From synchronous to asynchronous: an automatic approach

J. Cortadella
Univ. Politècnica de Catalunya
Barcelona, Spain

A. Kondratyev
Cadence Berkeley Labs
San Jose, CA, USA

L. Lavagno
Politecnico di Torino
Torino, Italy

K. Lwin
Cadence Berkeley Labs
Berkeley, CA, USA

C. Sotiriou
ICS-FORTH
Crete, Greece

Abstract

This paper presents a methodology to derive asynchronous circuits from optimized synchronous circuits by replacing the clock distribution tree by a handshaking network. A case study shows the applicability of the method and the potential benefits of de-synchronizing synchronous circuits.

1. Introduction

We propose a methodology to design asynchronous circuits that *does not require any knowledge of asynchronous design*. The essential idea is to start from a synchronous synthesized (or manually designed) circuit, and *replace directly the global clock network with a set of local handshaking circuits*. The circuit is the one *implemented with standard tools, using a flow originally developed for synchronous circuits*. The only modification is the clock tree generation algorithm. The method still provides several key advantages of asynchronicity, such as low clock power, low EMI, global idling, and modularity. To prove that the suggested methodology is sound we compared synchronous and de-synchronized designs of a DLX processor. Both designs were implemented using the same set of commercial EDA tools for synthesis, placement and routing.

The automatic generation of asynchronous circuit has also been faced by other authors. There have been approaches to derive delay-insensitive circuits [4], handshake circuits to generate local clocks [5] and doubly-latched asynchronous pipeline [3]. Our work is the first that proposes a method with negligible overhead while providing a theoretical foundation for the de-synchronization approach.

2. De-synchronization

The approach presented in this paper assumes that the circuit has combinational blocks (CL) and registers implemented with D flip-flops (FF), all of them working with the same clock edge (e.g. rising in Figure 1(a)).

It proceeds in three steps: (1) *Conversion into a latch-based synchronous circuit (M and S latches in Figure 1(b))*,

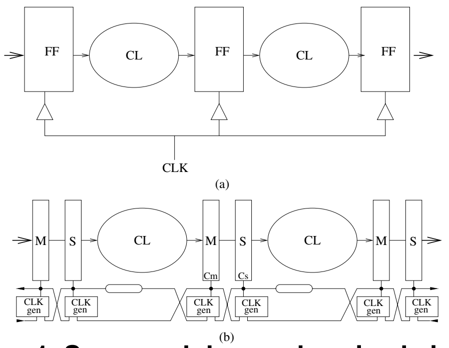

Figure 1. Sync. and de-synchronized circuit.

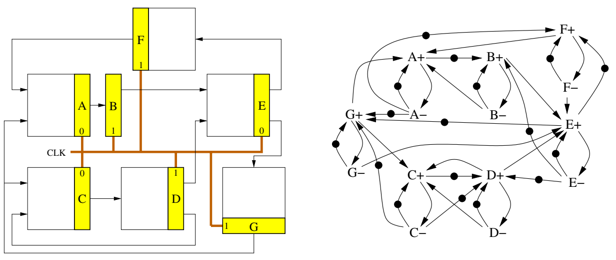

Figure 2. Synchronous circuit with a global clock and its de-synchronization model.

(2) generation of matched delays for combinational logic, and (3) interconnection of controllers for local clocks.

Figure 2 depicts a synchronous netlist after conversion into latch-based design. The shadowed boxes represent latches, whereas the white boxes represent combinational logic. Latches with a label 0 (1) represent the *even* (*odd*) latches, transparent when the clock is low (high).

Pipeline de-synchronization model. A timing diagram and the corresponding marked graph for a simple pipeline is depicted in Fig. 3. The latches are transparent when the control signal is high. Initially, only half of the latches contain data ($\mathcal{D}$). Data items flow in such a way that a latch

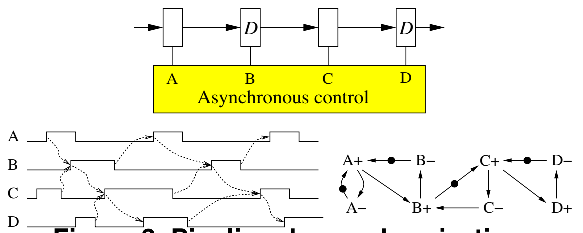

Figure 3. Pipeline de-synchronization.

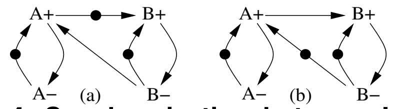

Figure 4. Synchronization between latches: (a) even → odd, (b) odd → even.

never captures a new item before the successor latch has captured the previous one. Data overwriting can never occur, even though the pulses for the latch control can overlap. This model is based on the observation that a data item can ripple through more than one latch, as long as the previous values stored in those rippling latches have already been captured by the successor latches.

General de-synchronization model. The pipeline model can be generalized for any netlist, while preserving a property that makes de-synchronized circuits equivalent to their synchronous versions: *flow equivalence* [2].

The method for de-synchronizing a netlist relies on composition of the controllers. We just identify pairwise interactions between adjacent latches, and then the overall clock generation circuit is obtained through composition of these partial descriptions. The specification of a pairwise interaction between even-odd and odd-even latches for overlapping de-synchronization is shown in Figure 4. It models the communication of data from latch A to latch B. These patterns exactly correspond to the ones used in the behavioral specification of a linear pipeline in Figure 3. The only difference is that we added two auxiliary arcs $A- \to A+$ and $B- \to B+$ to model the behavior of the abstracted parts of the system (those that precede A and succeed B).

The marked graph in Fig. 2 depicts the de-synchronization model for the netlist at the left, as obtained by composition of patterns from Figure 4. In [1], an implemention of the controllers and a formal proof of the correctness of the approach are presented.

3. De-Synchronization Case Study

We present results on the application of de-synchronization to a DLX processor. The circuit has been obtained by using synthesis tools from Cadence (G2C-RC, SoC Encounter, Amoeba and NanoRoute). Table 1 contrasts the characteristics of the synchronous and of the de-synchronized DLX. The data are post-layout results based on gate-level simulations with back-annotation of extracted parasitics.

	Sync. DLX	**De-Sync. DLX**
Cycle Time	4.4ns	4.45ns
Dyn. Power Cons.	70.9mW	71.2mW
Area	372,656 μm^2	378,058 μm^2

Table 1. Sync. vs. De-Synchronized DLX.

To the best of our knowledge, this is the first successful attempt of delivering an automated design flow for asynchronous circuits that does not introduce significant penalties to corresponding synchronous designs. This opens wide opportunities of exploring the implementation space within the very same set of industrial tools. This, we believe, is a valuable feature for a designer.

4. Conclusions

This paper presented a de-synchronization model that can be used to automatically substitute the clock network of a synchronous circuit by a set of asynchronous controllers. This results in EMI improvements, shortens the design cycle, and can allow one to measure more easily the performance of each manufactured circuit. We believe that these techniques are a significant step toward spreading the use of asynchronous circuits among mainstream designers.

Acknowledgment. This work has been partially funded by CICYT TIC2001-2476, ACiD-WG (IST-1999-29119), the ASPIDA project (IST-2002-37767) and a Distinction by the *Generalitat de Catalunya*.

References

[1] J. Cortadella, A. Kondratyev, L. Lavagno, and C. Sotiriou. A concurrent model for de-synchronization. Extended version of paper presented at IWLS03. www.lsi.upc.es/˜jordicf/publications/pdf/iwls03_extended.pdf.

[2] P. L. Guernic, J.-P. Talpin, and J.-C. L. Lann. Polychrony for system design. *Journal of Circuits, Systems and Computers*, Apr. 2003.

[3] R. Kol and R. Ginosar. A doubly-latched asynchronous pipeline. In *Proc. International Conf. Computer Design (ICCD)*, pages 706–711, Oct. 1996.

[4] D. H. Linder and J. C. Harden. Phased logic: Supporting the synchronous design paradigm with delay-insensitive circuitry. *IEEE Transactions on Computers*, 45(9):1031–1044, Sept. 1996.

[5] V. Varshavsky, V. Marakhovsky, and T.-A. Chu. Logical timing (global synchronization of asynchronous arrays. In *The First International Symposium on Parallel Algorithm/Architecture Synthesis*, pages 130–138, Aizu-Wakamatsu, Japan, Mar. 1995.