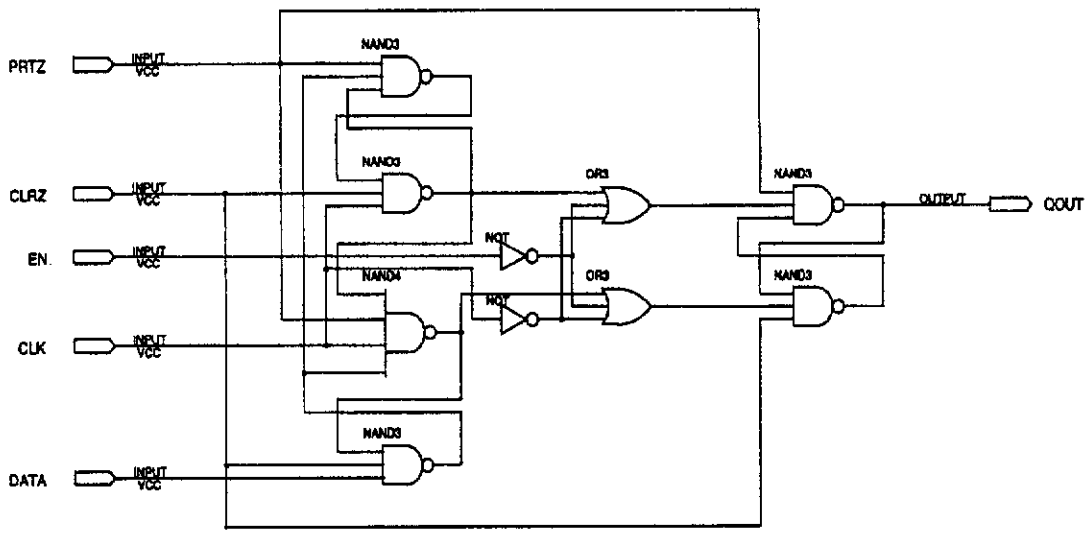


Figure 2.10: Alternative enable flip-flop implementation.

2.3.4 Comparison of the Synchronizer Designs

As described in Section 2.3.1, the four indices of synchronizer performance are MTBF, data delay time, sampling rate and complexity. For the comparison of the different synchronizer designs to be fair, it has to be made for settling times which are the same number of system clock cycles, for a given value of N. Comparison between different MTBFs will then give an indication of the influence of the synchronizer design on the MTBF. For a given value of N, synchronizer complexity and sampling rate may vary and these other factors may need to be weighed against the MTBF.

In Table 2.1, expressions are given for the MTBF and data delay time using the shift register synchronizer as the basis for comparison. From these expressions, it can be seen that designs which do not propagate the settling metastable state through successive devices enjoy an improved MTBF. The AFE synchronizer has an additionally improved MTBF due to an extra half of a system clock period resulting from the clocking scheme used. However, as a result, the AFE synchronizer suffers from a penalty in the data delay time. The SFE synchronizer, however, has a worse

MTBF than the shift register synchronizer due to the additional propagation delay through the multiplexer stage.

Table 2.1: Synchronizer Performance Comparison

Synchronizer Type	$\frac{MTBF}{MTBF_{SR}}$ [a]	$\frac{T_d}{T_{d,SR}}$ [b]	Sampling Rate	Number of Flip-flops
Shift Register	1	1	f_c	N+1
Divided Clock	$(N-1)e^{\frac{(N-2)t_{pd}-t_{di}}{\tau}}$	$1-\frac{t_{di}}{NT_c}$	$\frac{f_c}{N-1}$	3
AFE Synchronizer	$e^{\frac{\frac{T_c}{2}+(N-1)t_{pd}+t_{pi}-t_{ptg}-t_{pmux}}{\tau}}$	$1+\frac{\frac{T_c}{2}+t_{pi}-t_{ptg}}{NT_c}$	f_c	N+2
SFE Synchronizer	$e^{\frac{-t_{pmux}}{\tau}}$	1	f_c	N+2
SFE-NFF Synchronizer	$e^{\frac{(N-1)t_{pd}}{\tau}}$	1	f_c	N+2

a. $MTBF_{SR}$ Mean time between failure of the shift register synchronizer.

b. $T_{d,SR}$ Data delay time of the shift register synchronizer.

The comparisons are evaluated in Table 2.2. In order to accommodate the clock division in the divided clock synchronizer, the minimum value of N for which the performance comparison can be made is $N = 3$. When computing the ratios, it is necessary to assume that the shift register synchronizer and the design with which it is being compared are built with the same type of flip-flop. For the divided clock synchronizer and the AFE synchronizer, 74HC D type flip-flops are assumed with the following parameters $t_{pd} = 16$ ns, $\tau = 1.3$ ns, the clock divider has $t_{di} = 15$ ns and the system clock period is $T_c = 100$ ns [49]. The other delay terms are assumed to be 74AC components with $t_{pi} = 1.7$ ns, $t_{ptg} = 12.3$ ns, $t_{pmux} = 13$ ns where the delays have been assumed to be worst case, that is, to reduce the MTBF. For the comparison with the SFE synchronizer, it is only necessary to assume that it and the shift register synchronizer be built of the same type of flip-flop. For comparison with the SFE-NFF synchronizer, it is not necessary to assume that the shift register synchronizer is built with the new type of flip-flop.

Table 2.2: Quantitative Synchronizer Performance Comparison.

Synchronizer Type	$\frac{MTBF}{MTBF_{SR}}$ [a] $N = 3$	$\frac{T_d}{T_{d,SR}}$ [b] $N = 3$	$\frac{MTBF}{MTBF_{SR}}$ $N = 9$	$\frac{T_d}{T_{d,SR}}$ $N = 9$
Shift Register	1	1	1	1
Divided Clock	4.32	0.95	2.03×10^{33}	0.98
AFE Synchronizer	3.23×10^{19}	1.13	3.81×10^{51}	1.04
SFE Synchronizer	4.54×10^{-5}	1	4.54×10^{-5}	1
SFE-NFF Synchronizer	4.9×10^{10}	1	5.77×10^{42}	1

a. $MTBF_{SR}$ Mean time between failure of the shift register synchronizer.
b. $T_{d,SR}$ Data delay time of the shift register synchronizer.

Table 2.2 confirms that the designs which do not require propagation through many cascaded devices are expected to perform better. The divided clock synchronizer is only slightly better than the shift register synchronizer for small N but the effect of delays such as t_{di} on the MTBF is overcome for larger values of N. However, as N increases there is a penalty in the $1/(N-1)$ times lower sampling rate.

The AFE and SFE-NFF synchronizers have better performance for a given N, than the shift register synchronizer, but are much more complex in design. The shift register synchronizer may still be designed to a desired MTBF and may be favoured because it is simpler. However, to characterize a practical realization of the shift register synchronizer is difficult. The simple sampling model of metastability transfer which has been assumed is a simplification and real systems may introduce additional dynamics between the devices. It also assumes that all flip-flops in the chain have identical parameters, whereas in a real system this may not be the case.

The SFE synchronizer overcomes one of these problems because it is equivalent to a shift register synchronizer with the first N flip-flops in the chain being the same flip-flop, hence the assumption that the flip-flop parameters are identical is valid. The

slightly worse performance than the shift register synchronizer may be overcome by using a fast multiplexer stage and acceptable performance can still be obtained by selecting an appropriate value of N.

Although the new designs use more flip-flops and are more complex in their design, they provide much better performance. Their main advantage however, lies in the greater reliance which can be placed on the MTBF calculation. Manufacturers provide data for the flip-flop parameters τ and Δt_{max} under the assumption that the flip-flops will be used in a simple two flip-flops synchronizer [1], [2]. In calculating the MTBF for the simple two flip-flops synchronizer, assumptions about metastability transfer are not required as failure is defined to occur if the second flip-flop samples a metastable state. The determination of the MTBFs for the AFE synchronizer and for the SFE-NFF synchronizer also makes no assumptions about metastability transfer between flip-flops nor places any requirement on flip-flops in the design to be closely matched. Characterization reduces to considering a pair of synchronizing flip-flops and the parameters of the worst case sampling flip-flop can be used. Thus, these designs can provide an acceptable performance with a greater certainty that the actual performance will correspond to the predicted performance.

Appendix 2A: Glossary

t_d - delay time before loop output reaches a valid logic level

t_{pd} - normal propagation delay of a flip-flop

t_m - delay value defining metastable failure

Δt_{max} - maximum size of metastable window

λ - mean rate of arrival of data edges (Poisson process) in probability model of metastability

τ - regeneration time constant in probability model of metastability

T_c, f_c - system clock period and system clock rate

T_s - settling time allowed for synchronizer output to resolve

T_d - data delay time in a synchronizer

t_{di} - propagation delay of clock divider circuit in divided clock synchronizer

t_{pi} - propagation delay of inverter in new synchronizer designs

t_{ptg} - propagation delay of timing generator in new synchronizer designs

t_{pmux} - propagation delay of multiplexer in new synchronizer design

Appendix 2B: Metastable Failure Probability Equations

This appendix gives the equations which describe the probability of metastable failure in bistable devices [31], [34], [39], [49].

Metastable failure is defined as the failure of a device to resolve by some time t_m, which is the time available for its output to settle before the output is required by other devices in a system.

The metastable failure probability $P(t_d > t_m)$, the probability that the delay time of the latch is greater than some value t_m, is given by [49]

$$P(t_d > t_m) \approx \lambda \Delta t_{max} e^{-\left(\frac{t_m - t_{pd}}{\tau}\right)}, t_m > t_{pd} \qquad (2B\text{-}1)$$

where it is assumed that the arrival times of the data edges are determined by a Poisson process with a mean rate of arrival of edges given by λ. (2B-1) says that the probability of metastable failure is approximately equal to the mean rate of arrival of data edges multiplied by the size of the metastable window corresponding to the desired delay time of t_m.

If it is assumed that the probability of metastable failure is an independent probability in each clock period then it is possible to calculate the mean time between failure (MTBF) for a given settling time, T_s, as [34], [49]

$$MTBF = \frac{T_c e^{\frac{T_s}{\tau}}}{\lambda \Delta t_{max}} \qquad (2B\text{-}2)$$

where T_c is the clock period.

Chapter 3
Jitter

3.1 Introduction

In digital systems, the value of signals matters only at particular instants in time. For example, digital transmission link repeaters regenerate the signal based on the detection of each bit at a significant instant set by the associated (recovered) clock. Digital systems are susceptible to errors caused by the displacement of the significant instants of a signal. The disturbance which causes the significant epochs of a signal to vary from their ideal positions in time is known as jitter.

The errors caused by jitter are not confined to bit errors, as may occur in digital repeaters when there is excessive jitter. Jitter on digital data which is to be converted back into analog form may cause unwanted phase modulation on the resulting analog signal [8]. Variations in rates of writing or reading a buffer due to jitter on clock signals can cause buffer slips leading to loss of data.

The purpose of this chapter is to provide an introduction to jitter, covering a range of topics in jitter modelling and analysis which are of relevance to later chapters. In Section 3.2, time jitter is formally defined and the necessary framework and models for describing jitter are presented. Phase jitter is introduced in Section 3.3 and the question of when time jitter can be approximated using phase jitter is explored. Section 3.4 presents the analysis of an important class of jitter recurrence equation, the jitter equation of the synchronization process, which has been analysed for the first time by the author [60], [62]. In Section 3.5, techniques for Fourier analysis of jitter are developed and applied to the analysis of the jitter spectrum of the synchronization process. In Section 3.6 the analysis of the jitter equation of the synchronization process is extended to include input jitter and the spectrum of the synchronization process jitter in the presence of input jitter is derived in Section 3.7.

3.2 Jitter - Definitions, Framework and Modelling

3.2.1 Definition of Time Jitter

Time jitter is defined as the difference between the ideal position of the significant edge of a signal and its actual position as shown in Figure 3.1. If a jittered pulse is delayed with respect to the ideal pulse, the time jitter is positive; if it precedes the

corresponding ideal pulse then the time jitter is negative. The time jitter values are defined at the regular time instants of the ideal signal and are thus discrete in time but continuous in amplitude. Such a discrete time sequence can be viewed as samples of a continuous time jitter function $e(t)$ at times $t = nT_c$.

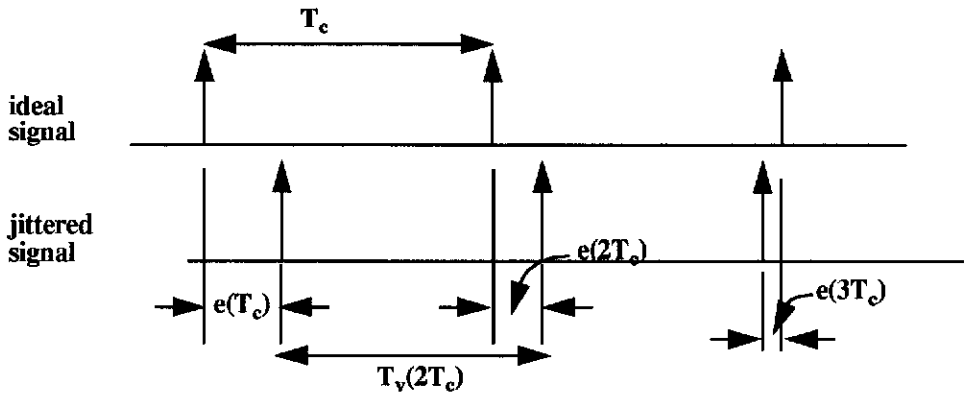


Figure 3.1: Definition of time jitter

As time jitter has dimensions of time, a time jitter function will be denominated in seconds and the derivative of a time jitter function will be dimensionless. It is also possible to consider normalised time jitter where the time jitter is normalised to the period of the ideal signal, for example, for the time jitter samples $e(nT_c)$

$$\hat{e}(nT_c) = \frac{e(nT_c)}{T_c} \tag{3-1}$$

Clearly, the normalised time jitter is dimensionless.

The amplitude of jitter is often referred to in unit intervals. The unit interval is a way of normalising the jitter amplitude to the period of the ideal or reference signal. When dealing with jitter on a particular signal, the unit interval is defined to be equal to the ideal period of that signal and the amplitude of the jitter in unit intervals is given by

$$\hat{A}_j = \frac{A_j}{T_c} \tag{3-2}$$

that is where

$$1\text{UI} = T_c \tag{3-3}$$

As time jitter forms the major subject of this work, it will generally be referred to simply as jitter.

3.2.2 Jitter Recurrence Equation

Referring to Figure 3.1, it can be seen that it is possible to define an expression for the variable length time interval between any two jittered pulses in terms of the period of the ideal signal and the value of the jitter at either end of the time interval in question. In particular, $T_v(nT_c)$, the interval between adjacent jittered pulses, can be considered the instantaneous period of the jittered signal. The instantaneous period of the jittered signal is expressible only in terms of the discrete jitter samples and is defined only at the regular instances of the corresponding ideal period T_c, that is at one or other of its endpoints, as shown in (3-4) where it has been defined at the endpoint corresponding to the jitter value $e(nT_c)$.

$$T_v(nT_c) = T_c + e(nT_c) - e((n-1)T_c) \tag{3-4}$$

3.3 Time Jitter and Phase Jitter

In this section, another type of jitter called phase jitter [18] is introduced. There is a clear distinction between time jitter and phase jitter [10], [18] which is summarised here. For mathematical tractability, it may sometimes be desirable to approximate one form of jitter using the other. Under certain conditions it is possible to do this. It is important to appreciate the limitations imposed on the jitter approximation as it is used in experimental work described in Section 6.4 of Chapter 6.

Phase jitter is defined as the difference between the jittered and reference phases sampled at the regularly spaced instants of the reference clock, that is

$$p(nT_c) = \phi_{jit}(t) - \phi_{ref}(t)\big|_{t = nT_c} \tag{3-5}$$

By analogy with time jitter in Section 3.2.1, the phase jitter can be viewed as samples, taken at the instants of the reference clock, of some phase jitter function,

$p(t) = \phi_{jit}(t) - \phi_{ref}(t)$, where $\phi_{ref}(t)$ is the phase of an ideal reference clock signal which has perfectly regularly spaced clock pulses with period, T_c

$$\phi_{ref}(t) = \frac{t}{T_c} \tag{3-6}$$

and the phase of the jittered clock signal is given by some function, $\phi_{jit}(t)$, such that

$$\phi_{jit}(nT_c + e(nT_c)) = n \tag{3-7}$$

As illustrated on the graph of time versus phase for reference and jittered clocks shown in Figure 3.2, both phase jitter and time jitter are defined at the regularly spaced instants of the reference clock.

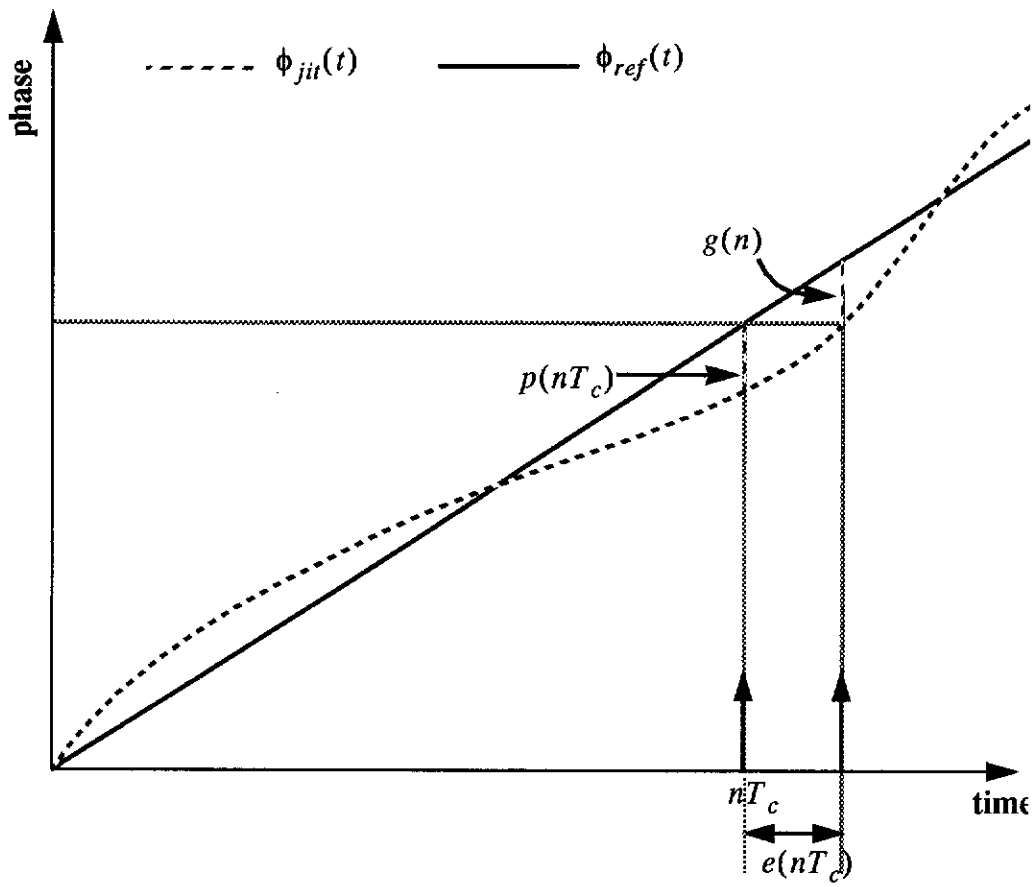


Figure 3.2: Distinction between time jitter and phase jitter.

For purposes of comparing time jitter and phase jitter, it is convenient to define a quantity $g(n)$ which is related to time jitter but is dimensionless like phase jitter. $g(n)$ is defined as samples of the phase jitter function, $p(t)$, at the irregularly spaced instants of the jittered clock

$$g(n) = p(t)\big|_{t = nT_c + e(nT_c)} = -\frac{e(nT_c)}{T_c} \tag{3-8}$$

As shown in (3-8), $g(n)$ is defined to be of the opposite sign to the normalised time jitter. For example, if the jittered pulse is delayed with respect to the reference pulse and so the time jitter is positive, the phase of the jittered signal must be behind that of the reference signal and so the phase difference at that time, which is given by $g(n)$, is negative. $g(n)$ can be approximated by phase jitter as [10]

$$g(n) \approx \frac{p(nT_c)}{1 + p(nT_c) - p((n-1)T_c)} \tag{3-9}$$

(3-9) applies when the jitter is linearly varying and can be described using a first order Taylor series expansion. Furthermore, from (3-9) if the difference of adjacent samples of $p(nT_c)$ is very small compared to unity [10] then

$$g(n) \approx p(nT_c) \tag{3-10}$$

If it is desired to approximate time jitter using phase jitter, as in (3-10), then it is shown in Appendix 3B that the restriction on the derivative of the phase jitter function is

$$\left|\frac{d}{dt}p(t)\right|_{max} \ll f_c \tag{3-11}$$

For the case of sinusoidal jitter, these conditions result in the following restriction on the jitter frequency and amplitude

$$f_j A_j \ll \frac{1}{2\pi} \tag{3-12}$$

The restriction on sinusoidal jitter frequency and amplitude is explored quantitatively in Section 6.4 of Chapter 6 where phase jitter approximation of time jitter is used in experimental work.

3.4 The Synchronization Process Jitter Equation for an Ideal Asynchronous Input

In its simplest and most straightforward sense, the term synchronization refers to the aligning of the timing instants of an asynchronous external signal to the timing instants of the local system clock. One result of the asynchronism is that the asynchronous input signal period is not an integer number of local clock periods. It is shown in this section that, for an ideal asynchronous input, the synchronized output signal does not have a fixed period but varies between two values. The two values are the closest integer number of local clock periods on either side of the input signal period. As a result of the varying output period, the synchronized output signal is jittered with respect to the input signal. Another important result of the asynchronism is the possibility of metastability. However, as has been considered in Chapter 2, the use of a properly designed synchronizer can make the probability of metastable failure acceptably low. Hence, the possibility of metastable failure can be ignored in the jitter analysis.

In this section a jitter equation that models the process of synchronization and which was derived for the first time by the author [60], [62] is presented. The model is applicable to two applications of practical importance examined in subsequent chapters.

The synchronization process is modelled in terms of an asynchronous timing reference input and a local clock. The essential point is that the synchronized output is aligned with the significant edges of the local clock and is jittered with respect to the asynchronous input. A diagram which shows the relationships between the asynchronous input, the synchronized output and the jitter is in Figure 3.3.

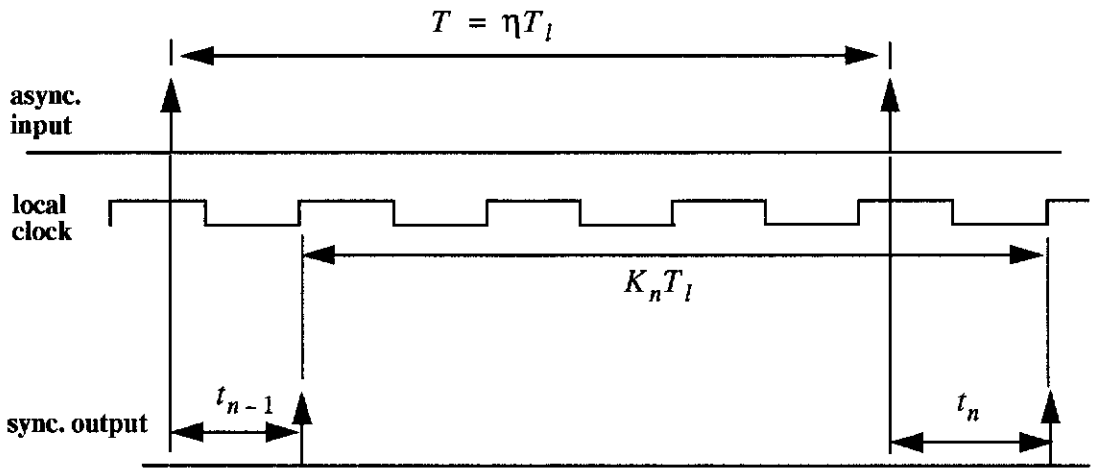


Figure 3.3: Synchronization process jitter diagram.

An expression for the varying delay between the time of the asynchronous input and the time of the synchronized output is obtainable directly from Figure 3.3 and is given in (3-13)

$$t_n = t_{n-1} + K_n T_l - \eta T_l \tag{3-13}$$

where T is the period of the asynchronous input, T_l is the local clock period, $\eta = T/T_l \in \Re$, K_n is an integer number of periods of T_l, and t_n is the jitter on the synchronized output. The model allows for the fact that practical synchronizers have a positive delay.

Mapping the local clock period onto T in the synchronization process results in two possible patterns as illustrated in Figure 3.4.

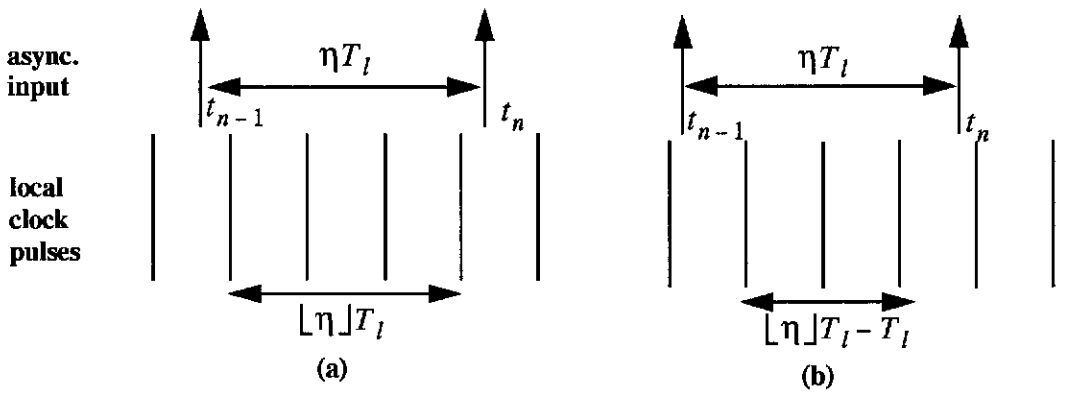


Figure 3.4: Mapping local clock onto period of asynchronous input.

Part (a) of Figure 3.4 shows the first case where the integer part of η is provided by a corresponding integer number of full periods of T_l giving (3-14).

$$\eta T_l = \lfloor \eta \rfloor T_l + t_{n-1} + (T_l - t_n) \tag{3-14}$$

Part (b) of Figure 3.4 illustrates the second case where the integer number of full periods of T_l is one period less than the integer part of η giving (3-15).

$$\eta T_l = \lfloor \eta \rfloor T_l + t_{n-1} - t_n \tag{3-15}$$

where $\lfloor a \rfloor$ = greatest integer $\leq a$.

It is shown in Appendix 3C that these are the only two possible patterns and the derivation for the combined form in (3-16) is given, where use is made of the new variable $x_n = t_n / T_l$.

$$\eta = \lfloor \eta \rfloor + x_{n-1} - x_n + 1 - u(x_{n-1} + \lfloor \eta \rfloor - \eta) \tag{3-16}$$

The closed form of (3-16) is also derived in Appendix 3C and is

$$x_n = x_0 - n\eta - \lfloor x_0 - n\eta \rfloor \tag{3-17}$$

which can be written in terms of absolute time, that is with $t_n = x_n T_l$

$$t_n = t_0 - n\eta T_l - \left\lfloor \frac{t_0}{T_l} - n\eta \right\rfloor T_l \tag{3-18}$$

Finally, it is shown in Appendix 3D that the jitter equation for the synchronizer is stable. Stability means that the long term average period of the output signal is equal to the period of the input signal, although individual periods of the output signal may equal either $\lfloor \eta \rfloor$ or $\lfloor \eta \rfloor + 1$ local clock periods. Because of the varying period of the output signal, the significant instants of the output signal are jittered with respect to the significant instants of the input signal. The resulting jitter on the output signal is expressed by (3-18).

3.5 Fourier Transforms of Jitter

The spectrum of the jitter provides an alternate and, in some applications, more useful characterization of the jitter. For example, it is useful in characterizing the effect of filtering by a phase-locked loop. The spectrum of the synchronization process jitter is used in Chapter 4 and Chapter 5 to investigate the performance of two different practical applications of the synchronization process.

As discussed in Section 3.2, jitter is defined as a series of discrete time samples taken at the regular instants of the reference clock. When the samples are the samples of some continuous time function, there is a straightforward technique for finding the spectrum of the jitter using the Fourier Transform of the continuous time function.

Consider time jitter, $e(nT_c)$, which can be considered as samples of some jitter function $e(t)$ taken at $t = nT_c$. If the continuous time jitter function has some Fourier Transform $E(f)$, then

$$\begin{aligned} \mathcal{F}\{e(nT_c)\} &= f_c \mathcal{F}\left\{ e(t) \sum_{n=-\infty}^{\infty} \delta(t - nT_c) \right\} \\ &= f_c E(f) \otimes \sum_{n=-\infty}^{\infty} \delta(f - nf_c) \\ &= f_c \sum_{n=-\infty}^{\infty} E(f - nf_c) \end{aligned} \tag{3-19}$$

where $\otimes$ represents the convolution operator.

Applying this approach to the synchronizer jitter equation, the function $z(t)$ when sampled at $t = nT$ yields (3-18).

$$z(t) = t_0 - \frac{\eta t T_l}{T} - \left\lfloor \frac{t_0}{T_l} - \frac{\eta t}{T} \right\rfloor T_l \tag{3-20}$$

As $z(t)$ is periodic with period $\frac{T}{\eta}$ secs, it can also be written as a Fourier Series

$$z(t) = \sum_{k=-\infty}^{\infty} d_k e^{\frac{j2\pi k\eta t}{T}} \tag{3-21}$$

where

$$d_k = \begin{cases} \frac{T_l}{2}, & k = 0 \\ \frac{e^{-j2\pi k x_0 T_l}}{j2\pi k}, & k \neq 0 \end{cases} \tag{3-22}$$

The Fourier Transform of (3-21) is

$$Z(f) = \sum_{k=-\infty}^{\infty} d_k \delta(f - k\eta F) \tag{3-23}$$

where the coefficients d_k are as defined in (3-22), $F = 1/T$, and use is made of the property that

$$\sum_{k=-\infty}^{\infty} e^{\frac{-j2\pi kt}{T}} \leftrightarrow \sum_{k=-\infty}^{\infty} \delta(f - kF) \tag{3-24}$$

Applying (3-19)

$$\begin{aligned} J(f) &= \mathcal{F}\{t_n\} \\ &= F \sum_{k=-\infty}^{\infty} d_k \sum_{n=-\infty}^{\infty} \delta(f - k\eta F - nF) \end{aligned} \tag{3-25}$$

It is shown in Appendix 3E that the RMS jitter power is

$$P = F^2\left(\frac{T_l^2}{4} + \frac{T_l^2}{12}\right) \tag{3-26}$$

The first term on the right is the DC power.

3.6 The Synchronization Process Jitter Equation for a Jittered Asynchronous Input

In this section, the synchronizer jitter equation will be extended by deriving the expressions which apply when the asynchronous input signal is jittered with respect to the ideal asynchronous input considered in Section 3.4. In the practical applications presented in Chapter 4 and Chapter 5, it is not likely that an ideal, perfectly regular asynchronous input is obtainable, so it is useful to derive the jitter equation and spectrum under these conditions.

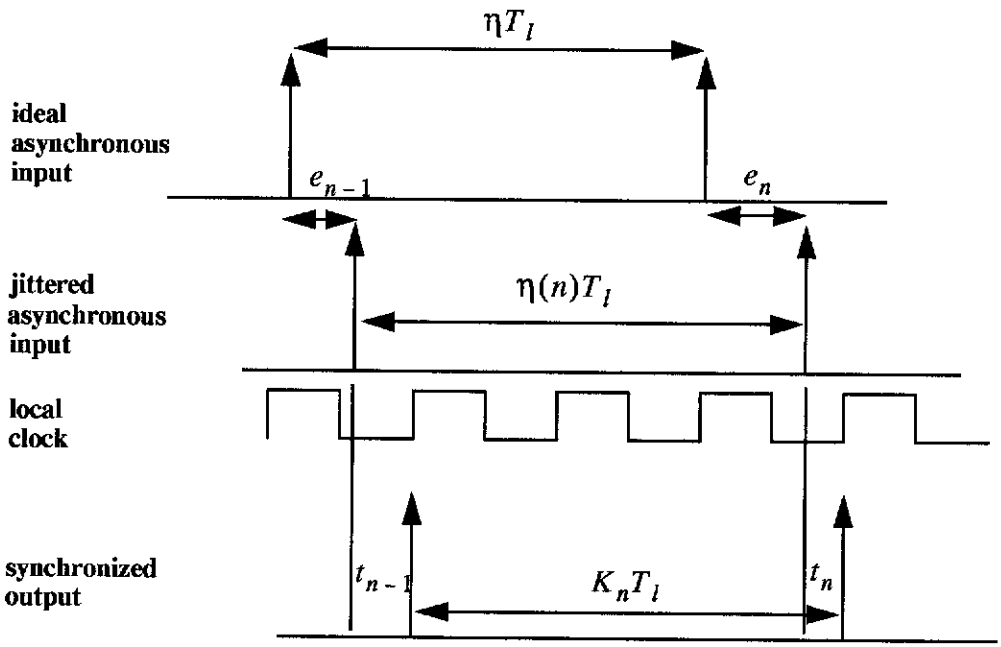


Figure 3.5: Timing diagram for a jittered asynchronous input.

When the asynchronous input is jittered, the period of the asynchronous input signal is varying and the ratio of its period to the local clock period is not given by some

constant, but is also varying. As shown in Figure 3.5, the relationship between the ideal asynchronous input and the jittered asynchronous input is given by

$$\eta T_l + e_n - e_{n-1} = \eta(n) T_l \tag{3-27}$$

where e_n is the nth sample of the jitter on the asynchronous input. The varying period of the jittered asynchronous input is expressed by $\eta(n)T_l$. Neither η nor $\eta(n)$ is necessarily an integer. Also from Figure 3.5, it can be seen that (3-13) still applies to the relationship between the input and the output of the synchronizer. Hence, combining the two equations it is possible to write

$$\eta T_l + e_n - e_{n-1} = t_{n-1} + K_n T_l - t_n \tag{3-28}$$

where K_n is an integer number of local clock periods but is not a constant.

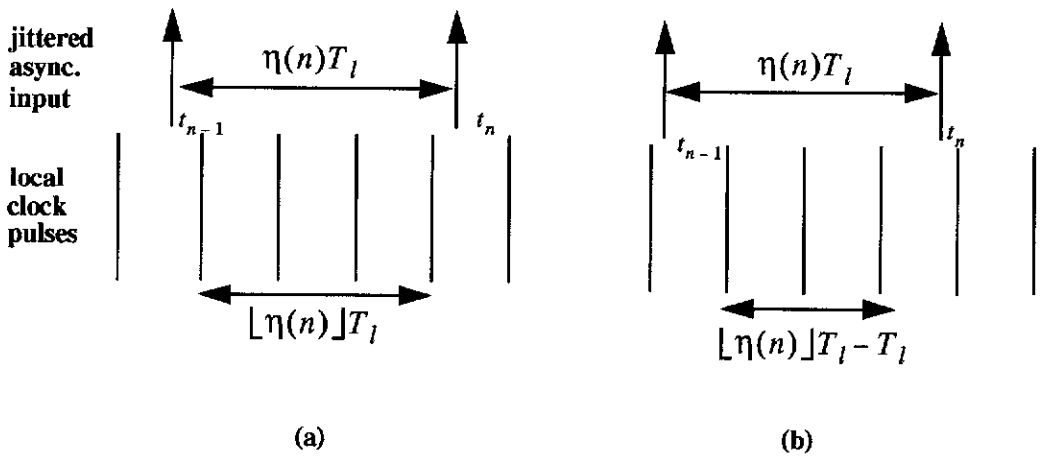


Figure 3.6: Mapping local clock onto jittered asynchronous input period.

Using the same approach as in Section 3.4, Figure 3.6 shows that the local clock period may be mapped onto the varying period, $\eta(n)T_l$, of the jittered asynchronous input resulting in two possible patterns. The proof given in Appendix 3C which shows that these are the only two possibilities for a constant period applies equally to the case of a varying period.

Part (a) of Figure 3.6 shows the first case where the integer part of $\eta(n)$ is provided by a corresponding integer number of full periods of T_l giving (3-29).

$$\eta(n)T_l = \lfloor \eta(n) \rfloor T_l + t_{n-1} + (T_l - t_n) \tag{3-29}$$

Part (b) of Figure 3.6 illustrates the second case where the integer number of full periods of T_l is one period less than the integer part of $\eta(n)$ giving (3-30).

$$\eta(n)T_l = \lfloor \eta(n) \rfloor T_l + t_{n-1} - t_n \tag{3-30}$$

In Appendix 3F, it is shown how (3-29) and (3-30) can be combined into a single equation and the closed form of (3-31), using the new variable $x_n = t_n/T_l$, is derived.

$$x_n = x_0 - n\eta - \frac{e_n}{T_l} + \frac{e_0}{T_l} - \left\lfloor x_0 - n\eta - \frac{e_n}{T_l} + \frac{e_0}{T_l} \right\rfloor \tag{3-31}$$

(3-31) can be written in terms of absolute time, that is with $t_n = x_n T_l$ as

$$t_n = t_0 - n\eta T_l - e_n + e_0 - \left\lfloor \frac{t_0}{T_l} - n\eta - \frac{e_n}{T_l} + \frac{e_0}{T_l} \right\rfloor T_l \tag{3-32}$$

It is shown in Appendix 3G that the jitter equation for the synchronizer is stable when there is jitter on the asynchronous input, as long as the jitter on the asynchronous input is bounded. If the jitter on the input signal is bounded, then the long term average period of the jittered input signal will be equal to the ideal period of the input signal and so the long term average period of the synchronized output will equal the ideal period of the input signal.

(3-33) shows the total jitter on the synchronized output which consists of two components. One component has the same form as the jitter on the synchronized output when there is no input jitter. Now that there is input jitter, this component is a function of the samples of input jitter as well as a function of time. The second component is a feedthrough term of the input jitter.

$$\upsilon_n = \left(t_0 - n\eta T_l - e_n + e_0 - \left\lfloor \frac{t_0}{T_l} - n\eta - \frac{e_n}{T_l} + \frac{e_0}{T_l} \right\rfloor T_l\right) + e_n \tag{3-33}$$

3.7 Jitter Spectrum for the Jittered Asynchronous Input

Determining the spectrum of the jitter when the asynchronous input is jittered is more difficult than for the case where the asynchronous input period is constant. It is shown in Appendix 3H, that the spectrum is given by

$$J_{JJ}(f) = F \sum_{n=-\infty}^{\infty} (D(f - nF) + E(f - nF)) \tag{3-34}$$

where $E(f)$ is the spectrum of the feedthrough component of the input jitter on the asynchronous input and $D(f)$ is the spectrum of the jitter produced by the synchronization process when there is input jitter. In general, $D(f)$ is given by

$$D(f) = \sum_{k=-\infty}^{\infty} \gamma_k A(f - k\eta F) \tag{3-35}$$

where the coefficients γ_k are

$$\gamma_k = \begin{cases} \dfrac{T_l e^{-j2\pi k\left(x_0 + \frac{e_0}{T_l}\right)}}{j2\pi k}, & k \neq 0 \\ \dfrac{T_l}{2}, & k = 0 \end{cases} \tag{3-36}$$

It is then shown in Appendix 3H that for the particular case of sinusoidal input jitter $e(t) = A_j \sin(2\pi f_j t + \theta_j)$, $A(f)$ is given by

$$A(f) = \sum_{l=-\infty}^{\infty} J_l\left(\frac{2\pi k A_j}{T_l}\right) e^{jl\theta_j} \delta(f - lf_j) \tag{3-37}$$

where $J_l((2\pi k A_j)/T_l)$ are Bessel functions of the first kind. Hence, combining (3-34), (3-35) and (3-37), the full expression for the jitter on the synchronized output when there is sinusoidal input jitter on the asynchronous input is

$$J_{IJ}(f) = F \sum_{m=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} \gamma_k \sum_{l=-\infty}^{\infty} J_l\left(\frac{2\pi k A_j}{T_l}\right) e^{jl\theta_j} \delta(f - lf_j - k\eta F - mF) + E(f - mF) \quad (3\text{-}38)$$

where the feedthrough term is

$$\sum_{m=-\infty}^{\infty} E(f - mF) = \sum_{m=-\infty}^{\infty} \frac{jA_j}{2}(e^{-j\theta_j}\delta(f + f_j - mF) - e^{j\theta_j}\delta(f - f_j - mF)) \quad (3\text{-}39)$$

It is also shown in Appendix 3H that the RMS power of the jitter when there is sinusoidal input jitter is given by

$$P = F^2\left(\frac{T_l^2}{4} + \frac{T_l^2}{12} + \frac{A^2{}_j}{2}\right) \quad (3\text{-}40)$$

that is, the same as in the no input jitter case plus the power of the feedthrough jitter term.

Appendix 3A: Glossary

T_c, f_c - period and frequency of ideal signals in general jitter discussion

$e(t)$ - jitter as a continuous time function

f_j - the frequency of sinusoidal jitter

A_j - the amplitude of jitter

$\hat{A}_j$ - the amplitude of jitter in unit intervals

θ_j - phase offset of sinusoidal jitter

$T_v(nT_c)$ - the "instantaneous" period of a jittered signal

$\phi_{ref}(t)$ - phase of ideal signal

$\phi_{jit}(t)$ - phase of jittered signal

$e(nT_c)$ - time jitter

$\hat{e}(nT_c)$ - time jitter in unit intervals

$g(n)$ - a quantity measuring the difference between the phases of the ideal and jittered signals at the time the jittered signal reaches a phase of n, equal to $-\hat{e}(nT_c)$

$p(nT_c)$ - phase jitter

$p(t)$ - continuous time phase jitter function

UI - unit interval

T, F - the period and frequency of the asynchronous input to the synchronization process

t_n - jitter between the asynchronous input and the synchronized output in the synchronization process jitter equation

T_l, f_l - period and frequency of local clock

K_n - integer number of local clock periods in nth period of synchronized output

η - ratio of T to T_l

x_n - t_n normalised to T_l

$z(t)$ - periodic, continuous time function which when sampled appropriately yields synchronization process jitter samples

$Z(f)$ - Fourier Transform representation of the Fourier Series of $z(t)$

$J(f)$ - jitter spectrum for the synchronization jitter

d_k - coefficients of the jitter spectrum for synchronization jitter and of the Fourier series of $z(t)$

$\eta(n)$ - time varying ratio of the jittered asynchronous input period to T_l

e_n - input time jitter on the jittered asynchronous input

υ_n - total jitter between the synchronized output when the asynchronous input has input jitter

$D(f)$ - one component of the jitter spectrum for synchronization jitter when the asynchronous input has input jitter

$E(f)$ - feedthrough component of the jitter spectrum for synchronization jitter when the asynchronous input has input jitter

$J_{IJ}(f)$ - jitter spectrum for synchronization jitter when the asynchronous input has input jitter

γ_k - coefficients of the jitter spectrum for synchronization jitter when the asynchronous input has input jitter

Appendix 3B: Jitter Approximations

In order for time jitter to be approximated by phase jitter, from (3-9) it is required that

$$|p(nT_c) - p((n-1)T_c)| \ll 1 \tag{3B-1}$$

If the phase jitter function is bandlimited to some f_b such that $T_c = \frac{1}{2f_b}$ then (3B-1) can be rewritten as

$$|p(t) - p(t - T_c)| \ll 1 \tag{3B-2}$$

Applying the Mean Value Theorem gives

$$p(t) - p(t - T_c) = T_c \cdot \dot{p}(\tau), \quad (t - T_c) < \tau < t \tag{3B-3}$$

and hence the assumption $|p(t) - p(t - T_c)| \ll 1$ requires that

$$|\dot{p}(\tau)| \ll \frac{1}{T_c} \tag{3B-4}$$

and this will be true if

$$|\dot{p}(t)|_{max} \ll f_c \tag{3B-5}$$

where $f_c = \frac{1}{T_c}$.

Appendix 3C: Derivation of the Synchronization Process Jitter Equation

It is convenient to define a new variable

$$x_n = \frac{t_n}{T_l} \tag{3C-1}$$

where x_n is a fraction of T_l and so $0 \le x_n < 1$.

The length of a given period of the asynchronous input, normalised to the local clock period, will always be made up from some whole number plus x_{n-1} and $(1 - x_n)$. As the number η is, in general, not an integer, it is possible to express η as $\eta = \lfloor \eta \rfloor + \mathrm{fr}(\eta)$. Letting $\xi = 1 - x_n$

$$0 < x_{n-1} + \xi < 2 \tag{3C-2}$$

(3C-2) can be split into two cases

- Case (i) $0 < x_{n-1} + \xi < 1$

In the first case, $\lfloor \eta \rfloor$ is provided by the appropriate number of whole periods of T_l and

$$\mathrm{fr}(\eta) = x_{n-1} + \xi \tag{3C-3}$$

giving rise to equation (3-14).

- Case (ii) $1 \le x_{n-1} + \xi < 2$

In the second case, the number of whole local clock periods is less than $\lfloor \eta \rfloor$ and the additional clock period is provided by the sum of x_{n-1} and ξ. Thus, to retain $\lfloor \eta \rfloor$ in the equation it is necessary to subtract the extra 1,

$$\mathrm{fr}(\eta) = x_{n-1} + \xi - 1 \tag{3C-4}$$

giving equation (3-15).

To combine these in a single equation, since all the quantities involved are positive, in the first case it follows from (3C-3) that

$$x_{n-1} - \mathrm{fr}(\eta) < 0 \tag{3C-5}$$

hence $u(x_{n-1} - \mathrm{fr}(\eta)) = 0$ where $u(x) = \begin{cases} 0, & x < 0 \\ 1 & x \geq 0 \end{cases}$ is the step function.

In the second case, from (3C-4)

$$x_{n-1} - \mathrm{fr}(\eta) \geq 0 \tag{3C-6}$$

hence $u(x_{n-1} - \mathrm{fr}(\eta)) = 1$.

It is then easily verified that the following equation will generate either equation (3-14) or (3-15) as appropriate.

$$\eta = \lfloor \eta \rfloor + x_{n-1} - x_n + 1 - u(x_{n-1} + \lfloor \eta \rfloor - \eta) \tag{3C-7}$$

Iterating (3C-7)

$$x_n = n\lfloor \eta \rfloor - n\eta + x_0 + n - \sum_{i=0}^{n-1} u(\lfloor \eta \rfloor - \eta + x_i) \tag{3C-8}$$

Using the fact that $\lfloor x_n \rfloor = 0$ means that

$$0 = n\lfloor \eta \rfloor + n - \sum_{i=0}^{n-1} u(\lfloor \eta \rfloor - \eta + x_i) + \lfloor x_0 - n\eta \rfloor \tag{3C-9}$$

and using this in (3C-8) results in the closed form

$$x_n = x_0 - n\eta - \lfloor x_0 - n\eta \rfloor \tag{3C-10}$$

Appendix 3D: Stability of the Synchronization Process Jitter Equation

Equation (3-13) can be rewritten as

$$t_n = T_F(n) - T + t_{n-1} \tag{3D-1}$$

where $T_F(n) = K_n T_l$.

Iterating (3D-1) and rearranging gives

$$t_n - t_0 = \sum_{i=1}^{n} T_F(i) - nT \tag{3D-2}$$

Since, by definition $0 \le t_n < T_l$

$$0 \le \left| \sum_{i=1}^{n} T_F(i) - nT \right| < T_l \tag{3D-3}$$

Multiplying through (3D-3) by $\frac{1}{n}$ and taking the limit for $n \to \infty$ gives

$$\lim_{n \to \infty} \frac{1}{n} \left| \sum_{i=1}^{n} T_F(i) - nT \right| = 0 \tag{3D-4}$$

but $\lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} T_F(i)$ is the average output period. Hence the average output period is T in the limit.

Appendix 3E: RMS Jitter Power of the Synchronization Process Jitter

The RMS jitter power is calculated from

$$P = \int_{-\infty}^{\infty} J(f)J^*(f)df \tag{3E-1}$$

Substituting for $J(f)$ from (3-25) and simplifying gives

$$P = F^2 \sum_{k=-\infty}^{\infty} |d_k|^2 = F^2\left(\frac{T_f^2}{4} + 2\frac{T_f^2}{4\pi^2}\sum_{k=1}^{\infty}\frac{1}{k^2}\right) \tag{3E-2}$$

$\sum_{k=1}^{\infty}\frac{1}{k^2} = \frac{\pi^2}{6}$ is the Riemann Zeta function for zeta equals 2 and (3E-2) becomes

$$P = F^2\left(\frac{T_f^2}{4} + \frac{T_f^2}{12}\right) \tag{3E-3}$$

Appendix 3F: Derivation of the Synchronization Process Jitter Equation with Input Jitter

Defining the new variable $x_n = t_n/T_l$, it follows from Appendix 3C that the two forms of (3-29) and (3-30) can be written in a combined form as

$$\eta(n) = \lfloor \eta(n) \rfloor + x_{n-1} - x_n + 1 - u(x_{n-1} - \mathrm{fr}(\eta(n))) \tag{3F-1}$$

where $\eta(n) = \eta + \frac{e_n}{T_l} - \frac{e_{n-1}}{T_l}$.

Iterating (3F-1) and expanding $\eta(n)$ gives

$$x_n = x_0 + \sum_{i=1}^{n} \lfloor \eta(i) \rfloor - n\eta - \left(\sum_{i=1}^{n} \frac{e_i}{T_l} - \sum_{i=1}^{n} \frac{e_{i-1}}{T_l} \right) + n - \sum_{i=1}^{n} u(x_{i-1} - \mathrm{fr}(\eta(i))) \tag{3F-2}$$

The two summations involving the input jitter on the asynchronous input cancel out simplifying (3F-2) to

$$x_n = x_0 + \sum_{i=1}^{n} \lfloor \eta(i) \rfloor - n\eta - \frac{e_n}{T_l} + \frac{e_0}{T_l} + n - \sum_{i=1}^{n} u(x_{i-1} - \mathrm{fr}(\eta(i))) \tag{3F-3}$$

Then, using the same approach as in Appendix 3C, the closed form can be written as

$$x_n = x_0 - n\eta - \frac{e_n}{T_l} + \frac{e_0}{T_l} - \left\lfloor x_0 - n\eta - \frac{e_n}{T_l} + \frac{e_0}{T_l} \right\rfloor \tag{3F-4}$$

Appendix 3G: Stability of the Jitter Equation for a Jittered Asynchronous Input

(3-28) can be rewritten

$$\eta(n)T_l = t_{n-1} + T_F(n) - t_n \tag{3G-1}$$

where $\eta(n)T_l = \eta T_l + e_n - e_{n-1}$ is the varying input period of the jittered asynchronous input and $T_F(n) = K_n T_l$ is the period of the output from the synchronizer. From Appendix 3D it follows that the long term average of $T_F(n)$ is equal to the long term average of $\eta(n)T_l$.

It then remains to show under what conditions the long term average of the jittered asynchronous input period will be equal to the long term average of the ideal asynchronous input period. Iterating (3-27) gives

$$\sum_{i=1}^{n} \eta(i) = n\eta + \frac{e_n - e_0}{T_l} \tag{3G-2}$$

If it is assumed that the input jitter is bounded, that is, $\kappa_1 \le \frac{e_n}{T_l} \le \kappa_2$ for κ_1, κ_2 finite constants, then

$$0 \le \left| \sum_{i=1}^{n} \eta(i) - n\eta \right| \le |\kappa_2 - \kappa_1| \tag{3G-3}$$

Dividing through (3G-3) by $\frac{1}{n}$ and taking the limit as $n \to \infty$ shows that

$$\lim_{n \to \infty} \sum_{i=1}^{n} \eta(i) = \eta \tag{3G-4}$$

that is, as long as the jitter on the asynchronous input is bounded, the long term average of the jittered asynchronous input period is the period of the ideal asynchronous input and thus the long term average of $T_F(n)$ is also η.

Appendix 3H: Spectrum of the Jitter Equation for a Jittered Asynchronous Input

Following the approach in Section 3.5, the total jitter on the synchronized output, υ_n, can be considered as a sampled version of $\upsilon(\tau)$ where $\tau = t + e(t)$ for $t = nT$

$$\upsilon(\tau) = d(\tau) + e(\tau - e(t)) \tag{3H-1}$$

It is possible to write

$$\mathcal{F}\{\upsilon_n\} = F \sum_{n=-\infty}^{\infty} (D(f - nF) + E(f - nF)) \tag{3H-2}$$

where $\mathcal{F}\{\upsilon(\tau)\} = D(f) + E(f)$.

The most complex part of the analysis is finding $D(f)$, the Fourier Transform of $d(\tau)$ which is a function of time and of the input jitter function $e(t)$.

$$d(\tau) = t_0 + e_0 - \tau - \left\lfloor \frac{t_0}{T_l} + \frac{e_0}{T_l} - \frac{\tau}{T_l} \right\rfloor T_l \tag{3H-3}$$

$d(\tau)$ can be written as a Fourier Series

$$d(\tau) = \sum_{k=-\infty}^{\infty} \gamma_k exp\left(\frac{j2\pi k}{T_l}\tau\right) = \sum_{k=-\infty}^{\infty} \gamma_k exp\left(\frac{j2\pi k}{T_l}(t + e(t))\right) \tag{3H-4}$$

where

$$\gamma_k = \begin{cases} \dfrac{T_l e^{-j2\pi k\left(x_0 + \frac{e_0}{T_l}\right)}}{j2\pi k}, & k \neq 0 \\ \dfrac{T_l}{2}, & k = 0 \end{cases} \tag{3H-5}$$

Taking the Fourier Transform of (3H-4) gives

$$D(f) = \sum_{k=-\infty}^{\infty} \gamma_k \delta(f - k\eta F) \otimes \mathcal{F}\left\{exp\left(\frac{j2\pi k}{T_l}e(t)\right)\right\} \tag{3H-6}$$

where $\otimes$ is the convolution operator. If $\mathcal{F}\left\{exp\left(\frac{j2\pi k}{T_l}e(t)\right)\right\} = A(f)$ then

$$D(f) = \sum_{k=-\infty}^{\infty} \gamma_k A(f - k\eta F) \tag{3H-7}$$

When the input jitter is sinusoidal, it is possible to find a closed form for (3H-7). If $e(t)$ is a sinusoid with some initial phase offset, θ_j, then

$$A(f) = \mathcal{F}\left\{exp\left(\frac{j2\pi k}{T_l}A_j \sin(2\pi f_j t + \theta_j)\right)\right\} \tag{3H-8}$$

and using the identity

$$exp(j\lambda \sin(x)) = \sum_{l=-\infty}^{\infty} J_l(\lambda) e^{jlx} \tag{3H-9}$$

where $J_l(\lambda)$ are Bessel functions of the first kind, $A(f)$ can be written in a closed form.

$$\begin{aligned} A(f) &= \mathcal{F}\left\{\sum_{l=-\infty}^{\infty} J_l\left(\frac{2\pi k A_j}{T_l}\right) exp(jl(2\pi f_j t + \theta_j))\right\} \\ &= \sum_{l=-\infty}^{\infty} J_l\left(\frac{2\pi k A_j}{T_l}\right) e^{jl\theta_j} \delta(f - l f_j) \end{aligned} \tag{3H-10}$$

Putting $A(f)$ for the sinusoidal input jitter case, into (3H-7) gives

$$D(f) = \sum_{k=-\infty}^{\infty} \gamma_k \sum_{l=-\infty}^{\infty} J_l\left(\frac{2\pi k A_j}{T_l}\right) e^{jl\theta_j} \delta(f - l f_j - k\eta F) \tag{3H-11}$$

In the case of sinusoidal input jitter, the calculation of $E(f)$, the feedthrough term, is straightforward,

$$E(f) = \frac{jA_j}{2}(e^{-j\theta_j}\delta(f + f_j) - e^{j\theta_j}\delta(f - f_j)) \tag{3H-12}$$

As in Appendix 3E, the RMS power can be calculated from

$$P = \int_{-\infty}^{\infty} J_{IJ}(f) J_{IJ}^{*}(f) df \tag{3H-13}$$

For the particular case of sinusoidal input jitter, substituting from (3-38) for $J_{IJ}(f)$ and simplifying gives

$$P = F^2\left(\sum_{k=-\infty}^{\infty} |d_k|^2 \cdot \sum_{l=-\infty}^{\infty} \left| J_l\left(\frac{2\pi k A_j}{T_l}\right)\right|^2 + \frac{A^2{}_j}{2}\right) \tag{3H-14}$$

Expanding (3H-14) gives

$$P = F^2\left(\frac{T_l^2}{4}\left(1 + \frac{2}{\pi^2}\sum_{k=1}^{\infty}\frac{1}{k^2}\right) \cdot \left(J^2{}_0\left(\frac{2\pi k A_j}{T_l}\right) + 2\sum_{l=1}^{\infty} J^2{}_l\left(\frac{2\pi k A_j}{T_l}\right)\right) + \frac{A^2{}_j}{2}\right) \tag{3H-15}$$

but $J^2{}_0(x) + 2\sum_{l=1}^{\infty} J^2{}_l(x) = 1$ and so (3H-15) simplifies to

$$P = F^2\left(\frac{T_l^2}{4} + \frac{T_l^2}{12} + \frac{A^2{}_j}{2}\right) \tag{3H-16}$$

Chapter 4
Jitter Generation in External Timing Source Injection

4.1 Introduction

For correct operation, nodes in a communications network need to share a common timing reference [4], [52]. Without a common reference it is difficult to provide certain types of services such as constant bit rate services. A timing reference is transported in the network, for example, as a framing signal and is usually sourced from an accurate external clock [8]. An external timing source has to be injected into the network at some point and as it is desirable that the transported timing reference be relatively jitter-free, the process of injecting the timing source should not introduce or generate excessive jitter.

This chapter presents an analysis of two different methods of accomplishing the external timing source injection. The first is a conventional method of injection, analysed using a new time domain approach developed by the author and which yields exact expressions for frame-level jitter [62]. The second is a new method of timing injection developed and analysed for the first time by the author [60], [62].

Section 4.2 is an introduction to timing adjustment, jitter and the use of external timing sources in networks. Section 4.3 describes the conventional stuffing method and analyses the jitter produced using a new time domain approach which yields exact expressions for the jitter on the injected external timing source. Section 4.4 applies the analysis of jitter generation due to the process of synchronization developed in Section 3.4 of Chapter 3 to the new synchronization method of external timing source injection. In Section 4.5, the results of the jitter analyses for the two approaches to external timing source injection are compared and both are compared with waiting time jitter. Issues relating to waiting time jitter and how the choice of parameters can minimise the waiting time jitter are then reviewed.

4.2 Timing Reference Transfer and Jitter in Digital Communications Networks

In digital communications networks, recovery of the transmitted information is dependent upon accurate timing linked to the data in order to decode the stream of bits correctly. Thus there is a need to transport and recover clock information. Timing information is often transported as part of the encoding of the data and then recovered

using special clock recovery circuitry. The clock recovery circuitry must be designed to minimise alignment jitter so as to avoid bit errors [13].

A telecommunications node usually has a local clock which provides the source of timing for the processing within the node and often also for the output of the data stream from the node [4], [52]. When data enters the node, it will be written into a buffer using the recovered clock and then read out from the buffer for processing. If the local clock is used for this purpose a certain amount of frequency variation between the two clocks can be absorbed by the buffer but, if the clocks are unrelated, slips in the buffer and consequent loss of data are inevitable. Alternatively, if a filtered version of the input clock is used to clock the data from the buffer, jitter reduction can be achieved [13].

High speed transmission is possible in digital networks using the technique of multiplexing where many lower rate signals are interleaved and carried at the higher rate. The Plesiochronous Digital Hierarchy (PDH) uses asynchronous time division multiplexing (ATDM) to map lower rate signals onto higher rate signals for digital transmission [7]. When it is required to access the lower rate signals, for example for switching, it is necessary to demultiplex the higher rate signal. Well developed techniques for timing adjustment allow the PDH to operate with clocks that are not synchronized to each other but are plesiochronous, that is asynchronous but constrained to have a certain frequency relationship or to have a frequency within a given tolerance.

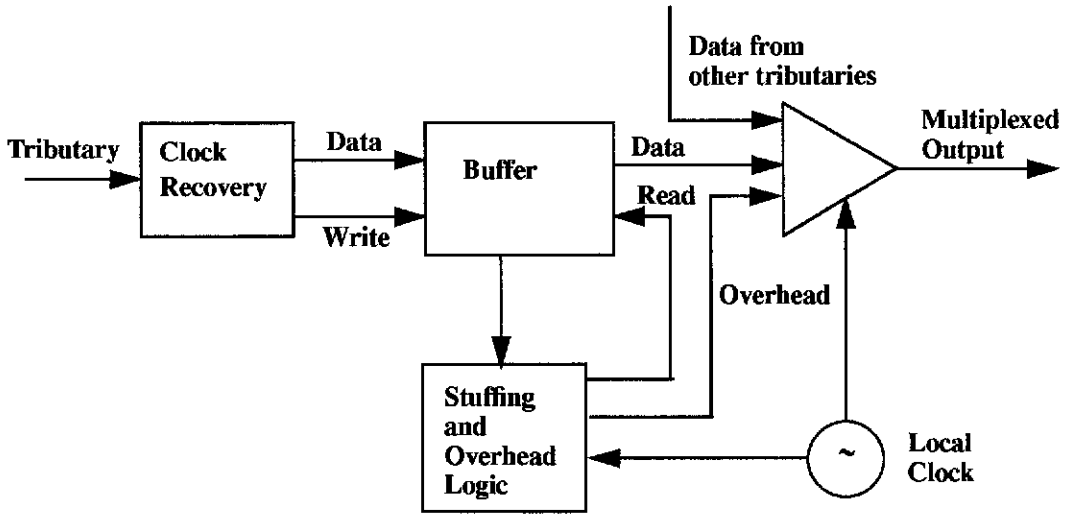


Figure 4.1: Simple block diagram of a synchronizer in a PDH multiplexer.

When a number of lower rate tributaries are to be multiplexed to a higher rate, the higher rate is greater than the simple sum of the tributary rates. A higher multiplex channel rate is required to provide overhead bits such as those needed for framing, control, alarm and error checking functions. A simple block diagram of a synchronizer used in multiplexing is in Figure 4.1 [13]. It shows that the tributary data is written to a buffer at the rate of the recovered tributary clock. Tributary data is read from the buffer using a gapped clock generated from the local clock of the multiplexing node. The clock has gaps where overhead bits and data from the other tributaries are inserted onto the higher rate output. However, in order for there to be no data loss in the buffer, either from underflow or overflow, the gapped clock should be the same long term average frequency as the tributary clock and as has been noted above, in general, it is generated from an independent source. The problem is dealt with by a timing adjustment mechanism known as pulse stuffing or justification.

In such systems, the most commonly used form of justification is positive stuffing. In positive stuffing, the number of data bits from the tributary into the higher rate frame can be reduced by one, the data bit being replaced by a dummy or stuff bit. To be sure that positive stuffing only will operate, it is necessary that the multiplex channel rate be guaranteed to be greater than the tributary rate and this is achieved through the frame structure and by restricting the tolerance allowed on the tributary and local clocks. As a result, the effective read clock at the buffer is faster than the write clock

at the buffer and so the buffer tends to empty. When the level of the buffer falls below a threshold a positive stuff is made where, instead of taking a data bit from the buffer, the stuff bit is inserted into the frame. The adjustment is made as often as necessary for the input and output rates at the buffer to be maintained equal.

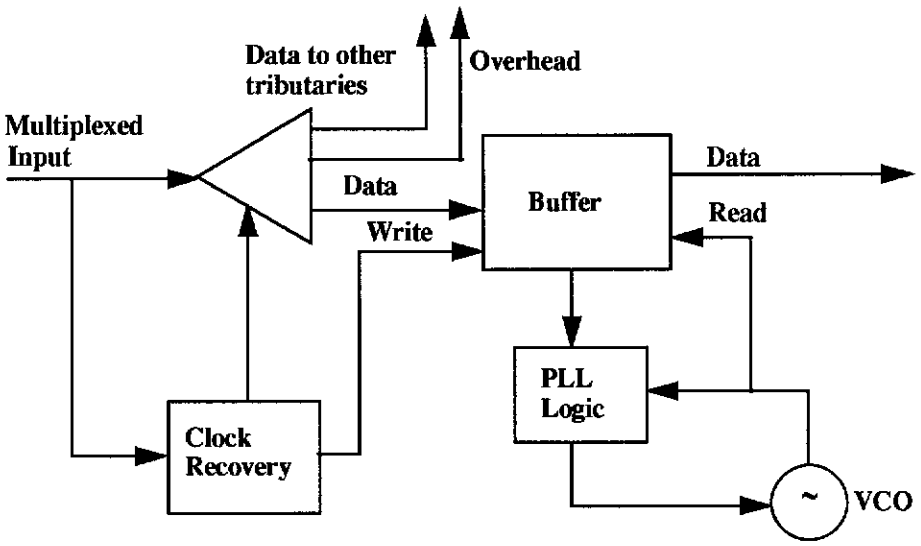


Figure 4.2: Simple block diagram of a desynchronizer in a PDH demultiplexer.

At the end of a digital transmission link, to access the signals carried within the high rate multiplexed stream, demultiplexing is required. A simple block diagram of a desynchronizer unit within the demultiplexing equipment is in Figure 4.2 [13]. The clock recovered from the data stream and used to write the data to the desynchronizer buffer is exactly the same as the clock produced at the output of the synchronizer and which contains the stuff information. The key requirement is to regenerate the tributary clock which is done using the buffer fill to control a phase lock loop.

An example to illustrate typical frequency relationships and stuffing rates will be given based on the ITU-T Recommendation G.742 for a second-order multiplex to combine four 2048 kbit/s streams into an 8448 kbit/s stream [7], [47]. The frame format is irregular, that is, the ratio of data to overhead in the frame is not an integer but is equal to $206/6 = 34.33333$. With the write clock frequency, f_w, equal to 2.048 MHz and the multiplex channel (data) rate, which is effectively the read clock at the data buffer, f_r, equal to $2.112\text{MHz} \times 206/212$, then the fixed justification rate

is +4226.41 Hz. That is, even with the clocks at their nominal values, a fixed amount of positive stuffing is required. With clock tolerances of ±50ppm on the write clock and ±30ppm on the read clock, the actual range of justification rates is +4062.44 Hz to +4390.38 Hz. The maximum stuffing rate occurs if there is a stuff in every frame and so is equal to the frame rate of $F = 2.112/212 = 9.962\text{kHz}$. An important parameter is the stuff ratio which is defined as the ratio of the possible justification rates to the maximum justification rate $\rho = (f_r - f_w)/F$ and falls in the range +0.41 to +0.44. Use of the stuff ratio parameter, normalised to the maximum possible justification rate, allows comparison of the performance of justification schemes implemented in different systems with different clock frequencies.

The operations of multiplexing and demultiplexing generate jitter which can affect the tributary timing signal. A number of different kinds of jitter are produced. The jitter components due to the gapping required to insert overhead bits are predictable and relatively high frequency components which can be easily removed in the desynchronizer, as can the component, often called justification jitter, at the mean justification rate [47]. Finally, there is a problematic component, which can be at arbitrarily low frequency, called waiting time jitter [22], [47], [48].

Other forms of jitter can also affect digital transmission. Some of these such as transmission link jitter and wander and clock oscillator jitter are usually relatively insignificant. However, when a transmission link has to pass through a number of repeaters, there is a problem of jitter accumulation. There are two types of jitter which accumulate: random jitter, which as its name suggests is uncorrelated and so only accumulates slowly, and systematic jitter which can accumulate significantly from repeater to repeater. After a certain number of repeaters, the systematic jitter can be enough to cause not only bit errors but also slips in the buffers of the multiplex-demultiplexers [13].

One way to eliminate the effect of systematic jitter accumulation is the use of autonomous nodes with independent clocks such as used in the 802.6 DQDB MAC protocol [4], [52]. In this scheme, a local clock at each node allows the generation of a clean outgoing frame. Data arriving at the node will be written to a buffer at the rate of the upstream node's clock and read from the buffer at the local clock rate and,

although both clocks will be within a certain tolerance of a nominal rate, without the application of a stuffing algorithm, buffer slips (overflow or underflow) will occur.

The stuffing algorithm is different from that described for the PDH as it has provision for stuffing at the end of the frame in the form of stuff bytes and the stuffing mechanism varies the length of the frame [4], [63]. If the buffer fill falls below a given threshold, positive stuffing is required and a stuff byte is added to the frame, thereby lengthening it; whereas if the buffer fill rises above a different threshold, negative stuffing is required and a stuff byte is deleted from the frame, thereby shortening it. If the buffer fill is between thresholds and no stuffing is required, the number of stuff bytes is left unaltered. In this method, the number of data bytes in the frame is constant. Both positive and negative stuffing are possible in the method since the frequency difference between two successive nodes can be in either direction.

In the operation of this stuffing method in a large network with many nodes, it may be the case that the frequency differentials between successive node clocks are in the same direction for several nodes, hence the frame will become progressively longer or shorter as stuffing occurs in the same direction in each node. There is therefore provision in the algorithm for maximum and minimum frame lengths, when these are reached a node cannot stuff so as to exceed these limits but must wait for the next frame which is inside the limits [63]. A desynchronizer is also not used in this method, as the local clock in the node is used to generate the new frame for the outgoing signal. Although, as previously discussed, use of the local clock prevents the build-up of systematic repeater bit jitter, it does not prevent the build-up of frame level jitter as the jitter in the frame is carried by the varying frame length due to the stuffing [63].

It is possible to show that the two methods of stuffing discussed are equivalent and this will be done formally later in this chapter. However, here it can be noted that in both cases the outgoing data rate is varied by the use of stuffing. In the PDH, a fixed length frame carries a varying number of data bits; in the DQDB method, a fixed number of data bytes is carried by a frame of varying length. As a result, in each method, the long term average outgoing data rate is made equal to the incoming data rate.

4.2.1 External Timing Source Injection

When a timing reference is required in a network, if it is not to be carried by a separate wire, it must be carried as part of the ordinary transmission system. In a shared media data network such as the DQDB [4], it may be carried by the framing signal. However, as has been discussed above, stuffing introduces jitter to the frame, which, after a series of nodes, and depending upon the frequency differentials involved, may build up to high levels and contain undesirable low frequency components [63]. Hence, the transfer of a frame rate timing reference signal of sufficiently good quality is a particularly demanding function. An external timing source (XTS) can be injected into the node to generate a clean frame which can be used as a timing reference. When the XTS is supplied by a public network, information exchange with the public network becomes possible.

When injecting an XTS connection, there are two problems to be solved, firstly there is the obvious task of locking the frame timing to the XTS so that the frame can transmit the timing reference. Secondly and somewhat less obviously, as the XTS is asynchronous to the local clock of the node where it is injected, synchronization is required for the XTS to be accessible in the timing domain of the local clock.

Two methods of locking the frame timing to an XTS connection are considered. The first approach is the conventional method which requires a buffer and control blocks to monitor buffer fill and implement stuffing. The requirement for synchronization is less obvious in this approach, but nevertheless exists. Measuring the buffer fill requires comparison of the positions of the write pointer, in the timing domain of the XTS, and the read pointer, in the timing domain of the local clock. Therefore, the design for the conventional method of XTS injection will necessarily incorporate a synchronizer to allow the communication between the two timing domains to take place. The second approach to be presented is a new and very straightforward method, the use of a frame sampling synchronizer which performs generation of a clean frame and synchronization in a single operation. With this approach, the overall design of the locking method is much simpler than the conventional method. The problem of metastable failure and the design of synchronizers which provide an adequate MTBF was dealt with in Chapter 2.

4.3 The Conventional Stuffing Approach to XTS Injection

For the conventional stuffing approach to injecting an XTS, a signal is needed, not only at the nominal frame rate, but also at the higher rate of the data units to provide the write clock at the buffer. An XTS could be available at both of these frequencies, or at only one, in which case it would need to be multiplied up or divided down to generate the other timing signal. Alternatively, it could be available at an unrelated frequency and a combination of these techniques would be necessary. The particular details of these methods are beyond the scope of this work. For the purpose of the analysis to be performed in this section, it will be assumed that the XTS is available as a frame rate signal and that there is also available a write clock derived directly from the XTS such that if the XTS has period T_{XTS} and frequency F_{XTS}, the write clock generated from it has period T_w and frequency f_w where

$$T_{XTS} = NT_w \tag{4-1}$$

The frame follows the format discussed in Section 4.2 for the DQDB stuffing scheme. It is thus modelled as consisting of N bytes, the first D of which are considered as data, followed by a number of stuffing bytes. N and D are constants. In the normal operation of the stuffing algorithm, as a frame passes through nodes in the network, the operation of stuffing varies the number of stuffing bytes in a frame by either positive or negative stuffing within an allowed range. In the application of external timing injection, however, the frame generated from the XTS always has S_{nom}, the nominal number of stuffing octets which lies midway between the constraints, thus

$$N = D + S_{nom} \tag{4-2}$$

and it is possible to stuff in both directions. The frame is read out of the buffer using the local clock with period T_r and frequency f_r which will, in general be different from the period and frequency of the write clock, but within a tolerance which will allow stuffing to adjust the one to the other.

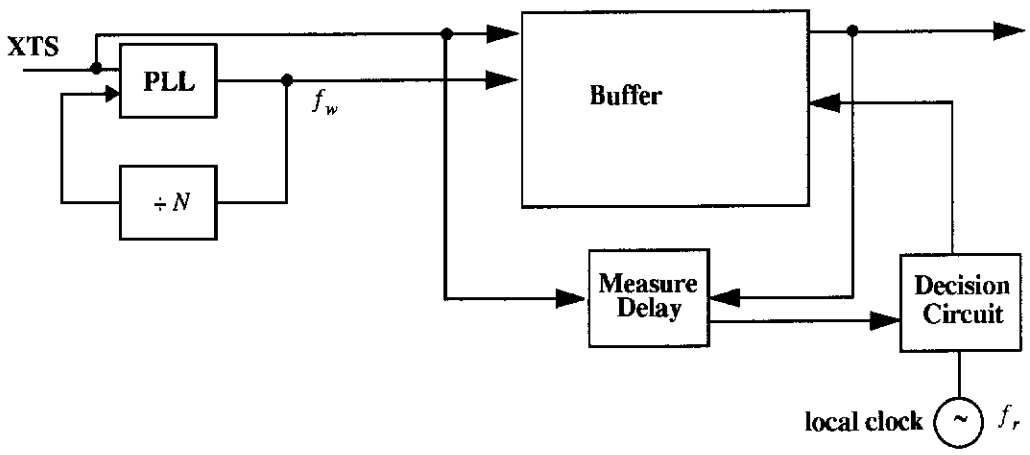


Figure 4.3: Conventional stuffing synchronizer.

Consider two possible methods of operating the stuffing algorithm. The first, which is depicted in Figure 4.3, operates by direct measurement of the time delay from the XTS entering the buffer to the synchronized XTS leaving it. If this delay is $< t_{min}$, which is some minimum value, then a decision to positive stuff is made. If the delay is $\geq t_{max}$, the upper threshold, then a decision is made to negative stuff. A second method is to measure the buffer fill at the time the XTS enters the buffer and compare it with two thresholds, if it is below some minimum fill level, positive stuffing is performed and if above a maximum then negative stuffing is necessary. The relationship between the two approaches is as follows, if τ is the delay between the XTS entering and the synchronized XTS leaving the buffer as measured in the first method, then the buffer fill, as found by the second method, is numerically equal to $\lfloor \tau / T_r \rfloor$ which means that either method can be used when making a decision about stuffing.

4.3.1 The Jitter Equation for the Conventional Stuffing Synchronizer

The jitter generated by a conventional stuffing approach to injecting an external timing source can be analysed in the time domain using direct consideration of the time delays involved. This approach draws on some of the techniques employed by

Pierobon and Valussi, who use phase differences to analyse jitter caused by stuffing [48], and results in the same equations.

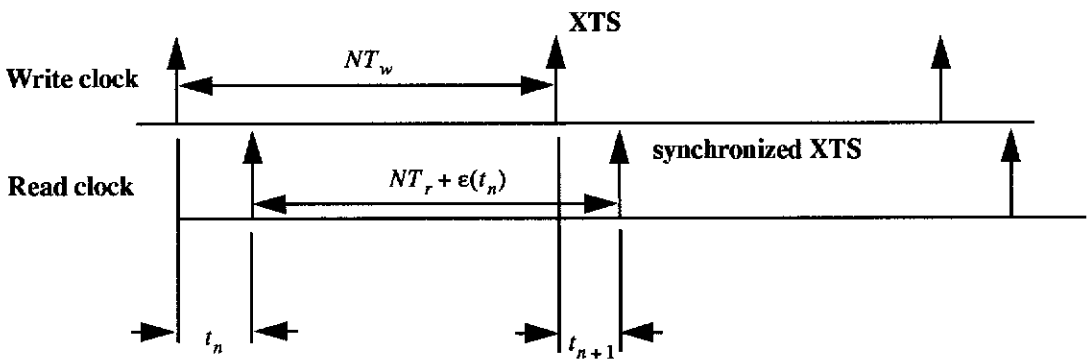


Figure 4.4: Conventional stuffing synchronizer jitter diagram.

Figure 4.4 shows the relationship in the time domain between the XTS on the write side of the buffer and the synchronized XTS on the read side of the buffer, which can be expressed as

$$t_{n+1} = t_n + NT_r - NT_w + \varepsilon(t_n) \tag{4-3}$$

where t_n is the delay between the XTS and the synchronized XTS and $\varepsilon(t_n)$ represents the stuffing, that is

$$\varepsilon(t_n) = \begin{cases} T_r, & t_n < t_{min} \\ -T_r, & t_n \geq t_{max} \\ 0, & t_{min} \leq t_n < t_{max} \end{cases} \tag{4-4}$$

Note that $\varepsilon(t_n)$ only changes at that place in the frame where stuffing can occur and it holds the value corresponding to the stuffing decision until the beginning of the next frame when it becomes zero.

It is shown in Appendix 4B that the jitter equation for the conventional stuffing synchronizer is stable, that is, the average outgoing period equals T_{XTS} where

$$T_{XTS} = \overline{N} T_r \tag{4-5}$$

for some $\overline{N}$ which is, in general, not an integer. In Appendix 4C it is shown that (4-3) can be written as

$$y_{n+1} = y_n - \rho + 1 - u(y_n - y_{min}) - u(y_n - y_{max}) \tag{4-6}$$

where the stuffing function has been replaced using the unit step function $u(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$, the equation has been rewritten using the new variable $y_n = t_n / T_r$ and with the stuffing ratio given by

$$\rho = \frac{f_r - f_w}{F_{XTS}} \tag{4-7}$$

It is then further shown in Appendix 4C that $y_n \in [L, U)$ where

$$\begin{aligned} L &= y_{min} - \rho \\ U &= y_{max} - \rho \end{aligned} \tag{4-8}$$

and this is used to rewrite (4-6) in closed form as

$$y_n = y_0 - n\rho - \lfloor y_0 - n\rho - L \rfloor \tag{4-9}$$

4.3.2 Measurement Quantization

In reality, both the thresholds and the measured value of y_n are examined and compared in a digital system, hence the system is only aware of their quantized values. When quantization is taken into account, it can be seen that it is necessary for the stuffing thresholds to be asymmetric as in equation (4-4). With asymmetric thresholds, positive stuffing occurs for y_n strictly less than the lower threshold, y_{min}, but negative stuffing occurs for y_n greater than or equal to the upper threshold, y_{max}.

Consider the case where the quantization step size is the period of the local clock, T_r and the thresholds, which are a unit apart, that is $y_{max} = y_{min} + 1$, lie on the quantization boundaries. This is illustrated below in Figure 4.5.

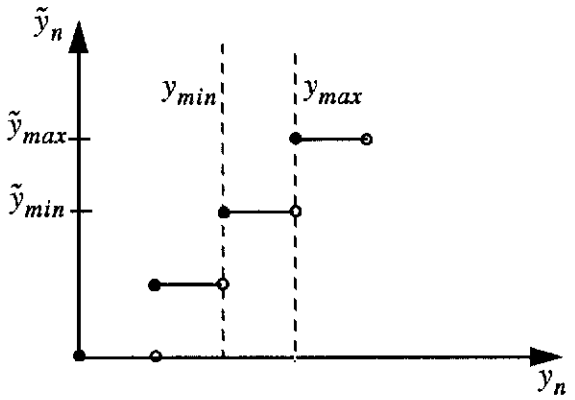


Figure 4.5: Thresholds on quantization boundaries.

Let $\tilde{y}_n$, $\tilde{y}_{min}$ and $\tilde{y}_{max}$ denote the quantized values and note that $\tilde{y}_{min} = y_{min}$ and $\tilde{y}_{max} = y_{max}$. Any value of y_n which lies between the thresholds will be quantized to $\tilde{y}_{min}$. If positive stuffing were to be allowed to occur for $\tilde{y}_n = y_{min}$ then consecutive positive and negative stuffing could occur, for example, if the quantized value of $y_n - \rho + 1$ is $\geq y_{max}$. Similarly, if ρ were negative then positive stuffing could occur unnecessarily when y_n is between the thresholds. The severity of this problem would diminish with an increase in the number of quantization levels, but it is avoided altogether by making the thresholds asymmetric.

It is shown in Appendix 4D that the effect of quantization is to make the limits of the jitter equation the same whether the thresholds lie on or between quantization boundaries. Recall that buffer fill rather than time delay can be used as the measure for deciding whether or not to stuff. Buffer fill is given by $\lfloor t_n/T_r \rfloor$ and so it is equivalent to operating in the quantized domain as described in this section.

Assuming therefore, that the thresholds are on quantization boundaries, L and U are as given in (4-8) and clearly for $\rho > 0$, only the lower threshold will be crossed and

only positive stuffing will occur; the converse is true for $\rho < 0$. It has already been seen that y_n can get arbitrarily close to L, so let $y_0 = L$ to simplify equation (4-9) to

$$y_n = L - n\rho + \lceil n\rho \rceil \tag{4-10}$$

where $\lceil x \rceil$ = least integer $\geq x$.

(4-10) can be written in terms of absolute time, with $t_n = y_n T_r$, as

$$t_n = LT_r - n\rho T_r + \lceil n\rho \rceil T_r \tag{4-11}$$

Equation (4-11) is the jitter of the synchronized XTS, that is the variation in its position in time relative to an ideal equally spaced reference for the conventional stuffing synchronizer.

4.3.3 Jitter Spectrum of the Conventional Stuffing Synchronizer

The Fourier Transform of the sequence generated by equation (4-11) will give the spectrum of the jitter on the synchronized XTS introduced by the conventional stuffing synchronizer. It is given in (4-12) and (4-13) below. The derivation is in Appendix 4E.

$$\begin{aligned} J_{CSS}(f) &= \mathcal{F}\{t_n\} \\ &= F_{XTS} \sum_{k=-\infty}^{\infty} b_k \sum_{n=-\infty}^{\infty} \delta(f - k\rho F_{XTS} - nF_{XTS}) \end{aligned} \tag{4-12}$$

where

$$b_k = \begin{cases} L + \dfrac{T_r}{2}, & k = 0 \\ \dfrac{T_r}{j2\pi k}, & k \neq 0 \end{cases} \tag{4-13}$$

The spectrum of the jitter introduced by the conventional stuffing approach to XTS injection contains some of the same terms as the spectrum of the jitter generated by

pulse stuffing in PDH multiplexing. In particular, both systems generate the so-called waiting time jitter [22], [48]. This issue will be discussed in more detail in Section 4.5 where comparison will also be made with the spectrum of the jitter introduced by the frame sampling synchronizer.

4.4 The Frame Sampling Synchronizer Approach to XTS Injection

The conventional stuffing approach to XTS injection is relatively complicated as it requires buffers and a control block to carry out the stuffing algorithm. As the write and read sides of the buffer are in asynchronous timing domains, a synchronizer is still required in order to make the necessary buffer fill or delay measurement.

The frame sampling synchronizer approach, by contrast, is very simple as it generates a clean frame and performs the synchronization in a single operation. All that is required for the frame sampling synchronizer approach is a well-designed synchronizer which provides an adequate MTBF to handle the problem of metastability.

4.4.1 Effect of Different Synchronizer Designs

The details of synchronizer designs and their relative performance were covered in Section 2.3 of Chapter 2; the expressions for the data delay time for each synchronizer were also given. In Section 3.4 of Chapter 3, the equation for the jitter on the synchronized output was derived based on the time domain relationship between the synchronized output and the asynchronous input due to the synchronization process, without reference to the detail of the synchronizer implementation. In this section, it will be shown that the jitter equation for a synchronizer does not depend on the particular form of the synchronizer and the expression for its data delay time. Note also that the jitter analysis of Section 3.4 of Chapter 3 ignored the occurrence of metastability. It was possible to do this because, as explained in Chapter 2, the synchronizer will be designed to ensure that the probability of metastable failure is acceptably low.

All synchronizer designs can be reduced to the generalised design of a synchronizer shown in Figure 4.6. The initial synchronization is made by the input flip-flop when the asynchronous input is clocked in by the local clock. The design will then provide an adequate settling time to reduce the probability of metastable failure to an acceptable level. Note however that the physical separation between the input flip-flop and the block providing the settling time in the figure is only for conceptual purposes. In some designs detailed in Section 2.3 of Chapter 2 the provision of the settling time may physically involve the input flip-flop and in all designs the settling time starts from the clock edge at the input flip-flop. Finally, there is an output flip-flop. Recall from Section 2.3.1 of Chapter 2 that metastable failure is considered to have occurred if the output flip-flop samples a metastable state hence the output flip-flop constitutes a separate block in the design concept.

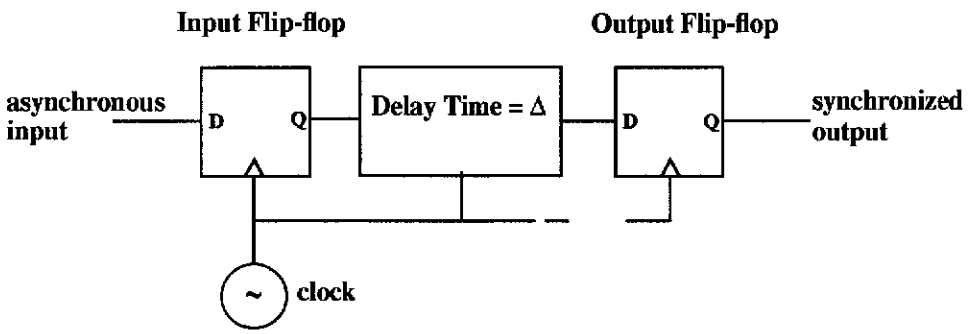


Figure 4.6: Generalised design of a synchronizer.

The timing diagram corresponding to Figure 4.6 is shown in Figure 4.7. It shows that the time from the asynchronous input arriving at the input flip-flop to the appearance of the synchronized output, denoted $t_d(n)$, is made up of three main components. Firstly, there is the varying time interval between the arrival of the asynchronous input and the next local clock edge, denoted t_n, secondly there is the data delay time of the particular synchronizer design, denoted $\Delta - t_{po}$ and thirdly there is the output propagation delay of the output flip-flop, denoted t_{po}.

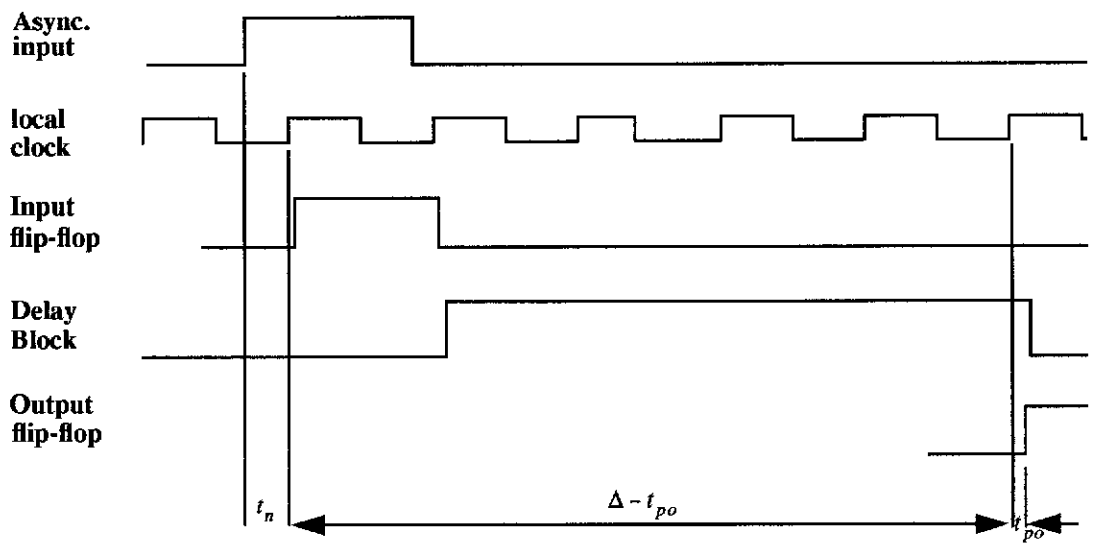


Figure 4.7: Timing diagram for the generalised synchronizer design.

From Figure 4.7,

$$t_d(n) = t_n + (\Delta - t_{po}) + t_{po} \tag{4-14}$$

It is assumed that the data delay time due to any synchronizer design remains constant, both because the flip-flop and other device characteristics are assumed to remain constant and because it is assumed that there is no jitter on the local clock, so that T_l the period of the local clock is constant. If the flip-flop characteristics are assumed constant, then t_{po} can also be assumed constant. As a result of these assumptions, the only varying element in (4-14) is the interaction between the asynchronous input and the rising clock edge, that is, t_n. It is clear then that the particular form taken by the data delay time of the synchronizer is not relevant to the derivation of the jitter equation. In fact, the jitter introduced by a synchronizer is due solely to the interaction between the time of arrival of the asynchronous input and the next edge of the local clock. In the application described in the following sections, the asynchronous input is the XTS and the synchronized output is the synchronized XTS.

4.4.2 The Jitter Equation for the Frame Sampling Synchronizer

Equation (3-18) of Chapter 3, repeated below for convenience, gives t_n, the jitter on the synchronized XTS

$$t_n = t_0 - n\eta T_l - \left\lfloor \frac{t_0}{T_l} - n\eta \right\rfloor T_l \tag{4-15}$$

where T_l is the local clock period and $\eta = \frac{T_{XTS}}{T_l}$.

4.4.3 Jitter Spectrum of the Frame Sampling Synchronizer

In Section 3.5 of Chapter 3, the spectrum of the jitter on the synchronized XTS is derived. It is shown in (4-16) and (4-17).

$$\begin{aligned} J_{FSS}(f) &= \mathcal{F}\{t_n\} \\ &= F_{XTS} \sum_{k=-\infty}^{\infty} d_k \sum_{n=-\infty}^{\infty} \delta(f - k\eta F_{XTS} - nF_{XTS}) \end{aligned} \tag{4-16}$$

where

$$d_k = \begin{cases} \frac{T_l}{2}, & k = 0 \\ \frac{T_l e^{-j2\pi k x_0}}{j2\pi k}, & k \neq 0 \end{cases} \tag{4-17}$$

4.5 External Timing Source Injection and Waiting Time Jitter

In this section, it is first shown that the two different methods of XTS injection produce equivalent jitter spectra. It is also possible to elucidate an equivalence between the parameters of the two methods. As the conventional stuffing synchronizer approach has already been shown to introduce waiting time jitter, it thus follows that

the frame sampling synchronizer approach also produces waiting time jitter. A discussion of waiting time jitter concludes the section.

The close similarity between the spectra produced by the two different approaches to XTS injection can be seen by rewriting (4-16) as follows

$$J_{FSS}(f) = F_{XTS} \sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} d_k \delta(f - k\mathrm{fr}(\eta)F_{XTS} - k\lfloor \eta \rfloor F_{XTS} - nF_{XTS}) \quad (4\text{-}18)$$

Let $m = k\lfloor \eta \rfloor + n$ then

$$J_{FSS}(f) = F_{XTS} \sum_{k=-\infty}^{\infty} d_k \sum_{m=-\infty}^{\infty} \delta(f - k\mathrm{fr}(\eta)F_{XTS} - mF_{XTS}) \quad (4\text{-}19)$$

Note that both $|\mathrm{fr}(\eta)|$ and $|\rho| < 1$. $\mathrm{fr}(\eta)$ determines the position of the spectral lines in (4-19) in the same way that ρ determines the position of the spectral lines in the jitter spectrum of the conventional stuffing synchronizer in (4-12). The two spectra have different average values, but otherwise the coefficients b_k of (4-13) and d_k of (4-17) are of the same form although d_k has a different phase.

In the case where the local clock period in the two implementations is the same, that is, $T_l = T_r$, then b_k and d_k will have the same magnitude, furthermore, in this case since $T_{XTS} = \eta T_l = \bar{N} T_r$, it follows that

$$\eta = \bar{N} \quad (4\text{-}20)$$

Recalling the definition of ρ in (4-7), if (4-20) applies then

$$\eta = N + \rho \quad (4\text{-}21)$$

If $\rho > 0$, then

$$\lfloor \eta \rfloor = N \text{ and } \mathrm{fr}(\eta) = \rho \quad (4\text{-}22)$$

and the positions of the spectral lines in (4-19) are identical to those of (4-12). If $\rho < 0$, then

$$\lceil \eta \rceil = N \text{ and } \mathrm{fr}(\eta) = 1 + \rho \tag{4-23}$$

and (4-16) can be rewritten as follows

$$J_{FSS}(f) = F_{XTS} \sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} d_k \delta(f - kNF_{XTS} - k\rho F_{XTS} - nF_{XTS}) \tag{4-24}$$

Let $m = kN + n$ then

$$J_{FSS}(f) = F_{XTS} \sum_{k=-\infty}^{\infty} d_k \sum_{m=-\infty}^{\infty} \delta(f - k\rho F - mF) \tag{4-25}$$

and the positions of the spectral lines in (4-25) and (4-12) are identical. Even when these special conditions do not obtain, comparison of the spectra in (4-12) and (4-19) makes clear the equivalence between ρ and $\mathrm{fr}(\eta)$ which means that they play the same role in determining the placing of the spectral lines and the amount of waiting time jitter which is present.

It is also possible to show that waiting time jitter is produced from the time domain jitter equation. As explained in Section 3.5 of Chapter 3, (4-15) can be produced by sampling the continuous periodic function below at the instants $t = nT_{XTS}$.

$$z(t) = t_0 - \frac{\eta t T_l}{T_{XTS}} - \left\lfloor \frac{t_0}{T_l} - \frac{\eta t}{T_{XTS}} \right\rfloor T_l \tag{4-26}$$

The period of $z(t)$ is $T_{XTS}/\eta = T_l$, thus the period of $z(t)$ and the period with which it is sampled to produce the sequence of numbers described by (4-15) are not related by an integer, since, in general, η will not be an integer. Waiting time jitter results from the sampling of a periodic function at intervals which are not integer multiples of the period of the function.

Strictly speaking, there is no waiting time jitter, as it was originally defined [22], [35], when ρ is a rational number. When $\rho = P/Q$ for P, Q co-prime integers, it can be shown [22], [48] that the spectrum of (4-12) simplifies to

$$\sum_{r=0}^{Q-1} c_r \delta\left(f - \frac{rP}{Q} F_{XTS} + \left\lfloor \frac{rP}{Q} \right\rfloor F_{XTS}\right) \tag{4-27}$$

where

$$c_r = \sum_{m=-\infty}^{\infty} b_{mQ+r} = \frac{1}{j2Q} \cot\frac{\pi r}{Q}, r = 1, \ldots, Q-1 \tag{4-28}$$

and that the spectral lines occur at multiples of $\frac{F_{XTS}}{Q}$.

In the time domain, [22] has shown that in the case of rational ρ, the phase difference between the read and write clocks at the synchronizer buffer is a relatively high frequency ramp function with no low frequency envelope. Samples of the ramp function give the jitter on the read clock at the synchronizer and because of the regularity of the ramp function, the same samples occur cyclically, thus confirming that the jitter frequency is related to the sampling frequency. However, if the value of ρ is not rational, but is close to some rational number, then a low frequency envelope becomes superimposed on the high frequency ramp function and the jitter samples do not repeat. As a result, jitter of arbitrarily low frequency can occur. These two different cases are illustrated in Figure 4.8.

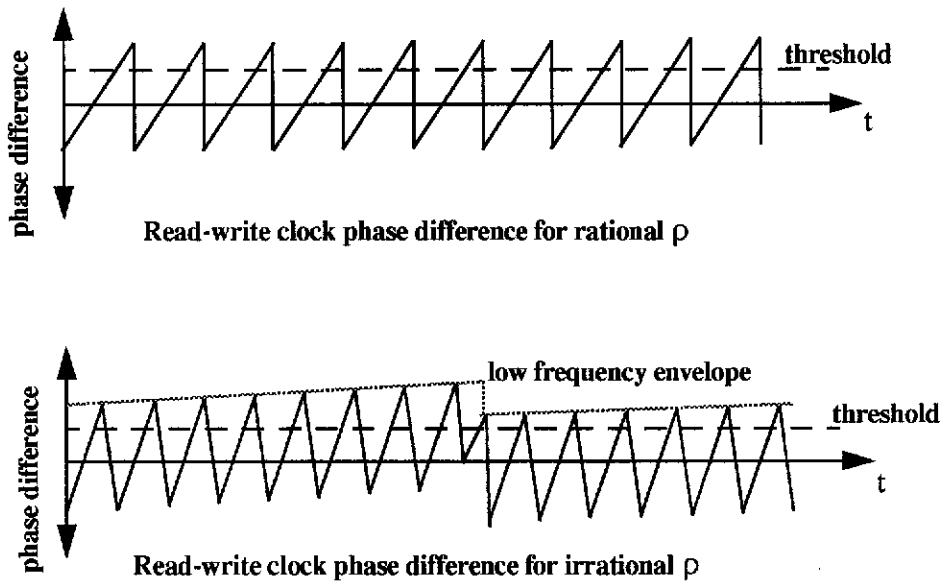


Figure 4.8: Presence of low frequency envelope for irrational rho.

According to [35], for some $\rho = P/Q + \Delta x$, the waiting time jitter frequency is $|\Delta x F_{XTS}|Q$ with a peak-to-peak amplitude of $1/Q$. The classical waiting time jitter characteristic can be plotted. In Figure 4.9, the peak-to-peak amplitude of the jitter when ρ is rational is plotted against ρ. When ρ is an irrational number, the peak-to-peak amplitude of the waiting time jitter is given by the nearest co-prime integer stuff ratio [35], [47]. However, as pointed out in [22], such an approach is an approximation as it is difficult to decide what is the nearest integer stuff ratio to a given ρ. More precise information about the frequencies and amplitudes of the waiting time jitter can be gained from examination of the jitter spectrum as shown by [22], [48].

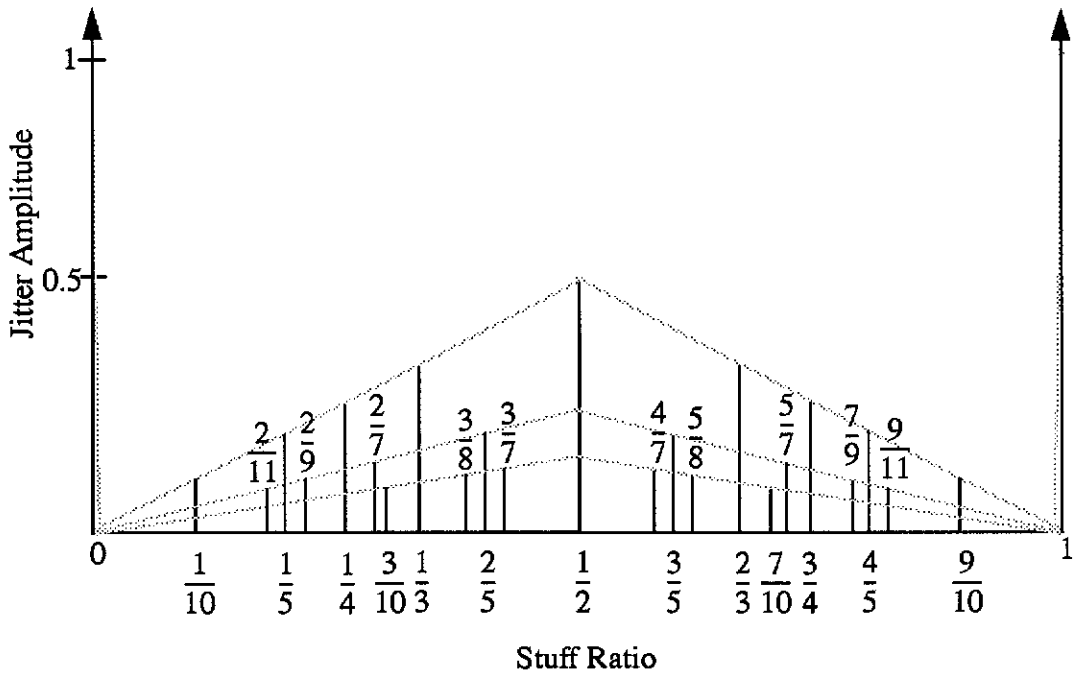


Figure 4.9: Classical waiting time jitter characteristic.

In real systems, it is unlikely that a rational value of ρ would be obtained and, even if it were, variation in actual clock frequencies due to temperature changes and ageing would cause ρ to depart from the rational value. Waiting time jitter would then occur. Waiting time jitter is problematic because it can occur at arbitrarily low frequencies, sometimes at quite large amplitudes [21]. In systems where waiting time jitter occurs and a desynchronizer is used, for example, multiplexer-demultiplexer systems [7], [22], [47], [48] a certain amount of waiting time jitter will always be passed by the desynchronizer, which has a lowpass characteristic. In the application under consideration in this chapter, the external timing source is intended to provide a clean frame, so it is desirable to minimise the amount of waiting time jitter that appears on it as a result of the injection process.

In previous studies of waiting time jitter [21], [22], [48], the effect of different values of ρ on the amount of unfiltered and filtered waiting time jitter that results has been studied. For example, in Duttweiler's work [22], the RMS amplitude of filtered waiting time jitter expected for two different filter transfer functions was plotted against ρ. From these plots, it was concluded that a careful choice of ρ could reduce

the RMS amplitude of the filtered waiting time jitter. According to [21] for the case of a multiplexer which can positive stuff only once per frame, as shown by the classical characteristic, the waiting time jitter can have a peak to peak amplitude of up to 1 UI depending upon the value of ρ. In [48], simulation of the peak-to-peak excursion of the filtered pulse stuffing jitter shows that for values of $\rho < 0.01$, the peak-to-peak excursion approaches 1. Some more detailed analysis [19], [20] suggests that the actual jitter generated in PDH multiplexing depends on the precise frame format, the buffer size and the phase detector implementation. The resulting filtered waiting time jitter amplitude may be less than that predicted by the ideal case. Nevertheless, the classical waiting time jitter provides a useful worst case analysis.

In the case of XTS injection, the jitter equation has been calculated using a time domain approach which shows that it is an exact representation of the jitter on the frame timing signal. Hence, when ρ is irrational, the classical waiting time jitter characteristic will apply and the value of ρ will be important in determining the peak-to-peak amplitude of the jitter present on the frame.

In the ideal cases analysed here, the value of ρ or equivalently $\mathrm{fr}(\eta)$ is set by the values of T_{XTS} and T_r or T_l. However, in reality, clock tolerances mean that the actual value of ρ or $\mathrm{fr}(\eta)$ can vary from the ideal value over a small range. As pointed out in [21] for the analogous case of a multiplexer, this means that it is a good idea to make the nominal value of ρ such that no waiting time jitter peaks occur within the operating range expected due to clock tolerances. Analysis of the waiting time jitter amplitude versus ρ has shown that optimal ranges for ρ include $(0.41, 0.44)$ [47]. Such an approach has been taken into account in the design of frames for the multiplexing hierarchies [48], for example, the ITU-T in its Recommendation G.742 on second order digital multiplex equipment using positive justification has designed the frame structure such that the nominal justification ratio is $\rho = 0.424$ [7].

In this chapter original analysis by the author [62] of two different methods of XTS injection has been presented. One of the methods analysed is the conventional stuffing approach and the other is a new simpler approach, first reported by the author [60],

[62], using a frame sampling synchronizer. The analysis has shown that the two different injection methods generate the same type of jitter on the resulting synchronized XTS and that this jitter is the so-called waiting time jitter. It has been pointed out in the discussion of waiting time jitter that appropriate choice of system parameters is important in order to minimize the amount of waiting time jitter which will appear on an XTS.

Appendix 4A: Glossary

T_{XTS}, F_{XTS} - period and frequency of the external timing source (XTS)

T_w, f_w - period and frequency of the write clock at the buffer

N - number of octets in a nominal frame

D - number of data octets in all frames

S_{nom} - number of stuffing octets in a nominal frame

T_r, f_r - period and frequency of the read (local) clock at the buffer

t_{min}, t_{max} - delay thresholds for stuffing at the buffer

τ - delay through the buffer

α, β - limits on the buffer delay if stuffing algorithm is operating correctly

t_n - jitter between XTS and synchronized XTS in the conventional stuffing synchronizer; also jitter between the XTS and the synchronized XTS in the frame sampling synchronizer

$\varepsilon(t_n)$ - stuffing function

T_F - length of a nominal (no stuffing) read (output) frame

$T_F(n)$ - length of the nth read (output) frame

$\overline{N}$ - ratio of T_{XTS} to T_r, in general not an integer

y_n - jitter between XTS and synchronized XTS normalised to T_r

y_{min}, y_{max} - stuffing thresholds normalised to T_r

ρ - the stuffing ratio, difference between write and read frequencies at the buffer normalised to the XTS (frame) frequency

L_n, U_n - lower and upper limits for jitter equation for the general case of the conventional stuffing synchronizer

L, U - lower and upper limits for jitter equation for the conventional stuffing synchronizer with fixed thresholds one unit apart

$\tilde{y}_n$ - quantized value of y_n

$\tilde{y}_{min}, \tilde{y}_{max}$ - quantized value of y_{min}, y_{max}

$g(t)$ - periodic, continuous time function which when sampled appropriately yields samples of the conventional stuffing synchronizer jitter

$G(f)$ - Fourier Transform representation of the Fourier Series of $g(t)$

$J_{CSS}(f)$ - jitter spectrum for the conventional stuffing synchronizer

b_k - coefficients of the jitter spectrum for the conventional stuffing synchronizer

$t_d(n)$ - total delay between asynchronous input and synchronized output in generalised synchronizer

Δ - constant part of the delay in generalised synchronizer

t_{po} - propagation delay of the output flip-flop in generalised synchronizer

T_l, f_l - period and frequency of local clock

η - ratio of T_{XTS} to T_l

x_n - t_n normalised to T_l

$z(t)$ - periodic, continuous time function which when sampled appropriately yields samples of the frame sampling synchronizer jitter

$J_{FSS}(f)$ - jitter spectrum for the frame sampling synchronizer

d_k - coefficients of the jitter spectrum for the frame sampling synchronizer

P, Q - if ρ is rational, $\rho = \frac{P}{Q}$ for P, Q co-prime integers

c_r - coefficients of jitter spectrum for rational ρ

Δx - small deviation in ρ from $\frac{P}{Q}$

Appendix 4B: Stability of the Conventional Stuffing Synchronizer Jitter Equation

Equation (4-3) can be rewritten using (4-1)

$$t_{n+1} = t_n + T_F(n) - T_{XTS} \tag{4B-1}$$

where the length of the nth read frame is $T_F(n) = T_F + \varepsilon(t_n)$, $T_F = NT_r$ is the length of a read frame when there is no stuffing and $\varepsilon(t_n)$ adjusts the length of the nth frame for stuffing. (4B-1) can be iterated to give

$$t_{n+1} = t_0 + \sum_{i=0}^{n} T_F(i) - (n+1)T_{XTS} \tag{4B-2}$$

When the stuffing is functioning correctly, it is designed to maintain the level of the buffer without underflow or overflow, hence the delay of the synchronized XTS, t_{n+1}, will be maintained between some finite limits

$$\alpha < t_{n+1} < \beta \tag{4B-3}$$

and since t_0 is an initialisation value which can be chosen to be between these limits then it follows that

$$0 \leq \left| \sum_{i=0}^{n} T_F(i) - (n+1)T_{XTS} \right| < |\beta - \alpha| \tag{4B-4}$$

Multiplying through (4B-4) by $\frac{1}{n+1}$ gives and taking the limit as $n \to \infty$

$$\lim_{n \to \infty} \frac{1}{n+1} \sum_{i=0}^{n} T_F(i) = T_{XTS} \tag{4B-5}$$

but $\lim_{n \to \infty} \frac{1}{n+1} \sum_{i=0}^{n} T_F(i)$ is the average output frame length. Hence, in the limit, the average output frame length equals the average input frame length and so equation (4-3) is stable.

Appendix 4C: Derivation of the Conventional Stuffing Synchronizer Equation

(4-3) can be rewritten as

$$t_{n+1} = t_n + NT_r - NT_w + T_r(1 - u(t_n - t_{min}) - u(t_n - t_{max})) \quad (4C\text{-}1)$$

where the stuffing function has been represented using the step function $u(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$. The stuffing function of (4-4) is asymmetric and this is necessary to maintain a consistent definition of the step function; then $u(t_n - t_{min})$ will always be equal to 1 except under the conditions for positive stuffing and $u(t_n - t_{max})$ will always be equal to 0 except under the conditions for negative stuffing.

For convenience, define a new variable $y_n = \frac{t_n}{T_r}$ and use it to re-express (4C-1) as

$$y_{n+1} = y_n + N - \frac{T_{XTS}}{T_r} + 1 - u(y_n - y_{min}) - u(y_n - y_{max}) \quad (4C\text{-}2)$$

where the representation of the stuffing function has been adjusted to take account of the new variable. Note that it is not necessarily the case that $|y_n|$, $|y_{min}|$ and $|y_{max}| < 1$. Equation (4C-2) can be further simplified by noting that

$$N - \frac{T_{XTS}}{T_r} = N - \overline{N} = -\rho \quad (4C\text{-}3)$$

where $\rho = \frac{f_r - f_w}{F_{XTS}}$ is the stuffing ratio. Thus equation (4C-2) becomes

$$y_{n+1} = y_n - \rho + 1 - u(y_n - y_{min}) - u(y_n - y_{max}) \quad (4C\text{-}4)$$

It has been shown, in Appendix 4B, that the jitter equation represented by (4C-4) is stable. Note that this implies, from the form of equation (4C-4), that $|\rho| < 1$ so that y_n is not a divergent sequence.

Now consider the values y_n can take on. If $y_{n-1} = y_{min}$ then no stuffing will occur (yet) and

$$y_n = y_{min} - \rho \tag{4C-5}$$

else if $y_{n-1} = y_{max}$ then negative stuffing will occur and

$$y_n = y_{max} - \rho - 1 \tag{4C-6}$$

Now, let y_n approach y_{min} from below that is, $y_{n-1} = y_{min} - \varepsilon$, $\varepsilon > 0$ then positive stuffing occurs and

$$y_n = y_{min} - \rho - \varepsilon + 1 \tag{4C-7}$$

that is $y_{min} - \rho + 1$ is approached from below. Similarly, for $y_{n-1} = y_{max} - \varepsilon$, $\varepsilon > 0$ no stuffing occurs and

$$y_n = y_{max} - \rho - \varepsilon \tag{4C-8}$$

and $y_{max} - \rho$ is approached from below. Thus, equation (4C-4) can take on values in the range $[L_n, U_n)$ where

$$\begin{aligned} L_n &= min(y_{min} - \rho, y_{max} - \rho - 1) \\ U_n &= max(y_{min} - \rho + 1, y_{max} - \rho) \end{aligned} \tag{4C-9}$$

This is the most general case. Since the two thresholds y_{min} and y_{max} are fixed and $y_{min} = y_{max} - 1$, this simplifies to

$$\begin{aligned} L &= y_{min} - \rho \\ U &= y_{max} - \rho \end{aligned} \tag{4C-10}$$

To achieve the closed form for equation (4C-4), iterate to give

$$y_n = y_0 - n\rho + n - \sum_{i=0}^{n-1} (u(y_i - y_{min}) + u(y_i - y_{max})) \tag{4C-11}$$

From the above knowledge of the range of values y_n can take on, it is clear that

$$0 \leq y_n - L < 1 \tag{4C-12}$$

and this can be used in (4C-11)

$$\lfloor y_n - L \rfloor = 0 = \lfloor y_0 - n\rho - L \rfloor + n - \sum_{i=0}^{n-1} (u(y_i - y_{min}) + u(y_i - y_{max})) \tag{4C-13}$$

to rewrite it as

$$y_n = y_0 - n\rho - \lfloor y_0 - n\rho - L \rfloor \tag{4C-14}$$

Appendix 4D: Measurement Quantization in the Conventional Stuffing Synchronizer

It can be shown that, when the thresholds lie on the quantization boundaries, the limits $[L, U)$ are the same as determined in Appendix 4C. Now, consider the case shown in Figure 4D.1 below, where the thresholds are still a unit apart but the quantization is finer, to the nearest half-unit, and the thresholds lie in between the quantization boundaries. Although the thresholds lie at intermediate values, the system can only see the quantized values, given by $\tilde{y}_n$, $\tilde{y}_{min}$ and $\tilde{y}_{max}$.

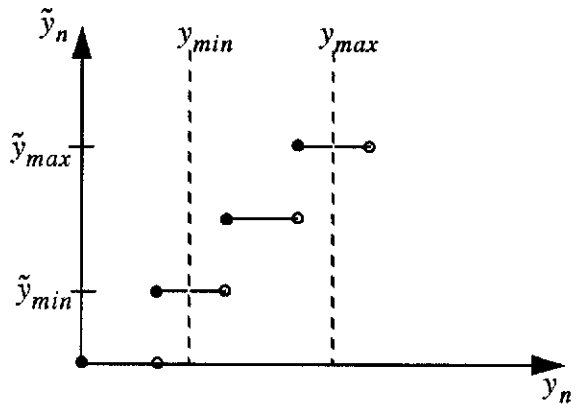


Figure 4D.1: Thresholds between quantization boundaries.

Since for positive stuffing to occur it is necessary that $\tilde{y}_{n-1} < \tilde{y}_{min}$, then the minimum value which can be reached is

$$y_n = \tilde{y}_{min} - \rho \tag{4D-1}$$

If $y_{n-1} = y_{min} - \varepsilon$ for some $\varepsilon > 0$ then positive stuffing will not occur until $y_{min} - \varepsilon < \tilde{y}_{min}$ and this means that the upper limit for y_n is

$$y_n = y_{min} - \varepsilon + 1 - \rho < \tilde{y}_{min} + 1 - \rho \tag{4D-2}$$

Similarly, since negative stuffing will occur once $\tilde{y}_{n-1} \geq \tilde{y}_{max}$ then the maximum value reached will be

$$y_n = \tilde{y}_{max} - 1 - \rho \tag{4D-3}$$

If $y_{n-1} = y_{max} - \varepsilon$ for $\varepsilon > 0$ then no negative stuffing will occur as long as $y_{n-1} = y_{max} - \varepsilon < \tilde{y}_{max}$ hence, the maximum value which can be reached is

$$y_n = y_{max} - \varepsilon - \rho < \tilde{y}_{max} - \rho \tag{4D-4}$$

That is, when the thresholds lie between quantization boundaries, the bounds of equation (4-6) are

$$\begin{aligned} L &= \tilde{y}_{min} - \rho \\ U &= \tilde{y}_{max} - \rho \end{aligned} \tag{4D-5}$$

and the system will behave as if the thresholds lie on quantization boundaries.

Appendix 4E: The Spectrum of the Jitter for the Conventional Stuffing Synchronizer

In order to find the jitter spectrum for the conventional stuffing synchronizer, the method outlined in Section 3.5 of Chapter 3 is applied. It is possible to define a continuous time function $g(t)$ which, when sampled at the frame rate, yields samples which equal the jitter generated by the conventional stuffing synchronizer.

$$g(t) = LT_r + \left\lceil \frac{t}{T_{XTS}}\rho \right\rceil T_r - \frac{t}{T_{XTS}}\rho T_r \tag{4E-1}$$

As $g(t)$ is periodic, with period $\frac{T_{XTS}}{\rho}$ secs, it can also be written as a Fourier Series

$$g(t) = \sum_{k=-\infty}^{\infty} b_k e^{\frac{j2\pi k\rho t}{T_{XTS}}} \tag{4E-2}$$

where

$$b_k = \begin{cases} \left(L + \frac{1}{2}\right)\frac{T_r}{2}, & k = 0 \\ \frac{T_r}{j2\pi k}, & k \neq 0 \end{cases} \tag{4E-3}$$

The Fourier Transform of $g(t)$ is

$$G(f) = \sum_{k=-\infty}^{\infty} b_k \delta(f - k\rho F_{XTS}) \tag{4E-4}$$

where the coefficients b_k are as defined in (4E-3), $F_{XTS} = \frac{1}{T_{XTS}}$ and use is made of the property that

$$\sum_{k=-\infty}^{\infty} e^{\frac{-j2\pi kt}{T}} \leftrightarrow \sum_{k=-\infty}^{\infty} \delta(f - kF) \tag{4E-5}$$

for $F = \frac{1}{T}$. Then applying equation (3-19) of Chapter 3 gives

$$J_{CSS}(f) = \mathcal{F}\{t_n\}$$

$$= F_{XTS} \sum_{k=-\infty}^{\infty} b_k \sum_{n=-\infty}^{\infty} \delta(f - k\rho F_{XTS} - nF_{XTS}) \qquad \text{(4E-6)}$$

where the b_k are as in (4E-3).

Chapter 5
Jitter Generation and the Synchronous Residual Time Stamp

5.1 Introduction

The Broadband Integrated Services Digital Network (B-ISDN) will transport all kinds of traffic in the same format: fixed length ATM (Asynchronous Transfer Mode) cells [11], [25], [44]. Within the B-ISDN, the AAL1 (ATM Adaptation Layer 1) will perform the functions necessary to allow CBR (continuous bit rate) services such as voice and video to be transported in ATM cells [5]. An important part of the function of the AAL1 is the transfer of the timing information required by a CBR source. While transmitting the data for a CBR source, packaged into ATM cells, the AAL1 must also encode the CBR service clock and transmit the timing information along with the data. Two methods have been approved for carrying out this task. One method suggested for end-to-end synchronization is the adaptive clock method which reconstructs the clock at the receiver by adjusting a local clock so that the level of the receive buffer is maintained at a fixed level [5], [53]. The second method proposed is the SRTS (Synchronous Residual Time Stamp) method [5].

In Section 5.2, the SRTS method is described in detail. The recovery of the timing signal at the destination is of particular interest. Other authors [46], [57] have all employed the same approach as that in [37]. Here, a new, more direct and easily verifiable method for destination timing signal recovery due to the author is presented.

In Section 5.3, a model of the SRTS method, when all clocks are assumed ideal and unjittered, is presented and exact expressions for the jitter generated and its spectrum are given. Previous authors [37], [46], [57] have noted that the jitter generated by the SRTS method is equivalent to the so-called waiting time jitter [22]. However, exact expressions applying to the jitter spectrum generated by the SRTS method have not previously been given.

In Section 4.5 of Chapter 4, the relationship between the value of the stuff ratio, ρ, and the resulting amount of waiting time jitter was presented. In general, ρ is chosen so that the peak-to-peak amplitude of the waiting time jitter is small. In Section 5.4, original analysis by the author explores whether it is possible to specify parameters of the SRTS method so as to reduce the amplitude of the waiting time jitter found on the recovered timing signal at the destination. It is found that, for some service clocks

with the allowed tolerance specified in standards [5], it is not possible to choose the equivalent ρ to guarantee a small waiting time jitter amplitude.

5.2 The SRTS Method

The SRTS method allows the transport of a service clock which is asynchronous to the network clock [5]. A number representing the residual part of the frequency difference between the service clock and a reference clock derived from the network clock [5] is generated and transported across the network in ATM cells. At the destination, knowledge of some parameters of the method, including the nominal part of the frequency difference between the clocks, permits regeneration of the service clock using the reference clock.

In this section, a detailed description of the SRTS method will be presented including the recovery of the timing signal at the destination using a new method developed by the author. A clear understanding of the working of the SRTS method is a necessary background to understanding how it results in jitter and how the jitter may affect the regenerated service clock at the destination.

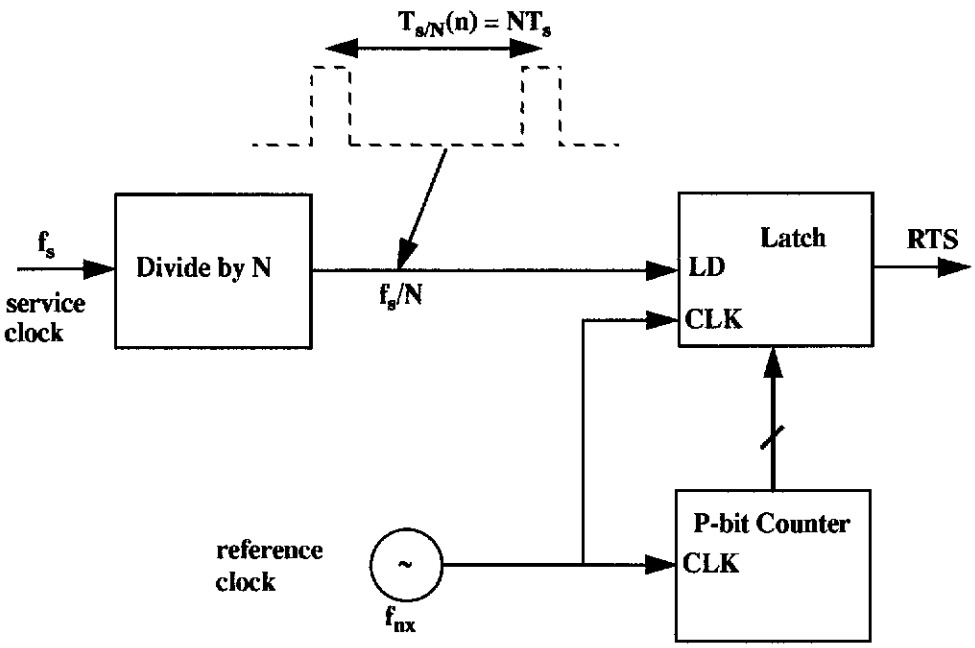


Figure 5.1: RTS Generation in the SRTS Method.

5.2.1 RTS Generation

A block diagram of RTS generation as outlined in [5], [37] is shown in Figure 5.1. The service clock with frequency f_s is first divided down by N. A counter is clocked by the reference clock which has frequency f_{nx}. A latch is used to capture the counter output at the regular instants of the divided-down service clock. The whole number of reference clock cycles between successive latched counter outputs gives the divided down service clock period encoded to the nearest reference clock cycle, a value denoted by S_n. The timing diagram in Figure 5.2 illustrates the principles involved.

The exact number of reference clock cycles in $T_{s/N}(n)$ is given by $M = (Nf_{nx})/f_s$, which is normally not an integer. Due to clock tolerance, the exact value of M cannot be known, but can be assumed to be within some known range $[M_{min}, M_{max}]$. Furthermore, the value of M may change slowly over time due to clock ageing. It will be shown in Section 5.3 that, even if M were a constant, the successive values of S_n are not all the same but will vary between two values. Given that in practice M itself can vary, then the S_n values can potentially vary even more. The SRTS method has been designed, however, so that the variation in S_n is very small compared to its size. Thus, the successive values of S_n can be divided into a nominal part and a residual part. If it is assumed that the destination has available the reference clock and that both N and the nominal part of S_n are known, then only the residual part, known as the RTS (Residual Time Stamp) need be sent [5], [37].

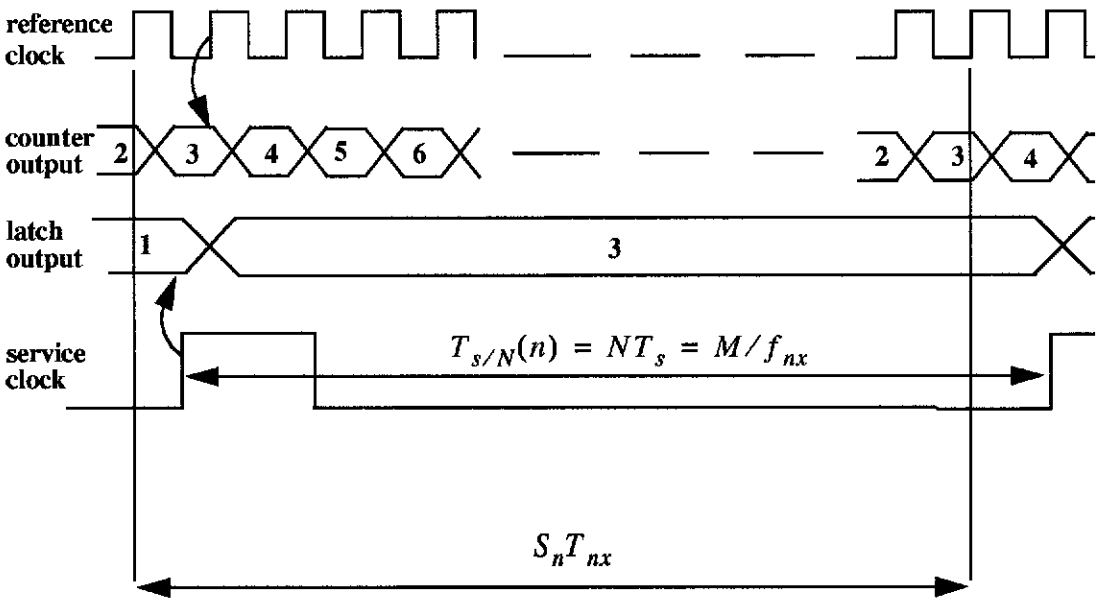


Figure 5.2: Timing Diagram of SRTS Method.

The variation in the size of the RTS is given by $2y = M_{max} - M_{min}$ which depends on the reference clock to service clock ratio, the service clock tolerance and N as shown below

$$y = N\frac{f_{nx}}{f_{s,nom}}\varepsilon \tag{5-1}$$

where $f_{s,nom}$ is the nominal value of the service clock frequency before division and ε is the tolerance of the service clock [5], [37]. For the RTS to be uniquely identifiable from P bits, it is shown in Appendix 5B that

$$2^{(P-1)} > \lceil y \rceil \tag{5-2}$$

where $\lceil y \rceil$ is the least integer $\geq y$.

The P-bit latch output in Figure 5.1 is the RTS which can be characterized as follows [37]

$$RTS_{n+1} = RTS_n + S_n \qquad \text{(modulo } 2^P\text{)} \tag{5-3}$$

The RTS is transported in AAL1 cells through the ATM network, where the AAL1 cells which carry it may be subject to a variable end to end delay. At the destination, the timing information is decoded and used to produce a reference signal for input to a phase-lock loop (PLL) which regenerates the service clock. Note that for correct operation, the SRTS method assumes the existence of common network clocks at the origin and the destination [5], [37]. The common reference clock may be obtained in a number of ways: an out-of-band signal, an inband signal or the line rate [38].

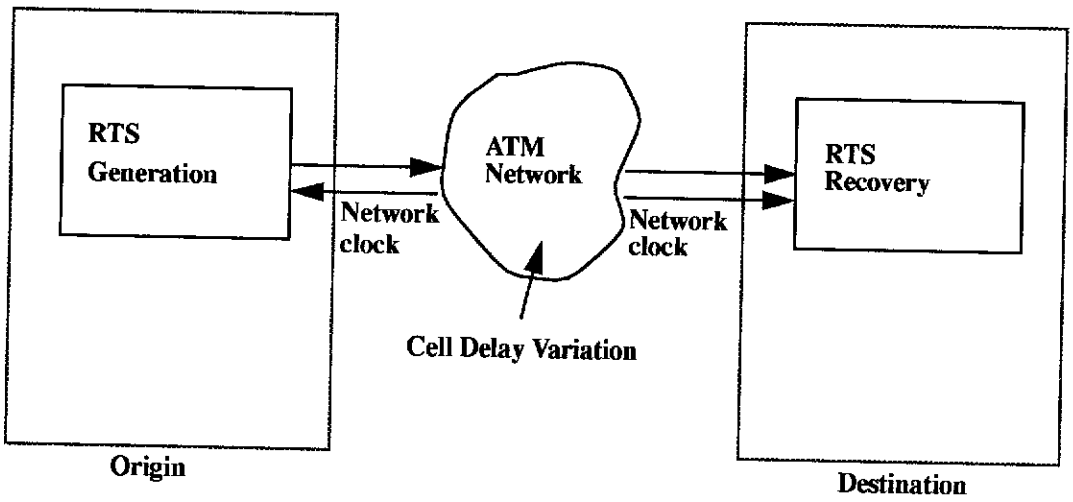


Figure 5.3: General principle of timing transport by AAL1 for CBR sources.

5.2.2 Transport of the RTS

As described in [5], generation of the RTS is performed in the Convergence Sublayer (CS) of the AAL1. The CS provides to the Segmentation and Reassembly (SAR) sublayer of the AAL1 a 47 octet data payload together with the RTS and a sequence number for the data payload. The SAR then adds a single octet SAR-PDU header to form the 48 octet SAR-PDU. The first four bits of the SAR-PDU header form the SN (Sequence Number) field which is further subdivided into the CSI bit and the sequence count field. The sequence count field provides a modulo-8 sequence count which provides the opportunity for the destination to detect lost or out of sequence cells. The four bit RTS is transmitted serially using the CSI bit in SAR-PDU headers which have an odd sequence number. Thus, it takes eight SAR-PDU headers, or eight SAR-PDU payloads, to fully transmit each RTS.

Due to multiplexing in ATM switches in the network, the ATM cells which transport the bits of the RTS are subject to cell delay variation as they travel across the ATM network [6]. The cell delay variation (CDV) is constrained by network dimensioning and resource allocation but nevertheless exists. As a result, the rate of arrival of the RTS bits at the destination will vary and so will the apparent arrival rate of the RTS as it is reconstructed from the serial bitstream.

5.2.3 Regeneration of the Service Clock at the Destination

At the destination, a FIFO is used to absorb the cell delay variation [41] as illustrated in Figure 5.4. Reading from and writing to the FIFO are both synchronous to the reference clock. The select function allows generation of the reference clock from either a locally available network clock or from the network clock recovered from the incoming bitstream. On the write side of the FIFO, the arrival rate of the RTS is variable due to the cell delay variation. As shown in Figure 5.4, reading the RTS is performed at a rate controlled by the successive values of S_n regenerated from the RTS values which were calculated by the SRTS method at the origin.

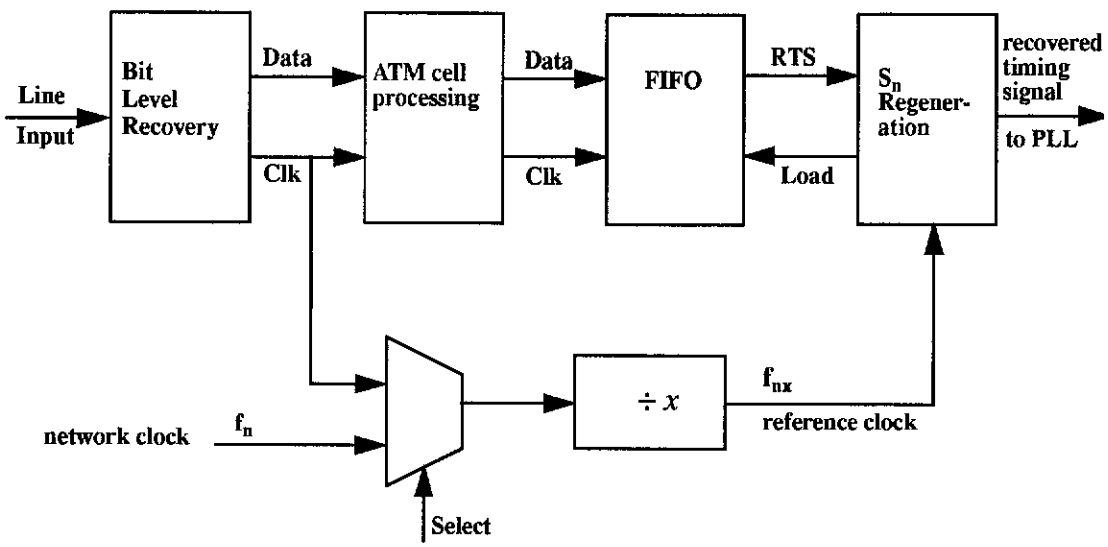


Figure 5.4: Overview of regeneration of the timing signal at the destination.

Figure 5.5 shows one approach to reconstructing the pulse stream with intervals $S_n T_{nx}$ [37], [57].

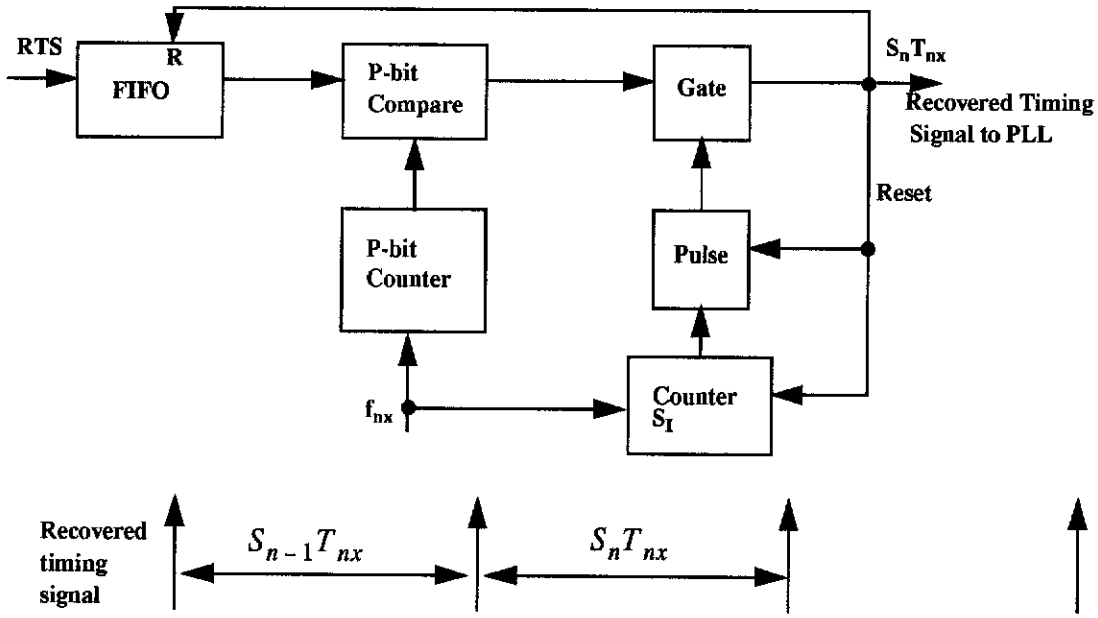


Figure 5.5: Possible implementation of S_n regeneration.

The P-bit comparator generates a pulse when the output of the free-running P-bit counter equals the received RTS. After the first pulse, this results in a pulse every 2^P cycles. The second counter counts down from $S_I = \lfloor M_{max} \rfloor - 2^{P-1}$ generating a gating pulse which selects the correct comparator pulse since $\lfloor M_{max} \rfloor - 2^P < S_I < \lfloor M_{min} \rfloor$. The chosen comparator pulse provides the reset pulse and the time intervals between reset operations are given by the successive values, that is, $\ldots, S_{n-1}T_{nx}, S_nT_{nx}, S_{n+1}T_{nx}, \ldots$. The recovered timing signal is then filtered using a phase-locked loop to reduce high frequency jitter.

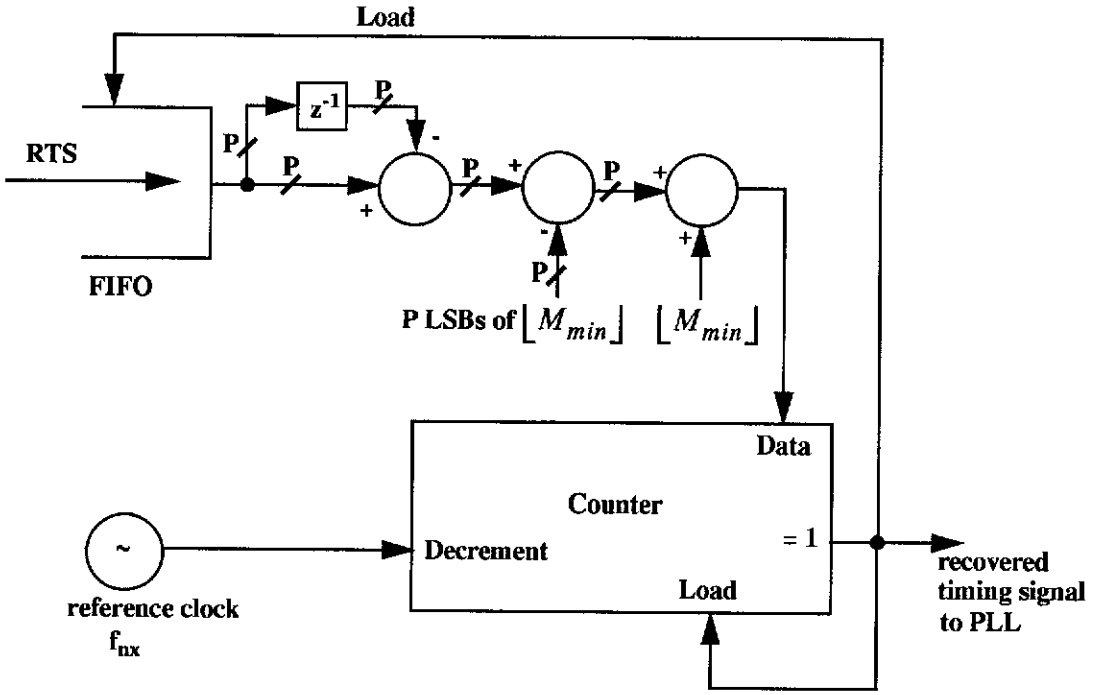


Figure 5.6: New method for regeneration of the timing signal at the destination.

Figure 5.6 shows a new method of S_n regeneration developed by the author. As illustrated, by subtracting (modulo 2^P) successive RTS values, the residual part of S_n modulo 2^P can be reproduced at the receiver. In this method, the nominal portion which must be held at the receiver is $\lfloor M_{min} \rfloor$. As this is the smallest possible value of S_n, other values of S_n can be calculated from it by addition. To find out how much to add to $\lfloor M_{min} \rfloor$, it is necessary to subtract (modulo 2^P) the P least significant bits of $\lfloor M_{min} \rfloor$ from the recovered P LSBs of S_n. Adding the result of this second subtraction to $\lfloor M_{min} \rfloor$ gives the correct value of S_n. The new method of S_n regeneration is simpler and avoids one feedback loop present in the method of Figure 5.5. Furthermore, its performance can readily be verified analytically as shown in Appendix 5C. Simulations have been used to confirm that the new method can regenerate the correct value of S_n.

The final step to convert the series of numbers ..., S_{n-1}, S_n, S_{n+1}, ... into a signal in time is also shown in Figure 5.6. The corresponding timing diagram which illustrates the relationships between the signals involved is shown in Figure 5.7.

The recovered value of S_n is loaded into a counter which is clocked by the reference clock and counts down from this value. A signal generated when the counter reaches one is used to reload the counter with the next value. The time intervals between successive counter load operations are given by the successive values, that is, $\ldots, S_{n-1}T_{nx}, S_nT_{nx}, S_{n+1}T_{nx}, \ldots$. The recovered timing signal is the resulting output of pulses coincident with the reload signal and recreates the divided down service clock to the nearest reference clock cycle.

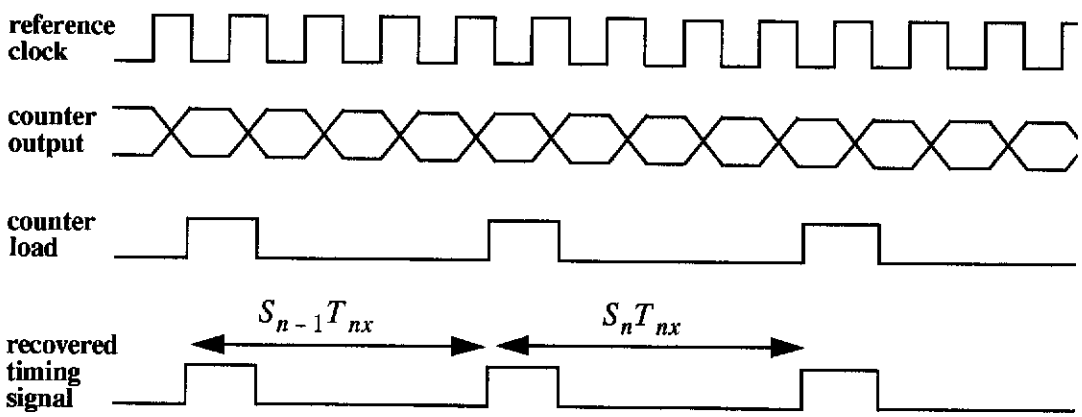


Figure 5.7: Timing diagram for regeneration of the timing signal.

5.2.4 Requirement for a Synchronizer in RTS Generation

As the service clock is asynchronous to the reference clock then, as discussed in Chapter 2, there is always the possibility of metastable failure occurring at the RTS latch in Figure 5.1. A synchronizer is required to achieve an acceptable MTBF due to metastable failure. The placement of the synchronizer is shown in Figure 5.8.

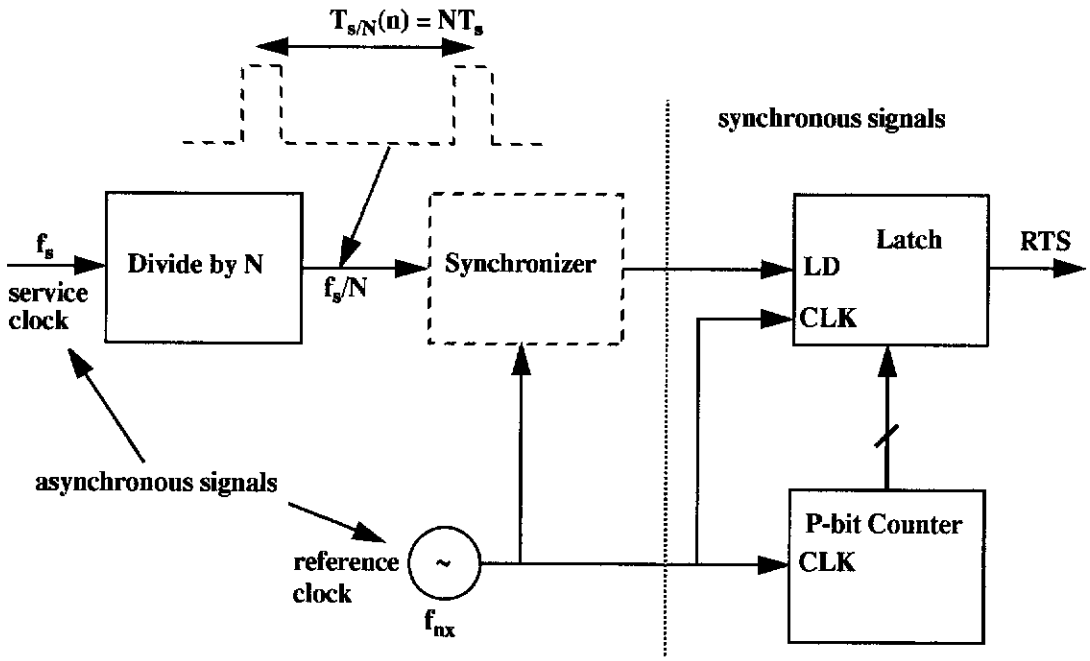


Figure 5.8: Need for a synchronizer in RTS generation.

The addition of the synchronizer does not alter the basic operation of RTS generation. Based on the timing diagram shown in Figure 5.9, it can be verified that the systems in Figure 5.1 and Figure 5.8 produce the same sequence S_n, albeit with some delay difference.

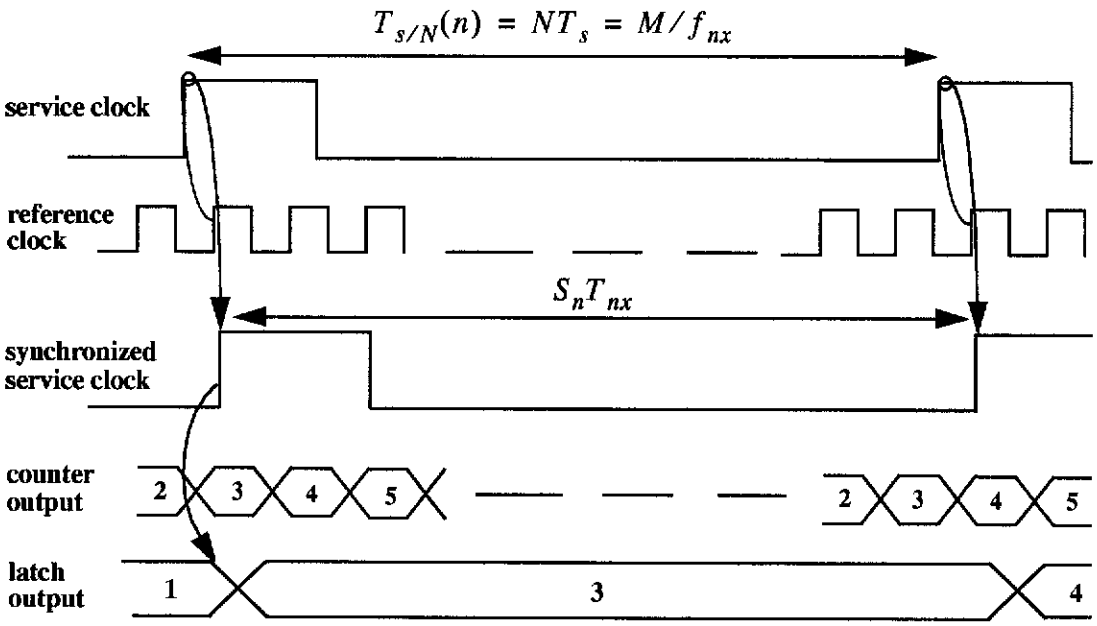


Figure 5.9: Timing diagram of SRTS Method with synchronizer.

5.3 Jitter Generation in the SRTS Method

The overview of the SRTS method in the previous section has shown that the RTS which carries the timing information about the service clock rate is a number. However, it is an inherent characteristic of SRTS frequency encoding that the timing signal reconstructed from the RTS number has jitter and that this jitter is equivalent to the so-called waiting time jitter [37], [46], [57]. An exact expression for the jitter and its spectrum has not previously been produced. In this section, it will be shown that RTS generation inevitably involves the synchronization of one clock into the timing domain of another clock and hence the jitter equation derived for the synchronization process in Section 3.4 of Chapter 3 applies.

5.3.1 Jitter Generation in an Ideal SRTS System

In this section, an analysis of an ideal SRTS system is presented. As illustrated in Figure 5.10, it is assumed that the reference clocks at the origin and destination have identical frequencies, although there may be a known constant phase difference, ϕ, between the origin and destination reference clocks. Furthermore, it is assumed that the reference clock and the service clock are unjittered. In addition, it will be assumed

that cell delay variation in RTS transport is taken care of by appropriate buffering [41] and thus there is a fixed end to end transport delay. In this model, the only source of jitter, which may subsequently appear on the output reference timing signal delivered to the PLL at the destination, is that due to the SRTS method itself.

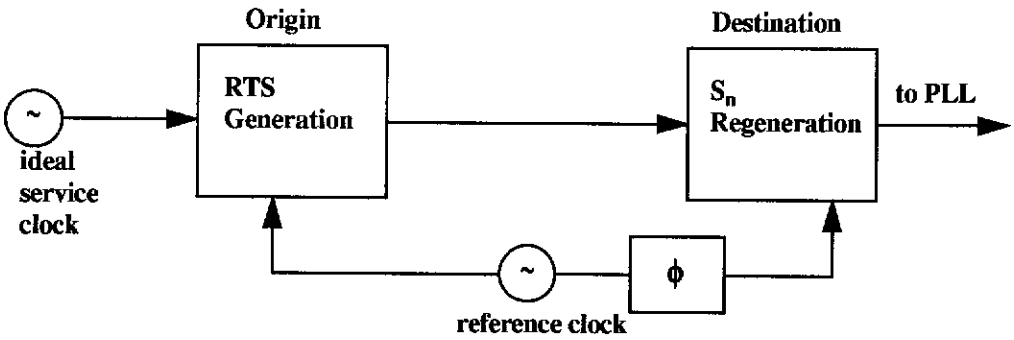


Figure 5.10: SRTS system model.

Figure 5.11 illustrates the relationships between the signals involved in RTS generation.

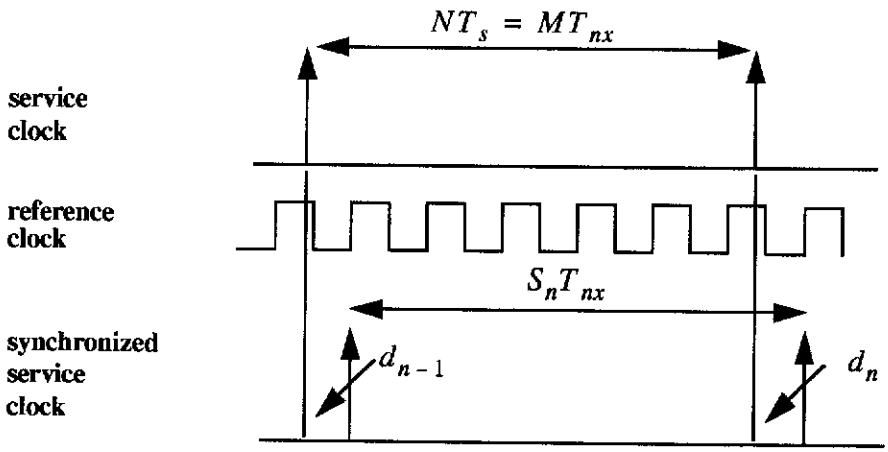


Figure 5.11: Timing diagram to show jitter in RTS generation.

From Figure 5.11, a jitter recurrence equation can be written for RTS generation

$$MT_{nx} + d_n - d_{n-1} = S_nT_{nx} \tag{5-4}$$

where d_n is the jitter on the synchronized service clock, $T_{nx} = 1/f_{nx}$ is the period of the reference clock, $M = (NT_s)/T_{nx}$ and it is assumed that the service clock is ideal with a constant but not necessarily nominal period.

In Section 3.4 of Chapter 3, it was shown that a jitter recurrence equation such as (5-4) can be written as

$$M = \lfloor M \rfloor + \frac{d_{n-1}}{T_{nx}} - \frac{d_n}{T_{nx}} + 1 - u\left(\frac{d_{n-1}}{T_{nx}} + \lfloor M \rfloor - M\right) \tag{5-5}$$

and applying the analysis of Appendix 3C of Chapter 3 to (5-5), the closed form for d_n, the jitter generated on the synchronized service clock, is

$$d_n = d_0 - nMT_{nx} - \left\lfloor \frac{d_0}{T_{nx}} - nM \right\rfloor T_{nx} \tag{5-6}$$

It also follows from equations (5-4) and (5-5) that

$$S_n = \lfloor M \rfloor + 1 - u\left(\frac{d_{n-1}}{T_{nx}} - \mathrm{fr}(M)\right) \tag{5-7}$$

that is the value of S_n, the synchronized service clock period, is either $\lfloor M \rfloor$ or $\lfloor M \rfloor + 1$. Applying the stability analysis in Appendix 3D of Chapter 3 to equation (5-4), the long term average of S_n can be shown to be equal to M.

In Section 5.2.3, techniques for the regeneration of the service clock at the destination were presented. The RTS values are used to recover the series of numbers $\ldots, S_{n-1}, S_n, S_{n+1}, \ldots$ which are then converted into a signal in time, that is, the synchronized service clock is re-created at the destination, using the destination reference clock. As shown in Figure 5.10, in a practical SRTS system, it is likely that the origin and destination reference clocks would not be in phase. For a complete system model, the effect of this phase difference should be examined.

Without loss of generality, it will be assumed that the destination reference clock is always delayed with respect to the origin reference clock, by some constant amount ϕT_{nx} where $0 \le \phi < 1$. In SRTS recovery, the synchronized service clock period at

the destination, S_{n2}, is a reproduction of that at the origin, S_{n1}, that is, $S_{n1} = S_{n2} = S_n$. It follows from this constraint that the only effect of the constant phase difference is to add a constant delay. Hence, when there is a constant phase difference between origin and destination reference clocks, the jitter on the recovered timing signal at the destination is still given by d_n.

5.3.2 Jitter Spectrum for the Ideal Model of RTS Generation

Equation (5-6) is the jitter on the synchronized service clock for the case where the service clock period is assumed constant. The spectrum associated with such a jitter equation has been found in Section 3.5 of Chapter 3 and is shown in equation (5-8) below.

$$J_{SRTS}(f) = \frac{f_s}{N} \sum_{k=-\infty}^{\infty} \alpha_k \sum_{n=-\infty}^{\infty} \delta\left(f - kM\frac{f_s}{N} - n\frac{f_s}{N}\right) \tag{5-8}$$

where the coefficients α_k are given by

$$\alpha_k = \begin{cases} \dfrac{T_{nx}}{2}, & k = 0 \\ \dfrac{T_{nx}}{j2\pi k} e^{-j2\pi k \frac{d_0}{T_{nx}}}, & k \neq 0 \end{cases} \tag{5-9}$$

As shown by equation (4-19) of Chapter 4, (5-8) can be rewritten as

$$J_{SRTS}(f) = \frac{f_s}{N} \sum_{k=-\infty}^{\infty} \alpha_k \sum_{m=-\infty}^{\infty} \delta\left(f - k\mathrm{fr}(M)\frac{f_s}{N} - m\frac{f_s}{N}\right) \tag{5-10}$$

It can be seen by comparing (5-10) with (4-19) and (4-12) of Chapter 4 that the jitter spectrum from RTS generation has the same form as in the two previously discussed cases, that is RTS generation can also produce the so-called waiting time jitter. Hence, $\mathrm{fr}(M)$ determines the position of the spectral lines in (5-10), in the same way that $\mathrm{fr}(\eta)$ and ρ determine the position of the spectral lines in the case of external timing

injection. For use in the next section, as the notation ρ is more concise, therefore define $\rho \equiv \mathrm{fr}(M)$.

If ρ is a simple rational number, there will be a large jitter amplitude (1 UI, where the Unit Interval is the reference clock period) but at a relatively high frequency of $f_j = (f_s\rho)/N$ which can be filtered by the PLL [37]. If ρ deviates from a simple rational number there can be significant low frequency components in the spectrum [37], [46], [57]. For the case of asynchronous service clock transport, it is more likely that ρ will be irrational and that waiting time jitter will therefore be present. In an experimental application of the SRTS method in a CLAD (Cell Assembly and Disassembly) device, Uematsu and Ueda [57] found that for certain values of ρ (0, 1/3, 1/2, 2/3, 1) the peak-to-peak jitter was large regardless of the PLO loop gain. The reason for this finding would be that in practice, the actual value of ρ differed very slightly from the exact integer ratios.

5.4 Choosing Parameters in RTS Generation

In this section original analysis by the author is presented which explores whether it is possible to choose ρ to reduce the waiting time jitter amplitude in RTS generation.

Recall that the parameters of the SRTS method described in Section 5.2 are as follows:

- an ideal and unjittered reference clock which is identical at the origin and destination, and has frequency $f_{nx} = f_n/x$ where f_n = 155.520Mbit/s is the network clock and x is an integer
- an ideal and unjittered service clock which has frequency f_s and which is divided down by N to produce the input signal for the RTS generation process
- the number of cycles of the reference clock in one period of the divided down service clock is given by $M = (Nf_{nx})/f_s$ where M is a constant in the range $[M_{min}, M_{max}]$

First, consider the case when the frequency of the service clock is constant and nominal, that is $f_{s,nom}$, and ρ_{nom} is to be located within some desired range (ρ_1, ρ_2). M is nominal as shown in equation (5-11).

$$M = M_{nom} = \frac{N f_{nx}}{f_{s,nom}} \tag{5-11}$$

The reference clock to service clock ratio is restricted to

$$1 < \frac{f_{nx}}{f_s} \leq 2 \tag{5-12}$$

that is, either $f_{nx}/f_{s,nom} = 1 + \mathrm{fr}(f_{nx}/f_{s,nom})$ or $f_{nx}/f_{s,nom} = 2$. The second possibility is both unlikely for asynchronous clocks and not useful for the purpose of determining ρ and so will not be considered.

Defining $\Upsilon = \frac{f_n}{f_{s,nom}}$, (5-11) can be rewritten as

$$M_{nom} = N\frac{\Upsilon}{x} \tag{5-13}$$

Υ is not usually an integer for the general case of asynchronous circuit transport and from (5-12), $\frac{\Upsilon}{2} < x < \Upsilon$.

Expanding (5-13) gives

$$\lfloor M_{nom} \rfloor + \rho_{nom} = N + \left\lfloor N\mathrm{fr}\left(\frac{\Upsilon}{x}\right) \right\rfloor + \mathrm{fr}\left(N\mathrm{fr}\left(\frac{\Upsilon}{x}\right)\right) \tag{5-14}$$

For each value of x and for ρ_1 and ρ_2 which define the limits of the range for the desired ρ_{nom}, it is possible to write

$$\begin{aligned} N\frac{\Upsilon}{x} - N - \rho_1 &= \left\lfloor N\mathrm{fr}\left(\frac{\Upsilon}{x}\right) \right\rfloor + \mathrm{fr}\left(N\mathrm{fr}\left(\frac{\Upsilon}{x}\right)\right) - \rho_1 = L_1 \\ N\frac{\Upsilon}{x} - N - \rho_2 &= \left\lfloor N\mathrm{fr}\left(\frac{\Upsilon}{x}\right) \right\rfloor + \mathrm{fr}\left(N\mathrm{fr}\left(\frac{\Upsilon}{x}\right)\right) - \rho_2 = L_2 \end{aligned} \tag{5-15}$$

where L_1 and L_2 are not usually integers and $L_2 < L_1$.

It is shown in Appendix 5D, that when $\lfloor L_1 \rfloor \neq \lfloor L_2 \rfloor$, then there exists an integer solution $L = \lfloor L_1 \rfloor$ for that particular x, and so there exists a ρ_{nom} within the desired range (ρ_1, ρ_2) and M_{nom} can be written

$$M_{nom} = N + L + \rho_{nom} \tag{5-16}$$

The following nominal service clock frequencies are listed in [5] as examples of CBR timing signals requiring asynchronous circuit transport using the SRTS method: $\{1544, 2048, 6312, 8448, 32064, 34368, 44736\}$kbit/s (G.702 Recommendation signals). A computer program was used for each of the service clock frequencies listed to determine whether it would be possible for each to find an x and an L such that $\rho_{nom} \in (0.41, 0.44)$. The results are given below in Table 5.1 where it is shown that for some nominal service clock frequencies, attempting to choose ρ_{nom} within such a narrow range will not be possible.

Table 5.1: Values of x and L for nominal rho in a narrow recommended range

$f_{s,nom}$ (kbit/s)	ρ_{nom}	x	L
1544	0.437931	73	1142
1544	0.423929	99	52
2048	0.434782	69	302
6312	0.414449	18	1109
8448	-	-	-
32064	0.425150	4	639
34368	-	-	-
44736	-	-	-

Now consider the case when the service clock has tolerance ε, that is, $M \in [M_{min}, M_{max}]$ where

$$\begin{aligned} M_{max} &= M_{nom} + y \\ M_{min} &= M_{nom} - y \end{aligned} \tag{5-17}$$

and $y = N\dfrac{f_{nx}}{f_{s,\,nom}}\varepsilon$ as in equation (5-1). Substituting from (5-13) and (5-16), the possible range of values for M is

$$M_{nom} \pm y = N\frac{\Upsilon}{x}(1 \pm \varepsilon) = (N + L + \rho_{nom})(1 \pm \varepsilon) \quad (5\text{-}18)$$

Hence, the value of ρ when the effect of service clock tolerance is included is given by

$$\rho = \rho_{nom} \pm \rho_{nom}\varepsilon \pm \mathrm{fr}(N\varepsilon) \pm \mathrm{fr}(L\varepsilon) \quad (5\text{-}19)$$

According to [5], the value of N is determined by the number of SAR-PDU payloads required to transmit the RTS. As this number is 8 and each payload consists of 47 octets, then the usual value of N is $8 \times 8 \times 47 = 3008$. [5] implies that 3008 is a maximum value for N because additional requirements for CS layer overhead in the SAR-PDU will reduce the number of bits available for user data and the value of N, which must remain constant, has to be correspondingly reduced. The maximum tolerance expected for the asynchronous circuit transport of the G.702 Recommendation signals is given as $\varepsilon = \pm 200 \times 10^{-6}$ [5] and as the maximum value of N is 3008, it follows that $\lfloor (N\varepsilon) \rfloor = 0$. Furthermore, as $L = \lfloor N\mathrm{fr}(\Upsilon/x) \rfloor$ then $\lfloor L\varepsilon \rfloor = 0$ and so (5-19) becomes

$$\rho = \rho_{nom} \pm \rho_{nom}\varepsilon \pm N\varepsilon \pm L\varepsilon \quad (5\text{-}20)$$

The most important thing to note from equation (5-20) is that because of the presence of the term $N\varepsilon$, the range in ρ for different service clocks with the same nominal frequency can be quite large. For example with $\varepsilon = \pm 200 \times 10^{-6}$, ρ varies by ± 0.6016. To further illustrate this point, assume that having chosen the values of ρ_{nom} as given in Table 5.1, it is desired to calculate the tolerance on the service clock necessary to allow values of ρ within some range (ρ_3, ρ_4), where this range is wider than the one used to determine ρ_{nom}. (5-20) can be used to solve for the tolerance. Some illustrative results are shown in Table 5.2. The first five rows are for $\rho \in (0.35, 0.45)$ using the values of ρ_{nom} from Table 5.1. The sixth row widens the allowable range for ρ_{nom} to $\rho_{nom} \in (0.35, 0.45)$. For the two higher frequencies in the bottom two rows of the table, it was not possible to find values for ρ_{nom} even using this wider range. Where values for the tolerance are given, it is usually wider in one direction than the other. For symmetrical tolerances, the smaller value will of

course have to be taken, thus the Table 5.2 shows that tolerances would need to be less than 10 ppm in order to restrict the value of ρ.

Table 5.2: Required tolerances when restricting rho to a narrow range.

$f_{s,nom}$ (kbit/s)	ρ_{nom}	positive tolerance (x 10^{-6})	negative tolerance (x 10^{-6})
1544	0.437930	2.908	21.19
1544	0.423928	8.519	24.16
2048	0.434783	4.597	25.61
6312	0.414449	8.634	15.65
32064	0.425150	6.813	20.60
8448	0.363636	28.07	4.433
34368	-	-	-
44736	-	-	-

The chief conclusions of this section are that while it is possible given a nominal service clock frequency to select x such that ρ_{nom} lies within a suitable range, this becomes more and more difficult as the service clock frequency increases. Furthermore, when the service clock frequency, while still constant may vary from the nominal, the retention of ρ within a suitable range means that the tolerance of the service clock must be very small. For service clock tolerances as large as 100-200 ppm, it is not possible to restrict the value of ρ in any meaningful way.

Appendix 5A: Glossary

T_s, f_s - period and frequency of the service clock

f_n - frequency of the network clock

N - usually equal to 3008, the divide down ratio for the service clock

x - the divide down ratio for deriving the reference clock from the network clock

T_{nx}, f_{nx} - period and frequency of reference clock

$T_{s/N}(n)$ - nth period of service clock divided down by N

M - exact number of reference clock cycles in $T_{s/N}(n)$

M_{min}, M_{max} - lower and upper limits for values M can take on due to service clock tolerance

S_n - number of reference clock periods in the nth period of the synchronized service clock to the nearest reference clock cycle

ε - tolerance of the service clock

$f_{s,\,nom}$ - nominal service clock frequency

y - $2y$ is the total variation allowed in M

P - number of bits used for the RTS

d_n - jitter between the synchronized service clock edge and the divided down service clock

$J_{SRTS}(f)$ - the spectrum of the jitter on the recovered timing signal in the ideal model of the SRTS method

α_k - coefficients of the spectrum of the jitter on the recovered timing signal in the ideal model of the SRTS method

ρ - equal to $\mathrm{fr}(M)$ when discussing parameters of the SRTS method

M_{nom} - the value of M for the nominal service clock frequency

Υ - the ratio of the network clock and the nominal service clock frequency

ρ_{nom} - the value of ρ for the nominal service clock frequency

$\rho_1, \rho_2, \rho_3, \rho_4$ - the lower and upper limits of the acceptable range of values for ρ_{nom} and for ρ when the tolerance on the service clock is considered

L_1, L_2 - numbers, corresponding to ρ_1, ρ_2 used in determining whether ρ_{nom} exists

L - an integer, if it exists, then an acceptable ρ_{nom} can be found

x_n - used to represent RTS_n, the nth RTS value, in proof of new regeneration method

ϕ - value of constant phase difference between origin and destination reference clocks

Appendix 5B: Derivation of the Number of Bits to Uniquely Identify RTS

$2y$ can be written as

$$2y = \lfloor M_{max} \rfloor - \lfloor M_{min} \rfloor - (\mathrm{fr}(M_{max}) - \mathrm{fr}(M_{min})) \quad \text{(5B-1)}$$

since neither M_{max} nor M_{min} is necessarily an integer. Furthermore, for $M \in [M_{min}, M_{max}]$ the range of possible values for Sn is

$$\{\lfloor M_{min} \rfloor, \lfloor M_{min} \rfloor + 1, \ldots, \lfloor M_{max} \rfloor, \lfloor M_{max} \rfloor + 1\} \quad \text{(5B-2)}$$

that is $\lfloor M_{max} \rfloor - \lfloor M_{min} \rfloor + 2$ unique values. To represent these $\lfloor M_{max} \rfloor - \lfloor M_{min} \rfloor + 2$ unique values using P bits, it is required that

$$\lfloor M_{max} \rfloor - \lfloor M_{min} \rfloor \leq 2^P - 2 \quad \text{(5B-3)}$$

Now, from (5B-1) there are three possible cases. In the first case, $2y$ is not an integer and $\lceil 2y \rceil = \lfloor M_{max} \rfloor - \lfloor M_{min} \rfloor$ and so it follows that

$$y < 2^{(P-1)} - 1 \quad \text{(5B-4)}$$

or

$$\lceil y \rceil \leq 2^{(P-1)} - 1 \quad \text{(5B-5)}$$

In the second case, $2y$ is not an integer and $\lfloor 2y \rfloor = \lfloor M_{max} \rfloor - \lfloor M_{min} \rfloor$ and so it follows that

$$y < 2^{(P-1)} - \frac{1}{2} \quad \text{(5B-6)}$$

or

$$\lceil y \rceil \leq 2^{P-1} \quad \text{(5B-7)}$$

In the third case, $2y = \lfloor M_{max} \rfloor - \lfloor M_{min} \rfloor$ and the conclusion is the same as in (5B-5). The intersection of these three requirements is (5B-5) or, equivalently, $\lceil y \rceil < 2^{(P-1)}$.

Appendix 5C: Proof of Correctness of the New Method of S_n Regeneration

Let RTS_n be represented by x_n, then rewriting (5-3) and using the identity $(x+y)modM = (xmodM + ymodM)modM$

$$x_{n+1} = (x_n mod 2^P + S_n mod 2^P) mod 2^P \tag{5C-1}$$

$x_n mod 2^P = x_n$ as x_n is a P bit number and $S_n mod 2^P = x_n$ by definition, then (5C-1) simplifies to

$$x_{n+1} = (2x_n) mod 2^P \tag{5C-2}$$

The first step of the regeneration procedure is the modulo 2^P subtraction of successive received RTS values. Expanding and applying the above identity

$$\begin{aligned}(x_{n+1} - x_n) mod 2^P &= (2x_n mod 2^P - x_n mod 2^P) mod 2^P \\ &= (2x_n - x_n) mod 2^P \\ &= x_n\end{aligned} \tag{5C-3}$$

and since $x_n = S_n mod 2^P$, $(x_{n+1} - x_n) mod 2^P = S_n mod 2^P$.

The next step of the regeneration process is the modulo 2^P subtraction of $\lfloor M_{min} \rfloor mod 2^P$ from x_n. Expanding and applying the identity,

$$\begin{aligned}(x_n - \lfloor M_{min} \rfloor mod 2^P) mod 2^P &= (S_n mod 2^P - \lfloor M_{min} \rfloor mod 2^P) mod 2^P \\ &= (S_n - \lfloor M_{min} \rfloor) mod 2^P\end{aligned} \tag{5C-4}$$

From (5B-2), $S_n \in \{\lfloor M_{min} \rfloor, \lfloor M_{min} \rfloor + 1, \ldots, \lfloor M_{max} \rfloor, \lfloor M_{max} \rfloor + 1\}$ and so it follows that $S_n - \lfloor M_{min} \rfloor \in \{0, 1, \ldots, \lfloor M_{max} \rfloor - \lfloor M_{min} \rfloor, \lfloor M_{max} \rfloor - \lfloor M_{min} \rfloor + 1\}$ and so from (5B-3) it can be concluded that

$$0 \le S_n - \lfloor M_{min} \rfloor < 2^P \tag{5C-5}$$

If $S_n - \lfloor M_{min} \rfloor < 2^P$ then $(S_n - \lfloor M_{min} \rfloor) mod 2^P = S_n - \lfloor M_{min} \rfloor$ and so using (5C-5) the final step becomes

$$(S_n - \lfloor M_{min} \rfloor) mod 2^P + \lfloor M_{min} \rfloor = S_n - \lfloor M_{min} \rfloor + \lfloor M_{min} \rfloor \tag{5C-6}$$

Appendix 5D: Existence of L and the nominal stuff ratio for a given x

Rearranging (5-15) and noting that if for a certain x, $\rho_1 < \mathrm{fr}\left(N\,\mathrm{fr}\left(\frac{\Upsilon}{x}\right)\right) < \rho_2$ then since $|\mathrm{fr}(c) - \mathrm{fr}(d)| < 1$

$$\begin{aligned} \left\lfloor N\,\mathrm{fr}\left(\frac{\Upsilon}{x}\right)\right\rfloor < L_1 < \left\lfloor N\,\mathrm{fr}\left(\frac{\Upsilon}{x}\right)\right\rfloor + 1 \\ \left\lfloor N\,\mathrm{fr}\left(\frac{\Upsilon}{x}\right)\right\rfloor - 1 < L_2 < \left\lfloor N\,\mathrm{fr}\left(\frac{\Upsilon}{x}\right)\right\rfloor \end{aligned} \qquad \text{(5D-1)}$$

hence $\lfloor L_2 \rfloor \neq \lfloor L_1 \rfloor$ and there exists an integer $L = \left\lfloor N\,\mathrm{fr}\left(\frac{\Upsilon}{x}\right)\right\rfloor = \lfloor L_1 \rfloor$.

Chapter 6
The Experimental Study of the Jitter Spectrum

6.1 Introduction

Previous chapters have developed the theory describing the jitter spectrum of the synchronization process. In particular, from the analysis of the synchronization process jitter in the presence of input jitter, it has been possible to predict the jitter spectrum when there is sinusoidal input jitter. Whenever possible, it is important to verify theoretical predictions with experimental evidence. The appearance of the spectrum of pulse stuffing jitter, which also contains waiting time jitter, has been verified experimentally by Duttweiler [22]. The presence of waiting time jitter in the jitter spectrum generated by the ideal SRTS method has been shown experimentally by [57]. However, experimental evidence of the appearance of the synchronization process jitter spectrum in the presence of sinusoidal input jitter has not previously been provided.

In this chapter, the results of an experimental investigation into the jitter spectrum of the synchronization process are described. The experimental approach is briefly outlined in Section 6.2. In Section 6.3, the experimental results are compared with the theoretical predictions of the synchronization process jitter spectrum without input jitter. In Section 6.4, original experimental work is presented which shows that the jitter spectrum in the presence of sinusoidal input jitter corresponds to the predictions in Section 3.7 of Chapter 3. Furthermore, the experiments suggest that sinusoidal input jitter might be useful in the suppression of unwanted waiting time jitter. This question is briefly examined in Section 6.5.

6.2 Description of the Experiment

In this section, a brief description of the experiment undertaken to verify the theoretical results found in Section 3.5 and Section 3.7 of Chapter 3 is given. Information is provided on the signal sources and measurement techniques. The experimental results obtained allow comparison with the theoretical plots for the synchronization process jitter spectrum both with and without input jitter.

A block diagram of the experimental system is shown in Figure 6.1. There are two inputs: a 10 MHz clock and the asynchronous input. The 10 MHz clock is divided down by four. The asynchronous input passes through an initialization block

controlled by a manual set-reset switch. The two outputs are the asynchronous input which is fed forward to an output and the synchronized output. Use of an EPLD (erasable programmable logic device) allows the synchronizer, the divide-by-four circuit and the initialization circuit to be implemented in a single IC.

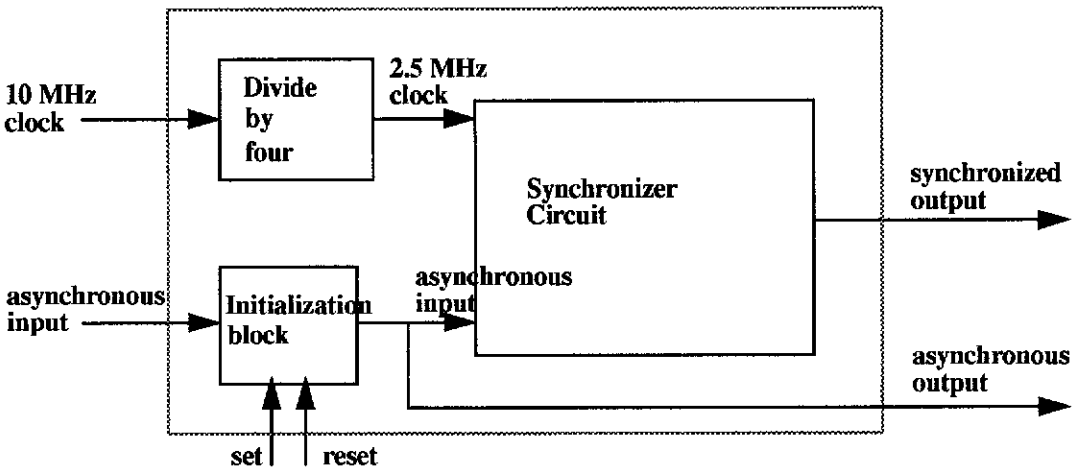


Figure 6.1: Functional block diagram of experimental system.

The measuring instrument used in the experiment is the HP5372A Time and Frequency Analyser which is used both as a signal source and as the measuring instrument. The local clock input for the synchronizer has to be provided by a very accurate and jitter free source so as not to introduce confounding factors into the experiment. The frequency standard 10 MHz clock output of the HP5372A is used. The asynchronous input is provided by an HP8904A signal generator, which also allows modulation of the signal, allowing generation of an asynchronous input affected by a known amount of jitter.

There are two ways to measure the jitter on the synchronizer output. When the asynchronous input to the synchronizer and the synchronized output of the synchronizer are both supplied to the HP5372A, it can measure the jitter directly by measuring the time intervals between the edges of two signals, this is the direct Time Interval measurement approach. Alternatively, the synchronized output can be analysed using the Time Deviation function of the HP5372A. The Time Deviation

function of the HP5372A allows the analysis of a jittered signal with respect to a reference signal. The value of the reference signal can be entered manually or can be calculated automatically [3]. In the experiments described in this chapter, automatic calculation of the reference signal was used.

6.3 Verification of the Synchronization Process Jitter Spectrum Model

6.3.1 Plotting the Predicted Jitter Spectrum

The jitter spectrum for the synchronization process without input jitter is given by (4-19) of Chapter 4 and shown below for reference

$$J(f) = F \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} d_k \delta(f - k\rho F - nF) \tag{6-1}$$

where $F = 1/T$ is the asynchronous input frequency, T_l is the local clock period and $\rho = \mathrm{fr}(T/T_l)$ and the coefficients d_k are

$$d_k = \begin{cases} \dfrac{T_l e^{-j2\pi k\left(\frac{t_0}{T_l}\right)}}{j2\pi k}, & k \neq 0 \\ \dfrac{T_l}{2}, & k = 0 \end{cases} \tag{6-2}$$

The N time domain samples collected by the HP5372A are windowed by an N-point Hanning window and a single-sided FFT of $N/2$ points is plotted from 0 to $F/2$ as the deviation from F, the sampling frequency [9]. In the jitter spectrum study, the sampling frequency is always equal to the asynchronous input frequency. The spectral resolution of the FFT produced by the HP5372A is given by $1/T_0$ where $T_0 = NT$ is the total duration of the time domain record [9].

For comparison purposes, some manipulation of the theoretical spectrum of (6-1), which shows the double-sided Fourier Transform, is required. It can be shown that, for $X(f)$, the Fourier Transform of some continuous time function $x(t)$, $T_0 w(k) \cdot x_p(kT)$ and $W(l) \otimes X_p\left(\frac{lF}{N}\right)$ form a Discrete Fourier Transform pair, where

$$x_p(t) = \sum_{k=-\infty}^{\infty} x(t-kT_0)$$
$$X_p(f) = \sum_{l=-\infty}^{\infty} X(f-lF) \quad (6\text{-}3)$$

$w(k)$ is the window function, $W(l)$ is the DFT of the window function and $\otimes$ is the convolution operator.

Hence, the FFT produced by the HP5372A is an approximation to $2W(l) \otimes \frac{2}{F}J(f)$ at $f = (lF)/N$. For the case of the synchronization process jitter spectrum, it can only be an approximation because, in general, the spectrum is not bandlimited and so it is always aliased.

The predicted spectrum, $J_T(f)$, can be calculated from

$$J_T(f) = 2W(l) \otimes 2 \sum_{k=-\infty}^{\infty} d_k \delta(f - (kp - \lfloor kp \rfloor)F) \quad (6\text{-}4)$$

for $0 < f < F/2$. As it is impossible to plot an infinite number of components, a maximum value for k must be chosen and it is convenient to choose $k_{max} = N/2$. The d_k values are calculated from

$$d_k = \left| \frac{T_l}{j2\pi k} \right| \quad (6\text{-}5)$$

and the DC component, $k = 0$, is not plotted as the experimental results do not show the contents of frequency bin 0 [9]. As is the case with the experimental plots, only the absolute value of the coefficient amplitudes vs frequency is plotted.

6.3.2 Experimental Results

The results of two measurements of the synchronization process jitter spectrum in the absence of input jitter are shown in Figure 6.2 and Figure 6.3. The bottom half of Figure 6.2 shows the result of Time Interval measurement where the asynchronous input frequency was nominally set to 123.131 kHz and 16 blocks of 4096 measurements were taken using spectral averaging. As the Time Interval measurement technique does not provide a measurement of the asynchronous input

frequency, the predicted jitter spectrum, shown in the top half of Figure 6.2, was calculated and plotted on the basis of the nominal frequency setting of 123.131 kHz. The spectral resolution of the experimental graph is 30.06 Hz/bin.

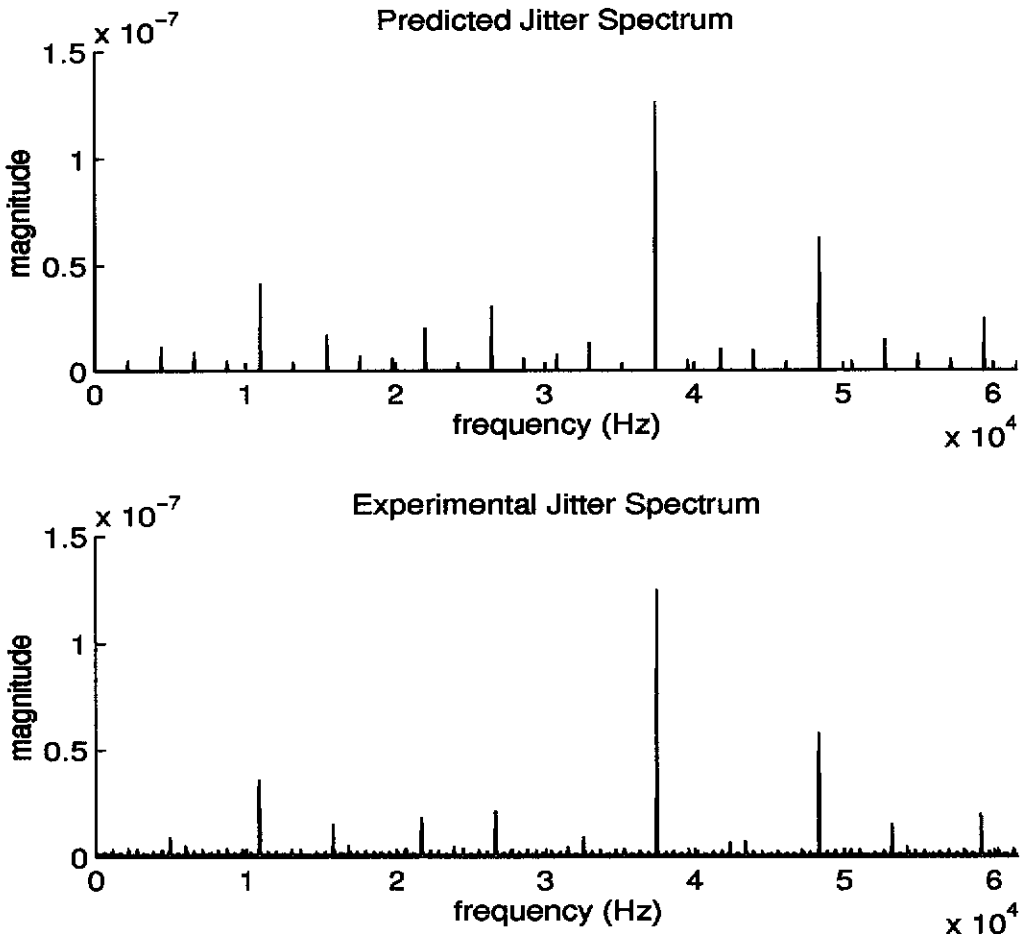


Figure 6.2: Predicted and Experimental Jitter Spectra of Time Interval measurement.

Table 6.1 compares the frequencies of the first twelve spectral components predicted by (6-1) with the experimental values as given by the midpoints of the frequency bins where the peak amplitude is found. Table 6.2 compares the amplitudes predicted by (6-4) and the experimental amplitudes. The predicted and experimental amplitude ratios are also compared.

Table 6.1: Spectral Frequencies for Time Interval Measurement

k	Predicted Frequency (Hz)	Experimental Frequency Bin Midpoint (Hz)
1	37380	37425
2	48371	48277
3	10991	10852
4	26389	26574
5	59362	59099
6	21982	21674
7	15398	15752
8	52778	53177
9	32973	32526
10	4407	4900
11	41787	42325
12	43964	43377

Table 6.2: Spectral Amplitudes for Time Interval Measurement

k	Predicted Amplitude (ns)	Experimental Amplitude (ns)	Predicted Amplitude ratio	Experimental Amplitude ratio
1	126.17	125.17	1	1
2	62.56	57.61	0.496	0.46
3	41.36	35.81	0.33	0.29
4	30.61	21.31	0.24	0.17
5	24.42	19.91	0.19	0.16
6	20.19	18.20	0.16	0.15
7	16.89	15.00	0.13	0.12
8	14.59	15.07	0.12	0.12
9	13.17	9.237	0.10	0.074
10	11.35	8.903	0.090	0.071
11	10.16	6.808	0.081	0.054
12	9.674	7.525	0.077	0.060

The bottom half of Figure 6.3 shows the result of Time Deviation measurement where the asynchronous input frequency was nominally set to 123.131 kHz and 8 blocks of 8192 measurements were taken using spectral averaging. The HP5372A calculated the asynchronous input frequency to be 123.12857207 kHz. The estimate of the

asynchronous input frequency provided by the HP5372A is used to plot the predicted jitter spectrum shown in the top half of Figure 6.3. The spectral resolution of the experimental graph is 15.03 Hz/bin.

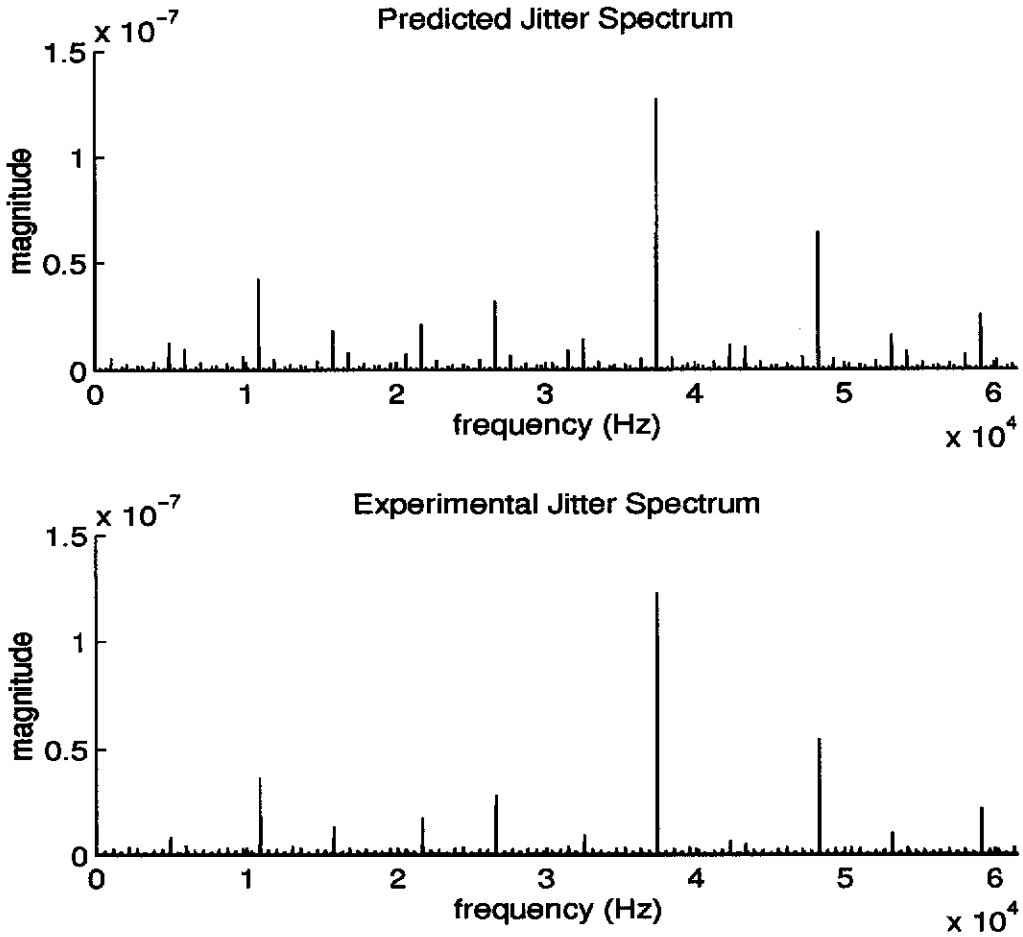


Figure 6.3: Predicted and Experimental Jitter Spectra of Time Deviation measurement.

Table 6.3 compares the frequencies of the first twelve spectral components predicted by (6-1) with the experimental values as given by the midpoints of the frequency bins where the peak amplitude is found. Table 6.4 compares the amplitudes predicted by (6-4) and the experimental amplitudes. The predicted and experimental amplitude ratios are also compared.

Table 6.3: Spectral Frequencies for Time Deviation Measurement

k	Predicted Frequency (Hz)	Experimental Frequency Bin Midpoint (Hz)
1	37429	37426
2	48271	48278
3	10843	10837
4	26586	26589
5	59114	59115
6	21686	21689
7	15743	15737
8	53171	53178
9	32529	32526
10	4900	4900
11	42328	42326
12	43372	43378

Table 6.4: Spectral Amplitudes for Time Deviation Measurement

k	Predicted Amplitude (ns)	Experimental Amplitude (ns)	Predicted Amplitude ratio	Experimental Amplitude ratios
1	127.31	122.47	1	1
2	63.65	54.52	0.50	0.45
3	42.42	36.52	0.33	0.30
4	31.82	28.45	0.25	0.23
5	25.45	22.13	0.20	0.18
6	21.20	17.52	0.17	0.14
7	18.17	13.34	0.14	0.11
8	15.90	10.45	0.125	0.085
9	14.13	9.539	0.11	0.078
10	12.72	8.463	0.10	0.069
11	11.56	6.863	0.091	0.056
12	10.59	5.420	0.083	0.044

The agreement in both measurement methods is good, showing that the experimentally determined spectrum does have the shape predicted. The absolute values of the experimental amplitudes are smaller than the predicted values and the amplitude ratios have better agreement than the absolute amplitude values.

There are several reasons for the discrepancy in absolute amplitude values. Firstly, although each peak in the predicted spectrum has a clear corresponding peak in the experimental spectrum, the two or three bins to each side also have amplitudes well above the noise floor, thus some of the energy from the spectral peak is found in the smaller but still significant peaks in the adjacent bins. This reflects the finite resolution and finite number of frequency bins in the FFT which is approximating a theoretical spectrum of delta functions occurring at specific frequencies which are not necessarily integer multiplies of F/N, the bin width. Secondly, although the Hanning window was applied in calculating both predicted and experimental spectra, spectral averaging was also used in the experimental case.

The agreement between predicted and experimental frequencies is very much better for the Time Deviation measurement function than for the Time Interval measurement, because the asynchronous input frequency is known more accurately for use in calculating the predicted spectrum. Agreement between predicted and experimental absolute amplitudes is about the same for the two measurement techniques.

6.4 Verification of Jitter Spectrum for Sinusoidal Input Jitter Case

6.4.1 Plotting the Predicted Jitter Spectrum

In Section 3.7 of Chapter 3 the jitter spectrum for the synchronizer when there is input jitter is derived. The actual appearance of such spectra, either predicted or experimentally obtained does not seem previously to have been investigated. In this section, theoretical plots which predict the appearance of the spectra will be given and compared with plots from experimental data.

The predicted spectrum can be calculated from

$$J_{T,IJ}(f) = \sum_{k=-\infty}^{\infty} 2\gamma_k \sum_{l=-\infty}^{\infty} J_l\left(\frac{2\pi k A_j}{T_l}\right)\delta(f - ((k\rho + l\vartheta) - \lfloor k\rho + l\vartheta \rfloor)F) - jA_j\delta(f - f_j) \quad (6\text{-}6)$$

for $0 < f < F/2$ and where $\vartheta = f_j/F$. It is necessary to specify, l, the number of input jitter dependent components to plot. When the frequency of the input jitter is small relative to the asynchronous input frequency, then it is necessary to specify a large number for l_{max}. For example, in the plots shown here, $k_{max} = 20, l_{max} = 60$. The coefficients of the jitter spectrum with sinusoidal input jitter have to be calculated for all k, l using

$$\left|\gamma_k J_l\left(\frac{2\pi A_j}{T_l}\right)\right| = \left|\frac{T_l}{j2\pi k} J_l\left(\frac{2\pi A_j}{T_l}\right)\right| \tag{6-7}$$

and only the absolute value of the component amplitudes vs frequency is plotted. Windowing of the predicted jitter spectrum is not performed as it is impractical due to the large number of components resulting from the input jitter aliasing.

The time of occurrence of the nth jittered edge of the asynchronous input signal affected by sinusoidal input jitter is given by

$$nT + A_j \sin(2\pi f_j nT) \tag{6-8}$$

Production of a signal with sinusoidal jitter directly affecting the timing of the significant edges is difficult, therefore it was decided to produce a good approximation to such a signal using sinusoidal phase jitter. The theoretical basis of this approach is described in Section 3.3 of Chapter 3.

To produce the sinusoidal phase jitter, FM modulation of a carrier using a sinusoidal waveform (FM tone modulation) was used. In FM tone modulation, the instantaneous frequency of the jittered signal is

$$f(t) = f_c + f_\Delta x(t) \tag{6-9}$$

where $x(t)$ is the modulating signal and f_c is the nominal frequency of the jittered signal when there is no jitter. From equation (3-5) of Chapter 3, the resulting sinusoidal phase jitter is given by $p(nT_c) = f_\Delta \int_{-\infty}^{t} x(\tau)d\tau \Big|_{t = nT_c}$.

To approximate sinusoidal time jitter, that is $e(t) = A_j \sin(2\pi f_j t + \theta_j)$, the required frequency deviation is

$$f_\Delta x(t) = -f_c 2\pi f_j A_j \cos(2\pi f_j t) \tag{6-10}$$

the modulating sinusoid is $x(t) = \sin(2\pi f_j t + (3\pi)/2)$, $f_\Delta = f_c 2\pi f_j A_j$ which is denominated in Hz and, without loss of generality, $\theta_j = 0$.

The conditions under which the approximation $e(nT_c) \approx (-T_c)p(nT_c)$ applies were discussed in Section 3.3 of Chapter 3 where it was shown that the condition

$$f_j A_j \ll \frac{1}{2\pi} \tag{6-11}$$

should apply in the case of sinusoidal jitter. Equation (6-11) represents the error involved in neglecting the next term in the Taylor series expansion. In the experiments described below, the biggest error is of the order of $2\pi(977)(3\times10^{-6}) \approx 2\times10^{-2}$ or 2%. From the experimental results, it can be seen that with this error the phase jitter acceptably approximates the timing jitter. However, the experiments also showed that when the error reached the order of 6%, the sinusoidal phase jitter no longer accurately modelled the timing jitter.

6.4.2 Experimental Results

The experiments undertaken to visualise the jitter spectrum in the presence of sinusoidal input jitter are summarised in Table 6.5. The figures given in this table are the nominal settings of the asynchronous input frequency and the input jitter frequency and amplitude.

Table 6.5: Settings of Jitter Spectrum Experiments with Sinusoidal Input Jitter

Exp.No.	Asynchronous input frequency (kHz)	Jitter Frequency (Hz)	Jitter Amplitude (Hz)	Jitter Phase (degrees)	Measurement Type	Spectral Averaging No. of blocks No. of samples
3	123.131	5849.8	576.2	0	Time Interval	16 4096
4	123.131	5849.8	576.2	0	Time Deviation	8 8192
5	83.191	459	360	270	Time Deviation	32 8192
6	84.977	459	360	270	Time Deviation	32 8192
7	83.191	977	1534	270	Time Deviation	32 8192
8	84.977	977	1534	270	Time Deviation	32 8192

The results of the measurements made in the jitter spectrum experiments with sinusoidal input jitter are given in Table 6.6. The figures for the asynchronous input frequency and the input jitter frequency and amplitude are as measured by the HP5372A.

Table 6.6: Results of Jitter Spectrum Experiments with Sinusoidal Input Jitter

Exp. No.	Measured asynchronous input frequency (kHz)	ρ	Measured jitter frequency (Hz)	Measured jitter amplitude (us)	Figure Number	Spectral resolution (Hz/bin)
3	123.131	0.303579110	5849.8	0.1273	Figure 6.4	30.06
4	123.12856597	0.303980490	5849.8	0.127319	Figure 6.5	15.03
5	83.189327724	0.051931760	456.98	1.464	Figure 6.6	15.03
6	84.975248648	0.420337680	456.42	1.413	Figure 6.7	15.03
7	83.189320112	0.051934510	974.90	2.921	Figure 6.8	15.03
8	84.975330389	0.420303380	975.03	2.874	Figure 6.9	15.03

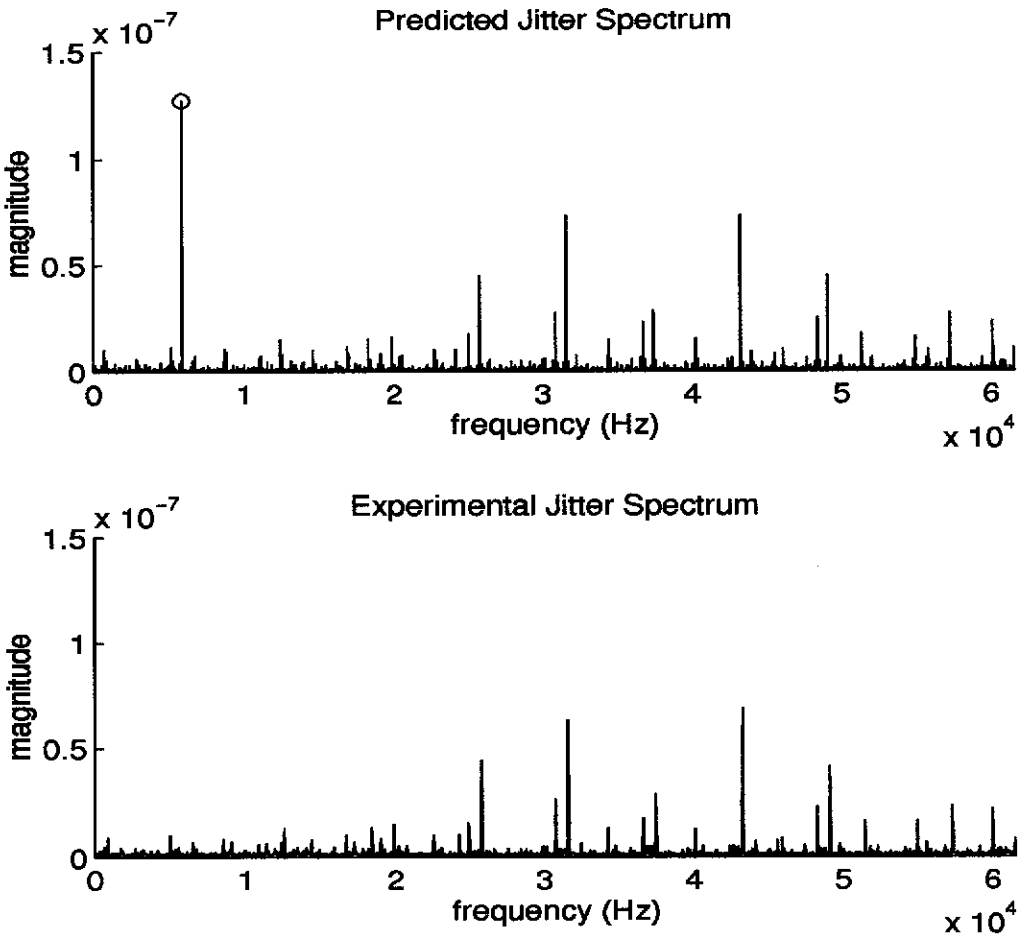


Figure 6.4: Predicted and Experimental Jitter Spectra for Experiment 3.

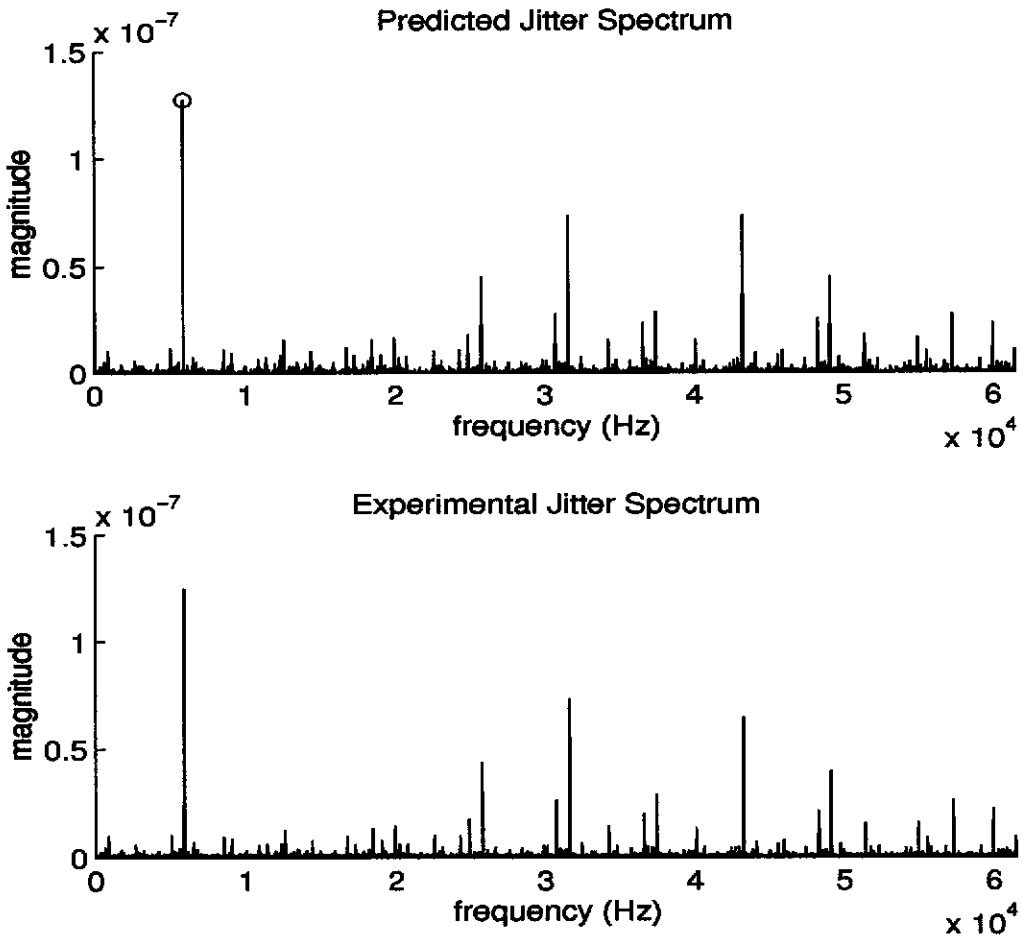


Figure 6.5: Predicted and Experimental Jitter Spectra for Experiment 4.

Comparison of the predicted and experimental jitter spectra for Experiment 3 and Experiment 4 shows that both the Time Interval and the Time Deviation approaches to measurement are successful in capturing the synchronization process jitter spectrum with sinusoidal input jitter. Note that the Time Interval measurement does not detect the feedthrough component of the input jitter because this measurement method can only detect the jitter between the jittered asynchronous input and the synchronized output. The Time Interval measurement approach has no way to reference the jittered asynchronous input to the ideal, unjittered asynchronous input. The Time Deviation method of measurement reconstructs an unjittered reference input and thus does detect the feedthrough jitter component. In order to be able to compare the magnitudes of the synchronization process jitter and the input jitter, only the Time Deviation method is used in the subsequent experiments.

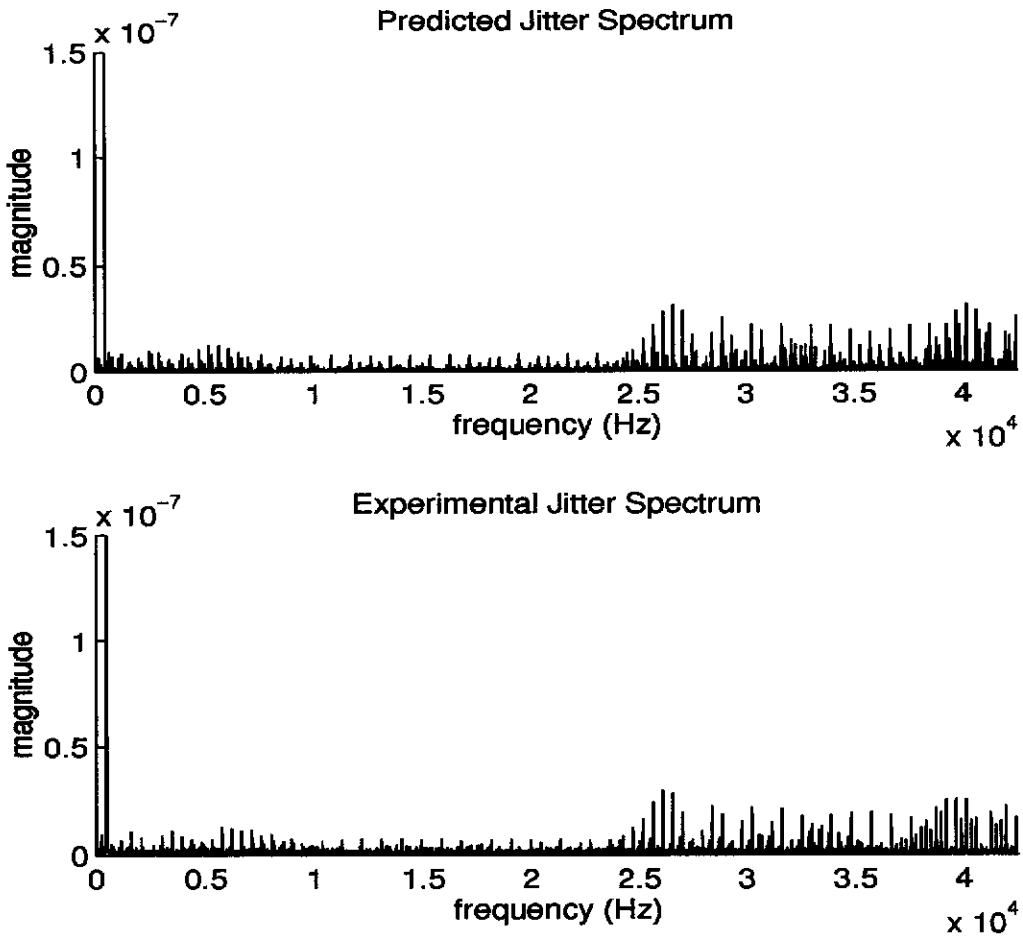


Figure 6.7: Predicted and Experimental Jitter Spectra for Experiment 6.

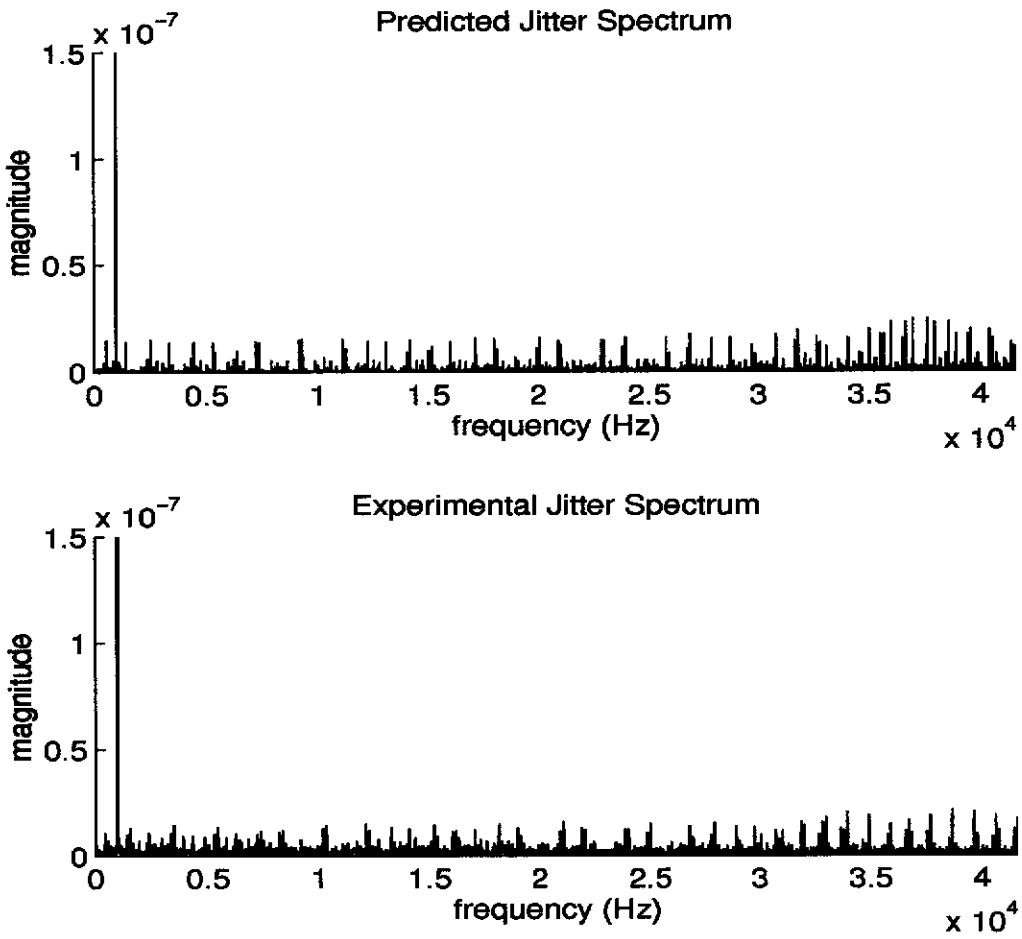


Figure 6.8: Predicted and Experimental Jitter Spectra for Experiment 7.

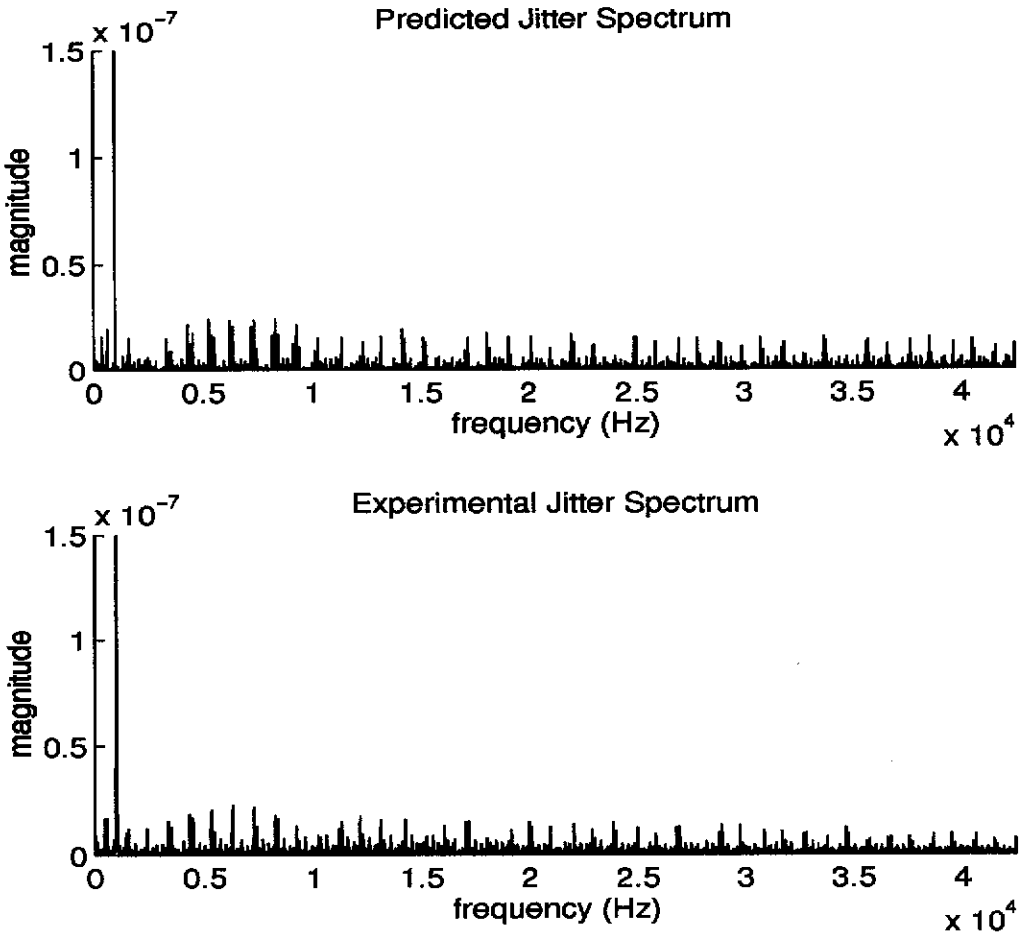


Figure 6.9: Predicted and Experimental Jitter Spectra for Experiment 8.

In all cases illustrated above, the predicted and experimental jitter spectra show a very similar appearance. Spectral components appear at the same frequencies and with similar relative amplitudes. The amplitude of the multiply aliased synchronizer jitter relative to the much greater amplitude of the feedthrough jitter is the same in the pairs of plots. Confirmation is thus provided that the sinusoidal phase jitter is adequately approximating the sinusoidal timing jitter and that the spectrum of the synchronization process jitter is as predicted when there is sinusoidal input jitter.

6.5 Use of Sinusoidal Input Jitter as Jitter Reduction Mechanism

An interesting effect of the sinusoidal input jitter is the reduction in amplitude of the synchronization process jitter components and the shifting in location of the most prominent spectral components. Similar effects can be obtained by modification of the

threshold used in stuffing decision circuits. The effect of stuff threshold modulation and adaptive threshold modulation has been examined [24], [36], [45], [48], [58].

As shown in Section 3.7 of Chapter 3, when there is input jitter the total RMS power in the waiting time jitter remains the same and there is the addition of the RMS power of the feedthrough component of the input jitter. However, the multiplication of the magnitude of the spectral lines by the Bessel functions reduces the magnitude of individual spectral lines, since $|J_l(\beta)| < 1, l \neq 0, \beta \neq 0$, and the replication of the synchronizer jitter spectral lines around each multiple of the input jitter frequency spreads the jitter energy over a wide frequency range and thus flattens out its peaks.

Comparing the plots from the different experiments, it appears that for a given ρ value, the frequency and amplitude of the sinusoidal input jitter can influence how effectively the unwanted peaks are flattened. In Figure 6.8, for example, the higher frequency input jitter with a larger amplitude appeared to be more effective at reducing the low frequency spectral lines in the synchronizer jitter than the lower frequency, smaller amplitude input jitter in Figure 6.6. In Figure 6.7, however, the lower frequency, smaller amplitude input jitter is more effective in reducing the amount of jitter at lower frequencies than the higher frequency, larger amplitude jitter in Figure 6.9.

However, given a particular value of ρ and the corresponding synchronization jitter, it is not trivial to determine the frequency and amplitude of sinusoidal input jitter which would reduce the low frequency synchronization jitter. Due to the presence of Bessel functions in the spectrum, there is a complex interaction between the effects of the amplitude and the frequency of the sinusoidal input jitter. Further study is required to investigate the choice of suitable amplitude-frequency input jitter combinations.

It must also be noted that jitter reduction using sinusoidal input jitter would only be useful where a PLL is used to filter the resulting timing signal as any feedthrough jitter must be reduced to an acceptable level. It is therefore necessary to establish whether, after filtering by a PLL, the amount of jitter reduction achieved compensates for the feedthrough jitter.

6.5.1 An Example of Jitter Reduction Using Sinusoidal Input Jitter

In this section, an example of jitter reduction using sinusoidal input jitter will be given for the case of SRTS timing transfer. The aim is to examine briefly the proposed method of jitter reduction to see if it merits further study. The example is based on the use of SRTS timing transfer in the DS-3 CLAD (Cell Assembly and Disassembly device) described in [57].

The parameters are as follows: the SRTS method reference clock with frequency, $f_l = 1/T_l = 77.76$ MHz, corresponds to the local clock, as described in this chapter; the service clock to be transported by the SRTS method has nominal frequency, $f_{s,\,nom} = 44.736$ MHz; and with $N = 3008$, the asynchronous input frequency is $F = 14872.3404$ Hz and $\rho = 0.497854$. [57] shows that for $f_{s,\,nom}$, $\rho = 116/233$, which is very close to $1/2$, and waiting time jitter peaks appear for $k \geq 2$ at 63.8 Hz and its harmonics. [57] also shows that this low frequency waiting time jitter cannot be suppressed by a simple PLL with transfer function equivalent to a first-order low pass filter, that is,

$$Y(s) = \frac{\alpha}{s+\alpha} \tag{6-12}$$

and with cut-off frequency $\omega_\alpha = \alpha$ equal to 350 rad/s or 41 rad/s.

Figure 6.10 compares the predicted unfiltered waiting time jitter, shown in (a), with the expected outcome of filtering or use of sinusoidal input jitter plus filtering. The waiting time jitter when filtered by a first-order low pass filter with $f_\alpha = 6.5$ Hz and

$$|Y(f)| = \frac{1}{\sqrt{1+\left(\frac{f}{f_\alpha}\right)^2}} \tag{6-13}$$

is shown in (b). Two examples of the possible use of sinusoidal input jitter applied to the asynchronous input as a jitter reduction mechanism are shown in (c) 13.5 ns/3900 Hz sinusoidal input jitter and (d) 9 ns/6000 Hz sinusoidal input jitter where both are followed by filtering by the first-order low pass filter of (6-13) but with $f_\alpha = 56$ Hz.

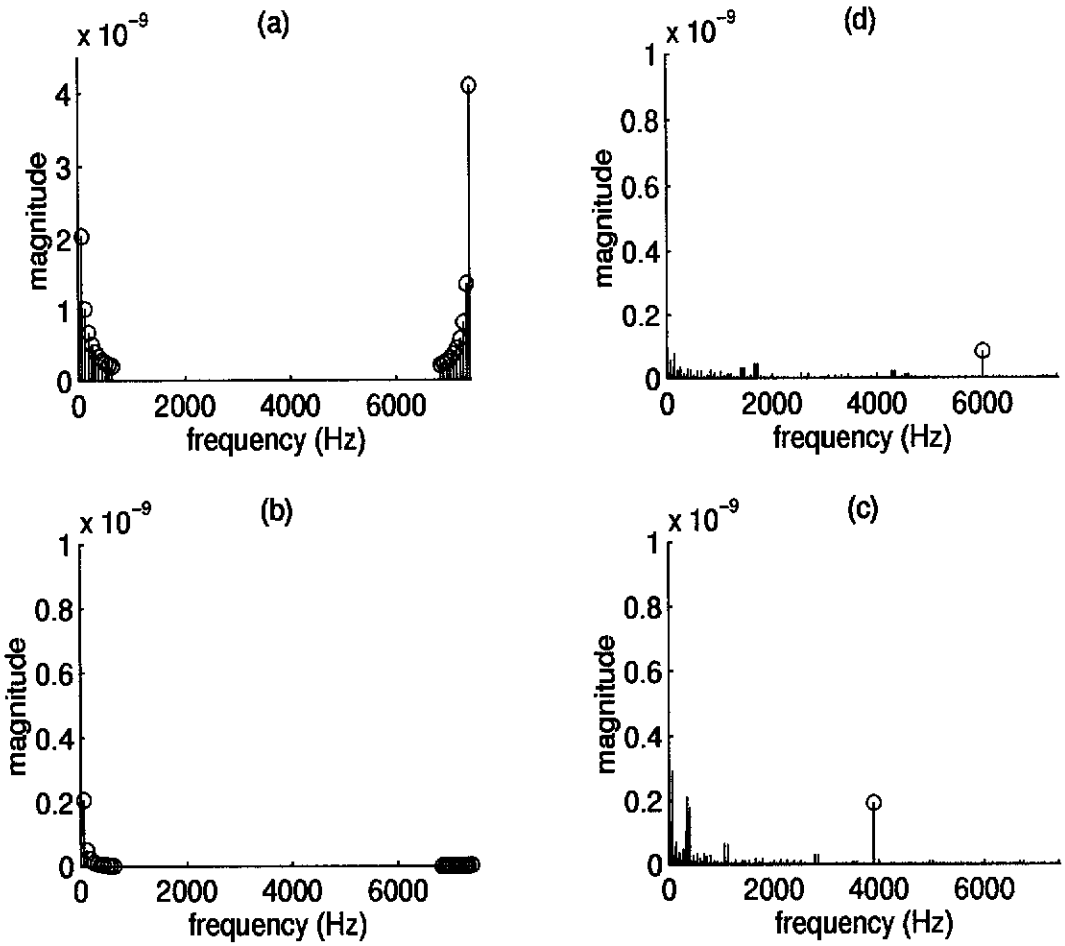


Figure 6.10: Jitter Reduction from Sinusoidal Input Jitter: SRTS method example.
(a) unfiltered, (b) filtered, (c) 3900 Hz jitter, (d) 6000 Hz jitter.

The amount of input jitter used has been kept small. At 3900 Hz, the amplitude of the input jitter is 0.604 UI and at 6000 Hz, it is 0.403 UI, where 1 UI = $1/f_{s,nom}$. A considerable amount of jitter reduction is achieved particularly by the 9 ns/6000 Hz sinusoidal input jitter. The use of the sinusoidal input jitter achieves jitter reduction comparable to the narrower bandwidth PLL alone and allows the use of a wider bandwidth PLL with its corresponding faster response time. After filtering, the feedthrough jitter is present at a level comparable to the rest of the jitter spectrum.

Appendix 6A: Glossary

T, F - period and frequency of the asynchronous input

N - number of samples collected by the HP5372A

$T_0 = NT$ - total duration of the time record collected by the HP5372A

η - ratio of the asynchronous input period to the local clock period

T_l, f_l - local clock period and frequency

ρ - $\mathrm{fr}(\eta)$, parameter determining the location of the spectral lines in waiting time jitter

d_k - coefficients of the jitter spectrum for no input jitter

$J(f)$ - synchronization process jitter spectrum for no input jitter

$J_T(f)$ - theoretical synchronization process jitter spectrum for no input jitter

$J_{T,IJ}(f)$ - theoretical synchronization process jitter spectrum when there is input jitter

γ_k - coefficients of the jitter spectrum when there is input jitter

$w(k)$ - window function

$W(l)$ - Fourier Transform of window function

f_j, A_j, θ_j - frequency, amplitude and phase of sinusoidal input jitter

$f(t)$ - instantaneous frequency in FM tone modulation

f_c - carrier frequency in FM tone modulation

$x(t)$ - modulating signal in FM tone modulation

f_Δ - frequency deviation constant in FM tone modulation

$e(nT_c), p(nT_c)$ - samples of time jitter and phase jitter

$E(f)$ - feedthrough spectrum of the input jitter

ϑ - ratio of input jitter frequency to the asynchronous input frequency

$f_{s,nom}$ - nominal service clock frequency in the SRTS method

$Y(s), Y(f)$ - transfer function of simple PLL

ω_α, f_α - cut-off frequency of first-order loop approximation to PLL in radians and Hz

Chapter 7

Further Work

The metastable failure performance of the synchronizer designs presented in this thesis was analysed using the sampling model [34] of metastability transfer. This model is simple but requires the assumption that the devices have identical parameters and that the device behaviour while resolving is monotonic. It also does not allow for further metastable triggering in the second and subsequent devices.

A second approach to metastability transfer exists [49] which assumes that the devices have an ideal Schmitt Trigger output which effectively shields the following device from the metastability except for the uncertainty the metastability produces in the transition time of the first device's output. Making such an assumption allows the use of the metastable window model for the triggering of metastability in the subsequent device. However, it also biases the analysis because it removes one of the possible mechanisms for triggering metastability in the subsequent device, that is, the presence of resolving metastable voltages at the output of the preceding device.

The development of a more accurate model of metastable transfer is a challenging area of metastability research. A model which properly accounts for both the metastable state sampling and possible subsequent metastability triggering would allow more accurate and reliable determination of the performance of complex synchronizer designs. It is also important to obtain experimental verification for this model.

In the development of the model of the SRTS timing transfer method, it was assumed that the destination reference clock was jitter free. If the destination reference clock is jittered, an additional term will appear in the jitter on the reconstructed timing signal at the destination. The additional term is due to the sampling of the reference clock jitter at the rate of the reconstructed timing signal. However, the period of the reconstructed timing signal is not a constant number of reference clock cycles and so the sampling of the reference clock jitter is non-periodic. Determination of the full spectrum of the jitter in these circumstances needs further study.

In Chapter 6, it was shown that sinusoidal input jitter could be used to suppress waiting time jitter allowing the use of a wider bandwith PLL, for example, when filtering the recovered timing signal at the destination in the SRTS method. An

effective algorithm for selecting appropriate input jitter frequency and amplitude combinations is a worthwhile area of further research.

References

[1] Application Notes, SN74AS3374 Metastable-Resistant Octal D-type Edge Triggered Dual Ranking Flip-flops with 3-State Outputs: Metastable Characteristics, Texas Instruments Inc., Dallas.

[2] Application Notes, SN74AS4374 Metastable-Resistant Octal D-type Edge Triggered Dual Ranking Flip-flops with 3-State Outputs: Metastable Characteristics, Texas Instruments Inc., Dallas.

[3] HP 5372A Frequency and Time Interval Analyser: Operating Manual, Hewlett-Packard, 2nd ed., Aug. 1992.

[4] IEEE Standard 802.6-1990, *Distributed queue dual bus (DQDB) subnetwork of a metropolitan area network (MAN)*, IEEE, NEW York, July 1991.

[5] ITU-T Study Group XVIII, "B-ISDN ATM Adaptation Layer (AAL) Specification", Recommendation I.363, Geneva, Mar. 1993.

[6] ITU-T Study Group 13, "Traffic Control and Congestion Control in B-ISDN", Recommendation I.371, Paris, Mar. 1995.

[7] ITU-T Recommendation G.742, "Second Order Digital Multiplex Equipment Operating at 8448 kbit/s and Using Positive Justification", Fascicle III.4, Blue Book, Geneva, 1993.

[8] ITU-T Recommendation G.823, "The Control of Jitter and Wander Within Digital Networks Which are Based on the 2048 kbit/s Hierarchy", ITU, Geneva, Mar. 1993.

[9] User's Guide: HP 5372A Option 040 Jitter Spectrum Analysis, Hewlett-Packard, 1st ed., Aug. 1991.

[10] S. S. Abeysekara, "Jitter Reduction in Synchronous Networks", Technical Report: NRL-TR-10, Australian Telecommunications Research Institute, Curtin University of Technology, Perth, 1993.

[11] K. Asatani, K. R. Harrison and R. Ballart, "CCITT Standardization of Network Node Interface of Synchronous Digital Hierarchy", *IEEE Communications Magazine*, vol. 28, no. 8, pp. 15-20, Aug. 1990.

[12] J. C. Barros and B. W. Johnson, "Equivalence of the Arbiter, the Synchronizer, the Latch and the Inertial Delay", *IEEE Trans. Comput.*, vol. 32, no. 7, pp. 603-614, July 1983.

[13] J. C. Bellamy, "Digital Network Synchronization", *IEEE Communications Magazine*, vol. 33, no. 4, pp. 70-83, April 1995.

[14] A. Cantoni, personal communication.

[15] J. Calvo, J. I. Acha and M. Valencia, "Asynchronous Modular Arbiter", *IEEE Trans. Comput.*, vol. 35, no. 1, pp. 67-70, Jan. 1986.

[16] I. Catt, "Time Loss Through Gating of Asynchronous Logic Signal Pulses", *IEEE Trans. Electronic Comput.* (Short Notes), vol. EC-15, no. 1, pp. 108-111, Feb. 1966.

[17] T. J. Chaney and C. E. Molnar, "Anomalous Behavior of Synchronizer and Arbiter Circuits", *IEEE Trans. Comput.*, vol. 22, no. 4, pp. 421-422, Apr. 1973.

[18] J. P-C. Chin, "Jitter Transfer Characteristics of Phase-Locked Loops", Technical Report: NRL-TR-12, Australian Telecommunications Research Institute, Curtin University of Technology, Perth, 1995.

[19] P. E. K. Chow, "Jitter Due To Pulse Stuffing Synchronization", *IEEE Trans. Commun.*, vol. 21, no. 7, pp. 854-859, July 1973.

[20] P. E. K. Chow, "Generalised Formula for Pulse Stuffing Jitter", *Electronics Letters*, vol. 22, no. 24, pp. 1313-1314, 20 Nov. 1986

[21] C. C. Cock, A. E. Edwards and A. Jessop,"Timing Jitter in Digital Line Systems", IEE Conference Publication no. 131, *Telecommunication Transmission*, pp. 33-36, Sep. 1975.

[22] D. L. Duttweiler, "Waiting Time Jitter", *The Bell System Technical Journal*, vol. 51, no. 1, pp. 165-207, Jan. 1972.

[23] W. Fleischhammer and O. Dortok, "The Anomalous Behavior of Flip-Flops in Synchronizer Circuits", *IEEE Trans. Comput.*, vol. 28, no. 3, pp. 273-276, Mar. 1979.

[24] W. D. Grover, T. E. Moore and J. A. McEachern, "Waiting Time Jitter Reduction by Synchronizer Stuff Threshold Modulation", in GLOBECOM'87, pp. 13.7.1-13.7.5, 1987.

[25] R. Handel and M. N. Huber, *Integrated Broadband Networks: an introduction to ATM-based networks*, Addison-Wesley, 1991.

[26] J. H. Hohl, W. R. Larsen & L. C. Schooley, "Prediction of Error Probabilities for Integrated Digital Synchronizers", *IEEE J. Solid-State Circuits*, vol. 19, no. 2, pp. 236-244, Apr. 1984.

[27] J. E. Horstmann, H. W. Eichel & R. L. Coates, "Metastability Behavior of CMOS ASIC Flip-Flops in Theory and Test", *IEEE J. Solid-State Circuits*, vol. 24, no. 1, pp. 146-157, Feb. 1989.

[28] T. A. Jackson and A. Albicki, "Analysis of Metastable Operation in D Latches", *IEEE Trans. Circuits and Systems*, vol. 36, no. 11, pp. 1392-1404, Nov. 1989.

[29] T. Kacprzak, "Analysis of Oscillatory Metastable Operation of an RS Flip-Flop", *IEEE J. Solid-State Circuits*, vol. 23, no. 1, pp. 260-266, Feb. 1988.

[30] T. Kacprzak and A. Albicki, "Analysis of Metastable Operation in RS CMOS Flip-Flops", *IEEE J. Solid-State Circuits*, vol. 22, no. 1, pp. 57-64, Feb. 1987.

[31] D. J. Kinniment and J. V. Woods, "Synchronisation and arbitration circuits in digital systems", *Proc. IEE*, vol. 123, no. 10, pp. 961-966, Oct. 1976.

[32] L. Kleeman, "The Jitter Model for Metastability and Its Application to Redundant Synchronizers", *IEEE Trans. Comput.*, vol. 39, no. 7, pp. 930-942, July 1990.

[33] L. Kleeman and A. Cantoni, "On the Unavoidability of Metastable Behavior", *IEEE Trans. Comput.*, vol. 36, no. 1, pp. 109-111, Jan. 1987.

[34] L. Kleeman and A. Cantoni, "Metastable Behaviour in Digital Systems", *IEEE Design and Test of Computers*, vol. 4, no. 6, pp. 4-19, Dec. 1987.

[35] N. Kuroyanagi, H. Saito, S. Kozuka & Y. Okamoto, "Multiplexing Systems for PCM Hierarchial Systems", *IEEE Trans. Commun.*, vol. 17, no. 6, pp. 711-720, Dec. 1969.

[36] R. G. Kusyk, T. E. Moore and W. A Krzymien, "Spectral Analysis of Waiting Time Jitter in the Presence of Stuff Threshold Modulation", *Electron. Lett.*, vol. 26, no. 8, pp. 526-528, 12 Apr. 1990.

[37] R. C. Lau and P. E. Fleischer, "Synchronous techniques for timing recovery in BISDN," in GLOBECOM'92, pp. 814-820, 1992.

[38] C-S. Li and Y. Ofek, "Distributed Source-Destination Synchronization Using Inband Clock Distribution", *IEEE J. Select. Areas Commun.*, vol. 14, no. 1, pp. 153-161, Jan. 1996.

[39] W. Y-P. Lim and J. R. Cox Jr, "Clocks and the performance of synchronisers", *IEE Proceedings*, Pt E, vol. 130, no. 2, pp. 57-64, Mar. 1983.

[40] L. R. Marino, "General Theory of Metastable Operation", *IEEE Trans. Comput.*, vol. 30, no. 2, pp. 107-115, Feb. 1981.

[41] G. Mercankosk and A. Cantoni, "Characterization of a CBR Connection over a Channel with Known Bounded Delay Variation", *Proc. IEEE INFOCOM'93*, San Francisco, vol. 3, pp. 1170-1177, Mar. 1993.

[42] D. G. Messerschmitt, "Synchronization in Digital System Design", *IEEE J. Select. Areas Commun.*, vol. 8, no. 8, pp. 1404-1419, Oct. 1990.

[43] H. Meyr, L. Popken and H. R. Mueller, "Synchronization Failures in a Chain of PLL Synchronizers", *IEEE Trans. Commun.*, vol. 34, no. 5, pp. 436-445, May 1986.

[44] S. E. Minzer, "Broadband ISDN and Asynchronous Transfer Mode (ATM)", *IEEE Communications Magazine*, vol. 27, no. 9, pp. 17-24, Sep. 1989.

[45] T. E. Moore, R. G. Kusyk and W. A. Krzymien, "Waiting Time Jitter Results for a Reduced Complexity SONET Interface Circuit", *Electron. Lett.*, vol. 26, no. 22, pp. 1884-1886, 25 Oct. 1990.

[46] K. Murakami, "Jitter in Synchronous Residual Time Stamp", *IEEE Trans. Commun.*, vol. 44, no. 6, pp. 742-748, June 1996.

[47] F. F. E. Owen, *PCM and Digital Transmission Systems*, McGraw-Hill, New York, 1982.

[48] G. F. Pierobon and R. P. Valussi, "Jitter Analysis of a Double Modulated Threshold Pulse Stuffing Synchronizer", *IEEE Trans. Commun.*, vol. 39, no. 4, pp. 594-602, April 1991.

[49] A. Pfister, *Metastability in digital circuits with emphasis on CMOS technology*, Hartung-Gorre, Konstanz, 1989.

[50] L. M. Reyneri, D. Del Corso and B. Sacco, "Oscillatory Metastability in Homogeneous and Inhomogeneous Flip-flops", *IEEE J. Solid-State Circuits*, vol. 25, no. 1, pp. 254-264, Feb. 1990.

[51] F. Rosenberger and T. J. Chaney, "Flip-Flop Resolving Time Test Circuit", *IEEE J. Solid-State Circuits*, vol. 17, no. 4, pp. 731-738, Aug. 1982.

[52] F. E. Ross and J. R. Hamstra, "Forging FDDI", *IEEE J Select. Areas Commun.*, vol. 11, no. 2, pp. 181-190, Feb. 1993.

[53] R. P. Singh, S-H. Lee and C-K. Kim, "Jitter and Clock Recovery for Periodic Traffic in Broadband Packet Networks", *IEEE Trans. Commun.*, vol. 42, no. 5, pp. 2189-2196, May 1994.

[54] W. K. Stewart and S. A. Ward, "A Solution to a Special Case of the Synchronization Problem", *IEEE Trans. Comput.*, vol. 37, no. 1, pp. 123-125, Jan. 1988.

[55] P. A. Stoll, "How to Avoid Synchronization Problems", *VLSI Design*, pp. 56-59, November/December 1982.

[56] P.R. Trischitta and E. L. Varma, *Jitter in Digital Transmission Systems*, Artech House, Boston, 1989.

[57] H. Uematsu and H. Ueda, "Implementation and experimental results of CLAD using SRTS method in ATM networks", in GLOBECOM'94, pp. 1815-1821, 1994.

[58] R. Urbansky, "Pointer Processing for Synchronization Signals in SDH Equipment", ETSI STC-TM3, TD 2/39, The Hague, April 8-12, 1991.

[59] H. J. M. Veendrick, "The Behavior of Flip-Flops Used as Synchronizers and Prediction of Their Failure Rate", *IEEE J. Solid-State Circuits*, vol. 15, no. 2, pp. 169-176, Apr. 1980.

[60] J. Walker and A. Cantoni, "A New Design for a Frame Sampling Synchronizer", in *Proc. IEEE Int. Symp. Circuits and Systems*, London, vol. 3, pp. 97-100, June 1994.

[61] J. Walker and A. Cantoni, "A New Synchronizer Design" *IEEE Trans. Comput.*, vol. 45, no. 11, pp.1308-1311, Nov. 1996.

[62] J. Walker and A. Cantoni, "Jitter Analysis for Two Methods of Synchronization for External Timing Injection", *IEEE Trans. Commun.*, vol. 44, no. 2, pp. 269-276, Feb. 1996.

[63] M. Warner, "Jitter in Cascaded Links Employing Constrained Stuffing", A. T. R., vol. 23, no. 2, pp. 11-22, 1989.