PERFORMANCE ANALYSIS OF SYNCHRONIZATION CIRCUITS

A thesis submitted to the University of Manchester
for the degree of Master of Philosophy
in the Faculty of Engineering and Physical Sciences

2010

By
Zhen Zhang
School of Computer Science

Contents

List of Tables

List of Figures

Abstract

Synchronization interfaces are necessary when signals from one clock domain are imported into another. As multiple different clocks become increasingly common on chips, synchronizers also proliferate. To achieve high performance it is important that the system designer is aware of the timing characteristics of different synchronizers -which are non-deterministic by nature and can choose a design to meet their system requirements. This thesis presents a systematic method for analyzing and depicting behaviour of synchronizers and applies it to three widely recognized designs. The major contributions of this thesis are outlined as follows:

A method of analyzing probabilistic behaviour of several major synchronizer performances has been proposed. Analytical expressions for predicting single-word-transfer synchronizer performance, and the average cases are derived for certain clock relationships.

The cycle time dependencies are studied in detail for these synchronizers. The synchronizers are firstly modelled and abstractions of cycle time information is obtained from the analyses of the model simulation.

Extension of the above analysis is made to predict synchronizer performances under burst-mode data transfer. Effect of each data transfer on the next one is fully investigated and analytical models are drawn to describe this behaviour for certain clock relationships. The resultant influence on overall synchronizer performance is then evaluated. The Synchronizer Data Cycle Analyzer (SyDCA) was developed based on these results.

A new synchronizer, the three-flop synchronizer design is proposed. It originates from the analyses of the known synchronizer data cycle times in terms of their phase relationships. Preliminary investigation is carried out to analyze its performance and reliability.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns any copyright in it (the "Copyright") and s/he has given The University of Manchester the right to use such Copyright for any administrative, promotional, educational and/or teaching purposes.

ii. Copies of this thesis, either in full or in extracts, may be made only in accordance with the regulations of the John Rylands University Library of Manchester. Details of these regulations may be obtained from the Librarian. This page must form part of any such copies made.

iii. The ownership of any patents, designs, trade marks and any and all other intellectual property rights except for the Copyright (the "Intellectual Property Rights") and any reproductions of copyright works, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property Rights and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property Rights and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and exploitation of this thesis, the Copyright and any Intellectual Property Rights and/or Reproductions described in it may take place is available from the Head of School of School of Computer Science (or the Vice-President).

Acknowledgements

First, I would like to thank my supervisor, Dr. Jim Garside. For the last two years, Jim has been patient enough to mentor me. I have grown tremendously through his invaluable guidance and teaching. I have learnt many things from Jim in regard to research, academia and life in general.

Next, I would like to thank my parents, who have been supporting me in the past many years. It is their encouragement that helped me to arrive at this juncture.

I would also like to extend my gratitude to the people at APT group, who welcomed me two years ago and have provided a friendly atmosphere for me to carry out my everyday work.

Chapter 1

Introduction

1.1 Background and Overview of Synchronizers

Future VLSI systems will integrate an increasing number of modules and operate at faster clock frequencies than nowadays. This may mean that the popular synchronous design paradigm will become less useful as technology progresses. Advances in silicon technology also complicate the design style where a single clock is distributed on an entire chip [Fri01]. In a fully synchronous design, the global clock must be distributed in a way that all clocked elements can receive clock signals at the same time and that communicating units can pass data safely within the clock period. This requirement can be easily achieved in 'old-fashioned' processes with slow transistors and fast wires, because the wire delay is negligible. A standard design approach is that a single clock is generated and distributed together with other locally generated signals such as complementary clocks, pulses and delayed clocks where needed [WH04]. However, as operating frequency gets faster and process shrinks, it is becoming increasingly difficult to maintain low delays in clock distribution[MN06]. The consequence of clock distribution would cause a skew of several times of the cycle time between clocks near and far from the clock generator [WH04]. Also, random skew, drift and jitter from the clock distribution are proportional to the delay through the network because they are caused by process or environmental variations in the distribution elements [WH04].

As single clock distribution encounters these difficulties, an alternative is to provide each module with its own clock. This approach can effectively avoid the

said difficulties in distributing a synchronous clock across an entire chip. Additionally, it can easily meet the requirement of integrating modules with different operating frequencies on a System on Chip (SoC) device. Another motivating factor for independently clocking modules is to allow clock frequencies and voltages to change dynamically during chip operation [SAD+02, NNSvB94].

Dfficulties in clock distribution, different module frequencies on a single chip and low power assumption requirements call for new design paradigm rather than the conventional hierarchy-of-clocks approach. One promising solution is to construct a Globally Asynchronous Locally Synchronous (GALS) system [Cha84]. A SoC device with GALS architecture typically consists of multiple modules, each operating with a potentially unrelated local clock. Global communication between different modules is asynchronous, where no common clock is used to implement sequencing. Asynchronous solution eliminates complicated clock distributions and re-synchronization of signals is not needed when crossing clock domains. It also exhibits more flexible performance with voltage and temperature variations [BF02, BCV+05, FF04, BS05, DVFG05].

Data synchronization and communication across clock domains is a major challenge in designing GALS system. The difficulty lies in providing interfaces with robustness and high performance for crossing domain operations. Circuits that perform this function are called *synchronizers*. Synchronizers are expected to achieve high performance while maintain correct data transfer. Various synchronization circuits have been proposed to meet these challenges. A detailed description of some of the most common ones is presented in next chapter.

1.1.1 Classification of clock relationships

The mutual relationships of a pair of clock domains are classified in Table 1.1 according to their frequency and phase differences of the two domains. Each of the clock relationship is given detailed theoretical analysis in [Mes90]. *Mesochronous* domains share the same frequency and their phase difference remains constant. Simple synchronizers described in [Men91]and [DP98] can be used in such domains by measuring the phase difference and delaying the input enough to ensure correct data transfer. *Plesiochronous* signals can be synchronized in a similar way, but the delay must be occasionally adjusted to overcome the slow drift in phase difference. Also because there exists small frequency difference, data will be likely be lost due to the faster transmitter, or duplicated due to the

Classification	Phase Shift	Freq. Shift	Synchronization
Synchronous	0	0	None
Mesochronous	constant	0	Phase Compensation
Plesiochronous	varies slowly	small	Adaptive Phase Compensation
Multi-synchronous	varies slowly	0	Adaptive Phase Compensation
Periodic	varies rapidly	large	Predictive Synchronizer
Asynchronous	unknown	unknown	Universal Synchronizers

Table 1.1: Clock Relationship Classification

faster receiver. Certain control flow is therefore required to avoid this deficiency. Adaptive phase compensation [KG98] has been presented to solve this problem. Multi-synchronous domains[SG03, KG98] have similar property to plesiochronous domains and adaptive phase compensation can also be used. In periodic domains, a clock predictor is needed to calculate where the next clock signal will occur and whether the signal should be delayed to guarantee correct operation. A predictive synchronizer in [FKG06] foresees and prevents contentions between input signal and receiver clock and is suitable for this case.

The most general case is where corresponding units are asynchronous. In this case there is no known frequency or phase relationship between them. This enforces the need for synchronization during each communication and general purpose synchronizer circuits, such as two-flop synchronizer family, are required. These universal synchronizers can be employed to all clock relationships mentioned above. However, performance overhead is possible to occur for certain specific clock domains because these synchronizers can not be optimized for these domains.

1.1.2 Metastability

In general, data transmission between two clock domains requires flip-flops to store data. The correct operation of a flip-flop depends on meeting its timing paramenters. In this context, of interest are the *setup time* and *hold time* for the input data. If these times are violated and data is changed as the clock transition occurs, the flip-flop's behaviour is unpredictable. In particular the flip-flop may enter a *metastable* state, which is neither a logic 1 or 0 but rather something in between. In theory, this metastable state can last infinitely long. In a real

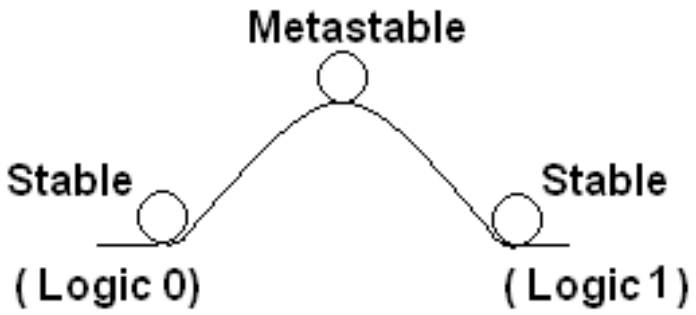


Figure 1.1: Metastability analogy of a ball on the hill[WH04]

circuit, the metastable state can be resolved by circuit noise which causes output to settle to logic one or logic zero. However, it may take unbounded time for the device to resolve to valid logic values. The device that enters a metastable state and the failure to produce a legal logic value is called *synchronization failure* [NM02]. A good analogy to metastable effect is shown in Figure 1.1. The top of the hill represents a metastable state. A flip-flop entering this state can stay on the top for indefinite time. However, noise in the circuit is capable of disturbing this balance and causing the ball to roll down to one of the stable states on either side of the hill.

Metastability is inevitable in a bistable device such as a flip-flop, where the feedback pulling the output into a defined state is exactly balanced. However, the physical construction of the device can influence the time constant, effectively making the top of the hill sharper and thus achieving metastability resolution more likely in a given time. In practical synchronizers, a delay of one or more clock cycles is reserved for metastability resolution. A detailed description on metastability and its effect in synchronization efficiencies are presented in next chapter.

1.1.3 Initial motivation and existing problems

In general, synchronizers are constructed to handle virtually all possible clock relationships. Designers using synchronizers often seek for best balance between robustness and high performance to match system requirements. Both the clock relationships and required throughput (or latency) can influence designer's choice of synchronizers. Probability of synchronization failure rate should also be reduced to a satisfactory level based on the design requirements. Because these factors directly affect the choice or design of a synchronizer and their properties are in fact difficult to predict by cursory observation, a systematic analysis applied to major synchronizer designs is needed. This was the initial motivation that drives this research. The starting point of this research was chosen to investigate the performance of a universal synchronizer, two-flop synchornizer. Primary research reveals an interesting property of this synchronizer: the best, worst and average performance (measured in cycle time) of a single data transfer can be predicted. Moreover, the average cycle time can be obtained by calculating all possible outcomes of cycle time with varying phase shift. Together with best and worst case scenarios, average performance can make an accurate and complete prediction of synchronizers.

1.2 Thesis Contributions

The two-flop synchronizer is already widely employed in industry [DP98] [Gin03]. Recent developments have produced several advanced synchronizers and their improvement in performance is shown when compared with standard synchronizers. However, few publications have fully explored the probabilistic behaviour of synchronizer performance. At system level, the two modules requiring synchronization almost always exhibit a certain clock relationship. For example, under mesochronous domains, a two-flop synchronizer incurs different data cycles(time required to complete a data transfer) for different phase shifts. The occurrence of difference in resulted data cycles behaves probabilistically. It is likely that other synchronizers also exhibit similar data cycle variations with different probability when operating under the same condition. Similar phenomena also appear in other classes of clock relationships. Therefore, when the clock relationship is known, optimized synchronizers can be selected based on analysis of their performance. More importantly, during burst data transfer, average performance

provides more realistic information due to the fact that it accounts for influences between two adjacent single data transfers. This thesis presents a full analysis and several analytical models for predicting average performance of several universal synchronizers. On the other hand, when the timing of input is unknown, the two domains are mutually asynchronous. Under these circumstances, only universal synchronizers can be employed in order to guarantee the safe data transfer. As stated before, these synchronizers may cause performance penalty. Hence, a detailed performance analysis and comparison on major universal synchronizers would benefit designers in their choice of synchronizers.

Also, the dependencies of data transfers across clock domains have not yet been fully explored. When a stream of data crosses the clock boundaries, each data has to be re-timed before it can be recognized. Intuitively, the synchronization of each data word should take the same time since they all go through the same procedures. However, this is not the case in reality. What happens is that the synchronization of the first data word will impact the synchronization of the next data word. This brings difficulties when evaluating the total time needed for the whole stream to cross domains. This thesis carries out in-depth analysis on the cycle time dependencies and study the behaviour of several major synchronizers in the burst-mode data transfer.

It is expected from the author that with the knowledge of characteristics of various synchronizers, designers could save time and efforts in evaluating synchronizer performances and synchronization failures. Also, it is expected that they would gain more thorough insights on the best and worst, more importantly, the average performance of several major synchronizers from the analysis.

The major contributions of this thesis are outlined in more detail, as follows:

A method of analyzing probabilistic behaviour of several major synchronizer performances has been proposed. Analytical expressions for predicting single-word-transfer synchronizer performance, and the average cases are derived for certain clock relationships.

The cycle time dependencies are studied in detail for these synchronizers. The synchronizers are firstly modelled and abstractions of cycle time information is obtained from the analyses of the model simulation.

Extension of the above analysis is made to predict synchronizer performances under burst-mode data transfer. Effect of each data transfer on the next one is fully investigated and analytical models are drawn to describe this behaviour

for certain clock relationships. The resultant influence on overall synchronizer performance is then evaluated. The Synchronizer Data Cycle Analyzer (SyDCA) was developed based on these results.

A new synchronizer, the three-flop synchronizer design is also proposed. Its originates from the analyses of the known synchronizer data cycle times in terms of their phase relationships. Preliminary investigation is carried out to analyze its performance and the reliability of this synchronizer is also analyzed.

1.3 Thesis Overview

A detailed description on existing major synchronizers and metastability is presented in Chapter 2, with discussions and examples on failure rate. Chapter 3 to 6 in this thesis focuses on analysis of performance on three major universal synchronizers: two-flop, fast two-flop four-phase, two-phase synchronizers. The new three-flop synchronizer is discussed in the second half of Chapter 5. Finally, Chapter 7 gives conclusions and possible directions for future research. The outline of Chapter 3 to 6 is as follows:

Analytical estimation of synchronizers are made in Chapter 3. Timing boundaries for each synchronizer operation is firstly developed. Analytical expressions are derived for some average cycle times.

Behavioural modelling of clock domains as well as the three synchronizers are presented in Chapter 4. The cycle time calculation of the two-flop synchronizer including the cycle time dependencies is presented.

The analysis is extended to the other two synchronizers in Chapter 5. The new three-flop synchornizer is proposed in this chapter and preliminary analyses of its data cycle time are also presented.

Chapter 6 summarizes the conclusions on the prediction of synchronizer data cycle times from the previous chapters. The SyDCA tool is developed based on these conclusions. Both single data cycle time and data cycle times under burst-mode data transfer are investigated with the help of SyDCA tool.

Chapter 2

Overview of Synchronization Circuits and Metastability

This chapter outlines the metastability in the flip-flop with a focus on the failure rate. Examples are presented to calculate the failure rate for a single flip-flop first and then the failure rate of many flip-flops are dealt with. Several major synchronization circuits from the literature study are presented in the second half of this chapter.

2.1 Metastability in the flip-flop

From Section 1.1.2, it was stated that the origin of producing a metastable output from a flip-flop was attributed to the violation of the flip-flop timing conditions. These timing conditions refer to the *setup time* and *hold time* for the input data. If the data input changes before the setup time, the output reflects the new value after a one clock cycle. If the data changes after the hold time, the output reflects the old value after one clock cycle. However, if the data changes after the setup time and before hold time, the output may be unpredictable and is possible to stay halfway between a valid logic low and high. This halfway state if known as a *metastable* state because it is not stable in the long term, it will eventually reach either high or low. However, the circuit has no drive towards either the high or low output values. Although the output will eventually settle to a valid logic value, the time taken is unbounded. This breaks the rules for a reliable digital system and can cause it to fail unpredictably. This is because the output after the metastable resolution is non-deterministic and it is possible that the output

settles to a different logic value from its original input. According to [Kin07], the probability of a failure in the synchronization process can be very low for an individual flip-flop, but because digital processors have to deal with many inputs per second, the probability of catastrophic failure over a long period of time is not low. Since it is impossible to completely prevent any synchronization failure as long as there is a need to synchronize data, it must be accepted that the synchronization failure rate can only be reduced by allowing more time for the synchronizer to settle.

2.1.1 MTBF

In [WH04] and [Kin07], small signal model for the cross-coupled inverters was developed to estimate the input conditions necessary to produce a metastable response of a given time, and the probability that the latch propagation delay exceeds this time. An important metric in assessing the reliability of a synchronizer flop is the Mean Time Between Failure [DB99]:

$$MTBF = \frac{e^{\frac{S}{\tau}}}{WF_cF_d} \tag{2.1}$$

where S is the settling time, τ is the settling time constant of the flip-flop, F_c is the sampling clock frequency, F_d is the frequency of changes in the input data and W is known as the metastability window and it is measured in seconds. The metastability window is related to the setup/hold window of a flip-flop [DB99]. If the input data to a synchronizer changes within the metastability window, then the synchronizer is assumed to have become metastable [Gin08].

In [Gin08], the derivation of Equation 2.1 was briefly conducted. This paragraph gives a more detailed description of it, and some of the intermediate results will be used in the next section. Assume that the input data can arrive at anytime to the synchronizer clock and the arrival time is uniformly distributed over the synchronizer's clock cycle. Given that the input data changes during a clock cycle, the probability of the flip-flop entering metastability is

$$p(enter\ ms) = \frac{W}{T} = W \cdot F_c, \tag{2.2}$$

where T is the sampling clock cycle time and $T = \frac{1}{F_c}$. If the input data does not change every clock cycle, but rather with frequency F_d, then the rate of entering

metastability is

$$rate(enter\ ms) = F_d \cdot p(enter\ ms) = W \cdot F_c \cdot F_d. \tag{2.3}$$

From the latch model, it can be inferred that given that a synchronizer is metastable at time 0, the conditional probability that it is still unstable at time t ($t > 0$) (namely the probability of failure) is

$$p[failure(t)|failure(0)] = e^{\frac{-S}{\tau}} \tag{2.4}$$

Assuming the probability of entering metastability $p(enter\ ms)$ and the probability of failure are independent, then the probability of failure is

$$p[failure(t)] = W \cdot F_c \cdot e^{\frac{-S}{\tau}}. \tag{2.5}$$

Given the rate of data change, the rate of failure is

$$rate[failure(t)] = p[failure(t)] \cdot F_d = W \cdot F_c \cdot F_d \cdot e^{\frac{-S}{\tau}}. \tag{2.6}$$

The $MTBF$ is the inverse of the failure rate and Equation 2.1 is obtained.

In reality, τ and W are typically 2 and 4 FO4 inverter delays [DB99] [Gin08]. Consider the 130nm process technology:a typical FO4 inverter delay is 33ps [Gin08]. The settling time constant τ is 66ps and the metastability window W is 132ps. If F_c is 200MHz and F_d is 20MHz and the settling time S is set to be one clock cycle, i.e.$S = \frac{1}{F_c}$, the $MTBF$ is calculated as shown below:

$$MTBF = \frac{e^{\frac{5n\ sec}{66p\ sec}}}{132p\ sec \cdot 200MHz \cdot 20MHz} \approx \frac{e^{76}}{5 \cdot 10^5}\ sec \approx 10^{20}\ years$$

The resultant $MTBF$ is longer than the age of the universe, which is less than 10^{10} years, so it is good enough to guarantee the reliable synchronization operation. Now if a more aggressive design is considered where the settling time S is halved, F_c = 1GHz, F_d = 100MHz and the rest parameters remain unchanged. The resultant $MTBF$ is:

$$MTBF = \frac{e^{\frac{2.5n\ sec}{66p\ sec}}}{132p\ sec \cdot 1GHz \cdot 100MHz} \approx \frac{e^{38}}{132 \cdot 10^5}\ sec \approx 68\ years.$$

These two simple examples illustrate how $MTBF$ changes with the shortened settling time and increased synchronizer clock frequency. In the first example, the $MTBF$ was solved under the condition that one clock cycle is allowed for metastability resolution. The clock frequency is 200MHz, and the settling time works out to be 75τ. In the second example where the clock frequency becomes 1GHz, the clock period is 1ns and is only about 15τ long. Note that in this example, the settling time S is 2.5ns, which is longer than one clock cycle. More than one clock cycles were reserved to solve metastability. However, significant decline in $MTBF$ value is still resulted in the second example, although the second $MTBF$ might still be acceptable in certain applications.

The effect of decreasing the settling time S can significantly increase the probability of synchronization failure in the synchronizing flip-flops, since the $MTBF$ decreases exponentially as the settling time reduces. If in the second example, only one clock cycle for the metastability settling time were allowed, the $MTBF$ would become less than 1 second! For the aggressive design example, if a better flip-flop with $\tau = 1$FO4 is used, the halved metastability settling time S can still provides satisfactory reliability.

2.1.2 MTBF for many synchronizers

The above analysis only deals with the $MTBF$ of a single synchronizer. If a SoC device has many synchronizers and the $MTBF$ requirement is given for all the synchronizers, how should each individual synchronizer be determined to meet the requirement? This section deals with this problem.

If a SoC device has H synchronizers, it is considered to fail if one of the H synchronizer fails. The probability of one synchronizer not entering metastable state is obtained from Equation 2.5 as,

$$p[\text{one synchronizer does not fail}] = 1 - W \cdot F_c \cdot e^{\frac{-S}{\tau}} \tag{2.7}$$

The probability that all H synchronizers do not fail is

$$p[H \text{ synchronizers do not fail}] = (1 - W \cdot F_c \cdot e^{\frac{-S}{\tau}})^H. \tag{2.8}$$

Applying the first order Taylor expansion to Equation 2.8, Equation 2.8 becomes,

$$p[H \text{ synchronizers do not fail}] \approx 1 - H \cdot W \cdot F_c \cdot e^{\frac{-S}{\tau}}. \tag{2.9}$$

The probability of at least one synchronizer fails is therefore,

$$p[\text{at least one synchronizer fails}] = 1 - p[H \text{ synchronizers do not fail}] \quad (2.10)$$
$$\approx H \cdot W \cdot F_c \cdot e^{\frac{-S}{\tau}}. \quad (2.11)$$

The rate of the these failures is obtained by multiplying the probability of at least one synchronizer fails by the frequency of changes in the input data, F_d, and the mean time between failure for all H synchronizers $MTBF(H)$ is its inverse:

$$rate[\text{at least one synchronizer fails}] \quad (2.12)$$
$$= F_d \cdot p[\text{at least one synchronizer fails}] \quad (2.13)$$
$$\approx H \cdot W \cdot F_c \cdot F_d \cdot e^{\frac{-S}{\tau}} \quad (2.14)$$

$$MTBF(H) = \frac{1}{rate[\text{at least one synchronizer fails}]}$$
$$\approx \frac{e^{\frac{S}{\tau}}}{H \cdot W \cdot F_c \cdot F_d}.$$
$$MTBF(H) = \frac{MTBF}{H}. \quad (2.15)$$

From Equation 2.15, it can be inferred that if the SoC $MTBF$ requirement is 100 years and there are 1000 synchronizers in the SoC, then the desired $MTBF$ for a single synchronizer is 10^5 years.

2.2 Synchronization circuits in digital systems

The basic function of any synchronization circuits is to correctly re-time the data passing from one clock domain (the transmitter) to another (the receiver). In a GALS system, the communication between two different clock domains calls for the need of synchronization circuits to reliably synchronize data that crosses clock domains. Many flavours of synchronization interfaces have been proposed. Synchronizer performance is usually evaluated in terms of *latency* and *throughput*. The latency is defined as the time from writing a data word into the output register of the sender to writing the same data into the first register of the receiver. The *data cycle* refers to the time between two consecutive writings of the first register of the receiver. Throughput is the reciprocal of the data cycle.

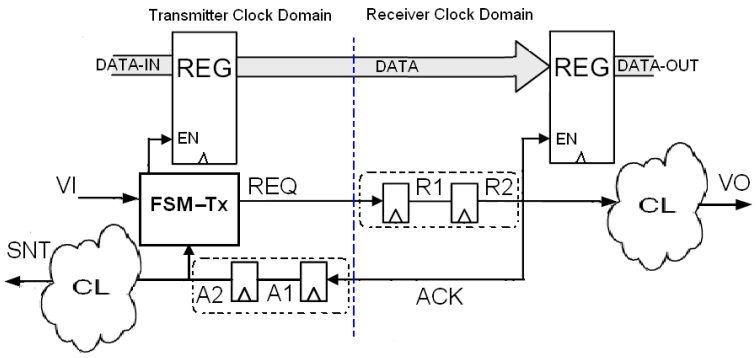


Figure 2.1: A two-flop synchronizer[Gin03]

This section outlines several synchronization circuits and briefly discusses their performances.

2.2.1 Family of the two-flop synchronizers

A very common and simple synchronization circuit is the two-flop synchronizer as is shown in Figure 2.1. This synchronizing interface applies simple control flow to ensure the correct transfer of data across clock domains. When the data is ready to be sent at transmitter side, it sends the request signal REQ to the receiver. Since the REQ signal originates from the sender it must be synchronized to the receiver clock before the request can be granted. In Figure 2.1, the first flop at the receiver side samples REQ on an active receiver clock edge, and one receiver clock cycle later, it is sampled by the second flip flop. The receiver then transfers the data to an internal register (not shown in Figure 2.1) when it is ready, and sends an acknowledgment signal ACK back to the transmitter that it is safe to send the next data. For the same reason, because the ACK signal originates from the receiver clock domain, it has to be re-timed to the transmitter clock domain before it can be recognized by the transmitter and another data can be sent.

For the two-flop synchronizer, on each side of the clock domain, one transmitter/receiver clock cycle is reserved for the metastability resolution. Since the reliability of the synchronizer depends on the time allocated to resolve metastability, the choice of flip-flop should meet the system $MTBF$ requirements. The $MTBF$ requirement for a particular system depends on the application. In the case where reliability is crucial and performance is less important, such as medical

equipment [WH04], synchronizers with long $MTBF$ can be chosen. For applications which are less critical, synchronizers with acceptable $MTBF$ but probably lower latency will satisfy the requirement. Also, for fast clock designs, such as high speed processors or high speed ASIC modules, where clock period is typically less than 50τ, a different approach based on multi-cycle resolution time or multi-synchronous clocking is required[DG07].

The two-flop synchronizer incurs some performance overhead, as can be seen from the analyses in the subsequent chapters. It is because only half of its data cycle contributes to the useful data transfer between two clock domains, and the other half is used purely for completing the handshake control protocol. Many attempts have been made to optimize the two-flop synchronizer to achieve higher throughput [Gin03] [DG09b]. Two design paradigms, the fast four-phase synchronizer [DG09a] and fast two-phase synchronizer [DG09a] have demonstrated shorter data cycle times (higher throughput) while maintaining the reliability of the two-flop synchronizer. The fast four-phase synchronizer deploys the asynchronous reset flip-flops to reduce the cycle that are not used for actual data transfer while the fast two-phase synchronizer fully utilizes all the cycle times by converting the four-phase handshake signal to two-phase handshake signal. Descriptions of operations and performance evaluations of these two synchronizers will be dealt with in the rest chapters of this thesis.

Because the fast four-phase and fast two-phase synchronizers are modified versions of the two-flop synchronizer, they share some common properties with the two-flop synchronizer. It is for that reason that the author categorize them as the two-flop synchronizer family. Comparing with other types of synchronization circuits, they have relatively simple structure and they support any clock relation.

2.2.2 Two-clock FIFO

A two-clock FIFO synchronizer has been proposed in [DP98], [CN01], [IM02], [CG03] and [CG02]. A general structure of a two-clock FIFO is shown in Figure 2.2. The FIFO is used to isolate the reading and writing processes [Kin07]. It therefore enables data transfers on every clock cycle if the FIFO is neither full or empty. Because of that the throughput can be improved comparing to the two-flop synchronizer, as it does not have to wait for the handshake control signal to finish before transferring the next data word, provided there is always some valid data in the FIFO. The implementation of the FIFO usually contains a set of

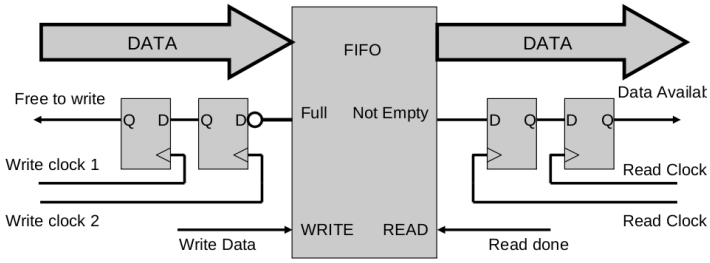


Figure 2.2: Two-clock FIFO [Kin07]

circulated registers and two address pointers, a read pointer and a write pointer. The write pointer should always points to a free location in the FIFO and it is updated immediately after a new data is written in. The read pointer always points to the most recent data that has been written and is also updated immediately after the read operation. To guarantee smooth write operation, sufficient space in the FIFO should satisfy certain conditions. It should be able to contain the stored data that has not yet been read, a free location for the current writing operation and at least one more free location to store data from the next writing without waiting for the read operation to release one free location. Similarly, to perform a smooth read operation, there must be sufficient data stored in the FIFO. Two flags, Empty and Full, are implemented to indicate conditions that may hamper the smooth read and write operations.The Full flag becomes true if there is not enough space to store any incoming data before the transmitter is stopped. The Empty is true if the unread items in the FIFO become insufficient before the receiver can be stopped. Therefore, necessary distance should be kept between these pointers to accommodate the delays required to stop the transmitter and receiver.

In [CG02], a self-timed single-stage FIFO was proposed for mesochronous clock domains and it was extended in [CG03] to operate in multi-synchronous, plesiochronous and asynchronous clock relations. The extended design requires

special treatment at both transmitter and receiver sides. In [CN01], a mixed-timing FIFO was presented and was designed to handle data transfers between arbitrary clock domain combinations. Mixed timing relay stations were also presented to provide a solution to deal with long interconnects, which were not supported by common two-clock FIFO design because they were intolerant to the delay variations over the long wires [DG09b]. In [Kin07], the performance of two-clock FIFO was compared with the two-flop synchronizer. It concluded that the throughput was improved because of the overlapping operations between synchronization of Full flag and the write operation, and synchronization of Empty flag and the read operation. The latency however did not improve because of the delays in synchronizing the two flags with their corresponding clock domains.

2.2.3 Synchronization circuits under specific clock relationships

The two-flop synchronizer family and the two-clock FIFO synchronizer described so far are able to handle all clock relationships [Gin03] [DG07]. However, they may incur performance overhead under a specific clock relationship, because they are not optimized for this particular clock relationships. On the other hand, if the clock relationship is known to the designer, it is possible to use the optimized synchronizers to achieve and higher performance with acceptable reliability. This section briefly discusses several of these synchronizers. In mesochronous domains, the two clocks driving the transmitter and receiver are derived from a common source, so the clock frequencies are the same on both sides but there is a constant phase shift between them. One way to deal with this interface is to use a simple FIFO [Gre93] [Gre95], as in Figure 2.2. Because the two domains have the same clock frequency, for every clock cycle, the transmitter writes one data item in the FIFO and the receiver reads one data item. If the FIFO is left half full when the data items start to transmit, given sufficient registers in the FIFO, the full or empty conditions will never meet. It therefore saves the effort for synchronization between Full and the transmitter clock, and Empty and the receiver clock.

In a plesiochronous system, the clock frequencies are nearly the same on both sides and the phase difference drifts slowly over time. It is necessary to apply flow control to the synchronization interface in this clock scheme, so that the possible

dropped or duplicated data items due to the phase drift can be avoided. According to [Kin07], the two clocks are generated from very stable crystal oscillators and are predictable in advance. In [DDX95], [KG98] and [SG03], adaptive phase compensation was applied to achieve synchronization by detecting the possible future clock conflicts and stopping data transfers when the two clocks could conflict. The adaptive phase compensation can be used as part of the flow control for data in a FIFO interface, and gives an average gain of half a cycle in latency over a single synchronizer, and considerably more over a multi-cycle synchronizer [Kin07].

2.2.4 Stoppable clocks

The synchronization interface discussed so far assume that the two clocks can not be changed, which leads the need for synchronizers to synchronize the incoming data from another clock domain. In a GALS system, it is possible to stop the clock [MM07] [YD96] [YD99]. Therefore, synchronization is not needed as the need for re-timing data is no longer necessary. A data-driven clock [KPWK02] activates only when the data arrives and it stops as soon as the data is processed. In this case, there is no competition between the clock activity and the new data and the metastability can be avoided [Kin07]. The stoppable clocks technique incorporates a local ring-oscillator clock generator and several MUTEXes[MC80] which can stop the local clock temporarily when new input data arrives. A stoppable clock technique designed for linear pipelines was presented in [SM00]. Stoppable clock can also integrate with FIFOs to achieve performance improvement [MTMR02].

2.3 Summary

The first half of this chapter described the cause of failure in a flip-flop due to metastability and the quantitative measure of its reliability, Mean Time Between Failure(MTBF). The expression of the MTBF was derived in detail. Two examples of MTBF evaluation were presented to illustrate the significant impact of choice of settling time and clock speed on the reliability of a single synchronizer. The MTBF for many synchronizer was then derived and was found to be H times shorter than MTBF for a single synchronizer, where H is the number of synchronizers in a system.

The second half of this chapter reviewed several major synchronization circuits in the literature. The two-flop synchronizer family is easy to implement but may incur performance overhead. The two-clock FIFO synchronizer improves the data throughput but results long latency. In some circumstances where the clock relationship is known, several optimized synchronizers can to achieve better performance. The stoppable clock interface avoids the conflict between clock signal and new data by pausing the clock temporarily to allow the new data to arrive.

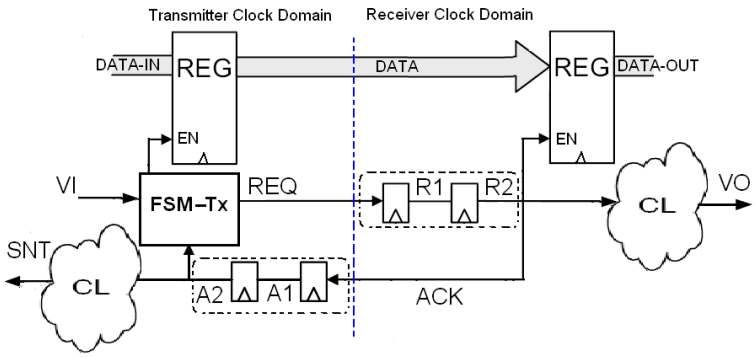
Chapter 3

Timing Boundaries and Average Data Cycles of the Two-Flop Synchronizer Family

This chapter describes a method of performance evaluation of synchronizers, including the two-flop synchronizer, fast four-phase synchronizer [DG09a] and fast two-phase synchronizer [DG09a]. Timing boundaries are derived for each of the said synchronizers. The average data cycles of these synchronizers are analyzed and compared.

3.1 Two-flop Synchronizer

Standard two-flop synchronizers are widely used in industry. The schematic diagram for a simple two-flop synchronizer was shown in Figure 2.1 in the last chapter and is reproduced as shown below. The principle behind this, and other synchronizers, is that the first flip-flop in either domain may have its operating conditions violated when sampling data coming from outside its domain. In the case of two-flop synchronizer shown in Figure 2.1, R1 may be left metastable. The second flip-flop imposes a one-clock delay which allows metastability to resolve before it samples the data. The following analysis of the standard two-flop synchronizer is based on the circuit proposed in [DG07]. In this work, it is assumed that the data register at the receiver side is always ready to receive data.



A two-flop synchronizer[Gin03]

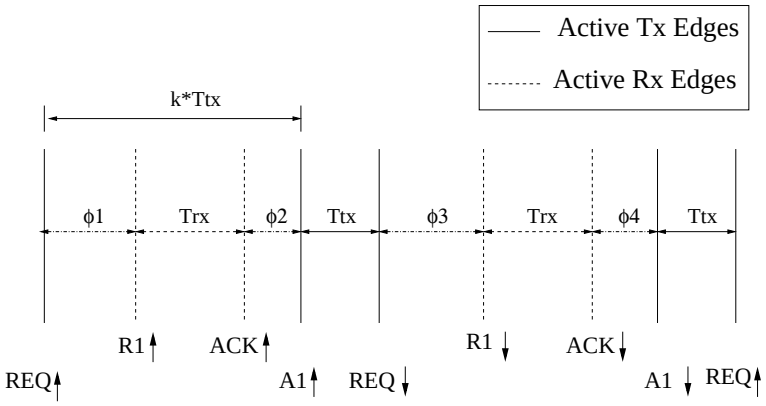


Figure 3.1: Timing relation of the two-flop synchronizer

3.1.1 Principles of operation

The forward cycle is defined as the time from assertion of request(REQ) signal to de-assertion of it, i.e. REQ+ → REQ-. The forward cycle is an integral number of transmitter clock cycles, although the multiplier may vary from cycle to cycle. The backward cycle refers to the time from de-assertion of REQ to the next assertion of it, i.e. REQ-→ REQ+, assuming the next data is ready to be sent. The complete data cycle consists of one forward cycle and one backward cycle.

Figure 3.1 shows the timing relation for signal transitions of the data cycle neglecting, for the moment, any metastability effects. When data is ready to be sent at the transmitter, it raises the REQ signal. On the next rising edge of receiver clock, REQ is sampled and R1 rises. The acknowledge signal (ACK) rises consequently one receiver clock later. On the next rising edge of transmitter clock, A1 goes high. It then takes another transmitter clock before A2 rises,

which brings REQ down. Following the same sequence, R1, ACK, A1 and A2 consequently fall to zero. The falling of A2 indicates a complete and acknowledged transfer of a single word.

3.1.2 Timing boundaries for the two-flop synchronizer

At the beginning of this chapter, the definition of data cycle is given as the time between two successive writings of the first register on receiver side. The measurement of the data cycle can be done by observing two successive assertions of REQ in the two-flop synchronizer. Because the forward and backward cycles are symmetrical for the two-flop synchronizer, only the forward cycle is analyzed here. The time boundary of the backward cycle is the same as that of the forward cycle.

Equation 3.1 can be obtained from Figure 3.1. This basically says that the time from raising to dropping REQ has to be an integral multiple of the transmitter clock period. Here, ϕ_1 represents phase shift between sender clock and receiver clock. The range of ϕ_1 is between 0 and T_{rx} and the range of ϕ_2 is between 0 and T_{tx}, as shown in Inequality 3.2 and 3.3. ϕ_1 and ϕ_2 must always be greater than zero as the propagation delay of the first flip-flop is finite. The range of the integral multiple k can be derived from Equation 3.1 and Inequality 3.4 as follows:

$$kT_{tx} = \phi_1 + T_{rx} + \phi_2 \tag{3.1}$$

$$\text{where } k \in Z \text{ and } k \geq 1$$

$$0 < \phi_1 \leq T_{rx} \tag{3.2}$$

$$0 < \phi_2 \leq T_{tx} \tag{3.3}$$

$$0 < \phi_1 + \phi_2 \leq T_{rx} + T_{tx} \tag{3.4}$$

Combining (3.1) and (3.4),

$$T_{rx} < kT_{tx} \leq 2T_{rx} + T_{tx} \tag{3.5}$$

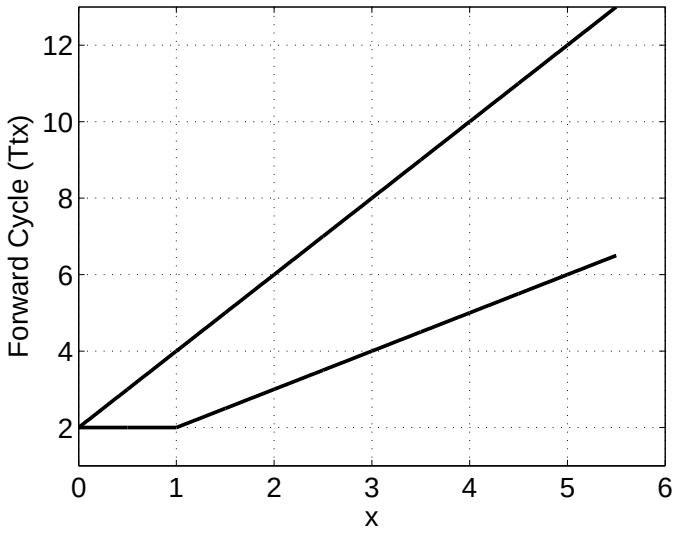


Figure 3.2: Timing boundaries for the forward cycle of two-flop synchronizer

Let $x = \frac{T_{rx}}{T_{tx}}$, then (3.5) becomes:

$$x < k \leq 2x + 1 \tag{3.6}$$

The forward cycle of a two-flop synchronizer is $(k + 1)T_{tx}$, as is shown in Figure 3.1. Inequality (3.7) describes the timing boundaries for the forward cycle:

$$x + 1 < k + 1 \leq 2x + 2 \tag{3.7}$$

The area semi-enclosed by the thick black lines in Figure 3.2 shows the forward cycle timing boundaries. The forward cycle is described in terms of an integral number of transmitter clock cycles. When the receiver and transmitter clock ratio x is less than 0.5, it always takes $2T_{tx}$ to complete the forward cycle. As x increases, the range of forward cycle becomes wider. This means that for a given clock ratio x, it is possible to have more than one forward cycle time. For example, when x is 3, the possible value for forward cycle spans from 4 to 8. The reason is that, although the clock ratio is fixed, the phase shift between the two clocks varies. In that sense, there are certain phase shifts where the shortest forward cycle time may occur, which is often referred to the best case performance. On the contrary, if the clock ratio remains fixed but phase difference is allowed to vary, the longest possible data cycle time may occur for this clock ratio, which

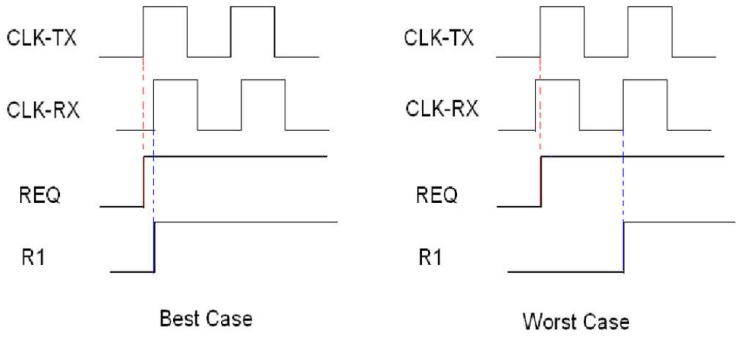


Figure 3.3: Best and worst cases for sampling REQ

is referred to the worst case scenario. The best and worst cases are exemplified in Figure 3.3. In the best case, the phase shift is so small that the receiver clock rises immediately after the rising edge of transmitter clock. The REQ signal is sampled on this rising edge of receiver clock and R1 is raised. The worst case is that the REQ signal just misses the rising edge of receiver clock and is delayed for an additional receiver clock cycle.

3.1.3 Average Data Cycle of the Two-Flop Synchronizer

Analysis of timing boundaries indicates that in the case when x goes beyond 0.5, different forward cycle values can result due to different starting phase ϕ_1 between transmitter and receiver clocks. Therefore, for a fixed clock ratio, it is not possible to tell the exact forward cycle time without the information of phase relation between the two clock domains. Although predicting the forward cycle time can be achieved by analyzing the phase relation under certain clock relation, as is presented in Chapter 4, the phase relation is generally difficult to obtain with unrelated clocks. During burst-mode data transfer, where a stream of data words are sent to the receiver, the synchronization of one word may affect the synchronization of the next one. The starting phase varies for synchronization of each word. These facts add significant amount of complexity in predicting the phase relation between two clock domains, and hence complicates calculation of the overall data cycle of the two-flop synchronizer, as can be seen from the subsequent chapters.

The average data cycle is, however, a good, yet simpler way of predicting cycle time. For a fixed clock ratio, the average data cycle is obtained from calculating

n	x	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	Average
0	$(0, 0.5]$	2								2
1	$(0.5, 1]$	$1/x-1$	$2-1/x$							$4-1/x$
2	$(1, 1.5]$		$2/x-1$	$2-2/x$						$5-2/x$
3	$(1.5, 2]$		$2/x-1$	$1/x$	$2-3/x$					$7-5/x$
4	$(2, 2.5]$			$3/x-1$	$1/x$	$2-4/x$				$8-7/x$
5	$(2.5, 3]$			$3/x-1$	$1/x$	$1/x$	$2-5/x$			$10-12/x$
6	$(3, 3.5]$				$4/x-1$	$1/x$	$1/x$	$2-6/x$		$11-15/x$
7	$(3.5, 4]$				$4/x-1$	$1/x$	$1/x$	$1/x$	$2-7/x$	$13-22/x$

Table 3.1: Probability and Average of Forward Cycle for the Two-Flop Synchronizer

each possible data cycle and its probability of occurrence, assuming all possible values of ϕ_1 are equiprobable. For example, when x is 3.5, from Figure 3.2, forward cycle varies from $5T_{tx}$ to $9T_{tx}$. If the starting phase ϕ_1 is allowed to sweep across its range with a proper time step, all the possible forward cycle times can be obtained. Table 3.1 lists the forward cycle times for clock ratio x in the range $(0, 4]$. At $x = 3.5$, probability of getting a forward cycle of $5T_{tx}$ is 1/7, and this probability is denoted as P_5. Similarly, P_6, P_7 and P_8 have the same probability of 2/7. For a different clock ratio, the same analysis can be applied. Table 3.1 summarizes the results of probability of forward cycle with the clock ratio ranging from 0 to 4. The interval of x is incremented in steps of 0.5, because from the timing boundaries plots in Figure 3.2, the upper limit for forward cycle is 2x+2, which reaches a new integer value at step of 0.5. Some interesting regularity appears in Table 3.1. For instance, within range (3, 3.5], non-zero probabilities are from P_5 to P_8. All the intermediate probabilities, namely P_6 and P_7, exhibit the same value, which is the reciprocal of x. Also, the probability values in both light and dark grey boxes show some regularity. With the knowledge of all possible values for forward cycle and their probabilities, the average data cycle can be easily calculated by taking the average of the all possible data cycles. The following analysis derives forward cycle probabilities for any clock ratio. The average forward cycle is then calculated based on these probabilities.

The following conclusions can be drawn from observations of the results in Table 3.1:

1. `Upper probability` P_{upper}. For each range of x listed in the table, P_{upper}

refers to the probability of the largest forward cycle time. The light grey boxes in the table show the P_{upper} for each range of x. A generic expression of the upper probability can be drawn as P_{n+2} and

$$P_{n+2} = 2 - \frac{n}{x} \tag{3.8}$$

where n is an integer and $n \geqslant 0$. The range of x corresponding to each P_{upper} can also be expressed in terms of n as $(\frac{n}{2}, \frac{n+1}{2}]$. Take the 4th row in the table as an example. Here, n is equal to 3. The range of x is therefore $(\frac{3}{2}, \frac{3+1}{2}]$, i.e.,$(0.5, 1]$. The upper probability P_{upper} in this case is $P_{3+2} = P_5$, and its expression is $2 - \frac{3}{x}$.

2. `Lower probability` P_{lower}. For each x range, there also exist the probability of the smallest forward cycle time, and this probability is denoted as P_{lower}. The dark grey boxes in the table show the distribution of these probabilities. The subscripts of the lower probabilities can be expressed as

$$P_{lower} = P_{\frac{n+3}{2}} \text{ (n is odd)}$$
$$P_{lower} = P_{\frac{n}{2}+2} \text{ (n is even).}$$

The contents in the dark grey boxes can also be generalized in terms of n and x as shown in Equations 3.9 and 3.10.

$$P_{\frac{n+3}{2}} = \frac{n+1}{2x} - 1 \text{ (n is odd)} \tag{3.9}$$

$$P_{\frac{n}{2}+2} = \frac{n+2}{2x} - 1 \text{ (n is even)} \tag{3.10}$$

3. `Intermediate Probability` P_{interm}. The white boxes with a uniform value $1/x$ are attributed to the intermediate probability, P_{interm}. They are the probabilities of all other possible forward cycle times. Since the subscripts of both lower and upper probabilities have been expressed in terms of n, those for intermediate probability can be obtained as is shown below:

$$P_{\frac{n+3}{2}+1}, P_{\frac{n+3}{2}+2}, \ldots P_{\frac{n+3}{2}+\frac{n+1}{2}} \text{ (n is odd)}$$
$$P_{\frac{n}{2}+3}, P_{\frac{n}{2}+4}, \ldots P_{\frac{n}{2}+\frac{n}{2}} \text{ (n is even and } n \geq 2)$$

4. `Average forward cycle time.` The average forward cycle time $Aver$ can be calculated from all probabilities described previously, as is shown below: If n is odd,

$$\begin{aligned} Aver =& \frac{n+3}{2} P_{\frac{n+3}{2}} + (\frac{n+3}{2}+1) P_{\frac{n+3}{2}+1} + \cdots + (n+2) P_{n+2} \\ =& \frac{3}{2}n + \frac{5}{2} - (\frac{3}{8}n^2 + \frac{1}{2}n + \frac{1}{8})\frac{1}{x}. \end{aligned} \tag{3.11}$$

If n is even,

$$\begin{aligned} Aver =& (\frac{n}{2}+2) P_{\frac{n}{2}+2} + (\frac{n}{2}+3) P_{\frac{n}{2}+3} + \cdots + (n+2) P_{n+2} \\ =& \frac{3}{2}n + 2 - (\frac{3}{8}n^2 + \frac{n}{4})\frac{1}{x}. \end{aligned} \tag{3.12}$$

The rightmost column in Table 3.1 lists the average forward cycle times calculated from Equations 3.11 and 3.12.

The average forward cycle plotted against x in Figure 3.4 shows certain non-linearity when two clocks have similar frequencies. However, as the clock ratio becomes larger, the average tends to be linear. When the clocks are significantly dissimilar, the faster clock will cycle several times in one period of the slower clock, thus its phase becomes relatively unimportant. The slow clock period dominates the communication time, so the curve for average forward cycle becomes linear. However, when the two clocks have similar frequencies, the effect of phase difference between them becomes more obvious. As the forward cycle does not vary linearly with the phase difference, the non-linearity emerges.

Table 3.1 also explains the average forward cycle behaviour. As x increases, the number of the intermediate probabilities increases. When the ratio becomes large enough, these probabilities become dominant and are the main contributions to the average value. Because these probabilities all have the same value, the resultant average will tend to be linear. The above analysis only considers the forward cycle which is first half of the data cycle. Because the symmetrical behaviour of the the two-flop synchronizer, there is a similar time taken for the second half as the handshake signals fall. As this is not carrying data it can be optimised, resulting in a faster synchronizer as described below.

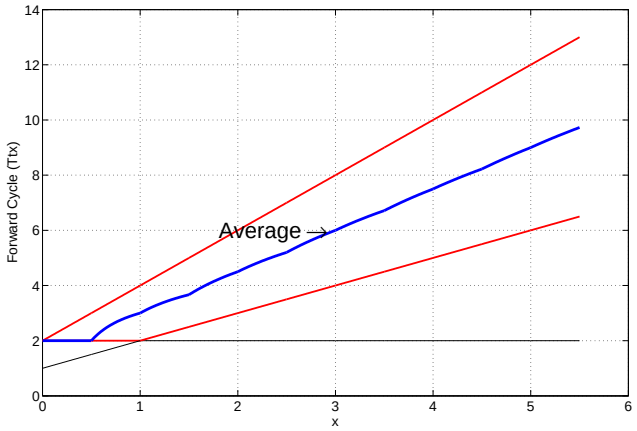


Figure 3.4: Average forward cycle of the two-flop synchronizer

3.2 Fast Four-Phase Synchronizer

The standard two-flop synchronizer enables reliable communication between two clock domains[Gin03]. However, it takes long data cycle for data transfer and hence results in long data cycle time. A fast four-phase synchronizer [DG09a] has been proposed to reduce data cycle by shortening the backward cycle. This section analyzes the timing boundaries and average data cycle of this synchronizer.

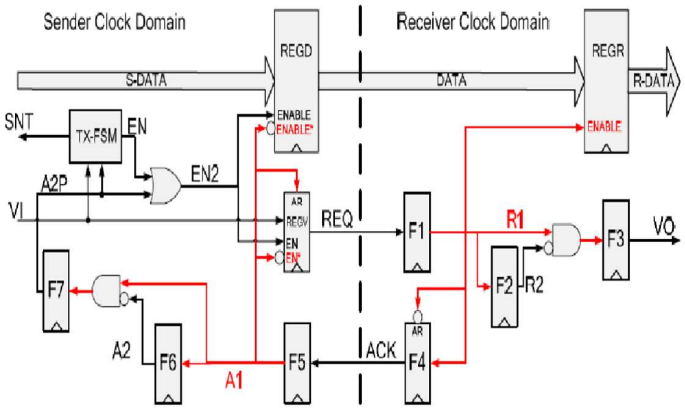


Figure 3.5: The fast four-phase synchronizer [DG09a]

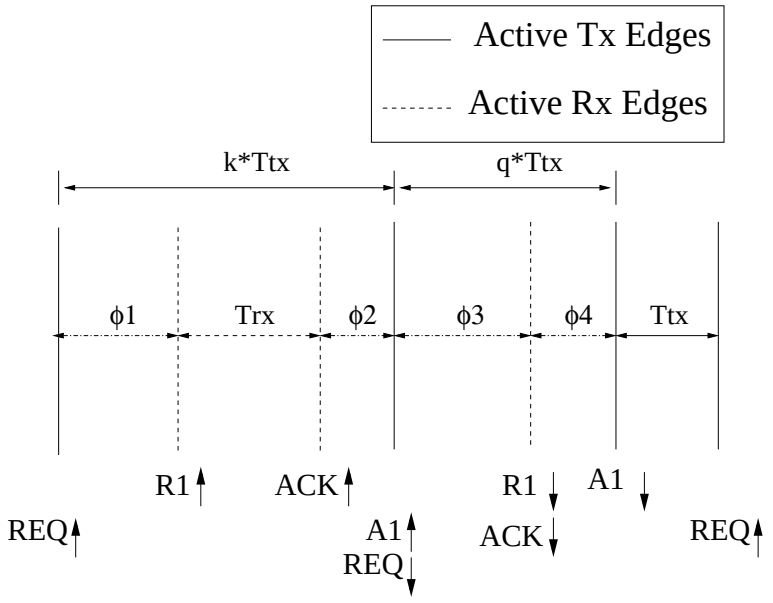


Figure 3.6: Timing relation of forward and backward cycles of the fast four-phase synchronizer

3.2.1 Operation principles

Figure 3.5 shows the circuit diagram of the fast four-phase synchronizer. Two flip-flops, namely REGV and F4, have asynchronous resets. Figure 3.6 shows the timing relation for signal transitions of a complete data cycle. Time intervals where signals cross domains are named as ϕ_1,ϕ_2,ϕ_3 and ϕ_4. Both forward and backward cycles are integral multiples of the transmitter clock cycle T_{tx} and they are denoted as k and q in Figure 3.6.

When data is ready to be sent at the transmitter, it raises REQ signal. On the next rising edge of receiver clock, REQ is sampled and R1 rises. The acknowledge signal (ACK) rises consequently on the next receiver clock. Then on the next transmitter clock, A2 goes high. Unlike the two-flop synchronizer where REQ falls, one transmitter clock later, the assertion of A2 asynchronously resets REQ immediately. Therefore, one transmitter clock is saved before the backward cycle starts. Similarly, the falling edge of R1 triggers an asynchronous de-assertion of ACK during the backward cycle. This saves one receiver clock.

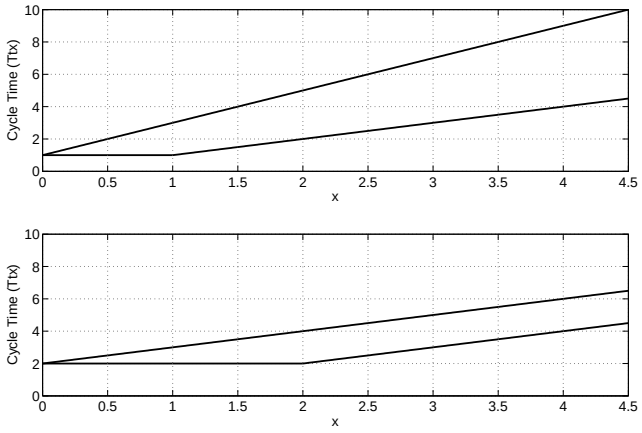


Figure 3.7: Timing boundaries for the forward and backward cycles, upper: forward cycle, lower: backward cycle.

3.2.2 Timing Boundaries for the Fast Four-Phase Synchronizer

Derivation of timing boundaries for forward cycle is similar to that of the two-flop synchronizer and is omitted here. Inequality 3.13 describes the timing boundaries for the fast four-phase synchronizer forward cycle:

$$x < k \leqslant 2x + 1. \tag{3.13}$$

Unlike the two-flop synchronizer where backward cycle and forward cycle are symmetric, the implementation of asynchronous reset in fast four-phase synchronizer breaks this symmetry. The asymmetry leads to separate analysis for backward cycle. From Figure 3.6, the time boundary for backward cycle is $(q+1)T_{tx}$. Analyses of the backward cycle time boundary were performed and the results are shown in (3.14) and (3.15). Figure 3.7 shows the timing boundaries for the forward cycle and backward cycles.

$$x < q + 1 < x + 2 \quad (x \geq 2) \tag{3.14}$$

$$2 \leq q + 1 \leq x + 2 \quad (0 < x < 2) \tag{3.15}$$

3.2.3 The average Data Cycle for the Fast Four-Phase Synchronizer

The calculation of the average data cycle for the fast four-phase synchronizer requires separate analysis for the forward and the backward cycles. The forward cycle is identical to that of the two-flop synchronizer. This can be seen from the comparison of the two timing relation diagrams in Figure 3.1 and Figure 3.6. The forward cycle of the fast four-phase synchronizer is one transmitter clock faster than that of the two-flop synchronizer, if the time taken by the asynchronous reset is neglected. Therefore, if the forward cycle of the two-flop synchronizer is $(k+1)T_{tx}$, then its counterpart for the fast four-phase synchronizer is kT_{tx}. The same analysis can be applied here to calculate the average forward cycle for fast four-phase synchronizer as shown below:

$$Aver_{n=odd} = \frac{3}{2}(n+1) - (\frac{3}{8}n^2 + \frac{1}{2}n + \frac{1}{8})\frac{1}{x} \tag{3.16}$$

$$Aver_{n=even} = \frac{3}{2}n + 1 - (\frac{3}{8}n^2 + \frac{n}{4})\frac{1}{x} \tag{3.17}$$

where $n \in Z^+$, $n \geqslant 0$ and $x \in [\frac{n}{2}, \frac{n+1}{2}]$.

The average forward cycle for the fast four-phase synchronizer is similar to that of the two-flop synchronizer except it is always one transmitter clock faster.

The backward cycle of the fast four-phase synchronizer is much shorter than that of the two-flop synchronizer. The timing boundary of the two-flop synchronizer becomes wider as x increases, whereas the width of boundary for the fast four-phase remains unchanged. This means the range of possible values for backward cycle increases significantly for the two-flop synchronizer, whereas it remains constant for the fast four-phase synchronizer as is shown in Figure 3.7. Unlike forward cycle where analytical expressions can be derived for probabilities and averages, the backward cycle does not show regular pattern. This is largely due to the dependency of the backward cycle on the forward cycle, as is the case in the two-flop synchronizer. The probabilities for forward cycle are obtained by monotonically increasing ϕ_1. However, this change in ϕ_1 causes non-linear behaviour in the starting phase of the backward cycle ϕ_3, which then causes the backward cycle time to fluctuate. The in-depth analysis on this dependency as well as the backward and data cycle times are presented in Chapter 4.

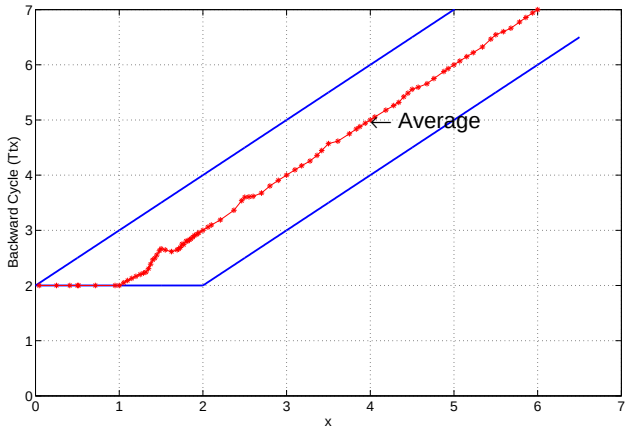


Figure 3.8: Average backward cycle of the fast four-phase synchronizer

Fortunately, the possible backward cycle times are limited to a small range and does not grow as the clock ratio increases, as can be seen from the lower figure in Figure 3.7. Some preliminary analyses were applied to the average backward cycle times and the resultant average backward cycle is in Figure 3.8. The small bubbles on the plot are the ratios selected in the analyses. Bumps occur on the curve where x is equal to 1.5, 2.5, 3.5 etc. This is due to the fact that forward cycle takes in a new value of k every interval of length 0.5, because the upper boundary has the slope of 2 (Figure 3.7). Since the backward cycle depends on the forward cycle, this advance in k can cause distortions in the average of backward cycle. However, these fluctuations are gradually smoothed out as the clock ratio increases. The average tends to be a straight line with a slope of 1. Together with the narrow timing boundaries indicated by the two parallel red lines, it is possible to predict the approximate backward cycle time. For example, if T_{rx}/T_{tx} is 3.2, a good approximation of backward cycle is $4T_{tx}$; if T_{rx}/T_{tx} is 3.8, then the backward cycle can be approximated as $5T_{tx}$.

3.3 Fast Two-Phase Synchronizer

The data cycle can be significantly improved by employing a two phase protocol over the communication channel. The fast two-phase synchronizer [DG09a] uses additional control logic to implement the two-phase protocol, where the backward cycle can be eliminated altogether. The circuit diagram is shown in Figure 3.9.

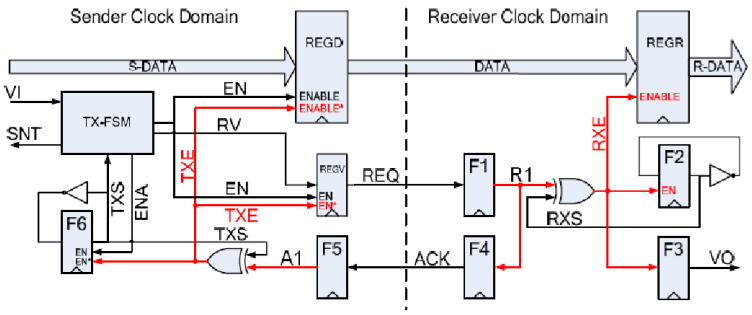


Figure 3.9: Fast two-phase synchronizer [DG09a]

On the receiver side, the XOR gate and the toggle flop F2 converts two-phase REQ into four-phase RXE and a single cycle pulse VO. The similar circuit at the transmitter side converts two-phase ACK into four-phase TXE.

The fast two-phase synchronizer completes its data transfer within one high phase (or low phase) of request (REQ) signal. The data cycle is equal to the forward or backward cycle. Intuitively, the data cycle should be only a half of that of the two-flop synchronizer. The forward cycle operation timing diagram is shown in Figure 3.10. The whole forward cycle takes $(k+1)T_{tx}$ to complete. This is the same as forward cycle for the two-flop synchronizer. Therefore, the same timing boundary can be applied as shown below:

$$x + 1 < k + 1 \leq 2x + 2 \tag{3.18}$$

where k is a positive integer. Observation shows that both the timing boundary and the average forward cycle are the same as those of the two-flop synchronizer.

3.4 Comparison of Three Synchronizers

In this section, the average data cycle (forward cycle plus backward cycle) for two-flop, fast two-phase and fast two-phase synchronizers are compared. Here it is assumed that for the two-flop synchronizer, the average backward cycle is the same as the average forward cycle. The influence of forward cycle on the backward cycle is not considered here and will be analyzed in the next chapter. Figure 3.11 shows a detailed comparison. All the curves for average data cycle tend to be linear as the T_{rx}/T_{tx} ratio increases. The slope of each line represents

the rate of increment of the data cycle. The slope for the two-flop, fast four-phase and fast two-phase synchronizers is 3, 2.5 and 1.5 respectively. This means that as the clock ratio increases, the fastest increase in the data cycle is the two-flop synchronizer, producing the worst performance among the three. The fast two-phase has the shortest data cycle time among them, and the rate of increase in the average data cycle is the lowest as well. All three synchronizers show distinct non-linear behaviour in the range $x \in (0.5, 3]$. It indicates that as clock frequencies get closer, the overall data cycle times show non-linear behaviour. As is said before, this non-linearity is a result of the dependencies of the backward cycle on the forward cycle. Therefore the effect of these dependencies on the data cycle deserves a more thorough analysis.

3.5 Summary

This chapter outlined the operations of three styles of two-flop synchronizers: the two-flop, the fast-four phase and the two-phase synchronizers. Timing boundaries are derived for the said three synchronizers. For the two-flop synchronizer, the timing boundaries for the forward and backward cycles are identical and the forward cycle timing boundary is derived. Separate analyses on the forward cycle and backward cycle timing boundaries are applied to the fast four-phase synchronizer due to the shortened backward cycle in the synchronizer design. The backward timing boundary is found to be much tighter than the forward one, which helps to increase the accuracy of the estimation of the average backward cycle

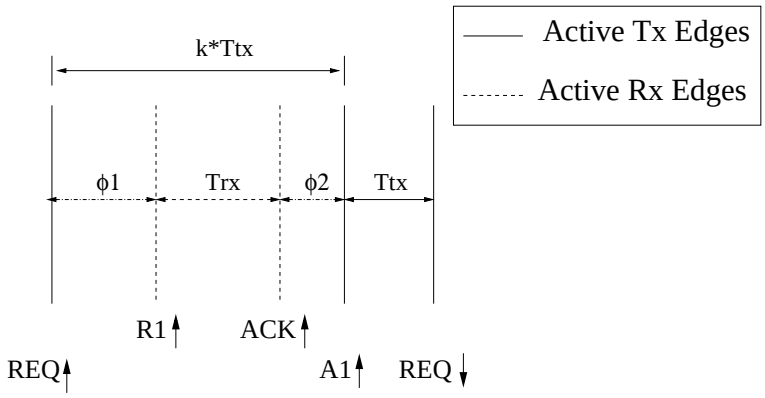


Figure 3.10: Forward cycle timing relation of the fast two-phase synchronizer

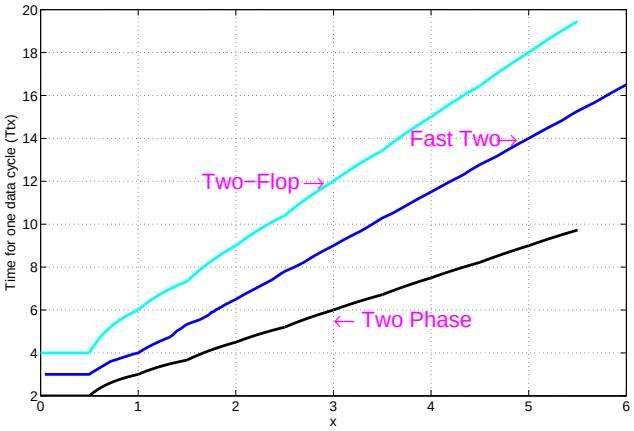


Figure 3.11: Comparison of Average Data Cycles of three synchronizers

time. The fast two-phase synchornizer has only forward cycle, whose boundary is the same as that of the two-flop synchronizer.

The analytical average forward cycle times were derived for the three synchronizers. They are obtained by taking the average of the weighted forward cycle times. Some preliminary exploration on the backward cycle time for the fast four-phase synchronizer is performed empirically. The dependency of the starting phase of the forward cycle on that of the backward cycle is found to cause non-linear behaviour of the backward cycle for both two-flop and fast four-flop synchronizers. This non-linearity also influences the average data cycle times, especially when the clock frequencies are close to each other.

Chapter 4

Behavioural Modelling and Cycle Times Prediction of The Two-Flop Synchronizer

In the previous chapter, the timing relation of the two-flop synchronizer was described for the *forward cycle*, defined as the time between the assertion and the next de-assertion of the REQ signal. The timing boundaries and the average forward cycle for the two-flop synchronizer were then analyzed. The *backward cycle*, defined as the time from the de-assertion to the next assertion of the REQ signal, and the complete data cycle, consisting of both forward and backward cycles, were not discussed because the dependency of backward cycle on forward cycle was unknown. This chapter presents a detailed analysis of this dependency as well as the backward and complete data cycle. Behavioural modelling of the two-flop synchronizer is introduced. The dependency of the starting phase of the backward cycle on that of the forward cycle is then analyzed. This information is used to obtain the backward cycle and the data cycle times.

4.1 Behavioural modelling of the two-flop synchronizer

4.1.1 An example of behavioural modelling of the two-flop synchronizer

The two-flop synchronizer is used for data communication between two different clock domains. In order to model the synchronizer behaviour, the two clock domains need to be modelled first. The timing relation of the data cycle is shown in Figure 3.1. To better illustrate the modelling algorithms for the two clock domains, an example is described first. It is assumed that no clock frequency drift exists in the model.

If the ratio of receiver clock to transmitter clock is 7/4, i.e. $T_{rx}/T_{tx} = 7/4$, then the length of one receiver clock cycle is defined as 7 units and that of the transmitter clock is 4 units. For the purpose of this analysis, a discrete time step, Δt, is introduced. This is set arbitrarily at a quarter of one unit. The size of Δt is chosen as a comprise resolution: any sampling may hide some fine detail, introduced by the effect of both clock edges becoming coincident (they rise at the same time) but this appears to expose all the major features as the relative phase changes. The effect of coincident clock edges is discussed in detail later. Since the receiver clock is seven-unit long, its clock cycle T_{rx} is $28\Delta t$. Similarly, the transmitter clock cycle T_{tx} is $16\Delta t$. Figure 4.1 shows signal transitions of the two-flop synchronizer for some sample cases with different starting phase ϕ_1. The red and blue solid lines represent the rising edges of the transmitter and receiver clock respectively. The transmitter clock is used as a fixed reference for all cases. So the phase difference results solely from shifting the receiver clock. It is also assumed that the request signal (REQ) rises at the first active edge of the transmitter clock. This figure illustrates the major phase changes for ϕ_3 as ϕ_1 is increased, with discontinuities at, for example $\phi_1 = 4\Delta t$. It also shows that the backward cycle time does not increase monotonically with the linear increase of ϕ_1, e.g. at the point where $\phi_1 = 9\Delta t$.

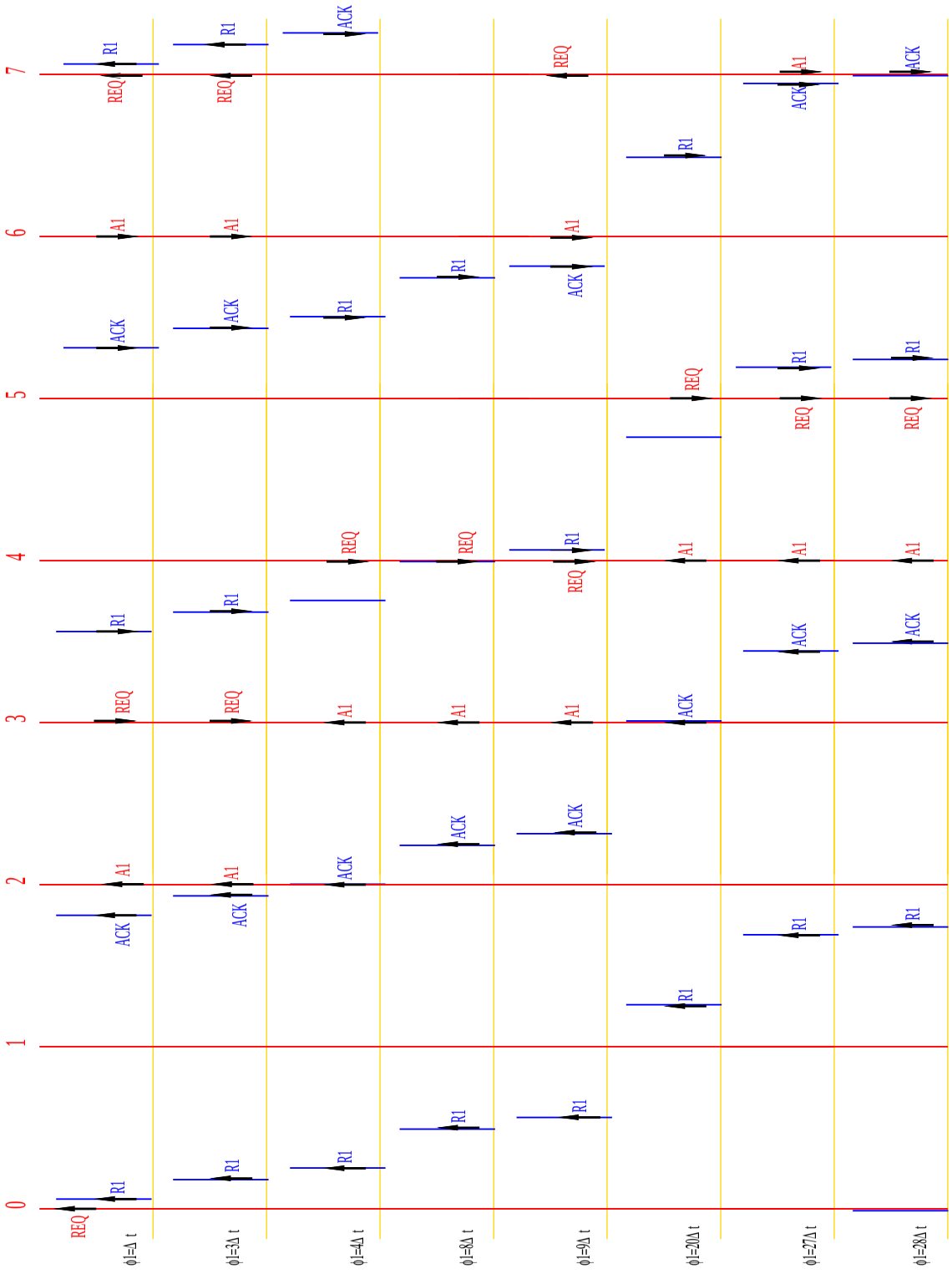


Figure 4.1: Two Clocks Timing Model

4.1.2 Count of Forward and Backward Cycles

4.1.2.1 Choice of sampling resolution

The previous section provided an example of modelling clock domains and two-flop synchronizer signal transitions. The sampling resolution was set to be a quarter of one unit and it was regarded to be sufficient to expose the major features of the synchronizer behaviour. This section provides justification for the choice of sampling resolution for any clock ratios.

Firstly, to demonstrate that the sampling resolution in the previous example was adequate to reveal all possible synchronizer features under the seven-to-four clock ratio, several other resolutions were applied to the model and the results are shown in Table 4.1. The first column lists forward and backward cycle times, expressed as a pair (fw, bw). For example, $(3, 4)$ means that time to complete the forward cycle and backward cycle are $3T_{tx}$ and $4T_{tx}$. Here, the number of the time step (Δt) within one unit is defined as the *resolution factor* and denoted as λ. So if the step time is a quarter of one unit, then λ is equal to 4. For each λ listed in the table, the number of occurrences of a particular (fw, bw) pair is obtained from counting the forward and backward cycle times at each ϕ_1 as it sweeps across its range in steps of Δt. Take the 4th column where λ is equal to 4 as an example. From the previous chapter, the range of phase ϕ_1 was found to be $(0, T_{rx}]$. In this example, T_{rx} is seven units long. So T_{rx} can be expressed as $7\lambda\Delta t = 28\Delta t$. The range of ϕ_1 becomes $(0, 28\Delta t]$. As ϕ_1 sweeps across its range in steps of Δt, the total number of the (fw, bw) pairs is 28. Among these pairs, three of them are $(3, 4)$. The probability of the occurrence of this pair is therefore denoted as 3/28 in the table. All other fractions in this table are obtained in the same way.

In Table 4.1, it can be seen that λ should be greater than or equal to 2. Otherwise, some (fw, bw) pairs will not be detected by the algorithms describing the model. On the other hand, as λ is increased, it produces no new information other than higher resolution around coincident clock edges. So the probability of each behaviour settles to an asymptotic value. In some cases, e.g. the $(5, 4)$ pair in the bottom row, the pair may vanish as λ becomes infinitely large. This is because, in theory, there is only one out of many cases when the clock edges are exactly coincident. As λ increases, the probability of getting into this case tends to be infinitesimally small. However, in a practical situation, this probability will be

(fw, bw)	$\lambda = 1$	$\lambda = 2$	$\lambda = 4$	$\lambda = 8$	$\lambda = 12$	$\lambda = 16$	λ
$(3, 4)$	–	$\frac{1}{14}$	$\frac{3}{28}$	$\frac{7}{56}$	$\frac{11}{84}$	$\frac{15}{112}$	$\frac{1}{7} - \frac{1}{7\lambda}$
$(4, 5)$	$\frac{2}{7}$	$\frac{3}{14}$	$\frac{5}{28}$	$\frac{9}{56}$	$\frac{13}{84}$	$\frac{17}{112}$	$\frac{1}{7} + \frac{1}{7\lambda}$
$(4, 3)$	–	$\frac{1}{14}$	$\frac{3}{28}$	$\frac{7}{56}$	$\frac{11}{84}$	$\frac{15}{112}$	$\frac{1}{7} - \frac{1}{7\lambda}$
$(4, 4)$	$\frac{2}{7}$	$\frac{4}{14}$	$\frac{8}{28}$	$\frac{16}{56}$	$\frac{24}{84}$	$\frac{32}{112}$	$\frac{2}{7}$
$(5, 5)$	$\frac{2}{7}$	$\frac{3}{14}$	$\frac{5}{28}$	$\frac{9}{56}$	$\frac{13}{84}$	$\frac{17}{112}$	$\frac{1}{7} + \frac{1}{7\lambda}$
$(5, 3)$	–	$\frac{1}{14}$	$\frac{3}{28}$	$\frac{7}{56}$	$\frac{11}{84}$	$\frac{15}{112}$	$\frac{1}{7} - \frac{1}{7\lambda}$
$(5, 4)$	$\frac{1}{7}$	$\frac{1}{14}$	$\frac{1}{28}$	$\frac{1}{56}$	$\frac{1}{84}$	$\frac{1}{112}$	$\frac{1}{7\lambda}$

Table 4.1: Comparison of forward and backward cycles of the two-flop synchronizer with different resolution factors

bigger as clocks that are close enough will act coincidentally. Therefore, the final model should take this into account. In sum, the underlying principle of sampling resolution selection is not to pursue the ultimate clock edge resolution but to expose all possible (fw, bw) pairs in order to evaluate the overall performance. Therefore, the choice made in the previous example is adequate to serve this purpose. The next question is that how the sampling resolution should be selected for an arbitrary clock ratio? It is clear from the previous example that λ has a certain range (greater than or equal to 2 in the previous example). Will this range change for any other clock ratios? If it does change, how does this affect the choice of sampling resolution? The choice of a quarter of one unit was justified empirically for the range $x \in (0.5, 3]$. Finer resolutions were experimented with to see if any further features were revealed with higher frequency sampling. None was found from the results. It is therefore believed that any sampling period less than half of one unit can reveal all the features, as a consequence of Nyquist-Shannon Sampling Theorem. Since the choice of sampling period as a quarter of one unit meets this requirement, this sampling resolution will be used for all clock ratios throughout the rest of this thesis.

4.1.2.2 Effects of coincident edges

The last column in Table 4.1 shows an interesting phenomenon: most of the fractions have a deviation of $\frac{1}{\lambda M}$ (M is 7 in this case) from a constant 1/7. Here, T_{rx} is expressed as the product of M, λ and Δt, so the range of ϕ_1 becomes

$(0, \lambda M \Delta t]$. Note that these results are obtained by increasing ϕ_1 from 0 to $\lambda M \Delta t$ in steps of Δt. The number of steps is therefore λM. This means that the deviation is caused by one out of $M\lambda$ steps for each (fw, bw) pair. In fact, the deviation is a result of coincident clock edges.

Two cases, namely $\phi_1 = 3\Delta t$ and $\phi_1 = 4\Delta t$, in Figure 4.1 exemplify the effect of coincident edges. The focus is put on signal transitions from the rise of ACK to the fall of REQ in Figure 4.1 for each case. The sequence of signal assertion and de-assertion is like this: ACK$\uparrow$ $\rightarrow$ A1$\uparrow$ $\rightarrow$ REQ$\downarrow$. The numbers on top of the figure represent active transmitter clock edges. When $\phi_1 = 3\Delta t$, ACK rises one Δt before the transmitter clock edge 2. As ϕ_1 changes to $4\Delta t$, the assertion of ACK occurs at edge 2. Unable to rise due to the sequence of signal assertions stated above, A1 has to wait until edge 3 before it rises. An extra delay of T_{tx} is created, causing the forward cycle to increase from $3T_{tx}$ to $4T_{tx}$. As ϕ_1 increases, the assertion of ACK is delayed accordingly.

The last column in Table 4.1 also reveals the fact that as λ increases, i.e. as the sampling resolution improves, all the fractions tend to constant values and the effect of coincident clock edges seems to fade. Theoretically speaking, the chance of getting into such a situation is infinitesimal and one may decide not to consider it at all. However, the author considers it important and indispensable to the behavioural modelling of the two-flop synchronizer, for the following reasons. Firstly, one direct consequence of clock edges becoming coincident is that it takes more clock cycles to complete the data transfer, which might lead to the worst case scenario. Take the ratio $M/N = 2$ as an example. The resultant (fw, bw) pairs are shown in Table 4.2. The two worst cases, $(6, 6)$ and $(5, 6)$, are both produced by the coincident edges when ϕ_1 is equal to $8\Delta t$ and $4\Delta t$ respectively. If these two scenarios were omitted, the worst case would be $(5, 4)$ according to the model. One basic principle of the behavioural modelling is that it should describe all the possible outcomes of the two-flop synchronizer under any clock ratios. When its overall performance is evaluated, the best and worst cases of data transfer should be reflected from the model itself. Although the probability of having coincident clock edges is generally low compared to the non-conflicting cases, it is still possible to occur in the real circuits and affect the overall performance.

Secondly, clock skew and practical flip-flops have larger timing window for coincident edges, as opposed to the infinitely narrow window described in the

$\phi_1(\Delta t)$	$fw, bw(T_{tx})$
1 to 3	$(4, 4)$
4	$(5, 6)$
5 to 7	$(5, 4)$
8	$(6, 6)$

Table 4.2: All possible (fw, bw) pairs for $M/N = 2$ with $\lambda = 4$

theoretical model. Some cases, especially those with very close clock edges, predicted as non-conflicting edges from the model, may become conflicting because of the larger timing window. Consequently, the possibility of getting the worst case performance may increase. Furthermore, the coincident clock edges can also produce metastable output for the first sampling flip-flops in each clock domain. For example, in Figure 4.1, when ϕ_1 is equal to $3\Delta t$, the assertion of ACK is close to that of A1. If the timing window is larger than the interval between these two signals, the flop sampling A1 is likely to become metastable. If this is the case, then A1 may not rise on edge 2 but later edges. Although the model described so far does not deal with metastability, it is still necessary to include the effect of coincident clock edges as it is possible to cause metastability in the sampling flops. With the indication of coincident clock edges from this model, one can further add the metastability effect to predict more precisely the overall performance.

Considering the stated factors, the author thinks it necessary that the behavioural model includes the effect of coincident clock edges, regardless of its probability of occurrence. Of course, it is not the aim of this model to deal with all the practical issues, but the author believes that it will be useful if this model can show all possible cases under certain ideal assumptions, especially the best and worst cases. Once these corner cases are identified, the designer can add practical factors into the model to get a more accurate performance evaluation. A behavioural model of the two-flop synchronizer under different clock ratios was constructed, based on the assumptions and mechanisms described so far. This model was used to simulate the phase changes for all clock ratios of interest, so that the relationship between the starting phases of the forward and backward cycles can be determined. It also gives simulation results for the forward, backward and data cycle times, which can be verified against the results from analyses.

$\phi_1(\Delta t)$	$\phi_3(\Delta t)$	$ForwardCycle$	$BackwardCycle$	$DataCycle$
1	3	3	4	7
3	1	3	4	7
4	16	4	5	9
8	12	4	5	9
9	1	4	3	7
20	16	5	5	10
27	9	5	3	8
28	8	5	3	8

Table 4.3: Comparison of starting phases, forward, backward and data cycles of the two-flop synchronizer with clock ratio of 7/4

4.2 Dependency of ϕ_3 on ϕ_1

The time between the successive drops of REQ and R1 represents the initial phase of the backward cycle, which is named as ϕ_3 in Figure 3.1. In Figure 4.1, several cases of signal transitions between the transmitter and the receiver are drawn based on different values of initial phase ϕ_1. The resultant phase ϕ_3, forward cycle, backward cycle and the complete data cycle are shown in Table 4.3. The cycle times are all multiples of T_{tx}. The monotonic increase in ϕ_1 produces a monotonic increase in forward cycle, as was analyzed in the previous chapter. However, the other quantities, including ϕ_3, the backward cycle and the data cycle, do not change monotonically as ϕ_1 increases. Because ϕ_3 is the starting phase of the backward cycle, its behaviour will influence the backward cycle and hence the data cycle. It is therefore necessary to analyze the behaviour of ϕ_3 before the characteristics of the backward cycle and the data cycle can be obtained.

The change in ϕ_3 with the linear increase of ϕ_1 is obtained by the model simulation. Recall that from the previous section, the following relationship was defined: $T_{rx}/T_{tx} = M/N = x$, where M and N were relatively prime and the precision of x was limited to 0.01 and remains the same throughout this thesis unless specified otherwise. The region of interest is $x \in (0.5, 3]$ where the influence of phase shift is relatively significant on the data cycle.

The relationship between ϕ_3 and ϕ_1 in the region $x \in (0.5, 1)$ is analyzed first. It is found that although there is no direct linear relationship between the two phases, ϕ_3 is a piecewise linear function of ϕ_1, M and N. Figure 4.2 gives several examples of plots of ϕ_3 versus ϕ_1. Empirical results of ϕ_3 reveal that the slope

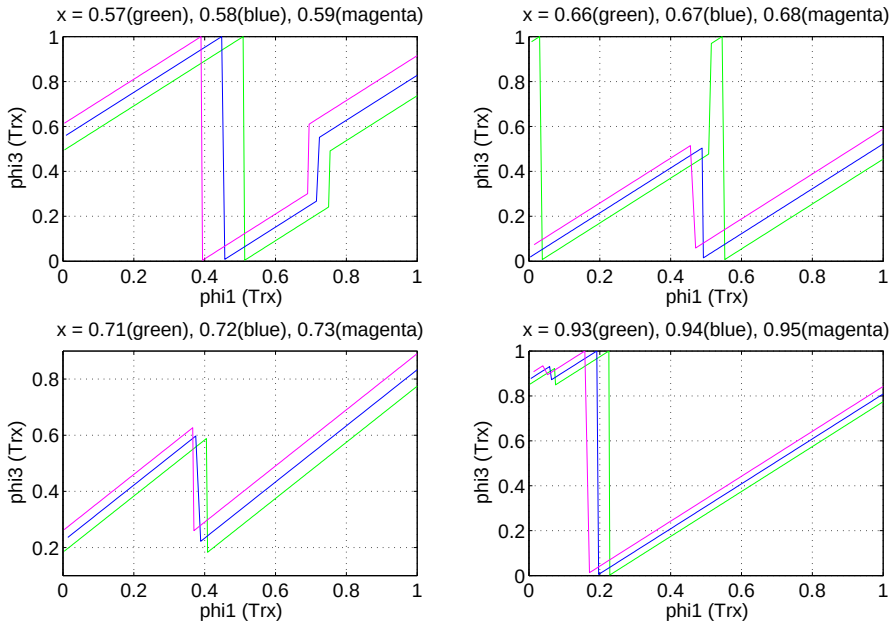


Figure 4.2: Example plots of ϕ_3 with different x values

of each line segment is 1. The relationship beween ϕ_3 and ϕ_1 can be written as $\phi_3 = \phi_1 + K$ where K represents the intersection of each line on the ϕ_3 axis. From Figure 4.2, it can be seen that K varies for different x values. The number of linear equations also changes when clock ratio changes. For example, when $x = 0.57$, three piecewise linear equations are required to express ϕ_3 whereas only two are needed when $x = 0.73$. Therefore, in order to describe the behaviour of ϕ_3, two things need to be decided: how many equations are needed and what is the value of K corresponding to each equation. A close investigation of the simulation results reveals that there exist three x values, 3/5, 2/3, 3/4, dividing the range $(0.5, 1)$ into four regions. Within each region, the number of line segments is fixed. The four sets of plots in Figure 4.2 are selections from each of these regions. The top right hand corner diagram illustrates the change in number of line segements when x crosses the 2/3 boundary. The green trace has four line segements at $x = 0.66$ but the blue trace has only two at $x = 0.67$. Because the number of line segments is equal to the number of equations needed to represent the relationship between ϕ_3 and ϕ_1, it can be inferred that four equations are needed when $x = 0.66$ but only two are needed when $x = 0.67$. Empirical results indicate that K can be expressed as multiples of λ. Therefore, the piecewise linear equation shown in the Equation 4.1 is created to describe ϕ_3.

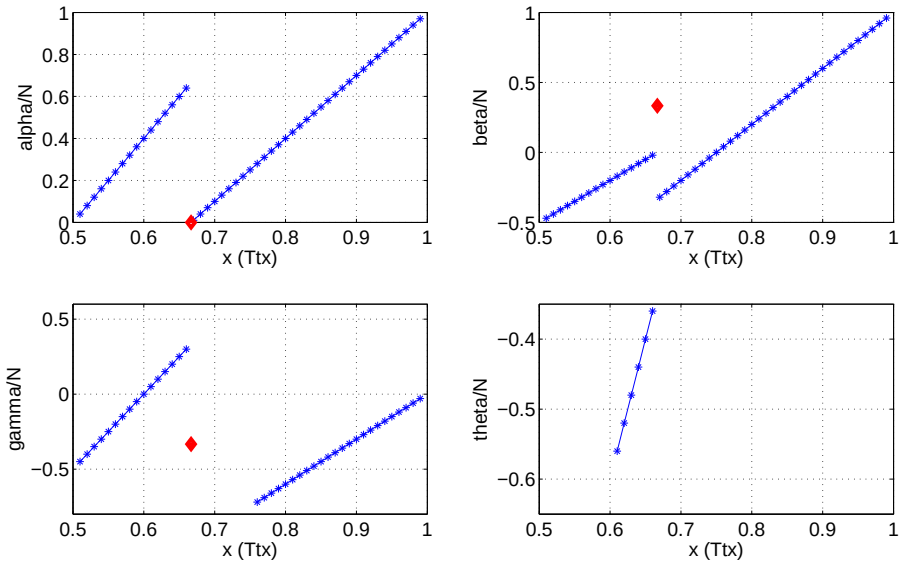


Figure 4.3: Plots of normalized interception parameters

$$\phi_3 = \begin{cases} \phi_1 + \alpha\lambda & \text{if } 0 < \phi_1 \leq A \\ \phi_1 + \beta\lambda & \text{if } A < \phi_1 \leq B \\ \phi_1 + \gamma\lambda & \text{if } B < \phi_1 \leq C \\ \phi_1 + \theta\lambda & \text{if } C < \phi_1 \leq M\lambda \end{cases} \tag{4.1}$$

The next thing to do is to determine the unknown parameters, α, β, γ, θ, A, B and C in Equation 4.1. These parameters are categorized into two groups according to their function in the equation set: interception parameters and boundary parameters. The interception parameters include α, β and γ and the boundary parameters include A, B and C. The interception parameters are analyzed first. As x represents the ratio of M to N, it is useful if these parameters are normalized in the same way as x, i.e, divided by N. Figure 4.3 shows the plots of the normalized interception parameters. The stars in each plot represent data from each simulation and the red diamonds represent the case when x is $2/3$. The discontinuity of each graph occurs at $x = 2/3$, with the exception of the normalized θ where it ceases to exist from beyond $2/3$. These anomalies at $2/3$ corresponds to significant changes in plots of ϕ_3 between $x = 0.66$ and $x = 0.67$, as is demonstrated in Figure 4.2. Below is an example of calculations for α, which

is a piecewise linear function of x. Four points, $(0.55, 0.2)$, $(0.63, 0.52)$, $(0.79, 0.37)$ and $(0.95, 0.85)$, on the plot of normalized α in Figure 4.3 are selected to calculate the slope. Let α_n represent the normalized α, so $\alpha_n = \alpha/N$.

$$\begin{aligned}
\frac{1}{2} < x < \frac{2}{3}, & \\
slope = \frac{0.52 - 0.2}{0.63 - 0.55} &= 4 \\
\alpha_n - 0.2 &= 4(x - 0.55) \\
\alpha_n &= 4x - 2 \\
\alpha &= 4M - 2N \\
\frac{2}{3} < x < 1, & \\
slope = \frac{0.85 - 0.37}{0.95 - 0.79} &= 3 \\
\alpha_n - 0.37 &= 3(x - 0.79) \\
\alpha_n &= 3x - 2 \\
\alpha &= 3M - 2N
\end{aligned}$$

The other three interception parameters can be obtained in the similar fashion. Table 4.4 lists expressions of these four parameters. The boundary parameters are also normalized in the same way as the interception parameters and are plotted in Figure 4.4. Unlike the interception parameters where their normalized forms show linear patterns, the plots for boundary parameters also display certain non-linearity. Investigation of the empirical data in the non-linear regions reveals that they can be easily represented by linear combinations of interception parameters. Table 4.5 presents expressions of these parameters. Table 4.6 lists these boundary parameters in terms of M, N and λ.

$$\phi_3 = \begin{cases} \phi_1 + \alpha\lambda & \text{if } 0 < \phi_1 \leq A \\ \phi_1 + \beta\lambda & \text{if } A < \phi_1 \leq B \\ \phi_1 + \gamma\lambda & \text{if } B < \phi_1 \leq C \\ \phi_1 + \theta\lambda & \text{if } C < \phi_1 \leq D \\ \phi_1 + \mu\lambda & \text{if } D < \phi_1 \leq M\lambda \end{cases} \tag{4.2}$$

So far, both interception and boundary parameters in Equation Set 4.1 have

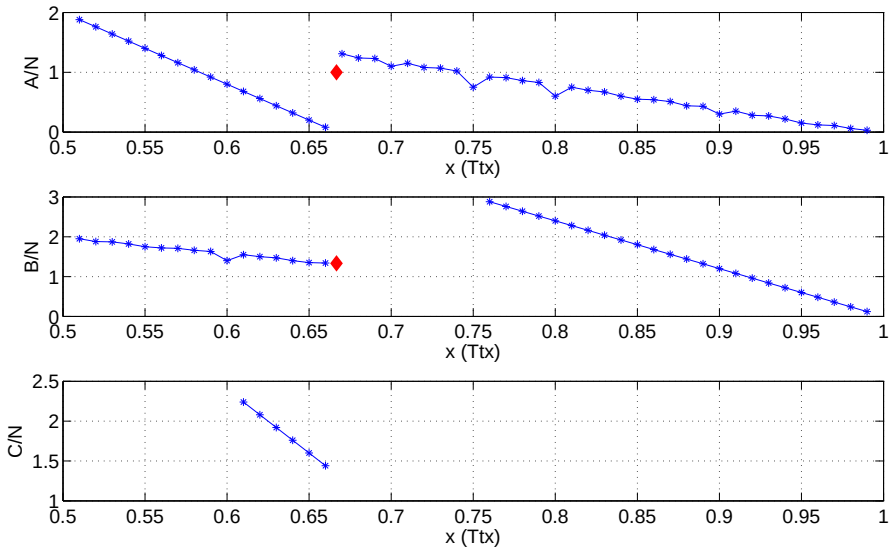


Figure 4.4: Plots of normalized boundary parameters

x	α	β	γ	θ
$(0.5, 0.6]$	$4M-2N$	$3M-2N$	$5M-3N$	–
$(0.6, 0.66]$	$4M-2N$	$3M-2N$	$5M-3N$	$4M-3N$
$(0.66, 0.75]$	$3M-2N$	$4M-3N$	–	–
$(0.75, 1)$	$3M-2N$	$4M-3N$	$3M-3N$	–

Table 4.4: Expressions of the interception parameters when $x \in (0.5, 1)$

been expressed in terms of M, N and λ. This indicates that given a clock ratio (M/N) and the resolution factor(λ), one can predict the starting phase of backward cycle, ϕ_3, based on Equation Set 4.1 and Tables 4.4 and 4.6.

The similar approach is adopted to obtain expressions of ϕ_3 in the range $x \in (1, 2)$ and $x \in [2, 3]$ separately. For the region $x \in (1, 2)$, five linear equations are needed and are shown in Equation Set 4.2. The corresponding parameters are expressed in terms of M,N and λ and the results are listed in Table 4.7 and Table 4.8.

In the case of $x \in (2, 3)$, Equation Set 4.1 is sufficient to depict $\phi 3$ and the corresponding parameters are shown in the first rows in Table 4.9 and Table 4.10. Lastly, several special cases are also worth mentioning. When the value of x becomes a whole number, the parameters for $\phi 3$ do not match any of the stated

x	A	B	C
$(0.5, 0.6]$	$-\beta\lambda$	$(\gamma - 2\beta)\lambda - 1$	$-$
$(0.6, 0.66]$	$-\beta\lambda$	$(\gamma - 2\beta)\lambda - 1$	$-\theta\lambda$
$(0.66, 0.75]$	$(\alpha - \beta)\lambda - 1$	$-$	$-$
$(0.75, 1)$	$(\alpha - \beta)\lambda - 1$	$-\gamma\lambda$	$-$

Table 4.5: Expressions of boundary parameters in terms of interception parameters when $x \in (0.5, 1)$

x	A	B	C
$(0.5, 0.6]$	$(2N - 3M)\lambda$	$(N - M)\lambda - 1$	$-$
$(0.6, 0.66]$	$(2N - 3M)\lambda$	$(N - M)\lambda - 1$	$(3N - 4M)\lambda$
$(0.66, 0.75]$	$(N - M)\lambda - 1$	$-$	$-$
$(0.75, 1)$	$(N - M)\lambda - 1$	$(3N - 3M)\lambda$	$-$

Table 4.6: Expressions of boundary parameters in terms of M, N and λ for $x \in (0.5, 1)$

x	α	β	γ	θ	μ
$(1, 1.33]$	$3M - 3N$	$2M - 3N$	$3M - 4N$	$-$	$-$
$(1.33, 1.5)$	$3M - 3N$	$2M - 3N$	$3M - 4N$	$2M - 4N$	$-$
$[1.5, 1.66]$	$2M - 3N$	$3M - 4N$	$2M - 4N$	$3M - 5N$	$-$
$(1.66, 2)$	$2M - 3N$	$3M - 4N$	$2M - 4N$	$3M - 5N$	$2M - 5N$

Table 4.7: Expressions of interception parameters for $x \in (1, 2)$

x	A	B	C	D
$(1, 1.33]$	$(3N - 2M)\lambda$	$(2N - M)\lambda - 1$	$-$	$-$
$(1.33, 1.5)$	$(3N - 2M)\lambda$	$(2N - M)\lambda - 1$	$(4N - 2M)\lambda$	$-$
$[1.5, 1.66]$	$(2N - M)\lambda - 1$	$(4N - 2M)\lambda$	$(3N - M)\lambda - 1$	$-$
$(1.66, 2)$	$(2N - M)\lambda - 1$	$(4N - 2M)\lambda$	$(3N - M)\lambda - 1$	$(5N - 2M)\lambda$

Table 4.8: Expressions of boundary parameters for $x \in (1, 2)$

scenarios and therefore need to be dealt with separately. In the region $x \in (0.5, 3]$, these cases are $x = 1$, $x = 2$ and $x = 3$. When $x = 1$, $\phi 3$ is always equal to $\phi 1$. In the other two cases, Equation Set 4.1 can be used and the expressions for the parameters are shown as the second and third rows in Table 4.9 and Table 4.10.

x	α	β	γ	θ
$(2, 3)$	$2M - 4N$	$2M - 5N$	$2M - 6N$	$2M - 7N$
2	0	1	-1	0
3	1	0	-1	-2

Table 4.9: Expressions of interception parameters for $x \in [2, 3]$

x	A	B	C
$(2, 3)$	$(3N - M)\lambda - 1$	$(4N - M)\lambda - 1$	$(5N - M)\lambda - 1$
2	$\lambda - 1$	λ	$2\lambda - 1$
3	$\lambda - 1$	$2\lambda - 1$	$3\lambda - 1$

Table 4.10: Expressions of boundary parameters for $x \in [2, 3]$

So far, expressions of the starting phase of the backward cycle, ϕ_3, within the region $x \in (0.5, 3]$ have been derived. This information will be used to obtain the backward cycle characteristics later.

4.3 Relationships between ϕ_1 and the cycle times

The goal of this modelling is to investigate the influence of the starting phase ϕ_1 on the behaviour of two-flop synchronizer cycle times, including the forward, backward and complete data cycles. Based on the analysis in the previous section, the starting phase of the backward cycle ϕ_3 can be determined by a set of linear equations, provided the information about ϕ_1, the clock ratio M/N and the resolution factor λ is known. Now the question is how can this information help to acquire forward, backward and complete data cycle times. This section presents a detailed analysis on this problem.

4.3.1 Prediction of the forward cycle time

In the previous chapter, the forward cycle times are counted for $x \in (0, 4]$ and listed in Table 3.1. For a given x range, it only shows the possible forward cycles and the probability of getting them. For instance, when $x \in (0.5, 1)$, the probability of getting a forward cycle of $2T_{tx}$ is $1/x - 1$ and the probability of getting a forward cycle of $3T_{tx}$ is $2 - 1/x$. However, it does not give any information on how forward cycle changes from $2T_{tx}$ to $3T_{tx}$. In other words, during the linear increase in ϕ_1, at what value of ϕ_1 does this change in the forward cycle occur?

The behavioural model for the two-flop synchronizer is used to explore the relationship between ϕ_1 and the forward cycle. The notations and assumptions remain unchanged. The procedures taken are outlined in Algorithm 4.1. As is stated before, the x region of interest is $(0.5, 3]$. The clock ratio x is assigned a value in this range, and the model is run by sweeping ϕ_1 from Δt to $M\Delta t$ in steps of Δt. For each step change in ϕ_1, forward cycle time is counted by the function $countFwCycle()$ and then compared to the previous forward cycle. If there is an increase in the forward cycle, then the ϕ_1 value is stored as a *critical value* ϕ_{1c}.

Algorithm 4.1 Find critical value for ϕ_1

```
given x
for φ1 = Δt to MΔt step Δt do
  i = φ1/Δt
  fw[i] = countFwCycle(φ1, x)
  if i ≥ 2 then
    increment = fw[i] − fw[i − 1]
    if increment == 1 then
      φ1c = φ1
    else
      continue
    end if
  end if
end for
```

This algorithm is used to find critical values for all the clock ratios in the range $(0.5, 3]$. Simulations using the said model were performed to establish the relationship between ϕ_1 and the forward cycle times. Figure 4.5 and 4.6 show plots of the forward cycles and the critical values for the range $x \in (0.5, 1)$ separately. In both figures, ϕ_1 is normalized by dividing by $(M\lambda)$, i.e. $\phi_{1,norm} = \phi_1/(M\lambda)$.

The range of ϕ_1 becomes $(0, 1]$. In Figure 4.5, as the $\phi_{1,norm}$ sweeps across its range, forward cycle jumps from $2T_{tx}$ to $3T_{tx}$ when the normalized ϕ_1 reaches its critical value. As the clock ratio x increases, the critical value gradually becomes smaller, leading forward cycle to jump at a smaller starting phase. This figure suggests that the starting phase imposes obvious influence on the forward cycle time. For example, to have a forward cycle time of $2T_{tx}$, x has to be less than 0.55 for a starting phase of $0.8T_{tx}$. However, to achieve the same forward cycle time, x can be as large as 0.83 with a starting phase of $0.2T_{tx}$. In other words, a small starting phase is likely to produce a short forward cycle. So if the starting phase is reduced, an increase in the overall synchronizer performance can be expected. Figure 4.6 provides a two-dimensional plot of Figure 4.5, viewed from the top. The red asterisks represent the normalized critical points from simulation. This plot clearly indicates that a function can be established between x and the normalized critical value $\phi_{1c,norm}$. This relationship is described in Equation 4.3 and it is plotted as the blue curve in Figure 4.6. Using Equation 4.3, the expression of ϕ_{1c} can be easily obtained as shown in Equation 4.4.

$$\phi_{1c,norm} = 1 - \frac{1}{x} \tag{4.3}$$

$$\phi_{1c} = (1 - \frac{1}{x})M\lambda \tag{4.4}$$

A similar approach is applied to obtain critical values of ϕ_1 for the regions of $x \in (1, 2)$ and $x \in (2, 3)$. From Table 3.1, it can be seen that when $x \in (1, 2)$, there are three possible forward cycles: $3T_{tx}$, $4T_{tx}$ and $5T_{tx}$. It therefore requires two sets of critical points, as is shown by $\phi_{1c,a}$ and $\phi_{1c,b}$ in Figure 4.7. The region below $\phi_{1c,a}$ has a forward cycle of $3T_{tx}$ and the one above $\phi_{1c,b}$ is $5T_{tx}$. The region between the two curves has a forward cycle of $4T_{tx}$. When x is in the range $(2, 3)$, the possible forward cycle times becomes four and Figure 4.8 shows the distribution of these forward cycle times as well as curves of critical points. As for the situation where x is an integer, the same analysis can be applied. The *Cycle-time Lookup (CL) table*, shown in Table 4.11, summarizes the critical values and the distribution of forward cycle times for the range $x \in (0.5, 3]$.

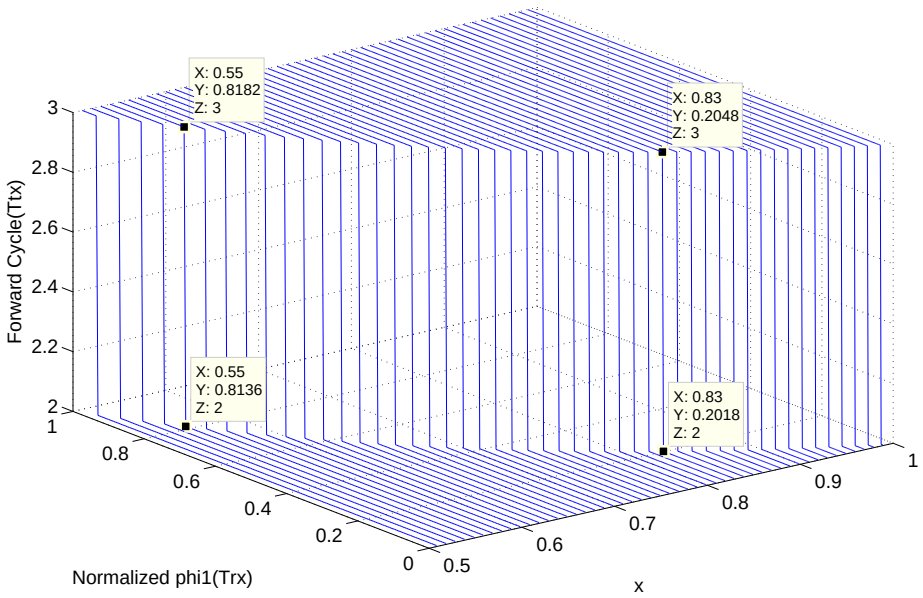


Figure 4.5: Plot of forward cycles in the range $x \in (0.5, 1)$

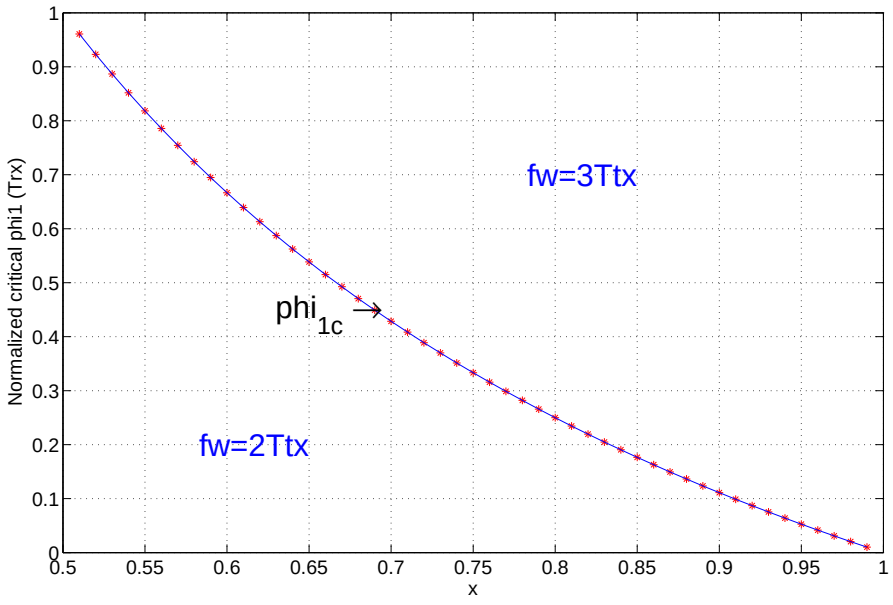


Figure 4.6: Plot of the normalized ϕ_{1c} when $x \in (0.5, 1)$

x	ϕ_{1c}	$ForwardCycles(T_{tx})$
$x \in (0.5, 1)$	$\phi_{1c} = (\frac{1}{x} - 1)M\lambda$	$\phi_1 \in (0, \phi_{1c})$ Forward Cycle=2 $\phi_1 \in [\phi_{1c}, M\lambda]$ Forward Cycle=3
$x \in [1, 1.5)$	$\phi_{1c} = (\frac{2}{x} - 1)M\lambda$	$\phi_1 \in (0, \phi_{1c})$ Forward Cycle=3 $\phi_1 \in [\phi_{1c}, M\lambda]$ Forward Cycle=4
$x \in [1.5, 2)$	$\phi_{1c,a} = (\frac{2}{x} - 1)M\lambda$ $\phi_{1c,b} = (\frac{3}{x} - 1)M\lambda$	$\phi_1 \in (0, \phi_{1c,a})$ Forward Cycle=3 $\phi_1 \in [\phi_{1c,a}, \phi_{1c,b})$ Forward Cycle=4 $\phi_1 \in [\phi_{1c,b}, M\lambda]$ Forward Cycle=5
$x \in [2, 2.5)$	$\phi_{1c,a} = (\frac{3}{x} - 1)M\lambda$ $\phi_{1c,b} = (\frac{4}{x} - 1)M\lambda$	$\phi_1 \in (0, \phi_{1c,a})$ Forward Cycle=4 $\phi_1 \in [\phi_{1c,a}, \phi_{1c,b})$ Forward Cycle=5 $\phi_1 \in [\phi_{1c,b}, M\lambda]$ Forward Cycle=6
$x \in [2.5, 3)$	$\phi_{1c,a} = (\frac{3}{x} - 1)M\lambda$ $\phi_{1c,b} = (\frac{4}{x} - 1)M\lambda$ $\phi_{1c,c} = (\frac{5}{x} - 1)M\lambda$	$\phi_1 \in (0, \phi_{1c,a})$ Forward Cycle=4 $\phi_1 \in [\phi_{1c,a}, \phi_{1c,b})$ Forward Cycle=5 $\phi_1 \in [\phi_{1c,b}, \phi_{1c,c})$ Forward Cycle=6 $\phi_1 \in [\phi_{1c,c}, M\lambda]$ Forward Cycle=7
$x = 3$	$\phi_{1c,a} = (\frac{4}{x} - 1)M\lambda$ $\phi_{1c,b} = (\frac{5}{x} - 1)M\lambda$ $\phi_{1c,c} = (\frac{6}{x} - 1)M\lambda$	$\phi_1 \in (0, \phi_{1c,a})$ Forward Cycle=5 $\phi_1 \in [\phi_{1c,a}, \phi_{1c,b})$ Forward Cycle=6 $\phi_1 \in [\phi_{1c,b}, \phi_{1c,c})$ Forward Cycle=7 $\phi_1 \in [\phi_{1c,c}, M\lambda]$ Forward Cycle=8

Table 4.11: Critical values and the distribution of forward cycle times for the two-flop synchronizer

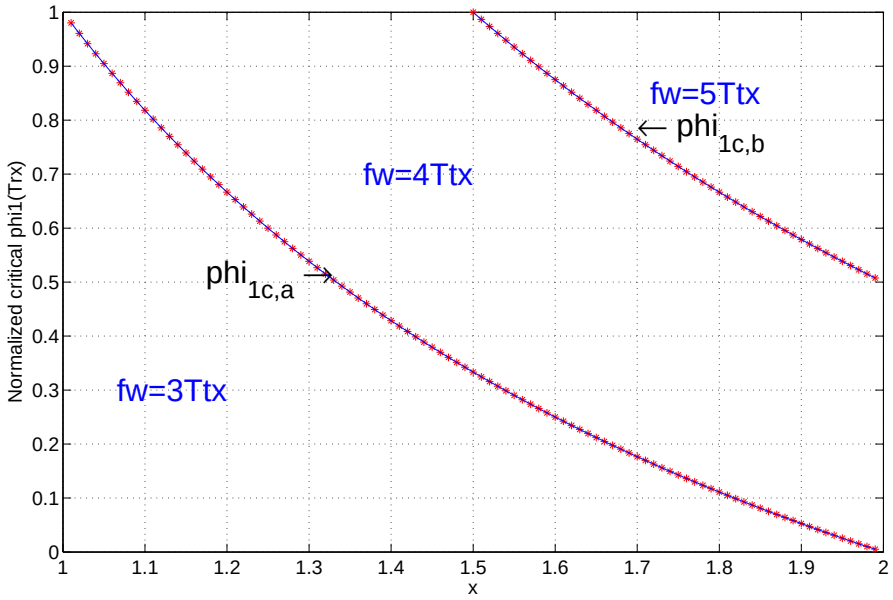


Figure 4.7: Plots of normalized ϕ_{1c} when $x \in (1, 2)$

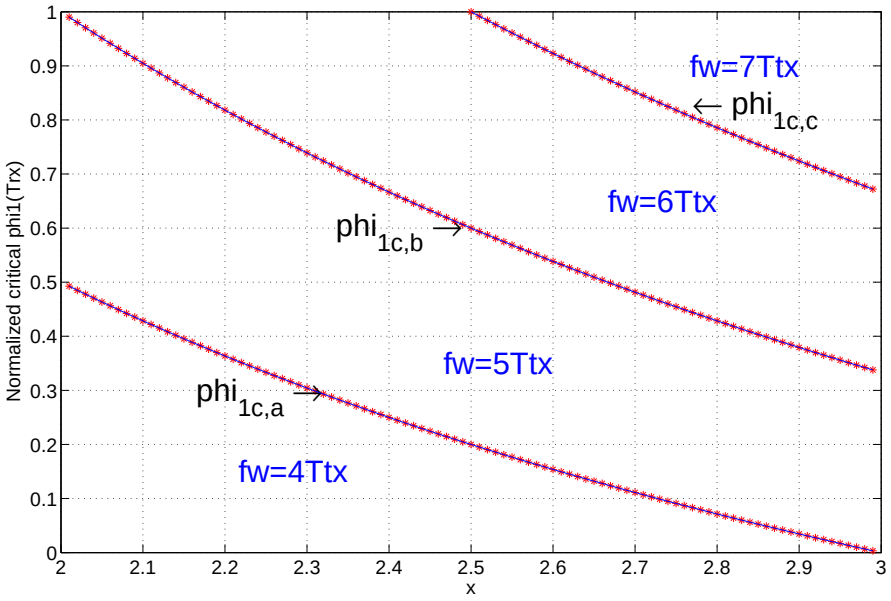


Figure 4.8: Plots of normalized ϕ_{1c} when $x \in (2, 3)$

4.3.2 The relationship between ϕ_3 and the backward cycle time

Now as the relationship between ϕ_1 and ϕ_3 is fully understood, together with the Cycle-time Lookup (CL) Table, the behaviour of the backward cycle time can be obtained. Because the operations of the forward and the backward cycles are symmetrical for the two-flop synchrnoizer, the backward cycle has the same characteristics as the forward cycle. This means that ϕ_3 influences the backward cycle in the same way as ϕ_1 influences the forward cycle. Therefore, the CL table shown in Table 4.11 can be used to obtain the backward cycle with a given ϕ_3. The interest of this analysis, however, focuses on finding how ϕ_1 influences the backward cycle time. The procedures to determine the backward cycle from ϕ_1 are outlined in Algorithm 4.2.

Algorithm 4.2 Find critical value for ϕ_1

given x
for $\phi_1 = \Delta t$ to $M\Delta t$ **step** Δt **do**
1.Calculate ϕ_3 from ϕ_1.
2.Look up the backward cycle in Table 4.11 that corresponds to ϕ_3.
end for

4.3.3 Analyses of the data cycle times

Both forward and backward cycle time are calculated given the clock ratio M/N, the resolution factor λ and the starting phase ϕ_1, the complete data cycle can be easily obtained by adding the two cycles. The analyses of the complete data cycle against the clock ratio x and ϕ_1 reveals some irregular patterns. As an example, the forward, backward and data cycles are plotted for $x \in (0.5, 1)$ in Figure 4.9. The figure of the forward cycle time shows that for a fixed ϕ_1, it increases as x increases. The forward cycle also increases with the increase in ϕ_1, given a fixed clock ratio. These facts indicate that the increase in either clock ratio or the starting phase can contribute to the increase of the forward cycle time. So a small clock ratio with a small starting phase can produce a short forward cycle. Things get complicated for the backward cycle time. The irregular trend in the backward cycle time with the increase in x is the result of the non-monotonic change in ϕ_3. If ϕ_3 were allow to increase in a linear fashion, it could be expected that the backward cycle time would increase in the same way as the forward cycle. Generally speaking, the backward cycle increases as x increases, which is reflected by the increase in the blue area in its plot in Figure 4.9. However, there are several exceptions. For example, the red area is bigger at $x = 0.67$ than $x = 0.59$. The reason can be found from the comparisons of ϕ_3 at these clock ratios in Figure 4.2. From this figure, it can be seen that as ϕ_1 increases from 0 to T_{rx}, the overall ϕ_3 is bigger at $x = 0.59$ than that at $x = 0.67$. As the bigger starting phase produces longer cycle time, the overall backward cycle time for 0.59 is longer. Although the increase in x also produces longer cycle time, which would produces a longer backward cycle at 0.67, it is overshadowed by the influences of the starting phase in this example.

For the data cycle time shown in the bottom of Figure 4.9, although the behaviour is also irregular, some patterns can still be identified. Generally, for a fixed clock ratio x, the data cycle increases as ϕ_1 gets bigger. In other words, reducing the starting phase can, to some extent, shorten the data cycle time. Another noticeable fact is that the complication of the data cycle time trend. This behaviour makes a pure analytical approach for the data cycle very difficult. The approach adopted in this chapter combines analytical solutions for ϕ_3 and a Cycle Lookup (CL) table for the prediction of the backward cycle time. In this way, the complicated analytical approach can be avoided and, at the same time, the data cycle time can be precisely obtained from the model. Therefore,

the author considers this approach appropriate for the modelling of the two-flop synchronizer.

4.4 Summary

This chapter presented detailed analyses of the forward, backward and data cycle times for the two-flop synchronizer. A behavioural model of the two-flop synchronizer was described and the adequate selection of the resolution factor was illustrated. With the help of this model, the dependency of ϕ_3 on ϕ_1 was analyzed. The relationship between them was then described by Equation Sets 4.1 and 4.2, as well as the parameters listed in Tables 4.4 to 4.10. The Cycle-time Lookup (CL) table shown as Table 4.11 was then constructed to make predictions of forward cycle time. This table, together with the relationship between ϕ_1 and ϕ_3 was used to calculate the backward cycle time and the data cycle time. The analysis of the data cycle time revealed a general pattern: by reducing the starting phase, the cycle time can be shortened. This observation may lead to improvements of the two-flop synchronizer later.

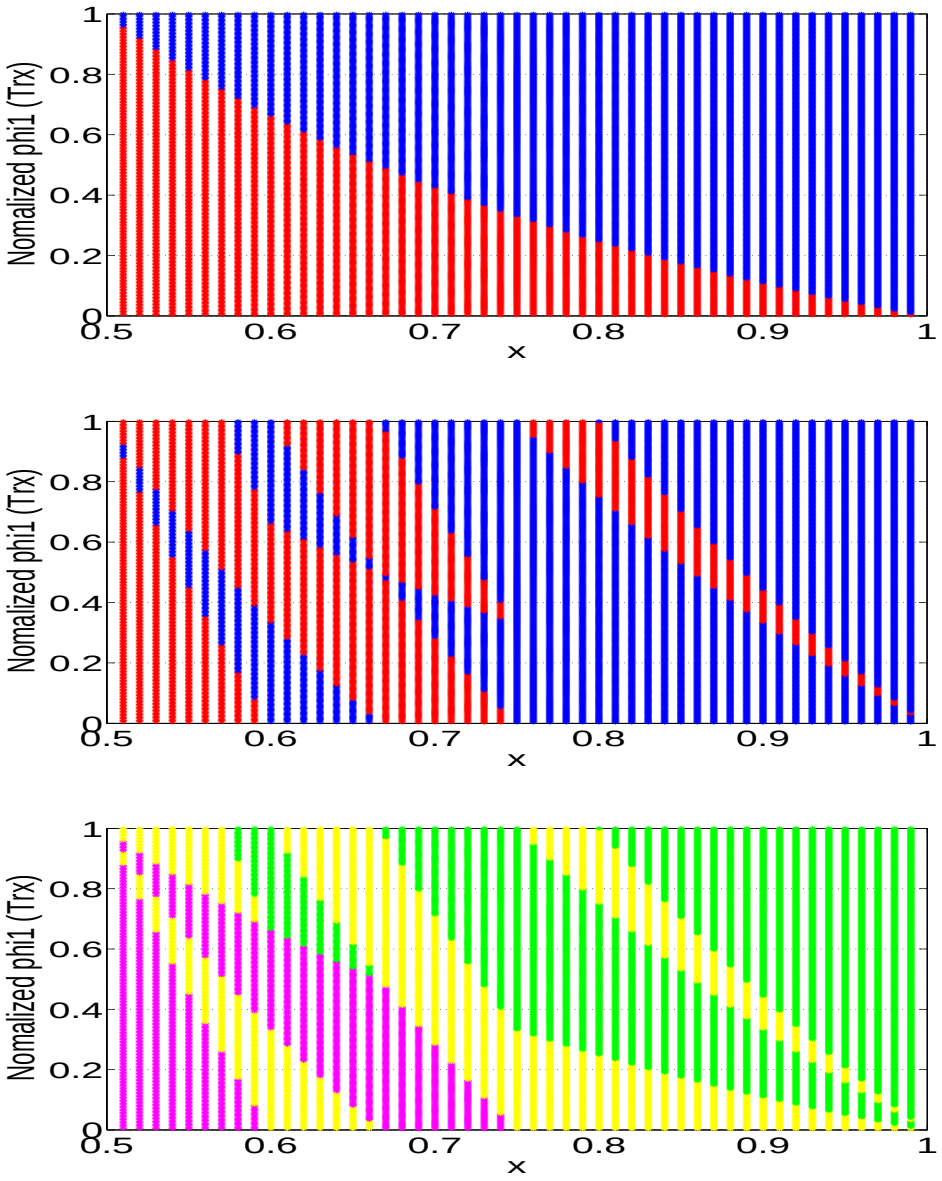


Figure 4.9: Plots of forward cycle(top), backward cycle(middle) and data cycle(bottom) when $x \in (0.5, 1)$. The color representations are: red=$2T_{tx}$, blue=$3T_{tx}$, magenta=$4T_{tx}$, yellow=$5T_{tx}$, green=$6T_{tx}$.

Chapter 5

Modelling of fast four-phase and fast two-phase synchronizers and preliminary investigation of a three-flop synchronizer

This chapter extends the analyses of the starting phases dependencies as well as the data cycle times to the fast four-phase synchronizer and the fast two-phase synchronizer. The region of transmitter-to-receiver clock ratio x is limited to $(0.5, 3]$ as the cycle times show distinct non-linearity in this region. The forward cycle times of these synchronizers are analyzed. A new approach in calculating the backward cycle time of the fast four-phase synchronizer is presented in detail. The analyses of the data cycle times of the two-flop, fast four-phase and two-phase synchronizers reveals potential synchronizer performance improvement, when the starting phase ϕ_1 is shortened. A new three-flop synchronizer with double clock frequencies is proposed. Preliminary investigation of the data cycle times as well as its failure rate analysis are presented at the end of this chapter.

5.1 Forward cycle times for the fast four phase and fast two phase synchronizers

Comparing the timing relation diagrams for the fast two-phase synchronizer in Figure 3.10 and the two flop synchronizer in Figure 3.1, the forward cycle times

turn out to be the same in both cases. It is therefore feasible to use the conclusions summarized in Table 4.11 to delineate the forward cycle behaviour of the fast two-phase synchronizer. Because the fast two-phase synchronizer utilizes both high and low phases of REQ signal to transmit data, the data cycle is actually the forward cycle. Also, for the two-flop synchronizer, the backward cycle is symmetrical to the forward cycle. So it can be inferred that the influence of the first data transfer of the fast two-phase synchronizer on the next data word transfer is equivalent to the influence of the forward cycle on the backward cycle for the two-flop synchronizer. The results on dependencies of ϕ_3 on ϕ_1 and the Cycle-time Lookup table for the two-flop synchronizer can be directly applied to the fast four-phase synchornizer.

Comparing the forward cycle timing relation of the fast four-phase synchronizer(Figure 3.6) and that of the two-flop synchronizer(Figure 3.1), one transmitter clock T_{tx} is saved for the fast four-phase synchronizer. However, the critical values for ϕ_1 are identical for these synchronizers. To prove this, the behavioural model for the fast four-phase synchronizer is constructed, under the same clock domains model presented in Section 4.1.1. Investigation of the critical ϕ_1 from the simulation results confirms its consistency between the two-flop and the fast four-phase synchronizer. The Cycle-time Lookup table for the fast four-phase synchronizer shown in Table 5.1 summarizes the critical values and forward cycle times for the fast four-phase synchronizer.

5.2 Backward cycle time for the fast four phase synchronizers

The backward cycle time behaviour for the fast four-phase synchronizer requires a separate analysis. As is stated before, because the flip-flops with asynchronous reset were used in the fast four-phase synchronizer design, the backward cycle is no longer symmetrical to the forward cycle. This fact changes the way of predicting the backward cycle times. In Figure 3.6, the backward cycle consists of ϕ_3, ϕ_4 and one transmitter clock. The goal is to establish the relationship between ϕ_1 and the backward cycle time. So the two phase relationships, namely ϕ_1 vs ϕ_3 and ϕ_1 vs ϕ_4 need to be analyzed before the backward cycle time can be predicted from ϕ_1.

x	ϕ_{1c}	$ForwardCycles(T_{tx})$
$x \in (0.5, 1)$	$\phi_{1c} = (\frac{1}{x} - 1)M\lambda$	$\phi_1 \in (0, \phi_{1c})$ Forward Cycle=1 $\phi_1 \in [\phi_{1c}, M\lambda]$ Forward Cycle=2
$x \in [1, 1.5)$	$\phi_{1c} = (\frac{2}{x} - 1)M\lambda$	$\phi_1 \in (0, \phi_{1c})$ Forward Cycle=2 $\phi_1 \in [\phi_{1c}, M\lambda]$ Forward Cycle=3
$x \in [1.5, 2)$	$\phi_{1c,a} = (\frac{2}{x} - 1)M\lambda$ $\phi_{1c,b} = (\frac{3}{x} - 1)M\lambda$	$\phi_1 \in (0, \phi_{1c,a})$ Forward Cycle=2 $\phi_1 \in [\phi_{1c,a}, \phi_{1c,b})$ Forward Cycle=3 $\phi_1 \in [\phi_{1c,b}, M\lambda]$ Forward Cycle=4
$x \in [2, 2.5)$	$\phi_{1c,a} = (\frac{3}{x} - 1)M\lambda$ $\phi_{1c,b} = (\frac{4}{x} - 1)M\lambda$	$\phi_1 \in (0, \phi_{1c,a})$ Forward Cycle=3 $\phi_1 \in [\phi_{1c,a}, \phi_{1c,b})$ Forward Cycle=4 $\phi_1 \in [\phi_{1c,b}, M\lambda]$ Forward Cycle=5
$x \in [2.5, 3)$	$\phi_{1c,a} = (\frac{3}{x} - 1)M\lambda$ $\phi_{1c,b} = (\frac{4}{x} - 1)M\lambda$ $\phi_{1c,c} = (\frac{5}{x} - 1)M\lambda$	$\phi_1 \in (0, \phi_{1c,a})$ Forward Cycle=3 $\phi_1 \in [\phi_{1c,a}, \phi_{1c,b})$ Forward Cycle=4 $\phi_1 \in [\phi_{1c,b}, \phi_{1c,c})$ Forward Cycle=5 $\phi_1 \in [\phi_{1c,c}, M\lambda]$ Forward Cycle=6
$x = 3$	$\phi_{1c,a} = (\frac{4}{x} - 1)M\lambda$ $\phi_{1c,b} = (\frac{5}{x} - 1)M\lambda$ $\phi_{1c,c} = (\frac{6}{x} - 1)M\lambda$	$\phi_1 \in (0, \phi_{1c,a})$ Forward Cycle=4 $\phi_1 \in [\phi_{1c,a}, \phi_{1c,b})$ Forward Cycle=5 $\phi_1 \in [\phi_{1c,b}, \phi_{1c,c})$ Forward Cycle=6 $\phi_1 \in [\phi_{1c,c}, M\lambda]$ Forward Cycle=7

Table 5.1: Critical values and the distribution of forward cycle times for the fast four-phase synchronizer

5.2.1 The relationship between ϕ_1 and ϕ_3

Comparing with the backward cycle of the two-flop synchronizer(Figure 3.1), the backward cycle for the fast four-phase synchronizer is one transmitter clock shorter. This fact introduces different ϕ_3 expressions for the fast four-phase synchronizer. Therefore, conclusions of ϕ_3 for the two-flop synchronizer can not be applied to obtain ϕ_3 for the fast four-phase. However, given the experience from the two-flop synchronizer, it is assumed that ϕ_3 is a piecewise linear function of ϕ_1: $\phi_3 = \phi_1 + K$ for the fast four-phase synchronizer as well.

The fast four-phase synchronizer model is simulated to obtain values for ϕ_3. Investigation of the relationship between ϕ_3 and ϕ_1 concurs with the assumption and only four linear equations are needed to describe ϕ_3 within the range $x \in (0.5, 3]$. So the expression 4.1 is sufficient to depict ϕ_3. The derivations of the parameters in this function is similar to that of the two-flop synchronizer and are omitted here. The expressions of these parameters are summarized in Table 5.2.

5.2.2 The relationship between ϕ_1 and ϕ_4

Experimental results of ϕ_4 were obtained from the simulation of the fast four-phase synchronizer model, with ϕ_1 sweeping across its range and x increasing from 0.51 to 3. The results are divided into four categories according to the region of clock ratio x: $x \in (0.5, 1)$,$x \in (1, 2)$, $x \in (2, 3)$ and $x = 1, 2, 3$. The behaviour of ϕ_4 within these categories is analyzed individually.

5.2.2.1 Uniform backward cycle times in region $x \in (0.5, 1)$

In Chapter 3, the backward cycle timing boundaries for the fast four-phase synchronizer was analyzed. From the plots of the timing boundaries in Figure 3.7, the backward cycle time remains constant at $2T_{tx}$ when $x \in (0.5, 1)$. This conclusion is also confirmed by the simulation results of ϕ_4 in this region. In this case, there is no need to derive expressions for ϕ_4 as the backward cycle time in this region is uniform regardless of the ϕ_4 values.

5.2.2.2 Expressions of ϕ_4 in $(1, 2)$

Within this region, the possible backward cycle time is either T_{tx} or $2T_{tx}$, as can be inferred from Inequality 3.15. However, it is not clear that how the backward cycle time changes between these two possible values. So it is necessary to explore the behaviour of ϕ_4 in this region. Here, in order to show the pattern of changes in ϕ_4 more conveniently, both ϕ_1 and ϕ_4 are normalized as $\phi_{1,norm}$ and $\phi_{4,norm}$:

$$\begin{aligned} \phi_{1,norm} &= \frac{\phi_1}{M\lambda} \\ \phi_{4,norm} &= \frac{\phi_4}{N\lambda}. \end{aligned} \tag{5.1}$$

x	α	β	γ	θ	A	B	C
$(0.5, 0.66]$	$2M-N$	$3M-2N$	–	–	$(N-M)\lambda-1$	–	–
$(0.66, 1)$	$2M-N$	$3M-2N$	$2M-2N$	–	$(N-M)\lambda-1$	$(2N-2M)\lambda-1$	–
1	0	–	–	–	–	–	–
$(1, 1.33]$	$2M-2N$	$2M-3N$	–	–	$(2N-M)\lambda-1$	–	–
$(1.33, 1.49]$	$2M-2N$	$2M-3N$	–	–	$(2N-M)\lambda-1$	–	–
$(1.49, 2)$	$2M-2N$	$2M-3N$	$2M-4N$	–	$(2N-M)\lambda-1$	$(3N-M)\lambda-1$	–
$[2, 2.49]$	$2M-3N$	$2M-4N$	$2M-5N$	–	$(3N-M)\lambda-1$	$(4N-M)\lambda-1$	–
$(2.49, 2.99]$	$2M-3N$	$2M-4N$	$2M-5N$	$2M-6N$	$(3N-M)\lambda-1$	$(4N-M)\lambda-1$	$(5N-M)\lambda-1$
3	2	1	0	-1	$\lambda-1$	$2\lambda-1$	$3\lambda-1$

Table 5.2: Parameters for ϕ_3 when $x \in (0.5, 3]$

Recall that the clock ratio x was defined before as $x = \frac{Trx}{T_{tx}} = \frac{M}{N}$. The choice of the divisor for ϕ_1 and ϕ_4 in Equation 5.1 is made based on the range of these two quantities, which are $(0, T_{rx}]$ and $(0, T_{tx}]$. Hence, the range for both $\phi_{1,norm}$ and $\phi_{4,norm}$ becomes $(0, 1]$. Observation of $\phi_{4,norm}$ shows that it can be expressed by a set of linear equations, as it is the case for ϕ_3, but with different slope and parameters. Figure 5.1 gives an example plot of $\phi_{4,norm}$(red) and $\phi_{3,norm}$(blue), where $\phi_{3,norm}$ is obtained in the same way as $\phi_{1,norm}$. One distinction is that the lines representing these two quantities has the opposite sign. Clearly, the lines for $\phi_{4,norm}$ have negative slope and are parallel to each other. Further investigation of all $\phi_{4,norm}$ reveals the relationship between $\phi_{4,norm}$ and $\phi_{1,norm}$ as shown in Equation 5.2. The iteration i is the number of linear equations needed and $K_{i,norm}$ is the normalized interception parameter K_i for the $i-$th equation. Equation 5.3 expands Equation 5.2 into three linear equations with the boundary parameters, A and B. The interception parameters in Equation 5.3 are $K_{1,norm}$, $K_{2,norm}$ and $K_{3,norm}$. The next step is to solve these parameters.

$$\phi_{4,norm} = -x\phi_{1,norm} + K_{i,norm}(i = 1, 2, 3) \tag{5.2}$$

$$\phi_{4,norm} = \begin{cases} -x\phi_{1,norm} + K_{1,norm} & \text{if } 0 < \phi_1 \leq A \\ -x\phi_{1,norm} + K_{2,norm} & \text{if } A < \phi_1 \leq B \\ -x\phi_{1,norm} + K_{3,norm} & \text{if } B < \phi_1 \leq M\lambda \end{cases} \tag{5.3}$$

The interception parameters are determined by calculating expressions for each straight line. Take Figure 5.1 as an example. The clock ratio x in this case is 1.57. Four points are chosen and the expressions of the lines are

$$\phi_{4,norm} = \begin{cases} -1.57\phi_{1,norm} + 0.86 & \text{if } 0 < \phi_1 \leq 0.546 \\ -1.57\phi_{1,norm} + 1.86 & \text{if } 0.546 < \phi_1 \leq 1. \end{cases}$$

The interception parameters $K_{1,norm}$ and $K_{2,norm}$ are 0.86 and 1.86 respectively. Please note in this case, the number of equations to represent $\phi_{4,norm}$ is 2. It is possible that in other cases, more are required. This method is used to find all the interception parameters in the region $x \in (1, 2)$. The maximum

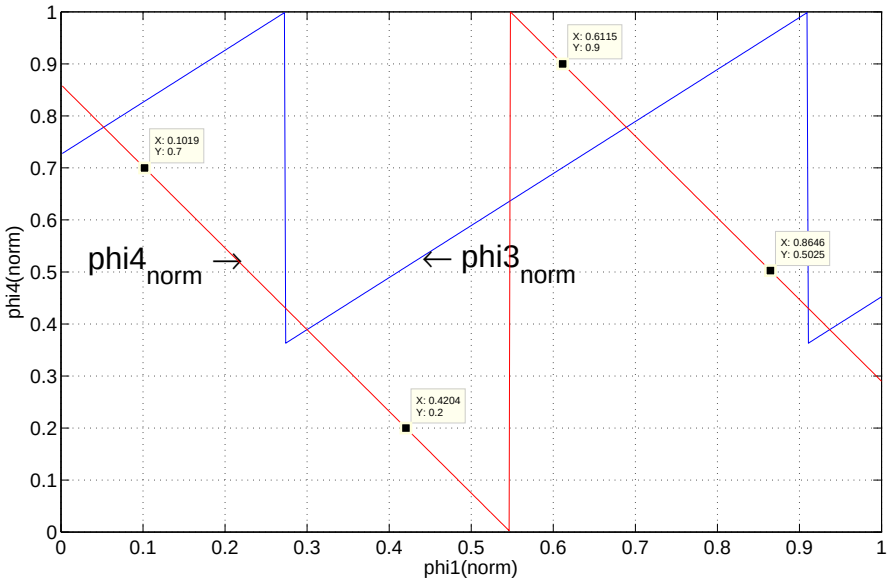


Figure 5.1: Example plots of $\phi_{4,norm}$ (red) and $\phi_{3,norm}$ (blue) at $x = 1.57$

x	$K_{1,norm}$	$K_{2,norm}$	$K_{3,norm}$
$(1, 1.33]$	$-2x+3$	$-2x+4$	–
$(1.33, 1.49]$	$-2x+3$	$-2x+4$	$-2x+5$
$(1.49, 1.66]$	$-2x+4$	$-2x+5$	–
$(1.66, 2)$	$-2x+4$	$-2x+5$	$-2x+6$

Table 5.3: Normalized interception parameters for $x \in (1, 2)$

number of interception parameters in the region is three. Once these parameters are calculated, their behaviour can be explored. It is expected that these parameters to be expressed in terms of known factors, which are M,N and λ. From the previous experience, it is likely that these parameters are related to the clock ratio $x = \frac{M}{N}$. Figure 5.2 are the plots of these parameters against x. Expressions for these parameters are summarized in Table 5.3.

Given the results for the normalized parameters in Table 5.3, Equation 5.3 can be re-written in terms of ϕ_4 and ϕ_1. To give an example, the $K_{1,norm}$ when x is in $(1, 1.33]$ is selected to illustrate this procedure. In this region, $K_{1,norm}$ is $-2x + 3$. From the definition of $\phi_{1,norm}$ and $\phi_{3,norm}$ in Equation 5.1, the first

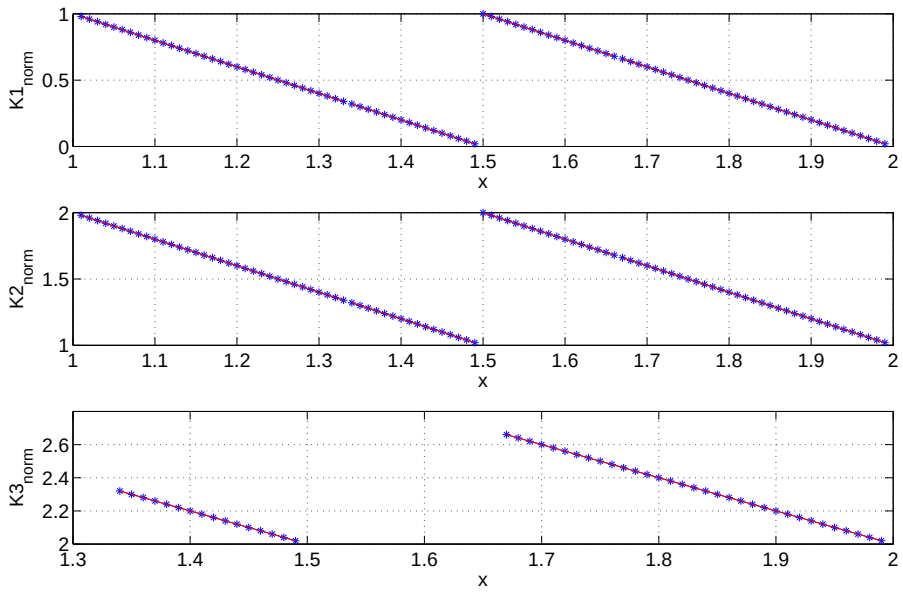


Figure 5.2: Interception parameters for $x \in (1, 2)$

equation in Equation 5.3 becomes

$$\begin{aligned} \frac{\phi_4}{N\lambda} &= -x\frac{\phi_1}{M\lambda} + (-2x + 3) \\ \Rightarrow \frac{\phi_4}{N\lambda} &= -\frac{M}{N}\frac{\phi_1}{M\lambda} + (-2\frac{M}{N} + 3) \\ \Rightarrow \phi_4 &= -\phi_1 + (3N - 2M)\lambda. \end{aligned} \tag{5.4}$$

Equation 5.4 describes the relationship between ϕ_4 and ϕ_1 when $x \in (1, 1.33]$. The same procedure is repeated to obtain other interception parameters and the expressions of these parameters are listed in Table 5.4. The relationship between ϕ_4 and ϕ_1 in this region is summarized in Equation 5.5.

$$\phi_4 = \begin{cases} -\phi_1 + K_1 & \text{if } 0 < \phi_1 \leq A \\ -\phi_1 + K_2 & \text{if } A < \phi_1 \leq B \\ -\phi_1 + K_3 & \text{if } B < \phi_1 \leq M\lambda \end{cases} \tag{5.5}$$

Next, the boundary parameters in Equation 5.5 are derived. Again, from the previous experience, the boundary parameters are expected to be solved by linear

x	K_1	K_2	K_3
$(1, 1.33]$	$(3N-2M)\lambda$	$(4N-2M)\lambda$	–
$(1.33, 1.49]$	$(3N-2M)\lambda$	$(4N-2M)\lambda$	$(5N-2M)\lambda$
$(1.49, 1.66]$	$(4N-2M)\lambda$	$(5N-2M)\lambda$	–
$(1.66, 2)$	$(4N-2M)\lambda$	$(5N-2M)\lambda$	$(6N-2M)\lambda$

Table 5.4: Interception parameters for $x \in (1, 2)$

combinations of interception parameters. However, this approach was proved to be unsuccessful in developing boundary parameters for ϕ_4 of the fast four-phase synchronizer. The reason is briefly discussed here. Assume for the moment, there exists the following relationship between the boundary parameter A and the interception parameters P and Q:

$$A = Pu + Qv$$

where u and v are two unknown variables.To solve them, two linearly independent equations:

$$A_a = P_a u + Q_a v$$
$$A_b = P_b u + Q_b v$$

are needed. In the case of the forward cycle, these two equations can be established by selecting two sets of simulation data of ϕ_3 and calculating their interception and boundary parameters. When it comes to the backward cycle, if the same method is applied, then the unknown variables u and v are expected be solved from the following two linear equations,

$$A_a = K_{1,a} u + K_{2,a} v$$
$$A_b = K_{1,b} u + K_{2,b} v$$

where the boundary parameters A_a,A_b and the corresponding interception parameters $(K_{1,a}, K_{2,a})$ and $(K_{1,b}, K_{2,b})$ are calculated from the ϕ_4 simulation results for two different clock ratios within the same region. However, the resultant two

x	A	B
$(1, 1.33]$	$-\beta\lambda - 1$	$-$
$(1.33, 1.49]$	$-\beta\lambda - 1$	$(\alpha - 2\beta)\lambda - 1$
$(1.49, 1.66]$	$(\alpha - 2\beta)\lambda - 1$	$-$
$(1.66, 2)$	$(\alpha - 2\beta)\lambda - 1$	$(\beta - 2\gamma)\lambda - 1$

Table 5.5: Boundary parameters expressed by interception parameters from ϕ_3

x	A	B
$(1, 1.33]$	$(3N - 2M)\lambda - 1$	$-$
$(1.33, 1.49]$	$(3N - 2M)\lambda - 1$	$(4N - 2M)\lambda - 1$
$(1.49, 1.66]$	$(4N - 2M)\lambda - 1$	$-$
$(1.66, 2)$	$(4N - 2M)\lambda - 1$	$(5N - 2M)\lambda - 1$

Table 5.6: Boundary parameters expressed by interception parameters from ϕ_3

linear equations are always linearly dependent. Therefore, determining boundary parameters demands the search for alternative ways. One such way that the author adopted was to express these boundary parameters in terms of interception parameters from the corresponding ϕ_3 expressions and the resultant boundary parameters for ϕ_4 are listed in Table 5.5. These interception parameters are then substituted by the corresponding M, N and λ, as is shown in Table 5.6.

To sum up, the expressions of the phase shift ϕ_4 in the region $x \in (1, 2)$ are developed. Equation 5.5 describes the relationship between ϕ_4 and ϕ_1, with the parameters summarized in Table 5.4 and Table 5.6. In this way, the behaviour of ϕ_4 is delineated by ϕ_1, M, N and λ, which makes it possible to predict ϕ_4 from ϕ_1.

5.2.2.3 Summary of ϕ_4 in $(0.5, 3]$

The behaviour of ϕ_4 in the x region $(2, 3)$ can be outlined in a similar fashion as that in $(1, 2)$. The derivations are omitted and only the final results are presented here. Equation 5.6 are suitable in this region and the parameters are outlined in Table 5.7.

x	K_1	K_2	K_3	K_4	A	B	C
$(2, 2.33]$	$(5N-2M)\lambda$	$(6N-2M)\lambda$	$(7N-2M)\lambda$	–	$(5N-2M)\lambda-1$	$(6N-2M)\lambda-1$	–
$(2.33, 2.49]$	$(5N-2M)\lambda$	$(6N-2M)\lambda$	$(7N-2M)\lambda$	$(8N-2M)\lambda$	$(5N-2M)\lambda-1$	$(6N-2M)\lambda-1$	$(7N-2M)\lambda-1$
$(2.49, 2.66]$	$(6N-2M)\lambda$	$(7N-2M)\lambda$	$(8N-2M)\lambda$	–	$(6N-2M)\lambda-1$	$(7N-2M)\lambda-1$	–
$(2.66, 3)$	$(6N-2M)\lambda$	$(7N-2M)\lambda$	$(8N-2M)\lambda$	$(9N-2M)\lambda$	$(6N-2M)\lambda-1$	$(7N-2M)\lambda-1$	$(8N-2M)\lambda-1$

Table 5.7: Parameters for ϕ_4 when $x \in (2, 3)$

$$\phi_4 = \begin{cases} -\phi_1 + K_1 & \text{if } 0 < \phi_1 \leq A \\ -\phi_1 + K_2 & \text{if } A < \phi_1 \leq B \\ -\phi_1 + K_3 & \text{if } B < \phi_1 \leq C \\ -\phi_1 + K_4 & \text{if } C < \phi_1 \leq M\lambda \end{cases} \tag{5.6}$$

Lastly, the cases when the clock ratio x becomes integer, i.e when $x = 1, 2, 3$ are discussed. The following conclusions can be made from the analyses of these cases:

1. ϕ_4 `at` $x = 1$ `and` $x = 2$. When x is equal to 1, the parameters are consistent with those when x is in $(1, 1.33]$. Similarly, parameters for $x = 2$ concur with those for the x region $(2, 2.33]$.

2. ϕ_4 `at` $x = 3$. In this case, the parameters do not fall into any of the said categories and are listed as follows: $K_1 = 4$, $K_2 = 8$, $K_3 = 12$, $K_4 = 16$, $A = 3$, $B = 7$, $C = 11$.

So far, the phase ϕ_4 has been quantified in terms of ϕ_1, M, N and λ when the clock ratio (M/N) is in $[1, 3]$. The expressions of ϕ_4 in the region $(0.5, 1)$ is excluded because the backward cycle can be directly obtained from the timing constraints in this region. For $x \in [1, 3]$, Equations 5.5 and 5.6 depict the ϕ_4 behaviour. The corresponding parameters are summarized in Table 5.4, 5.6 and 5.7.

5.2.3 The backward cycle times

As ϕ_3 and ϕ_4 have been developed, the backward cycle times are the summation of ϕ_3, ϕ_4 and T_{tx}, according to the timing diagram in Figure 3.6. The predicted

backward cycle times obtained from the analyses described in the previous sections are checked against from the simulation. The result shows perfect match between them. The data cycle times of the fast four-phase have less variations than the two-flop synchronizer, due to the tighter boundaries on the backward cycle times. The data cycle generally increases with the increases in clock ratio and the starting phase ϕ_1.

5.3 Preliminary investigation of a three-flop synchronizer design

From the study of the data cycle times of the two-flop, fast four-phase and fast four-phase synchronizers, it is found that the starting phase of each cycle ϕ_1 plays an important role. For a fixed clock ratio x, small ϕ_1 are more probable to produce short data cycle time. If the starting phase is reduced, a shorter data cycle time can occur. Since the range of the starting phase ϕ_1 was $(0, T_{rx}]$ from before, a natural choice would be to narrow down this range. One such attempt is to double the clock frequencies on both transmitter and receiver sides. The ranges of the crossing boundary times, previously denoted as ϕ_1 to ϕ_4 are halved as a consequence. To maintain the circuit reliability, an extra flip-flop is inserted on both sides. It adds an extra signal sampling for each clock domain, and is expected to balance the potential increase in the probability of getting into metastable state, due to the frequency doubling. Figure 5.3 demonstrates the proposed circuit. Note that the control logic that handles the signal transfer from the halved clocks(T'_{rx} and T'_{tx}) to the original clocks(T_{rx} and T_{tx}) is not included in this preliminary investigation and will be considered in the future. So it is assumed that the REG flop is always ready to receive data. This section outlines some preliminary study on the three-flop synchronizer design in the context of its timing boundaries, examples of data cycle time reduction and the metastability analysis.

5.3.1 Timing boundaries of the three-flop synchronizer

In Figure 5.3, on the receiver side, the clock that drives the three flip-flops is twice faster as the receiver clock. This clock cycle is denoted as T'_{rx}, in order to differentiate the receiver clock T_{rx}, and $T'_{rx} = T_{rx}/2$. The timing relation

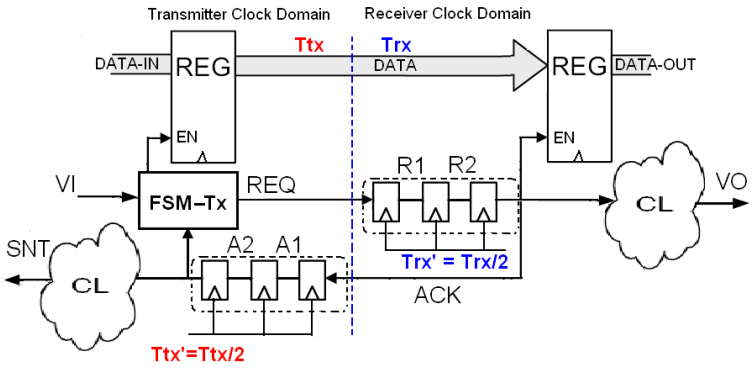


Figure 5.3: Three-flop synchronizer with halved clocks

diagram is drawn in Figure 5.4. The forward cycle and backward cycle are also symmetrical. From the timing diagram, the timing boundaries can be derived as shown below,

$$\phi_1 + 2T'_{rx} + \phi_2 = k'T'_{rx} \tag{5.7}$$

where $k' \in Z$ and $k' \geq 1$

$$0 < \phi_1 \leq T'_{rx} \tag{5.8}$$

$$0 < \phi_2 \leq T'_{tx} \tag{5.9}$$

Combining them,

$$2T'_{rx} < k'T'_{rx} \leq 3T'_{rx} + T'_{tx} \tag{5.10}$$

Let $x' = \dfrac{T'_{rx}}{T'_{tx}}$, then

$$2x' < k' \leq 3x' + 1. \tag{5.11}$$

From the timing diagram, the forward cycle time is $(k'+2)T'_{tx}$. Inequality 5.12 represents the timing boundary of the forward cycle time, expressed in multiples of T'_{tx}. It is also useful to express these timing boundaries in T_{tx} units, as is shown in Inequality 5.13. The term $(\frac{1}{2}k' + 1)$ is the forward cycle times in T_{tx} units. Because k' is a positive integer and is greater than 0, the forward cycle time is

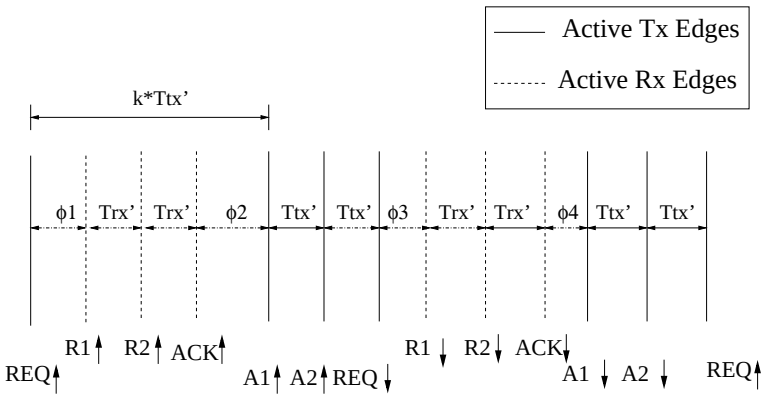


Figure 5.4: Three-flop synchronizer timing diagram

actually greater or equal to $\frac{3}{2}T_{tx}$.

$$2x' + 2 < k' + 2 \leq 3x' + 3 \tag{5.12}$$

$$x + 1 < \frac{1}{2}k' + 1 \leq \frac{3}{2}x + \frac{3}{2} \tag{5.13}$$

Figure 5.5 shows the plots of the two-flop, three-flop and fast four-phase synchronizers. For each synchronizer, the actual forward cycle time is between the upper and lower boundaries. So an estimate of the forward cycle times can be made by comparing their corresponding timing boundaries. Table 5.8 lists the range of forward cycle times based on different clock ratios. Note that the units of all the ranges is T_{tx}. It can be seen that the timing boundary for the three-flop synchronizer is more constrained than the other two synchronizers, which means that the variation in the actual forward cycle time is small compared with the other two. This is the consequence of reducing the cross boundary time of each cycle. Hence, the number of possible forward cycle times drops. From Figure 5.5 it therefore can be expected that the average forward cycle times for the three-flop synchronizer is lower than the two-flop synchronizer. If the three-flop is compared with the fast four-phase synchronizer, it shows improvement as well. For example, in the region $x \in (2, 2.5]$, the range of the three-flop synchronizer starts from $3.5T_{tx}$ till $5T_{tx}$. The range of the fast four-phase synchronizer spans to $6T_{tx}$. Similarly, due to its tighter forward cycle time range, this advantage of

x	$(0, 0.5]$	$(0.5, 1]$	$(1, 1.5]$	$(1.5, 2]$	$(2, 2.5]$	$(2.5, 3]$
two-flop	[2,3]	[2,4]	[3,5]	[3,6]	[4,7]	[4,8]
three-flop	[1.5,2]	[2,3]	[2.5,3.5]	[3,4.5]	[3.5,5]	[4.5,6]
fast four-phase	[1,2]	[1,3]	[2,4]	[2,5]	[3,6]	[3,7]

Table 5.8: Timing boundaries for two-flop, three-flop and fast four-phase synchronizers

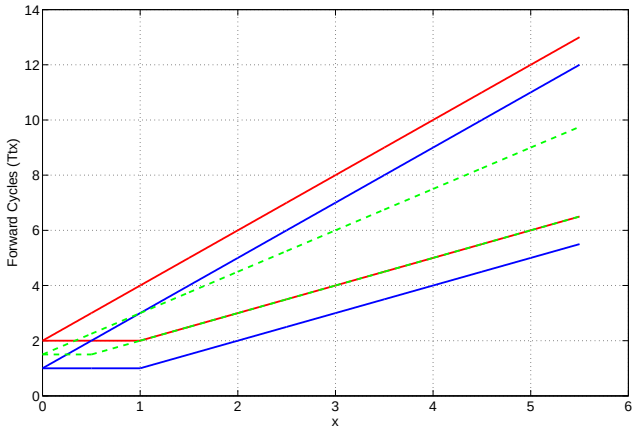


Figure 5.5: Plots of forward cycle timing boundaries for the three-flop(green), two-flop(red) and fast four-phase(blue) synchronizers

the three-flop synchronizer becomes more evident as the clock ratio increases.

An estimation of the data cycle timing boundaries is made from the plots of the data cycle boundaries of these synchronizers in Figure 5.6. The advantage of the three-flop synchornizer over the fast four-phase is degraded to some extent. This is mainly because of the overhead caused by the backward cycle times of the three-flop synchronizer. Its symmetrical structure forces the the low phase of REQ signal to propagated in the same way as the high phase, although no data is transmitted during this low phase at all. However, comparing to the two-flop synchronizer, improvement on the data cycle times can be predicted, judging from the slower increase in the data cycle time span on the three flop synchronizer in Figure 5.6.

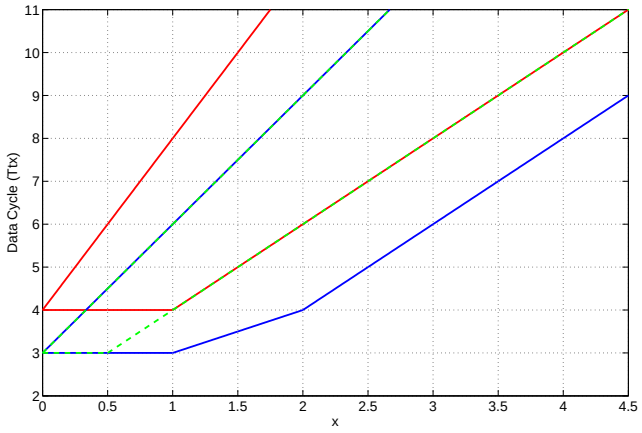


Figure 5.6: Plots of the data cycle timing boundaries for the three-flop(green), two-flop(red) and fast four-phase(blue) synchronizers

5.3.2 Two Examples of the three-flop synchronizer data cycle times

From the estimates of the timing boundaries in the last section, the three-flop synchronizer exhibit certain performance improvement over the two-flop and fast four-phase synchronizers. As the forward cycle time probability distribution for each clock ratio is unknown, it is difficult to evaluate how much performance improvement can be obtained for the three-flop synchronizer. This section provides two examples to quantify the data cycle time of the three-flop synchronizer and compares it to the other two synchronizers.

The clock ratios T'_{rx}/T'_{tx} chosen for these two examples are 7/4 and 3/4. Using the same clock domain model from Section 4.1.1, the cycle times of the three-flop synchronizers were calculated manually and are separately listed in Table 5.9 and 5.10. Abbreviations used in these tables are FW(forward cycle time), BW(backward cycle time) and DC(data cycle time). Note that in order to ease the comparison with other synchronizers, the cycle times for the three-flop synchronizer are expressed in terms of T_{tx} in these tables. Also, because the resolution factor λ for the clock domain model was set to 4 previously but both the transmitter and receiver clock period are halved for the three-flop synchronizer, the actual resolution factor for the three-flop synchronizer becomes 2 instead. Therefore, if the original transmitter and receiver clocks are $T_{tx} = 16\Delta t$ and

$\phi_1(\Delta t)$	FW	BW	DC	2-flop DC	fast 4-phase DC	$diff1$	$diff2$
1	3	3.5	6.5	7	5	0.5	−1.5
2	3	3.5	6.5	7	5	0.5	−1.5
3	3	3.5	6.5	7	5	0.5	−1.5
4	3.5	3.5	7	9	5	2	−2
5	3.5	3.5	7	9	5	2	−2
6	3.5	3.5	7	9	5	2	−2
7	3.5	3.5	7	9	5	2	−2
8	3.5	3.5	7	9	6	2	−1
9	3.5	3.5	7	7	6	0	−1
10	3.5	3.5	7	7	6	0	−1
11	3.5	3.5	7	7	6	0	−1
12	4	3.5	7.5	8	6	0.5	−1.5
13	4	3.5	7.5	8	6	0.5	−1.5
14	4	3.5	7.5	8	6	0.5	−1.5
15	3	3.5	6.5	8	6	1.5	−0.5
16	3	3.5	6.5	8	6	1.5	−0.5
17	3	3.5	6.5	8	6	1.5	−0.5
18	3.5	3.5	7	8	6	1	−1
19	3.5	3.5	7	8	6	1	−1
20	3.5	3.5	7	10	6	3	−1
21	3.5	3.5	7	10	6	3	−1
22	3.5	3.5	7	10	6	3	−1
23	3.5	3.5	7	10	6	3	−1
24	3.5	3.5	7	10	7	3	0
25	3.5	3.5	7	8	7	1	0
26	4	3.5	7.5	8	7	0.5	−0.5
27	4	3.5	7.5	8	7	0.5	−0.5
28	4	3.5	7.5	9	7	1.5	−0.5

Table 5.9: Comparison of the cycle times for two-flop, three-flop and fast four-phase synchronizers when $x = 7/4$

$T_{rx} = 28\Delta t$ as were described in Section 4.1.1, the counterparts after frequency doubling become $T'_{tx} = 8\Delta t$ and $T'_{rx} = 14\Delta t$. The range of ϕ_1 for the three-flop synchronizer also becomes $(0, 14\Delta t]$. The reduction in resolution however does not compromise the final results, because even with the resolution factor of 2, all the major features of the synchronizer behaviour can still be exposed, as is demonstrated in Table 4.1. The last two columns list the difference in data cycle times between the three-flop synchronizer and the other two synchronizers. The column entitled $diff1$ calculates the difference, $DC_{2flop} - DC_{3flop}$ for each ϕ_1. Similarly, the rightmost column entitled $diff2$ calculates the difference $DC_{fast4phase} - DC_{3flop}$.

Table 5.9 and 5.10 clearly tell that under these two clock ratios, the average data cycle time of the three-flop synchronizer are between that of the two-flop and the fast four-phase, which agrees with the conclusion from the analyses of

$\phi_1(\Delta t)$	FW	BW	DC	2-flop DC	fast 4-phase DC	$diff1$	$diff2$
1	2	2.5	4.5	5	3	0.5	−1.5
2	2	2.5	4.5	5	3	0.5	−1.5
3	2	2	4	5	3	1	−1
4	2.5	2	4.5	6	4	1.5	−0.5
5	2.5	1.5	4	6	4	2	0
6	2.5	2	4.5	6	4	1.5	−0.5
7	2	2.5	4.5	6	4	1.5	−0.5
8	2	2.5	4.5	6	4	1.5	−0.5
9	2	2	4	6	4	2	0
10	2.5	2	4.5	6	4	1.5	−0.5
11	2.5	1.5	4	6	4	2	0
12	2.5	2	4.5	5	4	0.5	−0.5

Table 5.10: Comparison of the cycle times for two-flop, three-flop and fast four-phase synchronizers when $x = 3/4$

the their timing boundaries. The three-flop design has demonstrated shorter data cycle times than the two-flop synchronizer, and this advantage becomes more evident as the starting phase ϕ_1 increases, which can be seen from the comparison of the two halves in each table. Also, the better performance improvement is obtained as the clock ratio x gets bigger. As for the fast four-phase synchronizer, these two examples demonstrated degraded performance of the three-flop compared to the fast four-phase synchronizer. However, they provide some interesting directions for further design refinement. One such direction is that whether the backward cycle time of the three-flop synchronizer can be made shorter. Currently, the forward and backward cycle times for the three-flop synchronizer are similar, as can be seen from the FW and BW columns in these tables. However, as is mentioned before, the backward cycle time is not used for useful data transfer at all. Hence, if the backward cycle time was reduced, the three-flop synchronizer could demonstrate potential to achieve shorter data cycle times than the fast four-phase synchronizer. This can also be seen from the plots of the forward(Figure 5.5) and backward (Figure 5.7) timing boundaries. The forward cycle times do not reveal substantial difference between these two synchronizers, while the backward cycle time for the fast four-phase is significantly shorter than the three-flop synchronizer.

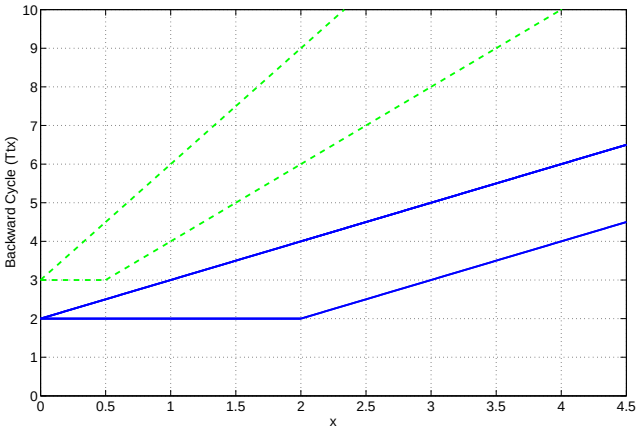


Figure 5.7: Backward cycle timing boundaries for the fast four-phase(blue) and three-flop(green) synchronizers

5.3.3 Reliability of the three-flop synchronizer

Lastly, the reliable operation of the three-flop synchronizer is briefly discussed. From Chapter 2, the synchronization failure rate of a flip-flop is usually measured in mean time between failure (MTBF), as

$$MTBF = \frac{e^{\frac{S}{\tau}}}{WF_cF_d}.$$

Using the parameters from the first example in Section 2.1.1, the $MTBF$ for the two-flop synchronizer, whose settling time S is one clock cycle, can be calculated as shown below:

$$MTBF = \frac{e^{\frac{5n\ sec}{66p\ sec}}}{132p\ sec \cdot 200MHz \cdot 20MHz} \approx 10^{20}\ years.$$

In the case of the three-flop synchronizer, both F_c and F_d are doubled. The metastability settling time S is twice the clock cycles, i.e.$S = \frac{2}{F_c}$. Its $MTBF$ can be calculated as shown below:

$$MTBF = \frac{e^{\frac{5n\ sec}{66p\ sec}}}{132p\ sec \cdot 400MHz \cdot 40MHz} \approx 10^{19}\ years.$$

Although the $MTBF$ is significantly shorter than that for the two-flop, it

still gives sufficient reliability. If more aggressive parameters are used, for example, $F_c = 1GHz$, $F_d = 100MHz$ and the settling time S allowed is $1ns$, both $MTBF$ values are shorter than one second. Therefore, for high speed applications, multi-cycle settling time and more aggressive flip-flops with shorter τ should be considered to achieve satisfactory reliability.

5.4 Summary

The first half of this chapter presents the analyses of the data cycle times for the fast four-phase and fast two-phase synchronizers. The forward cycle times for the said two synchronizers are obtained by modifying the two-flop synchronizer forward cycle time. The fast four-phase synchronizer backward cycle time is derived in two steps, ϕ_1 vs ϕ_3 and ϕ_1 vs ϕ_4. Equations defining these relationships are extracted from the simulation data.

The observations of the data cycle times of the two-flop, fast four-phase and fast two-phase synchronizers indicates that small starting phase tends to produce short data cycle time. A new design approach, the three-flop synchronizer with double frequencies, is proposed based on this observation and is outlined in the second half of this chapter. Analyses of the three-flop synchronizer data cycle timing boundaries and two examples of the data cycle times predict performance improvement over the two-flop synchronizer. It also shows longer data cycle time than the fast four-phase synchronizer and provides further refinement of the three-flop synchronizer design. The reliability of the flip-flop after frequency doubling is then analyzed and an increase in the $MTBF$ is found. However, since the three-flop synchronizer allows an additional clock cycle to resolve metastability, it gives sufficiently reliable performance for normal clock speed.

Chapter 6

Performance evaluation of the two-flop, fast four-phase and fast two-phase synchronizers

This chapter summarizes the data cycle times of the two-flop, fast-four phase and fast two-phase synchronizers and evaluates the average data cycle time for the data-burst transfer. Based on the analyses from Chpater 4 and Chapter 5, a tool, Synchronizers Data Cycle Analyzer (SyDCA), was developed to automatically calculate the data cycle times of the said three synchronizers. The best case, worst case and the average of these synchronizers' data cycle times are obtained from the SyDCA tool. Their average data cycle times are compared with the estimated ones from Chpater 3 and justifications are made on the accuracies of the estimation. The data cycle times are also obtained for the burst-mode data transfers. The average data cycle times are calculated for the data burst consisting of 100, 500 and 1000 words respectively and are compared. Examples of different starting phase of the first data word are described and its influence on the average data cycle time is analyzed.

6.1 Data cycle time evaluation

6.1.1 Summary of the data cycle times for the three synchronizers

In the previous two chapters, the data cycle times were studied by carrying out separate analyses for both forward and backward cycle times. The following relationships were analyzed for the two-flop, fast four-phase and fast two-phase synchronizers: (1) dependencies of the backward cycle starting phase ϕ_3 on the forward cycle starting phase ϕ_1, (2) the relationship between ϕ_1 and the forward cycle time and (3) the relationship between ϕ_1 and the backward cycle time. The outcome of these analyses makes it possible to attain the accurate data cycle times from the starting phase ϕ_1, the clock ratio $x = M/N$ and the resolution factor λ. This section briefly summarizes these relationships.

6.1.1.1 Two-flop and fast two-phase synchronizers

The relationship between ϕ_3 and ϕ_1 for the two-flop synchronizer can be described by a set of parallel lines. The expressions of these lines form a piecewise linear function of ϕ_1 as is described in Equation 4.2:

$$\phi_3 = \begin{cases} \phi_1 + \alpha\lambda & \text{if } 0 < \phi_1 \leq A \\ \phi_1 + \beta\lambda & \text{if } A < \phi_1 \leq B \\ \phi_1 + \gamma\lambda & \text{if } B < \phi_1 \leq C \\ \phi_1 + \theta\lambda & \text{if } C < \phi_1 \leq D \\ \phi_1 + \mu\lambda & \text{if } D < \phi_1 \leq M\lambda. \end{cases}$$

Two types of parameters were defined in Equation 4.2: the interception parameters expressed in Greek letters (except λ) and the boundary parameters expressed in capital letters. These parameters were then expressed as linear combinations of M and N. The availability and expressions of these parameters are listed in Table 6.1.

For the two-flop synchronizer, the relationship between the forward cycle and ϕ_1 was determined from its timing boundaries and the critical ϕ_1 values. The Cycle-time Lookup Table shown in Table 6.2 summarizes this relationship. For the relationship between ϕ_1 and the backward cycle time, the following two steps

x	α	β	γ	θ	μ	A	B	C	D
$(0.5, 0.6]$	$4M-2N$	$3M-2N$	$5M-3N$	–	–	$(2N-3M)\lambda$	$(N-M)\lambda-1$	–	–
$(0.6, 0.66]$	$4M-2N$	$3M-2N$	$5M-3N$	$4M-3N$	–	$(2N-3M)\lambda$	$(N-M)\lambda-1$	$(3N-4M)\lambda$	–
$(0.66, 0.75]$	$3M-2N$	$4M-3N$	–	–	–	$(N-M)\lambda-1$	–	–	–
$(0.75, 1)$	$3M-2N$	$4M-3N$	$3M-3N$	–	–	$(N-M)\lambda-1$	$(3N-3M)\lambda$	–	–
1	0	–	–	–	–	–	–	–	–
$(1, 1.33]$	$3M-3N$	$2M-3N$	$3M-4N$	–	–	$(3N-2M)\lambda$	$(2N-M)\lambda-1$	–	–
$(1.33, 1.5)$	$3M-3N$	$2M-3N$	$3M-4N$	$2M-4N$	–	$(3N-2M)\lambda$	$(2N-M)\lambda-1$	$(4N-2M)\lambda$	–
$[1.5, 1.66]$	$2M-3N$	$3M-4N$	$2M-4N$	$3M-5N$	–	$(2N-M)\lambda-1$	$(4N-2M)\lambda$	$(3N-M)\lambda-1$	–
$(1.66, 2)$	$2M-3N$	$3M-4N$	$2M-4N$	$3M-5N$	$2M-5N$	$(2N-M)\lambda-1$	$(4N-2M)\lambda$	$(3N-M)\lambda-1$	$(5N-2M)\lambda$
2	0	1	-1	0	–	$\lambda-1$	λ	$2\lambda-1$	–
$(2, 3)$	$2M-4N$	$2M-5N$	$2M-6N$	$2M-7N$	–	$(3N-M)\lambda-1$	$(4N-M)\lambda-1$	$(5N-M)\lambda-1$	–
3	1	0	-1	-2	–	$\lambda-1$	$2\lambda-1$	$3\lambda-1$	–

Table 6.1: Two-flop synchronizer interception parameters and boundary parameters for ϕ_3

are required: (1) determine ϕ_3 using the piecewise linear function 4.2 and (2) look up in Table 6.2 the backward cycle time corresponding to ϕ_3 calculated from the previous step. Note that the variable ϕ in Table 6.2 refers to either ϕ_1 or ϕ_3 where applicable.

As for the fast two-phase synchronizer, its data cycle only consists of one forward cycle. Also, because its forward cycle is identical to that of the two-flop synchronizer, Table 6.2 was used to calculate the forward cycle time. Besides, the Equation 4.2 can be applied to predict the starting phase for the next data cycle of the fast two-phase synchronizer.

6.1.1.2 Fast four-phase synchronizer

As stated before, because the flip-flops with asynchronous reset are used in the fast four-phase synchronizer design, the symmetry between the backward and forward cycles does not exist any more. Therefore, different approaches were applied to obtain the forward and backward cycle times for the fast four-phase synchronizer. For the forward cycle time, it has the same critical ϕ_1 as the two-flop synchronizer and the forward cycle time is always one T_{tx} shorter than the two-flop forward cycle time. So Table 6.2 is sufficient to depict the forward cycle time.

The difficulties arise for the calculation of the backward cycle times. From the timing relation diagram (Figure 3.6), the backward cycle consists of ϕ_3, ϕ_4 and one transmitter clock.One direct approach adopted by the author was to find the following relationships separately: ϕ_3 vs ϕ_1 and ϕ_4 vs ϕ_1. Equation 4.2 can also be used to define ϕ_3 and the parameters are listed in Table 5.2.

x	ϕ_c	$ForwardCycles(T_{tx})$
$x \in (0.5, 1)$	$\phi_c = (\frac{1}{x} - 1)M\lambda$	$\phi \in (0, \phi_c)$ Forward Cycle=2 $\phi \in [\phi_c, M\lambda]$ Forward Cycle=3
$x \in [1, 1.5)$	$\phi_c = (\frac{2}{x} - 1)M\lambda$	$\phi \in (0, \phi_c)$ Forward Cycle=3 $\phi \in [\phi_c, M\lambda]$ Forward Cycle=4
$x \in [1.5, 2)$	$\phi_{c,a} = (\frac{2}{x} - 1)M\lambda$ $\phi_{c,b} = (\frac{3}{x} - 1)M\lambda$	$\phi \in (0, \phi_{c,a})$ Forward Cycle=3 $\phi \in [\phi_{c,a}, \phi_{c,b})$ Forward Cycle=4 $\phi \in [\phi_{c,b}, M\lambda]$ Forward Cycle=5
$x \in [2, 2.5)$	$\phi_{c,a} = (\frac{3}{x} - 1)M\lambda$ $\phi_{c,b} = (\frac{4}{x} - 1)M\lambda$	$\phi \in (0, \phi_{c,a})$ Forward Cycle=4 $\phi \in [\phi_{c,a}, \phi_{c,b})$ Forward Cycle=5 $\phi \in [\phi_{c,b}, M\lambda]$ Forward Cycle=6
$x \in [2.5, 3)$	$\phi_{c,a} = (\frac{3}{x} - 1)M\lambda$ $\phi_{c,b} = (\frac{4}{x} - 1)M\lambda$ $\phi_{c,c} = (\frac{5}{x} - 1)M\lambda$	$\phi \in (0, \phi_{c,a})$ Forward Cycle=4 $\phi \in [\phi_{c,a}, \phi_{c,b})$ Forward Cycle=5 $\phi \in [\phi_{c,b}, \phi_{c,c})$ Forward Cycle=6 $\phi \in [\phi_{c,c}, M\lambda]$ Forward Cycle=7
$x = 3$	$\phi_{c,a} = (\frac{4}{x} - 1)M\lambda$ $\phi_{c,b} = (\frac{5}{x} - 1)M\lambda$ $\phi_{c,c} = (\frac{6}{x} - 1)M\lambda$	$\phi \in (0, \phi_{c,a})$ Forward Cycle=5 $\phi \in [\phi_{c,a}, \phi_{c,b})$ Forward Cycle=6 $\phi \in [\phi_{c,b}, \phi_{c,c})$ Forward Cycle=7 $\phi \in [\phi_{c,c}, M\lambda]$ Forward Cycle=8

Table 6.2: Cycle-time Lookup Table for the two-flop synchronizer

The relationship between ϕ_4 and ϕ_1 was derived in Section 5.2.2. It was found that when $x \in (0.5, 1)$, the backward cycle time was a constant $2T_{tx}$. So only the ϕ_4 expressions within $x \in [1, 3]$ were explored. Equation 6.1 was found to describe ϕ_4 and the parameters are listed in Table 6.3.

$$\phi_4 = \begin{cases} -\phi_1 + \alpha\lambda & \text{if } 0 < \phi_1 \leq A \\ -\phi_1 + \beta\lambda & \text{if } A < \phi_1 \leq B \\ -\phi_1 + \gamma\lambda & \text{if } B < \phi_1 \leq C \\ -\phi_1 + \theta\lambda & \text{if } C < \phi_1 M\lambda. \end{cases} \tag{6.1}$$

6.1.1.3 Synchronizers Data Cycle Analyzer (SyDCA)

The methods presented in Section 6.1.1.1 and Section 6.1.1.2 were programmed in Matlab to construct the Synchronizers Data Cycle Analyzer (SyDCA) tool. This tool calculates the forward and backward cycle times based on the starting phase ϕ_1 and the clock ratio $x = M/N$. The data cycle times are then obtained by

x	α	β	γ	θ	A	B	C
$[1, 1.33]$	$3N-2M$	$4N-2M$	–	–	$(3N-2M)\lambda-1$	–	–
$(1.33, 1.49]$	$3N-2M$	$4N-2M$	$5N-2M$	–	$(3N-2M)\lambda-1$	$(4N-2M)\lambda-1$	–
$(1.49, 1.66]$	$4N-2M$	$5N-2M$	–	–	$(4N-2M)\lambda-1$	–	–
$(1.66, 2)$	$4N-2M$	$5N-2M$	$6N-2M$	–	$(4N-2M)\lambda-1$	$(5N-2M)\lambda-1$	–
$[2, 2.33]$	$5N-2M$	$6N-2M$	$7N-2M$	–	$(5N-2M)\lambda-1$	$(6N-2M)\lambda-1$	–
$(2.33, 2.49]$	$5N-2M$	$6N-2M$	$7N-2M$	$8N-2M$	$(5N-2M)\lambda-1$	$(6N-2M)\lambda-1$	$(7N-2M)\lambda-1$
$(2.49, 2.66]$	$6N-2M$	$7N-2M$	$8N-2M$	–	$(6N-2M)\lambda-1$	$(7N-2M)\lambda-1$	–
$(2.66, 3)$	$6N-2M$	$7N-2M$	$8N-2M$	$9N-2M$	$(6N-2M)\lambda-1$	$(7N-2M)\lambda-1$	$(8N-2M)\lambda-1$
3	1	2	3	4	$\lambda-1$	$2\lambda-1$	$3\lambda-1$

Table 6.3: Fast four-phase synchronizer parameters for ϕ_4 when $x \in [1, 3]$

summing the forward and backward cycle times. All the cycle times, including the forward, backward and data cycle times for the three synchronizers, were verified against the cycle times from the model simulation and the results show perfect match. The data cycle look-up table between ϕ_1 and the data cycle times were constructed for each synchronizer. So each time the tool is run to calculate data cycle time, efforts of calculating the forward and backward cycle times as well as the dependencies between them can be saved.

6.1.2 Best and worst cases

Two important performance indicators of the synchronizer data cycle time are the best and worst cases. These cases provide the upper and lower limits for the cycle times, as an alternative to the timing boundaries introduced in Chapter 3. Figure 6.1 shows the best and worst cases for each of the three synchronizers. There are three spikes at $x = 1$ for the two-flop and fast four-phase synchronizers and at $x = 2$ for the two-flop synchronizer. For example, the worst data cycle time remains at $10T_{tx}$ before $x = 2$, but jumps to $12T_{tx}$ when the clock ratio becomes 2. It falls back to $10T_{tx}$ after x moves past this point. This observation of the worst data cycle time at $x = 2$ agrees with the example shown in Table 4.2. As was explained in this example, the existence of these spikes is a consequence of coincident transmitter and receiver active clock edges, as the coincident clock edges are sometimes the source of the worst case scenarios.

Within the x range $(0.5, 3]$, the worst data cycle spans from $5T_{tx}$ to $14T_{tx}$ for the two-flop synchronizer, $4T_{tx}$ to $11T_{tx}$ for the fast four-phase synchronizer and $3T_{tx}$ to $8T_{tx}$ for the fast two-phase synchronizer. Obviously, the worst case data cycle time increases more rapidly for the two-flop synchronizer than the other two synchronizers. A similar conclusion can be attained from the comparisons of the

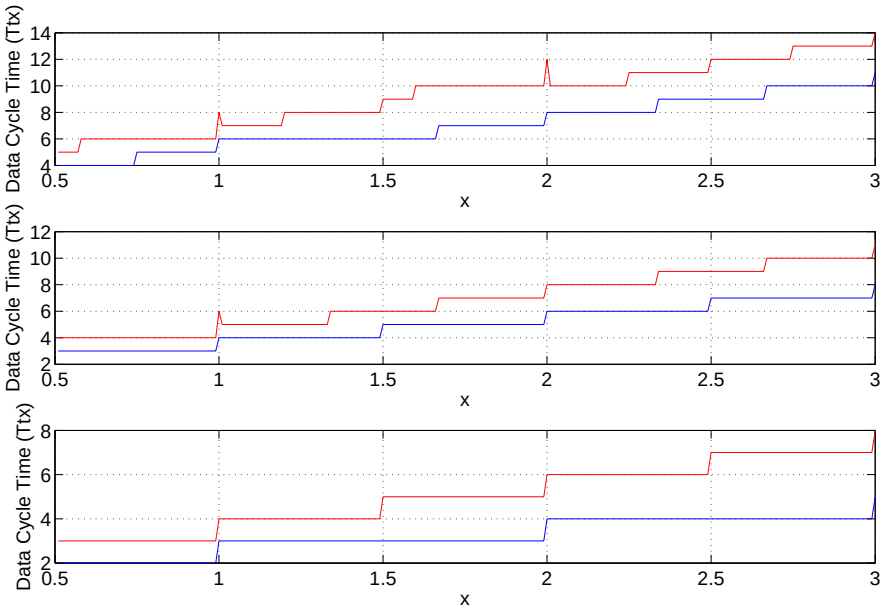


Figure 6.1: The best(blue) and worst(red) data cycle times for the two-flop (top), fast four-phase (middle) and two-phase (bottom) synchronizers

best case performances. These facts indicate that as the clock ratio increases, the data cycle time gets significantly longer for the two-flop synchronizer. Hence, the advantage of the fast two-phase synchronizer in saving data cycle times becomes more dominant at higher clock ratios.

The width of the gap between the best and worst case indicates the number of possible data cycle times. So a wider gap means that more possible data cycle times can occur. If the data cycle times within $x \in [1, 2]$ are considered for the two-flop and fast four-phase synchronizers, the gap for the two-flop synchronizer ranges from T_{tx} at $x = 1.1$ to $4T_{tx}$ at $x = 2$, whereas the gap only ranges from T_{tx} to $2T_{tx}$ in the same region for the fast two-phase synchronizer. This implies that there are more possible data cycle times for the two-flop synchronizer than the fast four-phase synchronizer. It in fact causes more variations in the average data cycle time in this region, as will be seen later, and complicates the estimation of the average data cycle time.

6.1.3 Average data cycle times

As another important indicator of the synchronizer performance, the average data cycle time provides information on the majority data cycle behaviour. With the help of the SyDCA tool, the data cycle times were obtained. The average data cycle times are calculated and plotted in Figure 6.2. The following conclusions are made from the comparisons of the average data cycle times in Figure 6.2:

1. Non-linearity in the two-flop average data cycle time. The two-flop synchronizer average data cycle time displays strong non-linear behaviour when $x \in (0.5, 2]$. Although the overall trend of the average data cycle time is increasing as the clock ratio increases, fluctuations in the average data cycle time results in a non-monotonic increase in this region.

2. Symmetrical non-linear regions of the two-flop average data cycle time. There are two regions where the magnitude of the fluctuations in the average data cycle time is significant. These regions are $x \in [0.6, 0.75]$ and $x \in [1.33, 1.67]$. Some interesting similarities exist in both regions. The boundaries of the said two ranges are expressed in fractional format, which are $[\frac{3}{5}, \frac{3}{4}]$ and $[\frac{4}{3}, \frac{5}{3}]$. Take the lower bound of the first region $\frac{3}{5}$ as an example. At $x = \frac{3}{5}$, the average data cycle time reaches a peak, after which it starts to drop. The reciprocal of this clock ratio, $x = \frac{5}{3}$, is actually the upper bound of the other region. The average data cycle time also reaches a local peak at this point. Similarly, the reciprocal of the higher boundary value $\frac{3}{4}$ in the first region is the lower bound of the second region. At both points, the average data cycle times also reach their local peak values respectively. Noticeable local peaks also occur at $x = 1$, $x = 1.5$ and $x = 2$. Extra average data cycle times incur at these ratios, as a result of the coincident clock edges.

3. Variations in the other two average data cycle time plots. Compared to the two-flop synchronizer case, the average data cycle times for the fast four-phase and fast two-phase synchronizers exhibits less fluctuations, though the spikes still exist at integral clock ratios and the increase in the average data cycle times is still non-linear within the x range $(0.5, 2]$.

4. Linear regions. All the three plots show linear increase in the average data cycle times when x becomes greater than 2. As the transmitter and receiver

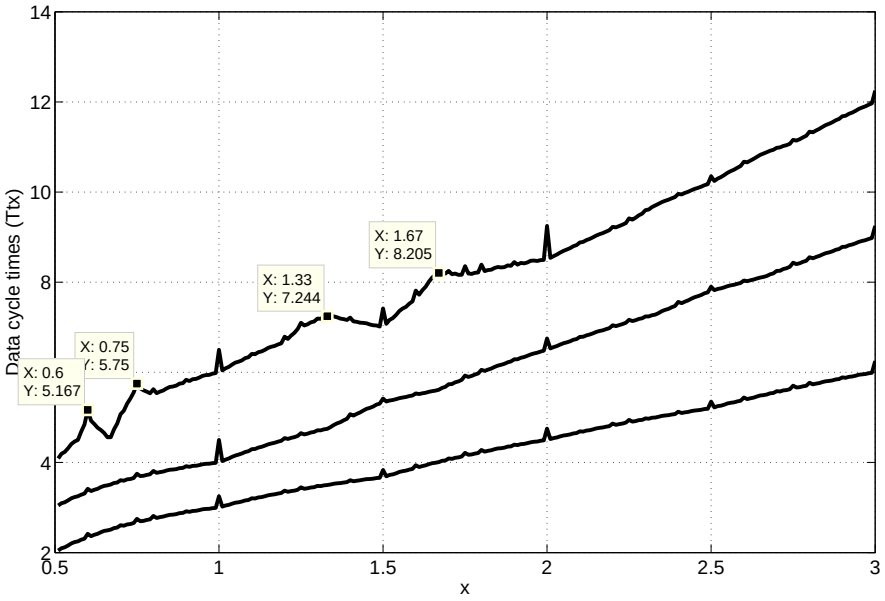


Figure 6.2: Average data cycle times for the two-flop (top), fast four-phase (middle) and fast two-phase (bottom) synchronizers

clocks are more dissimilar, the data cycle times is largely determined by the slower clock.

In Chpater 3, pure analytical approach was applied to calculate the average forward cycle times for these three synchronizers. For the two-flop synchronizer, the average backward cycle time was not dealt with due to the unknown backward cycle dependencies on the forward cycle. Instead, the data cycle time was approximated as twice as the forward cycle time. The fast four-phase synchronizer backward cycle time was approximated from several selected experimental backward data cycle times. In addition, the effect of coincident clock edges was not taken into account when the average data cycles were calculated. Since the accurate average data cycle times have been obtained at this stage, a comparison is made between the analytical estimation from two cases and the actual cycle times from the SyDCA tool calculation, as shown in Figure 6.3. Apart from the effect of the coincident clock edges, the analytical estimation of the fast four-phase and fast two-phase synchronizers shows good match with the actual average data cycle times. In the case of the two-flop synchronizer, more discrepancies occur between the estimation and the actual cycle times. The difference is mainly caused by the fluctuations in the actual data cycle time, which is a direct

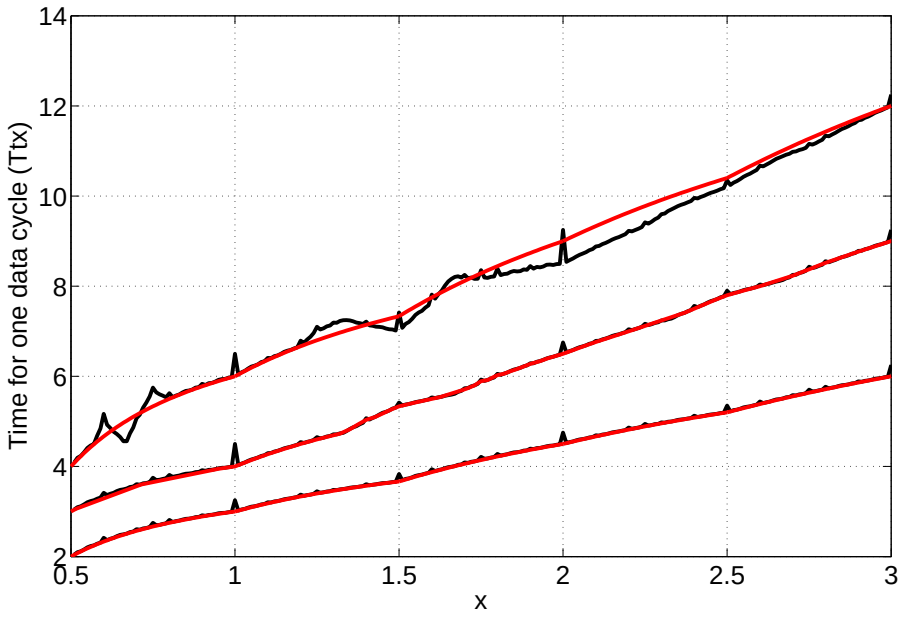


Figure 6.3: Comparisons of the estimated average data cycle times (red) with the actual ones (black) for the two-flop (top), fast four-phase (middle) and fast two-phase (bottom) synchronizers

consequence of the backward cycle dependencies on the forward cycle. Though a similar dependency also exists for the fast four-phase synchronizer, the tight backward cycle timing constraints limit the options of the possible backward cycle times and hence result less variations.

If the analytical estimation is used to predict the single data cycle behviour of the said three synchronizers, it gives relatively accurate results despite of the inaccuracies at the integer clock ratios. As for the estimation for the two-flop synchronizer, the author regards it inaccurate as it not only loses the effect of the coincident clock edges, but masks important non-linearity in the single data cycle time in certain regions. It can also be inferred from this comparison that for the data burst transfers, larger impact of the cycle time dependencies will occur, causing the more variations in the average data cycle times for all three synchronizers.

6.2 Evaluation of the burst-mode transfer data cycle times

So far, the single data cycle times have been investigated for the two-flop, fast four-phase and fast two-phase synchronizers. The difficulty in obtaining the data cycle times stems from the cycle time dependency. This dependency imposes variations in the average data cycle times, which degrades the accuracy of analytical estimation. During the burst-mode data transfer, this dependency exists for each data transfer, and it was expected that the overall data cycle times would incur more fluctuations. Given the complexity of the cycle time dependencies and their potentially larger impact (as it influences the data cycle time for every word), it is insufficient to evaluate the overall data cycle times purely from the analytical estimation. With the help of SyDCA tool, this section presents a relatively accurate evaluation of the average data cycle time under burst-mode transfer.

6.2.1 Extension of the SyDCA tool

SyDCA was previously described as a tool to analyze the synchronizers forward and backward cycle times, including the backward cycle dependency on the forward cycle. This section describes the extensions of the SyDCA tool to analyze dependencies between two consecutive data cycles, which will assist the study of data cycle times of the data burst. For the two-flop and fast two-phase synchronizers, the existing feature of this tool is sufficient to deal with these dependencies. In the case of the fast four-phase synchronizer, new relationship needs to be established between the two starting phases of two successive data cycles.

6.2.1.1 SyDCA for the two-flop and fast two-phase synchronizers

Previously, the relationship between the starting phases, ϕ_1 and ϕ_3, of the forward and backward cycles was established and ϕ_3 was found to be a piecewise linear function of ϕ_1. This relationship is denoted as $\phi_3 = F(\phi_1)$. The symmetrical nature of the two-flop synchronizer forward and backward cycle times implies that the starting phase $\phi_{1,next}$ of the next data cycle can be obtained by applying the same function F to ϕ_3, i.e., $\phi_{1,next} = F(\phi_3)$. Therefore, ϕ_1 and $\phi_{1,next}$ have the following relationship:

$$\phi_{1,next} = F[F(\phi_1)].$$

Recall that the data cycle look-up table was set up previously in SyDCA to build the relationship between ϕ_1 and the data cycle time. As the information on the starting phase of the next data cycle $\phi_{1,next}$ is known, the data cycle time corresponding to $\phi_{1,next}$ can be read off from the same table. During the burst-mode transfer, two steps, calculating the next starting phase $\phi_{1,next}$ and looking up the data cycle time, are repeated for each data word. The data cycle time for each data word is stored and hence the average data cycle time can be calculated.

The calculation of the fast two-phase synchronizer data cycle time is actually identical to the calculation of the backward cycle time for the two-flop synchronizer and the description of relevant procedures is omitted.

6.2.1.2 SyDCA for the fast four-phase synchronizers

Like the two-flop synchronizer, the calculation of the fast four-phase synchronizer data cycle times during the data burst also require two steps: calculating $\phi_{1,next}$ and looking up the data cycle time. The data cycle look-up table was already constructed during the course of obtaining the forward and backward cycle times but no relationship has been established between ϕ_1 and $\phi_{1,next}$. Since the forward and backward cycles are not symmetrical any more, no existing phase relation can be re-used to calculate $\phi_{1,next}$. It is therefore necessary to search for the relationship between ϕ_1 and $\phi_{1,next}$ for the fast four-phase synchronizer. Investigation of the model simulation shows that $\phi_{1,next}$ is a piecewise linear function of ϕ_1. Equation 4.1 describes this function and the parameters are shown in Table 6.4.

6.2.2 Average data cycle time for the burst-mode data transfer

6.2.2.1 Examples of data burst transfers

The average data cycle time for the data burst can be calculated using the SyDCA tool with the added functionality. Three streams of 100, 500 and 1000 data words were tested separately and the average data cycle times for the two-flop synchronizer were obtained and plotted in Figure 6.4.

The starting phase ϕ_1 for the first data word in each stream was set to be Δt, which was previously defined as the small time step Δt that ϕ_1 takes to sweep across its range $(0, T_{rx}]$. So Δt is the smallest phase difference between the transmitter and receiver. Apart from some minor discrepancies, the average

x	α	β	γ	θ	μ	A	B	C	D
$(0.5, 0.57]$	$6M-3N$	$5M-3N$	$7M-4N$	–	–	$(3N-5M)\lambda$	$(N-M)\lambda-1$	–	–
$(0.57, 0.59]$	$6M-3N$	$5M-3N$	$7M-4N$	$6M-4N$	–	$(3N-5M)\lambda$	$(N-M)\lambda-1$	$(4N-6M)\lambda$	–
0.6	$5M-3N$	1	-2	–	–	7	8	–	–
$(0.6, 0.66]$	$5M-3N$	$6M-4N$	–	–	–	$(N-M)\lambda-1$	–	–	–
$(0.66, 0.74]$	$5M-3N$	$4M-3N$	$6M-4N$	$5M-4N$	–	$(3N-4M)\lambda-1$	$(N-M)\lambda-1$	$(4N-5M)\lambda$	–
0.75	$4M-3N$	2	-1	–	–	3	4	–	–
$(0.75, 0.8]$	$4M-3N$	$5M-4N$	–	–	–	$(N-M)\lambda-1$	–	–	–
$(0.8, 0.99]$	$4M-3N$	$5M-4N$	$4M-4N$	–	–	$(N-M)\lambda-1$	$(4N-4M)\lambda$	–	–
1	0	–	–	–	–	–	–	–	–
$[1, 1.25]$	$4M-4N$	$3M-4N$	$4M-5N$	–	–	$(4N-3M)\lambda$	$(3N-2M)\lambda-1$	–	–
$(1.25, 1.33]$	$4M-4N$	$3M-4N$	$4M-5N$	$3M-5N$	–	$(4N-3M)\lambda$	$(3N-2M)\lambda-1$	$(5N-3M)\lambda$	–
$(1.33, 1.49]$	$3M-4N$	$4M-5N$	$3M-5N$	$4M-6N$	–	$(3N-2M)\lambda-1$	$(5N-3M)\lambda$	$(4N-2M)\lambda-1$	–
1.5	$4M-5N$	$3M-5N$	$4M-6N$	–	–	$(5N-3M)\lambda$	$(4N-2M)\lambda-1$	–	–
$(1.55, 1.66]$	$4M-5N$	$3M-5N$	$4M-6N$	$3M-6N$	–	$(5N-3M)\lambda$	$(4N-2M)\lambda-1$	$(6N-3M)\lambda$	–
$(1.66, 1.75]$	$3M-5N$	$4M-6N$	$3M-6N$	$4M-7N$	–	$(4N-2M)\lambda-1$	$(6N-3M)\lambda$	$(5N-2M)\lambda-1$	–
$(1.75, 1.99]$	$3M-5N$	$4M-6N$	$3M-6N$	$4M-7N$	$3M-7N$	$(4N-2M)\lambda-1$	$(6N-3M)\lambda$	$(5N-2M)\lambda-1$	$(7N-3M)\lambda$
2	0	1	-1	0	–	3	4	7	–
$(2, 2.33]$	$3M-6N$	$3M-7N$	$3M-8N$	–	–	$(5N-2M)\lambda-1$	$(6N-2M)\lambda-1$	–	–
$(2.33, 2.49]$	$3M-6N$	$3M-7N$	$3M-8N$	$3M-9N$	–	$(5N-2M)\lambda-1$	$(6N-2M)\lambda-1$	$(7N-2M)\lambda-1$	–
$(2.49, 2.66]$	$3M-7N$	$3M-8N$	$3M-9N$	–	–	$(6N-2M)\lambda-1$	$(7N-2M)\lambda-1$	–	–
$(2.66, 2.99]$	$3M-7N$	$3M-8N$	$3M-9N$	$3M-10N$	–	$(6N-2M)\lambda-1$	$(7N-2M)\lambda-1$	$(8N-2M)\lambda-1$	–
3	1	0	-1	-2	–	3	7	11	–

Table 6.4: Fast four-phase synchronizer parameters for $\phi_{1,next}$ when $x \in (0.5, 3]$

data cycle times are the same in three cases. As was expected, in each case, the average data cycle time experiences large amount of fluctuations in the range $x \in (0.5, 2]$. At certain clock ratios, if there is a small change in x, significant changes in the average data cycle time may incur. For instance, on average, 4 extra transmitter clock cycles incur if the clock ratio increases from 1.5 to 1.67, and as soon as the clock ratio reaches 1.75, only one extra transmitter clock cycle is needed. So from 1.5 to 1.75, the average data cycle is sensitive to the change of the clock ratio. If for example, the receiver-to-transmitter clock ratio is designed to be 1.75, an average data cycle time of $7T_{tx}$ is expected. However, for some reason, the receiver clock becomes a little faster and the clock ratio changes to 1.67, the resultant data cycle time is approximately 1.4 times longer than the original expectation.

The fluctuations diminish when x is greater than 2 and the average data cycle time tends to be linear. The rate of the increase in the average data cycle times in the linear region is about $4T_{tx}$ per x. This means that if the clock ratio T_{rx}/T_{tx} increases from 2 to 3, on average, additional 4 transmitter clocks are needed to complete a data cycle.

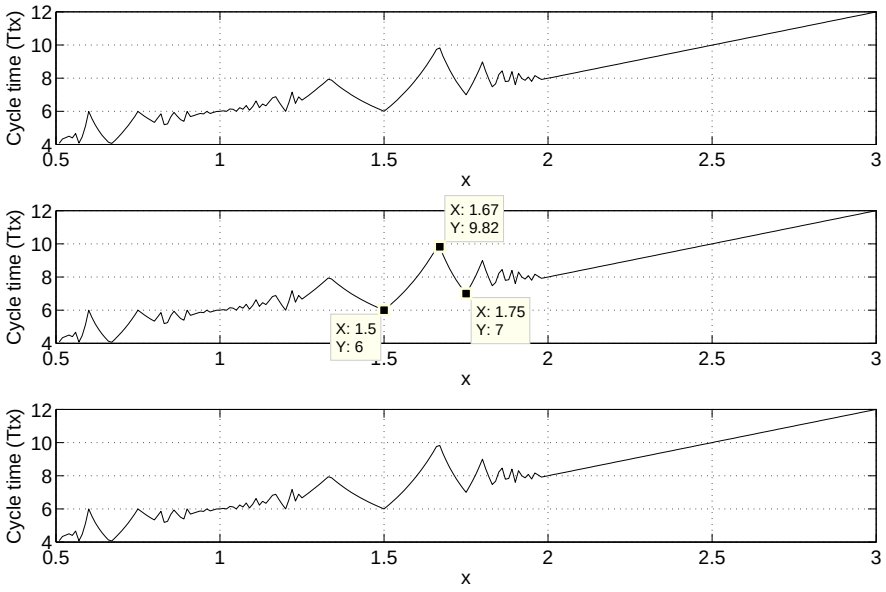


Figure 6.4: Average data cycle times for the two-flop synchronizers with data streams of 100 words (top), 500 words (middle) and 1000 words (bottom)

The average data cycle times under the same data-burst modes were also obtained for the fast four-phase and fast two-phase synchronizers. It was found that for each synchronizer, these three data streams produce similar average data cycle times, as was the case for the two-flop synchronizer. The average data cycle time for 1000 words stream will be used for the rest of this chapter, unless specified otherwise.

Comparisons of the average data cycle times are made among these three synchronizers and the plots are shown in Figure 6.5. Similar to the two-flop synchronizer, fluctuations in the average cycle times exist for both fast four-phase and fast two-phase synchronizer within the range $x \in (0.5, 2]$, and they diminish when x is beyond 2. In the region $x \in (0.5, 2]$, the average data cycle times behaviour for these synchronizers show some interesting phenomena:

1. All three plots exhibit similar sensitive regions. The sensitive region refers to the x regions where the amplitude of fluctuations in the average data cycle time reaches or exceeds T_{tx}. Each of the three plots shows two distinctive sensitive regions, which are $x \in [0.6, 0.75]$ and $x \in [1.33, 1.67]$. Between these two regions, fluctuations exist but with small amplitude. As was mentioned before, within the sensitive regions, the data cycle times may

incur large variations in amplitude with respect to a small shift in the clock ratio. This information however is hidden from the analytical estimation stated before and can not be acquired by cursory inspection of the data cycle time of a single data word transfer.

2. Similarities between the two-flop and fast two-phase synchronizers. The average data cycle times for these synchronizers show identical swing patterns but differ in the magnitude. For example, both plots reach a local peak at $x = 1.33$. They then drop to the bottom at $x = 1.5$ before reaching for the next peak at $x = 1.67$. The magnitude of the swing for the two-flop synchronizer is twice as much as that for the two-phase synchronizer. This observation also confirms the initial expectation where the fast four-phase synchronizer takes only half of the time of the two-flop synchronizer to complete one data cycle.

3. The complementary curve for the fast four-phase synchronizer. If the plots within the said two sensitive regions are compared between the two-flop and fast four-phase synchronizers, they show complementary patterns to each other. When the average data cycle time reaches a local maximum (minimum) for the two-flop synchronizer, the one for the fast four-phase gives a local minimum (maximum). At a certain clock ratio, for example $x = 1.5$, their average cycle times become identical. It implies that when the clock ratio is known to the designer and it happens to be one of those where the two-flop and the fast four-phase have the same average data cycle time, it would fail if the fast four-phase synchronizer was selected to achieve a shorter data cycle instead of the two-flop synchronizer. Due to the fact that their average data cycle times are identical at that clock ratio, performance improvement (if any) can be quite limited.

6.2.2.2 Effect of the starting phase on the average data cycle time

Previous discussions on the data burst average data cycle times were based on the same assumption: the starting phase of the first word is fixed and is set to the its minimum, which is small time step Δt. What happens if the starting phase of the first data word changes? How does it affect the average data cycle times? This section briefly discusses the influence of the first data word starting phase through several examples.

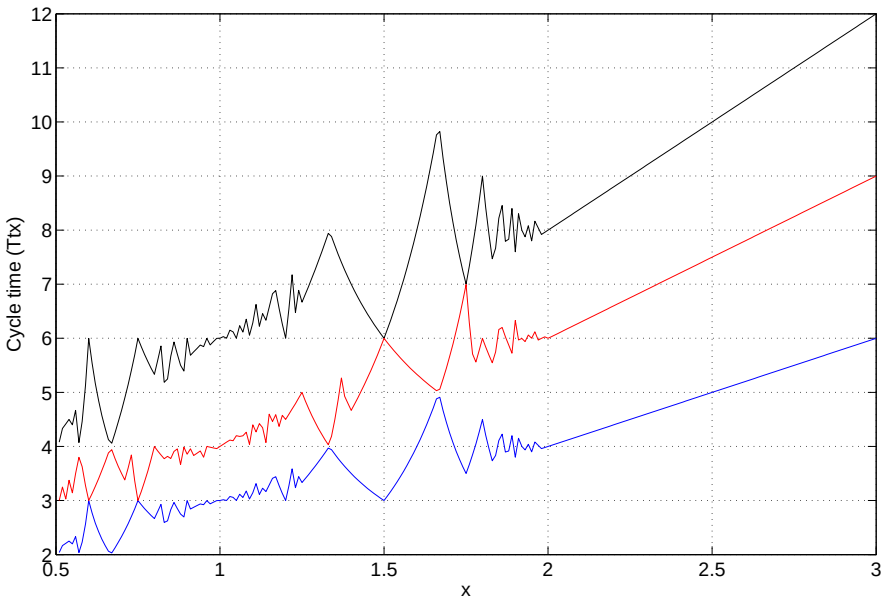


Figure 6.5: Average data cycle times of 1000 data words transfers for the two-flop(black), fast four-phase(red) and fast two-phase(blue) synchronizers

The average data cycle times for the two-flop synchronizer were calculated for two different starting phases, $\phi_1 = T_{rx}/2$ and $\phi_1 = T_{rx}$. They were then compared to the case with the minimal ϕ_1 as shown in Figure 6.6. At most of the clock ratios, the average data cycle times do not exhibit changes for different ϕ_1. However, significant performance degradation does incur at several clock ratios. When the clock ratio is 2, there is a sharp increase in the average data cycle time when ϕ_1 is set to $T_{rx}/2$ and T_{rx}. The average data cycle times in this two cases are $12T_{tx}$, as opposed to $8T_{tx}$ in the first case. Increase in the average data cycle time also occurs at $x = 1$ and $x = 1.5$ when ϕ_1 is set to T_{rx}. The reason is that at these clock ratios, for each word transfer, the worst case scenario always occurs. For example, at $x = 2$, the worst case data cycle time is $12T_{tx}$. The data cycle time for each word transfer turns out to be $12T_{tx}$ too and the starting phase of each word transfer is always T_{tx}, according to the simulation results. Obviously, the coincident clock edges affect every word transfer and produce the worst average data cycle times. Therefore, if the clock ratio is known to be one of the said values, i.e.,$x = 1, 1.5$ or 2, the first word starting phase should be carefully tuned to reduce the occurrence of the worst cases.

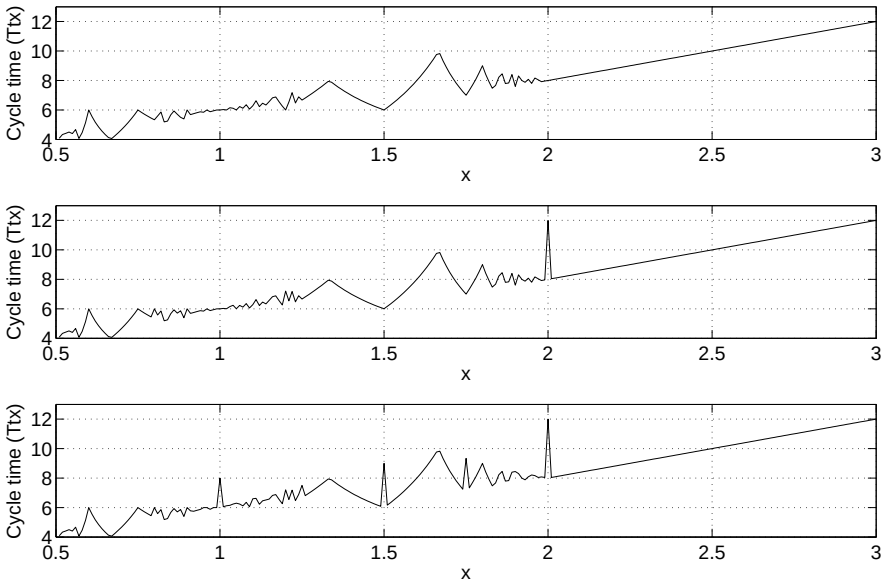


Figure 6.6: Average data cycle times with different first word ϕ_1 for the two-flop synchronizer: $\phi_1 = \Delta t$ (top),$\phi_1 = T_{rx}/2$ (middle) and $\phi_1 = T_{rx}$ (bottom)

6.3 Summary

This chapter described a data cycle time analysis tool, Synchronizer Data Cycle Analyzer (SyDCA), based on the conclusions obtained from previous chapters. This tool was built to analyze data cycle times under both single and burst data transfers for the two-flop, fast four-phase and fast two-phase synchronizers. The formation of the SyDCA tool was presented at the beginning of this chapter. It was then used to obtain the best, worst and average data cycle times for the said synchronizers. Strong non-linear behaviour in the two-flop average data cycle time was found. The effect of coincident clock edges at several clock ratios was also displayed in the average data cycle times for all three synchronizers. The estimated data cycle times from Chapter 3 were then compared with the actual ones and the quality of the estimation was assessed.

The SyDCA tool was then extended to deal with the burst-mode data transfers for the three synchronizers. Tests of 100, 500 and 1000 consecutive words were run in SyDCA and the resultant average data cycle times show negligible differences among them. The average data cycle times of 1000 data words were compared for the three synchronizers. Sensitive regions of the average data cycle times were

identified. Similarities were displayed between the two-flop and fast two-phase synchronizers while the average data cycle times between two-flop and fast four-phase synchronizers showed complementary patterns. Comparisons of the average data cycle times reveal important synchronizer characteristics under different clock ratios, which provides useful information for evaluation of their data cycle times. The influence of the starting phase of the first data word was then explored by comparing average data cycle times produced by different starting phase. It was found that the coincident clock edges caused performance penalty at certain clock ratios and suggestions were made to avoid these situations.

Chapter 7

Conclusions

7.1 Conclusions and summary of chapters

This text provided a systematic analysis of the data cycle times of three synchronizers: two-flop, fast four-phase and fast two-phase synchronizer. Timing boundary and analytical average data cycle time estimation provided a fast and moderately accurate way for the performance prediction. Models were developed for two clock domains with arbitrary ratios and ideal behavioural models for each synchronizer was constructed. With the help of these models, cycle time dependencies, including forward and backward dependency and dependency between adjacent data cycles, were analyzed and quantified. Abstractions of the single data cycle times were made from the simulation of the synchronizer models so that the data cycle times prediction was improved with greater accuracy. The Synchronizer Data Cycle Analyzer(SyDCA) tool was created to reflect these abstractions. This tool was then extended to perform analyses of the burst-mode data transfers. The average data cycle times obtained from SyDCA revealed complicated non-linear behaviour in data cycle times for both single and burst-mode data transfers.

Chapter 2 addressed the metastability issue in the synchronizer flip-flops. The focus was put on evaluations of the failure rate, Mean Time Between Failure(MTBF) for one flip-flop and many flip-flops. The second half of this chapter devoted to the descriptions of several synchronization circuits and their performances were briefly discussed.

Chapter 3 outlined the operations of three members of the two-flop synchronizer family: the two-flop, the fast-four phase and the two-phase synchronizers.

Timing boundaries were derived for each of the three synchronizers. The two-flop synchronizer has the identical forward and backward timing boundaries. The fast two-phase synchornizer also has identical forward cycle timing boundary to the two-flop synchronizer. The fast four-phase synchronizer's forward timing boundary is similar to the two-flop synchronizer's and its backward timing boundary is much tighter than its forward one. The timing boundaries defines the range of possible cycle times of each synchronizer. The best and worst case performances can be read off the plots of these boundaries for each clock ratio. Also, these timing boundaries provide information on rough estimation of data cycle times of each synchronizer.

This chapter also introduced the average data cycle times for the performance prediction of each synchronizer. The analytical average forward cycle times were derived for the three synchronizers. Preliminary exploration on the backward cycle time for the fast four-phase synchronizer was performed empirically. The dependency of the starting phase of the forward cycle on that of the backward cycle was found to cause non-linear behaviour of the backward cycle for both two-flop and fast four-flop synchronizers. This non-linearity was also found to influence the average data cycle times, especially when the clock frequencies are close to each other.

Chapter 4 presented the detailed analyses of the forward, backward and data cycle times for the two-flop synchronizer. The goal was to make predictions of the data cycle times. It addressed a behavioural model of the two-flop synchronizer. With the help of this model, the following two relationships were determined: (1) the dependency of the starting phases of the backward cycle on the forward cycle and (2)relationship between forward cycle starting phase and the forward cycle time. These two relationships were then combined to obtain the backward and data cycle times of the two-flop synchronizer. The analysis of the data cycle time revealed a pattern: shorter data cycle time can be achieved if the forward cycle starting phase is reduced. This observation later led to improvements of the two-flop synchronizer.

Chapter 5 extended the analysis of the two-flop synchronizer data cycle times to those of the fast four-phase and fast two-phase synchronizers. Observations of the data cycle times of the three synchronizers showed promising performance improvement with short starting phases. A new design approach, the three-flop synchronizer with halved transmitter and receiver clocks, was proposed based

on these observations. From the analyses of the three-flop synchronizer data cycle timing boundaries and two examples of the data cycle times, the three-flop design has demonstrated shorter data cycle times than the two-flop synchronizer but longer data cycle times than the fast four-phase synchronizer. The reliability of the three-flop design after frequency doubling is then analyzed and compared to the two-flop synchronizer. Although its $MTBF$ is shorter than the two-flop synchronizer, it still gives satisfactory reliability. For fast clock frequencies, more reliable flip-flop with shorter settling time constant should be used in both two-flop and three-flop designs to guarantee reliability.

Chapter 6 described a data cycle time analysis tool, Synchronizer Data Cycle Analyzer (SyDCA), based on the conclusions obtained from previous chapters. The SyDCA tool was then used to obtain the best, worst and average data cycle times for the three synchronizers. Strong non-linear behaviour in the two-flop average data cycle time was found. The effect of coincident clock edges at several clock ratios was also displayed in the average data cycle times for all three synchronizers. The estimated data cycle times from Chapter 3 were then compared with the actual ones and the quality of the estimation was assessed.

The SyDCA tool was then extended to deal with the burst-mode data transfers for the three synchronizers. It was found that the average data cycle times did not vary significantly with different number of data words in the data burst. Sensitive regions of the average data cycle times were identified. For the evaluation of the data cycle times, attention should be paid if the clock ratio of the transmitter and receiver falls in this region, as a small shift in clock ratio produces relatively large variations in the average data cycle times.

Similarities were displayed between the two-flop and fast two-phase synchronizers while the average data cycle times between two-flop and fast four-phase synchronizers showed complementary patterns. Comparisons of the average data cycle times reveal important synchronizer characteristics under different clock ratios, which provides useful information for evaluation of their data cycle times. The influence of the starting phase of the first data word was then explored by comparing average data cycle times produced by different starting phase. It was found that the cumulative effect of coincident clock edges aggregated performance penalty at certain clock ratios.

7.2 Future work

7.2.1 Possible improvement in the synchronizer models

The model developed in this text made several ideal assumptions and hence several practical features were neglected. The following features can be incorporated in the synchronizer models to improve the prediction accuracy.

1. `Asynchronous reset delay.` In the fast four-phase synchronizer model, the effect of asynchronous reset on the flip-flops were idealized. It was assumed a zero time reset. However, this may not be the case in reality. In Figure 3.5, the rise of A1 signal immediately brings down the REQ signal by asynchronously reset the REGV flop on the transmitter side. In the model, the fall of REQ happens simultaneously with the rise in A1, and therefore the REQ falls at the rising receiver clock edge. In reality, the asynchronous reset takes a finite time to respond and this means the de-assertion of REQ will happen after the rising receiver clock edge. If this delay is large enough to be neglected, the backward cycle time will no longer a integral multiple of the transmitter clock cycle any more. This will change the calculation of the timing boundary and data cycle time of the fast four-phase synchronizer. The consequent data cycle time may be slightly different.

2. `Effect of metastability.` All the synchronizer models do not include the metastability effect. One important part of the future work is to add this effect to the model. If the assumption that these synchronizers will resolve from metastable state in one clock cycle is made, it can be easily included in the models by creating an extra clock cycle delay. However, if the condition of getting into the metastable state is also to be reflected from the model, it may take some effort because the current model does not deal with the real flip-flop operations.

3. `Interconnection delay.` It is possible to incorporate this feature in the model if the interconnection delays between transmitter and receiver can be expressed as fractions of the transmitter clock cycle.

7.2.2 Extension in SyDCA

1. `Incorporation of the added features in the synchronizer models.` The added features in the synchronizer models directly affect the SyDCA tool calculation. So re-calculations of the cycle times are required if any of the features discussed above are added to the models.

2. `Calculation of synchronizer latencies.` As another important performance indicator, the latency of a synchronizer is equally important as the data cycle time. The author believes that the SyDCA tool can be extended to calculate the latencies of these synchronizers as long as they can be represented by the intervals between handshake signals.

3. `Analysis of the three-flop synchronizer` In the Chapter 2, the preliminary analyses were performed to the three-flop synchronizer. It is also possible to model its behaviour using the existing models and therefore the SyDCA tool can be extended to analyze its average data cycle times.

References

[BCV+05] E. Beigne, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin. An asynchronous noc architecture providing low latency service and multi-level design framework. In *Proceedings of ASYNC*, pages 54–63, 2005.

[BF02] J. Bainbridge and S. Furber. Chain:a delay-insensitive chip area interconnect. *IEEE Micro*, 22(5):16–23, 2002.

[BS05] Tobias Bjerregaard and Jen Sparsø. A scheduling discipline for latency and bandwidth guarantees in asynchronous network-on-chip. In *Proceedings of ASYNC*, pages 34–43, 2005.

[CG02] A. Chakraborty and M. R. Greenstreet. A minimalist source-synchronous interface. In *Proc. of ASIC/SOC Conference*, pages 443–447, 2002.

[CG03] A. Chakraborty and M. R. Greenstreet. Efficient self-timed interfaces for crossing clock domains. In *Proc. of ASYNC 2003*, pages 78–88, 2003.

[Cha84] D. M. Chapiro. *Globally-Asynchronous Locally-Synchronous Systems.* PhD thesis, Stanford University, 1984.

[CN01] T. Chelcea and S. M. Nowick. Robust interfaces for mixed-timing systems with application to latency insensitive protocols. In *Proc. 38th ACM/IEEE Design Automation Conference*, pages 21–26, 2001.

[DB99] Charles Dike and Edward Burton. Miller and noise effects in a synchronizing flip-flop. *IEEE Journal of Solid State Circuits*, 34 (6):849855, 1999.

[DDX95] L. R. Dennison, W. J. Dally, and D. Xanthopoulos. Low latency plesiochronous data retiming. In *Proc. 16th Anniversary Conference on Advanced Research in VLSI*, pages 304–315, 1995.

[DG07] Rastislav Dobkin and Ran Ginosar. Zero latency synchronizers using four and two phase protocols. Technical report, 2007.

[DG09a] Rostislav Dobkin and R. Ginosar. Fast universal synchronizers. In *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*, volume 5349/2009, pages 199–208. Springer Berlin / Heidelberg, 2009.

[DG09b] Rostislav Dobkin and Ran Ginosar. Two-phase synchronization with sub-cycle latency. In *Integration, the VLSi journal*, volume 42(3), pages 367–375. Elsevier, June 2009.

[DP98] William Dally and John Poulton. *Digital Systems Engineering*. Cambridge University Press, 1998.

[DVFG05] R. Dobkin, V. Vishnyakov, E. Friedman, and R. Ginosar. An asynchronous router for mutiple service levels networks on chip. In *Proceedings ASYNC*, pages 44–53, 2005.

[FF04] T. Felicijan and S. B. Furber. An asynchronous on-chip network router with quality-of-service(qos) support. In *Proceedings of IEEE International SOC Conference*, pages 274–277, 2004.

[FKG06] U. Frank, T. Kapschitz, and R. Ginosar. A predictive synchronizer for periodic clock domains. *Formal Methods in System Design*, 28(2):171–186, 2006.

[Fri01] Eby G. Friedman. Clock distribution networks in synchronous digital integrated circuits. In *Proc. IEEE*, pages 665–692, 2001.

[Gin03] R. Ginosar. Fourteen ways to fool your synchronizer. In *Proceedings of the Ninth International Symposium on Asynchronous Circuits and Systems*, pages 89–96, 2003.

[Gin08] R. Ginosar. Synchronization circuits for multi-clock domain soc. International Symposium on Circuits and Systems, May 2008. Tutorial.

[Gre93] M. R. Greenstreet. *STARI: A Technique for High-Bandwidth Communication.* PhD thesis, Department of Computer Science, Princeton University, 1993.

[Gre95] M. R. Greenstreet. Implementing a stari chip. In *Proc. 1995 International Conference on Computer Design*, pages 38–48, Austin, Texas, October 1995.

[IM02] A. Iyer and D. Marculescu. Power-performance evaluation of globally asynchronous, locally synchronous processors. In *Proc. 29th International Symposium on Computer Architecture*, pages 158–168, 2002.

[KG98] R. Kol and R. Ginosar. Adaptive synchronization. In *Proceedings of the IEEE International Conference on Computer Design*, page 188. IEEE Computer Society, 1998.

[Kin07] David J. Kinniment. *Synchronization and Arbitration.* Wiley, 2007.

[KPWK02] J. Kessels, A. Peelers, P. Wielage, and S. Kim. Clock synchronization through handshaking. In *Proceedings of the Eighth International Symposium on Asynchronus Circuits and Systems*, page 59, 2002.

[MC80] C. A. Mead and L. A. Conway. *Introduction to VLSI Systems.* Addison-Wesley, 1980.

[Men91] Teresa Meng. *Synchronization Design for Digital Systems.* Kluwer Academic Publishers, 1991.

[Mes90] D. Messerschmitt. Synchronization in digital system design. *IEEE Journal on Selected Areas in Communications*, 8:1404–1419, 1990.

[MM07] R. Mullins and S. Moore. Demystifying data-driven and pausible clocking schemes. In *Proc. ASYNC 2007*, pages 175–185, 2007.

[MN06] Alain Martin and M. Nyström. Asynchronous techniques for system-on-chip design. In *Proceedings of the IEEE*, volume 94, pages 1089–1120, 2006.

[MTMR02] S. Moor, G. Taylor, R. Mullins, and P. Robinson. Point to point gals interconnect. In *Proc. ASYNC 2002*, pages 69–75, 2002.

[NM02] M. Nyström and A. J. Martin. Crossing the synchronous-asynchronous divide. In *Workshop on Complexity-Effective Design*, 2002.

[NNSvB94] L. S. Nielsen, C. Niessen, J. Sparso, and C.H. van Berkel. Low-power operation using self-timed and adaptive scaling fo the supply voltage. In *IEEE Transactions on VLSI Systems*, volume 2(4), pages 391–397, 1994.

[SAD+02] G. Semeraro, D. H. Albonesi, S. G. Dropsho, G. Magklis, S. Dwar4kadas, and M. L. Scott. Dynamic frequency and voltage control for a multiple clock domain microarchitecture. In *IEEE/ACM International Symposium on Microarchitecture*, pages 356–367, 2002.

[SG03] Y. Semiat and R. Ginosar. Timing measurements of synchornization circuits. In *Proceedings of Ninth International Symposium on Asynchronous Circuits and Systems*, page 68, 2003.

[SM00] A. E. Sjogren and C. J. Myers. Interfacing synchronous and asynchronous modules with a high-speed pipeline. *IEEE Transactions on VLSI Systems*, 8(5):573–583, 2000.

[WH04] Neil Weste and David Harris. *CMOS VLSI Design: A circuit and systems perspective*. Pearson Addison-Wesley, 3rd edition, 2004.

[YD96] K. Yun and R. Donohue. Pausible clocking: a first step toward heterogeneous systems. In *Proc. Computer Design: VLSI in Computers and Processors*, pages 118–123, 1996.

[YD99] K. Yun and A. Dooply. Pausible clocking based heterogeneous systems. *IEEE Transactons VLSI Systems*, 7(4):482–487, 1999.