HIGH SPEED INTERFACE FOR SYSTEM-ON-CHIP DESIGN
BY SELF-TESTED SELF-SYNCHRONIZATION

Fenghao Mu1 and Christer Svensson2
1 SwitchCore, S-22363, Lund, Sweden
2 IFM, Linköping University, S-58183 Linköping, Sweden
e-mail: fenghao@ifm.liu.se

Abstract—Global synchronization has been commonly used to protect clocked I/O from data read failure due to metastability. For future high performance system-on-chip design, global synchronization is more difficult as both frequency and chip size increase quickly. This paper addresses a mesochronous clocking (MC) strategy which can be implemented with three self-tested self-synchronization (STSS) methods for parallel data transfer between processing elements (PEs). Compared with global synchronization, MC has many advantages: lower process cost; less power dissipation in clock distribution; no limit in system scale; less delay in long distance data transfer; more simplicity and flexibility in design. The STSS implementations are also very simple and robust, and the metastability in data read is avoided because STSS is completely insensitive to both clock skew and data delay.

Index Terms —ULSI/VLSI, high speed interconnect, synchronization, mesochronous clocking, metastability.

I. INTRODUCTION

As the clock frequency and the scale on silicon chips increase simultaneously for high performance system-on-chip, the clock phase becomes more difficult to calculate or predict. To prevent metastability in data read, global clock synchronization is traditionally used to keep a system working synchronously. However, this design methodology implemented with delay matched clock distribution networks has many drawbacks: more metal layers and high cost in process; large power dissipation for clock distribution; limited scale of systems because of constraints of timing; limited distance allowed for data transfer between two ports; tedious labor work needed in designing PLL or DLL to compensate the propagation delay of local clock driver; significant effort required to cope with delay and skew reductions; less flexibility in design. As for the system scale and clock frequency, physical limits will be eventually reached for the future high-performance system-on-chip design unless we can avoid the global synchronization.

Early studies on synchronization issue in digital system design can be found in [1],[2], where the term of mesochronous was proposed to describe the clocks with the same frequency but different phases.

In this paper, we address three methods of self-tested self-synchronization (STSS) in mesochronous clocking (MC) for large and high speed system design. Input signal with unknown delay can be correctly latched without suffering from data read failure. STSS uses mesochronous clock in each processing element (PE), and automatically selects a clock edge SC for sampling data or switches between two data paths so that an error free parallel data transfer is achieved.

Fig. 1. Global synchronization and mesochronous clocking

II. A MODEL OF MESOCHRONOUS CLOCKING

The main functions inside supercomputers, parallel or distributed systems and artificial neural networks, basically are computation, communication and control. When the computation ability of PEs improves, the load on communication becomes heavy. Synchronous communication is more efficient than asynchronous one, and global clock synchronization is used to avoid metastability in data reading.

An example of global clocked synchronous systems is shown in Fig.1.(a), where we assume that 4×4 PEs communicated with each other are integrated on a chip. A master clock has to be delivered to all PEs at the same clock phase φ0, illustrated by the solid lines, and it causes several problems which will be bottlenecks of next generation of high-performance system-on-chip design. Inside each PE, the local clock phase given by the clock driver, which may introduce a delay time as long as several clock cycles, is locked with phase φ0 by using DLL or PLL.

There is a delay constraint in the parallel data transfer. The data delay cannot be too large as the delay will cause data read failure, and the length of the data transfer between PEs is limited. If a very long distance data transfer is required between two PEs, as PE11 and PE44 in Fig.1.(a), the data transfer can only be implemented with the help of short data transfers between adjacent PEs, for instance, a path including PE11 ⇔ PE12, PE12 ⇔ PE13, ..., PE34 ⇔ PE44.

Because enough timing margin is needed for each data transfer between adjacent PEs, so the total delay time of a data transfer between distant PEs is considerably long.

Symmetric architecture is intensively preferred in the global synchronization or otherwise a powerful program should be used for the delay matching in clock networks. A modification on layout will cause a re-calculation of the clock delay, therefore, design flexibility is limited.

An example of systems with MC can be illustrated in Fig.1.(b), where the PEs can be different in sizes, and clock
can be freely delivered by arbitrary ways which make the clocking much easier than the global synchronization. There is no need to reduce the clock delay as the phases in each PE can be arbitrary. Therefore, the wide metal wires in clock distribution is completely unnecessary. As a result, the power dissipated by clock networks is significantly reduced. Moreover, the architecture of PE can be as precise as it has no thing to do with timing in MC. Data transfer between distant PEs, can be connected directly as long as the data signal can be correctly received. As a consequence, the total delay time in data transfer is greatly reduced. MC can be applied not only to the system-on-chip, but also to the MCM and the system-on-board. By using of MC, systems can be expanded to an extremely large scale without propagation delay constraint for data.

However, before the MC can be utilized, the error free vector transfer between PEs has to be established first. So in the following sections, we will focus on the issue of how to get error free parallel data transfer among the PEs by a self-tested self-synchronization(STSS). Here, we only focus on the input port, as for the output ports vector are retimed by a clock as the same as before. We discuss data transfer in one direction because of the same principle in backwards data transfer.

III. IMPLEMENTATIONS OF STSS

A. Failure Zone and Failure Angle

Failure zone is defined as a forbidden time window for half swing transient point (HSTP) of input. When the input reaches HSTP within the failure zone, edge triggered flip-flops or latches may malfunction because of metastability[3]. The timing of input port is depicted in Fig.2. When input data in (a), is sampled by the clock rise edge, data read failure (DRF) occurs, and the clock falling edge should be used to avoid DRF. For two clock edges, their failure zones are illustrated in (c), notice here the edges and their failure zones are not overlapped. Similarly, for data input in (d), the rise edges should be used. Respect to clock edges in (b), (e) shows the interlaced intervals where if the HSTP denoted by h falls into them, the related clock edge should be used so in phase domain there is a π shift. A time margin is needed to protect the error free parallel data transfer.

The failure zone concept can be expressed in terms of angle, shown in Fig.3, where $t_d$ and $t_a$ are the edges of the failure zone in time domain. $\psi$ is the failure angle in radiant, $t_e$ is the clock trigger point, and $h$ is the HSTP for input, respectively. From Fig.3, we can calculate the probability of data read failure without synchronization,

$$P_e = \frac{T_{fz}}{T_c} = \frac{\psi}{2\pi}$$

for $T_{fz} < T_c$ (1)

where $T_c$ is the clock period, $T_{fz} = t_a - t_d = \psi/\omega_c$ the time window of the failure zone, respectively. It is easily understood that the higher the running frequency, the larger $P_e$ will be. From Fig.3, phase margin is defined as

$$\theta = \min(\theta_a, \theta_b)$$

where $\theta_a$, $\theta_b$ and $\psi$ are less than $2\pi$, and counter clockwise defined.

B. Implementation 1: STSS-1 by Two-Phase Clock

We insert a DFF array to retime input data so DRF free data transfer is achieved. This DFF array is triggered by SC which is guaranteed by Fig.4. Here we describe the first method STSS-I[4] implemented by inserting a test signal, the square wave at half of the clock frequency, going along with the parallel data in a two-phase input port. The STSS-1 is illustrated in Fig.5, where FDDM is the failure detection and decision making circuitry.
Fig. 6. Failure angles in case 1 and 2, where \( t_{xp} \) and \( t_{xp} \) are the positive and negative clock triggering edges, respectively. (a) The case 1. (b) The balanced situation in case 2. (c) The normal situation in case 2.

Fig. 7. An example of artificial jitter injection

If the HSTP of the incoming test signal falls in the failure zone, another clock phase will be selected to trigger. A suitable phase SC must be found to trigger D* so that the error free parallel data transfer is obtained for both the flip-flop and the next latch.

To become more robust, an artificial jitter injection is employed. The artificial jitter blurs the edges of test signals so that if this blurred test signal can pass the testing, the data transfer is more safe.

To make the principle work, a delay match in data lines is needed to compensate the propagation delay of artificial jitter injection circuit. As the clock switch also has a propagation delay, suitable clock phases \( \phi_{10} \) and \( \phi_{11} \) are required at the inputs of the clock switch so that the output \( \psi \) will have the same phase as either \( \phi_{10} \) or \( \phi_{11} \).

According to the possible output statuses of the RS flip-flop, the following cases can occur:

- case 0. \( E0 \in [E] \) and \( E1 \in [E] \)
- case 1. either \( E0 \in [E] \) or \( E1 \in [E] \)
- case 2. \( E0 \in [E] \) and \( E1 \in [E] \)

where \( E \) is a signal output set which contains logic 0 after system warming up. Case 0 can be avoided if \( P_0 < 0.5 \) in principle. Case 1 is a robust choice because the HSTP on input of other path is in failure angle, see Fig. 6(a). However, in case 2, see Fig. 6(b) and (c), it is tricky to find a robust choice. We cannot expect to get maximum phase margin, however, if a proper method is employed we can gain a certain phase margin which may be enough to protect the error free parallel data transfer.

C. Artificial Jitter Injected Test Signal

In order to be foolproof against the unreliable, the test transfer is checked more strictly than the data transfer. Here we address a method of improving the robustness in STSS.

If more than one choice, as the case 2 mentioned previously, are available, a blind decision has to be made among the two possible choices. In such a case, the blind decision may not be robust if the phase margin is small and error free parallel data transfer cannot be absolutely guaranteed, as in the Fig.6(c)

when \( t_{xp} \) is selected. In order to examine the result more strictly, a small artificial jitter is injected to the test signal to make its receiving situation even worse than the parallel lines. If the deteriorated test signal passes the port without error, so does the parallel data. Therefore, the result is more robust and reliable. This is reason why the artificial jitter is introduced.

An example of the artificial jitter injection circuits are shown in Fig. 7. SWI is a low frequency square wave sequence. Fig. 7(a) imposes SWI on the test signal and jitter is coupled though the inverters I1 and I2, here I2 is much weaker than I1 and C is the total parasitic capacitance on the output node, respectively. Fig. 7(b) shows the waveforms on nodes.

In order to remove data read failure, see Fig. 8, following constraint must be met in terms of angle,

\[
\psi + 2\theta_n + 2\Delta \phi < \pi
\]  

where \( \Delta \phi > 0 \) is the guard angle to ensure that case 0 is eliminated. With the injection of the artificial jitter of angle \( \theta_n \), then the phase margin will be at least equal to \( \theta_n \) if a choice exists. Therefore, the wrong choice in Fig. 6(c), can be avoid if the phase angle is less than \( \theta_n \).

D. Implementation 2: STSS-2 by Two-Phase Clock

Another implementation, STSS-2 is illustrated in Fig. 9, where the square wave at half of the clock frequency is removed, and so it can be used when dimension \( m \) is small or \( m = 1 \). A test chip is fabricated and tested for both STSS-1 and STSS-2. The measurement and result of STSS-2 can be seen in [5]. The measurement of STSS-4 is configured as in Fig. 10, where an adjustable delay unit is inserted between STSS-1 and the pattern generator. We sweep delay in tiny steps (0.1ns) and adjust the variable delay unit (0.26ns) so that over one clock cycle the delay is continuously changeable to find if there is a DRF. Result shows the mechanism works well, and the waveform is shown in Fig. 11, where curve 1 is the input data, and curve 2 is output data.

E. Implementation 3: STSS by Single Phase Clock

STSS using single phase clock (STSSS-SPC) is illustrated in Fig. 12, which is quite similar to the STSS-1 in principle. However, we use N data path and P data path instead of inserting an array of DFF, and in test paths we employ P test and N test to check the failure. FDDM is the same as in the STSS-1. SPICE simulation shows that the STSS using single phase clock can reach bit rate at 1.4Gb/s with 0.6µm CMOS, which is much higher than STSS-1 and
STSS-2, because there is no need of double sampling during one clock period.

IV. CONCLUSION

In this paper, we investigate of parallel data transfer between two PEs with mesochronous clocking. Three methods of self-tested self-synchronization (STSS), remove the requirements of global synchronization and the constraint on clock skew. STSS is also robust if a proper artificial jitter is injected.

STSS-1 is accomplished with help of inserting a test signal and the error status of test is used to select a clock edge to get the error free parallel data transfer between PEs. Failure detection of STSS-2 is based on timing relation between clock and data. STSS-1 and STSS-2 use two-phase clock and retine the inserting an array of DFF. STSS-SPC selects a proper data path between P and N data paths, and it can reach much higher speed than the others. It also needs a test signal to find out the control signal and a skillful design to reach this speed. Compared to the solution in [2], our three methods are much simpler.

By these methods, the global clock synchronization is avoided. So there is no need of the delay matched trees in clock distribution and skew reduction techniques. Therefore, significant simplification is achieved by these methods. The power consumed by the clock distribution is efficiently reduced because there is no need of using of metal wires to shorten the delay, and it is more suitable for distributed clock drivers in each PE.

Based on the above results, the problem of synchronization is solved for huge ULSI systems, in which there is no need of the dedicated delay matching in the clock distribution, the attentions paid to the clock deskew by DLL or PLL in each PEs, the constraint on the delay of the parallel data transfer. Clock can be distributed by arbitrary ways, and the power consumption caused by the clock distribution and the peak current resulting from synchronously switching will be reduced greatly. There is no limitation on the scale of the systems. In addition, the mechanism of STSS makes design and design automation much easier. By this method, the parallel data transfer can reach full speed so maximum throughput can be obtained.

REFERENCES