

Narrowing the Margins with Elastic Clocks

Jordi Cortadella*, Luciano Lavagno[†], Djavad Amiri[‡], Jonàs Casanova*, Carlos Macián[§], Ferran Martorell[§], Juan A. Moya[§], Luca Necchi[§], Danil Sokolov[¶] and Emre Tuncer[‡]

*Universitat Politècnica de Catalunya, Barcelona, Spain.

[†]Politecnico di Torino, Italy.

[‡]Elastix Corp., Los Gatos, USA.

[§]Elastix Solutions S.L., Barcelona, Spain.

[¶]University of Newcastle, UK.

Abstract—The continuous shrinking of process geometries increases variability and demands for conservative margins that have a negative impact on performance. With conventional clocks, the cycle period has to be defined to accommodate the worst-case variations during the lifetime of the circuit. Elastic Clocks arise as a new paradigm to reduce the margins without sacrificing robustness. Their cycle-by-cycle adaptation to static and dynamic variability enables the use of reduced margins that only need to cover the differential variability of the circuit delays with regard to the elastic period. Given the substantial spatio-temporal correlation within every die, a significant reduction in the margins required to cover process variability, voltage and temperature fluctuations and aging can be achieved.

I. INTRODUCTION

Manufacturing process (“P”) and operating environment (“V, T”) variations can change the performance of digital integrated circuits by a factor of 2-3X, and we can foresee an order of magnitude difference between the fastest and slowest PVT conditions [5]. Even larger differences can already be observed for leakage power consumption. Aging also causes changes in performance over time, which are dependent on both transition frequencies and level probabilities of individual signals along critical paths. These tolerances are tackled in the current implementation flow by using *margins*, which force the designer to leave a huge amount of performance untapped, the equivalent of a couple of process generations. Of course, this waste is economically unacceptable, and several strategies, which are appropriate for different application domains, have been devised.

- 1) CPU and graphic processor manufacturers *bin* their chips according to their process-related *performance* variation, and sell them at different price points.
- 2) Low-power SOC designers use Adaptive Voltage Scaling to measure at runtime the PVT-related performance variations (including some aging effects). Since system performance usually determines the clock frequency, they turn the recovered performance margins into *power savings*, by scaling the voltage in order to match the current performance requirements.
- 3) Some researchers are reluctant to leave on the table the margins due to the fact the performance monitors react to PVT changes differently from the critical paths. They advocate to monitor the registers for *setup violations*, and then provide mechanisms that recover from the resulting meta-stability and synchronization failures, and at the same time tune the supply voltage in order to keep the probability of error under control [1], [3].

Unfortunately the first approach is suitable only for CPUs and other chips that have a huge volume (to justify the development of at-speed testing and to sustain the availability of different performance/price bins) and can support variable performance requirements. This excludes most embedded systems, which typically need to satisfy real-time performance requirements, and which seldom enjoy a large enough market. Moreover, it only tackles process-related variations.

The third approach allows one to run a circuit very close to the power/performance limit, including even some data-dependent performance variations, but it requires *architectural support for flushing and restarting the data flow*, since whenever meta-stability occurs, the content of *all the registers in the design where an incorrect value may have propagated by the time the failure is reliably detected* (two-three clock cycles in order to keep the Mean Time Between Failure (MTBF) low enough) must be erased and restored from some “safe checkpoint” state, which is always ensured to be valid by never critically clocking it. Processors already have this capability, in order to restart from errors (overflows, page faults, . . .), but supporting such capabilities for other domains would require a huge design effort *and area cost*, which are generally considered unacceptable.

We are thus left with various forms of dynamic voltage scaling [11], which can recover a significant portion of the above mentioned margins, with an acceptable NRE and testing cost, without making too many assumptions on the flexibility and size of the market. However, the response time of the voltage regulators and of the power supply network are typically a few millions of clock cycles. This means that *short-term supply voltage variations must still be margined out*. Such variations occur, for example, when power management circuitry turns on a neighboring power island, or when a circuit changes mode of operation, significantly changing switching activity. They may also be due to external interference. These short-term variations require a completely different mechanism to be handled, one that allows *instantaneous clock frequency to briefly deviate from the target*. This can be achieved *without synchronization failures* only by using asynchronous communication between blocks.

Asynchronous circuits were traditionally considered to be difficult to design, requiring a lot of expertise akin to “black magic”. However, we will argue that this is no longer the case, and that asynchronous design can be performed using essentially synchronous design tools and flows, with fairly localized changes on the clock tree generation step, and no changes to the manufacturing and testing steps.

As we mentioned above, in this scenario the instantaneous operating frequency of the circuit may temporarily change due to a sudden drop of the supply voltage before the regulator has had the time to respond. Moreover, one may still want to adapt the target average frequency of some resources to the computational demands of the application. Both circumstances may lead to synchronization failures at the boundary of the asynchronous circuitry, where it meets other blocks (e.g. communication controllers, timers, etc.) which must run at a fixed frequency. However, these interfaces are generally few, and already implemented using metastability-safe FIFOs (with a latency cost of a few clock cycles) in modern SOCs.

Another advantage of asynchronous implementation of SOC blocks is that this allows one to use a fully asynchronous communication infrastructure (bus or NOC) with much lower latency than a traditional

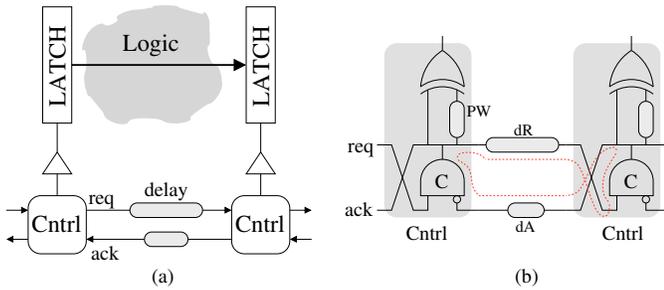


Fig. 1. Handshake controllers to implement Elastic Clocks.

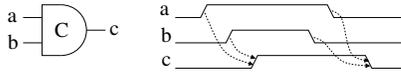


Fig. 2. C-element.

GALS approach. In a locally synchronous system where the clock is generated by a PLL, the latency of each domain crossing is at least 2-3 clock cycles, in order to allow meta-stability to resolve with high enough MTBF. Asynchronous blocks, on the other hand, where local clocks can be stopped in response to a meta-stability event, need to take that penalty only in those (very infrequent) clock cycles when the meta-stability actually occurs.

This paper is organized as follows. Section II presents the main paradigm of the paper: Elastic Clocks. The architecture, timing, design flow and performance are discussed. Sect. III analyzes the factors that enable a significant reduction of margins to cover variability. The tolerance to voltage noise and aging will be particularly studied. Finally, Sect. IV presents some techniques for voltage scaling that are suitable for Elastic Clocks and discusses an experimental example.

II. ELASTIC CLOCKS

Elastic Clocks [13] emerge from the combination of the asynchronous pipelines by Muller [7] and Sutherland [12] and a methodology to automatically transform synchronous circuits into asynchronous. This methodology is known as *Desynchronization* [2]. Elastic Clocks inherit the properties of asynchronous circuits while re-using the design flow of classical synchronous circuits.

The transformations required to introduce Elastic Clocks do not modify the logic and sequential elements of the original circuit. They substitute the global clock signal by a set of local clocks synchronized by a handshake protocol.

The basic principle of the operation of the Elastic Clocks is depicted in Fig. 1(a). The elastic controllers (*Cntrl*) generate the local clocks for the sequential elements and synchronize with other controllers through a pair of handshake signals (*req* and *ack*).

Various handshake protocols can be used to synchronize the controllers [2]. The C-element [7] was the basic component for the 4-phase protocol in Muller's pipeline. It was later used by Sutherland for the 2-phase handshake protocol of the *Micropipelines* [12].

A C-element is an asynchronous sequential element that acts as an event synchronizer. The behavior of a 2-input C-element is shown in Fig. 2. After receiving an event at each input, the C-element produces an event at the output. This behavior can be generalized for any number of inputs.

Figure 1(b) depicts one possible implementation of the handshake controllers using C-elements and a 2-phase protocol. It includes an XOR gate and a delay (PW) to generate the pulses for the local clocks that are connected to the latches. This structure is inspired by

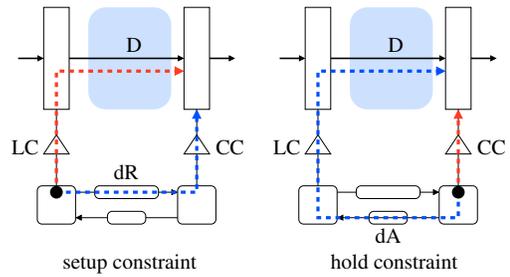


Fig. 3. Setup and hold constraints for Elastic Clocks.

the protocol of Micropipelines, but using pulse generators that enable the use of conventional latches or flip-flops.

A. Timing

The delays connected to the handshake signals are adjusted to meet various constraints that guarantee the correct timing behavior.

The dotted curve along the delays and the C-elements outlines an oscillator that synchronizes with the neighboring oscillators through the C-elements. The sum of the dR , dA and of the C-element delays determines the local cycle period for the associated logic in the circuit. The delay PW determines the pulse width generated at the local clocks.

The delays dR and dA are adjusted to meet the *setup* and *hold* constraints between the sequential elements triggered by different Elastic Clocks. Figure 3 shows the paths involved in the setup and hold timing constraints. D represents the delay of the logic of the circuit between the sequential elements, whereas LC and CC represent the delay of the launching and capturing local clocks, respectively. The symbolic delays dR and dA also account for the internal delays of the controllers. Both constraints involve two paths (short and long) with a common origin in the logic of the elastic controllers represented by the solid circle.

The timing constraints can be represented as follows:

$$\text{setup: } LC_{\max} + D_{\max} + \text{setup} < dR_{\min} + CC_{\min}$$

$$\text{hold: } CC_{\max} + \text{hold} < dA_{\min} + LC_{\min} + D_{\min}$$

where the min and max suffixes denote the minimum and maximum delays of the paths.

B. Design flow

Elastic circuits are generated by applying a set of simple transformations to an existing synchronous circuit. The main steps in the design flow are the following:

- Partition the system into a set of modules (clusters). Each cluster will preserve its local clock and internal behavior. Elasticity will be introduced between clusters.
- Insert latches at the boundaries of the clusters to hold the back-pressure generated by the communication between clusters.
- Substitute the global clock by a set of local clocks and controllers for each cluster.
- Synthesize the matched delays required to meet the timing constraints.

Figure 4 depicts a conventional circuit in which three clusters have been identified for elasticization. The rectangles inside the clusters represent sequential elements (latches or flip-flops), whereas the arrowed lines represent combinational paths. Figure 5 depicts the same circuit after the transformations. Note that in general the transformation is correct also for circuits with cyclic communication between clusters, not just for acyclic pipelines as in the figure [2].

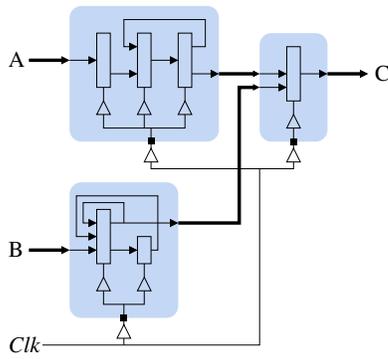


Fig. 4. Synchronous Circuit.

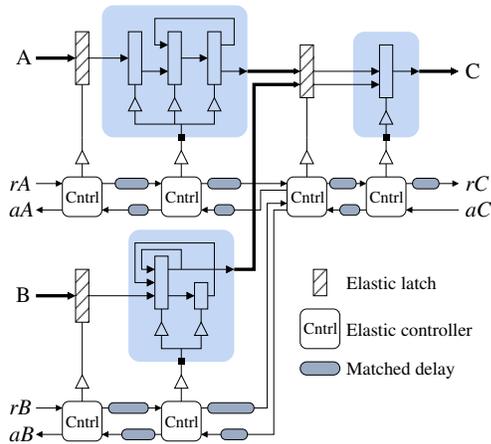


Fig. 5. Elastic Circuit.

In general, elasticization is done at a coarse level of granularity with clusters corresponding to large blocks in an SoC. Typical examples of clusters could be a microprocessor, a memory controller, an MPEG decoder, etc.

The original clusters have been preserved, including their local clocks. A latch is inserted in each cluster-to-cluster path, including the paths from inputs to clusters. These latches (called “elastic” in the following) are used to store the incoming data temporarily while the oscillators associated to the communicating clusters synchronize.

The global clock has been substituted by a set of local clocks generated by the elastic controllers. The timing of the clocks is determined by the matched delays that are defined to meet the internal timing constraints within each cluster and the setup/hold constraints between clusters. For the sake of efficiency, time borrowing can be used for the elastic latches.

After creating an elastic circuit, each input/output channel of the system is associated to a pair (req/ack) of handshake signals. For example, channel *A* in Fig. 5 is associated to the pair *rA/aA*.

C. Performance

The synchronization of the elastic controllers forces the elastic clocks to *work in unison*. In this way, the number of clock events generated in the system is the same at all local clocks. The oscillator with the longest $dR + dA$ delay is the one that determines the frequency of the system¹.

¹Tighter global cycles of delays involving multiple clusters may also exist. The impact of these cycles on the performance requires an analysis similar to that for synchronous systems with useful skew.

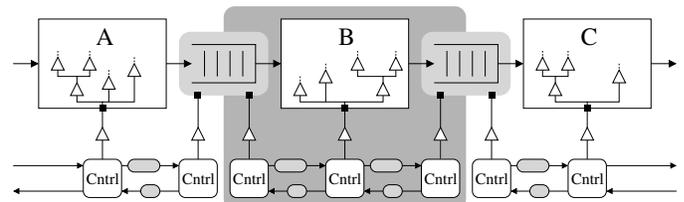


Fig. 6. Decoupled synchronization with asynchronous FIFOs.

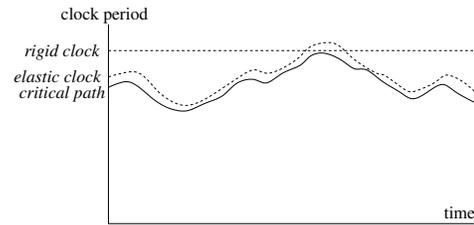


Fig. 7. Rigid and Elastic Clock periods.

The constraint of “working in unison” can also be relaxed. As in synchronous circuits with multiple clock domains, elastic circuits can also have clusters that are time-decoupled from the other clusters, as shown in Fig. 6 for cluster *B*. In these cases, some synchronization mechanism for clock-domain crossing (e.g. an asynchronous FIFO) is required to guarantee a metastability-free communication. This type of synchronization typically incurs a latency overhead of 2-3 cycles. However, one can also use the ability to stop elastic clocks while meta-stability is resolved, to implement safe synchronization with lower average latency [14].

III. TRACKING VARIABILITY WITH ELASTIC CLOCKS

Clock signals are often generated from crystal oscillators that provide a highly stable frequency reference to the PLLs. The locking schemes of the PLLs require a long response time when a change of frequency is required. Therefore, PLLs cannot provide a quick response to requirements for the variation of the clock period. For this reason, we refer to the PLL-based clocks as *rigid clocks*.

A major property of the Elastic Clocks is their immediate adaptability to the dynamic variations of the circuit. Their cycle-by-cycle adaptation allows the circuit to react to those fluctuations of the operating conditions that affect the delay of the critical paths.

A fundamental difference between elastic and rigid clocks lies in the way delay variability is accommodated by the clock period:

- With rigid clocks, dynamic variability must be handled by adding guard-band margins in the clock period that cover the full range of variability considering worst-case operating conditions of the circuit.
- With Elastic Clocks, dynamic variability can be handled by adjusting the clock period at every cycle and adding small margins to account for the differential variability between the critical paths and the clocks.

The correlation between the period of the Elastic Clocks and the delay of the critical paths is illustrated in Fig. 7. Rigid clocks always work at a constant frequency, thus wasting a significant amount of time in most of the cycles. Elastic Clocks follow the variability of the critical paths with much smaller margins, thus recovering a substantial part of the margins. Elastic Clocks do not work at a fixed frequency. However, *their average frequency is higher than that of rigid clocks*.

There are different sources of variability that must be considered when defining the maximum operating frequency of a circuit [5].

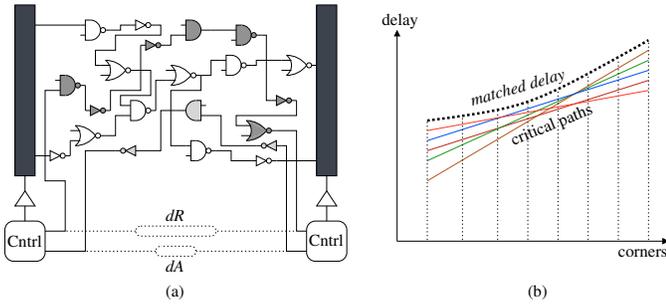


Fig. 8. Matched delays: (a) implementation embedded in the logic; (b) synthesis of multi-corner delays.

We can classify them into two groups: static (process) and dynamic (environmental). The margins for static variability can be partially recovered by doing a post-silicon characterization (e.g., speed binning).

Techniques like Dynamic Voltage and Frequency Scaling (DVFS) and Adaptive Voltage Scaling (AVS) can recover some of the margins for low-frequency dynamic variability. This is mostly relegated to temperature variations since the response time of voltage regulators is still sufficient to compensate the slowly changing thermal behavior of the chip.

A. Correlated matched delays

A key aspect of the Elastic Clocks is the adaptability to variability of the matched delays. They are implemented using cells from the same library and embedded in the circuit. Figure 8(a) shows how the matched delays (shadowed) live together with the logic of the circuit.

The proximity of the matched delays increases the spatio-temporal correlation with the variability of the logic. Besides the systematic process variability, the matched delays are also capable of tracking high-frequency dynamic variability such as the one associated to power supply noise.

The matched delays must be designed in such a way that they cover all the potential variability in all possible operational corners. This is illustrated in Fig. 8(b). A diversity of paths may become critical depending on their sensitivity to variations and, therefore, the matched delays must guarantee that the worst-case delay is always covered at any corner. This is achieved by selecting an appropriate mixture of gates and wires that minimizes the mismatch of delays and constraining them to be slower than the critical paths at all significant corners.

B. Power supply noise

Even with the most sophisticated techniques for binning and voltage scaling, significant guard-band margins are required to cover high-frequency variability. This is the case of power supply noise which has various sources [10]: Voltage Regulator, PCB, package, on-chip IR drop and switching noise.

A significant part of the power supply fluctuations have a uniform impact on all components of the circuit [6]. Most of the work done to analyze power supply noise has been focused on estimating the worst-case impact on delay variation. However, this is not the most relevant aspect for Elastic Clocks. The important parameter is the *differential* variability between the logic and the matched delays.

The proximity of the matched delays and the fact that they are using the same power rails minimizes the differential variability. An excellent example is found in [9] where the absolute and differential voltage noise was analyzed in a 2.53GHz microprocessor with a

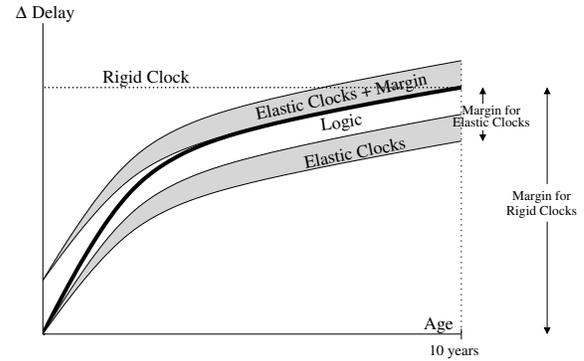


Fig. 9. Differential aging between the logic and the Elastic Clocks.

1.5V nominal voltage. While a peak-to-peak 63mV voltage drop was measured, the differential voltage noise between the clock paths was only 19mV. These voltage variations were estimated to have an impact of 5.1% and 0.8%, respectively. The reduced differential noise is a clear evidence of how well the matched delays can track the voltage variations cycle by cycle and the significant margin reductions that can be applied.

C. Aging

Phenomena like *Negative-bias-temperature-instability* (NBTI) and *Hot Carrier Injection* (HCI) have an increasing impact in the delay degradation (aging) of CMOS circuits.

While aging affects the combinational and sequential components of the circuit [4], it does not affect the frequency of the rigid clocks. For this reason, guard-band margins need to be included in the clock period to account for this degradation along the lifetime of the circuit.

The scenario with Elastic Clocks is different, since the matched delays also suffer the aging effect. In general, the matched delays may suffer less aging due to NBTI. However, they suffer more aging due to HCI, which highly depends on the switching activity. The result is that the aging of the logic is partially compensated by the aging of the matched delays, as shown in Fig. 9.

The thick line in the plot represents the worst estimated aging of the logic. The shadowed stripe labeled “Elastic Clocks” represents the estimated aging of the matched delays. The width of the stripe represents the interval of variability that aging can manifest. Even though aging has a certain degree of randomness, the aging of the matched delays can be accurately modeled, since the signal probability of the gates (0.5) and the transition density (1 switch per cycle) are exactly known.

The margins required for Elastic Clocks are significantly different and must be calculated according to the minimum guaranteed aging of the matched delays. As shown in Fig. 9, the reduced margins enable the circuit to run at better performance during infancy and youth, while adding a certain degradation during the old age. When using voltage scaling techniques, this degradation can be compensated with a slightly higher supply voltage and power consumption. However, the Elastic Clocks offer a significantly better power consumption when averaged over the lifetime of the circuit.

IV. POWER MANAGEMENT WITH ELASTIC CLOCKS

Elastic Clocks offer new opportunities for power management that can be implemented more efficiently than with rigid clocks. A good example of these techniques was presented in [8] in which the supply voltage of a data processing circuit was scaled according to the occupancy of the input FIFO buffer. In this way, the minimum supply voltage to guarantee a certain performance was obtained.

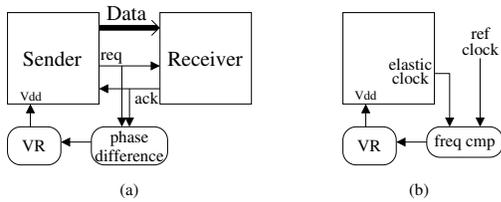


Fig. 10. Schemes for automatic voltage scaling.

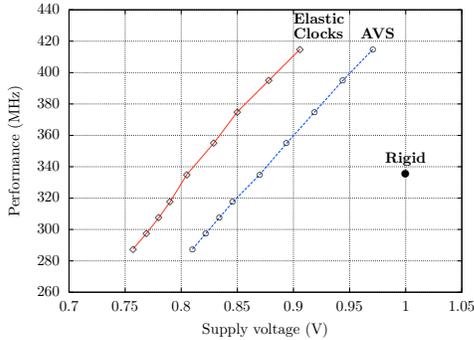


Fig. 11. Voltage scaling with AVS and Elastic Clocks.

Figure 10 depicts two possible schemes for automatic voltage scaling that can be used with Elastic Clocks. Both of them include some logic that controls a voltage regulator (VR). In the first scheme (Fig. 10(a)), the control logic compares the relative arrival of the *req/ack* signals in a communication channel between a sender and a receiver. Depending on who arrives first, the voltage is regulated so that the speed of both modules is similar, i.e. if the sender is “too fast” the voltage is lowered and if it is “too slow” the voltage is raised.

The second scheme (Fig. 10(b)) compares the speed of an elastic circuit with the frequency of a reference clock. In this scheme, a frequency comparator determines the relative speed of both clocks and raises/lowers the supply voltage in such a way that the frequency of the Elastic Clock approaches the reference frequency.

A. Validation in silicon

The efficiency of the Elastic Clocks has been validated in silicon with the implementation of an ARM926 microprocessor in a 65nm technology. For an apple-to-apple comparison, three cores were included on the same die: one with a conventional implementation with rigid clocks, one with an AVS scheme for voltage scaling and one with Elastic Clocks and a voltage scaling scheme similar to the one in Fig. 10(b).

A summary of the measurements is shown in Fig. 11. The original design was specified to work at 1V (nominal voltage) and 335MHz in worst-case conditions. The plot shows the results for one of the *typical* dies. To evaluate the impact of voltage noise, an AC component of $\pm 10\%$ was injected on the supply voltage. The effect of aging was not evaluated in this experiment.

The AVS scheme was able to reduce voltage down to 0.87V, thus achieving a power reduction of 23% at the same frequency. The Elastic Clocks were able to reduce voltage down to 0.8V with a power savings of 35%. Most of the benefits from the Elastic Clocks were obtained due to the tolerance to the voltage noise.

The plot also shows the explored range of voltages and frequencies for AVS and the Elastic Clocks. It is important to emphasize that the frequency adjustments for AVS must be done explicitly with an external control and in the appropriate order (e.g. lowering frequency

before lowering the voltage). On the other hand, the frequency adjustment with Elastic Clocks is fully automatic and does not require any external intervention.

V. CONCLUSIONS

This paper provided quantitative justifications to the thesis that some further power and performance margins can be recovered by using an asynchronous implementation style based on the notion of Elastic Clocks. The paper focused mostly on power savings achieved by careful adaptive supply voltage tuning *while tolerating short-term deviations from the average target frequency*. However, similar improvements, for applications that can exploit all the locally available computation power (e.g. large arrays of processors, such as GPUs, network of sensors where computation and sensing tasks can be dynamically distributed, ...) can be achieved in terms of *performance and area*. Moreover, asynchronous blocks are better suited for the asynchronous interconnect structures which reduce the number of clock domain crossings in a complex SOC, because they dramatically reduce the communication latency, while still avoiding synchronization failures altogether.

Acknowledgments. This work has been partially supported by the project CICYT TIN2007-66523, FPI grant BES-2008-004039 and AVANZA project TSI-020302-2009-20 from the Spanish Government, and the MODERN project (ENIAC-120003). We would like to thank Sachin Sapatnekar and Alex Yakovlev for their valuable support.

REFERENCES

- [1] K. Bowman, J. Tschanz, N. S. Kim, J. Lee, C. Wilkerson, S.-L. Lu, T. Karnik, and V. De, “Energy-efficient and metastability-immune resilient circuits for dynamic variation tolerance,” *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 49–63, Jan. 2009.
- [2] 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, Oct. 2006.
- [3] 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*, vol. 24, no. 6, pp. 10–20, 2004.
- [4] D. Lorenz, G. Gerogakos, and U. Schlichtmann, “Aging analysis of circuit timing considering NBTI and HCI,” in *15th IEEE Int. On-Line Testing Symp. (IOLTS)*, Jun. 2009, pp. 3–8.
- [5] P. McGuinness, “Variations, margins and statistics,” in *Int. Symp. on Physical Design (ISPD)*, Apr. 2008, pp. 60–67.
- [6] A. Muhtaroglu, G. Taylor, and T. Rahal-Arabi, “On-die droop detector for analog sensing of power supply noise,” *IEEE Journal of Solid-State Circuits*, vol. 39, no. 4, pp. 651–660, Apr. 2004.
- [7] D. E. Muller, “Asynchronous logics and application to information processing,” in *Symposium on the Application of Switching Theory to Space Technology*. Stanford University Press, 1962, pp. 289–297.
- [8] L. Nielsen, C. Niessen, J. Sparsø, and K. van Berkel, “Low-power operation using self-timed circuits and adaptive scaling of the supply voltage,” *IEEE Transactions on VLSI Systems*, vol. 2, no. 4, pp. 391–397, Dec. 1994.
- [9] M. Saint-Laurent and M. Swaminathan, “Impact of power-supply noise on timing in high-frequency microprocessors,” *IEEE Transactions on Advanced Packaging*, vol. 27, no. 1, pp. 135–144, Feb. 2004.
- [10] R. Schmitt and C. Yuan, “Power distribution design considerations and methodology for multi-gigabit I/Os,” in *Int. Symp. on Electromagnetic Compatibility (EMC)*, vol. 2, Aug. 2005, pp. 660–665.
- [11] N. Semiconductors, “Powerwise adaptive voltage scaling,” http://www.national.com/analog/powerwise/avs_overview.
- [12] I. E. Sutherland, “Micropipelines,” *Communications of the ACM*, vol. 32, no. 6, pp. 720–738, Jun. 1989.
- [13] E. Tuncer, J. Cortadella, and L. Lavagno, “Enabling adaptability through elastic clocks,” in *Proc. ACM/IEEE Design Automation Conference*, Jul. 2009, pp. 8–10.
- [14] A. Yakovlev, D. Kinniment, F. Xia, and A. Koelmans, “A FIFO buffer with non-blocking interface,” *IEEE Computer Society TC/VLSI Technical Bulletin*, pp. 11–14, Fall 1998.