Adapting Synchronizers to the Effects of On Chip Variability

Jun Zhou, David Kinniment, Gordon Russell, and Alex Yakovlev

Newcastle University, UK

Abstract

Two adaptation schemes based on on-chip measurement of failure rates have been proposed to reduce the effects of process, voltage, temperature and data rate variations on synchronizers on chip. One scheme is to select the best synchronizer out of a number to improve the synchronizer performance subject to process variation on chip. Compared to increasing the transistor size, this scheme can further reduce the effects of process variation without increasing the power consumption. The other scheme is to improve the performance of the system by adjusting the synchronization time according to the actual process, voltage, temperature and data rate variations on the condition that the required MTBF is met. It is targeted at overdesigned synchronization times due to synchronizer performance variability. To assess their feasibility, the two schemes have been implemented using a Xilinx’s 90nm FPGA Spartan 3. The on-chip overhead for the Synchronizer Selection scheme in terms of equivalent flipflops and gates is 9 and 6. For the Synchronization Time Adjustment scheme it is 33 and 104.

1. Introduction

Process, voltage, and temperature variations in nanometer technologies can be an important limit on the performance of systems on silicon. Components such as logic circuits, memories on chip are all affected, but the performance of synchronizers which are used to synchronize data passing between different clock regions in future SoCs may affect the system performance to a greater extent than other components for three reasons:

1. Vth is a major source of circuit parameter variability, and synchronizers are usually more strongly affected by Vth variation than logic circuits.
2. Synchronization time is usually one or more clock cycles in duration, and is directly affected by the synchronization time constant, whereas there are many logic gate delays in a clock cycle, allowing random variations in each gate within a critical path of gates to be subject to averaging.
3. In future SoCs, communication is likely to affect the system performance more than processing [1], and synchronization is an large overhead in a communication link.

A future multi-core system can incorporate as many as 100 synchronizers on the same chip. As part of communication, the synchronizers’ performance is critical to the system performance. The impact of process variation on circuit performance has been discussed in [2][3][4]. At 180nm, we can expect the standard deviation (σ) of the metastability time constant τ which determines the resolution time of metastability in synchronizers to be about 5% [2]. So one synchronizer out of 1000 may have a 15% worse value of τ. We measured τ in a batch of synchronizers chips fabricated on the UMC 180nm process and found a variability consistent with the results given in [2]. At 90nm σ is about 8% [2], so we can expect one synchronizer out of the 1000 to have a 24% worse value of τ. As the technology continues to advance, the impact of process variation on circuit performance becomes more and more severe. According to ITRS 2006 [4], at 45nm the circuit performance variability reaches to 50%. In addition to the process variation, power supplies and temperature variations disproportionately affect the synchronization time constant τ, since τ depends on the small signal parameters in metastability rather than large signal switching times [5]. As a result a 50% reduction in power supply voltage may result in over 100% increase in τ [6]. Additionally, for a synchronizer, the average data rate between different clock regions in systems on chip may vary over time. Hence the mean time between failures (MTBF) of the synchronizer which is determined by the data rate may vary over time [5].

Recently adaptive circuits have been used to mitigate the effect of process variation in microprocessors design [7]. In this paper we proposed two adaptation schemes to reduce effects of process, voltage, temperature and data rate variations on synchronizers on chip. One scheme is aimed at
improving synchronizer performance subject to process variation. Current practice to reduce the effects of process variation is to make the transistors in the synchronizer wider than normal so that the deviation is reduced. The main disadvantage of this technique is that it uses a significant proportion of the system power budget because the current is also increased. An alternative is to make a number of synchronizers with normal size and select the best one. After selection, all the others are powered down, as is the selection circuitry. Power during operation is therefore the same as for a single synchronizer, and the performance may even be improved.

The other scheme is to improve the system performance by adjusting the synchronization time according to the actual process, voltage, temperature and data rate variations on the condition that the required MTBF is met. This is targeted at overdesigned synchronization times due to synchronizer performance variability. For example, for a synchronizer in a multicore system, due to the variations discussed before, extra synchronization time is required to ensure that the synchronizer works in the worst case. The multicore system may incorporate 100 synchronizers on the same chip. Because it is not known which synchronizer will give the worst case all the synchronizer times on the chip need to be extended to ensure the system works in the worst case. However, the actual amount of the variations for some of the synchronizers may not be as great as the worst case. With this scheme, the synchronization time of each synchronizer on the chip can be adjusted according to the actual process, voltage, temperature and data rate variations to improve the performance of the system on the condition that the required MTBF is still met.

Both adaptation schemes proposed in this paper rely on the on-chip measurement of failure rates in individual synchronizers, followed in the first case by selection to reduce the effect of the process variation on synchronizer performance, and in the second case by adjustment of the synchronization time according to the actual variations to gain improvement in the system performance. The rest of the paper is organized as follows. In Section 2 the on-chip measurement of failure rates is described. In Section 3 the calculation of \( \tau \) and MTBF from failure rates is described. In Section 4 the two proposed adaptation schemes are described. In Section 5 the implementation details of the two adaptation schemes are described. In Section 6 the applications of the two adaptation schemes are discussed. In Section 7 the test results are presented and in Section 8 conclusions are drawn.

2. On-chip Measurement of Failure Rates

Figure 1 shows the on-chip measurement of failure rates. Here FF1 and FF2 sample the output of the synchronizer at two different times SCLK+T1 and SCLK+T2 (T1<T2). Their outputs are XORed with the output of FF3 which sample the output of synchronizer at the falling edge of SCLK. Assuming that there is a very high probability that metastability resolves before the falling edge of SCLK, if the output of synchronizer is in metastability at the sampling time SCLK+T1 or SCLK+T2, the output of FF1 or FF2 will have different value to the output of FF3, so the output of XOR will go high. This high output will then be clocked into FF5 or FF6 at the next SCLK+T1 time, which indicates that a failure has been detected. Here Counter1 and Counter2 are used to count the number of failures at the two sampling times, and Counter3 is used to count the number of clock cycles. When Counter1 reaches a preset value (say 200) all the three counters will be stopped counting.

![Figure 1. On-chip measurement of failure rates](image)

3. Calculation of \( \tau \) and MTBF

In order to be sure that the value of \( \tau \) measured by this technique is the same as the long term \( \tau \), we must ensure that any initial transients leading to variation in \( \tau \) have died away. We believe that this is usually the case for simple latch circuits at the sampling times we took. From the failure rates measured \( \tau \) and MTBF can be calculated.

3.1. Calculate \( \tau \) from Measured Failure Rates

The parameter \( \tau \) is the metastability time constant which determines the resolution time of metastability. It can be calculated from the measured failure rates using the MTBF formula below, where \( t \) is the synchronization time, \( Tw \) is the metastability window, \( fc \) is the clock frequency and \( fd \) is the data rate.
MTBF1 and MTBF2 are obtained by substituting t with T1 and T2 in the formula of MTBF.

\[\text{MTBF} = \frac{e^T}{T_{\text{rate}} f_{\text{rate}}}\]

\[\text{MTBF2} = e^{T_2 - T_1}\]

\[\frac{\text{Failure Rate}_1}{\text{Failure Rate}_2} = e^{T_2 - T_1}\]

\[\tau = \frac{T_2 - T_1}{\ln \left( \frac{\text{Failure Rate}_1}{\text{Failure Rate}_2} \right)}\]

### 3.2. Calculate MTBF from Measured Failure Rates

After the calculation of \(\tau\), the long term MTBF corresponding to the current synchronization time can be calculated using the formula below, where T3 is the current synchronization time and T1 is the earlier one of the two sampling times.

\[\text{MTBF}_3 = \text{MTBF}_1 \ast e^{\frac{T_3 - T_1}{\tau}}\]

However, MTBF3 is usually very large compared to MTBF1 (say 10\(^9\) times bigger), and the calculation would require floating point hardware. To reduce the hardware overhead for estimating MTBF3 we need to convert it to a simpler fixed point calculation.

\[\text{MTBF}_3 = \frac{\text{Counter}_3 \ast \text{output}}{\text{Clock period}} \ast e^{\frac{T_3 - T_1}{\tau}}\]

\[\frac{\text{Counter}_1 \ast \text{output} \ast \text{MTBF}_3}{\text{Clock period}} = \text{Counter}_3 \ast \text{output} \ast e^{\frac{T_3 - T_1}{\tau}}\]

Let \(X = \ln \left( \frac{\text{Counter}_3 \ast \text{output}}{\text{Clock period}} \right)\), \(Y = \ln \left( \text{Counter}_3 \ast \text{output} \right)\)

\[e^{X} = e^{Y} \ast e^{\frac{T_3 - T_1}{\tau}}\]

\[X = Y + \frac{T_3 - T_1}{\tau}\]

Now we only need to calculate X instead of real MTBF, and the required MTBF can also be converted to required X in the same way for later comparison.

### 4. Two Adaptation Schemes

#### 4.1. Synchronizer Selection Scheme

The synchronizer selection scheme is used to reduce the effects of process variation by selecting the synchronizer with the best performance from a number of redundant synchronizers. In the future a multi-core design can incorporate as many as 100 synchronizers on the same chip. Their performance is critical to the system performance. Let us assume that we have a synchronizer with a \(\tau\) of 11 ps, and a standard deviation, \(\sigma\), of 8% on 90nm technology. Assuming that the variability is completely random, then in the worst case we must allow for a 3.09\(\sigma\) to ensure that the probability of a synchronizer having \(\tau\) worse than this is 0.001. This means that the synchronization time must be set to allow for a \(\tau\) of 11 + 11\times 0.08 \times 3.09 = 13.72\) ps. The usual solution to this is to make the width of all transistors in the synchronizer N times larger (say N=4 here). Assuming this reduces most of the process variations and the deviation is now \(\sigma = \frac{8\%}{\sqrt{4}} = 4\%\). Now the worst case is 12.36 ps, but the power is increased by four times.

We propose to make N standard size synchronizers, measure their \(\tau\) on chip, and select the best one. After the selection, all the others are powered down, as is the measurement circuitry. Power during operation is therefore the same as for a single small synchronizer, but the performance is improved. The probability of one synchronizer having \(\tau\) worse than 11.81 ps is 17.8\%, but the probability of all four synchronizers having \(\tau\) worse than this is 0.178\(^4\), or 0.001. Thus we have achieved a small worst case improvement from 12.36 ps to 11.81 ps for \(\tau\), but a significant power saving. Normally a synchronization time of 40 \(\tau\) is required to give a MTBF of 4 months, so the improvement in synchronization time is about 22 ps.

These statistics assume that the variability is completely random over the four synchronizers. This is unlikely to be the case. There will be some correlation between circuits, but note that this correlation will be greater for the transistors in the large synchronizer, because the increase in size is located within a small area, so the selection technique will always give at least as good a result as the simple method of increasing transistor size. In addition, enlarging synchronizer size can not reduce all kinds of process variations, but only some of them. For example, it has no effect on variation in gate insulator thickness. The selection technique is used to deal with all kinds of process variations. We think it is a good way to improve the synchronizer performance.

#### 4.2. Synchronization Time Adjustment Scheme

The synchronization time adjustment scheme is used to improve the performance of the system by adjusting the synchronization time according to the actual process, voltage, temperature and data rate variations on the condition that the required MTBF is met. Table
1 shows the variation of the metastability time constant $\tau$ with Vdd and temperature variations for Jamb latch which is commonly used as a synchronizer because of its good performance and simple structure [8]. The results are obtained by extensive SPICE simulation using UMC 90nm technology.

Table 1 Jamb latch $\tau$ vs Vdd

<table>
<thead>
<tr>
<th>Vdd(vs)</th>
<th>$\tau$ (ps) at 27 °C</th>
<th>$\tau$ (ps) at -25 °C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2</td>
<td>11.47</td>
<td>9.23</td>
</tr>
<tr>
<td>1.1</td>
<td>12.19</td>
<td>10.24</td>
</tr>
<tr>
<td>1.0</td>
<td>13.67</td>
<td>12.06</td>
</tr>
<tr>
<td>0.9</td>
<td>15.46</td>
<td>14.28</td>
</tr>
<tr>
<td>0.8</td>
<td>19.64</td>
<td>18.66</td>
</tr>
<tr>
<td>0.7</td>
<td>30.71</td>
<td>36.33</td>
</tr>
<tr>
<td>0.6</td>
<td>60.55</td>
<td>97.81</td>
</tr>
<tr>
<td>0.5</td>
<td>159.45</td>
<td>338.43</td>
</tr>
<tr>
<td>0.4</td>
<td>525.82</td>
<td>1403.76</td>
</tr>
<tr>
<td>0.3</td>
<td>2742.56</td>
<td>8151.86</td>
</tr>
</tbody>
</table>

It can be seen from Table 1 that $\tau$ increases rapidly with Vdd decreasing. The value of $\tau$ is more than doubled at a Vdd of 0.7V, and more than an order of magnitude higher at 0.5V. Below 0.5V $\tau$ increases more rapidly with Vdd decreasing and lowering the temperature makes this situation worse. In other words, variations in Vdd, Vth and temperature could make the synchronizer unviable, especially for deep submicron processes.

In addition, the average data rate between different clock regions in a multi-clock system may vary over time. Hence the MTBF of the synchronizer which is determined by the data rate may vary over time.

Due to the variations discussed above, extra synchronization time is required to ensure that the system works in the worse case. For example, in a multi-core system which incorporates 100 synchronizers on the same chip, if $\tau$ can increase by 25% because of Vdd and temperature variation and a further 25% because of process variation, in the worse case synchronizer we will have an over 50% worse value of $\tau$. In order to achieve the required MTBF we have to extend the synchronization time of that synchronizer to over 1.5 times its original value, and because it is not known which synchronizer will give the worse case all the synchronizer times on the chip need to be extended to over 1.5 times their original values. However, the actual amount of the variations for some of the synchronizers may be less than the worst case. With synchronization time adjustment scheme, the synchronization time of each synchronizer can be adjusted according to the actual process, voltage, temperature and data rate variations to improve the performance of the system on the condition that the required MTBF is still met. For example, in above case, for the synchronizers which have nominal values of $\tau$, the synchronization time can be reduced by 33%. As a result the system performance is greatly improved.

Note that a Jamb latch is not necessarily the best synchronizer. Developing transistor level techniques for more robust synchronizer [6] can be a way to improve the synchronizer’s performance as well as reducing its sensitivity to Vdd and temperature variations, but all synchronizers exhibit variability, and the synchronizer’s performance can be further enhanced by adaptability.

5. Implementation

To assess their feasibility, the two adaptation schemes proposed have been implemented using a Xilinx’s 90nm FPGA Spartan 3.

5.1. Architecture of Synchronizer Selection Scheme

The synchronizer selection scheme is based on comparison of $\tau$. In Figure 1 counter1 is used to count the number of failure rates at the earlier sampling time SCLK+T1. When it reaches a preset value (say 200) both counter1 and counter2 will be stopped counting. Because $T2-T1 = \frac{\ln \text{Failure Rate1}}{\text{Failure Rate2}}$, and because $T2-T1$ and Failure_rate1 are constants, instead of comparing $\tau$, the selection can be done by directly comparing Failure_rate2. The smaller the Failure_rate2 is, the smaller the $\tau$ is. In this way we can avoid division and log calculations so that the implementation can be greatly simplified. Figure 2 shows the architecture of the synchronizer selection scheme. It consists of an on-chip part and an off-chip part. The on-chip part is per synchronizer. It includes N redundant synchronizers and the failure detector. The failure detector is used to detect failures and has to be placed on chip with the synchronizers to ensure that the measurement is accurate. Each failure detector is shared by the N redundant synchronizers from which the best synchronizer is to be selected. The off-chip part is shared by all the synchronizers on the chip. It can be put off chip because the selection scheme is used to deal with the process variation and only needs to operate once when setting up the chip. In this way the on-chip overhead is reduced. The off-chip part includes the failure counter, FIFO and comparator. The failure counters are used to count the number of the failures.
After that the values from counter2 are stored in a FIFO for comparison and then the best synchronizer is selected. After that all the other synchronizers are powered down as is selection circuitry so the power consumption is the same as for a single synchronizer.

Figure 2. Architecture of Synchronizer Selection Scheme

In the FPGA implementation of the synchronizer selection scheme, the on-chip overhead per synchronizer in terms of equivalent flipflops and gates is 9 and 6. The total off-chip overhead in terms of equivalent flipflops and gates is 34 and 110. It is possible to put this scheme entirely on chip since the off-chip overhead is not very big.

5.2. Architecture of Synchronization Time Adjustment Scheme

The synchronization time adjustment scheme is based on a calculation of MTBF. As shown in Figure 3, it also consists of an on-chip part and an off-chip part. The on-chip part includes a variable delay line (VDL), the registers and the failure detector. The VDL is used to control the synchronization time of the synchronizer and the registers are used to hold the delay of the VDL. Again, the on-chip part is per synchronizer and the off-chip part can be shared by all the synchronizers on the chip. This way the on-chip overhead is reduced. From the failure rates $\tau$ and MTBF is calculated. After that the calculated MTBF is compared with the user-required MTBF and then the synchronization time is adjusted to give the required value. After some iteration, the MTBF of the synchronizer will stabilize close to the user-required MTBF.

The synchronization time adjustment scheme can work in two modes.

a. Self-adjusting Mode: in this mode users need to input the required MTBF. The adjusting circuits will detect the failure rates and calculate the MTBF that would be given by the current synchronization time. This estimated MTBF is then compared to the user-required MTBF and the synchronization time is adjusted to give the required value. After some iteration, the MTBF of the synchronizer will stabilize close to the user-required MTBF.

b. User Mode: in this mode the adjusting circuits will detect failure rates, calculate the MTBF corresponding to the current synchronization time, and output the estimated MTBF for users to make any adjustment needed such as changing Vdd or clock frequency to meet the required MTBF. This mode is mainly used as a means for users to monitor the MTBF of the system and make any adjustment needed.

In both modes users needs to input the clock frequency used for the calculation of MTBF.

5.3. Failure Detector

The failure detector is used to detect the failure at two different sampling times of the output of synchronizer. It has to be put close to the synchronizer to ensure that the measurement is accurate. As shown in Figure 1, the synchronizer is clocked by the signal SCLK which is generated from the local clock signal CLK. The synchronization time is defined as the time from the rising edge of SCLK to the rising edge of CLK and is controlled by the variable delay line. FF1 and FF2 sample the output of synchronizer at two different times. Their outputs are XORed with the output of FF3 which samples the output of hardware. For example, an FPGA based variable delay line consisting 20 four-input lookup tables or 40 equivalent gates would need only 12 transistors or 3 equivalent gates when implemented on chip. This on-chip overhead can be reduced by 50% when implemented on chip. The off-chip overhead can also be reduced by making a trade off between the calculation accuracy and the hardware complexity. This is discussed in section 5.8.
synchronizer at the falling edge of SCLK as described in section 2. In the FPGA implementation the time between T1 and T2 is 100 ps which is achieved by using the connection delay difference.

5.4. Failure Counters

The failure counters count the number of failures detected at different sampling times. As shown in Figure 4, it consists of three counters. Counter1 and Counter2 are used to count the number of failures at the sampling times SCLK+T1 and SCLK+T2 (T1<T2). Counter3 is used to count the number of clock cycles. When Counter1 reaches a preset value it will send a stop signal to the control logic and then all the three counters will be stopped counting. Then the values stored in the counters will be used for the calculation of $\tau$ and MTBF. Note that for the synchronizer selection scheme the counter3 is not needed so the hardware overhead can be further reduced.

5.5. Synchronizer Selection Circuit

Figure 5 shows the synchronizer selection circuit. Here four P-type transistors are used to switch the power for the four synchronizers. After the best synchronizer is selected, the other three synchronizers are powered down as is selection circuitry so the power consumption is the same as for a single synchronizer. An OR gate is used to generate the output since all the powered down synchronizers have low outputs. Simulation on 90nm technology shows that the delay of the OR gate is about 18 ps. Considering that the improvement in synchronization time is 22 ps as mentioned in Section 4.1, the synchronizer selection scheme is at least as good as the synchronizer size enlargement scheme.

5.6. Variable Delay Line

Variable delay lines are usually implemented by transistor level circuits. However, in FPGAs they can only be implemented as inverter chains. Another problem is that in FPGAs inverters are implemented by using lookup tables [9]. In the device we used (Xilinx Spartan 3), the delay of lookup table plus wire delay is more than 1 ns, which is too big for the incremental delay considering that the metastability constant $\tau$ of a synchronizer has a typical value of 11 ps on 90nm technology. However, a smaller incremental delay can be achieved by using the connection delay difference on FPGA. As shown in Figure 6, by carefully placing the internal XOR gates we can get an incremental delay which is the difference between the connection delay in two neighbouring paths down to 100 ps [9]. Note that FPGA is used here simply to demonstrate the feasibility of the proposed schemes. Our eventual aim is to implement the schemes on chip. With a variable delay line implemented on chip an incremental delay of 1 ps can be easily achieved.

5.7. Implementation of $\tau$ and MTBF Calculation
5.7.1. Calculation Flow

Figure 7 shows the τ and MTBF calculation flow. Here F is τ and X is our MTBF measure.

\[ A = \text{MTBF}, \ B = \text{MTBF}, \ E = T2 - T1, \ G = T3 - T1, \ J = \text{Counter}_3_{\text{output}} \]

\[ Y = \frac{F}{E}, Y = \frac{F}{G}, Y = \frac{X}{J} \]

Figure 7. Calculation flow

5.7.2. Implementation of Division

As can be seen in Figure 7, the algorithm for calculating X contains three divisions. The divider required can be reused by multiplexing its inputs. Figure 8 shows the implementation details of the divider. A pipelined divider is used to achieve high performance and low area. Its divisor and dividend inputs are multiplexed to make it reusable. A counter is used to count the number of clock cycles used to do the division. When the counter reaches a preset value, it will send a completion signal to the control logic and then the divider will be disabled. Then the calculation will move to the next step. The output of the divider is stored in a register because this output will be used as input of the divider in later step.

Figure 8. Divider

5.7.3. Implementation of Log Calculation

To calculate X the log calculation needs to be done twice. Like the divider, the log calculation circuit can be reused by multiplexing its inputs. As shown in Figure 9, log calculation is done by using lookup tables. Because the value that needs to be calculated can be very large (up to \( 10^{10} \), which is from the output of counter3), it is impossible to build a full log lookup table. However, considering that the log curve is non-linear, different resolutions can be used for calculating different values (high resolution for small values and low resolution for large values). Here three lookup tables with different resolution are used to provide an accuracy of two decimals, which leads to an error of 1% in the calculated MTBF. For example, if the calculated MTBF is 10 years, the calculation error is only about 1 month. The memory used to implement the lookup tables is 250K Bytes, which can be reduced to less than 25K by increasing the calculation error to 10% which is still acceptable in the calculation of MTBF.

Figure 9. Log calculation circuit

5.8. Hardware Saving

Compared to the synchronizer selection scheme, the synchronization time adjustment scheme consumes relatively large amount of hardware. However, 80% of its on-chip overhead is caused by implementing the variable delay line on FPGA. When implemented on chip using transistors it will consume much less hardware and we expect that the on-chip overhead will be reduced by 50%. The off-chip part of the synchronization time adjustment scheme which is used to calculate τ and MTBF can be also reduced by using only the most significant bit of the values from the counters to do the division. In addition, the memory used for implementing log calculation can be reduced from 250KB to 25KB by increasing the calculation error from 1% to 10% which is acceptable in the calculation of MTBF. A trade off can thus be made between the calculation accuracy and the hardware complexity.

6 Applications of Two Schemes

The synchronizer selection scheme is aimed at improving synchronizer performance subject to process variation. It only needs to operate once when setting up the chip since the process variation is fixed when the chip is fabricated. After the best synchronizer is selected all the other redundant synchronizers are powered down as is selection circuitry so the power
consumption is the same as for a single synchronizer. This scheme has a small overhead and can be entirely put on chip.

The synchronization time adjustment scheme is used to deal with the process, voltage, temperature and data rate variations. It consumes relatively large amount of power and hardware. However, when used to deal with the process variation or fixed Vdd variation it only needs to operate once when setting up the chip like the synchronizer selection scheme. After that all the adjusting circuits can be powered down to reduce the power consumption. Also, without the need to track frequent variations, most of the adjusting circuits can be put off chip to reduce the on-chip overhead. When used to deal with frequent Vdd variation or data rate variation, the scheme needs to be put entirely on chip and operate frequently. It is possible, however, to reduce the power consumption by making the adjustment relatively rare and reduce the hardware complexity by using the methods discussed in Section 5.8.

7. Test Results

7.1. Calculated MTBF vs Data Rate

Figure 10 shows the calculated MTBF against the data rate for a synchronization time of 3.5 ns. The calculated MTBF decreases with the data rate increasing as expected, showing that the synchronization time could be reduced for data rates below 4MHz.

7.2. Calculated MTBF vs Synchronization Time

Figure 11 shows the calculated MTBF against the synchronization time for the data rate of 5 MHz. The calculated MTBF increases with the synchronization time increasing as expected.

7.3. Calculated Tau vs Vdd

Figure 12 shows the calculated Tau against Vdd. The calculated Tau increases with Vdd decreasing as expected.

8. Conclusions

Two adaptation schemes based on on-chip measurement of failure rates have been proposed to reduce the effects of process, voltage, temperature and data rate variations on synchronizers on chip. One scheme is to select the best synchronizer out of a number to improve the synchronizer performance subject to process variation on chip. Compared to simply increasing the transistor size, this scheme can further reduce the effects of process variation without increasing the power consumption.

The other scheme is targeted at overdesigned synchronization times due to synchronizer performance variability. It is used to improve the system performance by adjusting the synchronization time according to the actual process, voltage, temperature and data rate variations on the condition that the required MTBF is met. Assuming that the metastability
time constant \( \tau \) which determines the resolution speed of metastability in synchronizers can increase by 25% due to process variation and a further 25% due to Vdd and temperature variations, this scheme can improve the performance of the system by 33%.

To assess their feasibility, the two schemes have been implemented using a Xilinx’s 90nm FPGA Spartan 3. The synchronizer selection scheme is simple and consumes small amount of hardware (9 flipflops and 6 gates per synchronizer for the on-chip part, and 34 flipflops and 110 gates for the off-chip part). This scheme can be entirely put on chip since the off-chip overhead is not very big. Because the synchronizer selection scheme is used to deal with the process variation, it only needs to operate once when setting up the chip. After the best synchronizer is selected all other circuits including the redundant synchronizers are powered down as is selection circuitry so the power consumption is the same as for a single synchronizer.

The synchronization time adjustment scheme consumes relatively large amount of power and hardware (33 flipflops and 104 gates per synchronizer for the on-chip part, and 436 flipflops and 732 gates for the off-chip part). However, when used to deal with the process variation or fixed Vdd variation it only needs to operate once when setting up the chip like the synchronizer selection scheme. After that all the adjusting circuits can be powered down to reduce the power consumption. Also, without the need to track frequent variations, most of the adjusting circuits can be put off chip to reduce the on chip overhead. Only when used to deal with frequent Vdd variation or data rate variation, the scheme needs to be put entirely on chip and operate frequently. It is possible, however, to reduce the power consumption by making the adjustment relatively rare and reduce the hardware complexity by sacrificing the calculation accuracy a little.

10. References


