Timing Closure through a Globally Synchronous, Timing Partitioned Design Methodology

Anders Edman[1]
Computer Engineering
Linköping University
SE-581 83 Linköping, Sweden
+46 13 282779
anded@isy.liu.se

Christer Svensson
Electronic Devices
Linköping University
SE-581 83 Linköping, Sweden
+46 13 281223
chs@isy.liu.se

ABSTRACT
A method to mitigate timing problems due to global wire delays is proposed. The method follows closely a fully synchronous design flow and utilizes only true digital library elements. The design is partitioned into isochronous blocks at system level, where a few clock cycles latency is inserted between the isochronous blocks. This latency is then utilized to automatically mitigate unknown global wire delays, unknown global clock skews and other timing uncertainties occurring in backend design. The new method is expected to considerably reduce the timing closure effort in large high frequency digital designs in deep submicron technologies.

Categories and Subject Descriptors
B.7.1 [**Integrated Circuits**]: Types and Design Style C – *VLSI (very large scale integration*)

General Terms: Design

Keywords: Timing closure, wire delays, clock skew

1. INTRODUCTION
It has been recognized for a long time that integrated circuits in deep submicron technologies exhibits increased timing problems due to increased clock frequencies, increased complexity and increased wire delays [1,8,2]. These problems manifest themselves as a severe increase in verification cost (timing closure), increased problems to scale up the clock frequency and increased effort for clock distribution. Many methods to mitigate these problems have been proposed, often calling for more or less severe changes in a very well established standard design flow or for nonstandard library cells. However, any discrepancy from the established design flow severely hampers the acceptance of a new methodology.

We aim at finding a design methodology that manages unpredictable wire delays (data delays and clock skew) with an absolute minimum of influence on the established design flow.

DAC'04, June 7–11, 2004, San Diego, California, USA

We would like to see a methodology which allow the designer to complete his clock-true verification before moving into the synthesis step, to use only existing tools and design flow all the way to layout, and to use only standard library cells.

In this paper we propose a scheme to mitigate the timing problems. Our main goal is to keep the fully synchronous design flow even for large high-speed designs in deep submicron processes. The idea is to take care of the inevitable wire delays already at architecture level, and then guarantee that the functional description at this level is valid all the way to layout. This idea thus follows the conclusions in [7] that keeping performance when meeting wire delay problems requires architectural level considerations

Our scheme is based on the partition of a large design into isochronous blocks, still keeping global synchronism. Each isochronous block should be small enough not to exhibit severe wire delays [8]. Between the isochronous blocks we insert a few extra delays (pipelining). The extra delays are later utilized to automatically mitigate unknown wire delays and clock skew. The proposed scheme simplifies timing closure and relaxes clock distribution constraints.

In section 2 we describe previous work and in section 3 our proposed scheme. We describe the design methodology in section 4 and a proposed implementation section 5, followed by a description of design constraints in section 6. In section 7 we demonstrate a simulated example. The paper is finished by a discussion and a conclusion in sections 8 and 9.

2. PREVIOUS WORK
One approach is to mitigate the timing problem though the improvement of present design methodology and improved tools. Such methods are based on a traditional, synchronous high level design flow, followed by a modified synthesis and backend flow, only leading to a limited improvement (less than 50% improvement of worst case delay) [6].

More recent work propose synchronous solutions which makes a design functionally insensitive to the latency of long wires [2].

1 Author also at: Acreo AB, Bredgatan 34, SE-602 21 Norrköping, Sweden,Tel. +46 11 363634, email: anders.edman@acreo.se

This Latency Insensitive Design (LID) method partitions a large design into several communicating modules, where each module is "patient", that is its function only depends of the order of events reaching it. It is shown that the functionality of a system built with such modules is independent of the latency of the communications channels between the modules. A drawback with this method is that its performance (in cycle count for a certain operation) is not known until after backend.

Alternative implementations of the physical layer of the LID method, based on FIFO's, were proposed in [4]. These FIFO's perform synchronization between two clock domains in a similar way as some earlier proposed synchronizers based on parallel latches (or a pointer-FIFO) [4,5]. An elegant and very compact solution to the synchronization problem utilizes a single-stage FIFO, which behavior mimics a latch clocked from both clock domains [3].

3. PROPOSED SCHEME

Let us define an isochronous region as a partition of the design within which the clock is synchronous This means that the clock skew is small enough not to give rise to any races or hazards for any combination of cells in a given cell library. An isochronous region can therefore be automatically synthesized without risk for timing violations. The size of such a region is however limited, both in the number of gates and in physical size (making wire delays constrained to a given value). Let us further define a region boundary as the boundary between two isochronous regions. At system level, this boundary must allow a certain degree of latency for all signals passing the boundary; say n clock cycles (n>0). See fig. 2. Finally, output signals must be delivered directly from a flip-flop and inputs can be assumed to have a timing as if they were generated by internal flip-flops.

These definitions are sufficient constraints to system level design and guarantee that the design can be implemented without timing problems. When implementing a design of this kind, certain timing cells are inserted at each signal that passes a region boundary. Such a cell can manage a single signal or a bundle of signals. A signal passing a boundary may pass to a neighboring region or to a remote region. For each independent signal or bundle of signals we add a strobe.

The clock distribution network will deliver a clock to each isochronous region, with no skew constraint. Inside each region the clock distribution is stricter (which should not be a problem if the region is small enough).

4. DESIGN METHODOLOGY

The main goal with the present work is to device a design method that mitigates the problem of timing closure. This is achieved as follows. At early high-level system design, the design is partitioned into blocks of limited size (anticipated limited physical size) and with defined block-to-block link latency. Typically, the partitioning should follow the blocks in a functional block diagram, and not divide a function into several blocks. The size of each block should be small enough not to cause timing closure problems at the actual clock frequency when treated as a separate design in the target process. Next, the maximum delay for the longest path of the full chip is estimated. Also, easily achievable inter-block clock skew, i.e. difference between maximal and minimal clock insertion delay, is estimated. Using the equations derived in section 6, a value of n (the pipelining in the link) is calculated from these estimations. For certain critical block-to-block links a lower value of n can be chosen, if it can be foreseen that the floorplan allows a physically shorter link. However, if we can use the calculated value for n in all links, no floorplan decisions are required until back-end work starts. Selected values for each link are implemented as an n clock-cycle long delay unit in the clock-cycle true model of the design, (fig. 2). This model of the system is used throughout the design phase and also used for clock-cycle true verification. Here, the design methodology will also have an indirect impact. Although all link delays are early defined we can guarantee that back-end design will not require any changes in the design. Thus, re-design and re-verification due to implementation problems can be avoided.

During the following synthesis, the n-cycle delays are replaced by two-port memory synchronizers (see below). These memory blocks are generated using flip-flops from the standard cell library as storage elements. The cells for the two-port link memory are placed in the receiving block.

The high-level design partitioning mentioned above is course-grained and considered to be a natural part of system architecture. The isochronous regions or blocks are typically processors, controllers, or memories. These blocks are thus relatively self-contained and have well-defined interfaces to other blocks. In IP-based designs, each IP normally constitutes one isochronous region.

5. IMPLEMENTATION

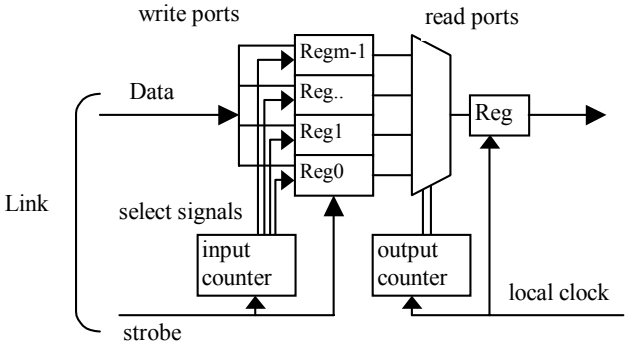

Figure 1. Two-port memory synchronizer.

We will now describe a possible implementation of the proposed scheme, based on the well-known FIFO synchronizer [5]. As said above, a certain timing cell is inserted at each signal or signal bundle passing a region boundary. Such a timing cell is an m-word two-port memory with separate writing and reading ports, inserted on the target side of the boundary. A schematic of the timing cell is shown in Figure 1. Any incoming word is consecutively written into the memory cells, using round robin addressing and clocked by the incoming strobe. The memory cells are simultaneously read, using a delayed round robin addressing and clocked by the local clock in the target region. The addressing delay between write and read, and the number of words in the memory, is chosen so that write and read instances can never collide, taking the maximum skew between the two regions into account.

In order to facilitate global synchronization and clock alignment, all local clock periods must be enumerated in the same way. This means that we must relate each word arriving to an isochronous region to a corresponding local clock cycle. A very simple procedure to accomplish this is as follows. The system is initiated by a global, asynchronous reset with no clock running. During this reset the counters is reset to zero and m-n respectively. Then the clock starts. The first clock cycle is distributed to each isochronous region via the clock distribution network. Each transmitter sends a strobe, which is equal to the clock at that transmitter. When the first strobe pulse arrives to the receiver, it corresponds to clock period #0 and the input counter starts to count up from period #0. In the same way, when the first clock pulse arrive to that same isochronous region, it corresponds to the local clock period #0, and the output counter starts to count up from that period. In such a way we have the same enumeration of all clock and strobe periods in the system, so we do not need any frame synchronization etc.

We may note that during the process described here we never experience any risk for metastability, not even during startup. The reason is that 1) the two FIFO pointers, clocked by different clocks, never collide, and 2) the global clock is turned off during reset, so the independently timed reset and clock never collide. We should also note that clock and link delays may vary (within certain limits) also during system operation, without any risk of failure. Finally we use no analog functions whatsoever in the design.

6. DESIGN CONSTRAINTS

Certain constraints must be fulfilled to guarantee error-free function of the proposed communication scheme. Fig. 2 shows an implementation with two links between three isochronous regions, A, B and C. Let us further explain the proposed communication process referring to the Tx1-Rx1 link in fig. 2. We show how a global clock is distributed from a clock root to each isochronous region. The total clock delays from the clock root to the transmitter and receiver inside A and B respectively are t_A and t_B. The clock at the transmitter is copied to the strobe sent from the transmitter. We further introduce the delay of the communication link, t_{L1}, assumed equal for the data wires and the strobe wire. Considering the 2-port memory as written and read at positive clock edges, we may characterize its timing constraint as the clock collision time t_{cc}. t_{cc} is defined as the minimum time between write and read. Using the clock root as reference as above, register0 is written by the positive clock edge at time t_A+t_{L1}. The reading process of register0 starts at time t_B+nT and must not start before t_{cc} after the write process. We therefore have:

$$t_A+t_{L1}+t_{cc} < t_B+nT \quad (1)$$

In the same way the start of next write process into that same register, which occurs at time $t_A+t_{L1}+mT$ must not start until t_{cc} after the read process. We therefor have the second constraint:

$$t_B+nT+t_{cc} < t_A+t_{L1}+mT \quad (2)$$

The total "skew", defined as $t_A+t_{L1}-t_B$ may then vary within the limits

$$(n-m)T+t_{cc} < t_A+t_{L1}-t_B < nT-t_{cc} \quad (3)$$

The nominal link delay, n, should thus be chosen from the maximum possible total skew, consisting of the link delay, t_{L1} and the clock skew, t_A-t_B. Note that in this case the absolute value of clock skew is relevant, as we use the first clock cycle for local clock cycle enumeration. After had chosen n, the number of memory cells, m, should be chosen from the minimum possible total skew (which may be negative if t_{L1} is small).

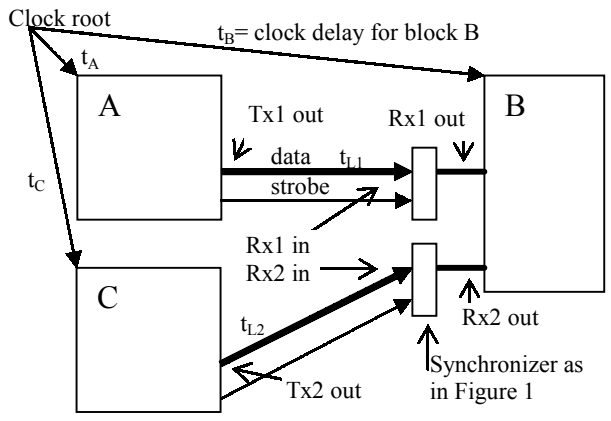

Figure 2. Clock and data links have delays and the n clock-periods delay is replaced by a synchronizer.

7. SIMULATION EXAMPLE

As a verification of the proposed method we performed a small behavioral simulation. Two transmitters in two different isochronous blocks, Tx1 and Tx2, and two receivers in the same isochronous block, Rx1 and Rx2, were implemented in behavioral VHDL, see fig. 3. In both links we have n=2 (two extra clock delays on each link) and m=4. Three different clock delays from clock root to Tx1, Tx2 and Rx resp. are defined, t_A, t_C and t_B, as are two different link delays, t_{L1} and t_{L2}. One extreme simulated example is shown in fig. 3, using a clock period of 5ns. The curves shows from top: clock root, output from Tx1 (a data sequence of 10110010110), data arrived to Rx1, Tx2 output (same data sequence), data arrived to Rx2, output from Rx1 into Rx block, output of Rx2 into Rx block, in-counter of Rx1, out-counter of Rx1, in-counter of Rx2 and finally out-counter of Rx2. In fig 3 we have all delays equal to 1ns except t_{L1} which is 9ns (1.8 clock cycles). The large delay is seen as a delay between Tx1 out and Rx1 in in the figure. We note that in spite of very different link delays, the data outputs from ports Rx1 and Rx2 in the Rx block are completely synchronized (note that we transmitted the same sequence from Tx1 and Tx2, aligned to respectively local clock). We also simulated a case with the same delays as above except the clock root to Rx block delay which is t_B=11ns (2.2 clock cycles). The large clock delay was noted as a delay of 4.2 clock cycles (n + 2.2 clock cycles) from the start of clk to the first data out at Rx1 out and Rx2 out. Again we observed that correct and aligned data streams are received at ports Rx1 out and Rx2 out in the Rx block.

The two cases represent extremes compared to the constraints eq. (3). In the first case we have (n-m)T=-10ns, $t_A+t_{L1}-t_B$=9ns and nT=10ns for the worst case (Tx1-Rx1), giving us a margin of 1ns for t_{cc}. In the second case we have (n-m)T=-10ns, $t_A+t_{L2}-t_B$=-9ns and nT=10ns for the worst case (Tx2-Rx2), again giving us a margin of 1ns for t_{cc}.

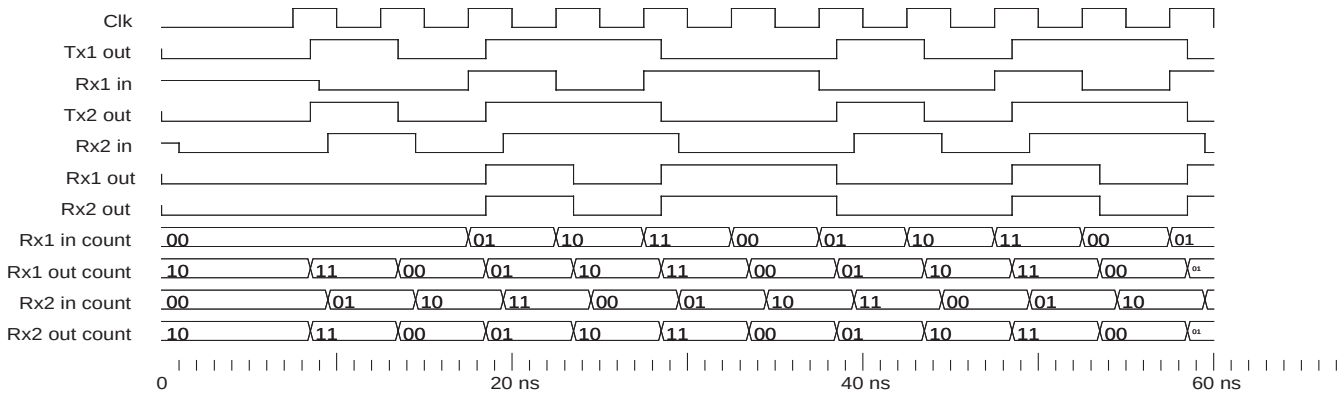

Figure. 3. Examples of waveforms of a virtually synchronous link between two transmitters in different blocks, Tx1 and Tx2, and two receivers in the same block, Rx1 and Rx2.

8. DISCUSSION

The proposed scheme is easily generalized to multiport isochronous regions, to any number of isochronous regions, to bidirectional links and to any number of links. Extending the number of ports into one isochronous region do not introduce any problem, as the local clock is the same for all ports and as the local clock period enumeration is based on this clock and not on the strobes. Any number of isochronous regions can be used, as each is managed independently of the others. A bidirectional link is simply created by using two independent unidirectional links (with individual strobes). Links can be extended to have multiple receiving blocks, each with its own synchronizer. Finally, any number of links between any isochronous regions can be implemented, as each link is independent of the other links.

9. CONCLUSION

We have proposed a new method addressing the timing closure problem in IC design. The method is based on early system partitioning into synchronous blocks of limited size and the insertion of fixed latencies between these blocks. These latencies are automatically utilized to mitigate layout-induced delays and clock skews during the physical design stage. The method has the following benefits:

- Correct clock-true verification before synthesis and physical design.
- Standard synchronous design framework kept unchanged.
- Any standard digital cell library supports the method.
- Both data delays and clock skews induced by place and route are mitigated.
- Robust to late changes in RTL-code or floorplan.
- Full clock alignment between blocks automatically satisfied.
- Prevents redesign due to timing issues in back-end design.
- Simplifies RTL sign-off for large high-speed designs.

10. REFERENCES

[1] M. Afghahi and C. Svensson, “Performance of Synchronous and Asynchronous Schemes for VLSI Systems”, IEEE Trans. on Computers, Vol. 41, pp. 858-872, 1992.

[2] L. P. Carloni, K. L. McMillan, A. Saldanha and A. L. Sangiovanni-Vincentelli, "A Methodology for Correct-by-Construction Latency Insensitive Design", 1999 IEEE/ACM International Conference on Computer Aided Design, pp. 309-315, Nov. 1999.

[3] A. Chakraborty and M. R. Greenstreet, "A Minimal Source-Synchronous Interface", 15th Annual IEEE International ASIC/SOC Conference, pp. 443-447, Sept. 2002.

[4] T. Chelcea and S. M. Nowick, "Robust Interfaces for Mixed-Timing Systems with Application to Latency-Insensitive Protocols", Proceedings of 2001 Design Automation Conference, pp. 21-26, June 2001.

[5] W. J. Dally and J. W. Poulton, Digital Systems Engineering, Cambridge University Press, 1998, chapter 10.3.

[6] W. Gosti, S. P. Khatri and A. L. Sangiovanni-Vincentelli, "Addressing the Timing Closure Problem by Integrating Logic Optimization and Placement", IEEE/ACM Int. Conference on CAD 1998, pp. 224-231, 2001.

[7] D. G. Messerschmitt, "Synchronization in Digital System Design", IEEE Journal on Selected Areas in Communications, vol. 8, pp. 1404-1419, 1990.

[8] D. Sylvester and K. Keutzer, "Getting to the bottom of deep submicron", IEEE/ACM Int. Conference on Computer Aided Design 1998, Digest of Technical Papers, pp. 203-211, 1998.