Design Automation of Real-Life Asynchronous Devices and Systems

Design Automation of Real-Life Asynchronous Devices and Systems

Alexander Taubin
Boston University, USA

Jordi Cortadella
Universitat Politècnica de Catalunya, Spain

Luciano Lavagno
Politecnico di Torino, Italy

Alex Kondratyev
Cadence Design Systems, USA

Ad Peeters
Handshake Solutions, The Netherlands

now
the essence of knowledge

Boston – Delft

Foundations and Trends® in Electronic Design Automation

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
USA
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is A. Taubin, J. Cortadella, L. Lavagno, A. Kondratyev and A. Peeters, Design Automation of Real-Life Asynchronous Devices and Systems, Foundations and Trends® in Electronic Design Automation, vol 2, no 1, pp 1–133, 2007

ISBN: 978-1-60198-058-8

Foundations and Trends® in Electronic Design Automation
Volume 2 Issue 1, 2007

Editorial Scope

Foundations and Trends® in Electronic Design Automation will publish survey and tutorial articles in the following topics:

- System Level Design
- Behavioral Synthesis
- Logic Design
- Verification
- Test
- Physical Design
- Circuit Level Design
- Reconfigurable Systems
- Analog Design

Information for Librarians

Foundations and Trends® in Electronic Design Automation, 2007, Volume 2, 4 issues. ISSN paper version 1551-3939. ISSN online version 1551-3947. Also available as a combined paper and online subscription.

Foundations and Trends® in
Electronic Design Automation
Vol. 2, No. 1 (2007) 1–133
© 2007 A. Taubin, J. Cortadella, L. Lavagno,
A. Kondratyev and A. Peeters
DOI: 10.1561/1000000006

Design Automation of Real-Life Asynchronous Devices and Systems

Alexander Taubin[1], Jordi Cortadella[2], Luciano Lavagno[3], Alex Kondratyev[4] and Ad Peeters[5]

[1] *Boston University, USA, taubin@bu.edu*
[2] *Universitat Politècnica de Catalunya, Spain, jordi.cortadella@upc.edu*
[3] *Politecnico di Torino, Italy, lavagno@polito.it*
[4] *Cadence Design Systems, USA, kalex@cadence.com*
[5] *Handshake Solutions, The Netherlands, ad.peeters@handshakesolutions.com*

Abstract

The number of gates on a chip is quickly growing toward and beyond the one billion mark. Keeping all the gates running at the beat of a single or a few rationally related clocks is becoming impossible. In static timing analysis process variations and signal integrity issues stretch the timing margins to the point where they become too conservative and result in significant overdesign. Importance and difficulty of such problems push some developers to once again turn to asynchronous alternatives.

However, the electronics industry for the most part is still reluctant to adopt asynchronous design (with a few notable exceptions) due to a common belief that we still lack a commercial-quality Electronic Design Automation tools (similar to the synchronous RTL-to-GDSII flow) for asynchronous circuits.

The purpose of this paper is to counteract this view by presenting design flows that can tackle *large* designs without significant changes with respect to synchronous design flow. We are limiting ourselves to four design flows that we believe to be closest to this goal. We start from the Tangram flow, because it is the most commercially proven and it is one of the oldest from a methodological point of view.

The other three flows (Null Convention Logic, de-synchronization, and gate-level pipelining) could be considered together as asynchronous re-implementations of synchronous (RTL- or gate-level) specifications. The main common idea is substituting the global clocks by local synchronizations. Their most important aspect is to open the possibility to implement large legacy synchronous designs in an almost "push button" manner, where all asynchronous machinery is hidden, so that synchronous RTL designers do not need to be re-educated. These three flows offer a trade-off from very low overhead, almost synchronous implementations, to very high performance, extremely robust dual-rail pipelines.

Contents

1 Introduction

1.1 Requirements for an Asynchronous Design Flow

With the increases in die size and clock frequency, it has become increasingly difficult to drive signals across a die following a globally synchronous approach. Process variations and signal integrity stretch the timing margins in static timing analysis to the point where they become too conservative and result in significant over-design. Asynchronous circuits measure, rather than estimate, the delay of the combinational logic, of the wires, and of the memory elements. They operate reliably at very low voltages (even below the transistor threshold), when device characteristics exhibit second and third order effects. They do not require cycle-accurate specifications of the design, but can exploit specification concurrency virtually at every level of granularity. They can save power because they naturally perform computation on-demand.

Asynchronous design also provides unique advantages, such as reduced electromagnetic emission, extremely aggressive pipelining for high performance, and improved security for cryptographic devices. In addition, asynchronous design might become a necessity for non-standard future fabrication technologies, such as flexible electronics based on low-temperature poly-silicon TFT technology [50] or nano-computing [97].

In the next section, we will discuss more in detail the main motivations to start using asynchronous approaches in real-life designs. Here we just mention that the problems listed above are suggesting some designers to consider asynchronous implementation strategies again, after decades of disuse.

Despite all its potential advantages, asynchronous design has traditionally been avoided by digital circuit and system designers due to several reasons:

- There were no good EDA tools and methodologies that completely covered the design flow (especially for the large, realistic designs).
- Testing without a clock did not benefit from the well-established and reliable set of procedures ensuring that a synchronous circuit will work as expected after fabrication.
- Asynchrony required a deep change in designers' mentality when devising the synchronization among various components of a digital system.

Custom and semi-custom design of asynchronous devices and systems, on the other hand, is a well established academic research area, offering several well-known success stories including realistic designs (e.g., [31, 113, 36, 50, 72, 79, 80]). Some asynchronous implementations have been reportedly developed and sometimes commercialized by major design companies (e.g., Intel's RAPPID design [107]; Sun's high-speed pipelined devices used in a commercial Sun Ultra are based on results from [15, 83, 124]; IBM/Columbia low-latency synchronous–asynchronous FIR filter chip [114, 128] was fabricated, etc.).

However, until recently design and testing automation was considered a major weakness of asynchronous design approaches.

Asynchronous design is a very large research domain, and it is almost impossible to cover it in depth within a single paper. The interested reader is referred to a number of excellent research papers, books, and surveys devoted to design automation tools and flows for asynchronous circuits (e.g., [17, 18, 23, 32, 33, 64, 88]) and in general to asynchronous design (e.g., [123, 51, 81]).

This article is devoted specifically to one topic: automation of asynchronous design based on industrial-quality tools and flows. This requires support throughout the design cycle, mostly re-using synchronous tools and modeling languages due to the otherwise prohibitive investment in training and development.

We also aim at dispelling a common misbelief, namely that asynchronous design is difficult, and that only specially educated PhDs can do it. We show that this is no true and that there exist flows and tools that satisfy the following key requirements:

- They must be able to handle real-size designs and to re-use the huge investments in tools, languages and methodologies that enable synchronous design.
- They must use standard hardware description languages (HDL), such as Verilog or VHDL, with a modeling style that does not differ much from the synthesizable subset, in order to handle legacy designs.
- They must use standard EDA tools to synthesize, map, place, route the design, and to perform timing analysis, equivalence checking, parasitics extraction, scan insertion, automated test pattern generation and so on.
- They must use CMOS standard-cell libraries, except when specially designed dynamic CMOS libraries are required for maximum performance and minimum power consumption.
- They must be scalable, in order to create industrial-quality and industrial-size designs.

Any change in the design methodology, no matter how small, must be strongly motivated. We believe that this is happening now, particularly since asynchrony is the most natural, powerful and effective mechanism to address manufacturing and operating condition variability. For integrated circuits at 45 nm and beyond, asynchrony is also a natural way to overcome power, performance and timing convergence issues related to the use of clock signals, supply voltage drop and delay uncertainty caused by noise.

1.2 Motivation for Asynchronous Approach

Feedback closed-loop control is a classical engineering technique used to improve the performance of a design in the presence of manufacturing uncertainty. In digital electronics, synchronization is performed in an open-loop fashion. That is, most synchronization mechanisms, including clock distribution, clock gating, and so on are based on a feed-forward network. All delay uncertainties in both the clock tree and the combinational logic must be designed out, i.e., taken care of by means of appropriate worst-case margins.

This approach has worked very well in the past, but currently it shows several signs of weakness. A designer, helped by electronic design automation tools, must estimate at every design stage (floor-planning, logic synthesis, placement, routing, mask preparation) the effect that uncertainties will have on geometry, performance and power (or energy) of the circuit. Cycles in the design and/or fabrication flow translate into both time-to-market delays and direct non-recurrent engineering costs. In the case of mask geometry and fabrication, these uncertainties have so far had mostly a local effect, which can be translated into design rules. However, as technology progresses towards 45 nm and beyond, non-local effects, e.g., due to geometry density and orientation, are becoming more and more significant. In the case of delay and power, these uncertainties add up to significant margins (e.g., 30% was reported in [89]) in order to ensure that a sufficiently large percentage of manufactured chips works within specifications.

Statistical timing analysis [3] attempts to improve the model of the the impact of correlated and independent variability sources on performance. However, it requires a deep change in the business relationship between foundries and design houses in order to make statistical process data available to design tools. The commercial demonstrations of statistical timing analysis viability have been scarce up to now. Moreover, it is still based on improving predictions, not on actual post-manufacturing measurements.

Signal integrity comes into play as both crosstalk-induced timing errors and voltage drop on power lines. Commercial tools for signal and power integrity analysis and minimization (e.g., [11, 112]) predict and

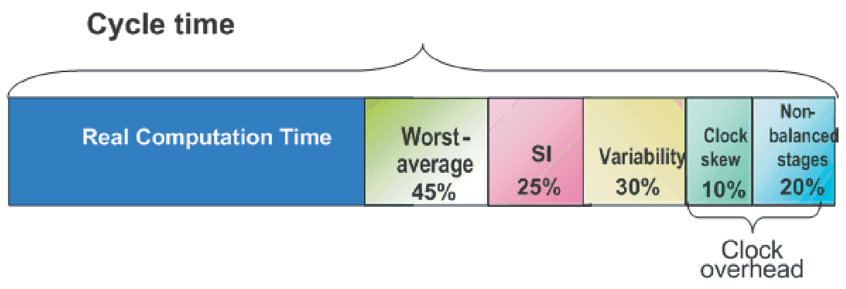

Fig. 1.1 Delay penalties in a synchronous system.

help reduce the impact of voltage drop and crosstalk noise on circuit performance. However, it is believed that up to 25% of delay penalty may be due to signal integrity.

Figure 1.1 (from [7]) summarizes the delay penalties that are typical for a state-of-the-art synchronous methodology.

In addition to design flow and manufacturing process uncertainties, modern circuit-level power minimization techniques, such as Dynamic Voltage Scaling and Adaptive Body Biasing, deliberately introduce performance variability. Changing the clock frequency in order to match performance with scaled supply voltage is very difficult since it multiplies the complexity of timing analysis by the number of voltage steps, and variability impact at low voltages is much more significant. Doing the same in the presence of varying body biasing, and hence varying threshold voltages, is even more complex. Moreover, phase-locked loops provide limited guarantees about periods and phases during frequency changes, hence the clock must be stopped while frequency is stepped up or down.

It is well-known that, under normal operating conditions, the delay of a CMOS circuit scales linearly with its voltage supply, while its power scales quadratically. Thus the normalized energy-per-cycle or energy-per-operation efficiency measure scales linearly with voltage supply. However, it is very difficult to use this optimization opportunity to the extreme by operating very close to the threshold voltage.

Two approaches have been proposed in the literature to tackle this problem with purely synchronous means. Both are based on sampling the output of a signal which is forced to make a transition very close to the clock cycle, and slow down the clock frequency or increase the voltage supply if this critical sampling happens at the current voltage and frequency conditions.

The Razor CPU [25] is designed with double slave latches and an XOR in each master-slave pair (thus increasing by over 100% the area of each converted latch). The second slave is clocked half a clock cycle later than the first slave. When the comparator detects a difference in values between the slaves, the inputs must have changed very close to the falling edge of the clock of the first slave, and the latch memorized an incorrect value. The Razor in that case "skips a beat" and restarts the pipeline with the value of the second latch, which is always (assuming that environmental conditions change slowly) latched correctly. An external controller always keeps voltage and clock frequency very close to this "critical clocking" condition in order to operate the processor very close to the best Vdd point for the required clock frequency, under the current temperature conditions.

The approach, while very appealing for processors, has an inherent problem that makes it not applicable to ASICs. Due to the near-critical clocking, it is always possible that the first latch goes meta-stable. In that case, the whole detector and the clock controller may suffer from meta-stability problems. That case is detected with analogue mechanisms, and the whole pipeline is flushed and restarted. This is easy to do in a processor, for which flushing and restarting is already part of a modern micro-architecture. However, it is very difficult, if not impossible, to achieve the same objective automatically for a generic logic circuit.

Another technique that has been proposed is to dynamically monitor the delay of the logic at the current voltage value and adjust the clock frequency accordingly (the PowerWise^{TM} technology from National Semiconductors). It samples, with a high frequency clock, the output of a digital delay line that toggles once per system clock cycle. This is used, more or less as in Razor, to measure the delay of the line in the current environment conditions (temperature, Vdd etc.). The scheme is safer than Razor, because it allows one to insert enough synchronizers after the delay line to reduce the meta-stability danger. However, it is an indirect measure, and requires a complicated (patented) logic to monitor and control the clock speed.

Asynchronous implementation, as demonstrated, e.g., in [52, 80, 90], achieves similar goals with much simpler logic, because the delay of the logic is directly used to generate the synchronization signals in a feedback control fashion.

On the other hand, several kinds of applications, in particular most of those that use complex processor architectures for part of the computation (e.g., general purpose computing and multi-media), and several others that are tolerant to environmental variations (e.g., wireless communications), do not have to obey strict timing constraints at all times. The widespread use of caches, the difficulty of tight worst-case execution time analysis for software, and the use multi-tasking kernels, require the use of DSP algorithms that are tolerant to internal performance variations and offer only average case guarantees. Hence, a design style in which the device provides average case guarantee, but may occasionally go slower or faster is quite acceptable for many applications. If the performance of that device on average is twice that of a traditionally designed device, then the performance advantage is significant enough to make a limited change in the design flow acceptable.

1.3 Asynchronous Design

Asynchronous design can be viewed as a method to introduce *feedback control* for synchronization of latches and flip–flops in a digital design. Asynchronous circuits measure, rather than estimate, the delay of the combinational logic, of the wires, and of the memorization elements. Handshaking between controllers that generate the clock signal for groups of flip–flops and latches ensures the satisfaction of both setup and hold constraints as illustrated in Figure 1.2.

Asynchronous circuits also reduce electromagnetic emission with respect to equivalent synchronous ones [53, 82] because they reduce the power consumption peaks in the vicinity of clock edges. Hence they produce a flatter power spectrum and exhibit smaller voltage supply drops.

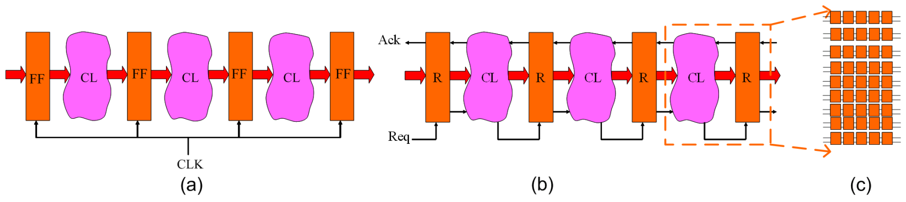

Fig. 1.2 Synchronous–Asynchronous Direct Translation: from synchronous (a) to desynchronized (b) and fine-grain pipelined (c) circuits.

Asynchronous circuits offer two kinds of power-related advantages. First, they provide a very fine-grained control over activation of both datapath and storage resources in a manner that is similar to clock gating but much easier to describe, verify, and implement robustly at the circuit level. Second, they reliably operate at very low voltages (even below the transistor threshold), when device characteristics exhibit second and third order effects. Synchronous operation becomes virtually impossible under these conditions [90] because:

- Library cells are seldom characterized by the manufacturer at such extreme operating conditions. Hence the normal synchronous ASIC design flow is unsuitable to guarantee correct operation.
- The transistor electrical models deviate significantly from those used under nominal conditions and make a straightforward scaling of performance and power impossible, or at least very risky.
- The effects of various random or hard-to-predict phenomena, such as threshold voltage variations, wire width variations, and local voltage supply variations due to IR drop, are dramatically magnified.

All this means that, even if one were able to use the traditional synchronous flow for circuits that will operate at a voltage supply that is close to or even below the transistor threshold voltage the performance margins that one would have to use to ensure correct operation would be huge. Robustness of asynchronous circuits to delay variations allows them to run at very low voltage levels. Recently, Achronix reported that an asynchronous FPGA chip, built in 90 nm CMOS, reliably operates with a supply of just 0.2 V and exhibits an 87% power consumption reduction by scaling the supply voltage from 1.2 V to 0.6 V [2].

1.4 An Overview of Asynchronous Design Styles

Clocking is a common, simple abstraction for representing the timing issues in the behavior of real circuits. Generally speaking, it lets designers ignore timing when considering functionality. Designers can describe both the functions performed and the circuits themselves in terms of logical equations (Boolean algebra). In general, synchronous designers do not need to worry about the exact sequence of gate switching as long as the outputs are correct at the clock pulses.

In contrast, asynchronous circuits must strictly coordinate their behavior. Logic synthesis for asynchronous circuits not only must handle circuit functionality but must also properly order gate activity (switching). The solution is to use functional redundancy to explicitly model computation flows without using abstract means such as clocks. Using logic to ensure correct circuit behavior under any delay distribution can be costly and impractical. Therefore, most asynchronous design styles use some timing assumptions to correctly coordinate and synchronize computations.

These assumptions can have different degrees of locality, from matching delays on some fanout wires, to making sure that a set of logic paths are faster than others. Localized assumptions are easier to meet in a design because they simplify timing convergence and provide more modularity. But ensuring the correctness of such assumptions can be costly because it requires more system redundancy at the functional level. Asynchronous design styles differ in the way they handle the trade-off between locality of timing assumptions [121] and design cost.

The main asynchronous design flows rely on the following assumptions:

- *Delay-insensitive* (*DI*) circuits [86] impose no timing assumptions, allowing arbitrary gate and wire delays. Unfortunately, the class of DI implementations is limited and impractical [77].

- *Quasi-delay-insensitive* (*QDI*) circuits [76] partition wires into critical and noncritical categories. Designers of such circuits consider fanout in critical wires to be safe by assuming that the skew between wire branches is less than the minimum gate delay. Designers thus assume these wires, which of course must be constrained to physically lie within a small area of the chip, to be isochronic. In contrast, noncritical wires can have arbitrary delays on fanout branches.
- *Bundled-delay* (*BD*) circuits assume that the maximum delay of each combinational logic island is smaller than that of a reference logic path (usually called matched delay) [121]. Matched delays are implemented using the same gate library as the rest of the datapath and they are subject to the same operating conditions (temperature, voltage). This results in consistent tracking of datapath delays by matched delays and allows one to reduce design margins with respect to synchronous design (bundled-data protocols).

As Figure 1.3 shows, the locality of timing assumptions decreases, from DI systems (which make no global assumptions) to synchronous circuits.

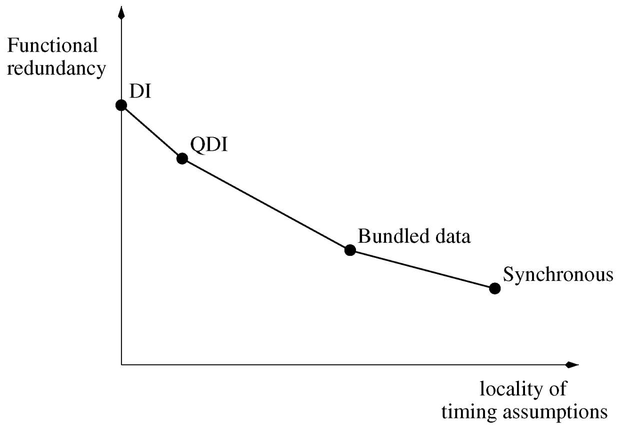

Fig. 1.3 Functional redundancy and locality of timing assumptions in asynchronous designs.

The imposed timing assumptions help in differentiating asynchronous implementations. For further categorizing of asynchronous design flows one needs to find out how the following key issues are addressed: (a) which way a designer expresses his/her intents, i.e., a design specification and (b) which way a designer proceeds with synthesis. The spectrum of the flow described in this paper ranges from think asynchronously — implement asynchronously to think synchronously — implement almost synchronously. Table 1.1 gives a high-level picture of the observed flows.

1.5 Asynchronous Design Flows

The ability to specify a design at a relatively high level, roughly equivalent to Register Transfer Level (RTL), is essential in order to enable enough designer productivity today. Two basic approaches have been proposed in the asynchronous domain for this purpose.

(1) The first one is to use an asynchronous HDL, generally based on the formal model of Communicating Sequential Processes (CSP [44]), because it fits very well the underlying asynchronous implementation fabrics [75, 134]. By nature, asynchronous circuits are highly concurrent and communication between modules is based on handshake channels (local synchronization). Haste, the design language used by the flow in Chapter 2, offers a syntax similar to behavioral Verilog, and in addition has explicit constructs for parallelism, communication, and hardware sharing. Starting from Haste, one can compile to behavioral Verilog for functional verification.

    Its main forte is the ability to achieve all the advantages of asynchronous implementation, e.g., by controlling the activation of both control and datapath units (and hence power and energy consumption) at a very fine level with direct and explicit HDL support.

    Several groups tried to combine RTL with handshake channel based specifications, particularly, to add channel as an extension of standard HDL (e.g., Verilog or VHDL) [103,

Table 1.1 High-level view on presented fully-automated design flows.

Design flow	Design style	Specification style	Type of synthesis	Implementation library	Summary
Haste	From QDI to Bundled Data	Asynchronous high-level and RTL (CSP-based)	Asynchronous	Asynchronous DesignWare mapped to standard cells	Think asynchronously — implement asynchronously
NCL	QDI	Synchronous RTL for datapath, asynchronous for control	Synchronous + scripts to map into async. library	Custom NCL. Possible to extend to standard cells	Think asynchronously — implement almost synchronously
Desync.	Bundled Data	Synchronous RTL	Synchronous + scripts to implement local clocking	Standard cells	Think synchronously — implement almost synchronously
Fine grain pipeline (Weaver)	QDI	Synchronous RTL	Synchronous + scripts to map into pipeline cells	Custom (dynamic logic) Possible to extend to standard cells	Think synchronously — implement asynchronously

108, 109] that could be considered as HDL level automation of Caltech group's ideas [79]. Unfortunately this approach which is attractive from theoretical point of view requires reeducation of RTL designers and rewriting of existing specifications which is not realistic for big and expensive designs.

(2) The other approach, that we call Synchronous–Asynchronous Direct Translation (SADT), starts from a synchronous synthesizable specification, translates it into a gate-level netlist using traditional logic synthesis tools, and then applies a variety of techniques to translate it into as asynchronous implementation [20, 68, 70, 117].

Its main advantage is to allow reimplementation of legacy designs without any designer intervention at the HDL level. Since eliminating logic bugs takes up to 50% of design time, this can potentially translate into a significant advantage in terms of design time and cost, with respect to approaches that require a significant redesign. This approach is represented by the following design styles.

(a) The Null-Convention Logic (NCL), that is used by Theseus Logic and is described in Chapter 3, is based on a proprietary library of majority gates and produces fully delay-insensitive circuits. This has very high overhead (about 3–4× in terms of area), but is also very robust, because functional correctness is achieved independent of gate and wire delays.

(b) De-synchronization, that is described in Chapter 4, uses a micropipeline-style architecture and an interconnection of small handshaking controllers to derive the clock for groups of up to 100 latches (derived by splitting synchronous flip–flops). It has very low overhead in terms of area (between 5% and 10%), but requires careful matching of delays all the way down through physical design. Note that the same implementation architecture is used by the Handshake Solution flow of Chapter 2, starting from the

Haste language, thus easing interfacing between logic blocks produced by one of these two flows.

(c) Automated fine-grain pipelining presented in Chapter 5. In the finest granularity (gate-level pipelining Figure. 1.2(c) in addition to replacing global synchronization with local handshake control this flow also removes functionally unnecessary synchronization. The flow offers support for automated pipelining therefore it is targeted to improve the original design performance. In this flow [115, 117] by default pipelining is done in the finest degree resulting in high-performance. The flow can exploit the use of aggressive pipelining to reduce the performance gap while maintaining low power. For example, for a standard cell library developed using 180 nm TSMC process the fine-grain pipelined cells are functional with VDDs down to 0.6 V. FIFO performance of 780 MHz at nominal 1.8 V drops to 135 MHz at the safe 0.8 V with 14.2× lower power consumption.

We believe that synchronous–asynchronous directed translation model could play for asynchronous design automation as important of a role as RTL did for synchronous EDA. The main contribution to EDA by RTL model is due to a separation of optimization and timing (all sequential behavior is in an interaction between registers, all synthesis and optimization are only about combinational clouds). The key idea that enables SADT flows is as follows. The RTL model (Figure 1.2(a)) is based on global synchronization and timing assumptions (computations complete in every stage before the next clock edge). During every clock cycle every latch undergoes two phases: pass and store. Master–slave flip–flops prevent the register from being transparent at any given time, but introduce the requirement to carefully control clock skew in order to avoid double-clocking failures (also called hold time violations), which are especially nasty because they cannot be fixed simply by slowing down the clock. Dynamic logic also has a two-phase operation: evaluate and precharge (reset). These phases naturally map to asynchronous handshake (request–acknowledge) protocols [121].

The separation into phases enables, in asynchronous just as in synchronous design, a clean separation between functionality and timing. A datapath implemented using any of the techniques described in this paper behaves just like combinational logic (and is in fact just plain combinational logic in the Haste and the desynchronization flows) during evaluation. A resetting or precharge phase is used in the NCL and fine-grain pipelining flows to ensure reliable delay-insensitive or high-speed operation. In other words, asynchronous implementation does not change the externally observable behavior, and the sequence of values that appears on the boundary registers is the same before and after the application of SADT.

Among other advantages (e.g., in terms of reuse of functional and timing verification simulation vectors), this enables easy interfacing with synchronous logic. The latter could be achieved by driving the clocks of the synchronous blocks by the request signals coming from the asynchronous blocks if the following two assumptions are satisfied:

- Each synchronous block has an interface whose fanin and fanout registers are all clocked by a single asynchronous controller.
- Each synchronous block is faster than the asynchronous one that drives its clock.

For this reason, synchronous legacy logic can be used unchanged in an asynchronous fabric if it is non-critical.

This is very different from GALS-based methods, which require the use of wrappers and introduce significant system-level integration problems due to the need of creating ad-hoc manual handshaking between blocks that belong to different clock domains.

The various SADT flows described in this paper differ in the granularity of the pipeline stages. The NCL and desynchronization approaches retain exactly the same position of registers, and hence pipelining level, as the original synchronous specification. Gate-level pipelining, on the other hand, significantly decreases the granularity of the pipelining, down to the gate level, as shown in Figure 1.2(c).

All the approaches considered in this paper share several common characteristics. First of all, control is derived through a syntax-directed translation from a specification, whether it is written in Haste or in synthesizable Verilog. Second, the datapath is generated (at least initially, for the NCL and fine-grain pipelining flows) using traditional logic synthesis techniques, starting from design libraries such as DesignWare [22]. Third, physical design and implementation verification (equivalence checking, extraction, back-annotation etc.) are essentially unchanged. Finally, testing of the datapath and of the controllers is performed mostly synchronously thanks to the fact that timing faults in the controller networks are easy to detect with simple functional tests. Hence design for testability and automated test pattern generation tools and techniques can be reused almost without change.

We present a variety of automated flows (Haste and the flows related to SADT approach) because we believe that the full advantages of asynchronous design to tackle power, energy, variability, and electromagnetic emission will come from a judicious mix of:

- Full redesigning performance and power-critical widely used components (e.g. microprocessors) using a language like Haste.
- Converting synchronous designs of special-purpose critical modules to asynchronous implementations, using one of the SADT techniques outlined in this paper. The choice of method will depend on whether the main goal is robustness, cost or performance.
- Leaving non-critical components as synchronous and clocking them with the handshake signals produced by the asynchronous interface controllers.

1.6 Paper Organization

We will start our review by considering in Chapter 2 the most radical design approach that has been applied so far to real-life designs. It is based on the Haste language, and commercialized by Handshake

Solutions. It is radical since it starts from non-standard (asynchronously specific model) and needs some specific education for designers. However, this non-standard HDL could lead to rather efficient solutions that could be unreachable from standard HDLs specifications. It was successfully used for real LARGE designs. Then we will describe the SADT approaches, starting from the pioneering NCL technique [57, 68] discussed in Chapter 3. NCL was the first approach to asynchronous design exploiting the idea of synthesizing large designs using commercial EDA tools. NCL circuits are dual-rail to enable completion detection. They are architecturally equivalent to the RTL implementation. However, full synchronization of completion detection at each register implies that NCL circuits are significantly slower and larger (by a factor of 2 to 4) than the synchronous starting point.

Reducing NCL overheads moves us to de-synchronization [19, 20], in Chapter 4, which uses delay matching in order to achieve a good compromise between robustness, performance, and cost. When run at their worst-case speed, desynchronized designs exhibit an almost negligible overhead with respect to synchronous ones. On the other hand, when run at the actual speed at the process, voltage, and temperature conditions. They can dramatically reduce the delay margins required by synchronous design.

None of the above approaches offers support for automated pipelining, therefore they cannot directly improve this aspect of the performance equation. Gate-level pipelining [117, 118], on the other hand, can pipeline at the level of individual gates, thus achieving performance levels that are virtually impossible to match with synchronous designs. We present this flow in Chapter 5.

Chapter 6 is dedicated to design examples that illustrate both the achievable results and the possible application areas of the various design flows. Finally, Chapter 7 presents some conclusions on the opportunities offered by asynchronous circuits and flows.

References

[1] A. Abrial, J. Bouvier, M. Renaudin, P. Senn, and P. Vivet, “A new contactless smart card IC using an on-chip antenna and an asynchronous microcontroller,” *IEEE Journal of Solid-State Circuits*, vol. 36, no. 7, pp. 1101–1107, 2001.

[2] Achronix Semiconductor, www.achronix.com, 2006.

[3] A. Agarwal, D. Blaauw, V. Zolotov, and S. Vrudhula, “Statistical timing analysis using bounds,” in *Design, Automation and Test in Europe (DATE)*, pp. 10062–10067, 2003.

[4] N. Andrikos, *A Fully Automated Desynchronization Flow for Synchronous Circuits*. PhD thesis, University of Crete, 2006.

[5] M. Baleani, F. Gennari, Y. Jiang, Y. Patel, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, “HW/SW partitioning and code generation of embedded control applications on a reconfigurable architecture platform,” in *Proceedings of the Tenth International Symposium on Hardware/Software Codesign, CODES 2002*, pp. 151–156, Estes Park, Colorado, USA, May 6–8 2002.

[6] A. Bardsley, *Implementing Balsa Handshake Circuits*. PhD thesis, Department of Computer Science, University of Manchester, 2000.

[7] P. Beerel, J. Cortadella, and A. Kondratyev, “Bridging the gap between asynchronous design and designers (Tutorial),” in *VLSI Design Conference*, (Mumbai), 2004.

[8] P. A. Beerel, M. Davies, A. Lines, and N.-H. Kim, “Slack matching asynchronous designs,” in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 184–194, March 2006.

[9] L. Benini, E. Macii, and M. Poncino, "Telescopic units: Increasing the average throughput of pipelined designs by adaptive latency control," in *Proc. ACM/IEEE Design Automation Conference*, pp. 22–27, 1997.

[10] Celoxica, *Handel-C Language Reference Manual*. Celoxica. 2003.

[11] CeltIC signal integrity analysis, www.cadence.com/products.

[12] T. Chelcea and S. M. Nowick, "Resynthesis and peephole transformations for the optimization of large-scale asynchronous systems," in *Proc. ACM/IEEE Design Automation Conference*, June 2002.

[13] D. Chinnery and K. Keutzer, *Closing the Gap between ASIC & Custom. Tools and Techniques for Gigh-Performance ASIC Design*. Kluwer Academic Publishers, 2002.

[14] T.-A. Chu, C. K. C. Leung, and T. S. Wanuga, "A design methodology for concurrent VLSI systems," in *Proc. International Conf. Computer Design (ICCD)*, pp. 407–410, IEEE Computer Society Press, 1985.

[15] W. S. Coates, J. K. Lexau, I. W. Jones, S. M. Fairbanks, and I. E. Sutherland, "FLEETzero: An asynchronous switch fabric chip experiment," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 173–182, IEEE Computer Society Press, March 2001.

[16] F. Commoner, A. W. Holt, S. Even, and A. Pnueli, "Marked directed graphs," *Journal of Computer and System Sciences*, vol. 5, pp. 511–523, 1971.

[17] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev, "Petrify: A tool for manipulating concurrent specifications and synthesis of asynchronous controllers," *IEICE Transactions on Information and Systems*, vol. E80-D, no. 3, pp. 315–325, March 1997.

[18] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev, *Logic Synthesis of Asynchronous Controllers and Interfaces*. Springer-Verlag, 2002.

[19] J. Cortadella, A. Kondratyev, L. Lavagno, K. Lwin, and C. Sotiriou, "From synchronous to asynchronous: An automatic approach," in *Proc. Design, Automation and Test in Europe (DATE)*, pp. 1368–1369, February 2004.

[20] J. Cortadella, A. Kondratyev, L. Lavagno, and C. Sotiriou, "Desynchronization: Synthesis of asynchronous circuits from synchronous specifications," *IEEE Transactions on Computer-Aided Design*, vol. 25, no. 10, pp. 1904–1921, October 2006.

[21] M. de Wit and A. Peeters, "Haste language reference manual," Tech. Rep., 2006.

[22] Designware, intellectual property: www.synopsys.com/products/designware/.

[23] D. Edwards and A. Bardsley, "Balsa: An asynchronous hardware synthesis language," *The Computer Journal*, vol. 45, no. 1, pp. 12–18, 2002.

[24] S. A. Edwards, "The challenges of hardware synthesis from c-like languages," in *DATE '05: Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 66–67, 2005.

[25] D. Ernst, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, N. S. Kim, and K. Flautner, "Razor: Circuit-level correction of timing errors for low-power operation," *IEEE Micro*, November 2004.

[26] K. M. Fant, *Logically Determined Design: Clockless System Design with NULL Convention Logic*. John Wiley & Sons, 2005.

[27] K. M. Fant and S. A. Brandt, “NULL conventional logic: A complete and consistent logic for asynchronous digital circuit synthesis,” in *International Conference on Application-specific Systems, Architectures, and Processors*, pp. 261–273, 1996.

[28] M. Ferretti and P. A. Beerel, “Single-track asynchronous pipeline templates using 1-of-N encoding,” in *Proc. Design, Automation and Test in Europe (DATE)*, pp. 1008–1015, March 2002.

[29] M. Ferretti, R. Ozdag, and P. Beerel, “High performance asynchronous ASIC back-end design flow using single-track full-buffer standard cells,” in *Proc. International Symposium on AdvancedResearch in Asynchronous Circuits and Systems*, pp. 95–105, IEEE Computer Society Press, April 2004.

[30] FIPS PUB 197: advanced encryption standard, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[31] First clockless processor for real-time chip designs – http://www.handshakesolutions.com/News/.

[32] R. M. Fuhrer and S. M. Nowick, *Sequential Optimization of Asynchronous and Synchronous Finite-State Machines: Algorithms and Tools*. Kluwer Academic Publishers, 2001.

[33] R. M. Fuhrer, S. M. Nowick, M. Theobald, N. K. Jha, B. Lin, and L. Plana, “Minimalist: An environment for the synthesis, verification and testability of burst-mode asynchronous machines,” Tech. Rep. TR CUCS-020-99, Columbia University, NY, July 1999.

[34] Fulcrum microsystems www.fulcrummicro.com/.

[35] S. B. Furber and P. Day, “Four-phase micropipeline latch control circuits,” *IEEE Transactions on VLSI Systems*, vol. 4, no. 2, pp. 247–253, June 1996.

[36] J. D. Garside, W. J. Bainbridge, A. Bardsley, D. A. Edwards, S. B. Furber, J. Liu, D. W. Lloyd, S. Mohammadi, J. S. Pepper, O. Petlin, S. Temple, and J. V. Woods, “AMULET3i — an asynchronous system-on-chip,” in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 162–175, IEEE Computer Society Press, April 2000.

[37] R. Goering, “Simple designs aren’t easy, speaker says,” *EE Times,* http://www.eetimes.com/showArticle.jhtml?articleID=184400784, no. 03/28/2006, 2006.

[38] D. Harris, *Skew-Tolerant Circuit Design*. Morgan Kaufmann Publishers, 2001.

[39] A. Hartstein and T. R. Puzak, “Optimum power/performance pipeline depth,” in *MICRO-36 International Symposium on Microarchitecture*, 2003.

[40] P. J. Hazewindus, *Testing Delay-Insensitive Circuits*. PhD thesis, California Institute of Technology, 1992.

[41] J. L. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publisher Inc., 1990.

[42] E. Hess, N. Janssen, B. Meyer, and T. Schutze, “Information leakage attacks against smart card implementations of cryptographic algorithms and countermeasures — a survey,” in *EUROSMART Security Conference*, 2000.

[43] C. A. R. Hoare, "Communicating sequential processes," *Communications of the ACM*, vol. 21, no. 8, pp. 666–677, August 1978.

[44] C. A. R. Hoare, *Communicating Sequential Processes.* Prentice-Hall, 1985.

[45] A. Hodjat, D. D. Hwang, B. Lai, K. Tiri, and I. Verbauwhede, "A 3.84 gbits/s AES crypto coprocessor with modes of operation in a 0.18-um CMOS technology," in *15th ACM Great Lakes symposium on VLSI*, (Chicago, Illinois, USA), pp. 60–63, 2005.

[46] M. S. Hrishikesh, N. P. Jouppi, K. I. Farkas, D. Burger, S. W. Keckler, and P. Shivakumar, "The optimal depth per pipeline stage is 6 to 8 fo4 inverter delays," in *29th Int'l Symp. Computer Architecture*, pp. 14–24, IEEE CS Press, 2002.

[47] H. Hulgaard, S. M. Burns, and G. Borriello, "Testing asynchronous circuits: A survey," *Integration, the VLSI Journal*, vol. 19, no. 3, pp. 111–131, November 1995.

[48] C. Jeong and S. M. Nowick, "Optimal technology mapping and cell merger for asynchronous threshold networks," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 128–137, March 2006.

[49] J. Kahle, "The myth of the optimal fo4," in *ACM/IEEE TAU Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, 2004.

[50] N. Karaki, T. Nanmoto, H. Ebihara, S. Utsunomiya, S. Inoue, and T. Shimoda, "A flexible 8b asynchronous microprocessor based on low-temperature poly-silicon TFT technology," in *International Solid State Circuits Conference*, pp. 272–274, February 2005.

[51] C. H. (Kees) van Berkel, M. B. Josephs, and S. M. Nowick, "Scanning the technology: Applications of asynchronous circuits," *Proceedings of the IEEE*, vol. 87, no. 2, pp. 223–233, February 1999.

[52] C. Kelly, V. Ekanayake, and R. Manohar, "SNAP: A sensor-network asynchronous processor," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 24–33, IEEE Computer Society Press, May 2003.

[53] J. Kessels and A. Peeters, "The Tangram framework: Asynchronous circuits for low power," in *Proc. of Asia and South Pacific Design Automation Conference*, pp. 255–260, February 2001.

[54] J. Kessels, A. Peeters, T. Kramer, M. Feuser, and K. Ully, "Designing an asynchronous bus interface," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 108–117, IEEE Computer Society Press, March 2001.

[55] R. Kol and R. Ginosar, "A doubly-latched asynchronous pipeline," in *Proc. International Conf. Computer Design (ICCD)*, pp. 706–711, October 1996.

[56] T. Kolks, S. Vercauteren, and B. Lin, "Control resynthesis for control-dominated asynchronous designs," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, March 1996.

[57] A. Kondratyev and K. Lwin, "Design of asynchronous circuits using synchronous CAD tools," *IEEE Design & Test of Computers*, vol. 19, no. 4, pp. 107–117, 2002.

[58] A. Kondratyev, L. Neukom, O. Roig, A. Taubin, and K. Fant, "Checking delay-insensitivity: 10^4 gates and beyond," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 149–157, April 2002.

[59] A. Kondratyev, L. Sorensen, and A. Streich, "Testing of asynchronous designs by inappropriate means. synchronous approach," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 171–180, April 2002.

[60] K. Kulikowski, M. Karpovsky, and A. Taubin, "Power attacks on secure hardware based on early propapation of data," in *12th IEEE International On-Line Testing Symposium*, 2006.

[61] K. Kulikowski, A. Smirnov, and A. Taubin, "Automated design of cryptographic devices resistant to multiple side-channel attacks," in *Chyptographic Hardware and Embedded Systems CHES*, (Yokohama), 2006.

[62] K. J. Kulikowski, M. Su, A. Smirnov, A. Taubin, M. G. Karpovsky, and D. MacDonald, "Delay insensitive encoding and power analysis: A balancing act," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 116–125, 2005.

[63] T. Larrabee, "Test pattern generation using boolean satisfiability," *IEEE Transactions on Computer-Aided Design*, vol. 11, no. 1, pp. 4–15, 1992.

[64] L. Lavagno and S. M. Nowick, "Asynchronous control circuits," in *Logic Synthesis and Verification*, pp. 255–284, Kluwer Academic Publishers, 2002.

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

[66] H. Li, A. Markettos, and S. W. Moore, "Security evaluation against electromagnetic analysis at design time," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2005.

[67] Liberty CCS: www.synopsys.com/products/libertyccs/libertyccs.html.

[68] M. Ligthart, K. Fant, R. Smith, A. Taubin, and A. Kondratyev, "Asynchronous design using commercial HDL synthesis tools," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 114–125, IEEE Computer Society Press, April 2000.

[69] D. H. Linder, *Phased Logic: A Design Methodology for Delay-Insensitive, Synchronous Circuitry*. PhD thesis, Mississippi State University, 1994.

[70] D. H. Linder and J. C. Harden, "Phased logic: Supporting the synchronous design paradigm with delay-insensitive circuitry," *IEEE Transactions on Computers*, vol. 45, no. 9, pp. 1031–1044, September 1996.

[71] A. Lines, *Pipelined Asynchronous Circuits*. PhD thesis, California Institute of Technology, 1998. (CaltechCSTR:1998.cs-tr-95-21).

[72] A. Lines, "Nexus: An asynchronous crossbar interconnect for synchronous system-on-chip designs," in *Proceedings of the 11th Symposium on High Performance Interconnects*, pp. 2–9, August 2003.

[73] D. J. MacDonald, *MS Thesis. A Balanced-Power Domino-Style Standard Cell Library for Fine-Grain Asynchronous Pipelined Design to Resist Differential Power Analysis Attacks*. PhD thesis, Boston University, 2005.

[74] R. Manohar and A. J. Martin, "Slack elasticity in concurrent computing," in *Proc. 4th International Conference on the Mathematics of Program Construction*, (J. Jeuring, ed.), pp. 272–285, 1998.

[75] A. J. Martin, "Programming in VLSI: From communicating processes to delay-insensitive circuits," in *Developments in Concurrency and Communication*, (C. A. R. Hoare, ed.). UT Year of Programming Series, pp. 1–64, Addison-Wesley, 1990.

[76] A. J. Martin, "Compiling communicating processes into delay-insensitive VLSI circuits," *Distributed Computing*, vol. 1, no. 4, pp. 226–234, 1986.

[77] A. J. Martin, "The limitations to delay-insensitivity in asynchronous circuits," in *Advanced Research in VLSI*, (W. J. Dally, ed.), pp. 263–278, MIT Press, 1990.

[78] A. J. Martin and P. J. Hazewindus, "Testing delay-insensitive circuits," in *Advanced Research in VLSI*, (C. H. Séquin, ed.), pp. 118–132, MIT Press, 1991.

[79] A. J. Martin, A. Lines, R. Manohar, M. Nyström, P. Pénzes, R. Southworth, and U. Cummings, "The design of an asynchronous MIPS R3000 microprocessor," in *Advanced Research in VLSI*, pp. 164–181, September 1997.

[80] A. J. Martin, M. Nyström, K. Papadantonakis, P. I. Pénzes, P. Prakash, C. G. Wong, J. Chang, K. S. Ko, B. Lee, E. Ou, J. Pugh, E.-V. Talvala, J. T. Tong, and A. Tura, "The lutonium: A sub-nanojoule asynchronous 8051 microcontroller," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 14–23, IEEE Computer Society Press, May 2003.

[81] J. Martin Alain and M. Nystrom, "Asynchronous techniques for system-on-chip design," *Proceedings of the IEEE*, vol. 94, no. 6, pp. 1089–1120, June 2006.

[82] J. McCardle and D. Chester, "Measuring an asynchronous processor's power and noise," in *SNUG*, 2001.

[83] C. E. Molnar, I. W. Jones, B. Coates, and J. Lexau, "A FIFO ring oscillator performance experiment," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 279–289, IEEE Computer Society Press, April 1997.

[84] S. Moore, R. Anderson, R. Mullins, G. Taylor, and J. J. A. Fournier, "Balanced self-checking asynchronous logic for smart card applications," *Microprocessors and Microsystems*, vol. 27, no. 9, pp. 421–430, October 2003.

[85] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an efficient SAT solver," in *Proc. ACM/IEEE Design Automation Conference*, 2001.

[86] D. E. Muller, "Asynchronous logics and application to information processing," in *Symposium on the Application of Switching Theory to Space Technology*, pp. 289–297, Stanford University Press, 1962.

[87] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–574, April 1989.

[88] C. Myers, *Asynchronous Circuit Design*. John Wiley & Sons, 2001.

[89] S. R. Nassif, "Modeling and forecasting of manufacturing variations," in *Asia-South Pacific Design Automation Conference (ASP-DAC)*, 2001.

[90] L. Necchi, L. Lavagno, D. Pandini, and L. Vanzago, "An ultra-low energy asynchronous processor for wireless sensor networks," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 78–85, March 2006.

[91] . Overview Datasheet -high performance aes (rijndael) cores for asic http://www.heliontech.com/downloads/aes_asic_helioncore.pdf.

[92] R. O. Ozdag and P. A. Beerel, "High-speed QDI asynchronous pipelines," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 13–22, April 2002.

[93] D. Patterson, "The future of computer architecture. berkeley eecs annual research symposium 2006, http://www.eecs.berkeley.edu/BEARS/presentations/06Patterson.ppt," 2006.

[94] A. Peeters, "Implementation of handshake components," in *Comunicating Sequential Processes, the First 25 Years,* Volume 3525 of *Lecture Notes in Computer Science*, (A. E. Abdallah, C. B. Jones, and J. W. Sanders, eds.), pp. 98–132, 2005.

[95] A. Peeters and K. van Berkel, "Synchronous handshake circuits," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 86–95, IEEE Computer Society Press, March 2001.

[96] A. M. G. Peeters, *Single-Rail Handshake Circuits.* PhD thesis, Eindhoven University of Technology, June 1996.

[97] F. Peper, J. Lee, S. Adachi, and S. Mashiko, "Laying out circuits on asynchronous cellular arrays: A step towards feasible nanocomputers?," *Nanotechnology*, vol. 14, pp. 469–485, 2003.

[98] O. A. Petlin and S. B. Furber, "Scan testing of micropipelines," in *Proc. IEEE VLSI Test Symposium*, pp. 296–301, May 1995.

[99] L. A. Plana, P. A. Riocreux, W. J. Bainbridge, A. Bardsley, J. D. Garside, and S. Temple, "SPA — A synthesisable amulet core for smartcard applications," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 201–210, April 2002.

[100] P. Prakash and A. J. Martin, "Slack matching quasi delay-insensitive circuits," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 195–204, March 2006.

[101] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits.* Prentice-Hall, 2nd ed., 2002.

[102] R. B. Reese and M. A. T. C. Traver, "A coarse-grain phased logic CPU," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 2–13, IEEE Computer Society Press, May 2003.

[103] M. Renaudin, P. Vivet, and F. Robin, "A design framework for asynchronous/synchronous circuits based on CHP to HDL translation," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 135–144, April 1999.

[104] M. Roncken and R. Saeijs, "Linear test times for delay-insensitive circuits: A compilation strategy," in *Asynchronous Design Methodologies*, (S. Furber and M. Edwards, eds.), pp. 13–27, Elsevier Science Publishers, 1993.

[105] M. Roncken, "Defect-oriented testability for asynchronous IC's," *Proceedings of the IEEE*, vol. 87, no. 2, pp. 363–375, February 1999.

[106] L. Y. Rosenblum and A. V. Yakovlev, "Signal graphs: From self-timed to timed ones," in *Proceedings of International Workshop on Timed Petri Nets*, (Torino, Italy), pp. 199–207, IEEE Computer Society Press, July 1985.

[107] S. Rotem, K. Stevens, R. Ginosar, P. Beerel, C. Myers, K. Yun, R. Kol, C. Dike, M. Roncken, and B. Agapiev, "RAPPID: An asynchronous instruction length decoder," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 60–70, April 1999.

[108] A. Saifhashemi and P. A. Beerel, "High Level modeling of channel-based asynchronous circuits using verilog," in *Communicating Process Architectures*, (J. F. Broenink *et al.*, ed.), pp. 275–288, IOS Press, September 2005.

[109] A. Saifhashemi and H. Pedram, "Verilog HDL, powered by PLI: A suitable framework for describing and modeling asynchronous circuits at all levels of abstraction," in *Proc. ACM/IEEE Design Automation Conference*, pp. 330–333, June 2003.

[110] Savant Project-http://www.cliftonlabs.com/savantp.htm.

[111] C. L. Seitz, "System timing," in *Introduction to VLSI Systems*, (C. A. Mead and L. A. Conway, eds.), ch. 7, Addison-Wesley, 1980.

[112] Sequence signal-integrity tools, www.sequencedesign.com.

[113] Silistix, self-timed interconnect technology, www.silistix.com.

[114] M. Singh, J. A. Tierno, A. Rylyakov, S. Rylov, and S. M. Nowick, "An adaptively-pipelined mixed synchronous-asynchronous digital FIR filter chip operating at 1.3 gigahertz," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 84–95, April 2002.

[115] A. Smirnov and A. Taubin, "Synthesizing asynchronous micropipelines with design compiler," in *Synopsys Users Group*, Boston, 2006.

[116] A. Smirnov, A. Taubin, and M. Karpovsky, "Automated pipelining in ASIC synthesis methodology: Gate transfer level," in *IWLS 2004 Thirteenth International Workshop on Logic and Synthesis*, 2004.

[117] A. Smirnov, A. Taubin, and M. Karpovsky, "An automated fine-grain pipelining using domino style asynchronous library," in *ACSD 2005: Fifth International Conference on Application of Concurrency to System Design*, (St.Malo, France), IEEE CS Press, 2005.

[118] A. Smirnov, A. Taubin, M. Karpovsky, and L. Rozenblyum, "Gate transfer level synthesis as an automated approach to fine-grain pipelining," in *Workshop on Token Based Computing (ToBaCo)*, (Bologna, Italy), 2004.

[119] G. E. Sobelman and K. Fant, "CMOS circuit design of threshold gates with hysteresis," in *Proc. International Symposium on Circuits and Systems*, pp. 61–64, June 1998.

[120] D. Sokolov, J. Murphy, A. Bystrov, and A. Yakovlev, "Design and analysis of dual-rail circuits for security applications," *IEEE Transactions on Computers*, vol. 54, no. 4, pp. 449–460, April 2005.

[121] J. Sparsø and S. Furber, eds., *Principles of Asynchronous Circuit Design: A Systems Perspective*. Kluwer Academic Publishers, 2001.

[122] J. Sparsø and J. Staunstrup, "Delay-insensitive multi-ring structures," *Integration, the VLSI Journal*, vol. 15, no. 3, pp. 313–340, October 1993.

[123] Special issue on asynchronous circuits and systems. *Proceedings of the IEEE*, vol. 87, no. 2, pp. 1–375, February 1999.

[124] I. Sutherland and S. Fairbanks, "GasP: A minimal FIFO control," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 46–53, IEEE Computer Society Press, March 2001.

[125] I. E. Sutherland, "Micropipelines," *Communications of the ACM*, vol. 32, no. 6, pp. 720–738, June 1989.

[126] F. te Beest and A. Peeters, "A multiplexer based test method for self-timed circuits," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 166–175, 2005.

[127] F. te Beest, A. Peeters, K. van Berkel, and H. Kerkhoff, "Synchronous full-scan for asynchronous handshake circuits," *Journal of Electronic Testing: Theory and Applications*, vol. 19, pp. 397–406, 2003.

[128] J. Tierno, A. Rylyakov, S. Rylov, M. Singh, P. Ampadu, S. Nowick, M. Immediato, and S. Gowda, "A 1.3 GSample/s 10-tap full-rate variable-latency self-timed FIR filter with clocked interfaces," in *International Solid State Circuits Conference*, February 2002.

[129] K. Tiri, M. Akmal, and I. Verbauwhede, "A dynamic and differential cmos logic with signal independent power consumption to withstand differential power analysis on smart cards," in *28th European Solid-State Circuits Conference (ESSCIRC 2002)*, 2002.

[130] K. Tiri, W. Hwang, A. Hodjat, L. Bo-Cheng, Y. Shenglin, P. Schaumont, and I. Verbauwhede, "Prototype IC with wddl and differential routing — dpa sesistance assessment," in *Chyptographic Hardware and Embedded Systems — CHES*, (Edinburgh), pp. 354–365, LNCS3659, Springer, 2005.

[131] K. Tiri and I. Verbauwhede, "Securing encryption algorithms against dpa at the logic level: Next generation smart card technology," *Workshop on Cryptographic Hardware and Embedded Systems (CHES 2003)*, 2003.

[132] K. Tiri and I. Verbauwhede, "A logic level design methodology for a secure dpa resistant asic or fpga implementation," *Design Automation and Test in Europe Conference (DATE 2004)*, 2004.

[133] TSMC 0.18mm Process 1.8-Volt SAGE-X TM Standard Cell Library Databook. September 2003.

[134] K. van Berkel, *Handshake Circuits: An Asynchronous Architecture for VLSI Programming*. Vol. 5 of *International Series on Parallel Computation*, Cambridge University Press, 1993.

[135] K. van Berkel and A. Bink, "Single-track handshaking signaling with application to micropipelines and handshake circuits," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 122–133, IEEE Computer Society Press, March 1996.

[136] K. van Berkel, F. Huberts, and A. Peeters, “Stretching quasi delay insensitivity by means of extended isochronic forks,” in *Asynchronous Design Methodologies*, pp. 99–106, IEEE Computer Society Press, May 1995.

[137] K. van Berkel, A. Peeters, and F. te Beest, “Adding synchronous and LSSD modes to asynchronous circuits,” *Microprocessors and Microsystems*, vol. 27, no. 9, pp. 461–471, October 2003.

[138] 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*, (Aizu-Wakamatsu, Japan), pp. 130–138, March 1995.

[139] V. I. Varshavsky, ed., *Self-Timed Control of Concurrent Processes: The Design of Aperiodic Logical Circuits in Computers and Discrete Systems*, (Dordrecht, The Netherlands), Kluwer Academic Publishers, 1990.

[140] G. Venkataramani, T. Bjerregaard, T. Chelcea, and S. C. Goldstein, “Hardware compilation of application-specific memory-access interconnect,” *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 25, no. 5, pp. 756–771, 2006.

[141] G. Venkataramani, M. Budiu, T. Chelcea, and S. Goldstein, “C to asynchronous dataflow circuits: An end-to-end toolflow,” in *IWLS*, pp. 501–508, Temecula, CA, June 2004.

[142] G. Venkataramani and S. Copen Goldstein, “Leveraging protocol knowledge in slack matching,” in *IEEE/ACM International Conference on Computer-Aided Design*, (San Jose, CA, USA), November 2006.

[143] T. Verhoeff, “Delay-insensitive codes—an overview,” *Distributed Computing*, vol. 3, no. 1, pp. 1–8, 1988.

[144] Weaver: GTL synthesis flow. http://async.bu.edu/weaver/.

[145] Y. Zhou, D. Sokolov, and A. Yakovlev, “Cost-aware synthesis of asynchronous circuits based on partial acknowledgement,” in *Proc. International Conf. Computer-Aided Design (ICCAD)*, pp. 255–260, IEEE Computer Society Press, November 2006.